UNIVERSIDAD LA SALLE

LICENCIATURA EN INGENIERÍA DE SISTEMAS



“SISTEMA INTEGRADO DE GESTIÓN DE DOCUMENTOS Y EMISIÓN DE FICHAS EMPLEANDO KUBERNETES PARA ORQUESTACIÓN DE CONTENEDORES PARA EL SEGURO MÉDICO DELEGADO DE COTEL”

MODALIDAD DE TITULACIÓN: PROYECTO DE GRADO

POSTULANTE: ALBERT OMAR ARAYA CORY

TUTOR: MSC. ESP. ING. OSAMU MANUEL YOKOSAKI P.

LA PAZ-BOLIVIA

2025

DEDICATORIA

Este trabajo está dedicado a:

A Dios, por ser mi guía constante, darme fortaleza en los momentos difíciles y sabiduría para continuar cada día con fe y esperanza.

A mis abuelos, que, aunque ya no están conmigo, me brindaron su cariño, enseñanzas y muchas de las herramientas que hoy aplico en mi vida profesional.

A mis padres, por ser el pilar fundamental de mi vida, por su apoyo, sacrificio y ejemplo de perseverancia que me motivan a seguir adelante.

A mis hermanos, por su compañía, comprensión y por ser una fuente constante de alegría e inspiración en cada etapa de mi camino.

AGRADECIMIENTOS

A Dios, por iluminar mi camino, fortalecer mi espíritu y brindarme la sabiduría necesaria para culminar esta etapa tan importante de mi vida.
A mis padres, por su esfuerzo constante, sacrificio y por ser el ejemplo que inspira mi superación diaria. A mis hermanos, por su cariño, alegría y motivación en cada momento de este proceso.

A mi tutor Msc. Esp. Ing. Osamu Manuel Yokosaki P., por su orientación, paciencia y dedicación durante todo mi proceso universitario, así como por compartir su conocimiento y experiencia, los cuales fueron fundamentales para la culminación de este proyecto.

A todos mis docentes, quienes con su enseñanza, exigencia y compromiso contribuyeron de manera significativa a mi formación profesional y personal.

Finalmente, a todos aquellos que, de una u otra manera, formaron parte de este camino, brindándome su apoyo, consejo o palabra de aliento, mi más profundo agradecimiento.

ÍNDICE

ÍNDICE DE CONTENIDOS

1 GENERALIDADES 1

1.1 INTRODUCCIÓN 1

1.2 ANTECEDENTES 1

1.2.1 Antecedentes institucionales 1

1.2.2 Antecedentes académicos 2

1.3 PLANTEAMIENTO DEL PROBLEMA 4

1.3.1 Identificación del problema 4

1.3.2 Formulación del problema 5

1.4 OBJETIVOS 6

1.4.1 Objetivo general 6

1.4.2 Objetivos específicos 6

1.5 JUSTIFICACIÓN 7

1.5.1 Justificación técnica 7

1.5.2 Justificación económica 7

1.5.3 Justificación social 8

1.6 ALCANCES 9

1.6.1 Alcance temático 9

1.6.2 Alcance geográfico 9

1.6.3 Alcance temporal 9

1.7 LIMITES 10

2 MARCO TEÓRICO 11

2.1 INTRODUCCIÓN 11

2.1.1 Sistemas de salud 11

2.1.1.1 Tipos de sistemas 11

2.1.2 Seguro médico 12

2.1.3 Seguros médicos en Bolivia 12

2.1.3.1 Situación actual 12

2.1.3.2 Seguros médicos delegados 13
2.1.3.3 Afiliación de personas 13
2.1.3.4 Desafiliación de personas 20
2.1.3.5 Procedimiento de desafiliación 21
2.1.4 Áreas involucradas en la atención médica en seguros sociales 23
2.1.4.1 Vigencia de derechos 23
2.1.4.2 Área de fichaje 23
2.1.4.3 Área de enfermería 23
2.1.4.4 Área médica 24
2.2 INGENIERÍA DE SOFTWARE 24
2.2.1 Software 24
2.2.2 Metodologías de desarrollo de software 25
2.2.3 Metodologías agiles 26
2.2.4 Metodología scrum 26
2.2.4.1 Principios de Scrum para el éxito de los proyectos 27
2.2.4.2 Valores de Scrum para los equipos de trabajo de los proyectos 27
2.2.4.3 Artefactos de Scrum 28
2.2.4.4 Scrum Team 29
2.2.4.5 Eventos de Scrum 32
2.2.5 Método MoSCoW 35
2.2.5.1 Características 36
2.2.5.2 Proceso de ejecución 36
2.2.6 Metodología UML-based Web Engineering 38
2.2.6.1 Actividades de modelado en UWE 39
2.2.6.2 Fases UWE 40
2.2.6.3 Ventajas de UWE 41
2.2.7 Arquitectura de software 42
2.2.7.1 Descomposición modular 43

2.2.7.2 Cliente-servidor 47
2.2.8 Seguridad de software 50
2.2.9 Pruebas de software 50
2.2.9.1 Tipos de pruebas 50
2.2.9.2 Técnicas de pruebas 52
2.2.9.3 Pruebas de caja negra 52
2.3 ESTIMACIÓN DE COSTOS DE SOFTWARE 55
2.3.1 Modelo COCOMO II 56
2.3.1.1 Estimación de esfuerzo 57
2.3.1.2 Estimación del Cronograma 61
2.3.1.3 Métricas de Software 61
2.4 HERRAMIENTAS 68
2.4.1 Diagramas 68
2.4.1.1 Diagramas de flujo 68
2.4.1.2 Diagramas de procesos 69
2.4.1.3 Diagrama entidad relación 71
2.4.2 Figma 73
2.4.3 Git 74
2.4.3.1 Aspectos básicos de Git 74
2.4.3.2 Ramas 75
2.4.3.3 Archivos y confirmaciones 75
2.4.4 Github 76
2.4.5 GitFlow 77
2.4.6 Notion 78
2.4.7 React 79
2.4.8 Electron 79
2.4.9 Laravel 80
2.4.10 JWT 81

2.4.10.1 Estructura de los JWT 82

2.4.11 Postman 86

2.4.11.1 Características de Postman 86

2.4.12 Kong 87

2.4.13 MySql 88

2.4.13.1 Ventajas 88

2.4.14 SocketIO 90

2.4.15 Docker 90

2.4.16 Kubernetes 91

2.4.16.1 Uso de Kubernetes 92

2.4.17 Digital Ocean 94

2.4.18 GitHub Actions 95

2.4.18.1 Componentes de GitHub Actions 95

3 MARCO APLICATIVO 99

3.1 INTRODUCCIÓN 99

3.2 PRODUCT BACKLOG 99

3.2.1 Visión del producto 99

3.2.1.1 Criterios de definición de funcionalidades clave 101

3.2.1.2 Relación entre la visión del producto y la estrategia del Product Backlog 103

3.2.2 Identificación de requerimientos iniciales 103

3.2.2.1 Proceso para identificar las necesidades y expectativas del cliente y stakeholders 103

3.2.2.2 Metodologías usadas para recopilar los requisitos 104

3.2.3 Refinamiento de requisitos 104

3.2.3.1 Clasificación y elaboración de las tablas de requerimientos 105

3.2.4 Historias de usuario 108

3.2.4.1 Estructura 108

3.2.4.2 Elaboración 109

3.2.5 Priorización 109
3.2.5.1 Criterios de priorización para historias de usuario 109
3.2.6 Estimación de esfuerzos 111
3.2.7 Estimación de tiempo 114
3.2.8 Estimación de personal necesario 116
3.2.9 Estimación del costo del sistema 117
3.2.10 Definición de esfuerzos por historias 117
3.2.11 Definición de criterios de aceptación 119
3.2.11.1 Criterios de aceptación por historia de usuario 119
3.2.12 Gestión continua 120
3.2.12.1 Estrategia para mantener y actualizar el Product Backlog a lo largo del proyecto 120
3.2.12.2 Proceso para agregar nuevas historias y re-priorizar según el progreso o nuevos requisitos 121
3.2.12.3 Proceso de cambios o ajustes en las prioridades del negocio 121
3.2.13 Herramientas 121
3.3 PLANIFICACIÓN 122
3.3.1 Priorización de los entregables 122
3.3.2 Capacidad de desarrollo 122
3.3.3 Estimación de esfuerzo 123
3.3.4 Criterios de priorización de tareas 123
3.3.5 Duración del sprint 124
3.4 DESARROLLO DEL PROYECTO 127
3.4.1 SP1: Diseño y desarrollo de la arquitectura 128
3.4.1.1 Sprint Planning 128
3.4.1.2 Sprint Review 129
3.4.1.3 Sprint Retrospective 129
3.4.2 SP2: Implementar el registro de datos generales para titulares y beneficiarios en el sistema. 134

3.4.2.1 Sprint Planning.....134

3.4.2.2 Sprint Review 136

3.4.2.3 Sprint Retrospective 137

3.4.3 SP3: Implementar la funcionalidad de registro para gestionar y almacenar fichas emitidas de manera eficiente.....141

3.4.3.1 Sprint Planning.....141

3.4.3.2 Sprint Review 143

3.4.3.3 Sprint Retrospective 145

3.4.4 SP4: Implementar el manejo de beneficiarios y gestionar el registro de altas y bajas de asegurados. 152

3.4.4.1 Sprint Planning.....152

3.4.4.2 Sprint Review 154

3.4.4.3 Sprint Retrospective 156

3.4.5 SP5: Implementar el escaneo y almacenamiento seguro de los documentos de los asegurados 162

3.4.5.1 Sprint Planning.....162

3.4.5.2 Sprint Review 164

3.4.5.3 Sprint Retrospective 166

3.4.6 SP6: Configurar en el sistema los documentos correspondientes para cada tipo de beneficiario.....171

3.4.6.1 Sprint Planning.....171

3.4.6.2 Sprint Review 173

3.4.6.3 Sprint Retrospective 174

3.4.7 SP7: Modificar, almacenar y actualizar la documentación periódica, asegurando la inclusión de nuevos documentos requeridos 179

3.4.7.1 Sprint Planning.....179

3.4.7.2 Sprint Review 181

3.4.7.3 Sprint Retrospective 182

3.4.8 SP8: Implementar una funcionalidad de registro de usuarios seguro y eficiente. 186

3.4.8.1 Sprint Planning.....186

3.4.8.2 Sprint Review 188

3.4.8.3 Sprint Retrospective 190

3.4.9 SP9: Implementar una funcionalidad de control de acceso basado en niveles de jerarquía de usuario 194

3.4.9.1 Sprint Planning.....194

3.4.9.2 Sprint Review 196

3.4.9.3 Sprint Retrospective 197

3.4.10 SP10: Implementar una funcionalidad para registrar solicitudes y gestionar la entrega de carnets de manera eficiente 201

3.4.10.1 Sprint Planning.....201

3.4.10.2 Sprint Review 203

3.4.10.3 Sprint Retrospective 204

3.4.11 SP11: Implementar la funcionalidad para registrar y gestionar los cambios en los horarios de atención de los médicos de manera eficiente. 209

3.4.11.1 Sprint Planning.....209

3.4.11.2 Sprint Review 211

3.4.11.3 Sprint Retrospective 212

3.4.12 SP12: Implementar la visualización de turnos en TV según la ficha correspondiente.....216

3.4.12.1 Sprint Planning.....216

3.4.12.2 Sprint Review 217

3.4.12.3 Sprint Retrospective 218

3.4.13 SP13: Desarrollar una plataforma funcional que permita a los usuarios agendar visitas en línea de manera sencilla 222

3.4.13.1 Sprint Planning.....222

3.4.13.2 Sprint Review 224

3.4.13.3 Sprint Retrospective 225

3.4.14 SP14: Implementar una función de detección y visualización de alertas para documentos caducados 230

3.4.14.1 Sprint Planning....230
3.4.14.2 Sprint Review....232
3.4.14.3 Sprint Retrospective....234
3.4.15 SP15: Implementar la generación de informes en el formato establecido y permitir la exportación de datos a Excel....237
3.4.15.1 Sprint Planning....237
3.4.15.2 Sprint Review....238
3.4.15.3 Sprint Retrospective....239
3.5 ANALISIS DE COSTOS....244
3.5.1 Determina el tamaño del sistema....244
3.5.2 Selecciona el nivel de COCOMO II....244
3.5.3 Identifica los factores de costo (Cost Drivers)....245
3.5.4 Cálculo del esfuerzo en Persona-Meses....247
3.5.5 Determina el tiempo de desarrollo....247
3.5.6 Calcula el costo total....248
4 CONCLUCIONES Y RECOMENDACIONES....249
4.1 Conclusiones....249
4.2 Recomendaciones....251
4.2.1 Planificación e investigación detallada de Kong Gateway....251
4.2.2 Continuidad en el uso de flujos de CI/CD....251
4.2.3 Infraestructura de servidores propios....251

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Descomposición modular. 47
Ilustración 2: Cliente - Servidor. 48
Ilustración 3: Checklist para la definición de una línea de código fuente. 67
Ilustración 4: Simbología de un diagrama de actividades 70
Ilustración 5: Simbología de diagrama entidad relación. 73
Ilustración 6: Encabezado JWT 82
Ilustración 7: Carga útil de JWT 83
Ilustración 8: Fimar de un JWT 84
Ilustración 9: JWT cifrado 84
Ilustración 10: Ejemplo de encabezado JWT 85
Ilustración 11: Flujo de uso de JWT 85
Ilustración 12: Funcionamiento de Docker. 91
Ilustración 13: Visión antigua y moderna del despliegue de dockers, ... 93
Ilustración 14: Componentes GitHub Actions. 96
Ilustración 15: Ejemplo de formato de historia de usuario general. 109
Ilustración 16: Ejemplo de formato de historia de usuario de un trabajador. 109
Ilustración 17: Diseño de la arquitectura para desarrollo. 130
Ilustración 18: Diseño de la arquitectura para producción. 130
Ilustración 19: Diseño del flujo CI / CD. 131
Ilustración 20:: Diagrama de procesos afiliación de funcionarios 132
Ilustración 21: Diagrama de procesos afiliación de funcionarios 133
Ilustración 22: Diseño de la base de datos de beneficiarios. 138
Ilustración 23: Lista para la visualización de los titulares o beneficiarios. 138
Ilustración 24: Formulario para agregar a un titular o beneficiario. 139
Ilustración 25: Formulario para eliminar a un titular o beneficiario. 139
Ilustración 26: Formulario para editar a un titular o beneficiario. 140
Ilustración 27: Diseño de la base de datos de fichaje. 146
Ilustración 28: Diseño de la base de datos de doctores. 147
Ilustración 29: Formulario para agregar un nuevo doctor. 147
Ilustración 30: Tabla para visualizar a los doctores registrados. 148
Ilustración 31: Modal para agregar un nuevo doctor externo. 148
Ilustración 32: Formulario de eliminación de doctores. 149
Ilustración 33: Ventana de búsqueda por carnets. 149
Ilustración 34: Ventana de asignación de especialidad. 150

Ilustración 35: Ventana de asignación de doctores. 150
Ilustración 36: Ventana de asignación de horarios. 151
Ilustración 37: Diseño de la base de datos de beneficiarios con las tablas de clasificación. 157
Ilustración 38: Diseño de la base de datos de beneficiarios con las tablas de altas y bajas. 158
Ilustración 39: Formulario de edición con la opción de cambio de tipo de beneficiario. 159
Ilustración 40: Formulario de habilitación de beneficiarios. 160
Ilustración 41: Formulario de baja de beneficiarios. 160
Ilustración 42: Tabla de visualización de los beneficiarios dados de baja. 161
Ilustración 43: Modal de cancelación de bajas o habilitaciones. 161
Ilustración 44: Diseño de la base de datos de beneficiarios con las tablas de almacenamiento de documentos. 167
Ilustración 45: Modal de subida de archivos. 167
Ilustración 46: Modal de ejemplo de titulares para elegir el documento a subir según el tipo de beneficiarios. 168
Ilustración 47: Modal de ejemplo con el archivo en el servidor. 169
Ilustración 48: Modal de visualización de documentos. 170
Ilustración 49: Diseño de la base de datos de beneficiarios con las tablas de configuración y asignación de documentos. 175
Ilustración 50: Tabla de configuración y listado de documentos. 176
Ilustración 51: Tabla de configuración y listado de documentos. 176
Ilustración 52: Tabla de configuración y listado de documentos. 177
Ilustración 53: Tabla de asignación de documentos según el tipo de beneficiario. 177
Ilustración 54: Modal de asignación de documentos según el beneficiario. 178
Ilustración 55: Modal de carpetas con archivos en el servidor para almacenamiento periódico. 183
Ilustración 56: Modal de actualización de documentos después de seleccionar el botón amarillo. 184
Ilustración 57: Modal de verificación de documentos guardados por fecha de registro. 184
Ilustración 58: Modal de verificación de documentos guardados por fecha de registro. 185
Ilustración 59: Diseño de la base de datos de usuarios. 191

Ilustración 60: Tabla de usuarios registrados en el sistema con sus respectivas opciones. 192
Ilustración 61: Modal para el registro de nuevos usuarios al sistema. 192
Ilustración 62: Modal para la edición de usuarios. 193
Ilustración 63: Modal de confirmación para la eliminación de usuarios. 193
Ilustración 64: Diseño de la base de datos de usuarios con las tablas de control de acceso jerárquico. 198
Ilustración 65: Modal para la asignación de acceso al sistema. 199
Ilustración 66: Modal para la asignación de acceso al sistema más la asignación de la sucursal. 199
Ilustración 67: Ventana de acceso al sistema 200
Ilustración 68: Diseño de la base de datos de beneficiarios con las tablas para solicitud y entrega de carnets. 205
Ilustración 69: Buscador para la verificación de credenciales. 206
Ilustración 70: Modal de confirmación para la creación de una solicitud de credencial. 206
Ilustración 71: Tabla de consulta de solicitudes de credenciales. 207
Ilustración 72: Modal para el registro de la impresión de credenciales. 207
Ilustración 73: Modal para el registro de la entrega de credenciales. 208
Ilustración 74: Tabla que lista todas las credenciales más su estado actual. 208
Ilustración 75: Diseño de tablas para la asignación de fichas y horarios según él día. 213
Ilustración 76: Modal de registro de horarios según sucursal y día. 214
Ilustración 77: Modal de registro de fichas para emitir según sucursal y día. 215
Ilustración 78: Tabla para la administración de Videos para la TV. 219
Ilustración 79: Modal para el registro de videos para la TV. 220
Ilustración 80: Modal para eliminar videos de TV. 220
Ilustración 81: Modal para editar un video de TV. 221
Ilustración 82: Pantalla de TV para visualización de fichas médicas. 221
Ilustración 83: Ingreso al sitio web para agnación de fichas para los beneficiarios. 226
Ilustración 84: Dashboard de los beneficiarios para agendar una ficha. 226
Ilustración 85: Vista para elegir la sucursal donde se registrará la ficha. 227
Ilustración 86: Ventana de asignación de especialidad. 227
Ilustración 87: Ventana de asignación de doctores. 228
Ilustración 88: Ventana de asignación de horarios. 228

Ilustración 89: Diagrama de procesos para generar una ficha como paciente 229
Ilustración 90: Diagrama de procesos para generar una ficha como paciente 229
Ilustración 91: Tabla de beneficiarios habilitados antes de la ejecución del job. ... 235
Ilustración 92: Modal después de la ejecución del job. 235
Ilustración 93: Tabla de beneficiarios después de la ejecución del job. 236
Ilustración 94: Tabla de alertas generadas después de la ejecución del job. 236
Ilustración 95: Ejemplo de reporte individual de beneficiario para vigencia de derechos. 240
Ilustración 96: Ejemplo de vista para obtener reportes para vigencia de derechos. 241
Ilustración 97: Ejemplo de reporte de falta de documentos de beneficiarios para vigencia de derechos. 241
Ilustración 98: Visualización para obtener el reporte personal de fichas para el área de fichaje. 242
Ilustración 99: Visualización para obtener el reporte de fichas por dr para el área de fichaje. 242
Ilustración 100: Visualización para obtener el reporte de fichas por especialidad para el área de fichaje. 243

ÍNDICE DE TABLAS

Tabla 1: Tabla de causas y efectos **..... 4**
Tabla 2: Condiciones y Acciones **..... 54**
Tabla 3: Modelo de Productividad **..... 58**
Tabla 4: Factores de Escala en COCOMO II. **..... 60**
Tabla 5: Esquema de Clasificación de Puntos Objetos. **..... 63**
Tabla 6: Peso de un Punto Objeto. **..... 63**
Tabla 7: Puntos Función. Determinación del Peso. **..... 65**
Tabla 8: Peso del Factor de Complejidad. **..... 65**
Tabla 9: Símbolos de diagramas de flujo **..... 68**
Tabla 10: Requisitos funcionales generales. **..... 106**
Tabla 11: Requisitos funcionales de vigencia de derechos. **..... 106**
Tabla 12: Requisitos funcionales de fichaje. **..... 107**
Tabla 13: Requisitos funcionales de sistemas. **..... 107**
Tabla 14: Requisitos funcionales de gerencia. **..... 107**
Tabla 15: Requisitos no funcionales. **..... 108**
Tabla 16: Requerimientos según prioridad. **.....110**
Tabla 17: Tabla de ejemplo de cálculo de LOC para una historia de usuario. **.....111**
Tabla 18: Historias de usuario con su respectivo SLOC. **.....112**
Tabla 19: Tabla guia factores de escala de costo 115
Tabla 20: Tabla factores de escala propuestos.....116
Tabla 21: Historias de usuario con puntos de esfuerzo. **.....118**
Tabla 22: Ejemplo de historia de usuario con sus respectivos criterios de aceptación.
..... 120
Tabla 23: Planificación de cada Sprint. **..... 124**
Tabla 24: Sprint Backlog SP1. 127
Tabla 25: Tabla resumen del sprint review SP1. 128
Tabla 26: Tabla resumen del sprint retrospective SP1. 129
Tabla 27: Sprint Backlog SP2. 134
Tabla 28: Tabla resumen del sprint review SP2. 136
Tabla 29: Tabla resumen del sprint retrospective SP2. 137
Tabla 30: Sprint Backlog SP3. 141
Tabla 31: Tabla resumen del sprint review SP3. 143
Tabla 32: Tabla resumen del sprint retrospective SP3. 145
Tabla 33: Sprint Backlog SP4. 152

Tabla 34: Tabla resumen del sprint review SP4. 154
Tabla 35: Tabla resumen del sprint retrospective SP4. 156
Tabla 36: Sprint Backlog SP5. 162
Tabla 37: Tabla resumen del sprint review SP5. 164
Tabla 38: Tabla resumen del sprint retrospective SP5. 166
Tabla 39: Sprint Backlog SP6. 171
Tabla 40: Tabla resumen del sprint review SP6. 173
Tabla 41: Tabla resumen del sprint retrospective SP6. 174
Tabla 42: Sprint Backlog SP7. 179
Tabla 43: Tabla resumen del sprint review SP7. 181
Tabla 44: Tabla resumen del sprint retrospective SP7. 182
Tabla 45: Sprint Backlog SP8. 186
Tabla 46: Tabla resumen del sprint review SP8. 188
Tabla 47: Tabla resumen del sprint retrospective SP8. 190
Tabla 48: Sprint Backlog SP9. 194
Tabla 49: Tabla resumen del sprint review SP9. 196
Tabla 50: Tabla resumen del sprint retrospective SP9. 197
Tabla 51: Sprint Backlog SP10. 201
Tabla 52: Tabla resumen del sprint review SP10. 203
Tabla 53: Tabla resumen del sprint retrospective SP10. 204
Tabla 54: Sprint Backlog SP11. 209
Tabla 55: Tabla resumen del sprint review SP11. 211
Tabla 56: Tabla resumen del sprint retrospective SP11. 212
Tabla 57: Sprint Backlog SP12. 216
Tabla 58: Tabla resumen del sprint review SP12. 217
Tabla 59: Tabla resumen del sprint retrospective SP12. 219
Tabla 60: Sprint Backlog SP13. 222
Tabla 61: Tabla resumen del sprint review SP13. 224
Tabla 62: Tabla resumen del sprint retrospective SP13. 225
Tabla 63: Sprint Backlog SP14. 231
Tabla 64: Tabla resumen del sprint review SP14. 232
Tabla 65: Tabla resumen del sprint retrospective SP14. 234
Tabla 66: Sprint Backlog SP15. 237
Tabla 67: Tabla resumen del sprint review SP15. 238
Tabla 68: Tabla resumen del sprint retrospective SP15. 239
Tabla 69: Tabla con todas las estimaciones slog. 244

Tabla 70: Tabla factores de costo del sprint. 245
Tabla 71: Tabla factores de costo 246

ÍNDICE DE ECUACIONES

Ecuación 1 **..... 57**
Ecuación 2 **..... 57**
Ecuación 3 **..... 58**
Ecuación 4 **..... 58**
Ecuación 5 **..... 58**
Ecuación 6 **..... 60**
Ecuación 7 **..... 61**
Ecuación 8 **..... 64**
Ecuación 9 **..... 64**
Ecuación 10 **..... 65**
Ecuación 11 **.....115**
Ecuación 12 **.....115**
Ecuación 13 **.....116**
Ecuación 14 **.....117**
Ecuación 15 **..... 247**
Ecuación 16 **..... 247**
Ecuación 17 **..... 248**
Ecuación 18 **..... 248**

CAPÍTULO 1:

GENERALIDADES

1 GENERALIDADES

1.1 INTRODUCCIÓN

El Seguro Médico Delegado de COTEL (SMDC) es una institución vital para brindar atención médica a los trabajadores de la Cooperativa de Telecomunicaciones de La Paz R.L. y sus familias, siendo delegada por la Caja Nacional de Salud, como policonsultorio central y policonsultorios en El Alto y la Zona Sur, garantiza un acceso amplio a servicios médicos de calidad.

Sin embargo, a pesar de su importancia, el SMDC enfrenta desafíos significativos en términos de eficiencia y gestión de documentos. La falta de integración entre las áreas de vigencia de derechos y fichaje ha generado problemas operativos, como la emisión de fichas a personas no habilitadas, duplicidad de esfuerzos administrativos y una gestión ineficiente de la información de los beneficiarios y atenciones.

Para abordar estos desafíos, se plantea el desarrollo e implementación de un sistema integrado de gestión de documentos y emisión de fichas, empleando Kubernetes para la orquestación de contenedores. Este enfoque busca mejorar la eficiencia y calidad en la gestión de documentos de vigencia y en el proceso de emisión de fichas, facilitando el acceso, organización y comunicación de la información dentro de las áreas del SMDC.

1.2 ANTECEDENTES

1.2.1 Antecedentes institucionales

El Seguro Médico Delegado de COTEL (SMDC) es una institución delegada por la Caja Nacional de Salud. Tiene domicilio legal en la calle Socabaya esquina Mercado Edificio Gran Centro II entre 3º y 4º Piso y también cuenta con policonsultorios en El Alto y San Miguel. (Barrios, 2019)

El Seguro otorga atención en consulta externa a los trabajadores(as) y familias dependientes de la Cooperativa de Teléfonos Automáticos La Paz Ltda., en el policonsultorio central ubicado calle Socabaya esquina Mercado 4to piso edifico Gran Centro, brindando servicios en las especialidades de Medicina General, Cirugía, Medicina

Interna, Gineco-Obstetricia, Pediatría, Neurología, Otorrinolaringología, Psiquiatría, Medicina del Trabajo, Hematología, Fisioterapia, Ecografía y Odontología además de los servicios de Farmacia y Laboratorio. El consultorio de la ciudad de “El Alto” está ubicado en la calle Raúl Salmón Nº 150 esquina calle 10 de la zona 12 de octubre; en el policonsultorio de San Miguel que se encuentra ubicado en la calle 23 de Calacoto presta atención en Medicina General y Odontología. El resto de las especialidades médicas son otorgadas a través de la compra de servicios bajo las modalidades de papeleta y monto fijo. (Barrios, 2019)

Las internaciones se efectúan en la Clínica Rengel, zona de Sopocachi Calle Víctor Sanjinés Nº 276 y en la Clínica C-MAIO Obrajes Calle 10; así como también en otras instituciones de salud de acuerdo a requerimiento, enfermedades y/o necesidades de nuestros asegurados. (Barrios, 2019)

El servicio que se realiza a trabajadores y beneficiarios dependientes de la Cooperativa de Teléfonos Ltda. COTEL LA PAZ Ltda., está sujeta a las previsiones contenidas en el código de seguridad social a corto plazo. (Barrios, 2019)

1.2.2 Antecedentes académicos

SISTEMA WEB DE ADMINISTRACION DE HISTORIAS CLÍNICAS, este proyecto fue desarrollado por el Ing. Cesar Ariel Gonzales Paz, en el año 2020, en la Universidad Pública De El Alto, cuyo objetivo fue: “Desarrollar un sistema web de administración de Historias Clínicas que permitan mejorar las tareas de registro, búsqueda y elaboración de reportes de manera rápida y eficiente, facilitando la entrega oportuna y precisa del registro de datos y manejo de la información en los consultorios del médico especialista, brindando una mejor atención a los pacientes” (Paz, 2020). Dicho proyecto se enfoca exclusivamente en la gestión de documentos médico-legales con un sistema web proveyendo un método de almacenaje y cifrado de archivos para documentos sensibles. En comparación con el proyecto, utilizaremos un cifrado de documentos similar utilizando el cifrado con el algoritmo AES-256-CBC de laravel y a diferencia se utilizará un entorno de escritorio realizado en electrón para la visualización.

DESARROLLO DE UN SISTEMA WEB DE GESTIÓN DE FICHAS MÉDICAS PARA LA ORGANIZACIÓN “SALUD INTEGRAL PARA TODOS”, BAJO LA METODOLOGÍA SCRUM, este proyecto fue desarrollado por el Ing. Luis Fernando Ordóñez Armijos, en el año 2019, en la Universidad Tecnológica Israel, cuyo objetivo fue: “Desarrollar un sistema web para la gestión de fichas médicas para la organización Salud Integral para Todos, usando la plataforma de programación JEE para aplicaciones web y la metodología de desarrollo ágil Scrum, con el fin de mejorar la calidad del servicio que ofrece a sus pacientes” (Armijos, 2019). Dicho proyecto se enfoca exclusivamente en el desarrollo de un sistema web para la emisión de fichas, aplicando la metodología de desarrollo ágil Scrum para la gestión del proyecto. En comparación con el proyecto, utilizaremos la misma metodología de desarrollo “SCRUM” y desarrollaremos un sitio web para que los beneficiarios puedan sacar fichas, pero a diferencia con proyecto que se está desarrollando el sitio web será un cliente programado en React para la visualización.

ORQUESTACIÓN DE CONTENEDORES CON KUBERNETES, este proyecto fue desarrollado por el Ing. Javier Nogués García, en el año 2021, en la Universidad Carlos III de Madrid, cuyo objetivo fue: “Estudiar de las ventajas que ofrece Kubernetes, al orquestar contenedores para el cumplimiento de las motivaciones de desarrollo de proyectos” (García, 2021). Dicho proyecto se enfoca exclusivamente en la aplicación de Kubernetes para la orquestación de contenedores lo cual es aplicable a cualquier proyecto de gran magnitud. En comparación con el proyecto, utilizaremos una orquestación similar, desplegando el CLUSTER en DigitalOcean en vez de AWS y se diseñara la arquitectura de una aplicación monolítica distribuida.

DESARROLLO E IMPLEMENTACION DE UN SISTEMA DE FICHAS MÉDICA, PARA EL CONSULTORIO DE LA DRA. SARA DELGADO, este proyecto fue desarrollado por el Ing. David Alejandro Ramírez Prado, en el año 2021, en la Universidad Politécnica Salesiana Sede Guayaquil, cuyo objetivo fue: “Implementar un sistema web que permita llevar el control de fichas médicas” (Prado, 2021). Dicho proyecto se enfoca exclusivamente en el desarrollo de un sistema web para el control de fichas proveyendo una herramienta que facilite la emisión de fichas mediante un sitio web accesible a los pacientes. En comparación con el proyecto, utilizaremos una organización y emisión de fichas similares, con la diferencia de que el proyecto que se está desarrollando tendrá características únicas al ser para el SMDC.

1.3 PLANTEAMIENTO DEL PROBLEMA

1.3.1 Identificación del problema

El seguro médico delegado de COTEL enfrenta un desafío significativo debido a la falta de integración y completitud de sus sistemas informáticos actuales. Las áreas clave, como la vigencia de derechos y el fichaje, operan con sistemas independientes que no están interconectados, lo que genera duplicidad de datos y procesos ineficientes. Además, el sistema actual carece de funcionalidades esenciales para una gestión óptima, lo que dificulta la administración de documentos, la emisión de reportes y la coordinación efectiva de las actividades relacionadas con los beneficiarios. Estas limitaciones no solo afectan la eficiencia operativa, sino que también comprometen la calidad del servicio y la capacidad del seguro para responder a las necesidades de los asegurados de manera rápida y precisa.

Las principales causas y efectos del problema son redactados en la siguiente tabla:

Tabla 1: *Tabla de causas y efectos*

Fuente: Elaboración propia

CAUSAS	EFECTOS
Falta de sincronización entre los sistemas de fichaje y vigencia: La carencia de un mecanismo efectivo para la sincronización de datos entre los sistemas de fichaje y vigencia contribuye a la desconexión y la falta de coherencia en la información relacionada con los beneficiarios del seguro médico.	- Duplicación de esfuerzos y recursos en la gestión de la información: La falta de integración entre los sistemas y la necesidad de realizar tareas manuales para reconciliar la información lleva a una duplicación innecesaria de esfuerzos y recursos, lo que afecta negativamente la eficiencia operativa. - Dificultad para realizar análisis y reportes precisos sobre la población asegurada y las tendencias de salud: La falta de integridad y precisión en los datos no ayuda en la generación de análisis y reportes precisos sobre la población asegurada y las tendencias de salud, lo que limita la capacidad de la organización para tomar decisiones informadas y estratégicas.

Ausencia de un proceso eficiente para la actualización y verificación de la vigencia de los derechos de los beneficiarios: La falta de un proceso claro y eficiente para actualizar y verificar la vigencia de los derechos de los beneficiarios resulta en una gestión ineficaz de la información, lo que podría llevar a la habilitación errónea o desactualizada de los beneficiarios.	- Atención a beneficiarios no habilitados: La falta de verificación adecuada de la vigencia de los derechos de los asegurados resulta en la atención médica a personas que no están habilitadas para recibir los servicios del seguro médico, lo que genera costos adicionales y problemas legales. - Riesgo de incumplimiento normativo y legal debido a la falta de verificación adecuada de los documentos requeridos para la habilitación: La falta de una verificación adecuada de la documentación necesaria para la habilitación de los beneficiarios aumenta el riesgo de incumplimiento con las regulaciones y normativas establecidas, lo que podría tener consecuencias legales y financieras para la organización.
Deficiencias en la comunicación y colaboración entre las áreas de fichaje y vigencia: La falta de una comunicación efectiva y una colaboración adecuada entre las áreas de fichaje y vigencia dificulta la identificación y resolución de problemas relacionados con la gestión de la información de los beneficiarios, lo que puede generar discrepancias y confusiones en los datos.	- Mayor probabilidad de errores en el registro y la gestión de los beneficiarios: La falta de un proceso claro y automatizado aumenta la probabilidad de errores humanos en el registro y la gestión de la información de los beneficiarios, lo que puede conducir a decisiones incorrectas y afectar la calidad del servicio.
Impacto en la satisfacción del usuario: La falta de integración y funcionalidad en el sistema provoca demoras y problemas en el procesamiento de la información.	- Frustración y baja satisfacción de los usuarios: Los beneficiarios y el personal administrativo experimentan dificultades debido a procesos lentos y poco transparentes, lo que afecta negativamente su experiencia con el servicio.
Consecuencias a largo plazo: El sistema no tiene capacidad para gestionar eficientemente el crecimiento y las futuras necesidades.	- Aumento de costos y pérdida de confianza: La falta de mejoras puede llevar a costos operativos más altos, pérdida de confianza de los asegurados y dificultades para cumplir con las normativas, poniendo en riesgo la sostenibilidad del seguro.

1.3.2 Formulación del problema

La limitada integración entre los sistemas independientes de vigencia de derechos y el fichaje del seguro médico delegado de COTEL han resultado en una gestión ineficiente de los procesos administrativos y de control de registros. Esta situación incrementa el riesgo

de errores, duplicaciones y omisiones en los registros de los beneficiarios. Como consecuencia, surgen inconsistencias en los datos y documentos, comprometiendo la calidad y precisión de la información crítica. Esto afecta la eficiencia administrativa y restringe la capacidad de tomar decisiones informadas y precisas.

1.4 OBJETIVOS

1.4.1 Objetivo general

Desarrollar un sistema integrado de gestión de documentos y emisión de fichas médicas empleando Kubernetes para la orquestación de contenedores en las áreas de fichaje y vigencia de derechos del seguro médico delegado de COTEL, con el fin de eliminar la inconsistencia significativa en los datos y documentos de los beneficiarios.

1.4.2 Objetivos específicos

- Realizar el análisis de las actividades de las áreas de vigencia de derechos y fichaje en el seguro médico delegado de COTEL para determinar los requerimientos funcionales y no funcionales del sistema.

- Diseñar y desarrollar la arquitectura y las APIs REST independientes para cada módulo del sistema, junto con sus bases de datos correspondientes, asegurando una adecuada comunicación entre los servicios y apoyando las actividades de las áreas de vigencia y fichaje.

- Diseñar y desarrollar una infraestructura de Kubernetes en un servidor para la orquestación de los contenedores en los que se desplegaran los módulos monolíticos de vigencia de derechos y fichaje.

- Validar el sistema de gestión realizando pruebas unitarias y funcionales para determinar el correcto funcionamiento

1.5 JUSTIFICACIÓN

1.5.1 Justificación técnica

La presente investigación busca desarrollar un sistema integrado de gestión de documentos y emisión de fichas médicas para el Seguro Médico Delegado de COTEL**,** con el objetivo de optimizar la administración de los datos y documentos de los beneficiarios. Esta investigación se justifica ante la necesidad apremiante de resolver problemas de inconsistencia de datos y falta de integración en los sistemas actuales, los cuales afectan directamente la calidad del servicio brindado a los asegurados.

La implementación de una arquitectura monolítica distribuida en Docker y orquestada por Kubernetes proporciona una infraestructura escalable y robusta que garantiza la disponibilidad y el rendimiento del sistema. Esta solución es fundamental para reducir la duplicidad de esfuerzos y errores en la gestión de información, lo que, a su vez, mejorará la satisfacción del usuario y garantizará que los beneficiarios reciban atención adecuada y oportuna. La investigación, por lo tanto, responde a un contexto real donde los procesos actuales presentan fallas significativas.

Desde un enfoque técnico, esta investigación tiene un alcance significativo en el área de ingeniería de sistemas, ya que propone la utilización de tecnologías modernas como Docker**,** Kubernetes**,** Websockets**,** React**,** y Lumen. Estas tecnologías no solo aportan robustez y escalabilidad al sistema, sino que también garantizan un rendimiento óptimo, permitiendo que el sistema se adapte a futuras demandas y requerimientos en el entorno de salud.

1.5.2 Justificación económica

a implementación de este sistema de gestión de documentos y emisión de fichas médicas conlleva importantes implicaciones económicas que justifican su desarrollo. En primer lugar, representa una inversión a largo plazo que tiene el potencial de optimizar de manera significativa los recursos de la organización. Al abordar y reducir errores en la gestión de datos, así como mejorar la eficiencia operativa, se prevé una disminución considerable de los costos asociados con la atención de beneficiarios no habilitados. Esta optimización no solo impacta directamente en la rentabilidad de la institución, sino que también permite

cumplir con las normativas y regulaciones, lo que es esencial para mantener la confianza de los usuarios y evitar sanciones.

La centralización de la información en un sistema integrado facilita una gestión más eficiente de los datos, lo que se traduce en menores costos de mantenimiento y administración. Al eliminar la necesidad de gestionar infraestructuras dispersas, se minimizan los riesgos de pérdida o duplicación de información, que pueden resultar en costos adicionales y en la pérdida de credibilidad ante los beneficiarios. Además, la reducción de estos riesgos es fundamental para garantizar la sostenibilidad financiera de la institución a largo plazo.

La adopción de una arquitectura modular basada en Docker y orquestada por Kubernetes ofrece una ventaja competitiva adicional. Esta estructura no solo permite una escalabilidad óptima, sino que también contribuye a la reducción de gastos relacionados con actualizaciones y expansiones futuras. Gracias a esta modularidad, la organización puede implementar nuevas funcionalidades o modificar las existentes sin la necesidad de realizar reconstrucciones o reconfiguraciones complejas de la infraestructura. Esto se traduce en un ahorro significativo en términos de tiempo y costos, permitiendo a la institución reorientar sus recursos hacia otras áreas estratégicas.

1.5.3 Justificación social

La implementación de un Sistema Integrado de Gestión de Documentos y Emisión de Fichas para el Seguro Médico Delegado de COTEL promete generar una transformación significativa en la eficiencia y calidad del servicio de consulta externa para todos los usuarios. Este nuevo sistema no solo reemplazará los procesos manuales de emisión de fichas, sino que también introducirá un enfoque automatizado que optimiza la gestión de documentos relacionados con la vigencia de derechos. Como resultado, se facilitará un acceso más ágil y sencillo a los servicios médicos, beneficiando directamente a los beneficiarios y a sus familias.

Este avance en la automatización tiene un impacto profundo en la reducción de la burocracia y los tiempos de espera, permitiendo a los usuarios recibir atención médica de manera más oportuna. Al disminuir los retrasos en la emisión de fichas y en la verificación

de la vigencia de derechos, se mejora la experiencia del usuario, lo que contribuye a un mayor nivel de satisfacción y confianza en el sistema de salud.

Además, la eliminación de cargas administrativas pesadas que actualmente afectan tanto a los usuarios como al personal de salud es un factor crucial. Los trabajadores del sistema de salud podrán concentrarse en brindar atención de calidad, en lugar de lidiar con trámites y procesos burocráticos innecesarios. Esta mejora en la eficiencia no solo beneficia a los trabajadores, sino que también promueve un ambiente laboral más positivo y productivo, lo cual es fundamental para la salud y el bienestar de todos los involucrados.

1.6 ALCANCES

1.6.1 Alcance temático

- **Área general:** Ingeniería de sistemas

- **Áreas específicas:** Bases de datos, Ingeniería de Software, Proyectos Web y Cloud Computing

- **Área temática:** Orquestación de contenedores y administración de clústeres en Kubernetes.

1.6.2 Alcance geográfico

Este trabajo se llevará a cabo para el Seguro Médico Delegado de COTEL, cuya sede está ubicada en la intersección de las calles Socabaya y Mercado, específicamente en el Edificio Gran Centro II, ocupando los pisos 3º y 4º, en la ciudad de La Paz. Para más detalles sobre esta ubicación, se puede consultar el ANEXO B.

1.6.3 Alcance temporal

El desarrollo de este proyecto de grado abarcará la totalidad de los dos semestres académicos del año 2024, en estricta observancia de las políticas y regulaciones establecidas por la Universidad La Salle Bolivia. Esta extensión temporal permitirá una

dedicación adecuada a cada etapa del proyecto, desde la investigación inicial hasta la implementación y evaluación final, garantizando así la calidad del trabajo realizado.

1.7 LIMITES

Los límites del Sistema Integrado de Gestión de Documentos y Emisión de Fichas para el Seguro Médico Delegado de COTEL son los siguientes:

- El proyecto solo contempla la migración de datos históricos y la conversión del sistema existente de vigencia de derechos no así del de fichaje. Para el sistema de fichaje solo se centrará en la emisión de fichas médicas a partir del momento de la implementación.

- No se incluirá la integración con otros sistemas externos a este.

- No se incluirá la provisión de hardware o infraestructura física. El proyecto se enfocará únicamente en el desarrollo de software y su implementación en la infraestructura existente de COTEL.

CAPÍTULO 2:

MARCO TEÓRICO

2 MARCO TEÓRICO

2.1 INTRODUCCIÓN

En este capítulo, se presentará una serie de conceptos y definiciones esenciales que se considerarán para alcanzar los objetivos establecidos en el desarrollo del proyecto. Estos elementos teóricos se han recopilado de diversas fuentes, lo que les confiere una sólida fundamentación para la descripción y análisis del problema planteado. A lo largo de esta sección, se profundizará en las teorías y enfoques relevantes que sustentan la investigación, proporcionando así un marco teórico claro y cohesivo que guiará el desarrollo del trabajo y permitirá una mejor comprensión de los desafíos que se abordarán.

2.1.1 Sistemas de salud

De acuerdo con el Diccionario de Acción Humanitaria y Cooperación al Desarrollo (DAHCD), un sistema de salud se define como un conjunto de infraestructuras sanitarias que, bajo una política específica, tiene como finalidad mejorar la salud de la población. Este sistema, conocido también como sistema sanitario, está formado por diversas instituciones, organismos y servicios que, mediante acciones organizadas y planificadas, implementan una variedad de programas de salud. En resumen, los sistemas de salud son modelos que describen cómo se gestiona la atención sanitaria, así como estructuras organizativas que reflejan la política de salud de cada país. (Abrisketa, n.d.)

2.1.1.1 Tipos de sistemas

La mayoría de los países en desarrollo no cuentan con un único sistema de salud; en cambio, disponen de varios subsistemas de financiamiento y provisión, que incluyen el sector público, el privado con fines de lucro, el privado gestionado por ONG, y las prácticas tradicionales o alternativas. (Abrisketa, n.d.) se ha establecido una clasificación de los sistemas de salud en naciones de bajos ingresos, basada en su ideología política, en cuatro categorías:

- Sistema de salud empresarial: en este modelo, las redes nacionales de atención sanitaria son débiles y se utilizan poco, mientras que gran parte de los servicios son privados.

- Sistema de salud socialista: se distingue por una política sanitaria en la que el gobierno ejerce un fuerte control sobre la planificación y provisión de servicios de salud.

- Sistema amplio de salud: este enfoque combina aspectos de atención sanitaria con acciones preventivas, tales como educación, nutrición y acceso a agua potable.

- Sistema sanitario orientado al bienestar: se caracteriza por un importante sector privado que opera bajo control gubernamental, con influencias socialistas y un enfoque en la Atención Primaria de Salud.

2.1.2 Seguro médico

La Asociación Estadounidense de Seguros Sanitarios define el seguro médico como una forma de cobertura destinada a garantizar el pago de indemnizaciones en caso de enfermedades o lesiones. Esta cobertura abarca no solo los gastos vinculados a tratamientos médicos, sino que también incluye seguros por pérdidas que pueden surgir de accidentes, gastos médicos directos y situaciones de incapacidad temporal o permanente. Además, el seguro médico se extiende a casos de fallecimiento o desmembramiento accidental, ofreciendo así una protección financiera adicional frente a eventos inesperados que pueden impactar significativamente la vida de una persona. (Caxton)

En este contexto, el seguro médico se considera una herramienta esencial para asegurar una atención adecuada y oportuna en situaciones de salud desfavorables, proporcionando tranquilidad y seguridad tanto a los individuos como a sus familias.

2.1.3 Seguros médicos en Bolivia

2.1.3.1 Situación actual

En Bolivia, el Instituto Nacional de Seguros de Salud (INASES) regula los seguros médicos sociales, siendo responsable de la supervisión y regulación de los seguros de salud de corto plazo, que abarcan a organizaciones como la Caja Nacional de Salud (CNS). La CNS se destaca como uno de los principales proveedores de seguros sociales, brindando cobertura a los trabajadores y sus familias a través de contribuciones obligatorias. (Infoleyes, 2000)

2.1.3.2 Seguros médicos delegados

Los seguros médicos delegados son una modalidad en la que las empresas gestionan directamente los servicios de salud de sus empleados mediante acuerdos con proveedores de salud, transfiriendo así la responsabilidad de la atención médica a estas entidades. Este tipo de seguros tiende a ofrecer una atención más inmediata para los afiliados, en contraste con los seguros sociales tradicionales que son administrados por organizaciones como la Caja Nacional. (Infoleyes, 2000)

2.1.3.3 Afiliación de personas

En Bolivia, se cuenta con un reglamento específico que regula las afiliaciones, el cual establece una serie de documentos esenciales para llevar a cabo una validación adecuada de los requerimientos. La normativa (ASSUS, 2022),detalla los requisitos necesarios que deben ser cumplidos para asegurar que el proceso de afiliación se realice de manera efectiva y conforme a las disposiciones legales. Estos requisitos son fundamentales para garantizar la transparencia y la correcta implementación de los procedimientos establecidos.

Los empleadores sujetos al campo de aplicación del Código de Seguridad Social, su Reglamento y Normas Conexas, tienen la obligación de registrar a sus trabajadores en el Ente Gestor de la Seguridad Social de Corto Plazo respectivo, dentro del plazo máximo de cinco (5) días hábiles, a partir de la fecha de iniciada la relación laboral, incluyendo el periodo de prueba, para lo que deberán utilizar el formulario de "Aviso de Afiliación del Trabajador". (ASSUS, 2022)

a. Requisitos para la afiliación de trabajadores:

Los trabajadores, al momento de su afiliación al Ente Gestor de la Seguridad Social de Corto Plazo correspondiente, tienen la obligación de presentar los siguientes requisitos:

- Formulario de Aviso de Afiliación o Reingreso del trabajador, debidamente llenado, firmado y sellado por el empleador, representante legal, jefe o encargado de la Unidad de Recursos Humanos (Original).

- Memorándum de designación, Resolución, contrato laboral u otro documento que acredite la relación laboral (fotocopia simple).

- Certificado de nacimiento del trabajador actualizado (Original).

- Cédula de Identidad del trabajador (Fotocopia simple) o Cédula de Identidad de Extranjería o de residencia para ciudadanos extranjeros (Fotocopia simple).

Para la presentación del certificado del examen pre ocupacional se otorgará un plazo máximo de treinta (30) días calendario al trabajador, no siendo óbice para la negativa de afiliación y/o atención de salud inmediata.

b. Requisitos para afiliación de beneficiario esposo o esposa:

El esposo u esposa del asegurado deberá presentar únicamente los siguientes requisitos:

- Carnet de Asegurado o formulario de Aviso de Afiliación o Reingreso del asegurado (Fotocopia simple).

- Certificado de matrimonio actualizado (Original).

- Certificado de nacimiento del esposo o la esposa actualizado (Original).

- Cédula de Identidad del esposo o la esposa (Fotocopia simple).

- Formulario o certificado de no afiliación del esposo o la esposa, sellado y rubricado por los Entes Gestores de la Seguridad Social de Corto Plazo o certificación emitida por la Autoridad de Supervisión de la Seguridad Social de Corto Plazo - ASUSS (Original).

- Estado de ahorro personal previsional extendido por la Administradora de Fondo de Pensiones - AFP respectiva, en su defecto certificado de no aportación actualizado, que evidencie la no existencia de aportes realizado por terceros vigente (Original).

- Formulario de Inserción de beneficiarios del Ente Gestor de la Seguridad Social de Corto Plazo correspondiente (Original).

- Última boleta de pago del o la titular del derecho, si corresponde (Fotocopia simple).

c. Requisitos para afiliación de beneficiarios pareja en unión libre

Él o la titular del derecho, podrá solicitar la afiliación de su pareja en Unión libre, para lo que deberá presentar únicamente los siguientes requisitos:

- Carnet de Asegurado o formulario de Aviso de Afiliación o Reingreso del o la titular del derecho (Fotocopia simple).

- Certificado de nacimiento del o el conviviente actualizado (Original).

- Cédula de Identidad del o la conviviente (Fotocopia simple).
- Formulario o certificado de no afiliación del o la conviviente, sellado y rubricado por los Entes Gestores de la Seguridad Social de Corto Plazo o certificación emitida por la Autoridad de Supervisión de la Seguridad Social de Corto Plazo - ASUSS (Original).

- Estado de ahorro personal previsional extendido por la Administradora de Fondo de Pensiones AFP respectiva, en su defecto certificado de no aportación actualizado, que evidencie la no existencia de aportes realizado por terceros vigente (Original).

- Resolución o certificado de registro de Unión Libre emitido por autoridad competente, según corresponda (Original o Fotocopia legalizada).

- Formulario de Inserción de Beneficiarios del Ente Gestor de la Seguridad Social de Corto Plazo correspondiente (Original).

- Última boleta de pago del o la titular del derecho, si corresponde (Fotocopia simple).

d. Requisitos para afiliación de recién nacidos

Los progenitores, tienen la obligación de afiliar a sus hijos recién nacidos presentando únicamente los siguientes requisitos:

- Carnet de Asegurado o formulario de Aviso de Afiliación o Reingreso del o la titular del derecho (Fotocopia simple).

- Certificado de nacimiento del recién nacido (Original).

- Formulario de Inserción de Beneficiarios del Ente Gestor de la Seguridad Social de Corto Plazo correspondiente (Original).

- Última boleta de pago del titular del derecho, si corresponde. (Fotocopia simple).

El asegurado tiene un plazo de treinta (30) días calendario para afiliar al recién nacido. Asimismo, cuando el recién nacido requiera de manera inmediata prestaciones en especie, los Entes Gestores de la Seguridad Social de Corto Plazo deben proceder a su afiliación inmediata.

e. Requisitos para afiliación de beneficiarios hijos hasta cumplidos diecinueve (19) años

El asegurado deberá registrar a cada uno de sus beneficiarios hijos hasta cumplidos los diecinueve (19) años de edad, presentando únicamente los siguientes requisitos:

- Carnet de Asegurado o formulario de Aviso de Afiliación o Reingreso del o la titular del derecho (Fotocopia simple).

- Certificado de nacimiento del beneficiario hijo actualizado (Original).

- Cédula de Identidad del beneficiario hijo (Fotocopia simple).

- Formulario de Inserción de Beneficiarios del Ente Gestor de la Seguridad Social de Corto Plazo correspondiente (Original).

- Formulario o certificado de no afiliación del hijo menor hasta cumplidos los diecinueve (19) años de edad, sellado y rubricado por los Entes Gestores de la Seguridad Social de Corto Plazo o certificación emitida por la Autoridad de Supervisión de la Seguridad Social de Corto Plazo - ASUSS (Original).

- Última boleta de pago del titular del derecho, si corresponde (Fotocopia simple).

f. Ampliación de prestaciones a beneficiarios hijos hasta cumplidos los veinticinco (25) años

Los beneficiarios hijos del asegurado, mayores de diecinueve (19) años, tienen derecho a la ampliación de prestaciones hasta cumplidos los veinticinco (25) años de edad en el Ente Gestor de la Seguridad Social de Corto Plazo correspondiente, debiendo cumplirse obligatoriamente con las siguientes condicionantes:

- Si estudian en establecimientos autorizados por el Estado.
- Los que acrediten encontrarse con el estado civil de soltero.

Los requisitos que deben presentar para la ampliación de prestaciones, únicamente son:

- Carnet de Asegurado o formulario de Aviso de Afiliación o Reingreso del o la titular del derecho (Fotocopia simple).

- Certificado de nacimiento del beneficiario hijo actualizado (Original).

- Cédula de Identidad del beneficiario hijo (Fotocopia simple).

- Formulario de Inserción de Beneficiarios del Ente Gestor de la Seguridad Social de Corto Plazo correspondiente (Original).

- Formulario o certificado de no afiliación de los beneficiarios hijos hasta cumplidos los veinticinco (25) años de edad, sellado y rubricado por los Entes Gestor de la Seguridad Social de Corto Plazo o certificación emitida por la Autoridad de Supervisión de la Seguridad Social de Corto Plazo - ASUSS (Original).

- Última boleta de pago del titular del derecho, si corresponde (Fotocopia simple).

- Matrícula de inscripción de la gestión vigente y/o certificación del establecimiento educativo que acredite que se encuentra inscrito en cursos de educación secundaria, cursos de educación superior técnico o educación superior, según corresponda (Originales).

- Certificado de soltería emitido por el Servicio de Registro Cívico - SERECI (Original).

- Estado de ahorro personal previsional extendido por la Administradora de Fondo de Pensiones AFP respectiva, en su defecto certificado de no aportación actualizado, que evidencie la no existencia de aportes realizado por terceros vigente (Original).

Para la ampliación de prestaciones a hijos mayores de diecinueve (19) años de edad, no se requiere aprobación de la Comisión de Prestaciones del Ente Gestor de la Seguridad Social de Corto Plazo.

Así también corresponderá la ampliación de prestaciones sin límite de edad, a los hijos que fueron declarados inválidos por los servicios médicos del Ente Gestor de la Seguridad Social de Corto Plazo, antes de cumplir las edades establecidas.

g. Requisitos para afiliación de beneficiarios hijos declarados inválidos:

Son requisitos para la afiliación de los beneficiarios hijos del asegurado, declarados inválidos por los servicios médicos del Ente Gestor de la Seguridad Social de Corto Plazo, los siguientes:

- Carnet de Asegurado o formulario de Aviso de Afiliación o Reingreso del o la titular del derecho (Fotocopia simple).

- Certificado de nacimiento del beneficiario hijo actualizado (Original).

- Cédula de Identidad del beneficiario hijo (Fotocopia simple).

- Formulario de Inserción de Beneficiarios del Ente Gestor de la Seguridad Social de Corto Plazo correspondiente (Original).

- Última boleta de pago del titular del derecho, si corresponde (Fotocopia simple). Certificado de soltería emitido por el Servicio de Registro Cívico - SERECI, si corresponde (Original).

- Estado de ahorro personal previsional extendido por la Administradora de Fondo de Pensiones AFP respectiva o en su defecto certificado de no aportación actualizado, que evidencie la no existencia de aportes realizados por terceros vigente (Original).

- Resolución de declaratoria de invalidez emitida por la Comisión de Prestaciones del Ente Gestor de la Seguridad Social de Corto Plazo, que podrá activar su afiliación dentro del Sistema de Seguridad Social.

h. Requisitos para afiliación de beneficiarios padres

Los asegurados pueden afiliar a su padre y/o madre, en el Ente Gestor de la Seguridad Social de Corto Plazo correspondiente cuando no accedan a un seguro por derecho propio.
Al efecto deben presentar únicamente los siguientes requisitos:

- Carnet de Asegurado o formulario de Aviso de Afiliación o Reingreso del o la titular del derecho (Fotocopia simple).

- Certificados de nacimiento del o los padres actualizado (Original).
- Cédulas de Identidad del o los padres (Fotocopias simples).

- Formulario de Inserción de Beneficiarios del Ente Gestor de la Seguridad Social de Corto Plazo correspondiente (Original).

- Última boleta de pago del titular del derecho (Fotocopia simple).

- Formulario o certificado de no afiliación de los beneficiarios padres, sellado y rubricado por los Entes Gestores de la Seguridad Social de Corto Plazo o certificación emitida por la Autoridad de Supervisión de la Seguridad Social de Corto Plazo - ASUSS (Original).

- Estado de ahorro personal previsional extendido por la Administradora de Fondo de Pensiones AFP respectiva o en su defecto certificado de no aportación actualizado, que evidencie la no existencia de aportes realizados por terceros vigente (Original).

- Certificado de otorgación o no de renta emitido por el Servicio Nacional del Sistema de Reparto SENASIR actualizado (Original).

- Certificado electrónico del Servicio de Impuestos Nacionales con estado Inactivo (Original).

2.1.3.4 Desafiliación de personas

La solicitud de Desafiliación y consecuente Reafiliación, deberá ser presentada por los empleadores, mediante nota fundamentada, en la cual se hará conocer los justificativos debidamente respaldados, mismos que serán analizados por el Comité Técnico de Desafiliación y Reafiliación (ASSUS, 2022)

a. Requisitos para el proceso de desafiliación:

Los empleadores que soliciten la Desafiliación y consecuente Reafiliación, deberán presentar de manera conjunta a la solicitud, los siguientes requisitos.

- Comprobantes del pago de aportes de los tres (3) últimos meses al Ente Gestor de la Seguridad Social de Corto Plazo.
- Formulario de Aviso de Afiliación del Empleador o Certificación por el cual se acredite la afiliación mínima de tres (3) años en el Ente Gestor de la Seguridad Social de Corto Plazo de origen (Fotocopia legalizada).

- Nómina de trabajadores activos, en la que se debe incluir el número de Cédula de Identidad, matrícula del asegurado y el número de beneficiarios asegurados (Original).

- Última planilla de pago de haberes cotizable, del empleador (Original).

- Certificado de no adeudo emitido por el Ente Gestor de la Seguridad Social de Corto Plazo de origen, actualizado a la fecha de solicitud de la Desafiliación (Original).

- Documentos de constitución de la empresa o de creación de la institución pública o privada (Fotocopias Legalizadas).

El requisito establecido en el inciso e) del parágrafo anterior será solicitado por el empleador ante el Ente Gestor de la Seguridad Social de Corto Plazo correspondiente, el cual deberá ser emitido en un plazo máximo de sesenta (60) días calendario. Si el Ente Gestor no emitiere el referido requisito en el plazo establecido, el trámite de Desafiliación podrá ser presentado omitiendo la presentación del mismo.

b. Comité técnico de desafiliación y reafiliación – ASUSS

El Comité Técnico de Desafiliación y Reafiliación, estará conformado por técnicos de las siguientes Direcciones:

- Dirección Técnica de Fiscalización y Control de Servicios de Salud de la Seguridad Social de Corto Plazo.

- Dirección Técnica de Fiscalización y Control Administrativo Financiero de la Seguridad Social de Corto Plazo.

- Dirección Jurídica - Unidad de Afiliación, Desafiliación y Reafiliación.

- Reunirse de acuerdo a procedimiento establecido en el presente reglamento y las veces que el Comité Técnico de Desafiliación y Reafiliación considere necesario.

2.1.3.5 Procedimiento de desafiliación

Presentada la solicitud, se proseguirá de acuerdo al siguiente procedimiento:

a. Fase inicial:

- Las solicitudes de Desafiliación y consecuente Reafiliación, deberán ser presentadas ante las oficinas regionales de la ASUSS, para que, por su área legal, se verifique el cumplimiento de los requisitos establecidos en el artículo 27 del presente reglamento. En caso de que se encuentren incompletos los requisitos exigidos, se otorgara un plazo de hasta quince (15) días hábiles para subsanar la observación, caso contrario la solicitud se tendrá como no presentada, disponiéndose su archivo por la oficina regional correspondiente, debiendo remitir una copia a la Unidad de Afiliación, Desafiliación y Reafiliación.

- Dentro del año de emitida y notificada la disposición de archivo por incumplimiento en la presentación de requisitos, los empleadores no podrán volver a presentar solicitud de desafiliación y consecuente Reafiliación.

b. Fase de conciliación:

- En las solicitudes que cumplan con la presentación de los requisitos, la oficina regional respectiva de la ASUSS, convocara de forma previa al solicitante y al Ente Gestor de la Seguridad Social de Corto Plazo, a audiencia de conciliación. En caso de que se llegue a una conciliación se procederá con el archivo de obrados, caso contrario se proseguirá con el trámite. En ambos casos se suscribirá un acta de conciliación o de imposibilidad de conciliación.
- Las solicitudes que cumplan con la presentación de los requisitos exigidos, serán remitidas ante el Director General Ejecutivo de la ASUSS, para su posterior procesamiento.

c. Fase de procesamiento:

- Radicada la solicitud en la Dirección General Ejecutiva de la ASUSS, se designara a los miembros del Comité Técnico de Desafiliación y Reafiliación.

- Una vez designado el Comité Técnico de Desafiliación y Reafiliación, se reunirá dentro el plazo máximo de diez días (10) hábiles, para determinar la metodología del análisis de la solicitud de Desafiliación y consecuente Reafiliación.

- Definida la metodología, el análisis correspondiente por parte de los miembros del Comité Técnico de Desafiliación y Reafiliacion, deberá realizarse en el plazo máximo de cuarenta (40) días hábiles, debiendo emitirse el correspondiente Informe Técnico.

- Concluido el periodo de análisis, en el plazo máximo de diez (10) días hábiles, el Comité Técnico de Desafiliación y Reafiliación, se reunirá en pleno para determinar mediante Informe Técnico conjunto, la aprobación o rechazo de la solicitud de Desafiliación y consecuente Reafiliación, recomendando a la Dirección General Ejecutiva, la emisión de Resolución Administrativa.

d. Fase final:

- Remitido el Informe Técnico de Desafiliación y Reafiliación a la Dirección General Ejecutiva de la ASUSS, el mismo previo visto bueno, será remitido a la Unidad de Afiliación, Desafiliación y Reafiliación, para la correspondiente elaboración del proyecto de Resolución Administrativa.

- La Unidad de Afiliación, Desafiliación y Reafiliación, en el plazo máximo de cinco (05) días hábiles, elaborara el proyecto de Resolución Administrativa, para su posterior remisión a la Dirección General Ejecutiva de la ASUSS.

- La Resolución Administrativa será notificada al empleador y a los Entes Gestores de la Seguridad Social de Corto Plazo para su cumplimiento.

2.1.4 Áreas involucradas en la atención médica en seguros sociales

En el proceso de atención médica dentro de los seguros médicos, se pueden identificar cuatro áreas clave: vigencia de derechos, fichaje, enfermería y atención médica por parte de doctores. A continuación, te proporciono una descripción de cada una de estas áreas:

2.1.4.1 Vigencia de derechos

Esta área se ocupa de asegurar que el asegurado tenga acceso a los servicios de salud según lo estipulado en las pólizas de su seguro. Esto implica verificar que la persona esté al corriente con sus pagos y que su cobertura esté vigente. La validez de los derechos garantiza que los pacientes cuenten con la protección financiera necesaria para acceder a la atención médica sin inconvenientes. (OMS, 2020)

2.1.4.2 Área de fichaje

El fichaje es el primer paso en el que se recopilan los datos del paciente al momento de su ingreso a la institución de salud. Este proceso abarca información personal, antecedentes médicos y el motivo de la consulta. Realizar un registro correcto es esencial para garantizar que el paciente reciba la atención correspondiente y que se conserve un historial clínico preciso. (Agudelo, 2017)

2.1.4.3 Área de enfermería

El personal de enfermería desempeña una función crucial en el ámbito de la atención médica. Su labor incluye la evaluación inicial del paciente, la administración de tratamientos, el monitoreo de la evolución del estado de salud del paciente y la educación sobre salud. Los enfermeros ofrecen atención constante y actúan como un puente entre el paciente y el médico. (OMS, 2020)

2.1.4.4 Área médica

Los médicos tienen la responsabilidad de identificar y tratar las afecciones de salud de los pacientes. Llevan a cabo exámenes clínicos, recomiendan tratamientos y realizan procedimientos médicos cuando es necesario. La interacción entre el médico y el paciente es crucial para el éxito del cuidado de la salud, ya que afecta tanto la confianza como la adherencia del paciente al tratamiento. (OMS, 2020)

2.2 INGENIERÍA DE SOFTWARE

2.2.1 Software

El software se describe como un conjunto organizado de instrucciones, algoritmos y elementos visuales que se combinan en un programa con el fin de interactuar con un procesador (Universitat Carlemany, 2024). En términos más accesibles, se puede considerar como el elemento esencial que activa y potencia las capacidades de un dispositivo, ya sea un ordenador, un teléfono inteligente u otros dispositivos electrónicos.

La naturaleza, complejidad y tamaño del software influyen en su compatibilidad con diferentes dispositivos. Por ejemplo, un software diseñado específicamente para un sistema operativo como Windows o macOS podría no funcionar en otros entornos. Además, las aplicaciones más sofisticadas, como los programas de edición de video o los videojuegos de alta definición, requieren hardware especializado y recursos específicos para funcionar de manera efectiva.

Asimismo, el software se divide en diversas categorías según su uso y funcionalidad. Esto incluye el software de sistema, que comprende los sistemas operativos que gestionan el hardware del dispositivo; el software de aplicación, que se refiere a programas diseñados para realizar tareas específicas, como el procesamiento de texto o la gestión de bases de datos; y el software de desarrollo, que abarca herramientas utilizadas por los programadores para crear otros programas (Universitat Carlemany, 2024). En un mundo cada vez más digital, el software desempeña un papel crucial en el funcionamiento y la eficacia de los dispositivos modernos, permitiendo a los usuarios llevar a cabo una amplia gama de tareas, desde las más simples hasta las más complejas. Esta relevancia del software en la tecnología contemporánea destaca su importancia tanto en el ámbito académico y profesional como en la vida cotidiana.

El software se define como un conjunto estructurado de instrucciones, algoritmos y elementos visuales que se integran en un programa con el propósito de interactuar con un procesador (Universitat Carlemany, 2024). En términos más simples, constituye el componente fundamental que permite activar y maximizar las capacidades de un dispositivo, ya sea un ordenador, un teléfono inteligente u otros aparatos electrónicos.

2.2.2 Metodologías de desarrollo de software

Como menciona (Gonzales, 2020), el INTECO indica que el desarrollo de software necesita de metodologías que orienten y faciliten las actividades y procesos, con el fin de alcanzar las metas y objetivos establecidos al inicio del proyecto de desarrollo, asegurando así un producto de calidad que se ajuste al ciclo de vida del proyecto.

Las metodologías de desarrollo son clave para gestionar el ciclo de vida de un proyecto. En este sentido, se pueden identificar tres tipos de metodologías, las cuales se diferencian por el tiempo de desarrollo y las iteraciones que se implementan para asegurar un proceso adecuado en la creación del software.

La necesidad de contar con procedimientos, técnicas, herramientas y documentación de soporte ha llevado a la creación de diversas metodologías para el desarrollo de productos de software. Estas metodologías tienen como objetivo guiar a los desarrolladores en la creación de nuevo software, aunque los requisitos varían considerablemente, lo que ha resultado en una amplia gama de metodologías (Carrillo, Pérez y Rodríguez, como se cita en César Gonzales, 2020).

Las metodologías de desarrollo de software se pueden clasificar en dos categorías principales:

- Metodologías orientadas al control de procesos: Estas metodologías definen con precisión las actividades a llevar a cabo, las herramientas a utilizar y las notaciones a implementar. Suelen ser conocidas como metodologías pesadas.

- Metodologías enfocadas en la interacción con el cliente y el desarrollo incremental: Estas metodologías permiten presentar versiones funcionales parciales del software al cliente en períodos cortos, brindándole la oportunidad de evaluar el producto y sugerir cambios durante el avance del desarrollo. A menudo se les denomina metodologías ligeras o ágiles.

2.2.3 Metodologías agiles

Las metodologías ágiles, según el Manifiesto Ágil (2001), se distinguen por un enfoque incremental en el desarrollo de software, donde las entregas son frecuentes y de menor tamaño, lo que facilita la rápida finalización de los ciclos de vida de los proyectos. Estas metodologías promueven una colaboración constante y activa entre los desarrolladores y los clientes, lo que permite adaptarse a los cambios que puedan surgir durante el proceso de desarrollo. Entre las ventajas de implementar métodos ágiles se encuentra la habilidad de responder de manera rápida a modificaciones en los requisitos y en el entorno de desarrollo (Agile Alliance, 2001). Algunos de los métodos más destacados dentro de este enfoque ágil son:

- Extreme Programming (XP)
- Scrum
- Familia de Metodologías Crystal
- Feature Driven Development
- Proceso Unificado Rational (RUP)
- OpenUP

2.2.4 Metodología scrum

En *The Scrum Guide*, se define Scrum como un marco de trabajo ágil que permite a las personas, equipos y organizaciones crear valor mediante soluciones adaptativas para enfrentar problemas complejos. (Schwaber & Sutherland, 2020) este enfoque se fundamenta en los principios de iteración e incrementalidad, organizando el trabajo en breves ciclos denominados *sprints*, que generalmente tienen una duración de entre 1 a 4 semanas. Cada *sprint* tiene como objetivo entregar una versión funcional y mejorada del producto, facilitando la obtención de retroalimentación continua y la posibilidad de ajustar el desarrollo según los cambios en los requisitos.

Según lo indicado por (Amazon Web Services, s.f.), la metodología Scrum se distingue por una serie de principios y valores:

2.2.4.1 Principios de Scrum para el éxito de los proyectos

a. Transparencia:

Los equipos operan en un entorno donde todos son conscientes de los desafíos que enfrentan los demás. Las interacciones periódicas cara a cara entre los integrantes del equipo multidisciplinario y los propietarios del proyecto ayudan a prevenir la falta de comunicación y los cuellos de botella en la información.

b. Reflexión:

Se incorporan momentos de reflexión frecuentes dentro del marco de trabajo para permitir que los miembros del equipo evalúen los avances realizados. Los gerentes de proyecto utilizan la información obtenida durante estas reuniones de revisión para realizar estimaciones y planificaciones futuras. Como resultado, los proyectos pueden ejecutarse de manera más eficiente, respetando tanto el presupuesto como los plazos establecidos.

c. Adaptación

Los integrantes del equipo tienen la capacidad de re-priorizar las tareas de acuerdo con los requisitos cambiantes de los clientes. Son ellos quienes determinan qué tareas se completan primero y cuáles se pueden dejar para más adelante.

2.2.4.2 Valores de Scrum para los equipos de trabajo de los proyectos

a. Compromiso

Los miembros del equipo Scrum están dedicados a cumplir con las tareas y objetivos definidos dentro de un marco temporal, así como a la mejora continua para encontrar la mejor solución posible.

b. Valentía

Los equipos Scrum muestran valentía al plantear preguntas abiertas y desafiantes. Mantienen conversaciones honestas y transparentes para descubrir la mejor solución.

c. Concentración

Durante un período específico, los miembros del equipo se enfocan en las tareas del Product Backlog, dedicándose a las tareas seleccionadas para generar resultados dentro de un tiempo limitado.

d. Actitud receptiva

Los integrantes del equipo Scrum son receptivos a nuevas ideas y oportunidades que fomentan el aprendizaje individual y mejoran la calidad general del proyecto.

e. Respeto

Los miembros del equipo demuestran respeto hacia los directores del proyecto, entre ellos mismos y hacia el proceso Scrum. Esta cultura de respeto promueve un ambiente de colaboración y cooperación mutua dentro del equipo de trabajo.

2.2.4.3 Artefactos de Scrum

Los artefactos en Scrum simbolizan trabajo o valor, y su diseño está orientado a maximizar la transparencia de la información crucial. De este modo, todas las personas que los revisan cuentan con la misma base para realizar adaptaciones. (Schwaber & Sutherland, 2020)

En su blog acerca de “¿Qué es Scrum?” (Amazon Web Services, s.f.) respaldados en (Schwaber & Sutherland, 2020) , mencionan que Scrum posee tres artefactos principales con sus respectivos objetivos:

a. Product Backlog

El Product Backlog es una lista en constante evolución que incluye características, requisitos, mejoras y correcciones necesarias para garantizar el éxito del proyecto. Esta lista actúa como un registro de tareas para el equipo, el cual se revisa periódicamente para ajustar las prioridades y adaptarse a los cambios del mercado. El responsable de mantener y actualizar esta lista es el propietario del producto, quien se encarga de eliminar elementos que ya no son relevantes y de incluir nuevas solicitudes que realicen los clientes.

b. Sprint Backlog

El Sprint Backlog es la lista de elementos que el equipo de desarrollo debe completar durante el Sprint actual. Antes de iniciar cada Sprint, el equipo selecciona los elementos en los que trabajará a partir del Product Backlog. Esta lista es flexible y puede cambiar a medida que avanza el Sprint.

c. Increment

Un Incremento se refiere a un progreso hacia el logro de un objetivo o visión. Es el resultado final y utilizable que se genera a partir de un Sprint. Los equipos pueden emplear diversos métodos para definir y demostrar los objetivos del Sprint. No obstante, a pesar de la flexibilidad en el proceso, el objetivo fundamental del Sprint, es decir, lo que el equipo espera alcanzar en el Sprint en curso, debe mantenerse intacto (Segun, como indica, etc.).

2.2.4.4 Scrum Team

El Scrum Team es la unidad esencial dentro de Scrum, compuesta por un grupo reducido de personas. Este equipo incluye a un Scrum Master, un Product Owner y desarrolladores. En un Scrum Team, no existen subequipos ni jerarquías, lo que permite que sea una unidad cohesiva de profesionales centrados en un único objetivo a la vez. (Schwaber & Sutherland, 2020)

Los miembros del Scrum Team son multifuncionales, lo que significa que cada uno cuenta con las habilidades necesarias para aportar valor en cada Sprint. Además, se autogestionan, lo que les da la capacidad de decidir internamente quién se encargará de qué tarea, así como cuándo y cómo se llevará a cabo.

El Scrum Team tiene la responsabilidad de todas las actividades relacionadas con el producto, incluyendo la colaboración con los interesados, la verificación, el mantenimiento, la operación, la experimentación, la investigación y el desarrollo, así como cualquier otra actividad que sea necesaria. Están organizados y cuentan con el empoderamiento necesario por parte de la organización para gestionar su propio trabajo. (Schwaber & Sutherland, 2020)

Como se menciona en el Blog de AWS acerca de “¿Qué es scrum?” (Amazon Web Services, s.f.),

un equipo de trabajo de Scrum necesita tres roles específicos: un *Product Owner* (propietario de producto), un *Scrum Master* (líder de Scrum) y un *Developers* (equipo de desarrollo).

En (Schwaber & Sutherland, 2020), describen cada uno de los roles de la siguiente manera:

a. Product Owner

El Product Owner tiene la responsabilidad de maximizar el valor del producto que resulta del trabajo del Scrum Team. También se encarga de gestionar eficazmente el Product Backlog, lo que implica:

- Definir y comunicar claramente el Objetivo del Producto.
- Elaborar y dar a conocer los elementos del Product Backlog.
- Priorizar los elementos del Product Backlog.
- Asegurar que el Product Backlog sea transparente, visible y comprensible.

El Product Owner puede llevar a cabo estas tareas personalmente o delegarlas en otros; sin embargo, sigue siendo el responsable del trabajo realizado. Es importante destacar que el Product Owner es una persona individual, no un comité, y tiene la capacidad de representar las necesidades de diversos interesados en el Product Backlog. Aquellos que deseen modificar el Product Backlog deben intentar convencer al Product Owner

b. Scrum Master

El Scrum Master es el encargado de garantizar la efectividad del Scrum Team, apoyándolo en la mejora de sus prácticas dentro del marco de trabajo de Scrum.

El Scrum Master contribuye al Scrum Team de diversas maneras, tales como:

- Orientar a los miembros del equipo para que sean autogestionados y multifuncionales.
- Ayudar al Scrum Team a centrarse en generar incrementos de alto valor que cumplan con lo acordado.

- Facilitar la eliminación de obstáculos que impidan el progreso del Scrum Team.

- Asegurarse de que todos los eventos de Scrum se realicen de manera positiva, productiva y dentro de los límites de tiempo recomendados en la guía.

El Scrum Master también apoya al Product Owner de varias formas, incluyendo:

- Asistir en la búsqueda de técnicas para definir efectivamente los Objetivos del Producto y gestionar el Product Backlog.

- Ayudar al Scrum Team a comprender la importancia de tener elementos del Product Backlog claros y concisos.

- Facilitar la planificación empírica de productos en un entorno complejo.

Fomentar la colaboración de los interesados según se requiera.

Asimismo, el Scrum Master brinda apoyo a la organización de diferentes maneras, tales como:

- Liderar, capacitar y guiar a la organización en la adopción de Scrum.

- Planificar y asesorar sobre la implementación de Scrum en la organización.

- Ayudar a empleados y interesados a entender y aplicar un enfoque empírico para abordar trabajos complejos.

- Eliminar barreras entre los interesados y los Scrum Teams.

c. Developers

Los Developers son las personas del Scrum Team que se comprometen a crear cualquier aspecto de un incremento utilizable en cada Sprint. Las habilidades específicas que requieren los Developers suelen ser diversas y varían según el ámbito de trabajo. No obstante, siempre son responsables de:

- Crear un plan para el Sprint, conocido como Sprint Backlog.

- Incorporar calidad al cumplir con una Definición de Terminado.

- Adaptar su plan diariamente para alinearse con el Objetivo del Sprint.

- Fomentar la responsabilidad mutua como profesionales.

2.2.4.5 Eventos de Scrum

El Sprint actúa como un contenedor para los demás eventos dentro de Scrum. Cada uno de estos eventos proporciona una ocasión formal para inspeccionar y adaptar los artefactos de Scrum. Están diseñados con el fin de facilitar la transparencia necesaria en el proceso. No seguir el formato establecido para cualquier evento puede llevar a la pérdida de oportunidades para realizar inspecciones y adaptaciones. En Scrum, los eventos se utilizan para establecer regularidad y reducir la necesidad de reuniones que no estén definidas dentro del marco de Scrum. (Schwaber & Sutherland, 2020)

En la guía que nos proponen Schwaber & Sutherland, (Schwaber & Sutherland, 2020) describen cada uno de los eventos de Scrum de la siguiente manera.

a. Sprint

Los Sprints son el corazón de Scrum, son eventos de duración fija de un mes o menos para crear consistencia. Un nuevo Sprint comienza inmediatamente después de la conclusión del Sprint anterior.

Durante el Sprint:

- No se realizan cambios que pongan en peligro el Objetivo del Sprint

- La calidad no disminuye

- El *Product Backlog* se refina según sea necesario

- El alcance se puede aclarar y renegociar con el *Product Owner* a medida que se aprende más

Los Sprints ofrecen previsibilidad al permitir la inspección y adaptación del progreso hacia un Objetivo del Producto al menos una vez al mes. Según se menciona, cuando el periodo de un Sprint es excesivamente largo, el Objetivo del Sprint puede volverse irrelevante, incrementando la complejidad y el riesgo. Implementar Sprints más breves puede facilitar ciclos de aprendizaje más frecuentes y restringir el riesgo de costos y esfuerzos a intervalos de tiempo más cortos. Cada Sprint puede ser visto como un pequeño proyecto.

Existen diversas técnicas para prever el progreso, como los gráficos de trabajo pendiente (burn-down), de trabajo completado (burn-up) o flujos acumulativos (cumulative flow). Aunque estas prácticas han demostrado ser efectivas, no reemplazan la importancia del enfoque empírico. En entornos complejos, el futuro es incierto y solo se puede utilizar la información del pasado para guiar las decisiones hacia adelante.

Un Sprint puede ser cancelado si el Objetivo del Sprint pierde su relevancia. Solo el Product Owner tiene la autoridad para decidir la cancelación del Sprint.

Sprint Planning

La *Sprint Planning* inicia el *Sprint* al establecer el trabajo que se realizará para el *Sprint*. El *Scrum Team* crea este plan resultante mediante trabajo colaborativo.

El *Product Owner* se asegura de que los asistentes estén preparados para discutir los elementos más importantes del *Product Backlog* y cómo se relacionan con el Objetivo del Producto. El *Scrum Team* también puede invitar a otras personas a asistir a la *Sprint Planning* para brindar asesoramiento.

La *Sprint Planning* aborda los siguientes temas:

- Tema uno: ¿Por qué es valioso este *Sprint*?
  El *Product Owne*r propone cómo el producto podría Incrementar su valor y utilidad en el *Sprint* actual. Luego, todo el *Scrum Team* colabora para definir un Objetivo del Sprint que comunica por qué el *Sprint* es valioso para los interesados. El Objetivo del Sprint debe completarse antes de que termine la *Sprint Planning*.

- Tema dos: ¿Qué se puede hacer en este Sprint?
  A través de una conversación con el *Product Owner*, los *Developers* seleccionan elementos del *Product Backlog* para incluirlos en el Sprint actual. El *Scrum Team* puede refinar estos elementos durante este proceso, lo que aumenta la comprensión y la confianza.

- Tema tres: ¿Cómo se realizará el trabajo elegido?
  Para cada elemento del *Product Backlog* seleccionado, los *Developers* planifican el trabajo necesario para crear un *Increment* que cumpla con la definición de terminado. A menudo, esto se hace descomponiendo los elementos del *Product Backlog* en elementos de trabajo más pequeños de un día o menos. La forma de hacerlo queda a criterio exclusivo de los *Developers*. La *Sprint Planning* tiene un límite de tiempo de máximo ocho horas para un *Sprint* de un mes. Para *Sprints* más cortos, el evento suele ser de menor duración.

Daily Scrum

El propósito de la *Daily Scrum* es inspeccionar el progreso hacia el Objetivo del *Sprint* y adaptar el *Sprint Backlog* según sea necesario, ajustando el trabajo planificado entrante.

La *Daily Scrum* es un evento de 15 minutos para los *Developers* del *Scrum Team*. Para reducir la complejidad, se lleva a cabo a la misma hora y en el mismo lugar todos los días hábiles del *Sprint*. Si el *Product Owner* o *Scrum Master* están trabajando activamente en elementos del *Sprint Backlog*, participan como *Developers*.

Las *Daily Scrums* mejoran la comunicación, identifican impedimentos, promueven la toma rápida de decisiones y, en consecuencia, eliminan la necesidad de otras reuniones.

La *Daily Scrum* no es el único momento en el que los *Developers* pueden ajustar su plan. A menudo se reúnen durante el día para discusiones más detalladas sobre cómo adaptar o volver a planificar el resto del trabajo del *Sprint*.

Sprint Review

El objetivo de la Sprint Review es evaluar los resultados del Sprint y decidir sobre futuras adaptaciones. Durante este evento, el Scrum Team presenta los resultados de su trabajo a los interesados clave y se analiza el avance hacia el Objetivo del Producto.

En este contexto, el Scrum Team y los interesados revisan lo que se logró durante el Sprint y cualquier cambio que haya ocurrido en su entorno. A partir de esta información, los participantes colaboran para definir los próximos pasos a seguir. Además, se puede realizar un ajuste en el Product Backlog para aprovechar nuevas oportunidades.

La Sprint Review es el penúltimo evento dentro del Sprint y tiene un límite de tiempo de hasta cuatro horas para un Sprint de un mes. Para Sprints más breves, la duración del evento suele ser menor.

Sprint Retrospective

El objetivo de la Sprint Retrospective es planificar maneras de mejorar tanto la calidad como la efectividad del trabajo. Durante esta reunión, el Scrum Team evalúa el último Sprint en términos de personas, interacciones, procesos, herramientas y su definición de "terminado". Los aspectos a evaluar pueden variar según el tipo de trabajo realizado. Además, se identifican los supuestos que llevaron a errores y se investigan sus causas. El equipo analiza qué aspectos fueron positivos durante el Sprint, así como los problemas que se encontraron y cómo se abordaron (o no) esos inconvenientes.

El Scrum Team determina cuáles cambios pueden ser más beneficiosos para incrementar su efectividad. Las mejoras más significativas se priorizan y se abordan lo antes posible, incluso pudiendo ser añadidas al Sprint Backlog para el próximo Sprint.

La Sprint Retrospective marca el cierre del Sprint y tiene un tiempo máximo de tres horas para un Sprint de un mes. Para Sprints más breves, la duración de este evento suele ser menor.

2.2.5 Método MoSCoW

El método MoSCoW, creado por Dai Clegg, es una técnica utilizada para establecer la priorización en proyectos que enfrentan limitaciones de tiempo. Es común en la metodología Agile, aunque

también puede ser aplicada fuera de ella. La esencia de este método radica en dar mayor peso y prioridad a las características de un producto que aportan mayor valor comercial. (Simões, 2020)

En otras palabras, las funcionalidades del software o sitio web que se consideran más valiosas tienen una mayor probabilidad de ser implementadas.

2.2.5.1 Características

El conjunto de criterios utilizado para evaluar la importancia de las ideas o iniciativas del producto en el método MoSCoW es bastante fácil de entender. Su objetivo es ayudar a los equipos a priorizar de manera estratégica y organizada. Estos criterios se dividen en cuatro categorías. (Simões, 2020)

a. Must have (Debe tener)

Estas características serán las absolutamente críticas para el proyecto, y sin ellas el proyecto es un fracaso.

b. Should have (Debería incluir)

Estos aspectos del proyecto son críticos también, pero no imprescindibles.

c. Could have (Podría incluir)

Estas iniciativas son las que estaría bien tenerlas, ya que añadirían valor al proyecto, pero no son críticas.

d. Won’t have (No se van a hacer)

Estas características no se merecen la inversión y no aportan ningún beneficio en este momento y se podrían considerar más tarde.

2.2.5.2 Proceso de ejecución

Según (Simões, 2020), se recomienda seguir los siguientes pasos inspirados en Lucidchart para usar la técnica de MoSCoW:

a. Identificar a los stakeholders clave

Es fundamental tener en cuenta que, aunque todas las personas en la organización son valiosas, no es conveniente que todas influyan en cada una de las características del producto. Por esta razón, es importante identificar y reunir a los stakeholders clave y al equipo responsable del desarrollo.

b. Establecer el arbitraje

Es importante decidir cómo se resolverán las diferencias antes de empezar el MoSCoW. Siempre se puede utilizar votaciones, o definir el valor de negocio de cada una de las características, y seleccionar la característica con más valor.

c. Dividir los recursos

La estrategia consiste en destinar la mayor parte de los recursos económicos a las iniciativas más críticas, lo cual debe ser facilitado por el equipo. En esta fase, las iniciativas clasificadas como "will not have" no recibirán recursos.

d. Consolidar los plazos

Al restringir los plazos en la categoría de "must have", se reduce la cantidad de ideas e iniciativas, lo que ayuda a prevenir problemas futuros al iniciar el desarrollo. Esta práctica contribuye a mantener un enfoque realista.

e. Realizar lista

Elaborar un listado de todas las características, iniciativas y requisitos del proyecto resulta crucial, ya que proporciona una visión general de cómo se están priorizando las tareas y si se está realizando de manera completa y ordenada. Además, esta lista estará accesible para otros miembros de la organización, lo que fomenta la transparencia.

f. Empezar a priorizar

Para decidir la mejor categoría para cada una de las iniciativas o características, el equipo debe hacerse preguntas clave como las siguientes:

Must have

- ¿Qué pasará si lanzamos el producto sin esta característica?
- ¿Hay alguna forma más simple de completar esto?
- ¿La aplicación funcionará sin esta característica?

Should have

- ¿Cuánto valor añade esta característica al proyecto?
- ¿Esto puede esperar hasta el próximo lanzamiento?

Could have

- ¿Es esto necesario para la función básica del proyecto?
- ¿Cuánto impacto tendrá esto en el proyecto si se deja de lado?

Will not have

- ¿Esto tiene cero o poca importancia para el proyecto en este momento?
- ¿Hay alguna posibilidad de que esto tenga prioridad en el futuro?

A partir de este momento, con las características e iniciativas divididas en categorías de priorización, el equipo tendrá una idea clara de dónde invertir y centrarse.

2.2.6 Metodología UML-based Web Engineering

Como describe (Gonzales, 2020), la metodología UWE es un enfoque de desarrollo para aplicaciones web que se centra en el diseño sistemático, la personalización y la generación semiautomática de escenarios que orienten el proceso de desarrollo. UWE se basa en las técnicas y notación del UML, así como en sus mecanismos de extensión, y es ampliamente utilizada en la

ingeniería web, con un fuerte enfoque en la sistematización y la adaptación de sistemas. Este método está basado en el proceso unificado de UML, pero adaptado específicamente para la web. En la etapa de requisitos, UWE separa las fases de captura, definición y validación, otorgando un tratamiento especial según el tipo de requisito.

UWE requiere la definición de un perfil UML mediante estereotipos, lo que permite asociar una semántica distinta a los diagramas tradicionales de UML, adaptándolos al dominio de las aplicaciones web. Entre los modelos principales que UWE maneja se encuentran el modelo de requerimientos, lógico-conceptual, de usuario, de navegación, de presentación y de adaptación. UWE define vistas gráficas especiales mediante diagramas UML y, aunque no limita la cantidad de vistas posibles, utiliza los mecanismos de extensión de UML para definir estereotipos específicos para el modelado de aplicaciones web. Esta notación se conoce como Perfil UML.

Además, UWE se especializa en la especificación de aplicaciones adaptativas, poniendo especial énfasis en la personalización, como la definición de un modelo de usuario y características adaptativas para la navegación según las preferencias, conocimientos o tareas del usuario.

2.2.6.1 Actividades de modelado en UWE

Como se menciona en (Gonzales, 2020), UWE propone diversas actividades clave para el modelado de aplicaciones web, tales como: análisis de requerimientos, modelo conceptual, modelo de navegación y modelo de presentación. Adicionalmente, es posible incluir otros modelos, como los de interacción y visualización de escenarios web. El proceso de modelado en UWE está compuesto por distintas etapas o submodelos, entre los que se encuentran: modelo de casos de uso, de contenido, de usuario, de estructura, abstracto, de adaptación, de flujo de presentación y de ciclo de vida del objeto.

a. Modelo lógico-conceptual

El objetivo de UWE es crear un modelo conceptual para una aplicación web, que no se enfoque en la navegación ni en la interacción, sino en los objetos implicados en las actividades que realizarán los usuarios. Este modelo debe construirse en función de los casos de uso definidos en la fase de requisitos.

b. Modelo de navegación

Consiste en dos modelos: el modelo del espacio de navegación, que especifica qué objetos serán accesibles en la aplicación, y el modelo de estructura de navegación, que define cómo se relacionarán entre sí dichos objetos.

c. Modelo de presentación

Este modelo describe de manera esquemática cómo y dónde se presentarán los objetos de navegación y los accesos primitivos al usuario, representando lo que el usuario verá.

d. Interacción temporal

Se refiere a la interacción entre los objetos involucrados y la secuencia de mensajes que intercambian.

e. Escenarios web

Detallan la parte dinámica del modelo de navegación, especificando los eventos que desencadenan situaciones y las acciones que se ejecutarán, proporcionando una representación funcional dinámica en conjunto con el modelo de interacción temporal.

2.2.6.2 Fases UWE

Según (Gonzales, 2020), La metodología UWE abarca todo el ciclo de vida de aplicaciones web, prestando especial atención a aquellas que son personalizadas o adaptativas. Las etapas del proceso son las siguientes:

a. Captura, análisis y especificación de requisitos

En esta fase, se recopilan y especifican las características tanto funcionales como no funcionales que la aplicación web debe cumplir. Se abordan diversas necesidades, como las de información, navegación, adaptación e interfaz de usuario, así como requisitos adicionales. El enfoque principal está en el estudio de los casos de uso, la creación de glosarios y el prototipado de la interfaz de usuario.

b. Diseño del sistema

Basándose en los requisitos especificados durante la fase de análisis, el diseño del sistema establece cómo se cumplirán estos requisitos y define la estructura necesaria para la aplicación web.

c. Codificación del software

En esta etapa, se llevan a cabo las tareas de programación, que consisten en traducir a código fuente, utilizando el lenguaje de programación seleccionado, todo lo que se diseñó en la fase anterior.

d. Pruebas

Esta fase se utiliza para verificar el correcto funcionamiento de las distintas secciones del código.

e. Instalación o Fase de Implementación

Este proceso implica transferir los programas desarrollados al ordenador de destino, donde se inicializan y configuran para su uso por el usuario final. Esto incluye la implementación de la arquitectura, la estructura del hiperespacio, el modelo de usuario, la interfaz de usuario, los mecanismos adaptativos y la integración de todas estas implementaciones.

f. Mantenimiento

Se refiere al proceso de control, mejora y optimización del software ya desarrollado e instalado, incluyendo la depuración de errores y defectos que puedan haberse pasado por alto en la fase de pruebas. Los problemas no detectados durante el desarrollo pueden ser identificados a través de diferentes métodos, principalmente a través de mensajes, opiniones de los usuarios y el comportamiento en el uso del sitio.

2.2.6.3 Ventajas de UWE

La especificación de UWE para aplicaciones adaptativas proporciona al usuario páginas más adecuadas, ya que estas se describen en función de sus preferencias o características contextuales, basándose en técnicas orientadas a aspectos. UWE permite el modelado de aplicaciones web que

responde a las necesidades particulares de cada usuario, separando requerimientos, enfoques, interfaces, adaptabilidad, aspectos y componentes para mayor flexibilidad. Esta metodología define un conjunto de procesos adecuados para todas las etapas del desarrollo, lo que ayuda a mantener la integridad del diseño y la funcionalidad del sistema. (Gonzales, 2020)

2.2.7 Arquitectura de software

Según (Wikipedia), la arquitectura de software, según Garlan y Shaw en "An Introduction to Software Architecture", se define como un nivel de diseño que se centra en la estructura general del sistema y considera elementos que trascienden los algoritmos y las estructuras de datos. Al igual que los planos de un edificio, la arquitectura de software determina la estructura, el funcionamiento y la interacción entre los componentes del sistema, ofreciendo un marco útil para abordar los desafíos complejos que surgen en el desarrollo de software contemporáneo.

Así como señala (Gonzales, 2020), la arquitectura de software, a menudo referida como arquitectura lógica, se define como un conjunto de patrones y abstracciones coherentes que establecen un marco claro para interactuar con el código fuente del software. Entre sus características se destacan:

- La selección y diseño de una arquitectura de software se basa en objetivos (requisitos) y restricciones. Los objetivos incluyen no solo los requisitos funcionales del sistema de información, sino también aspectos como mantenibilidad, auditabilidad, flexibilidad e integración con otros sistemas. Por otro lado, las restricciones son limitaciones derivadas de las tecnologías disponibles para la implementación de sistemas de información. Algunas arquitecturas son más adecuadas para ciertas tecnologías, mientras que otras no son compatibles con determinadas arquitecturas.

- De manera abstracta, la arquitectura de software define los componentes encargados de realizar tareas de computación, sus interfaces y la comunicación entre ellos. Es fundamental que toda arquitectura sea implementable en una arquitectura física, lo que implica determinar qué computadora se asignará a cada tarea específica.

2.2.7.1 Descomposición modular

La descomposición modular, o modularización, es el proceso de dividir un sistema en un conjunto de elementos que tienen un bajo acoplamiento (es decir, que son independientes entre sí) y una alta cohesión (lo que significa que cada módulo tiene un significado claro y específico). (ittgweb, 2016)

Como señala (ittgweb, 2016), este enfoque implica descomponer el problema a resolver en módulos o tareas más simples. Cada tarea representa una actividad completa que se puede codificar de manera independiente. Esto facilita un diseño descendente del problema, enfocándose en la resolución de subproblemas de menor envergadura.

El proceso de abordar cada uno de estos subproblemas de menor complejidad se conoce como refinamiento por pasos. Los módulos pueden ser planificados, codificados, probados y depurados de manera independiente, para luego ser combinados uno a uno con otros módulos.

El diseño modular sugiere dividir el sistema en partes distintas y definir sus interfaces correspondientes.

Según (ittgweb, 2016), la descomposición modular ofrece las siguientes ventajas:

- Claridad
- Reducción de costos
- Reutilización

ittweb recomienda seguir ciertos pasos para llevar a cabo la descomposición modular, que incluyen:

- Identificar los módulos
- Describir cada módulo
- Describir las relaciones entre módulos

Asimismo, indica que para que una descomposición modular se considere válida, debe poseer ciertas características mínimas.

a. Independiente Funcional

- Al final de los documentos ADD y DDD, debe incluirse una matriz de REQUISITOS/COMPONENTES. Inicialmente, cada función debe ser ejecutada en un módulo diferente. Si las funciones son independientes, se logra independencia funcional entre los módulos.

- Cada módulo debe cumplir una función específica o un conjunto de funciones relacionadas, siendo recomendable reducir al mínimo las relaciones entre módulos.

- Para evaluar la independencia funcional, se utilizan dos criterios: acoplamiento y cohesión.

b. Acoplamiento

El acoplamiento mide el grado de interrelación entre dos módulos, basado en el tipo de conexión y la complejidad de la interfaz:

Fuerte:

- Por contenido: Un módulo puede modificar datos locales de otro.
- Común: Varios módulos acceden a una zona común de datos

Moderado:

- De control: La zona común es un dispositivo externo al que están ligados los módulos, esto implica que un cambio en el formato de datos afecta a todos estos módulos

- Por etiqueta: En intercambio de datos se realiza mediante una referencia a la estructura completa de datos (vector, pila, árbol, grafo, …)

Débil:

- De datos: Viene dado por los datos que intercambian los módulos. Es el mejor posible

- Sin acoplamiento directo: Es el acoplamiento que no existe

c. Cohesión

Es fundamental que el contenido de cada módulo sea lo más coherente posible. Para evitar un número excesivo de módulos que compliquen el diseño, se agrupan elementos relacionados en un mismo módulo.

Alta:

- Cohesión abstraccional: Se logra cuando se diseña el módulo como tipo abstracto de datos o como una clase de objetos.

- Cohesión funcional: El módulo realiza una función concreta y específica.

Media:

- Cohesión secuencial: Los elementos del módulo trabajan de forma secuencial.

- Cohesión de comunicación: Elementos que operan con el mismo conjunto de datos de entrada o de salida.

- Cohesión temporal: Se agrupan elementos que se ejecutan en el mismo momento, por ejemplo, arrancar o parar dispositivos.

Baja:

- Cohesión lógica: Se agrupan elementos que realizan funciones similares, como módulos de E/S o de tratamiento de errores.

- Cohesión coincidental: Es la peor y se produce cuando los elementos de un módulo no guardan relación alguna.

La descripción del comportamiento de un módulo permite establecer el grado de cohesión:

- Si es una frase compuesta y contiene más de un verbo la cohesión será MEDIA

- Si contiene expresiones secuenciales (primero, entonces, cuando…), será temporal o secuencial

- Si la descripción no se refiere a algo específico (Ej. Todos los errores), cohesión lógica

- Si aparece “inicializar”, “preparar”, “configurar”, probablemente sea temporal.

d. Comprensibilidad

Para facilitar cambios, mantenimiento y reutilización de módulos, es esencial que cada uno sea comprensible de forma independiente. Para lograrlo, es deseable tener:

- Identificación: El nombre debe ser adecuado y descriptivo

- Documentación: Debe aclarar todos los detalles de diseño e implementación que no queden de manifiesto en el propio código

- Simplicidad: Las soluciones sencillas son siempre las mejores

e. Adaptabilidad

La adaptación de un sistema se complica con alta dependencia funcional, es decir, con elevado acoplamiento y baja cohesión, así como un diseño poco comprensible. Otros factores que facilitan la adaptabilidad son:

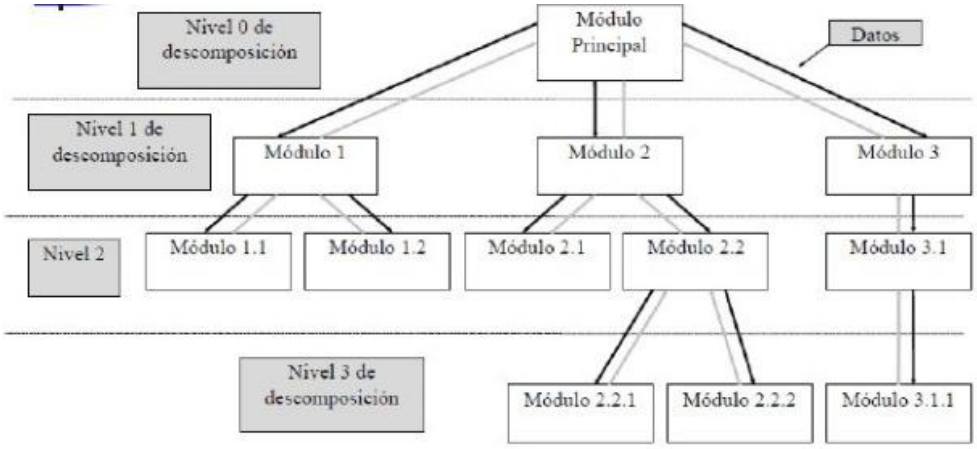
- Previsión: Es necesario prever que aspectos del sistema pueden ser susceptibles de cambios en el futuro, y poner estos elementos en módulos independientes, de manera que su modificación afecte al menor número de módulos posible

- Accesibilidad: Debe resultar sencillo el acceso a los documentos de especificación, diseño, e implementación para obtener un conocimiento suficiente del sistema antes de proceder a su adaptación

- Consistencia: Después de cualquier adaptación se debe mantener la consistencia del sistema, incluidos los documentos afectados

Ilustración 1: *Descomposición modular.*

Fuente: (ittgweb, 2016)



2.2.7.2 Cliente-servidor

Como se señala en (Wikipedia), la arquitectura cliente-servidor es un modelo de diseño de software que distribuye tareas entre los proveedores de servicios o recursos, conocidos como servidores, y los solicitantes, llamados clientes. Un cliente envía peticiones a un programa diferente, el servidor, que responde a estas solicitudes. Este concepto también se aplica a programas que funcionan en una sola computadora, aunque es más eficaz en un sistema operativo multiusuario que opera a través de una red de computadoras. Ejemplos de aplicaciones que utilizan el modelo cliente/servidor incluyen el correo electrónico, los servidores de impresión y la World Wide Web.

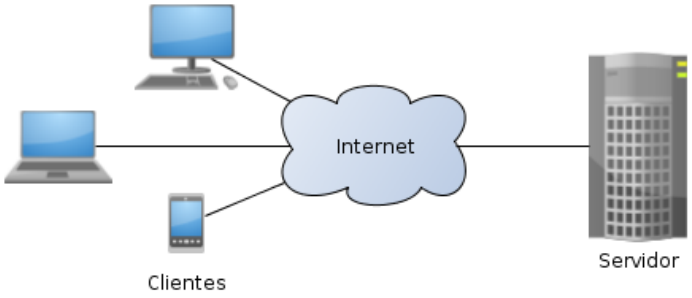
En esta arquitectura, la capacidad de procesamiento se reparte entre los clientes y los servidores. No obstante, las principales ventajas provienen de la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y aclara el diseño del sistema.

La separación entre cliente y servidor es de naturaleza lógica; es decir, el servidor no tiene que estar limitado a una única máquina ni a un solo programa. Hay varios tipos de servidores, como los servidores web, de archivos y de correo, cada uno con un propósito específico, pero todos mantienen la misma arquitectura básica.

Una configuración común en este modelo es la de sistemas multicapa, en la que el servidor se divide en varios programas que pueden ejecutarse en diferentes máquinas, aumentando así el nivel de distribución del sistema.

En una red cliente-servidor, los clientes se conectan a un servidor que centraliza los recursos y las aplicaciones, haciéndolos accesibles a los clientes cuando los requieren. Esto implica que todas las operaciones se gestionan en el servidor, donde se organizan las prioridades de las solicitudes de los clientes, se controla el acceso a archivos públicos o restringidos y se gestionan los permisos de lectura o modificación. Este tipo de red puede funcionar en conjunto con una red mixta si es necesario.

***Ilustración 2**: Cliente - Servidor.*
Fuente: Tiago de Jesus Neves publicado en (Wikipedia)



Como se indica en (Wikipedia), en la arquitectura cliente-servidor (C/S), el cliente es quien envía solicitudes. Sus características incluyen:

- Iniciar solicitudes, lo que le otorga un rol activo en la comunicación.

- Esperar y recibir respuestas del servidor.

- Tener la capacidad de conectarse a múltiples servidores simultáneamente.

- Interactuar directamente con los usuarios a través de interfaces gráficas.

Por otro lado, el servidor es quien recibe las solicitudes del cliente. Sus características incluyen:

- Permanecer inactivo hasta recibir una solicitud, desempeñando un rol pasivo en la comunicación.

- Procesar las solicitudes y enviar respuestas de vuelta al cliente.

- Aceptar conexiones de múltiples clientes, aunque en algunos casos puede haber un límite en el número de solicitudes que puede manejar.

- En general, la arquitectura C/S tiene las siguientes características:

El cliente y el servidor pueden actuar de forma conjunta o como entidades separadas, realizando tareas independientes.

- Las funciones de cliente y servidor pueden estar en plataformas diferentes o en la misma.

- Ambas plataformas son escalables de manera independiente, permitiendo actualizaciones o cambios sin afectar al usuario final.

- La interacción entre hardware y software se basa en una infraestructura robusta que oculta la complejidad de los diferentes formatos de datos y protocolos de red.

- Típicamente, el cliente es un dispositivo como un PC donde el usuario gestiona sus propias aplicaciones y bases de datos sin depender del sistema central de la organización.

- Los servidores pueden ser apátridas (no retienen información entre solicitudes) o con estado (mantienen información entre solicitudes, como Apache Tomcat).

Además, la interacción entre cliente y servidor se describe mediante diagramas de secuencia, estandarizados en UML. Es importante que los clientes no se comuniquen directamente entre sí, sino que todas las interacciones pasen a través del servidor.

2.2.8 Seguridad de software

Como indica (Gonzales, 2020), Gary McGraw, experto en seguridad, define la seguridad del software como un concepto que está completamente relacionado con la calidad. Según él, es esencial considerar aspectos como la seguridad, confiabilidad, disponibilidad y dependencia desde las fases iniciales del desarrollo, incluyendo el diseño, la arquitectura, las pruebas y la codificación, a lo largo de todo el ciclo de vida del software. En resumen, el software de baja calidad es susceptible a ataques por parte de intrusos, lo que incrementa indirectamente el riesgo de seguridad y conlleva una serie de costos y problemas.

McGraw también sugiere que las raíces de los problemas de seguridad se encuentran en la conectividad, la complejidad y la extensibilidad de los sistemas modernos. Su análisis se centra en dos conceptos clave dentro de los objetivos de la Ingeniería de Software:

- La seguridad de un producto terminado se enfoca en asegurar que dicho producto siga funcionando correctamente frente a ataques maliciosos.
- La seguridad del software en desarrollo se dirige hacia la resistencia proactiva ante posibles ataques.

2.2.9 Pruebas de software

Las pruebas de software implican el proceso de evaluar y validar que un producto o aplicación de software cumple con las funciones para las cuales fue diseñado. Según (según indica), algunas de las ventajas de realizar pruebas efectivas incluyen la detección de errores y la optimización del rendimiento. (IBM)

2.2.9.1 Tipos de pruebas

Existen muchos tipos diferentes de pruebas de software, cada uno con objetivos y estrategias específicos:

- Pruebas de aceptación: Verifican si todo el sistema funciona según lo previsto.

- Revisión del código: Confirman que el software nuevo y modificado esté siguiendo los estándares de codificación de una organización y se adhiera a sus mejores prácticas.

- Pruebas de integración: Garantizan que los componentes o las funciones del software funcionan conjuntamente.

- Unit tests: Validan que cada unidad de software funciona según lo esperado. Una unidad es el componente comprobable más pequeño de una aplicación.

- Pruebas funcionales: Comprueban las funciones emulando escenarios de negocio, basándose en los requisitos funcionales. Las pruebas de caja negra son una forma habitual de verificar las funciones.

- Pruebas de rendimiento: Comprueban cómo funciona el software bajo diferentes cargas de trabajo. Las pruebas de carga, por ejemplo, se utilizan para evaluar el rendimiento en condiciones de carga reales.

- Pruebas de regresión: Comprueban si las nuevas características rompen o degradan la funcionalidad. Las pruebas de cordura pueden utilizarse para verificar menús, funciones y comandos a nivel superficial, cuando no hay tiempo para una prueba de regresión completa.

- Pruebas de seguridad: Validando que su software no está expuesto a ataques de hackers u otro tipo de vulnerabilidades maliciosas que puedan ser explotadas para denegar el acceso a sus servicios o hacer que funcionen incorrectamente.

- Pruebas de estrés: Prueban cuánta tensión puede soportar el sistema antes de fallar. Las pruebas de estrés se consideran un tipo de pruebas no funcionales.

- Pruebas de usabilidad: Validan lo bien que un cliente puede utilizar un sistema o una aplicación web para completar una tarea.

La validación de los requisitos fundamentales es un aspecto esencial en todos los casos. De igual manera, las pruebas exploratorias son valiosas, ya que permiten al evaluador o al equipo de pruebas identificar escenarios y situaciones que pueden ser difíciles de anticipar, lo que podría resultar en errores en el software.

Incluso aplicaciones que parecen simples pueden requerir una amplia gama de pruebas. Un plan de gestión de pruebas es fundamental para priorizar los tipos de pruebas que ofrecen mayor valor, considerando el tiempo y los recursos disponibles. Según (autor), la eficacia de las pruebas se maximiza al realizar la menor cantidad de pruebas posible para detectar el mayor número de defectos.

2.2.9.2 Técnicas de pruebas

Para alcanzar la meta de asegurar que el producto cumpla con los estándares de calidad requeridos, se explorarán diversas técnicas de prueba que pueden aplicarse durante el proceso de evaluación. Estas técnicas buscan identificar las condiciones de prueba, los casos de prueba y los datos necesarios para las pruebas. (Gonzales, 2020)

Se reconocen tres categorías principales de pruebas:

- Técnicas estáticas. Estas técnicas no implican la ejecución de la aplicación y se aplican a nivel de especificaciones. Aunque no ejecutan código, realizan un análisis estático del mismo.

- Técnicas dinámicas. Estas técnicas se llevan a cabo mediante la ejecución de la aplicación y se utilizan para diseñar los casos de prueba.

El software se puede probar de dos maneras diferentes. Al conocer su funcionamiento interno, es posible verificar que todos los módulos funcionan correctamente entre sí, lo que se denomina pruebas de caja blanca. Por otro lado, al conocer las funciones específicas del producto, se pueden realizar pruebas que demuestren que estas funciones operan adecuadamente y se busca detectar errores en ellas. Estas pruebas se realizan desde una perspectiva externa, conocidas como pruebas de caja negra.

- Técnicas basadas en la experiencia. Según el ISTQB, estas técnicas se definen como aquellas en las que las pruebas se derivan de la habilidad, intuición y experiencia del probador con aplicaciones y tecnologías similares.

2.2.9.3 Pruebas de caja negra

Las pruebas de caja negra, o black box testing, son una técnica que se utiliza para verificar las funcionalidades de un software o aplicación sin considerar su código interno, las rutas internas o los

detalles de su implementación. Esto implica que las pruebas se realizan sin conocimiento del funcionamiento interno del sistema, centrándose en las entradas y salidas del software, basándose en sus especificaciones y requisitos. (KeepCoding, 2024)

Por lo tanto, se puede afirmar que el propósito de las pruebas de caja negra está vinculado a la validación de los recursos funcionales del software o la aplicación que se desea evaluar.

Las pruebas de caja negra poseen diversas características y propiedades que facilitan su funcionamiento:

- Uso de herramientas y plataformas externas que permiten verificar el desempeño del sistema y asegurar que todo opere correctamente. En este contexto, es el usuario o cliente quien define las métricas que se utilizarán para llevar a cabo el análisis.

- Proporcionan un nivel de abstracción del código, enfocándose en evaluar el comportamiento del sistema establecido. Esto mejora la comunicación entre los distintos módulos o subsistemas del software.

- Identifican errores relacionados con la interfaz, el rendimiento, el inicio y la finalización, así como funciones que no se ejecutan correctamente o que, directamente, no se llevan a cabo en el sistema.

- No se requiere que el usuario tenga conocimientos técnicos o de programación para realizar pruebas de caja negra, ya que no es necesario conocer a fondo los detalles de las implementaciones del software.

a. Pruebas de tabla de decisión

Las tablas de decisión son una representación tabular que facilita la descripción y el análisis de las relaciones entre diversas condiciones (o causas) y acciones (o efectos) en el ámbito del software. Estas tablas son especialmente útiles para probar decisiones lógicas complejas que involucran múltiples variables, donde cada combinación de estas variables puede conducir a un conjunto distinto de acciones. (Trejo, 2024)

Implementar tablas de decisión en las pruebas de software implica varios pasos clave:

- Identificación de Condiciones y Acciones: Determina todas las posibles entradas y las acciones resultantes que el sistema podría tomar en respuesta.

- Construcción de la Tabla: Organiza las condiciones y acciones en un formato tabular, donde cada fila representa un caso de prueba con una combinación específica de condiciones y las acciones esperadas.

- Análisis de Cobertura: Utiliza la tabla para identificar y asegurar que todos los posibles escenarios estén cubiertos en tus casos de prueba, lo cual es crucial para la calidad del software.

Según el desarrollo de las tablas de decisión que realizo (Gonzales, 2020), nos presenta esta tabla:

Tabla 2: *Condiciones y Acciones*

Fuente: (Gonzales, 2020)

	Regla 1	**Regla 2**	**. . .**	**Regla N**
Condiciones				
Condición 1				
Condición 2				
. . .				
Condición N				
Acciones				
Acción 1				
Acción 2				
. . .				
Acción N				

Donde:

- Condiciones: Muestran las entradas.

- Acciones: Son los resultados que sucederán, basados en la combinación de condiciones.

- Reglas: Define un único conjunto de condiciones que terminan en un conjunto de acciones.

Pasos:

- Identificar CONDICIONES y los posibles valores.

- Identificar ACCIONES de salida

- Realizar las combinaciones “viables” de CONDICIONES

- Definir los valores ACCIONES de acuerdo con CONDICIONES

- Identificar cada columna como una regla diferente

- Diseñar los casos de prueba.

Casos de prueba:

- Cada columna, cada regla es un caso de prueba

- Elimine las reglas que son “imposibles”

¿Son reales?, ¿Cómo puedo generarlas?

- Combinar con otras técnicas como casos de equivalencia y valores límite.
- ¿Cuál de las reglas de la Tabla de Decisión puede eliminarse? ¿O unificarse?

2.3 ESTIMACIÓN DE COSTOS DE SOFTWARE

El proceso de estimación del costo de un producto de software consiste en una serie de técnicas y procedimientos que se emplean dentro de la organización para lograr una predicción confiable. Este es un proceso que se realiza de manera continua y debe ser utilizado y revisado a lo largo de todo el ciclo de vida del proyecto. Se puede dividir en los siguientes pasos:

- Estimación del tamaño.

- Estimación del costo y del esfuerzo.

- Estimación del cronograma.

- Estimación de los recursos computacionales necesarios.

- Evaluación de riesgos.

- Inspección y aprobación.

- Elaboración de informes de estimación.

2.3.1 Modelo COCOMO II

Según la definición de (Linkedin), COCOMO II, que significa Modelo de Costo Constructivo II, es un método en ingeniería de software diseñado para estimar el costo, esfuerzo y duración de proyectos de software. Este modelo fue creado por Barry Boehm y su equipo en la Universidad del Sur de California en los años 90, como una evolución del modelo COCOMO original de los años 80. COCOMO II se divide en tres submodelos: el modelo de composición de la aplicación, el modelo de diseño temprano y el modelo posterior a la arquitectura. Cada uno de estos submodelos corresponde a diferentes fases del ciclo de vida del software y emplea diferentes entradas y resultados.

Los objetivos principales que se tuvieron en cuenta para construir el modelo COCOMO II según indican (Gómez, C.López, Migani, & Otazú) fueron:

- Crear un modelo para estimar los costos y plazos de los proyectos de software, adaptable tanto a las prácticas de desarrollo de la década de los 90 como a las futuras.

- Desarrollar una base de datos de proyectos de software que permitiera la calibración continua del modelo, aumentando así la precisión de las estimaciones.

- Implementar una herramienta de software que apoyara el uso del modelo.

- Ofrecer un marco analítico cuantitativo, junto con un conjunto de herramientas y técnicas, para evaluar el impacto de las mejoras tecnológicas en software sobre los costos y plazos en las diferentes fases del ciclo de vida del desarrollo.

COCOMO II consta de tres modelos principales: Composición de Aplicación, Diseño Temprano y Post-Arquitectura. Estos modelos fueron creados para responder a la diversidad presente y futura

del mercado de desarrollo de software, representada a través de un esquema que refleja dicha variedad.

2.3.1.1 Estimación de esfuerzo

El esfuerzo requerido para llevar a cabo un proyecto de desarrollo de software, independientemente del modelo utilizado, se mide en meses/persona (PM), lo que equivale a la cantidad de meses de trabajo a tiempo completo que una persona necesita para completar el proyecto. (Gómez, C.López, Migani, & Otazú)

a. Modelo Composición de Aplicación

La fórmula propuesta en este modelo es la siguiente:

Ecuación 1

$$PM = \frac{NOP}{PROD}$$

Donde:

NOP (Nuevos Puntos Objeto): Tamaño del nuevo software a desarrollar expresado en Puntos Objeto y se calcula de la siguiente manera:

Ecuación 2

$$NOP = \ OP * \ (100 - \%reuso)/100$$

OP (Puntos Objeto): Tamaño del software a desarrollar expresado en Puntos Objetos

%reuso: Porcentaje de reuso que se espera lograr en el proyecto.

PROD: Es la productividad promedio determinada a partir del análisis de datos de proyectos en (Boehm, 1995),

Tabla 3: *Modelo de Productividad*

Fuente: (Boehm, 1995)

Experiencia y capacidad de los desarrolladores	Muy bajo	Bajo	Normal	Alto	Muy alto
Madurez y Capacidad del ICASE	Muy bajo	Bajo	Normal	Alto	Muy alto
PROD	4	7	13	25	50

b. Modelo Diseño Temprano

Este modelo se aplica en las fases iniciales de un proyecto de software, cuando aún se tiene poca información sobre el tamaño del producto, la naturaleza de la plataforma, el equipo que se unirá al proyecto o los detalles específicos del proceso que se va a emplear. Puede ser utilizado en productos desarrollados en áreas como generadores de aplicaciones, sistemas integrados o infraestructura. (Gómez, C.López, Migani, & Otazú)

El modelo de Diseño Temprano ajusta el esfuerzo nominal mediante el uso de siete factores de costo. La fórmula para calcular dicho esfuerzo es la siguiente:

Ecuación 3

$$PM_{estimado} = PM_{nominal} * \prod_{i=1}^{7} E\,M_i$$

Ecuación 4

$$PM_{nominal} = A * (KSLOC)^B$$

Ecuación 5

$$B = 1.01 + 0.01 * \sum_{j=1}^{5} W_J$$

Donde:

- $PM_{estimado}$ es el esfuerzo Nominal ajustado por 7 factores, que reflejan otros aspectos propios del proyecto que afectan al esfuerzo necesario para la ejecución del mismo.

- KSLOC es el tamaño del software a desarrollar expresado en miles de líneas de código fuente.

- A es una constante que captura los efectos lineales sobre el esfuerzo de acuerdo a la variación del tamaño, (A=2.94).

- B es el factor exponencial de escala, toma en cuenta las características relacionadas con las economías y deseconomías de escala producidas cuando un proyecto de software incrementa su tamaño.

- EMi corresponde a los factores de costo que tienen un efecto multiplicativo sobre el esfuerzo, llamados Multiplicadores de Esfuerzo (Effort Multipliers). Cada factor se puede clasificar en seis niveles diferentes que expresan el impacto del multiplicador sobre el esfuerzo de desarrollo. Esta escala varía desde un nivel Extra Bajo hasta un nivel Extra Alto. Cada nivel tiene un peso asociado. El peso promedio o nominal es 1.0. Si el factor provoca un efecto nocivo en el esfuerzo de un proyecto, el valor del multiplicador correspondiente será mayor que 1.0, caso contrario el multiplicador será inferior a 1.0.

Clasificados en categorías, los 7 Multiplicadores de Esfuerzo son:

Del Producto

- RCPX: Confiabilidad y Complejidad del producto

- RUSE: Reusabilidad Requerida

De la Plataforma

- PDIF: Dificultad de la Plataforma

Del Personal

- PERS: Aptitud del Personal
- PREX: Experiencia del Personal

Del Proyecto

- FCIL: Facilidades

- SCED: Cronograma de Desarrollo Requerido

Vistos en una tabla podemos obtener lo siguiente:

Tabla 4: *Factores de Escala en COCOMO II.*

Fuente: (Boehm, 1995)

Factor de Escala	Descripción	Extra Bajo	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
PREC	Precedencia	6.20	4.96	3.72	2.48	1.24	0.00	-
FLEX	Flexibilidad en el desarrollo	5.07	4.05	3.04	2.03	1.01	0.00	-
RESL	Resolución de arquitectura y riesgo	7.07	5.65	4.24	2.83	1.41	0.00	-
TEAM	Cohesión del equipo	5.48	4.38	3.29	2.19	1.10	0.00	-
PMAT	Madurez del proceso	7.80	6.24	4.68	3.12	1.56	0.00	-

c. Modelo Post-Arquitectura

Este modelo de estimación es el más detallado y se utiliza cuando la arquitectura del proyecto ya ha sido completamente definida. Se aplica tanto en la fase de desarrollo como en el mantenimiento de productos de software, abarcando áreas como Sistemas Integrados, Infraestructura y Generadores de Aplicaciones. (Gómez, C.López, Migani, & Otazú)

El esfuerzo nominal es ajustado mediante 17 factores multiplicadores, lo que permite un análisis más preciso del conocimiento disponible en las etapas finales del desarrollo. Estos multiplicadores ajustan el modelo para reflejar de manera más fiel el producto de software que se está desarrollando. La fórmula utilizada para calcular el esfuerzo es la siguiente:

Ecuación 6

$$PM_{estimado} = PM_{nominal} * \prod_{i=1}^{17} EM_i$$

Nota: Llega a diecisiete porque son 17 los factores de costo correspondientes a este modelo

2.3.1.2 Estimación del Cronograma

La versión inicial de COCOMO II ofrece un modelo de estimación del cronograma que es comparable al presentado en COCOMO'81 y ADA COCOMO. (Gómez, C.López, Migani, & Otazú) la ecuación base para los tres modelos de COCOMO II es:

Ecuación 7

$$TDEV = \left[3.0 * PM_{*}^{(0.33+0.2*(B-1.01))}\right] * \frac{SCED\%}{100}$$

Donde:

- TDEV: es el tiempo calendario en meses que transcurre desde la determinación de los requerimientos a la culminación de una actividad que certifique que el producto cumple con las especificaciones.

- PM_*: El esfuerzo expresado en meses personas, calculado sin tener en cuenta el multiplicador de esfuerzo SCED.

- B: es el factor de escala

- SCED%: es el porcentaje de compresión/expansión del cronograma.

2.3.1.3 Métricas de Software

En la estimación del tamaño del software, COCOMO II emplea tres métodos principales: Puntos Objeto, Puntos Función No Ajustados y Líneas de Código Fuente. Además, se consideran otros factores relacionados con el tamaño del proyecto, como el reuso, la reingeniería, la conversión y el mantenimiento. (Gómez, C.López, Migani, & Otazú)

Es fundamental unificar los criterios de medición del tamaño del software, no solo para planificar y gestionar los proyectos de manera adecuada, sino también para llevar a cabo estudios y comparaciones entre proyectos con el fin de mejorar los procesos. (Park, 1992)

a. Puntos Objeto

Como indican (Gómez, C.López, Migani, & Otazú),Aunque la estimación mediante Puntos Objeto es un enfoque relativamente nuevo para medir el tamaño del software, resulta adecuado para aplicaciones con componentes y para estimar el esfuerzo en etapas de prototipación. En estas circunstancias, ha sido comparado con el método de estimación basado en Puntos Función. Un experimento realizado por Kaufman y Kumar en 1993 involucró a cuatro administradores de proyectos experimentados que utilizaron tanto Puntos Objeto como Puntos Función para estimar el esfuerzo requerido en dos proyectos que habían sido completados en 3.5 y 6 meses-persona, respectivamente. Las estimaciones se basaron en las descripciones disponibles al inicio de dichos proyectos.

Los resultados del experimento mostraron que:

- Tanto los Puntos Objeto como los Puntos Función produjeron estimaciones con un grado de precisión similar (los Puntos Objeto fueron ligeramente más precisos, aunque no de manera significativa desde el punto de vista estadístico).

- El tiempo promedio para realizar una estimación con Puntos Objeto fue aproximadamente el 47% del tiempo requerido para una estimación con Puntos Función. Además, los administradores consideraron que el método de Puntos Objeto era más fácil de usar.

Aunque estos resultados no son estadísticamente concluyentes, parecen lo suficientemente prometedores para justificar el uso de Puntos Objeto como una base en el modelo de estimación de Composición de Aplicación de COCOMO II.

El procedimiento para calcular Puntos Objeto en un proyecto de software es el siguiente:

- Primero: Determinar la cantidad de objetos: Se debe estimar el número de pantallas, reportes y componentes en 3GL que contendrá la aplicación.

- Segundo: Clasificar los objetos por complejidad: Cada objeto debe clasificarse según su nivel de complejidad (simple, media o difícil) de acuerdo con una tabla específica.

- Tercero: Asignar un peso a cada objeto: Según su complejidad, se le asigna un peso que refleja el esfuerzo relativo necesario para implementar una instancia de ese tipo de objeto.

- Cuarto: Calcular los Puntos Objeto: Se suman los pesos de todas las instancias de los objetos especificados para obtener el total de Puntos Objeto del proyecto.

Tabla 5: *Esquema de Clasificación de Puntos Objetos.*

Fuente: (Boehm, 1995)

Para Pantallas			
Cantidad de Vistas Contenidas	**Cantidad y fuente de las tablas de datos**		
	Total < 4 (< 2 servidor < 3 cliente)	Total < 8 (< 2 - 3 servidor < 3 - 5 cliente)	Total 8 + (> 3 servidor < 5 cliente)
< 3	Simple	Simple	Media
3 - 7	Simple	Media	Difícil
> 8	Media	Difícil	Difícil
Para Reportes			
Cantidad de Vistas Contenidas	**Cantidad y fuente de las tablas de datos**		
	Total < 4 (< 2 servidor < 3 cliente)	Total < 8 (< 2- 3 servidor < 3-5 cliente)	Total 8 + (> 3 servidor < 5 cliente)
0 o 1	Simple	Simple	Media
2 o 3	Simple	Media	Difícil
4 +	Media	Difícil	Difícil

Tabla 6: *Peso de un Punto Objeto.*

Fuente: (Boehm, 1995)

	Complejidad - Peso		
Tipo de Objeto	**Simple**	**Media**	**Difícil**
Pantalla	1	2	3
Reporte	2	5	8
Componente 3GL			10

b. Puntos función

Como explica (Gonzales, 2020), El modelo COCOMO II emplea Puntos Función y/o Líneas de Código Fuente (SLOC) como fundamentos para medir el tamaño en los modelos de estimación de Diseño Temprano y Post-Arquitectura. Las métricas relacionadas con los Puntos Función se basan en las directrices ofrecidas por el "*International Function Point User Group*" (IEPUG, 1994) (Kunkler,

1985) (Behrens, 1983). Los Puntos Función tienen como objetivo cuantificar la funcionalidad de un sistema de software, buscando obtener un número que represente de manera integral al sistema.

La fórmula de Albrecht (Albretch, 1979) para el cálculo de los puntos función es la siguiente:

Ecuación 8

$$\boldsymbol{FP = UFP * TCF}$$

Donde:

- UFP: Puntos Función no Ajustados
- TCF: Factor de Complejidad Técnica

Para calcular los UFP, se deben identificar los siguientes tipos de ítems:

- Entradas Externas (Inputs)
- Salidas Externas (Outputs)
- Archivo Lógicos Internos (Archivos)
- Archivos Externos de Interfase (Interfases)
- Solicitudes Externas (Queries)

Una vez identificados los ítems se clasifican de acuerdo al grado de complejidad en: bajo, promedio o alto. Se asigna un peso a cada ítem según el tipo y el grado de complejidad correspondiente. Finalmente, los UFP son calculados mediante la sumatoria de los pesos de todos los ítems identificados.

Ecuación 9

$$UFP = \sum_{i=1}^{15} (Cantidad_items_tipo_i) * (Peso_i)$$

La siguiente tabla muestra cómo se determinan los niveles de complejidad de cada tipo de ítem en función del número y tipo de elementos de datos y archivos involucrados.

Tabla 7: *Puntos Función. Determinación del Peso.*

Fuente: (Boehm, 1995)

Para archivos lógicos internos y archivos externos de interfase				**Para salidas y consultas externas**				**Para entradas externas**			
Elementos de Registro	**Elementos de datos**			**Tipos de archivos**	**Elementos de datos**			**Tipos de archivos**	**Elementos de datos**		
	1-19	20-50	51+		1-5	6-19	20+		1-4	5-15	16+
1	Bajo	Bajo	Prom.	0 ó 1	Bajo	Bajo	Prom.	0 ó 1	Bajo	Bajo	Prom.
2-5	Bajo	Prom.	Alto	2-3	Bajo	Prom.	Alto	2-3	Bajo	Prom.	Alto
6+	Prom.	Alto	Alto	4+	Prom.	Alto	Alto	3+	Prom.	Alto	Alto

La siguiente tabla muestra las ponderaciones asociadas a cada tipo de ítem. Estas ponderaciones han sido derivadas y validadas empíricamente mediante la observación de una gran variedad de proyectos.

Tabla 8: *Peso del Factor de Complejidad.*

Fuente: (Boehm, 1995)

Tipo de función	**Peso del Factor de Complejidad**		
	Bajo	**Promedio**	**Alto**
Entradas Externas (Inputs)	3	4	6
Salidas Externas (Outputs)	4	5	7
Archivo Lógicos Internos (Archivos)	7	10	15
Archivos Externos de Interfase (Interfases)	5	7	10
Consultas Externas (Queries)	3	4	6

Para el cálculo del Factor de Complejidad Técnica, TCF, se considera la siguiente fórmula:

Ecuación 10

$$TCF = 0.65 + 0.01 * \sum_{i=1}^{14} F_i$$

Donde los Fi corresponden a los pesos asignados a los siguientes factores:

- F1: Mecanismos de recuperación y back-up confiables
- F2: Comunicación de Datos
- F3: Funciones de Procesamiento Distribuido

- F4: Performance
- F5: Configuración usada rigurosamente
- F6: Entrada de datos on-line
- F7: Factibilidad Operativa
- F8: Actualización de archivos on-line
- F9: Interfaces Complejas
- F10: Procesamiento Interno Complejo
- F11: Reusabilidad
- F12: Fácil Instalación
- F13: Soporte de múltiples instalaciones
- F14: Facilidad de cambios y amigabilidad

Los pesos se consideran dentro de una escala de 0 a 5, descripta a continuación:

- 0: Sin influencia
- 1: Incidental
- 2: Moderado
- 3: Medio
- 4: Significativo
- 5: Esencial

Estas 14 características consideran aspectos como reusabilidad, performance, complejidad, confiabilidad, etc., contemplados por COCOMO II a través de los factores de costo. Es por ello que este modelo utiliza los UFP como métrica de determinación de tamaño.

c. Líneas de Código Fuente

COCOMO II toma la sentencia de código fuente lógica como una línea estándar de código. Sin embargo, establecer una definición precisa de lo que constituye una línea de código resulta

complicado, ya que hay variaciones conceptuales en la forma en que se cuentan las sentencias ejecutables y las declaraciones de datos en diferentes lenguajes de programación. El propósito de esta medida es evaluar el esfuerzo intelectual involucrado en el desarrollo de un programa. (Gómez, C.López, Migani, & Otazú)

Ilustración 3: *Checklist para la definición de una línea de código fuente.*

Fuente: (Columbusstate)

Definition Checklist for Source Statement Counts

Definition name: _____ Date: _____

_____ Originator: _____

Measurement unit:	Physical source lines ☐ Logical source statements ☐			
Statement type	**Definition** ☐	**Data array** ☐	**Includes**	**Excludes**
When a line or statement contains more than one type, classify it as the type with the highest precedence.				
1 Executable	**Order of precedence ->**	1		
2 Nonexecutable				
3 Declarations		2		
4 Compiler directives		3		
5 Comments				
6 On their own lines		4		
7 On lines with source code		5		
8 Banners and nonblank spacers		6		
9 Blank (empty) comments		7		
10 Blank lines		8		
11				
12				
How produced	**Definition** ☐	**Data array** ☐	**Includes**	**Excludes**
1 Programmed				
2 Generated with source code generators				
3 Converted with automated translators				
4 Copied or reused without change				
5 Modified				
6 Removed				
7				
8				
Origin	**Definition** ☐	**Data array** ☐	**Includes**	**Excludes**
1 New work: no prior existence				
2 Prior work: taken or adapted from				
3 A previous version, build, or release				
4 Commercial, off-the-shelf software (COTS), other than libraries				
5 Government furnished software (GFS), other than reuse libraries				
6 Another product				
7 A vendor-supplied language support library (unmodified)				
8 A vendor-supplied operating system or utility (unmodified)				
9 A local or modified language support library or operating system				
10 Other commercial library				
11 A reuse library (software designed for reuse)				
12 Other software component or library				
13				
14				
Usage	**Definition** ☐	**Data array** ☐	**Includes**	**Excludes**
1 In or as part of the primary product				
2 External to or in support of the primary product				
3				

Para minimizar esos problemas, se usa el checklist de definición desarrollado por el SEI, que permite unificar criterios en la definición de una línea de código fuente (Park, 1992), (Goethert, Bailey, & Busby, 1992). Ver la ilustración anterior. A los efectos de COCOMO II, se han efectuado cambios que consisten en eliminar las categorías de software que insumen poco esfuerzo. Así no están incluidas librerías de soporte de lenguajes, sistemas operativos, librerías comerciales, etc., ni tampoco el código generado con generadores de código fuente.

2.4 HERRAMIENTAS

2.4.1 Diagramas

Un diagrama es una representación gráfica que simplifica y organiza la información sobre un tema, teoría, proceso o sistema. Puede variar en complejidad, teniendo pocos o muchos elementos (Concepto, s.f.)

2.4.1.1 Diagramas de flujo

Según lo establece la (Universidad Veracruzana), un diagrama de flujo es una representación gráfica de un algoritmo. Estos diagramas utilizan diversos símbolos para ilustrar operaciones específicas, proporcionando así una visualización clara de las distintas acciones que deben llevarse a cabo para resolver un problema, junto con el orden lógico en que deben ejecutarse.

Se les denomina diagramas de flujo porque los símbolos se conectan mediante flechas, lo que indica la secuencia de operaciones. Para que los diagramas sean comprensibles para cualquier persona, se lleva a cabo una normalización de los símbolos; en otras palabras, se han creado símbolos casi universales. Esto es importante porque, en sus inicios, cada usuario podía tener sus propios símbolos para representar sus procesos en forma de diagrama de flujo, lo que resultaba en que solo aquellos familiarizados con esos símbolos podían interpretarlos. La simbología empleada en la creación de diagramas de flujo es variable y debe ajustarse a un patrón previamente definido.

Tabla 9: *Símbolos de diagramas de flujo*

Fuente: (Smartdraw)

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Linea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso

El diagrama de flujo representa la forma más tradicional y duradera para especificar los detalles algorítmicos de un proceso. Se utiliza principalmente en programación, economía y procesos industriales.

2.4.1.2 Diagramas de procesos

Según lo expuesto en el blog de (Venngage), un diagrama de actividades o diagrama de actividades UML ilustra el flujo o la secuencia de acciones que se llevan a cabo dentro de un sistema. Estos diagramas pertenecen a la categoría de diagramas de comportamiento, ya que representan cómo funciona un sistema al ejecutar una serie de acciones para completar un proceso o actividad.

Normalmente, las actividades se realizan de forma secuencial; no obstante, en los diagramas de actividades también pueden existir actividades que se ejecutan de manera concurrente o simultáneamente. Otra función importante de estos diagramas es mostrar un flujo de eventos y ayudar a identificar los requisitos de un proceso de negocio. Los diagramas de actividades proporcionan numerosos beneficios que transforman la gestión y comprensión de procesos. Uno de los aspectos más destacados es que brindan una representación visual clara que facilita la identificación de cada paso y su interrelación, lo que a su vez mejora la comprensión y eficiencia en la ejecución de tareas. (Venngage)

Como se menciona (Venngage), los diagramas de actividades tienen los siguientes componentes:

- Estado inicial y punto final: En un diagrama de actividades, el inicio y el final de un proceso se señalan mediante su estado inicial y su punto final. Este último es fundamental para entender la conclusión de un proceso o actividad, ya que representa el estado final en el diagrama.

- Actividad o estado de acción: Se refiere a una única actividad que inicia una secuencia de acciones. Por ejemplo, cuando un usuario inicia sesión en su cuenta del sistema de banca móvil.

- Acciones: Dentro de una actividad, se ejecuta una acción o paso, que puede ser realizado por el sistema o el usuario para completar una tarea. Siguiendo el ejemplo anterior, una acción podría ser cuando el usuario consulta el saldo de su cuenta.

- Objetos: Se consideran materiales o datos que se crean o utilizan en una actividad.

- Decisiones: Se toman antes de proceder con otra acción o actividad. Las respuestas deben ser obligatoriamente 'sí' o 'no'.

- Sincronización: Consta de dos nodos: el nodo de bifurcación, desde donde se ramifican los flujos concurrentes, y el nodo de unión, que recoge estos flujos en uno solo.

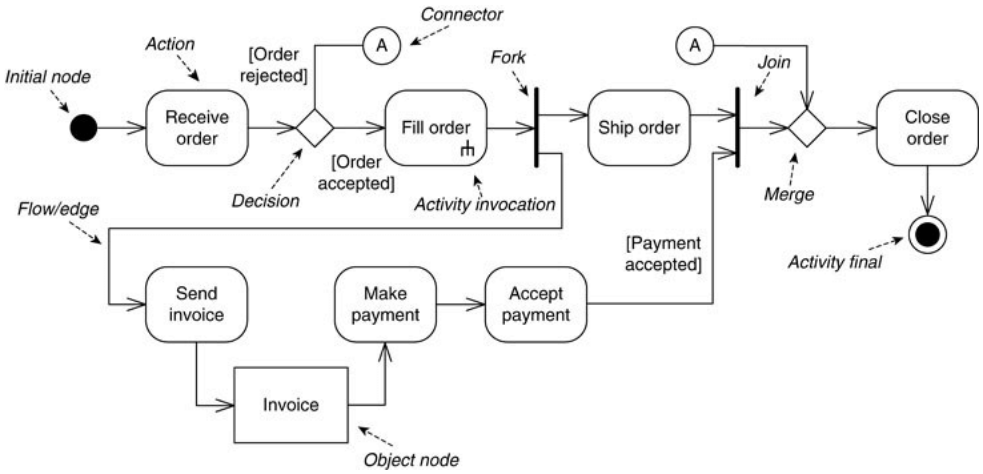
- Señales: Indican la ejecución de acciones externas al sistema que pueden alterar la actividad. Un ejemplo claro es durante un pago en línea, cuando el usuario recibe un código OTP como señal para autorizar la transacción.

- Carril: Se utilizan para agrupar, mediante columnas o categorías, actividades relacionadas que son realizadas por diferentes actores.

Los diagramas de actividades cuentan con la siguiente simbología:

Ilustración 4: *Simbología de un diagrama de actividades*

Fuente: (Venngage)



Según (Venngage), el significado de cada símbolo es el siguiente:

- Nodo de inicio: Simboliza el estado de inicio de la actividad. Se representa por medio de un pequeño círculo relleno.

- Nodo de actividad: Simboliza un estado de actividad o acción. Se representa por medio de un rectángulo con esquinas redondeadas.

- Nodo de acción: Se utiliza para representar una acción. El símbolo es una forma de estadio o cápsula.

- Flujo de acción: También se llama límite de actividad o flujo de control. Se representa por medio de una flecha para indicar el paso de una actividad o acción a otra.

- Nodo de objeto: Simboliza un objeto que se crea o utiliza dentro de la actividad. Se representa por medio de un rectángulo.

- Flujo de objetos: Se agrega luego de una acción para indicar la creación de un objeto, también se puede colocar antes de la acción para indicar que esta requiere un objeto. Se representa por medio de un codo o una flecha discontinua.

- Nodo de bifurcación y nodo de unión: El nodo de bifurcación divide una acción en flujos concurrentes, mientras que el nodo de unión recoge flujos concurrentes en una sola acción. Ambos se encuentran representados por una línea horizontal gruesa. .

- Evento de tiempo: Simboliza un periodo de tiempo dentro de una actividad. Se representa por medio de un reloj de arena.

- Señales enviadas y recibidas: Las señales enviadas se representan por medio de una flecha con forma de pentágono e informan que es necesario recibir una señal para finalizar una acción; las señales recibidas se representan por medio de una bandera con cola de golondrina e informan que la señal se ha recibido.

- Nodo final: Determina el estado final de la actividad. Se representa por medio de un pequeño círculo relleno dentro de otro círculo.

2.4.1.3 Diagrama entidad relación

Según lo menciona el blog de Miro (Miro), un diagrama de entidad-relación, conocido también como diagrama ER, ERD o simplemente ERD, ilustra cómo interactúan diversas entidades, que pueden ser personas, objetos o conceptos. Estos modelos conceptuales de datos son herramientas valiosas

para que desarrolladores y diseñadores visualicen las relaciones entre los elementos clave del software.

Los ERD fueron creados por el informático Peter Chen en la década de 1970 y son frecuentemente utilizados por las empresas para el diseño de bases de datos. Sin embargo, también pueden representar las relaciones entre otros elementos dentro de la organización, como las funciones (por ejemplo, la relación entre un product manager y un desarrollador), así como objetos tangibles (como un producto o servicio) e intangibles (como el product backlog).

En el blog de Miro (Miro), se detallan los siguientes componentes de un diagrama de entidad-relación:

- Entidades: Las entidades, que generalmente se representan como rectángulos, son sustantivos que pueden referirse a objetos, personas, conceptos o eventos. Diferentes símbolos suelen indicar distintos tipos de entidades; por ejemplo, un conjunto de entidades fuerte se representa con un rectángulo simple, mientras que un conjunto de entidades débil se ilustra con un rectángulo doble.

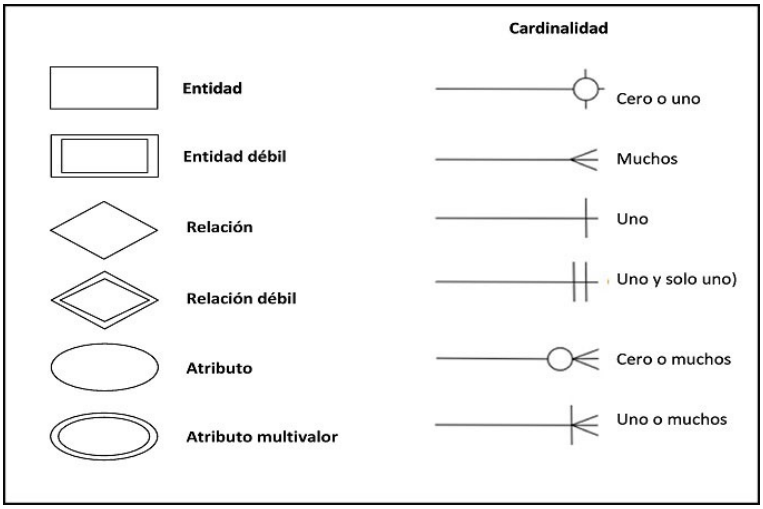
- Atributos: Los atributos son las características que definen a una entidad. Por ejemplo, si el diagrama ER representa el software utilizado por los estudiantes para solicitar una plaza en la universidad, bajo la entidad "Estudiante" se incluirían atributos como Número_de_estudiante, Nombre_de_estudiante y Curso_de_estudio.

- Relaciones: En un diagrama ER, las líneas y flechas de conexión están etiquetadas, indicando cómo interactúan las entidades y mostrando las relaciones entre ellas a los desarrolladores. Los diferentes tipos de líneas de conexión tienen significados distintos y representan cuántos datos fluyen de una entidad a otra. Estas relaciones son cardinales, las cuales se explicarán en mayor detalle más adelante.

Creados por el informático Peter Chen en los años 70, los ERD son utilizados habitualmente por las empresas en el diseño de bases de datos. Sin embargo, los diagramas entidad relación también pueden representar las relaciones entre otros elementos de la empresa, como las funciones (por ejemplo, la relación de un product manager con un desarrollador), los objetos tangibles de la empresa (como un producto o servicio) y los objetos intangibles de la empresa (como el product backlog).

Ilustración 5: *Simbología de diagrama entidad relación.*

Fuente: (Green, 2021)



Adicionalmente (Green, 2021) agrega algunos componentes clave del diagrama entidad relación:

- Entidad débil: a diferencia de una entidad fuerte que se define por sus atributos, una entidad débil depende únicamente de la existencia de otra entidad.

- Atributo derivado: se refiere a un atributo derivado o basado en otro atributo.

- Atributo multivaluado: un tipo de atributo que puede tener varios valores.

- Cardinalidad: se refiere a las ocurrencias de una relación. En concreto, especifica el número máximo de relaciones entre dos entidades.

- Ordinalidad: describe si una relación es obligatoria u opcional. También se utiliza para determinar el número mínimo absoluto de relaciones.

2.4.2 Figma

Figma es una herramienta ampliamente utilizada por diseñadores, gerentes de producto, redactores y desarrolladores, que permite la creación, compartición y prueba de diseños para sitios web, aplicaciones móviles y otras experiencias digitales (Figma, s.f.). Una de sus características más destacadas es que permite que varios diseñadores y desarrolladores colaboren en un mismo proyecto de manera simultánea, lo que facilita el trabajo en equipo.

Entre las funcionalidades que ofrece Figma se encuentran la creación de prototipos interactivos, el uso de componentes reutilizables y el acceso a bibliotecas de diseño. Además, al estar basada en la nube, permite compartir proyectos fácilmente a través de un enlace, lo que posibilita revisiones y retroalimentación en tiempo real. Figma también se integra con otras herramientas y permite la exportación de recursos gráficos para su utilización en el desarrollo de productos digitales.

2.4.3 Git

Git es un sistema de control de versiones distribuido, lo que implica que cada clon local del proyecto actúa como un repositorio completo de control de versiones. Esta funcionalidad de los repositorios locales permite a los usuarios trabajar sin conexión o de forma remota de manera sencilla. (Microsoft, s.f.) esta metodología contrasta con el modelo de control de versiones centralizado, que requiere que los usuarios se conecten a un servidor para actualizar o crear nuevas versiones del proyecto.

La versatilidad y popularidad de Git lo convierten en una herramienta óptima para equipos de desarrollo de diversos tamaños. Además, como muchos desarrolladores, incluidos los recién graduados, ya están familiarizados con Git, su implementación en proyectos resulta bastante fácil. La amplia comunidad de usuarios de Git ha producido una gran variedad de recursos educativos, facilitando el aprendizaje y el soporte para los desarrolladores cuando lo requieren. Igualmente, casi todos los entornos de desarrollo son compatibles con Git, y las herramientas de línea de comandos están disponibles en los principales sistemas operativos, lo que contribuye a su accesibilidad y facilidad de uso.

2.4.3.1 Aspectos básicos de Git

Cada vez que se guarda el progreso en Git, se crea una confirmación que funciona como una instantánea de todos los archivos en ese momento. Si un archivo no ha cambiado entre dos confirmaciones, Git utiliza la versión almacenada previamente de ese archivo. Este método difiere de otros sistemas de control de versiones, que suelen conservar una versión inicial del archivo y luego registran únicamente las diferencias a lo largo del tiempo.

Las confirmaciones en Git están interconectadas, formando una estructura que representa el historial completo del desarrollo. Este sistema facilita la reversión del código a un estado anterior, permite analizar los cambios entre confirmaciones y consultar detalles sobre cuándo y dónde se realizaron

las modificaciones. Cada confirmación cuenta con un hash criptográfico único que se genera a partir de su contenido (Microsoft, s.f.)

Gracias a esta implementación de hashes, Git puede identificar cualquier intento de alterar, perder o dañar información, lo que añade una capa extra de integridad y seguridad en el seguimiento de los cambios (Microsoft, s.f.)

2.4.3.2 Ramas

Cada desarrollador realiza sus cambios en su propio repositorio de código local. Esto puede dar lugar a múltiples modificaciones que se basan en la misma confirmación. Git ofrece herramientas que permiten aislar estos cambios y posteriormente volver a combinarlos. Las ramas, que actúan como punteros ligeros para el trabajo en progreso, gestionan esta separación. Una vez que se completa el trabajo en una rama, se puede integrar nuevamente en la rama principal (o troncal) del equipo. (Microsoft, s.f.)

2.4.3.3 Archivos y confirmaciones

Los archivos en Git pueden encontrarse en uno de tres estados: modificados, almacenados provisionalmente o confirmados. (Microsoft, s.f.) Cuando un archivo se modifica por primera vez, esos cambios solo existen en el directorio de trabajo y no forman parte de ninguna confirmación ni del historial del proyecto. Para que esos cambios se incluyan en la próxima confirmación, el desarrollador debe almacenarlos provisionalmente. El área de almacenamiento provisional alberga todos los archivos modificados que se están preparando para ser parte de la siguiente confirmación. Una vez que el desarrollador está satisfecho con los archivos en esta área, se agrupan en una confirmación, que va acompañada de un mensaje que describe los cambios realizados. Esta confirmación se añade posteriormente al historial de desarrollo.

El proceso de almacenamiento provisional proporciona a los desarrolladores una gran flexibilidad, ya que les permite elegir qué cambios específicos desean incluir en cada confirmación. Esto resulta particularmente útil en el caso de modificaciones significativas, ya que facilita la posibilidad de dividirlas en varias confirmaciones más pequeñas y manejables. Al reducir el tamaño y el alcance de cada confirmación, se hace más fácil revisar el historial, permitiendo localizar cambios específicos en los archivos con mayor precisión.

2.4.4 Github

GitHub es una plataforma donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código. (GitHub, s.f.)

Almacenar tu código en un "repositorio" en GitHub ofrece una serie de beneficios clave para desarrolladores y equipos de trabajo:

- Presentación y compartición del trabajo: GitHub permite que los proyectos sean fácilmente accesibles para cualquier persona, lo que es ideal tanto para presentar trabajos a posibles empleadores o clientes, como para compartir código con la comunidad de desarrolladores.

- Seguimiento y gestión de cambios: A lo largo del ciclo de vida de un proyecto, GitHub facilita el control y la administración de todas las modificaciones hechas al código. Esto se realiza mediante un historial detallado, permitiendo a los desarrolladores revisar cambios previos, revertir errores, y comparar versiones del código con facilidad.

- Revisión y sugerencias: GitHub permite que otros usuarios, ya sea dentro del equipo o desde la comunidad, revisen el código de manera colaborativa. A través de pull requests, pueden hacer sugerencias, señalar errores o proponer mejoras, promoviendo un entorno de trabajo colaborativo y de mejora continua.

- Colaboración en proyectos compartidos: Una de las mayores ventajas es la posibilidad de que múltiples personas trabajen en el mismo proyecto sin interferir en el trabajo de los demás. GitHub permite a los desarrolladores trabajar en diferentes ramas del código, integrando los cambios solo cuando se han revisado y aprobado, lo que asegura un flujo de trabajo organizado y libre de conflictos.

Además de todas estas funciones, el poder de GitHub radica en su integración con Git, el software de control de versiones de código abierto que actúa como la base de la plataforma. Git permite a los desarrolladores manejar versiones distribuidas del proyecto, lo que significa que cada colaborador tiene una copia completa del repositorio, pudiendo trabajar tanto en línea como fuera de línea. Este enfoque descentralizado proporciona mayor flexibilidad, seguridad y control sobre los cambios.

GitHub también ofrece otras herramientas adicionales, como la gestión de proyectos, integración continua, y automatización de flujos de trabajo, lo que lo convierte en una plataforma esencial para equipos de desarrollo de cualquier tamaño que buscan una colaboración eficiente y una administración organizada del código fuente.

2.4.5 GitFlow

Git Flow es una metodología de trabajo con Git que fue desarrollada por Vincent Driessen en 2010 y se basa en la idea de tener dos ramas principales: *develop* y *main*, y varias ramas secundarias para desarrollo de nuevas funcionalidades, correcciones de errores y lanzamientos. (Isla, 2022)

Git Flow se basa en tener dos ramas principales: *develop* y *main*. La rama *develop* es donde se desarrollan las nuevas funcionalidades y se hacen las correcciones de errores mientras que la rama *main* contiene el código estable y listo para ser lanzado. (Isla, 2022)

Para labores concretas, pueden usarse otro tipo de ramas, las cuales tienen un tiempo de vida definido. Es decir, cuando ya no son necesarias se eliminan: (GitHub, s.f.)

- Ramas de funcionalidad (feature branches)
- Ramas de versión (release branches)
- Ramas de parches (hotfix branches)

a. Feature branches

- Pueden partir de: *develop*
- Deben fusionarse con: *develop*
- Convención de nombres: feature-NUMissue-*.

b. Release branches

- Pueden partir de: develop

- Deben fusionarse con: *develop* y *main*

- Convención de nombres: release-*

c. Hotfix branches

- Pueden partir de: *main*

- Deben fusionarse con: *develop* y *main*

- Convención de nombres: hotfix-*

2.4.6 Notion

Notion es una herramienta en línea multifuncional que integra la gestión de proyectos, bases de datos y notas en una sola plataforma. Ofrece un espacio de trabajo digital donde los usuarios pueden crear, compartir y colaborar en diversos proyectos y tareas de manera centralizada. (Arimetrics, 2024)

Esta herramienta es altamente versátil y permite organizar proyectos, notas y tareas de forma eficiente, adaptándose a las necesidades de cada usuario. Debido a su flexibilidad, es útil para estudiantes, profesionales, empresas y cualquier persona que necesite gestionar tareas y proyectos de manera efectiva.

Según (Arimetrics, 2024), entre las principales características de Notion se encuentran:

- Interfaz intuitiva y personalizable: Los usuarios pueden adaptar la plataforma según sus preferencias, desde la organización de las páginas hasta el diseño de las notas.

- Integración con otras aplicaciones: Permite conectar con herramientas y servicios externos, como calendarios, hojas de cálculo y sistemas de gestión de tareas.

- Edición avanzada de texto: Ofrece funciones sofisticadas de edición con formatos avanzados y plantillas para documentos y notas profesionales.

- Colaboración en tiempo real: Varias personas pueden trabajar simultáneamente en un proyecto, viendo las actualizaciones de forma instantánea.
- Compartir información: Facilita el intercambio de notas y documentos con otros usuarios de manera segura.

- Almacenamiento en la nube: Incluye almacenamiento en la nube, asegurando el acceso y gestión segura de archivos.

- Gestión de proyectos: Permite la creación, administración de proyectos, asignación de tareas y seguimiento del progreso de cada una.

2.4.7 React

Según su mismo sitio web oficial, es una biblioteca de JavaScript de código abierto, desarrollada por Facebook, que se utiliza para construir interfaces de usuario, especialmente en aplicaciones web de una sola página. (React)

Fue diseñada para facilitar la creación de componentes reutilizables que gestionan su propio estado y se actualizan eficientemente cuando cambian los datos. React permite a los desarrolladores construir interfaces interactivas al dividir la UI en componentes más pequeños, lo que facilita la organización del código y su mantenimiento.

Una de las características más destacadas de React es el uso de un "Virtual DOM", un modelo que permite actualizar solo los elementos que han cambiado, lo que mejora el rendimiento de las aplicaciones web.

2.4.8 Electron

Según la propia página web de Electron js (Electron, s.f.), Electron es un framework para crear aplicaciones de escritorio usando JavaScript, HTML y CSS. Incrustando Chromium y Node.js dentro del mismo, Electron permite mantener una base de código JavaScript y crear aplicaciones multiplataforma que funcionan en Windows, macOS y Linux, así mismo, no requiere experiencia en desarrollo nativo.

Sus características principales son las siguientes:

- Multiplataforma: Las aplicaciones creadas con Electron funcionan en diferentes sistemas operativos sin modificaciones.

- Uso de tecnologías web: Utiliza las mismas tecnologías que se usan para desarrollar sitios web, lo que permite a los desarrolladores web crear aplicaciones de escritorio con facilidad.

- Acceso a APIs de Node.js: Los desarrolladores pueden acceder al sistema de archivos, procesos y otras APIs nativas de Node.js desde la aplicación.

- Chromium integrado: Electron incorpora Chromium para proporcionar una experiencia de interfaz de usuario rica y compatible con las últimas tecnologías web.

- Empaquetado de aplicaciones: Puedes empaquetar y distribuir tus aplicaciones de Electron como un archivo ejecutable nativo para cada sistema operativo.

2.4.9 Laravel

Laravel es un *framework* PHP gratis y de código abierto que brinda un conjunto de herramientas y recursos para crear aplicaciones modernas. Posee un ecosistema integral que combina funciones integradas y una variedad de paquetes y extensiones compatibles. (Cristancho, 2022)

Diseñado para desarrollar aplicaciones web siguiendo el patrón de arquitectura Modelo-Vista-Controlador (MVC). (Cristancho, 2022) si bien este año 2024 no está en el top de *frameworks* de desarrollo web (Developers Survey, 2024), su enfoque elegante y expresivo para simplificar tareas comunes en el desarrollo web, como enrutamiento, autenticación, manejo de bases de datos y sesiones, lo hace una herramienta completa que cumple muchos requerimientos.

Algunas de sus principales características son las siguientes:

- Eloquent ORM: Un sistema de mapeo objeto-relacional (ORM) que facilita el trabajo con bases de datos utilizando modelos PHP.

- Sistema de enrutamiento: Permite definir rutas de manera sencilla, haciendo que sea fácil mapear URLs a controladores.

- Motor de plantillas Blade: Un motor de plantillas ligero que permite usar PHP en vistas de manera sencilla y eficiente, incluyendo características como plantillas heredables.

- Migraciones y seeds de base de datos: Facilita la gestión de esquemas de base de datos y el llenado con datos iniciales o de prueba.

- Autenticación y autorización: Laravel ofrece herramientas integradas para gestionar de manera fácil y segura la autenticación de usuarios y la autorización de permisos.

- Colas de trabajo y tareas programadas: Soporta la ejecución de tareas en segundo plano y la programación de trabajos repetitivos, como correos electrónicos o procesamiento de datos.

- Testing: Ofrece soporte integrado para pruebas unitarias, lo que facilita verificar el correcto funcionamiento de la aplicación.

2.4.10 JWT

JSON Web Token (JWT) es un estándar abierto (RFC 7519) que establece un método compacto y autónomo para transmitir información de manera segura entre las partes, utilizando un objeto JSON. Esta información es confiable y puede ser verificada debido a que está firmada digitalmente. Los JWT pueden ser firmados mediante un secreto utilizando el algoritmo HMAC, o mediante un par de claves pública y privada con los algoritmos RSA o ECDSA. (JWT, 2024)

Aunque es posible cifrar los JWT para garantizar la confidencialidad entre las partes, nos enfocaremos en los tokens firmados. Los tokens firmados permiten verificar la integridad de las afirmaciones que contienen, mientras que los tokens cifrados ocultan dichas afirmaciones de otras partes. Cuando los tokens se firman con pares de claves pública y privada, la firma también asegura que solo la parte que posee la clave privada ha realizado la firma.

Según (JWT, 2024), presenta algunos contextos en los que los tokens web JSON (JWT) son beneficiosos:

- Autorización: Este es el uso más frecuente de los JWT. Después de que un usuario inicia sesión, cada solicitud subsiguiente incluirá el JWT, lo que le permitirá acceder a rutas, servicios y recursos que están habilitados para ese token. Una de las características más utilizadas de los JWT en la actualidad es el inicio de sesión único, gracias a su bajo peso y a su facilidad para ser utilizados en diferentes dominios.

- Intercambio de información: Los tokens web JSON son una opción eficaz para transmitir información de manera segura entre diferentes partes. Dado que los JWT pueden ser firmados (por ejemplo, utilizando pares de claves públicas y privadas), se puede tener confianza en que los remitentes son quienes afirman ser. Además, como la firma se genera a partir del encabezado y la carga útil, es posible verificar que el contenido no ha sido modificado.

2.4.10.1 Estructura de los JWT

En el blog (JWT, 2024), en su forma compacta, los tokens web JSON constan de tres partes separadas por puntos (.), que son:

a. Encabezamiento

El encabezado generalmente consta de dos partes: el tipo de token, que es JWT, y el algoritmo de firma utilizado, como HMAC SHA256 o RSA.

Ilustración 6: *Encabezado JWT*

Fuente: (JWT, 2024)

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Luego, este JSON se codifica en Base64Url para formar la primera parte del JWT.

b. Carga útil

La segunda sección del token es conocida como la carga útil, que alberga las reclamaciones. Estas reclamaciones son afirmaciones sobre una entidad (generalmente, el usuario) junto con información adicional. Existen tres categorías de reclamaciones: registradas, públicas y privadas.

- Reclamaciones registradas: se trata de un conjunto de reclamaciones predefinidas que, aunque no son obligatorias, son altamente recomendadas para ofrecer un conjunto de reclamaciones útiles e interoperables. Algunos ejemplos incluyen: iss (emisor), exp (fecha de expiración), sub (tema), aud (audiencia), entre otros.

- Reclamaciones públicas: los usuarios de JWT pueden definir estas reclamaciones según sus necesidades, pero para evitar conflictos, deben registrarse en el Registro de tokens web JSON de la IANA o definirse como una URL que incluya un espacio de nombres que prevenga colisiones.

- Reclamaciones privadas: estas son reclamaciones personalizadas creadas para intercambiar información entre partes que han acordado utilizarlas y no forman parte de las reclamaciones registradas ni públicas.

Un ejemplo de carga útil podría ser:

Ilustración 7: *Carga útil de JWT*

Fuente: (JWT, 2024)



Luego, la carga útil se codifica en Base64Url para formar la segunda parte del token web JSON.

c. Firma

Para generar la sección de la firma, es necesario combinar el encabezado codificado, la carga útil codificada, un secreto y el algoritmo mencionado en el encabezado, y luego proceder a firmarlo.

Por ejemplo, al emplear el algoritmo HMAC SHA256, la firma se construirá de la siguiente manera:

Ilustración 8: *Fimar de un JWT*

Fuente: (JWT, 2024)

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

La firma tiene como propósito asegurar que el mensaje no haya sido alterado durante su transmisión. Además, en el caso de los tokens firmados con una clave privada, también permite autenticar que el remitente del JWT es realmente quien dice ser.

El resultado consiste en tres cadenas de URL codificadas en Base64, separadas por puntos, que se pueden transmitir de manera sencilla en entornos HTML y HTTP. Además, son más compactas en comparación con estándares basados en XML, como SAML.

A continuación, se presenta un JWT que contiene el encabezado y la carga útil mencionados, los cuales han sido codificados y firmados con un secreto.

Ilustración 9: *JWT cifrado*

Fuente: (JWT, 2024)

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

d. Funcionamiento

En el proceso de autenticación, al momento en que el usuario inicia sesión con éxito utilizando sus credenciales, se genera un token web JSON (JWT). Debido a que los tokens funcionan como credenciales, es fundamental manejarlos con precaución para evitar riesgos de seguridad. En general, no se recomienda guardar los tokens más tiempo del necesario. (JWT, 2024)

Además, no se deberían almacenar datos sensibles de sesión en el almacenamiento del navegador por motivos de seguridad.

Cada vez que el usuario intente acceder a una ruta o recurso protegido, el agente de usuario debe enviar el JWT, generalmente en el encabezado de autorización utilizando el esquema de portador. El encabezado debe tener un formato similar al siguiente:

Ilustración 10: *Ejemplo de encabezado JWT*

Fuente: (JWT, 2024)

```
Authorization: Bearer <token>
```

En ciertas circunstancias, este puede funcionar como un mecanismo de autorización sin estado. Las rutas protegidas en el servidor verificarán la validez del JWT en el encabezado de autorización y, si es correcto, se permitirá al usuario acceder a los recursos restringidos. Si el JWT incluye la información necesaria, se puede reducir la necesidad de consultar la base de datos para realizar ciertas operaciones, aunque esto no siempre será aplicable. (JWT, 2024)

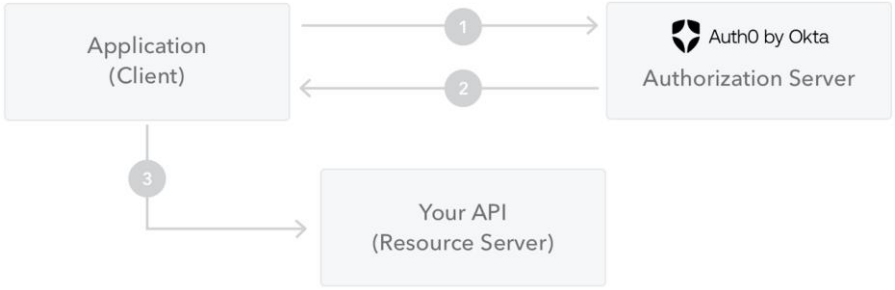
Si se envían tokens JWT a través de encabezados HTTP, se debe evitar que sean excesivamente grandes. Algunos servidores pueden no aceptar encabezados que superen los 8 KB. Si se intenta incluir demasiada información en un JWT, como todos los permisos del usuario, podría ser necesario considerar alternativas como Auth0 Fine-Grained Authorization. (JWT, 2024)

Si el token se transmite en el encabezado de autorización, no habrá problemas relacionados con el intercambio de recursos de origen cruzado (CORS), ya que no se utilizan cookies.

El siguiente diagrama ilustra el proceso de obtención y uso de un JWT para acceder a APIs o recursos.

Ilustración 11: *Flujo de uso de JWT*

Fuente: (JWT, 2024)



- La aplicación o el cliente solicita autorización al servidor de autorización. Esto se realiza a través de uno de los diferentes flujos de autorización. Por ejemplo, una aplicación web típica compatible con OpenID Connect pasará por el /oauth/authorize punto final mediante el flujo de código de autorización.
- Cuando se concede la autorización, el servidor de autorización devuelve un token de acceso a la aplicación.
- La aplicación utiliza el token de acceso para acceder a un recurso protegido (como una API).

2.4.11 Postman

Postman se originó como una herramienta diseñada para facilitar la creación de solicitudes a APIs, lo que permite a los usuarios probar su funcionamiento de manera sencilla. Inicialmente, se presentó como una extensión de Google Chrome. Los usuarios de Postman pueden ser desarrolladores que verifiquen el funcionamiento de una API en la que están trabajando, o operadores que realizan tareas de monitoreo sobre la misma. (Cuervo, 2019)

Además de su función principal de probar APIs, Postman ofrece una variedad de herramientas adicionales que simplifican la gestión de estas. Por esta razón, incluye funciones para documentar APIs, monitorearlas y crear equipos de trabajo colaborativo, convirtiéndose así en una plataforma de desarrollo de APIs que adopta un enfoque API First.

Actualmente, Postman ofrece aplicaciones para Windows, Linux y Mac, así como un módulo colaborativo basado en la nube para equipos.

2.4.11.1 Características de Postman

- Crear Peticiones: Permite a los usuarios crear y enviar solicitudes HTTP a servicios REST a través de una interfaz gráfica, guardando y reproduciendo dichas solicitudes posteriormente.

- Definir Colecciones: Los usuarios pueden organizar las APIs en colecciones y establecer el modelo de autenticación que se aplicará a cada solicitud. También es posible ejecutar pruebas y definir variables para la colección.

- Gestionar la Documentación: Postman genera documentación basada en las APIs y colecciones creadas, pudiendo hacerla pública.

- Entorno Colaborativo: Facilita la compartición de APIs entre miembros del equipo, apoyándose en herramientas colaborativas en la nube.

- Generar Código de Invocación: A partir de una API, Postman puede generar código de invocación para varios lenguajes de programación, incluyendo C, cURL, C#, Go, Java, JavaScript, NodeJS, Objective-C, PHP, Python, Ruby, Shell y Swift.

- Establecer Variables: Permite la creación de variables locales y globales que se pueden utilizar en las invocaciones o pruebas.

- Soportar el Ciclo de Vida de API Management: Postman facilita la gestión del ciclo de vida del API, desde su conceptualización y definición, hasta su desarrollo, monitoreo y mantenimiento.

- Crear Mockups: Los usuarios pueden crear un servidor de mockups o un entorno sandbox para probar sus APIs antes de que estén completamente desarrolladas.

2.4.12 Kong

Kong Gateway es una puerta de enlace de API diseñada para la nube que es ligera, rápida y adaptable. Actúa como un proxy inverso que facilita la gestión, configuración y enrutamiento de solicitudes hacia las API. (kong, 2024)

Esta herramienta se integra frente a cualquier API RESTful y se puede ampliar mediante módulos y complementos. Kong Gateway está optimizada para funcionar en arquitecturas descentralizadas, abarcando implementaciones de nube híbrida y multinube.

Los usuarios de Kong Gateway pueden:

- Beneficiarse de la automatización de flujos de trabajo y de las prácticas modernas de GitOps.

- Descentralizar aplicaciones y servicios, facilitando la transición hacia microservicios.

- Fomentar un ecosistema de desarrolladores de API activo y productivo.

- Detectar de manera proactiva anomalías y amenazas relacionadas con las API.

- Asegurar y gobernar los API y servicios, además de mejorar la visibilidad de las API en toda la organización.

2.4.13 MySql

MySQL es la base de datos relacional de código abierto más utilizada en el mercado. De acuerdo con DB-Engines, ocupa el segundo lugar en popularidad, justo detrás de Oracle Database. MySQL impulsa numerosas aplicaciones ampliamente utilizadas, como Facebook, Twitter, Netflix, Uber, Airbnb, Shopify y Booking.com. (Oracle, 2024)

En una base de datos relacional, la información se almacena en tablas distintas en lugar de agruparse en un único repositorio. La estructura de la base de datos está organizada en archivos físicos que están optimizados para mejorar la eficiencia. El modelo lógico de datos, que incluye elementos como tablas, vistas, filas y columnas, proporciona un entorno de programación versátil. Se definen reglas que regulan las relaciones entre diferentes campos de datos, como uno a uno, uno a muchos, y se especifica si son únicos, obligatorios u opcionales, así como las conexiones entre varias tablas. Gracias a estas normas, una base de datos bien diseñada asegura que la aplicación nunca se enfrente a datos inconsistentes, duplicados, huérfanos, obsoletos o faltantes.

2.4.13.1 Ventajas

MySQL es un sistema de gestión de bases de datos que se destaca por su rapidez, fiabilidad, escalabilidad y facilidad de uso. Inicialmente, se creó para gestionar grandes volúmenes de datos de manera eficiente y ha sido utilizado en entornos de producción exigentes durante muchos años.

A pesar de que MySQL sigue en constante evolución, cuenta con un conjunto de características enriquecido y valioso. Su capacidad de conectividad, velocidad y seguridad lo convierten en una opción ideal para acceder a bases de datos en línea.

Entre las principales ventajas de MySQL se encuentran:

- Facilidad de uso: Los desarrolladores pueden instalar MySQL en cuestión de minutos, y la gestión de la base de datos es sencilla

- Confiabilidad: MySQL es una de las bases de datos más consolidadas y utilizadas, con más de 25 años de experiencia en diversos escenarios, incluidas las grandes corporaciones. Muchas organizaciones dependen de MySQL para ejecutar aplicaciones críticas debido a su confiabilidad.

- Escalabilidad: MySQL puede adaptarse a las necesidades de las aplicaciones más demandantes. Su arquitectura de replicación nativa permite a empresas como Facebook escalar sus aplicaciones para manejar miles de millones de usuarios.

- Desempeño: HeatWave MySQL ofrece un rendimiento superior y costos más bajos en comparación con otros servicios de bases de datos, respaldado por referencias estándar del sector como TPC-H, TPC-DS y CH-benCHmark.

- Alta disponibilidad: MySQL incluye una gama completa de tecnologías de replicación nativas e integradas para garantizar alta disponibilidad y recuperación ante desastres. Para aplicaciones empresariales críticas y para cumplir con los acuerdos de nivel de servicio, los clientes pueden alcanzar objetivos de recuperación que permiten cero pérdidas de datos y tiempos de recuperación de segundos con conmutación por error automática.

- Seguridad: La protección de datos es esencial para cumplir con normativas gubernamentales y del sector, como el Reglamento General de Protección de Datos de la UE, la Norma de Seguridad de Datos del Sector de Tarjetas de Pago, la Ley de Portabilidad y Responsabilidad de Seguros Médicos, y las guías de implementación técnica de la Agencia de Sistemas de Información de Defensa. MySQL Enterprise Edition incluye características avanzadas de seguridad, como autenticación y autorización, cifrado de datos transparente, auditoría, enmascaramiento de datos y un firewall de base de datos.

- Flexibilidad: El almacén de documentos de MySQL permite a los usuarios disfrutar de una gran flexibilidad al desarrollar aplicaciones, ya sea en bases de datos SQL tradicionales o en entornos NoSQL sin esquema. Los desarrolladores pueden integrar datos relacionales y documentos JSON en una misma base de datos y aplicación.

2.4.14 SocketIO

Socket.IO es una biblioteca orientada a eventos diseñada para aplicaciones web en tiempo real, que facilita la comunicación bidireccional instantánea entre clientes web y servidores. Esta biblioteca se compone de dos partes: un cliente y un servidor, los cuales poseen una API casi idéntica. (Wikipedia, 2024)

Además, Socket.IO funciona como un protocolo, lo que permite que diversas implementaciones compatibles entre sí puedan comunicarse. La implementación principal se divide en dos partes: un cliente que opera en el navegador y un servidor diseñado para Node.js. Junto con la implementación principal, hay varias otras versiones, incluyendo servidores oficiales en Deno (JavaScript), C++, Java, Python y Swift.

Socket.IO se basa principalmente en el protocolo WebSocket, aunque también utiliza sondeo como alternativa. Si bien se puede emplear solo como un contenedor para WebSockets, ofrece numerosas características adicionales, como mecanismos de latidos y tiempos de espera. (Wikipedia, 2024)

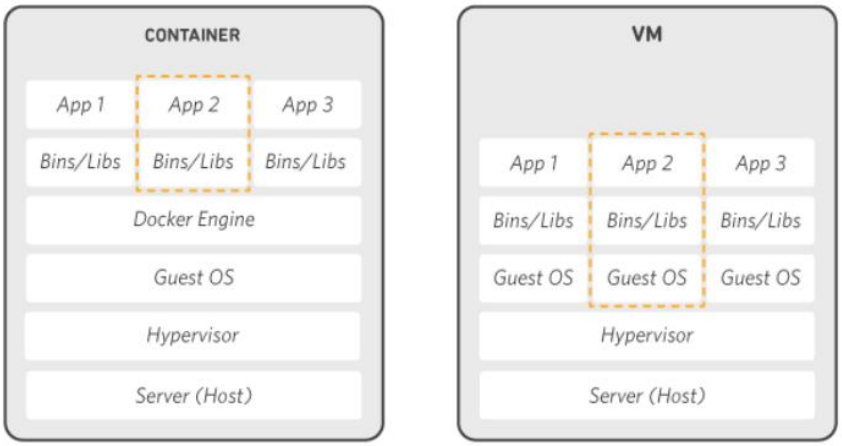
2.4.15 Docker

Docker es una plataforma de software que facilita la creación, prueba e implementación ágil de aplicaciones. Esta herramienta encapsula el software en unidades estandarizadas denominadas contenedores, los cuales contienen todo lo necesario para que el software funcione, incluyendo bibliotecas, herramientas del sistema, el código y el entorno de ejecución. Gracias a Docker, es posible desplegar y escalar aplicaciones rápidamente en cualquier entorno, con la seguridad de que su código funcionará de manera consistente. (AWS, 2024)

Docker ofrece un método estandarizado para ejecutar código. Actúa como un sistema operativo diseñado para contenedores. Al igual que una máquina virtual que virtualiza el hardware de un servidor y elimina la necesidad de gestionar este hardware directamente, los contenedores virtualizan el sistema operativo del servidor. Docker se instala en cada servidor y proporciona comandos simples que permiten crear, iniciar o detener contenedores de manera eficiente.

Ilustración 12: *Funcionamiento de Docker.*

Fuente: (AWS, 2024)



Docker permite la entrega más rápida del código, la estandarización de las operaciones de las aplicaciones, la fácil transferencia de código y el ahorro de costos al optimizar el uso de recursos. Con Docker, se obtiene un único objeto que se puede ejecutar de manera confiable en cualquier entorno. Su sintaxis clara y sencilla proporciona un control total sobre el proceso. Además, su amplia adopción ha dado lugar a un ecosistema robusto de herramientas y aplicaciones que son compatibles con Docker.

Los contenedores de Docker pueden ser utilizados como elementos fundamentales en la creación de aplicaciones y plataformas modernas. Facilita el desarrollo y la ejecución de arquitecturas de microservicios distribuidos, la implementación de código a través de canalizaciones estandarizadas de integración y entrega continua, así como la construcción de sistemas de procesamiento de datos altamente escalables. También permite desarrollar plataformas completamente administradas para los desarrolladores. La colaboración reciente entre AWS y Docker ha simplificado la implementación de artefactos de Docker Compose en Amazon ECS y AWS Fargate. (AWS, 2024)

2.4.16 Kubernetes

Kubernetes es una plataforma de código abierto, portátil y extensible diseñada para gestionar cargas de trabajo y servicios. Facilita la automatización y la configuración declarativa, y cuenta con un ecosistema amplio y en constante expansión, lo que permite un acceso generalizado a soporte, herramientas y servicios relacionados. (Kubernetes, 2024)

El proyecto Kubernetes fue lanzado por Google en 2014, basándose en la experiencia acumulada por la empresa en la ejecución de aplicaciones a gran escala en producción durante más de quince años, junto con las mejores prácticas y conceptos de la comunidad. (Kubernetes, 2024)

Kubernetes presenta varias características y puede conceptualizarse como:

- Una plataforma para contenedores

- Una plataforma para microservicios

- Una solución de nube portátil, entre otras.

Proporciona un entorno de gestión enfocado en contenedores, orquestando la infraestructura de computación, redes y almacenamiento para que los usuarios no tengan que hacerlo. Esto combina la simplicidad de las Plataformas como Servicio (PaaS) con la flexibilidad de la Infraestructura como Servicio (IaaS), permitiendo además la portabilidad entre distintos proveedores de infraestructura.

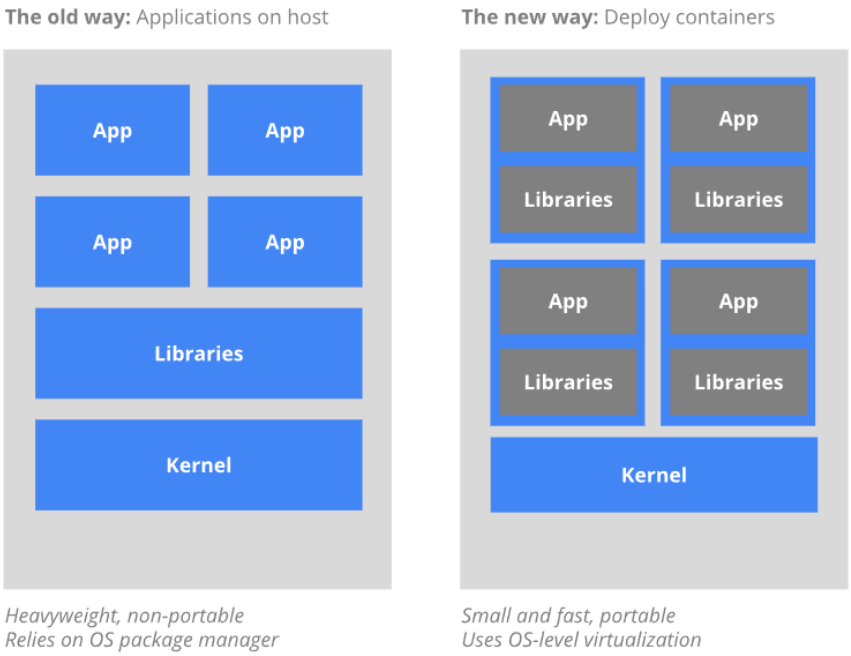
2.4.16.1 Uso de Kubernetes

Tradicionalmente, el proceso de desplegar aplicaciones implicaba su instalación en un servidor mediante el uso del administrador de paquetes del sistema operativo. Esta práctica presentaba desventajas significativas, ya que los ejecutables, configuraciones y bibliotecas estaban intrínsecamente interrelacionados, complicando el ciclo de vida de todos los componentes involucrados. Aunque era posible crear imágenes de máquinas virtuales inmutables para facilitar despliegues y retrocesos predecibles, las máquinas virtuales eran pesadas y carecían de portabilidad. (Kubernetes, 2024)

En contraste, el enfoque moderno implica el uso de contenedores que aprovechan la virtualización a nivel del sistema operativo, en lugar de depender del hardware. Estos contenedores están aislados entre sí y del servidor anfitrión, lo que significa que cada uno posee su propio sistema de archivos, no tiene acceso a los procesos de otros contenedores y se pueden limitar los recursos que utilizan. Además, son más fáciles de crear que las máquinas virtuales y, al no estar vinculados a la

infraestructura o al sistema de archivos del anfitrión, pueden trasladarse fácilmente entre distintas nubes y distribuciones de sistemas operativos.

Ilustración 13: *Visión antigua y moderna del despliegue de dockers,*
Fuente: (Kubernetes, 2024)



Según (Kubernetes, 2024) , se resume los beneficios de usar contenedores en:

- Creación y despliegue ágil de aplicaciones: Permiten una mayor eficiencia al generar imágenes de contenedor en lugar de máquinas virtuales.

- Desarrollo, integración y despliegue continuo: Facilitan la construcción y despliegue frecuentes y confiables de imágenes de contenedor, lo que simplifica los retrocesos, ya que estas son inmutables.

- Desacoplamiento de tareas entre desarrollo y operaciones: Las imágenes de contenedor se crean durante la compilación, lo que desvincula la aplicación de la infraestructura.

- Mejor observabilidad: Se pueden monitorear no solo las métricas del sistema operativo, sino también la salud de la aplicación y otros indicadores relevantes.

- Consistencia en entornos de desarrollo, pruebas y producción: La aplicación presenta el mismo comportamiento en un ordenador portátil que en la nube.

- Portabilidad entre nubes y distribuciones: Funciona en diferentes entornos, como Ubuntu, RHEL, CoreOS, y en centros de datos físicos o plataformas como Google Kubernetes Engine.

- Administración centrada en la aplicación: Eleva el enfoque desde el sistema operativo y el hardware virtualizado hacia la aplicación que opera en un entorno con recursos lógicos.

- Microservicios distribuidos y elásticos: Permite dividir aplicaciones en componentes pequeños e independientes que pueden desplegarse y gestionarse dinámicamente, en contraste con aplicaciones monolíticas que dependen de una sola máquina robusta.

- Aislamiento de recursos: Contribuye a que el rendimiento de la aplicación sea más predecible.

- Eficiencia en la utilización de recursos: Optimiza la eficiencia y densidad en el uso de recursos

2.4.17 Digital Ocean

DigitalOcean es una empresa estadounidense que ofrece servidores virtuales privados y tiene su sede en Nueva York. Esta compañía alquila espacios en centros de datos ya establecidos, ubicados en ciudades como Nueva York, Toronto, Bangalore, Ámsterdam, San Francisco, Londres y Singapur. (Wikipedia, 2018)

DigitalOcean utiliza el término "droplet" para referirse a sus servidores virtuales, específicamente servidores virtuales privados que están disponibles para alquilar. Estos servidores son considerados privados ya que DigitalOcean no se involucra en su instalación o gestión, limitándose a proporcionar imágenes de los sistemas operativos más comunes y sus repositorios de forma local. Esto permite un ahorro en tiempo y transferencia de datos a largo plazo gracias a los repositorios locales. Además, cada droplet perteneciente a un mismo cliente puede comunicarse únicamente con otros droplets del mismo cliente a través de una o más tarjetas de red, lo que permite la creación de una red privada virtual local. Por ejemplo, un droplet podría albergar un servidor MySQL, mientras que otro podría tener WordPress instalado, utilizando dos tarjetas de red virtuales: una para conectar a la base de datos y otra para la exposición a Internet. (Wikipedia, 2018)

DigitalOcean no vende dominios web, pero proporciona la interfaz web necesaria para que los clientes puedan agregar, si lo desean, los dominios que han adquirido a los registros de DNS de DigitalOcean.

La empresa sostiene que sus droplets, que es el término que utilizan para los servidores en la nube, pueden ser aprovisionados en un tiempo promedio de 55 segundos, y no más de dos minutos; cada uno viene inicialmente con una dirección IPv4 fija. Para lograr esto, DigitalOcean ofrece unidades de almacenamiento SSD para mayor velocidad y utiliza virtualización KVM. (Wikipedia, 2018)

2.4.18 GitHub Actions

GitHub Actions es una plataforma que permite la integración y despliegue continuos (IC/DC), facilitando la automatización de procesos de compilación, pruebas y despliegue. Con esta herramienta, puedes diseñar flujos de trabajo que compilen y prueben cada solicitud de cambios en tu repositorio, o bien, desplegar cambios que han sido fusionados en producción. (GitHub, 2024)

Además de sus funciones en DevOps, GitHub Actions te brinda la posibilidad de ejecutar flujos de trabajo en respuesta a otros eventos que ocurran en tu repositorio. Por ejemplo, puedes configurar un flujo de trabajo que añada automáticamente las etiquetas adecuadas cada vez que se genere una nueva propuesta en tu repositorio.

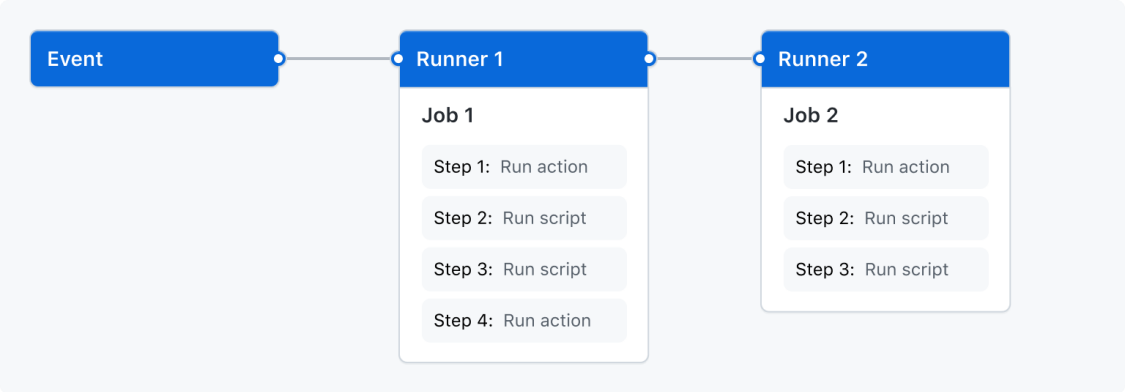
GitHub ofrece máquinas virtuales que funcionan con sistemas operativos Linux, Windows y macOS para ejecutar tus flujos de trabajo, o puedes optar por utilizar ejecutores auto-hospedados en tu propio centro de datos o en una infraestructura en la nube. (GitHub, 2024)

2.4.18.1 Componentes de GitHub Actions

Se puede establecer un flujo de trabajo en GitHub Actions que se active ante la ocurrencia de un evento en el repositorio, como la creación de una solicitud de incorporación de cambios o la apertura de un problema. Este flujo de trabajo incluye uno o más trabajos que pueden ejecutarse de manera secuencial o simultánea. Cada trabajo se lleva a cabo en su propio entorno de máquina virtual o en un contenedor y consiste en varios pasos que pueden ejecutar un script o una acción. Una acción es una extensión reutilizable que ayuda a simplificar el flujo de trabajo. (GitHub, 2024)

Ilustración 14: *Componentes GitHub Actions.*

Fuente: (GitHub, 2024)



a. Flujos de Trabajo

Un flujo de trabajo es un proceso automatizado que se puede configurar para ejecutar uno o más trabajos. Estos flujos se describen mediante un archivo YAML almacenado en tu repositorio, y su ejecución puede ser activada por un evento dentro del mismo, de forma manual o según un horario predefinido.

Los flujos de trabajo se localizan en el directorio. github/workflows de un repositorio. Es posible tener múltiples flujos de trabajo en un solo repositorio, y cada uno puede llevar a cabo diferentes tareas, tales como:

- Compilar y probar solicitudes de incorporación de cambios.

- Desplegar la aplicación cada vez que se crea una nueva versión.

- Añadir una etiqueta al abrir un nuevo problema.

b. Eventos

Un evento es una acción específica dentro de un repositorio que provoca la ejecución de un flujo de trabajo. Por ejemplo, un evento puede surgir en GitHub cuando alguien crea una solicitud de cambios, abre una propuesta o realiza un "commit" en el repositorio. También se puede iniciar un flujo de trabajo de acuerdo a un cronograma, mediante una publicación en una API REST o de manera manual.

c. Trabajos

Un trabajo es un conjunto de pasos dentro de un flujo de trabajo que se ejecuta en el mismo ejecutor. Cada paso puede ser un script de shell o una acción, y se ejecutan en secuencia, siendo dependientes unos de otros. Dado que todos los pasos se ejecutan en el mismo ejecutor, es posible compartir datos entre ellos. Por ejemplo, un paso puede compilar tu aplicación, seguido de otro que realice pruebas sobre la compilación.

Las dependencias entre trabajos se pueden configurar; por defecto, los trabajos se ejecutan en paralelo sin dependencias. Si un trabajo depende de otro, esperará a que el trabajo del que depende haya terminado antes de comenzar su ejecución.

Por ejemplo, puedes establecer varios trabajos de compilación para diferentes arquitecturas sin dependencias entre ellos, y un trabajo de empaquetado que dependa de estas compilaciones. Los trabajos de compilación se ejecutarán en paralelo y, al completarse con éxito, se iniciará el trabajo de empaquetado.

d. Acciones

Una acción es una aplicación personalizada diseñada para la plataforma de GitHub Actions que realiza tareas complejas que se repiten con frecuencia. Utiliza acciones para reducir la cantidad de código repetitivo en tus archivos de flujo de trabajo. Una acción puede clonar el repositorio de Git desde GitHub, configurar el entorno de compilación adecuado o gestionar la autenticación en un proveedor de servicios en la nube.

Puedes desarrollar tus propias acciones o encontrar acciones preexistentes para usar en tus flujos de trabajo en GitHub Marketplace.

e. Ejecutores

Un ejecutor es un servidor que ejecuta flujos de trabajo cuando son activados. Cada ejecutor puede procesar un solo trabajo a la vez. GitHub proporciona ejecutores para Ubuntu Linux, Microsoft Windows y macOS para llevar a cabo los flujos de trabajo. Cada ejecución de un flujo de trabajo se realiza en una máquina virtual recién provisionada.

GitHub también ofrece ejecutores más potentes disponibles en configuraciones más avanzadas. Para más información, consulta "Acerca de los ejecutores más grandes".

Si requieres otro sistema operativo o una configuración de hardware específica, tienes la opción de alojar tus propios ejecutores.

CAPÍTULO 3:

MARCO APLICATIVO

3 MARCO APLICATIVO

3.1 INTRODUCCIÓN

La implementación del proyecto se llevará a cabo utilizando la metodología ágil SCRUM, cuyo propósito principal es orientar el análisis, diseño y desarrollo de un Sistema Integrado de Gestión de Documentos y Emisión de Fichas, aprovechando Kubernetes para la orquestación de contenedores. Este sistema está diseñado específicamente para mejorar y optimizar los procesos del Seguro Médico Delegado de COTEL. Mediante esta metodología, se espera destacar las numerosas ventajas que ofrece la integración de sistemas de información avanzados, permitiendo una gestión más eficiente de la documentación, la emisión de fichas médicas y la centralización de recursos a través de la tecnología de contenedores, mejorando así la infraestructura y facilitando el mantenimiento y escalabilidad de los servicios médicos proporcionados.

3.2 PRODUCT BACKLOG

3.2.1 Visión del producto

El sistema integrado de gestión de documentos y emisión de fichas médicas tiene como propósito central optimizar y unificar los procesos de gestión documental y la emisión de fichas médicas en las áreas de vigencia de derechos y fichaje del Seguro Médico Delegado de COTEL. Este sistema permitirá:

- Eliminar la inconsistencia de datos y documentos: La implementación de un sistema automatizado y centralizado reduce errores en la gestión de la información de los beneficiarios, asegurando que los datos y documentos estén siempre actualizados y disponibles de manera eficiente.

- Mejorar la emisión y control de fichas médicas: El sistema proporcionará una plataforma integrada que agiliza la generación y manejo de fichas médicas, permitiendo un control preciso sobre la vigencia de los derechos y facilitando el trabajo de los empleados.

- Gestionar eficientemente documentos PDF: Un enfoque importante del sistema será la administración adecuada de documentos PDF relacionados con la habilitación de

beneficiarios. Se integrarán funcionalidades para la carga, almacenamiento, visualización y procesamiento de estos documentos, garantizando que se gestionen de manera segura y sean accesibles cuando sea necesario.

- Escalabilidad y fiabilidad con Kubernetes: Al emplear Kubernetes como tecnología de orquestación de contenedores, el sistema será altamente escalable y podrá manejar múltiples procesos simultáneamente, asegurando un alto rendimiento y disponibilidad, incluso durante aumentos en la demanda.

- Facilidad de mantenimiento y actualización: Al modularizar el sistema y separar los servicios de fichaje y vigencia de derechos en APIs REST independientes, se garantiza una integración y actualización fácil de los componentes, así como una comunicación clara entre los diferentes servicios.

- Visualización de información y creación de reportes: El sistema ofrecerá herramientas avanzadas para la visualización de la información de los beneficiarios y sus registros. Además, generará reportes detallados sobre el estado de las fichas, la vigencia de los derechos y otros indicadores clave, brindando a los usuarios una visión completa y transparente del sistema.

- Reducir los tiempos de respuesta y mejorar la experiencia del usuario: El uso de tecnologías modernas como contenedores y APIs permitirá que los usuarios internos (empleados del seguro) interactúen de manera más rápida y eficiente con el sistema, mejorando la experiencia tanto del personal como de los beneficiarios del seguro.

El resultado será un sistema robusto y fiable que no solo mejorará la eficiencia operativa del seguro médico delegado, sino que también asegurará la precisión en la gestión de datos, la correcta administración de documentos PDF, la visualización clara de la información y la creación de reportes útiles para la toma de decisiones, contribuyendo a la satisfacción de los beneficiarios.

3.2.1.1 Criterios de definición de funcionalidades clave

Para garantizar que las funcionalidades clave del sistema integrado de gestión de documentos y emisión de fichas médicas estén alineadas con la visión del producto, se han establecido los siguientes criterios:

a. Alineación con la visión del producto

Cada funcionalidad debe contribuir a los objetivos generales de la visión, asegurando que se alineen con las siguientes metas:

- Optimización de la gestión documental: Las funcionalidades deben facilitar la organización y el acceso a los documentos necesarios para la habilitación de beneficiarios, eliminando inconsistencias y mejorando la precisión en la información.

- Mejora de la emisión de fichas médicas: Las funcionalidades deben agilizar y simplificar el proceso de generación y gestión de fichas médicas, permitiendo un control eficiente sobre la vigencia de los derechos de los beneficiarios.

- Escalabilidad y fiabilidad con Kubernetes: Al emplear Kubernetes como tecnología de orquestación de contenedores, el sistema será altamente escalable y capaz de manejar múltiples procesos simultáneamente, asegurando un alto rendimiento y disponibilidad, incluso durante aumentos en la demanda.

- Facilidad de mantenimiento y actualización: Al modularizar el sistema y separar los servicios de fichaje y vigencia de derechos en APIs REST independientes, se garantiza una integración y actualización fácil de los componentes, así como una comunicación clara entre los diferentes servicios.

- Visualización de información y creación de reportes: El sistema ofrecerá herramientas avanzadas para la visualización de la información de los beneficiarios y sus registros. Además, generará reportes detallados sobre el estado de las fichas, la vigencia de los derechos y otros indicadores clave, brindando a los usuarios una visión completa y transparente del sistema.

- Reducción de tiempos de respuesta y mejora en la experiencia del usuario: El uso de tecnologías modernas como contenedores y APIs permitirá que los usuarios internos (empleados del seguro) interactúen de manera más rápida y eficiente con el sistema, mejorando la experiencia tanto del personal como de los beneficiarios del seguro.

b. Relevancia para los usuarios

Las funcionalidades deben ser relevantes y prácticas para los empleados del seguro médico, asegurando que realmente mejoren su eficiencia en las tareas diarias.

c. Viabilidad técnica

Se evaluará la factibilidad de implementar cada funcionalidad en el contexto técnico del sistema, tomando en cuenta las tecnologías empleadas y los recursos disponibles.

d. Prioridad basada en impacto

Se priorizarán aquellas funcionalidades que tengan un mayor impacto en la operación del sistema y en la satisfacción de los usuarios.

e. Escalabilidad y mantenibilidad

Las funcionalidades deberán permitir la escalabilidad del sistema y su fácil mantenimiento a lo largo del tiempo, facilitando futuras actualizaciones.

f. Facilidad de uso

Se tendrá en cuenta la experiencia del usuario, asegurando que las funcionalidades sean intuitivas y accesibles para todos.

g. Capacidad de medición

Cada funcionalidad incluirá métricas que permitan evaluar su efectividad y alineación con los objetivos del sistema, facilitando ajustes en función de los resultados.

3.2.1.2 Relación entre la visión del producto y la estrategia del Product Backlog

La visión del producto proporciona un marco claro para la estrategia del Product Backlog. Cada elemento del backlog se deriva de la necesidad de alcanzar los objetivos establecidos en la visión, asegurando que se prioricen las funcionalidades que mejoren la gestión documental y optimicen la emisión de fichas médicas. La estrategia del backlog también se centra en la satisfacción del usuario, garantizando que las funcionalidades no solo sean técnicamente viables, sino que también respondan a las necesidades reales de los empleados del seguro médico.

Al integrar continuamente la retroalimentación de los usuarios y alinear las prioridades con la visión del producto, el Product Backlog se convierte en una herramienta dinámica que evoluciona para satisfacer las demandas del sistema y sus usuarios.

3.2.2 Identificación de requerimientos iniciales

3.2.2.1 Proceso para identificar las necesidades y expectativas del cliente y stakeholders

La identificación de los requisitos iniciales se llevó a cabo mediante un enfoque integral y colaborativo, involucrando a las partes interesadas clave, incluidas las autoridades del Seguro Médico Delegado de COTEL, empleados de las áreas de vigencia de derechos y fichaje, así como el departamento de sistemas. Este proceso se desarrolló en varias etapas:

- Recolección de Documentación Firmada: Se obtuvieron hojas firmadas por cada empleado de las áreas de vigencia de derechos y fichaje, en las que se detallaban los requerimientos iniciales y generales que el sistema debía cumplir. Esta documentación proporcionó una base sólida para comprender las expectativas de los usuarios y garantizar su alineación con las necesidades del sistema. (ver Anexo C)

- Análisis de Manuales de Proceso y Procedimientos: Se revisó un manual de procesos y procedimientos proporcionado por COTEL, que contenía detalles sobre las operaciones actuales, como la habilitación de beneficiarios, la desafiliación, el registro de documentos, la habilitación para la atención, el proceso de atención, la emisión de

fichas y la entrega de expedientes clínicos. Este análisis permitió identificar las etapas críticas del flujo de trabajo y las áreas que requerían optimización. (ver Anexo D)

- Entrevistas con Stakeholders: Se realizaron entrevistas con representantes del departamento de sistemas, así como con los responsables de vigencia de derechos y fichaje en las sucursales central, Alto y Zona Sur. Estas entrevistas fueron esenciales para profundizar en los desafíos que enfrentan en sus actividades diarias y sus expectativas respecto a la funcionalidad del nuevo sistema.

- Reuniones de Validación: Se llevaron a cabo reuniones de validación con las autoridades del Seguro Médico Delegado, donde se presentaron los hallazgos preliminares y se discutieron posibles soluciones para abordar las necesidades identificadas.

3.2.2.2 Metodologías usadas para recopilar los requisitos

Para la recopilación de requisitos, se utilizaron las siguientes metodologías:

- Recolección de Documentación: Recopilación de hojas firmadas por los empleados que contenían requisitos iniciales y generales.

- Análisis de Documentación Existente: Revisión del manual de procesos y procedimientos para entender el flujo de trabajo actual.

- Entrevistas: Realización de entrevistas estructuradas con empleados de diferentes áreas para obtener información cualitativa sobre sus necesidades y desafíos.

- Reuniones de Validación: Discusiones con las autoridades del Seguro Médico Delegado para validar los requisitos y asegurar la alineación con los objetivos del proyecto.

3.2.3 Refinamiento de requisitos

En esta etapa, se procedió a clarificar y ajustar cada uno de los requerimientos, siguiendo las pautas descritas a continuación:

- Aclaración de dudas: A través de reuniones y entrevistas con las áreas involucradas y la gerencia, se resolvieron las dudas sobre los requerimientos presentados en los documentos entregados por las áreas, tomando en cuenta los objetivos y límites del proyecto, el manual de procesos y procedimientos, los objetivos institucionales, la normativa vigente de la ASSUS como ente regulador y los objetivos de la gerencia.

- Refinamiento según los objetivos y límites del proyecto: Se eliminaron los requerimientos que no se alineaban con los objetivos y límites definidos del proyecto, limitando el alcance a la construcción de un sistema integrado de gestión documental y emisión de fichas médicas para las áreas de vigencia de derechos y fichaje del SMDC.

- Refinamiento según el manual de procesos y procedimientos: Se descartaron aquellos requerimientos que no pertenecían al ámbito de las áreas involucradas ni formaban parte de los procesos que los usuarios del sistema deben seguir.

- Refinamiento según los objetivos institucionales: Se eliminaron los requerimientos que no guardaban relación con los objetivos institucionales, como los que implicaban conexiones externas con módulos de terceros o la implementación de software de pago no contemplado en el soporte del sistema, aunque pudieran beneficiar a ciertos usuarios.

- Refinamiento según la normativa vigente de ASSUS: Se descartaron los requerimientos que incumplían las normativas tecnológicas establecidas por la ASSUS, dado que es el ente regulador del SMDC, y el sistema está obligado a seguir dichas disposiciones.

- Refinamiento según los objetivos gerenciales: Se eliminaron los requerimientos que no se alineaban con los objetivos de la gerencia, como personalizaciones que solo beneficiarían a ciertos usuarios o funcionalidades innecesarias para el correcto funcionamiento del sistema.

3.2.3.1 Clasificación y elaboración de las tablas de requerimientos

La obtención de los requerimientos es un paso fundamental en el proceso de desarrollo, ya que permite clasificar y estructurar de manera efectiva las necesidades del proyecto. Este proceso implica la recopilación meticulosa de información que refleja las expectativas y

necesidades de los usuarios y las partes interesadas. A partir de esta recopilación, se procede a la clasificación y construcción de listas de requerimientos detallados, que se presentan en las tablas.

Cada requerimiento se clasifica en requerimientos funcionales y no funcionales, lo que permite una mejor gestión de los recursos y un enfoque más sistemático en el proceso de implementación.

A continuación, se detallan las tablas que reflejan esta clasificación y elaboración de requerimientos, proporcionando una visión clara y accesible de las necesidades que guiarán el desarrollo del sistema.

Tabla 10: *Requisitos funcionales generales.*

Fuente: Elaboración propia

Ref.	Requisitos funcionales generales
RGN1	Exportar datos a Excel
RGN2	Generación de informes según formato establecido por área
RGN3	Módulos independientes para cada área, cada uno gestionando su propia base de datos.
RGN4	Asegurar alta resiliencia y permitir escalamiento horizontal, de modo que se facilite la incorporación de nuevos módulos sin afectar el rendimiento del sistema
RGN5	Implementar un sistema de gestión centralizada para las APIs internas, que facilite el control, la seguridad y la monitorización del acceso a los recursos de los módulos

Tabla 11: *Requisitos funcionales de vigencia de derechos.*

Fuente: Elaboración propia

Ref.	Requisitos funcionales de vigencia de derechos
RV1	Registro de datos generales de titulares y beneficiarios
RV2	Registro de altas y bajas de los asegurados
RV3	Inclusión de beneficiarios (esposas, hijos y padres)
RV4	Escaneo y almacenaje de documentos de los asegurados
RV5	Modificación y almacenamiento de documentación periódica

RV6	Detector de alarmas de documentos caducados
RV7	Actualización e inclusión de nuevos documentos requeridos
RV8	Control de acceso según nivel de jerarquía de usuario
RV9	Configurar el documento asignado a cada tipo de beneficiario en el sistema

Tabla 12: *Requisitos funcionales de fichaje.*

Fuente: Elaboración propia

Ref.	**Requisitos funcionales de fichaje**
RF1	Leer credenciales con pistola lacer
RF2	Búsqueda por carnet y nombre del beneficiario
RF3	Registro de fichas emitidas
RF4	Visualización de alertas

Tabla 13: *Requisitos funcionales de sistemas.*

Fuente: Elaboración propia

Ref.	**Requisitos funcionales de sistemas**
RS1	Registro de usuarios
RS2	Registro de solicitudes y entrega de carnets

Tabla 14: *Requisitos funcionales de gerencia.*

Fuente: Elaboración propia

Ref.	**Requisitos funcionales de gerencia**
RG1	Registro de cambios en horarios de atención de los médicos
RG2	Visualización de turnos según ficha en TV
RG3	Gaceta de emisión de fichas para beneficiarios
RG4	Sitio web para agendar vistas por internet

Tabla 15: *Requisitos no funcionales.*

Fuente: Elaboración propia

Referencia	Requisitos no funcionales
Interfaz de usuario	El sistema debe ser ejecutado como una aplicación de escritorio en las maquinas autorizadas.
Interfaz de respuesta	La demora no debe exceder los 10 segundos
Acceso	El acceso al sistema debe ser restringido y controlado
Portabilidad	El sistema debe funcionar en un sistema operativo Windows 7 para adelante.

3.2.4 Historias de usuario

3.2.4.1 Estructura

Las historias de usuario son descripciones breves y simples de una funcionalidad desde la perspectiva del usuario final. Siguiendo la estructura "Como [tipo de usuario], quiero [acción] para [beneficio]", se busca capturar lo que el usuario necesita de manera clara y concisa. Esta estructura garantiza que las historias estén centradas en el usuario y alineadas con los objetivos de la institución.

Formato:

- Como [tipo de usuario]

- Quiero [acción]

- Para [beneficio o razón]

3.2.4.2 Elaboración

A partir de los requerimientos funcionales encontrados en las tablas: Tabla 12, Tabla 13, Tabla 14, Tabla 15 y Tabla 16. Y el formato planteado, se realizan el siguiente tipo de historias de usuario:

Ilustración 15: *Ejemplo de formato de historia de usuario general.*

Fuente: Elaboración propia

H1	**Exportar datos a Excel**
Como usuarios del sistema, **queremos** exportar los datos del sistema a Excel **para** poder generar informes personalizados y gráficos en esta herramienta.	

Ilustración 16: *Ejemplo de formato de historia de usuario de un trabajador.*

Fuente: Elaboración propia

H3	**Registro de datos generales de titulares y beneficiarios**
Como encargado de vigencia de derechos, **quiero** poder crear, leer, actualizar y eliminar los datos generales de titulares y beneficiarios **para** mantener la información siempre actualizada y precisa.	

Para ver todas las historias de usuario elaboradas a partir de los requerimientos encontrados en las tablas: Tabla 12, Tabla 13, Tabla 14, Tabla 15 y Tabla 16 (ver Anexo D).

3.2.5 Priorización

3.2.5.1 Criterios de priorización para historias de usuario

Se utilizaron las técnicas de MoSCoW para priorizar las historias de usuario, asignando a cada una su nivel de importancia. Al tratarse de un análisis de los requerimientos iniciales, se determinó que el 80% de los requisitos eran esenciales ("Must have") y el 20% restante deseables ("Should have"). Con base en esta clasificación, se priorizaron las tareas según su importancia y el orden de desarrollo de cada módulo. Así, las historias de usuario fueron organizadas conforme a la secuencia de implementación de los módulos, siguiendo el siguiente orden:

- Módulo de sistemas

- Módulo de vigencia de derechos

- Módulo de fichaje

- Módulo de gerencia

Este orden de desarrollo se generó a partir de los requerimientos de cada área. Se comprendió que el módulo de vigencia de derechos depende de las configuraciones y funciones proporcionadas por el módulo de sistemas; de igual forma, el módulo de fichaje no puede operar sin el módulo de vigencia de derechos y sistemas activos, y el módulo gerencial no funcionaría sin la implementación de los módulos anteriores.

A partir de este enfoque se genero la siguiente lista con las historias de usuario debidamente priorizadas:

Tabla 16: *Requerimientos según prioridad.*

Fuente: Elaboración propia

Ref.	Historia	Prioridad
H1	Exportar datos a Excel	Baja
H2	Generación de informes según formato establecido por área	Baja
H3	Registro de datos generales de titulares y beneficiarios	Muy Alta
H4	Registro de altas y bajas de los asegurados	Alta
H5	Inclusión de beneficiarios (esposas, hijos y padres)	Alta
H6	Escaneo y almacenaje de documentos de los asegurados	Alta
H7	Modificación y almacenamiento de documentación periódica	Media
H8	Detector de alarmas de documentos caducados	Baja
H9	Actualización e inclusión de nuevos documentos requeridos	Media
H10	Control de acceso según nivel de jerarquía de usuario	Media
H11	Configurar el documento asignado a cada tipo de beneficiario en el sistema	Alta

H12	Leer credenciales con pistola láser	Baja
H13	Búsqueda por carnet y nombre del beneficiario	Media
H14	Registro de fichas emitidas	Muy Alta
H15	Visualización de alertas	Baja
H16	Registro de usuarios	Media
H17	Registro de solicitudes y entrega de carnets	Media
H18	Registro de cambios en horarios de atención de los médicos	Media
H19	Visualización de turnos según ficha en TV	Baja
H20	Gaceta de emisión de fichas para beneficiarios	Baja
H21	Sitio web para agendar visitas por internet	Media
H22	Diseño y desarrollo de arquitectura	Muy Alta

3.2.6 Estimación de esfuerzos

Para calcular la estimación de esfuerzo para cada una de las historias de usuario obtenidas en la Tabla 18, se debe desglosar cada una de las historias en operaciones y funcionalidades más pequeñas, luego debemos asignarle un SLOC las micro-funcionalidades, nombre sugerido en este proyecto, según el Anexo F.

Como resultado obtendremos una historia con un SLOC total que se usará para el cálculo de esfuerzos por historia de usuario. En la Tabla 20 se puede observar un ejemplo de esta calculo.

***Tabla 17**: Tabla de ejemplo de cálculo de LOC para una historia de usuario.*

Fuente: Elaboración propia

H3	**Registro de datos generales de titulares y beneficiarios**	**SLOC: 960**
Funcionalidades Backend		LOC
HF3.1	API de adición	60

HF3.2	API de lectura	50
HF3.3	API de edición	70
HF3.4	API de borrado	40
Total SLOC Backend		**220**
Funcionalidades Frontend		LOC
HB3.1	Formulario de adición	120
HB3.2	Tabla para listar los datos	150
HB3.3	Formulario de edición	150
HB3.4	Formulario de borrado	80
HB3.5	Manejo de peticiones HTTP 60 SLOC x 4 FNS	240
Total SLOC Frontend		**740**

Para ver en detalle el análisis todas las historias de usuario (ver Anexo G)

A partir de este análisis se procede a listar las historias de usuario con su respectivo SLOC como se muestra en la Tabla 20.

Tabla 18: *Historias de usuario con su respectivo SLOC.*

Fuente: Elaboración propia

Ref.	Historia	Prioridad	SLOC
H1	Exportar datos a Excel	Baja	100
H2	Generación de informes según formato establecido por área	Baja	1295
H3	Registro de datos generales de titulares y beneficiarios	Muy Alta	960
H4	Registro de altas y bajas de los asegurados	Alta	750
H5	Inclusión de beneficiarios (esposas, hijos y padres)	Alta	960
H6	Escaneo y almacenaje de documentos de los asegurados	Alta	1040
H7	Modificación y almacenamiento de documentación periódica	Media	430

H8	Detector de alarmas de documentos caducados	Baja	610
H9	Actualización e inclusión de nuevos documentos requeridos	Media	1020
H10	Control de acceso según nivel de jerarquía de usuario	Media	1200
H11	Configurar el documento asignado a cada tipo de beneficiario en el sistema	Alta	340
H12	Leer credenciales con pistola láser	Baja	120
H13	Búsqueda por carnet y nombre del beneficiario	Media	610
H14	Registro de fichas emitidas	Muy Alta	990
H15	Visualización de alertas	Baja	840
H16	Registro de usuarios	Media	990
H17	Registro de solicitudes y entrega de carnets	Media	990
H18	Registro de cambios en horarios de atención de los médicos	Media	1190
H19	Visualización de turnos según ficha en TV	Baja	930
H20	Gaceta de emisión de fichas para beneficiarios	Baja	770
H21	Sitio web para agendar visitas por internet	Media	1640
H22	Diseño y desarrollo de arquitectura	Muy Alta	490

Al encontrarse en la fase de planificación, se procedió a utilizar el modelo de diseño temprano de COCOMO II, se empleó la Ecuación 11 para calcular los puntos de función o estimación de esfuerzo:

$$PM_{nominal} = A * (KSLOC)^{B}$$

Donde:

- A es constante que captura los efectos lineales sobre el esfuerzo de acuerdo a la variación del tamaño, (A=2.94).

- KSLOC es la sumatoria de todos los SLOC de las historias de usuario dividido entre 1000.

- B es el factor exponencial de escala, toma en cuenta las características relacionadas con las economías y deseconomías de escala producidas cuando un proyecto de software incrementa su tamaño. El cual se calcula con la Ecuación 12 junto con la Tabla 6:

$$B = 1.01 + 0.01 * \sum_{j=1}^{5} W_J$$

- Suma de los W_j: Se asignaron los siguientes valores nominales a los factores:

PREC (Precedencia): 4.96 (Muy Bajo)
FLEX (Flexibilidad en el desarrollo): 2.03 (Nominal)
RESL (Resolución de arquitectura y riesgo): 2.83 (Nominal)
TEAM (Cohesión del equipo): 2.19 (Nominal)
PMAT (Madurez del proceso): 4.68 (Bajo)

$$\sum_{j=1}^{5} W_J = 4.96 + 2.03 + 2.83 + 2.19 + 4.68 = 16.69$$

Entonces:

$$B = 1.01 + 0.01 * 16.69 = 1.1769$$

Consecuentemente, se procede a calcular el $PM_{nominal}$ para cada una de las historias de usuario, calculando de la siguiente manera:

$$PM_{nominal} = 2.94 * (18265/ 1000)^{1.1769}$$

$$\mathbf{PM_{nominal} = 89{,}77 \text{ persona- meses}}$$

3.2.7 Estimación de tiempo

Una vez calculado el esfuerzo en persona-meses, puedes estimar el tiempo de desarrollo total (TDEV) usando:

Ecuación 11

$$TDEV = C \times (PM)^F$$

Donde:

- C es una constante (usualmente 3.67).
- F es el exponente de tiempo, que usualmente varía de 0.28 a 0.91 según el tipo de proyecto. Para el proyecto 0.53 teniendo en cuenta:

-

Ecuación 12

$$F = B + 0.01 \times \sum(\text{Factores de Escala})$$

Tabla 19: Tabla guia factores de escala de costo

Fuente: Elaboración propia.

Factor de Escala	Nivel de Evaluación	Valor Asignado
PREC (Precedentedness)	Muy Bajo (VL)	6.20
	Bajo (L)	4.94
	Nominal (N)	3.72
	Alto (H)	2.48
	Muy Alto (VH)	1.24
FLEX (Development Flexibility)	Muy Bajo (VL)	5.07
	Bajo (L)	4.05
	Nominal (N)	3.04
	Alto (H)	2.03
	Muy Alto (VH)	1.01
RESL (Architecture / Risk Resolution)	Muy Bajo (VL)	7.07
	Bajo (L)	5.65
	Nominal (N)	4.24
	Alto (H)	2.83
	Muy Alto (VH)	1.41
TEAM (Team Cohesion)	Muy Bajo (VL)	5.48
	Bajo (L)	4.38
	Nominal (N)	3.29
	Alto (H)	2.19
	Muy Alto (VH)	1.10
PMAT (Process Maturity)	Muy Bajo (VL)	7.80
	Bajo (L)	6.24
	Nominal (N)	4.68
	Alto (H)	3.12
	Muy Alto (VH)	1.56

Tabla 20: Tabla factores de escala propuestos

Fuente: Elaboración propia.

Factor de Escala	Nivel Evaluado	Valor Asignado
PREC	Muy Bajo (VL)	3,72
FLEX	Muy Bajo (VL)	3,04
RESL	Alto	4,24
TEAM	Nominal	3,29
PMAT	Alto	4,68
Total Ajustes		18,97

$$F = 0.28 + 0.01 \times 31.62$$

$$\mathbf{F = 0,4697}$$

Remplazando:

Ecuación 13

$$TDEV = C \times (PM)^{F}$$

$$TDEV = 3.67 \times (107,43)^{\,0,4697}$$

$$\mathbf{TDEV = 30,34\ meses}$$

3.2.8 Estimación de personal necesario

Teniendo el tiempo de desarrollo se procedió a usar la siguiente fórmula para el cálculo del número de programadores:

Número de programadores = TDEV (en meses) / Esfuerzo total en persona-meses

Remplazando:

Número de programadores = 30,34 / 89,77

Número de programadores = 2.96

Redondeando:

Número de programadores = 3 programadores

3.2.9 Estimación del costo del sistema

Finalmente, para obtener el costo del proyecto, se multiplica el esfuerzo estimado (en persona-meses) por el costo promedio de un desarrollador por mes.

Ecuación 14

$$\text{Costo Total} = PM \times \text{Costo por Persona-Mes}$$

$$\text{Costo Total} = 157{,}711062 \times 350$$

Costo Total = 19750,14137 $us.

3.2.10 Definición de esfuerzos por historias

Consecuentemente se procedió a calcular el esfuerzo necesario para cada historia de usuario con una porción de la historia de la siguiente manera, por ejemplo:

$$SLOC_{Total} = 18265$$

$$SLOC_{Hisotria\text{-}2} = 1295$$

Para el cálculo se hace de la siguiente manera:

$$\text{Proporción de la historia} = SLOC_{Hisotria\text{-}2} / SLOC_{Total}$$

$$PM_{historia} = \text{Proporción de la historia} \times PM_{nominal}$$

Remplazando, por ejemplo, para el cálculo de la historia 2:

$$\text{Proporción de la historia-2} = 1295 / 18265$$

$$\text{Proporción de la historia-2} = 0{,}070090062962$$

$$PM_{historia\text{-}2} = 0{,}070090062962 \times 89{,}77$$

$$PM_{historia\text{-}2} = 6{,}36475 \text{ persona-mes}$$

Realizando el cálculo, obtenemos la siguiente tabla con las historias de usuario y sus respectivos puntos de esfuerzo:

Tabla 21: *Historias de usuario con puntos de esfuerzo.*

Fuente: Elaboración propia

Ref.	Historia	Prioridad	$PM_{historia}$
H1	Exportar datos a Excel	Baja	0,491504899
H2	Generación de informes según formato establecido por área	Baja	6,364988446
H3	Registro de datos generales de titulares y beneficiarios	Muy Alta	4,718447033
H4	Registro de altas y bajas de los asegurados	Alta	3,686286745
H5	Inclusión de beneficiarios (esposas, hijos y padres)	Alta	4,718447033
H6	Escaneo y almacenaje de documentos de los asegurados	Alta	5,111650952
H7	Modificación y almacenamiento de documentación periódica	Media	2,113471067
H8	Detector de alarmas de documentos caducados	Baja	2,998179886
H9	Actualización e inclusión de nuevos documentos requeridos	Media	5,013349973
H10	Control de acceso según nivel de jerarquía de usuario	Media	5,898058791
H11	Configurar el documento asignado a cada tipo de beneficiario en el sistema	Alta	1,671116658
H12	Leer credenciales con pistola láser	Baja	0,589805879
H13	Búsqueda por carnet y nombre del beneficiario	Media	2,998179886
H14	Registro de fichas emitidas	Muy Alta	4,865898503
H15	Visualización de alertas	Baja	4,128641154
H16	Registro de usuarios	Media	4,865898503
H17	Registro de solicitudes y entrega de carnets	Media	4,865898503

H18	Registro de cambios en horarios de atención de los médicos	Media	5,848908301
H19	Visualización de turnos según ficha en TV	Baja	4,570995563
H20	Gaceta de emisión de fichas para beneficiarios	Baja	3,784587724
H21	Sitio web para agendar visitas por internet	Media	8,060680348
H22	Diseño y desarrollo de la arquitectura	Muy Alta	2,408374006

3.2.11 Definición de criterios de aceptación

Los criterios de aceptación se definieron de manera individual, analizando las historias de usuario y sus requisitos. Utilizando la metodología WebQem, se basaron en las métricas del modelo de calidad ISO 9126-1, considerando las siguientes características clave: funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Durante este proceso, se realizaron las siguientes actividades:

- Revisión de Historias de Usuario: Se analizaron las historias de usuario propuestas, asegurando que cada una reflejara las necesidades del cliente y el contexto del sistema de gestión de documentos y emisión de fichas médicas.

- Identificación de Métricas: Se definieron métricas relevantes para cada historia basadas en las dimensiones de la calidad del modelo ISO, lo que permitió una mejor comprensión de las expectativas del usuario y garantizó que el sistema cumpla con los estándares requeridos.

- Validación Continua: Se mantuvo un proceso de revisión continua, donde los criterios se ajustaron según el progreso del desarrollo y la evaluación de las funcionalidades implementadas.

3.2.11.1 Criterios de aceptación por historia de usuario

Basándose en la metodología WebQEM se procedió a realizar criterios de aceptación a cada una de las historias de usuario, dando como resultado tarjetas con el siguiente formato:

Tabla 22: *Ejemplo de historia de usuario con sus respectivos criterios de aceptación.*

Fuente: Elaboración propia

H3	Registro de datos generales de titulares y beneficiarios
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir crear, leer, actualizar y eliminar datos de titulares y beneficiarios. - Debe validarse que los datos ingresados cumplan con las reglas establecidas.
Confiabilidad	- Las operaciones CRUD deben completarse sin errores y ser reversibles. - Los datos deben persistir en la base de datos después de las operaciones.
Usabilidad	- La interfaz debe ser fácil de navegar para realizar operaciones CRUD. - Debe haber mensajes de error claros para entradas inválidas.
Eficiencia	- Las operaciones CRUD deben completarse en menos de 5 segundos. - El sistema debe manejar múltiples solicitudes CRUD simultáneamente.
Mantenibilidad	- El código para las operaciones CRUD debe estar bien documentado. - Las estructuras de datos deben ser fácilmente modificables.
Portabilidad	- La lógica de CRUD debe ser compatible con futuras extensiones del sistema.

Para ver todos los criterios de aceptación de cada historia de usuario (ver Anexo I)

3.2.12 Gestión continua

3.2.12.1 Estrategia para mantener y actualizar el Product Backlog a lo largo del proyecto

El Product Backlog se mantuvo actualizado mediante revisiones continuas en cada Sprint Planning y durante las sesiones de refinamiento de backlog. La estrategia central consistió en realizar revisiones periódicas del backlog para asegurar que las historias de usuario estuvieran alineadas con los objetivos del proyecto y las prioridades del negocio. Durante estas revisiones, se ajustaron las descripciones de las historias, se desglosaron elementos

grandes en tareas más pequeñas (épicas en historias) y se actualizó la priorización según la capacidad del desarrollo.

3.2.12.2 Proceso para agregar nuevas historias y re-priorizar según el progreso o nuevos requisitos

Nuevas historias de usuario se añadieron al backlog siguiendo un proceso definido de recopilación de requisitos. Se recibían las nuevas solicitudes de los stakeholders, que luego eran analizadas para asegurar su viabilidad. Cada nueva historia se describía claramente con su valor de negocio y se le asignaba un nivel de prioridad usando el criterio MoSCoW (Must have, Should have, Could have, Won't have). Además, se re-priorizaba las historias existentes en función de los cambios en los requisitos del negocio o descubrimientos durante el desarrollo.

3.2.12.3 Proceso de cambios o ajustes en las prioridades del negocio

A lo largo del proyecto, los cambios en las prioridades del negocio fueron manejados con flexibilidad. En las reuniones semanales con la gerencia, se discutieron los cambios estratégicos del negocio y cómo impactarían en el backlog. Esto permitió la re-priorización inmediata de historias críticas y el aplazamiento de aquellas menos urgentes. También se usaron Sprints de ajuste o "hardening sprints" cuando era necesario adaptar grandes partes del sistema a nuevas necesidades sin afectar el progreso general.

3.2.13 Herramientas

Para gestionar y actualizar el Product Backlog durante el desarrollo del proyecto de grado, se utilizó la herramienta Notion. Esta plataforma resultó ideal debido a su flexibilidad para organizar, documentar y priorizar tareas de manera individual. Notion permitió crear un backlog personalizado en el que se gestionaron las historias de usuario, tareas y entregables del proyecto de forma visual y estructurada.

El uso de Notion ofreció varias ventajas clave:

- Organización sencilla y flexible: Notion permite crear bases de datos personalizables, lo que facilitó la estructuración de las historias de usuario y tareas del proyecto.

- Visualización clara del progreso: Se utilizaron vistas como listas y tableros Kanban para tener una visión clara del estado de cada tarea (pendiente, en progreso, completada).

- Documentación integrada: Notion permite agregar descripciones detalladas, adjuntar documentos y agregar enlaces, lo que hizo que la planificación del backlog fuera más precisa y fácil de gestionar.

- Adaptabilidad y simplicidad: Notion fue una solución liviana y adecuada, ya que no requería las funcionalidades más avanzadas de herramientas como Jira o Trello.

- Historial y seguimiento personal: Gracias a la capacidad de agregar comentarios y registros de cambios, fue posible documentar el progreso de cada tarea y las decisiones tomadas durante el desarrollo.

3.3 PLANIFICACIÓN

Se realizó un análisis exhaustivo de las historias de usuario, que se pueden ver detalladas en la Tabla 22. Basándonos en el orden de prioridad de cada historia de usuario y en su complejidad, se decidió dividir el proyecto en un total de **15 sprints**.

La planificación de los sprints fue realizada tomando en cuenta los siguientes factores:

3.3.1 Priorización de los entregables

Las historias de usuario que aportan más valor inmediato al sistema, como la arquitectura, el registro de beneficiarios, gestión de documentos y la emisión de fichas médicas, fueron priorizadas y asignadas a los primeros sprints.

3.3.2 Capacidad de desarrollo

Siendo un proyecto individual, la cantidad de trabajo a completar en cada sprint fue ajustada para evitar sobrecargas. Se optó por un enfoque realista en cuanto a la capacidad de entrega de funcionalidades dentro de los plazos establecidos.

3.3.3 Estimación de esfuerzo

Para cada sprint se realizó una estimación de esfuerzo basada en el análisis de las tareas a realizar. Utilizando la métrica de $PM_{historia}$ (Persona-Mes) de cada historia de usuario involucrada en el sprint, se calculó el esfuerzo necesario para completar cada funcionalidad. El valor de $PM_{historia}$ fue convertido a semanas de trabajo para su posterior representación en puntos de esfuerzo, de manera que los sprints se puedan organizar de forma ágil y adaptativa.

Proceso de estimación:

- Conversión a semanas de trabajo: Los valores de $PM_{historia}$ fueron convertidos a semanas, utilizando la relación de 1 mes ≈ 4.33 semanas.

- Asignación de puntos de esfuerzo: Posteriormente, los esfuerzos fueron representados como puntos de esfuerzo, donde 1 punto de esfuerzo equivale aproximadamente a una semana de trabajo.

3.3.4 Criterios de priorización de tareas

Para la planificación del sprint, se establecieron los siguientes criterios de priorización para las tareas que surjan de cada historia de usuario asignada a cada sprint:

- **Dependencias**: Se priorizaron las tareas que no tienen prerequisitos o son necesarias para completar otras tareas. Esto asegura un flujo de trabajo eficiente y evita bloqueos.

- **Impacto en el Proyecto**: Se evaluó el impacto de cada tarea en el éxito del proyecto. Las tareas fundamentales para el funcionamiento del sistema en producción o desarrollo fueron consideradas de mayor prioridad.

- **Facilidad de Implementación**: Se tomó en cuenta la complejidad y el tiempo estimado para completar cada tarea. Las tareas que son más fáciles de implementar y pueden aportar valor rápidamente recibieron mayor prioridad.

- **Requisitos de Pruebas**: Las tareas relacionadas con la creación y ejecución de pruebas fueron consideradas críticas para garantizar la calidad del sistema.

- **Documentación**: Se reconoció la importancia de la documentación para la mantenibilidad del sistema. Se priorizó la documentación de las arquitecturas o configuraciones completadas para facilitar el trabajo futuro.

3.3.5 Duración del sprint

La duración de cada sprint fue definida con base en el esfuerzo requerido para completar las tareas asignadas en cada uno. Siento 1 punto de esfuerzo = 1 semana de desarrollo.

En la Tabla 24, se muestra cómo las historias de usuario fueron distribuidas en los sprints según su prioridad y complejidad, así también los objetivos de cada sprint y su respectivo punto de esfuerzo.

Tabla 23: *Planificación de cada Sprint.*

Fuente: Elaboración propia

Sprint		**Historias de usuario involucradas**				**Puntos de esfuerzo**
Ref.	**Objetivo**	**Ref.**	**Descripción**	**$PM_{historia}$**	**Esfuerzo en semanas**	
SP1	Definir una arquitectura escalable y eficiente para soportar el desarrollo futuro del proyecto.	H22	Diseño y desarrollo de la arquitectura	2,408374006	10,42825945	11 punto
SP2	Implementar el registro de datos generales para titulares y beneficiarios en el sistema.	H3	Registro de datos generales de titulares y beneficiarios	4,718447033	20,43087565	21 punto
SP3	Implementar la funcionalidad de registro para gestionar y almacenar fichas	H14	Registro de fichas emitidas	4,865898503	21,06934052	45 punto
		H13	Búsqueda por carnet y nombre del beneficiario	2,998179886	12,9821189	

	emitidas de manera eficiente. Implementar la búsqueda de beneficiarios por carnet y nombre, incluyendo la lectura de credenciales mediante pistola láser.	H12	Leer credenciales con pistola láser	0,589805879	2,553859457	
		H20	Gaceta de emisión de fichas para beneficiarios	3,784587724	16,38726485	
SP4	Implementar el manejo de beneficiarios y gestionar el registro de altas y bajas de asegurados.	H5	Inclusión de beneficiarios (esposas, hijos y padres)	4,718447033	20,43087565	37 punto
		H4	Registro de altas y bajas de los asegurados	3,686286745	15,9616216	
SP5	Implementar el escaneo y almacenamiento seguro de los documentos de los asegurados.	H6	Escaneo y almacenaje de documentos de los asegurados	5,111650952	22,13344862	23 punto
SP6	Configurar en el sistema los documentos correspondientes para cada tipo de beneficiario.	H11	Configurar el documento asignado a cada tipo de beneficiario en el sistema	1,671116658	7,235935127	8 punto
SP7	Modificar, almacenar y actualizar la documentación periódica, asegurando la inclusión de nuevos documentos requeridos.	H7	Modificación y almacenamiento de documentación periódica	2,113471067	9,15132972	32 punto
		H9	Actualización e inclusión de nuevos documentos requeridos	5,013349973	21,70780538	
SP8	Implementar una funcionalidad de registro de usuarios seguro y eficiente.	H16	Registro de usuarios	4,865898503	21,06934052	22 punto
SP9	Implementar una funcionalidad de control de acceso basado en niveles de jerarquía de usuario.	H10	Control de acceso según nivel de jerarquía de usuario	5,898058791	25,53859457	26 punto
SP10	Implementar una funcionalidad para registrar solicitudes y gestionar la entrega de carnets de manera eficiente.	H17	Registro de solicitudes y entrega de carnets	4,865898503	21,06934052	22 punto
SP11	Implementar la funcionalidad para registrar y gestionar los cambios en los horarios de atención de los médicos de manera eficiente.	H18	Registro de cambios en horarios de atención de los médicos	5,848908301	25,32577294	26 punto
SP12	Implementar la visualización de turnos en TV según	H19	Visualización de turnos según ficha en TV	4,570995563	19,79241079	20 punto

	la ficha correspondiente.					
SP13	Desarrollar una plataforma funcional que permita a los usuarios agendar visitas en línea de manera sencilla.	H21	Sitio web para agendar visitas por internet	8,060680348	34,90274591	35 punto
SP14	Implementar una función de detección y visualización de alertas para documentos caducados.	H8	Detector de alarmas de documentos caducados	2,998179886	12,9821189	32 punto
		H15	Visualización de alertas	4,128641154	17,8770162	
SP15	Implementar la generación de informes en el formato establecido y permitir la exportación de datos a Excel.	H2	Generación de informes según formato establecido por área	6,364988446	27,56039997	31 punto
		H1	Exportar datos a Excel	0,491504899	2,128216214	

3.4 DESARROLLO DEL PROYECTO

3.4.1 SP1: Diseño y desarrollo de la arquitectura

3.4.1.1 Sprint Planning

a. **Objetivo del Sprint:** Definir e implementar una arquitectura sólida y escalable para el sistema en entornos de desarrollo y producción, asegurando la correcta configuración de contenedores y orquestación mediante Docker y Kubernetes.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H22** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 24: Sprint Backlog SP1.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H22-T1	Diseñar la arquitectura del sistema para producción	Crear un esquema detallado que defina los componentes, servicios, bases de datos y redes para el entorno de producción.	Ninguno	Muy Alta
H22-T2	Diseñar la arquitectura del sistema para desarrollo	Establecer un entorno de desarrollo con configuraciones optimizadas para testing y desarrollo local, asegurando consistencia con el entorno de producción.	Ninguno	Muy Alta
H22-T3	Crear los manifiestos para Docker en modo desarrollo	Desarrollar y configurar los archivos Docker (Dockerfile y docker-compose.yml) que definan los contenedores y su interacción en el entorno de desarrollo.	H22-T2	Alta
H22-T4	Crear los manifiestos para Kubernetes en modo producción	Crear los archivos YAML que definirán la orquestación de los contenedores en Kubernetes, incluyendo	H22-T1	Alta

		Deployments, Services, ConfigMaps, etc.		
H22-T5	Definir un flujo CI/CD básico para producción	Implementar un flujo de CI/CD para desplegar automáticamente las aplicaciones en el entorno de producción sobre Kubernetes.	H22-T1 H22-T4	Media
H22-T6	Probar la arquitectura de desarrollo	Ejecutar pruebas funcionales y de carga en el entorno de desarrollo para asegurar que la arquitectura cumple con los requisitos.	H22-T2 H22-T3 H22-T5	Media
H22-T7	Probar la arquitectura de producción	Realizar pruebas de stress, escalabilidad y alta disponibilidad en el entorno de producción para verificar su robustez.	H22-T1 H22-T4 H22-T6	Media
H22-T8	Documentar la arquitectura	Crear una documentación detallada de la arquitectura tanto de desarrollo como de producción, incluyendo instrucciones de despliegue en GitHub.	Todos	Muy Alta

3.4.1.2 Sprint Review

Tabla 25: Tabla resumen del sprint review SP1.

Fuente: Elaboración propia.

TABLA RESUMEN DEL SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H22-T1	Diseñar la arquitectura del sistema para producción	Completada
H22-T2	Diseñar la arquitectura del sistema para desarrollo	Completada
H22-T3	Crear los manifiestos para Docker en modo desarrollo	Completada
H22-T4	Crear los manifiestos para Kubernetes en modo producción	Completada
H22-T5	Definir un flujo CI/CD básico para producción	Completada
H22-T6	Probar la arquitectura de desarrollo	Completada
H22-T7	Probar la arquitectura de producción	Completada
H22-T8	Documentar la arquitectura	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		

Resultados	
Ref.	**Descripción**
Diseño de la arquitectura del sistema	Se diseñaron arquitecturas tanto para entornos de desarrollo como de producción, considerando los requisitos del sistema y las mejores prácticas. La arquitectura de desarrollo fue diseñada para facilitar cambios rápidos y pruebas locales, mientras que la de producción priorizó la estabilidad y el escalado mediante el uso de Kubernetes
Implementación del API Gateway	Se configuraron 3 contenedores Docker en entorno de desarrollo y 3 Pods en entorno de producción para el correcto funcionamiento de Kong.
Flujo CI/CD básico	Se definió un flujo de integración y despliegue continuo (CI/CD) básico para el entorno de producción, automatizando la entrega de nuevas versiones de software y asegurando un despliegue eficiente.
Pruebas y validación	Se probaron los entornos de desarrollo y producción para asegurar que los componentes del sistema funcionen correctamente bajo las configuraciones propuestas. Los resultados de estas pruebas se documentaron para futuras referencias
Documentación	Se creó una documentación detallada que describe la arquitectura y los manifiestos de despliegue, disponible en el **Anexo J.**

3.4.1.3 Sprint Retrospective

Tabla 26: Tabla resumen del sprint retrospective SP1.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Cumplimiento de los entregables: Se completaron las tareas según lo planificado. - Buena gestión del tiempo: Las tareas se distribuyeron eficientemente durante el sprint. - Uso efectivo de herramientas: Notion facilitó el seguimiento del progreso y la organización de tareas.
Desafíos enfrentados
- Complejidad técnica: La integración de APIs con Kubernetes resultó más compleja de lo esperado. - Dificultades con la generación de PDFs: Retrasos debido a problemas técnicos en la creación de documentos PDF.
Mejoras para futuros sprints
- Mayor enfoque en la investigación técnica previa: Evaluar mejor las tareas antes de comenzar su desarrollo. - Optimización de pruebas: Aumentar el tiempo destinado a pruebas unitarias y funcionales.

Ilustración 17: *Diseño de la arquitectura para desarrollo.*

Fuente: Elaboración propia

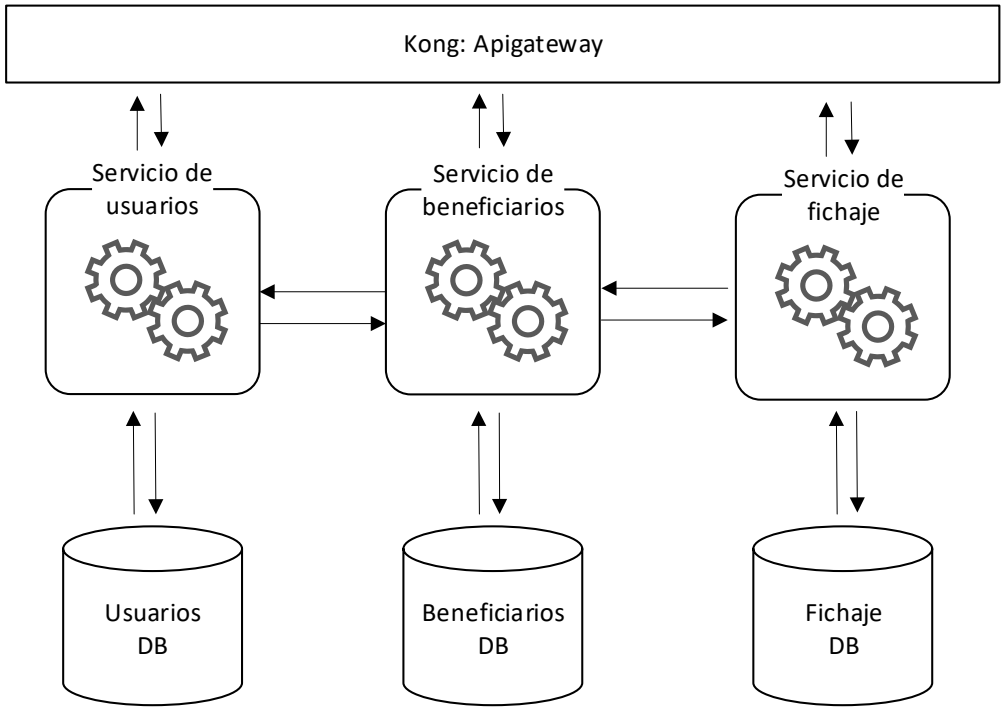


Ilustración 18: *Diseño de la arquitectura para producción.*

Fuente: Elaboración propia

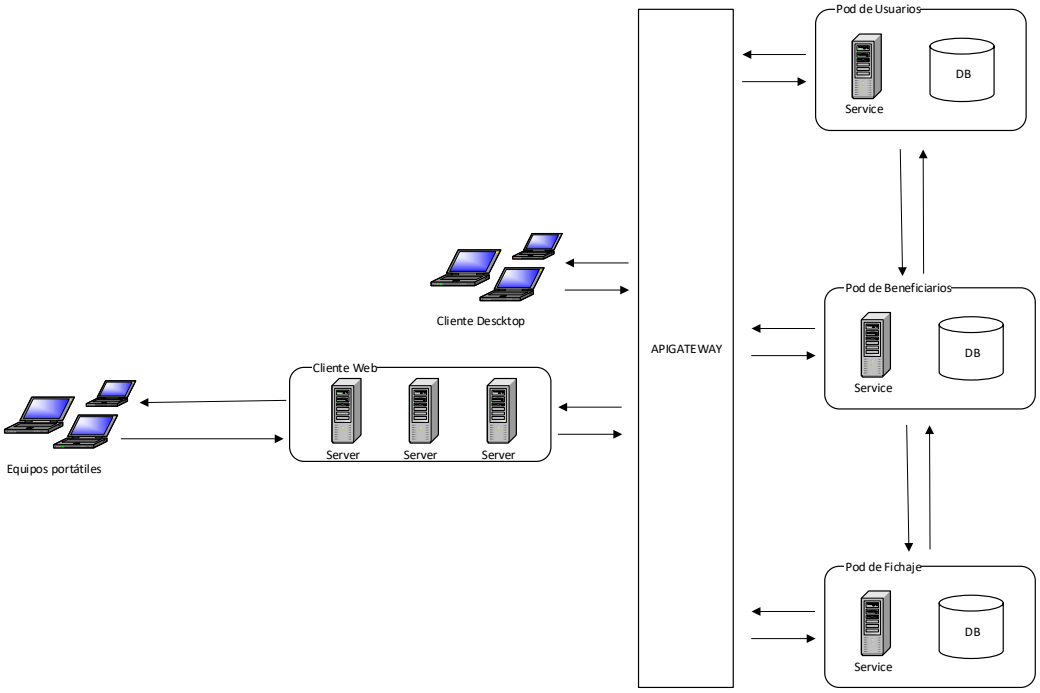


Ilustración 19: *Diseño del flujo CI / CD.*

Fuente: Elaboración propia

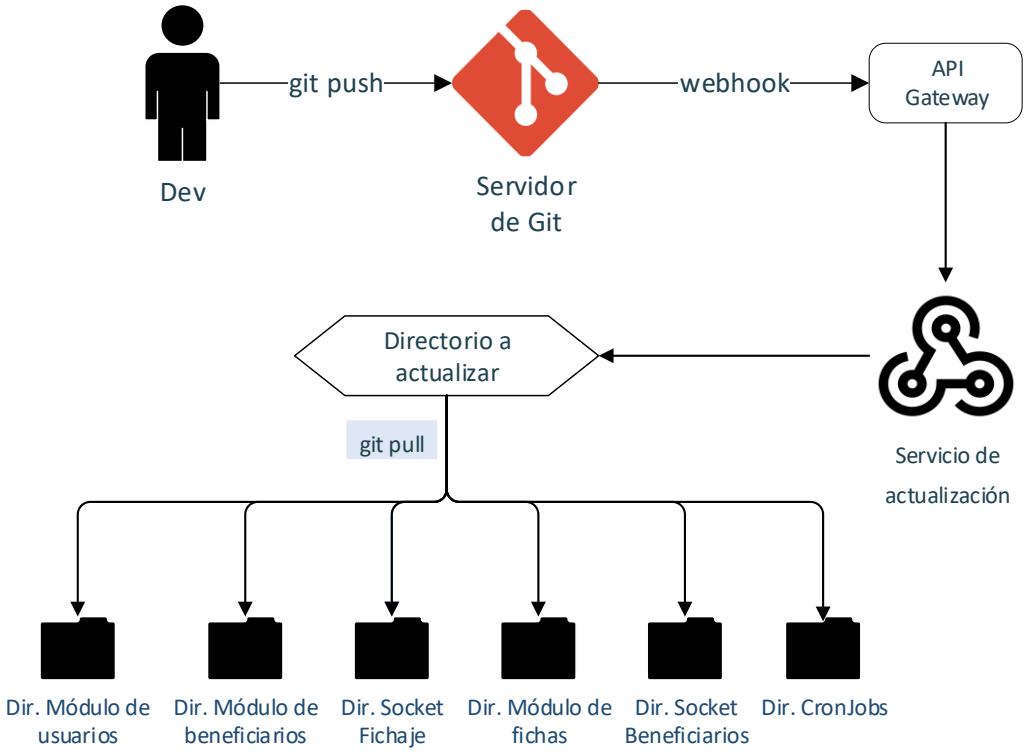


***Ilustración 20:**: Diagrama de procesos afiliación de funcionarios*

Fuente: Manual de procesos SMDC

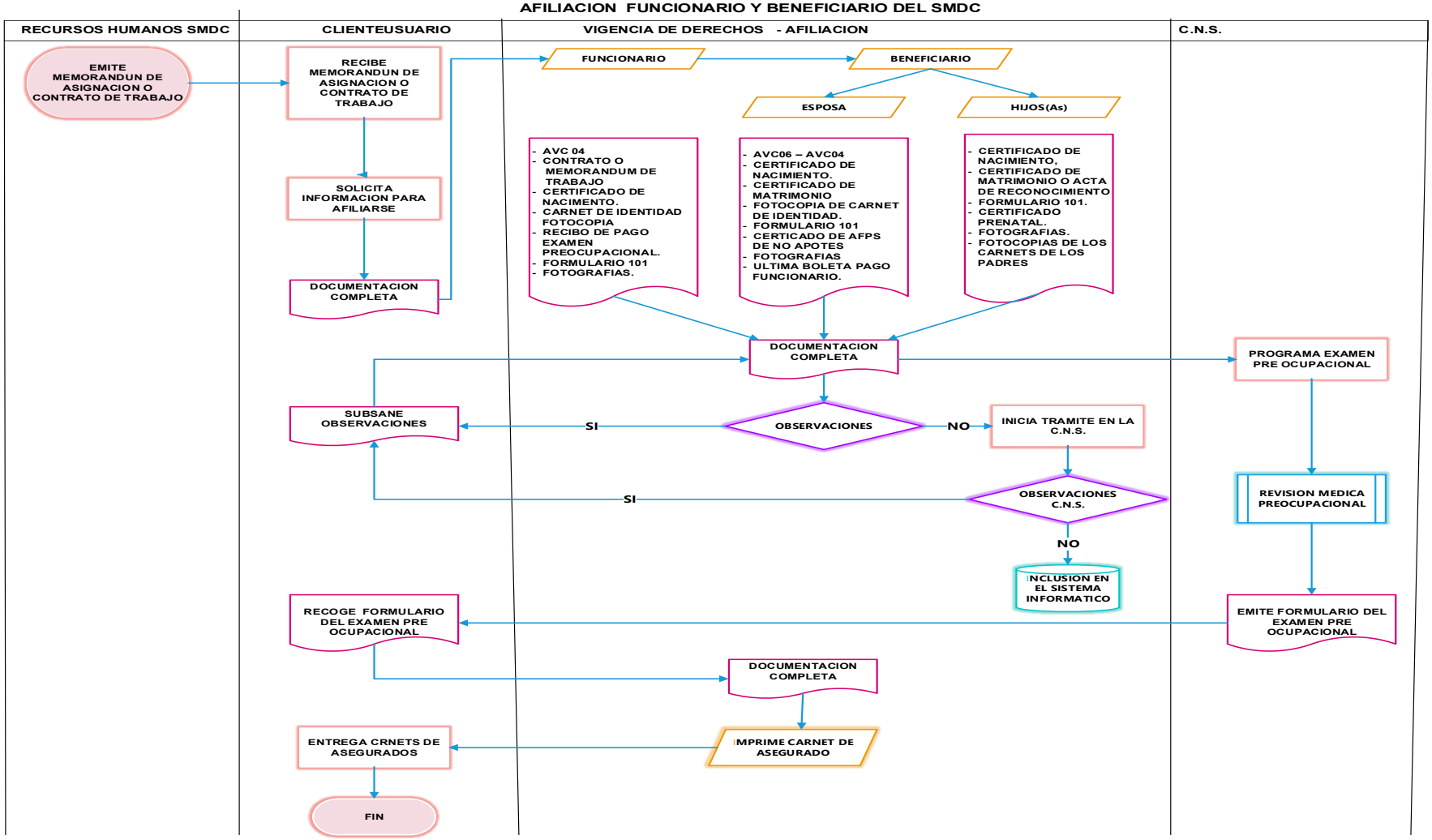
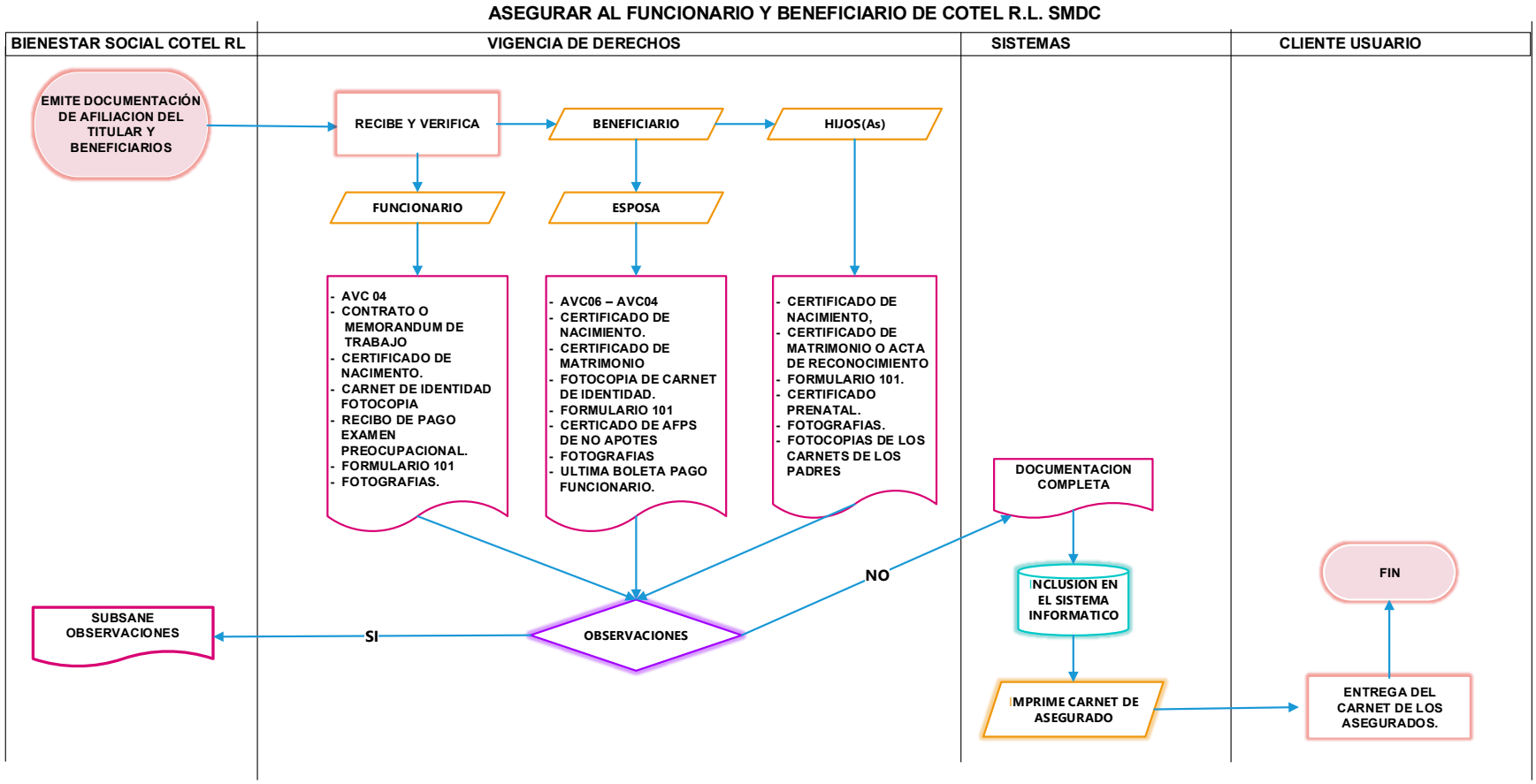


Ilustración 21: *Diagrama de procesos afiliación de funcionarios*

Fuente: Manual de procesos SMDC



3.4.2 SP2: Implementar el registro de datos generales para titulares y beneficiarios en el sistema.

3.4.2.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar el módulo de registro de datos generales para titulares y beneficiarios, permitiendo la gestión eficiente de la información en fichas médicas y documentos, utilizando Laravel para el backend, React para el frontend y asegurando el acceso al servicio mediante la integración con Kong.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H3** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 27: Sprint Backlog SP2.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H3-T1	Definir los requisitos de los datos generales	Identificar y documentar los campos y tipos de datos requeridos para los titulares y beneficiarios (nombre, apellido, fecha de nacimiento, etc.).	Ninguno	Muy Alta
H3-T2	Crear la base de datos para titulares y beneficiarios	Diseñar las tablas en la base de datos para almacenar los datos generales de titulares y beneficiarios, asegurando relaciones adecuadas entre las entidades.	H3-T1	Alta
H3-T3	Crear los modelos en Laravel	Implementar los modelos de Laravel para titulares y beneficiarios, estableciendo las relaciones y validaciones necesarias.	H3-T2	Media
H3-T4	Implementar las migraciones de la base de datos	Crear las migraciones de Laravel que reflejen las tablas y relaciones de la base de datos para asegurar consistencia entre el código y la base de datos.	H3-T2 H3-T3	Media

H3-T5	Implementar los controladores para el registro de datos	Desarrollar los controladores en Laravel que gestionen el flujo de creación, edición y eliminación de los datos generales de titulares y beneficiarios.	H3-T3	Media
H3-T6	Crear las rutas para las acciones CRUD	Definir las rutas en Laravel para que los controladores puedan gestionar las operaciones CRUD (crear, leer, actualizar y eliminar) de los titulares y beneficiarios.	H3-T5	Alta
H3-T7	Crear el formulario de registro de titulares y beneficiarios en React	Desarrollar el componente de formulario en React para que los usuarios puedan registrar los datos generales de los titulares y beneficiarios.	H3-T1	Alta
H3-T8	Validar la entrada de datos	Implementar validaciones en los formularios de registro para asegurar que los datos ingresados sean correctos (por ejemplo, fechas válidas, campos requeridos).	H3-T7	Baja
H3-T9	Registrar el servicio en Kong para acceso	Registrar el servicio en Kong para acceso	H3-T6	Muy Alta
H3-T10	Conectar el frontend con el backend (React y Laravel)	Configurar la comunicación entre React y Laravel para enviar y recibir datos desde el formulario al backend mediante peticiones HTTP.	H3-T9 H3-T8	Alta
H3-T11	Pruebas del módulo de registro	Realizar pruebas funcionales en el módulo para asegurar que los titulares y beneficiarios puedan registrarse, editarse y eliminarse correctamente.	H3-T10	Media
H3-T12	Documentar el módulo de registro	Crear la documentación del módulo que detalle su funcionamiento, estructura de la base de datos, controladores, rutas y vistas.	Todas	Baja

3.4.2.2 Sprint Review

Tabla 28: Tabla resumen del sprint review SP2.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H3-T1	Definir los requisitos de los datos generales	Completada
H3-T2	Crear la base de datos para titulares y beneficiarios	Completada
H3-T3	Crear los modelos en Laravel	Completada
H3-T4	Implementar las migraciones de la base de datos	Completada
H3-T5	Implementar los controladores para el registro de datos	Completada
H3-T6	Crear las rutas para las acciones CRUD	Completada
H3-T7	Crear el formulario de registro de titulares y beneficiarios en React	Completada
H3-T8	Validar la entrada de datos	Completada
H3-T9	Registrar el servicio en Kong para acceso	Completada
H3-T10	Conectar el frontend con el backend (React y Laravel)	Completada
H3-T11	Pruebas del módulo de registro	Completada
H3-T12	Documentar el módulo de registro	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición de requisitos	Se identificaron los campos y tipos de datos requeridos para titulares y beneficiarios según los informes entregados (ver Anexo J), los procesos y procedimientos (ver Anexo K) y reuniones con los empleados.	
Creación de base de datos	Se diseñaron las tablas necesarias para almacenar los datos generales de titulares y beneficiarios con las relaciones adecuadas entre las entidades.	
Implementación en Laravel	Se crearon los modelos y migraciones en Laravel, asegurando la correcta representación de los datos en la base de datos y estableciendo validaciones necesarias.	

Implementación de controladores y rutas	Se desarrollaron los controladores y rutas en Laravel para gestionar las operaciones CRUD para los titulares y beneficiarios.
Desarrollo del frontend	Se implementó el formulario en React para que los usuarios puedan registrar, editar y eliminar los datos de los titulares y beneficiarios, con validaciones en los formularios para asegurar que los datos ingresados sean correctos. Además de crear una lista para la visualización de los registros.
Integración con Kong	Se registró el servicio en Kong para permitir el acceso al backend desde el frontend.
Pruebas y validación	Se realizaron pruebas funcionales para asegurar que los procesos de registro, edición y eliminación de datos funcionen correctamente.
Documentación	Se documentó el funcionamiento del módulo de registro, incluyendo la estructura de la base de datos, las vistas y resultados de test.

3.4.2.3 Sprint Retrospective

Tabla 29: Tabla resumen del sprint retrospective SP2.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Definición clara de requisitos: Los requisitos de los datos generales se documentaron correctamente, lo que facilitó el diseño del resto de tareas. - Diseño eficaz de la base de datos: La estructura y relaciones entre titulares y beneficiarios fueron implementadas sin contratiempos. - Conexión exitosa entre frontend y backend: La comunicación entre React y Laravel para la gestión de datos se realizó de manera fluida mediante peticiones HTTP.
Desafíos enfrentados
- Integración con Kong: Hubo retrasos en la configuración del servicio en Kong para asegurar un acceso adecuado al sistema. - Validación de datos: Se enfrentaron problemas con las validaciones en el formulario, especialmente en la gestión de fechas y campos requeridos.
Mejoras para futuros sprints
- Mejor planificación en la integración de servicios externos (H3-T9): Realizar una planificación técnica más detallada al trabajar con servicios como Kong.

Ilustración 22: *Diseño de la base de datos de beneficiarios.*

Fuente: Elaboración propia

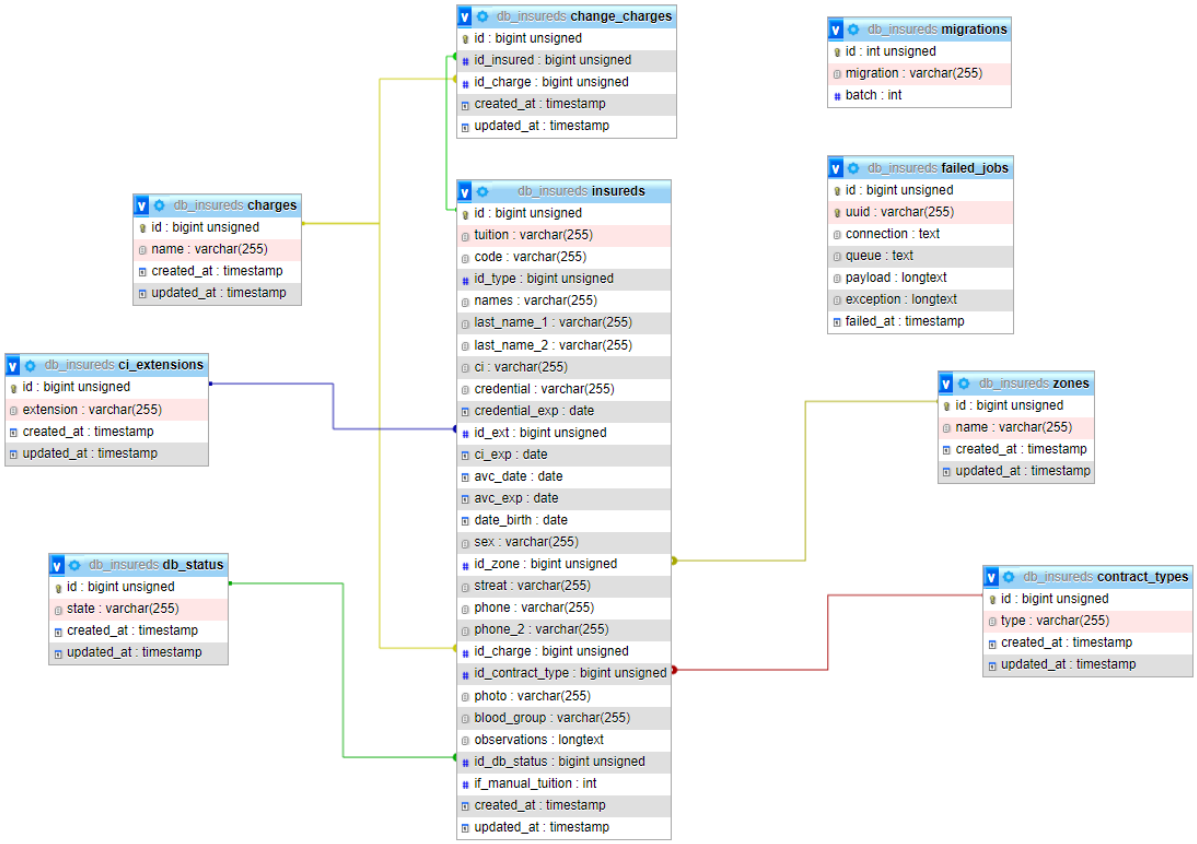
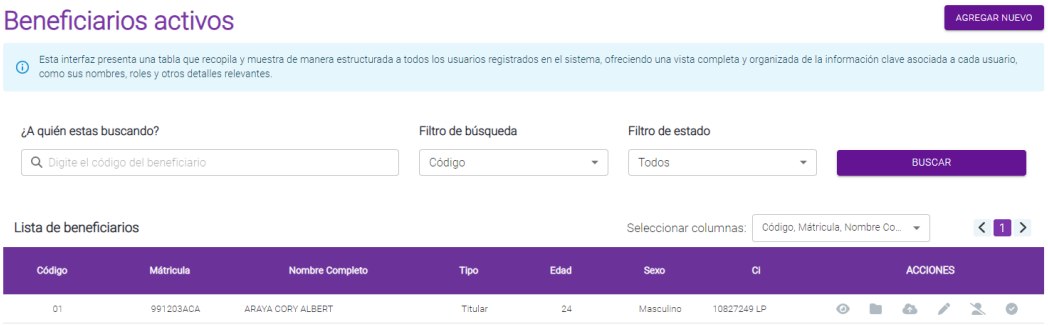


Ilustración 23: *Lista para la visualización de los titulares o beneficiarios.*

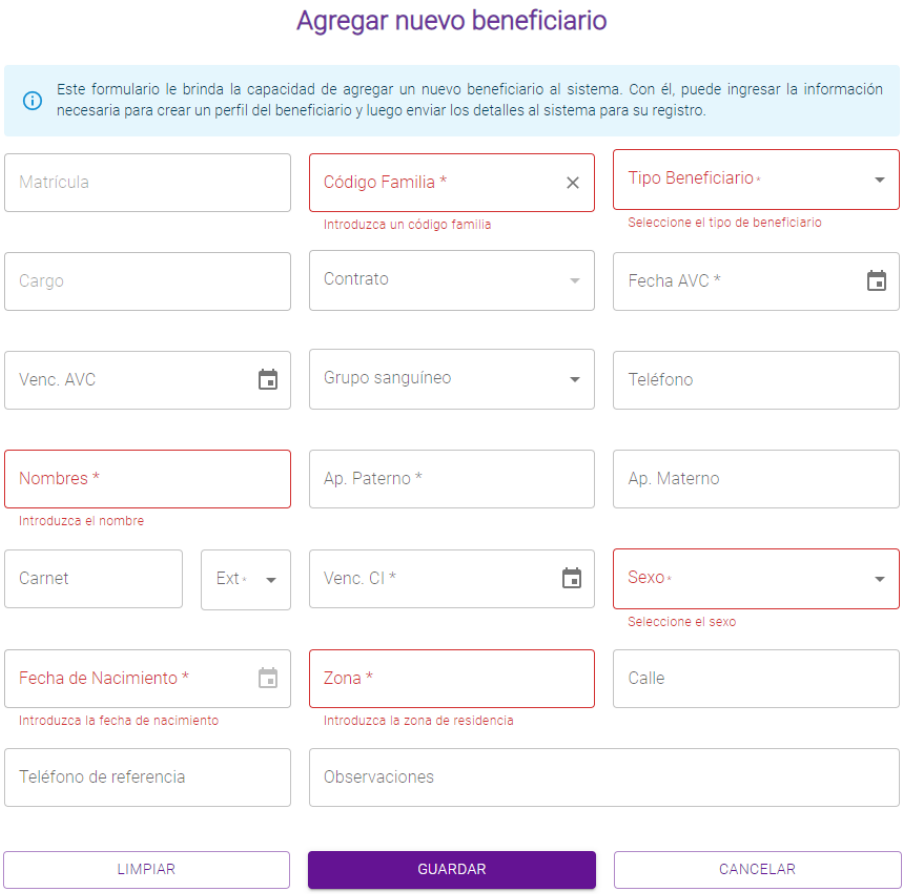
Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de los beneficiarios, mostrando una serie de acciones en la parte derecha que abren modales de configuración y registro de acciones.*

Ilustración 24: *Formulario para agregar a un titular o beneficiario.*

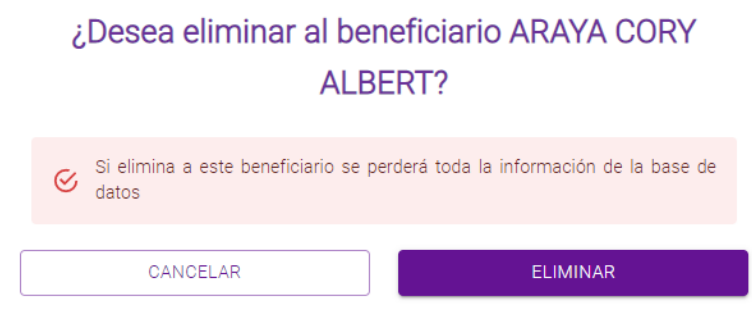
Fuente: Elaboración propia



Descripción: *Formulario empleado para el registro de nuevos beneficiarios con sus respectivas validaciones en el frontend para una mejor experiencia de usuario, se abre haciendo clic en la opción de agregar de la parte superior de la tabla de visualización de beneficiarios*

Ilustración 25: *Formulario para eliminar a un titular o beneficiario.*

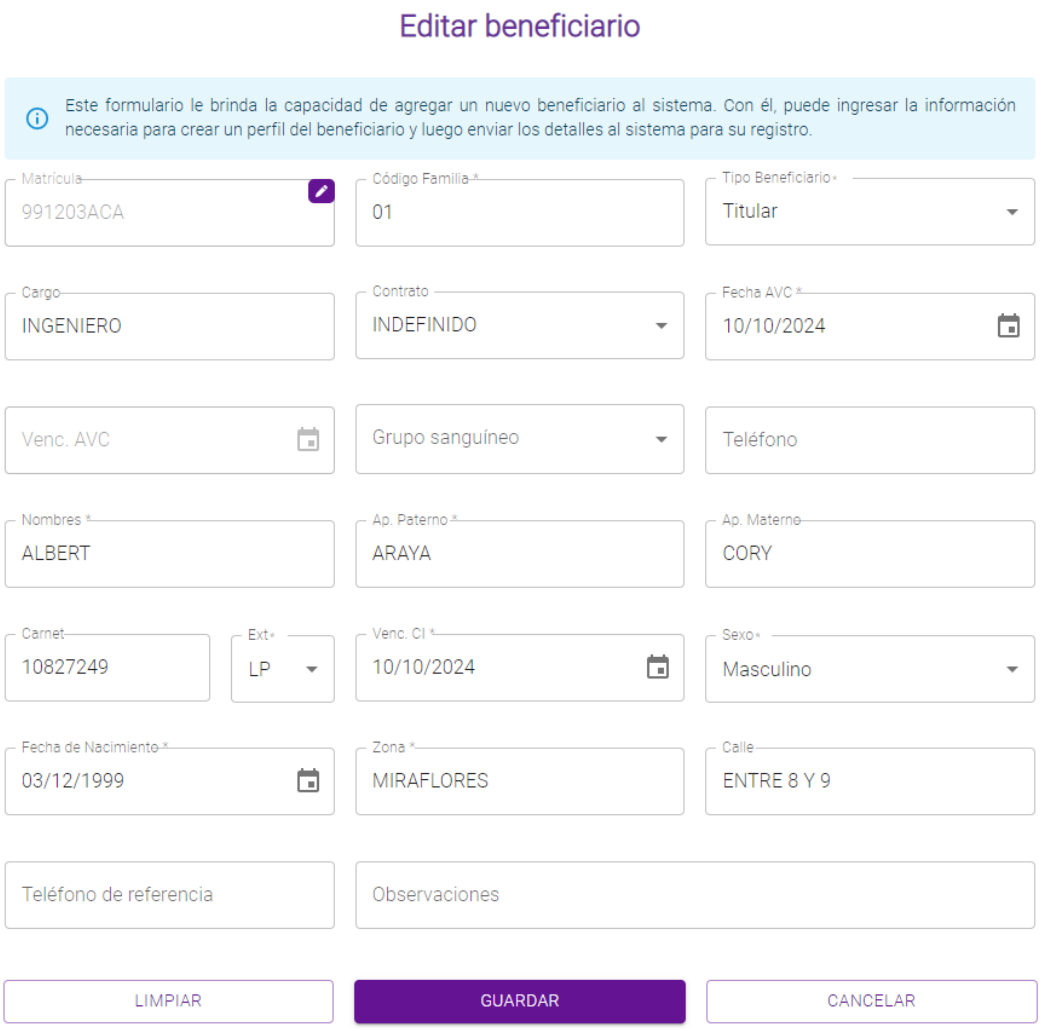
Fuente: Elaboración propia



Descripción: *Formulario empleado para la confirmación de eliminación de un beneficiario, se abre haciendo clic en la opción de eliminar de la tabla de visualización de beneficiarios.*

Ilustración 26: *Formulario para editar a un titular o beneficiario.*

Fuente: Elaboración propia



Descripción: *Formulario empleado para la edición de los beneficiarios con sus respectivas validaciones en el frontend para una mejor experiencia de usuario, se abre haciendo clic en la opción de editar de la tabla de visualización de beneficiarios.*

3.4.3 SP3: Implementar la funcionalidad de registro para gestionar y almacenar fichas emitidas de manera eficiente

3.4.3.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar la funcionalidad de registro que permita gestionar y almacenar fichas emitidas de manera eficiente, asegurando la correcta organización, acceso y manipulación de la información a través de Laravel para el backend y React para el frontend.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H14** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 30: Sprint Backlog SP3.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H14-T1	Definir los requisitos para las fichas emitidas	Identificar y documentar los campos y tipos de datos requeridos para las fichas	Ninguno	Muy Alta
H14-T2	Crear la base de datos para fichas emitidas	Diseñar las tablas en la base de datos que almacenarán las fichas emitidas, asegurando las relaciones necesarias con los beneficiarios y otros módulos.	H14-T1	Alta
H14-T3	Crear los modelos en Laravel	Implementar los modelos de Laravel para las fichas emitidas y doctores, estableciendo relaciones y validaciones necesarias con los modelos de titulares y beneficiarios.	H14-T2 H14-T13	Media
H14-T4	Implementar las migraciones de la base de datos	Crear las migraciones de Laravel que reflejen las tablas y relaciones de la base de datos para asegurar consistencia entre el código y la base de datos.	H14-T3	Media

H14-T5	Implementar los controladores para el registro de fichas	Desarrollar los controladores en Laravel que gestionen el flujo de creación, edición y eliminación de las fichas emitidas.	H14-T4	Media
H14-T6	Crear las rutas para las acciones CRUD de las fichas	Definir las rutas en Laravel que permitirán a los controladores gestionar las operaciones CRUD (crear, leer, actualizar y eliminar) de las fichas emitidas.	H14-T5	Alta
H14-T7	Crear el formulario de registro de fichas emitidas en React	Desarrollar el componente de formulario en React que permita a los usuarios registrar los datos de las fichas emitidas.	H14-T1	Alta
H14-T8	Validar la entrada de datos en el formulario	Implementar validaciones en el formulario de registro para asegurar que los datos ingresados sean correctos.	H14-T7	Baja
H14-T9	Registrar el servicio en Kong para acceso	Registrar el servicio en Kong para acceso	H3-T6	Muy Alta
H14-T10	Conectar el frontend con el backend (React y Laravel)	Configurar la comunicación entre React y Laravel para enviar y recibir datos desde el formulario de fichas emitidas mediante peticiones HTTP.	H14-T6 H14-T7 H14-T15 H14-T16	Alta
H14-T11	Pruebas de la funcionalidad de registro de fichas	Realizar pruebas funcionales en el módulo para asegurar que las fichas se registren, editen y eliminen correctamente.	H14-T9	Media
H14-T12	Documentar la funcionalidad de registro de fichas	Crear la documentación que detalle el funcionamiento del módulo de registro de fichas, incluyendo la estructura de la base de datos, controladores, rutas y vistas.	Todas	Baja
H14-T13	Crear la base de datos de doctores para registrar sus datos	Diseñar las tablas en la base de datos que almacenarán los datos de los doctores, asegurando las relaciones necesarias con los beneficiarios y las fichas.	H14-T1	Alta
H14-T14	Implementar los controladores para el registro de doctores	Desarrollar los controladores en Laravel que gestionen el flujo de creación, edición y eliminación de los doctores.	H14-T4	Media
H14-T15	Crear las rutas para las acciones CRUD de los doctores	Definir las rutas en Laravel que permitirán a los controladores gestionar las operaciones CRUD (crear, leer, actualizar y eliminar) de los doctores.	H14-T14	Alta

H14-T16	Crear el formulario de registro de doctores en React	Desarrollar el componente de formulario en React que permita a los usuarios registrar los datos de los doctores.	H14-T1	Alta
H13-T1	Crear un formulario para buscar a los beneficiarios por nombre, carnet, credencial o matricula.	Desarrollar el componente de formulario en React que permita a los usuarios buscar a los beneficiarios por nombre, carnet, credencial o matricula.	H14-T1	Media
H12-T1	Implementar la función para que se pueda leer credenciales con pistola láser	Implementar una función que permita leer el código de barra de las credenciales con pistola láser, definiendo que si se tarda mucho se suponga que el beneficiario se presentó sin carnet.	H13-T1	Baja

3.4.3.2 Sprint Review

Tabla 31: Tabla resumen del sprint review SP3.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H14-T1	Definir los requisitos para las fichas emitidas	Completada
H14-T2	Crear la base de datos para fichas emitidas	Completada
H14-T3	Crear los modelos en Laravel	Completada
H14-T4	Implementar las migraciones de la base de datos	Completada
H14-T5	Implementar los controladores para el registro de fichas	Completada
H14-T6	Crear las rutas para las acciones CRUD de las fichas	Completada
H14-T7	Crear el formulario de registro de fichas emitidas en React	Completada
H14-T8	Validar la entrada de datos en el formulario	Completada
H14-T9	Registrar el servicio en Kong para acceso	Completada
H14-T10	Conectar el frontend con el backend (React y Laravel)	Completada
H14-T11	Pruebas de la funcionalidad de registro de fichas	Completada
H14-T12	Documentar la funcionalidad de registro de fichas	Completada

H14-T13	Crear la base de datos de doctores para registrar sus datos	Completada
H14-T14	Implementar los controladores para el registro de doctores	Completada
H14-T15	Crear las rutas para las acciones CRUD de los doctores	Completada
H14-T16	Crear el formulario de registro de doctores en React	Completada
H14-T17	Crear la base de datos para doctores	(Nueva) Completada
H14-T18	Definir las especialidades asignadas a cada doctor	(Nueva) Completada
H14-T19	Registrar el servicio de doctores en Kong	(Nueva) Completada
H14-T20	Crear los formularios para el CRUD de los doctores	(Nueva) Completada
H13-T1	Crear un formulario para buscar a los beneficiarios por nombre, carnet, credencial o matricula.	(Nueva) Completada
H12-T1	Implementar la función para que se pueda leer credenciales con pistola láser	(Nueva) Completada

Observaciones: Por una falta de comunicación entre la gerencia y el área de sistemas, no se tomó en cuenta que los datos de los médicos serán cambiantes y es necesario crear un servicio puesto que a la larga este servirá para crear usuarios nuevos en el sistema, para la implementación de un módulo de atenciones médicas.
La historia de usuario H20 es producto de las historias H14 y H13, por lo cual solo se desarrolló la interfaz de usuario para dicha historia y se implementó el backend existente.

Problemas encontrados: En vista del desacuerdo se tuvo que incrementar la duración del sprint, puesto que se agregaron nuevas historias de usuario para completarse que serían críticas para el desarrollo de futuros módulos en el sistema.

Resultados

Definición de requisitos	Se identificaron y documentaron los campos y tipos de datos requeridos para las fichas emitidas según los informes entregados (ver Anexo J), los procesos y procedimientos (ver Anexo K) y reuniones con los empleados.
Creación de base de datos	Se diseñaron las tablas necesarias para almacenar las fichas emitidas, asegurando relaciones con los beneficiarios y otros módulos.
Implementación en Laravel	Se crearon los modelos y migraciones en Laravel, asegurando la correcta representación de los datos en la base de datos y estableciendo validaciones necesarias.
Implementación de controladores y rutas	Se desarrollaron los controladores y rutas en Laravel para gestionar las operaciones CRUD (crear, leer, actualizar y eliminar) de las fichas emitidas.
Desarrollo del frontend	Se implementó componentes en React para que los usuarios puedan registrar los datos de las fichas emitidas, con validaciones en los formularios para asegurar que los datos ingresados sean correctos.
Integración con Kong	Se registró el servicio en Kong para permitir el acceso al backend desde el frontend.
Pruebas y validación	Se realizaron pruebas funcionales para asegurar que los procesos de registro, edición

	y eliminación de fichas funcionen correctamente. Los resultados se documentaron para referencia futura.
Registro de doctores	Se desarrolló el módulo, la base de datos y las rutas necesarias para registrar a los doctores y sus especialidades, además de formularios en React para su gestión, tarea que no estaba contemplada originalmente y que añadió complejidad al sprint.
Documentación	Se documentó el funcionamiento del módulo de registro de fichas y doctores, incluyendo la estructura de la base de datos, los controladores, rutas y vistas. Esta documentación está disponible para consulta en el Anexo M.

3.4.3.3 Sprint Retrospective

Tabla 32: Tabla resumen del sprint retrospective SP3.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Comunicación efectiva con los empleados, lo que permitió resolver rápidamente problemas técnicos. - Integración fluida entre React (frontend) y Laravel (backend). - Implementación exitosa del módulo de fichas emitidas y el registro de doctores, a pesar de los cambios imprevistos. - Uso eficiente de las herramientas de gestión de APIs (Kong) para asegurar el acceso entre servicios.
Desafíos enfrentados
- Requisitos adicionales de último minuto para implementar el módulo de doctores, lo que aumentó el alcance del trabajo y la duración del sprint. - La necesidad de coordinar de manera rápida y efectiva para incorporar el nuevo módulo sin afectar otros objetivos del sprint. - Falta de una planificación adecuada para manejar posibles cambios en los requisitos, lo que causó cierta desorganización en la fase final del sprint.
Mejoras para futuros sprints
- Implementar una revisión de requisitos más rigurosa antes de comenzar el sprint para minimizar cambios de último minuto. - Incluir una capacidad de adaptación a cambios en el plan de sprint para abordar de manera más eficiente los nuevos requisitos. - Planificar con más anticipación posibles extensiones o ampliaciones de módulos importantes. - Definir mejor las próximas historias para no tener duplicaciones y falsas estimaciones de esfuerzos.

Ilustración 27: *Diseño de la base de datos de fichaje.*

Fuente: Elaboración propia

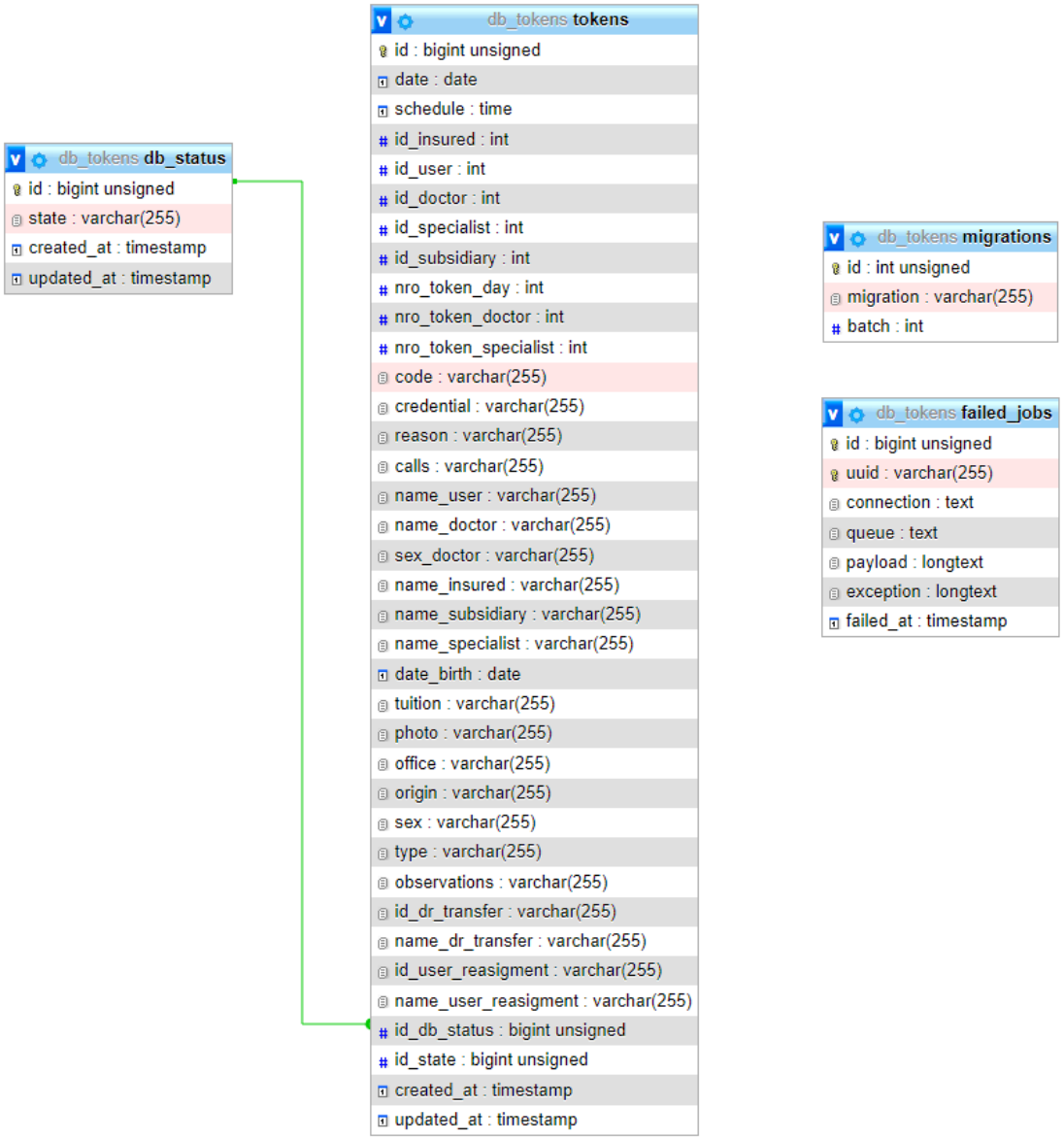


Ilustración 28: *Diseño de la base de datos de doctores.*

Fuente: Elaboración propia

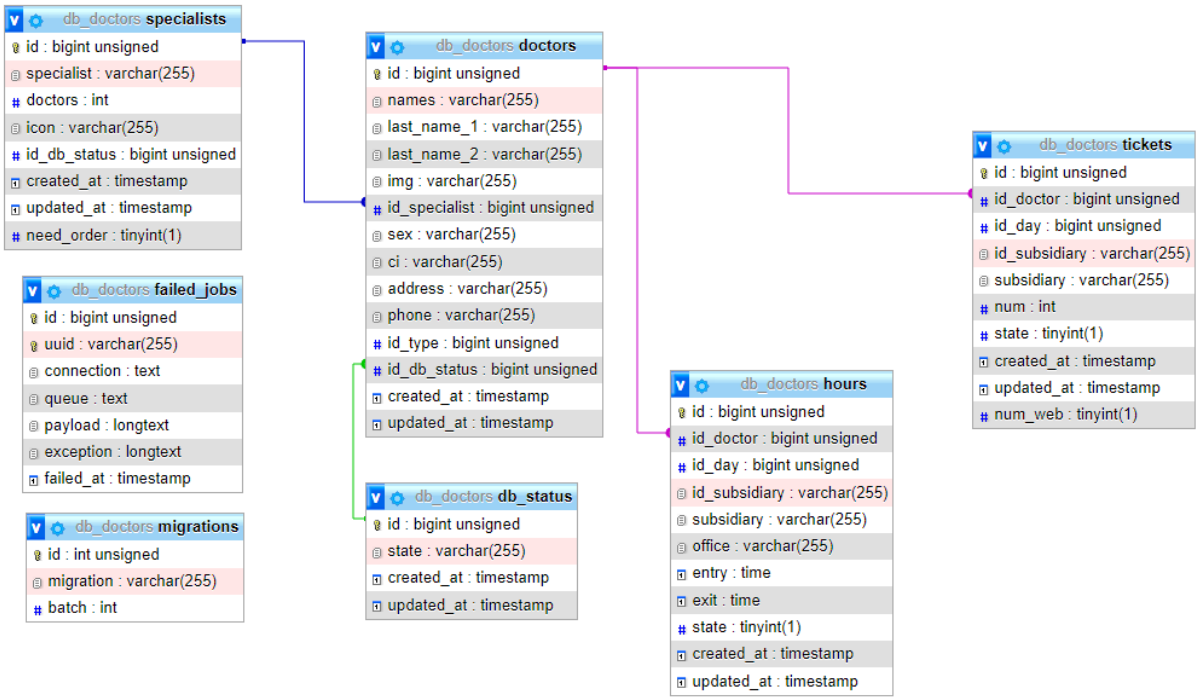


Ilustración 29: *Formulario para agregar un nuevo doctor.*

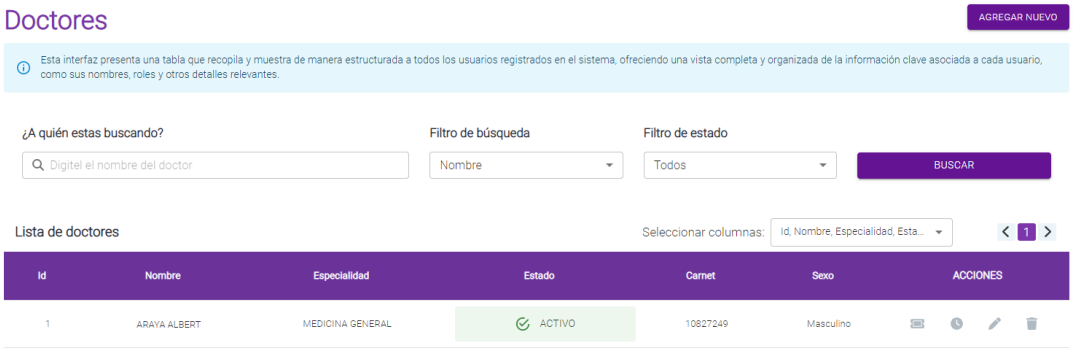
Fuente: Elaboración propia



Descripción: *Formulario empleado para el registro de nuevos doctores con sus respectivas validaciones en el frontend para una mejor experiencia de usuario, se abre haciendo clic en la opción de agregar de la parte superior de la tabla de visualización de doctores*

Ilustración 30: *Tabla para visualizar a los doctores registrados.*

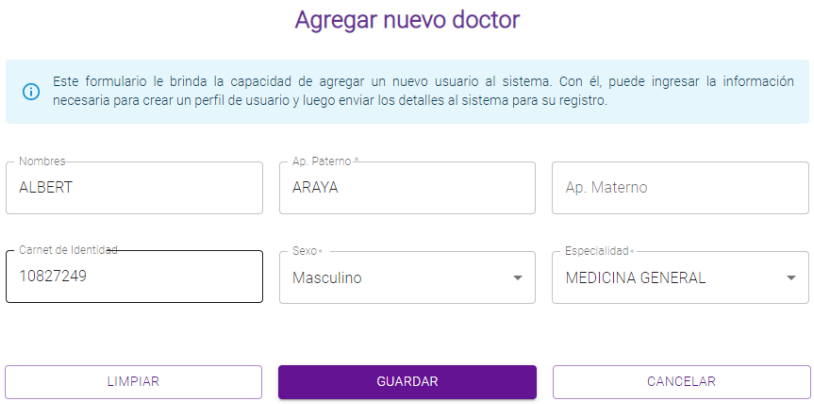
Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de los doctores, mostrando una serie de acciones en la parte derecha que abren modales de configuración y registro de acciones.*

Ilustración 31: *Modal para agregar un nuevo doctor externo.*

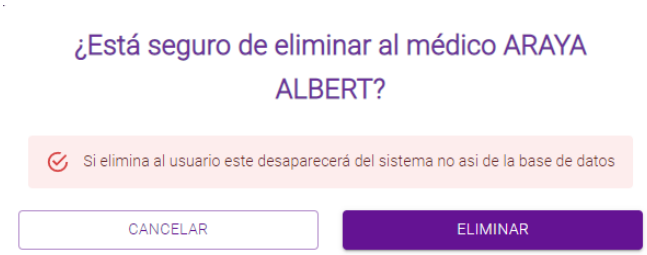
Fuente: Elaboración propia



Descripción: *Modal empleado para agregar doctores externos para la consulta de horarios y atenciones en vigencia de derechos parcial*

Ilustración 32: *Formulario de eliminación de doctores.*

Fuente: Elaboración propia



Descripción: *modal empleado para eliminar a los doctores*

Ilustración 33: *Ventana de búsqueda por carnets.*

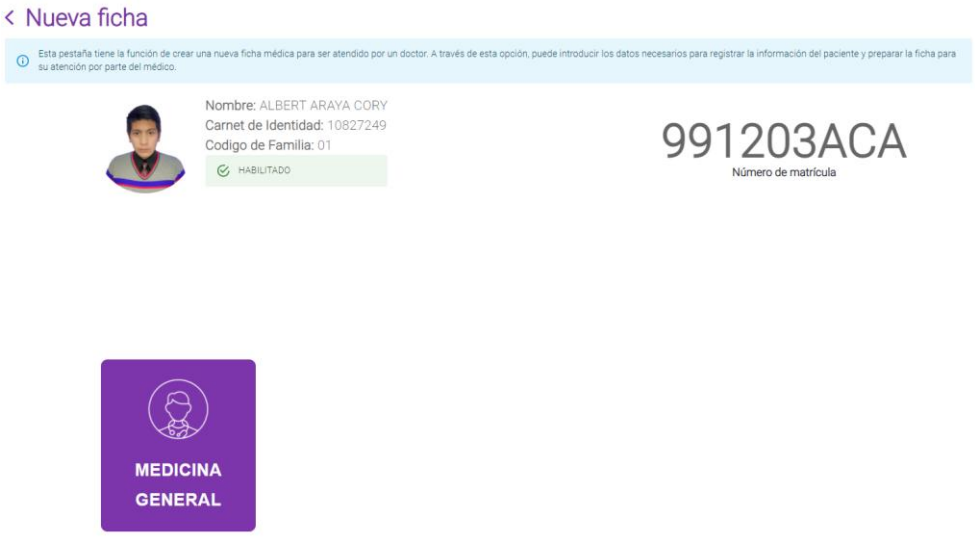
Fuente: Elaboración propia



Descripción: *Vista empleada para la búsqueda de los beneficiarios, para el registro de una ficha para la atención médica.*

Ilustración 34: *Ventana de asignación de especialidad.*

Fuente: Elaboración propia



Descripción: *Vista empleada para la elegir una especialidad al momento de registrar una ficha.*

Ilustración 35: *Ventana de asignación de doctores.*

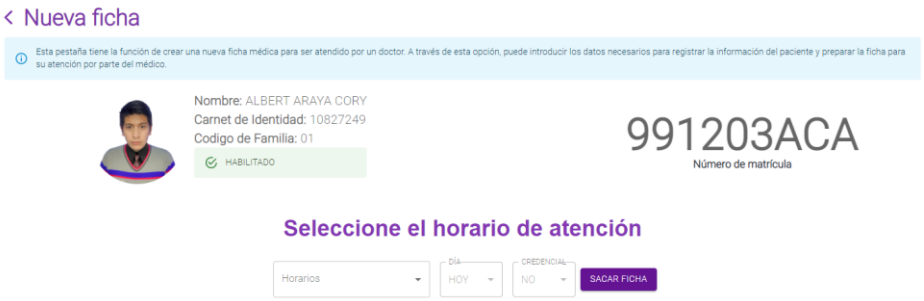
Fuente: Elaboración propia



Descripción: *Vista empleada para la elegir un doctor según la especialidad para registrar una ficha.*

Ilustración 36: *Ventana de asignación de horarios.*

Fuente: Elaboración propia



Descripción: *Vista empleada para la elegir un horario según el doctor y la especialidad para registrar una ficha.*

3.4.4 SP4: Implementar el manejo de beneficiarios y gestionar el registro de altas y bajas de asegurados.

3.4.4.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar la funcionalidad para gestionar el cambio de tipo de beneficiarios y el registro de altas y bajas de asegurados, garantizando la inclusión y correcta administración de los datos de asegurados y sus beneficiarios (esposas, hijos y padres) mediante Laravel para el backend y React para el frontend.

b. Sprint backlog

En base a que el sprint involucra a las historias de usuario **H5 y H4** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 33: Sprint Backlog SP4.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H4-T1	Definir los requisitos de datos para altas y bajas de asegurados	Identificar los campos y datos necesarios para registrar altas y bajas (fecha, motivo, asegurado, etc.).	Ninguno	Muy Alta
H5-T1	Definir los requisitos de datos para la clasificación de tipos de beneficiarios	Identificar los campos necesarios para la inclusión de beneficiarios (nombre, relación, fecha de nacimiento, asegurado relacionado, etc.).	Ninguno	Muy Alta
H4-T2	Diseñar las tablas de la base de datos para altas y bajas de asegurados	Diseñar las tablas en la base de datos que almacenarán el registro de altas y bajas de asegurados.	H4-T1	Alta
H5-T2	Diseñar las tablas de la base de datos para la clasificación de beneficiarios	Diseñar las tablas en la base de datos para almacenar la información de beneficiarios, asegurando relaciones adecuadas con los asegurados.	H5-T1	Alta
H4-T3	Crear los modelos en Laravel para altas y bajas de asegurados	Implementar los modelos de Laravel para las altas y bajas de asegurados, estableciendo relaciones y validaciones necesarias.	H4-T2	Alta

H5-T3	Crear los modelos en Laravel para cambio de tipo de beneficiarios	Implementar los modelos de Laravel para los beneficiarios, asegurando las relaciones con los asegurados.	H5-T2	Alta
H4-T4 H5-T4	Implementar las migraciones de la base de datos	Crear las migraciones de Laravel para las tablas de altas y bajas de asegurados y de beneficiarios para reflejar la estructura en la base de datos.	H4-T3 H5-T3	Alta
H4-T5	Implementar los controladores para gestionar altas y bajas de asegurados	Desarrollar los controladores en Laravel que gestionen el flujo de creación, edición y eliminación de las altas y bajas de asegurados.	H4-T3	Alta
H5-T5	Implementar los controladores para la inclusión de beneficiarios	Desarrollar los controladores en Laravel que gestionen el flujo de creación, edición y eliminación de beneficiarios.	H5-T3	Alta
H4-T6	Crear las rutas para las acciones CRUD (altas y bajas)	Definir las rutas en Laravel que gestionarán las operaciones CRUD para las altas y bajas de asegurados.	H4-T5	Alta
H5-T6	Crear las rutas para las acciones CRUD (beneficiarios)	Definir las rutas en Laravel para gestionar las operaciones CRUD de los beneficiarios (crear, leer, actualizar y eliminar).	H5-T5	Alta
H4-T7	Crear el formulario para altas y bajas en React	Desarrollar el componente de formulario en React que permita registrar altas y bajas de asegurados.	H4-T1	Media
H5-T7	Crear el formulario para la inclusión de beneficiarios en React	Desarrollar el componente de formulario en React para que los asegurados puedan registrar a sus beneficiarios.	H5-T1	Media
H4-T8	Validar la entrada de datos en los formularios (altas, bajas y beneficiarios)	Implementar validaciones para asegurar que los datos ingresados en los formularios sean correctos.	H4-T7	Media
H5-T8	Validar la entrada de datos en los formularios (cambio de tipo)	Implementar validaciones para asegurar que los datos ingresados en los formularios sean correctos.	H5-T7	Media
H4-T9	Conectar el frontend con el backend para altas y bajas	Configurar la comunicación entre React y Laravel para que los datos de altas y bajas se envíen al backend correctamente mediante peticiones HTTP.	H4-T12	Alta
H5-T9	Conectar el frontend con el backend para beneficiarios	Configurar la comunicación entre React y Laravel para que los datos de beneficiarios se envíen al backend correctamente.	H5-T12	Alta

H4-T10	Pruebas del módulo de altas y bajas	Realizar pruebas funcionales del módulo para asegurar que las altas y bajas se gestionen correctamente.	H4-T9	Media
H5-T10	Pruebas del módulo de beneficiarios	Realizar pruebas funcionales del módulo de beneficiarios para asegurarse de que los beneficiarios puedan ser registrados, editados y eliminados correctamente.	H5-T9	Media
H4-T11	Documentar el módulo de altas y bajas y beneficiarios	Crear la documentación que describa el funcionamiento del módulo, la estructura en la base de datos, controladores, rutas y formularios.	Todas las H4	Baja
H5-T11	Documentar el módulo de altas y bajas y beneficiarios	Crear la documentación que describa el funcionamiento del cambio de tipo de beneficiario, la estructura en labase de datos, controladores, rutas y formularios.	Todas las H5	Baja
H5-T12 H6-T12	Registrar el servicio en Kong para acceso	Registrar el servicio en Kong para acceso	H4-T6 H5-T6	Muy Alta

3.4.4.2 Sprint Review

Tabla 34: Tabla resumen del sprint review SP4.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H4-T1	Definir los requisitos de datos para altas y bajas de asegurados	Completada
H5-T1	Definir los requisitos de datos para la clasificación de beneficiarios	Completada
H4-T2	Diseñar las tablas de la base de datos para altas y bajas de asegurados	Completada
H5-T2	Diseñar las tablas de la base de datos para la clasificación de beneficiarios	Completada
H4-T3	Crear los modelos en Laravel para altas y bajas de asegurados	Completada
H5-T3	Crear los modelos en Laravel para cambio de tipo de beneficiarios	Completada
H4-T4	Implementar las migraciones de la base de datos	Completada
H5-T4	Implementar las migraciones de la base de datos	Completada
H4-T5	Implementar los controladores para gestionar altas y bajas de asegurados	Completada

H5-T5	Implementar los controladores para la inclusión de beneficiarios	Completada
H4-T6	Crear las rutas para las acciones CRUD (altas y bajas)	Completada
H5-T6	Crear las rutas para las acciones CRUD (beneficiarios)	Completada
H4-T7	Crear el formulario para altas y bajas en React	Completada
H5-T7	Crear el formulario para la inclusión de beneficiarios en React	Completada
H4-T8	Validar la entrada de datos en los formularios (altas, bajas y beneficiarios)	Completada
H5-T8	Validar la entrada de datos en los formularios (cambio de tipo)	Completada
H4-T9	Conectar el frontend con el backend para altas y bajas	Completada
H5-T9	Conectar el frontend con el backend para beneficiarios	Completada
H4-T10	Pruebas del módulo de altas y bajas	Completada
H5-T10	Pruebas del módulo de beneficiarios	Completada
H4-T11	Documentar el módulo de altas y bajas y beneficiarios	Completada
H5-T11	Documentar el módulo de altas y bajas y beneficiarios	Completada
H5-T12	Registrar el servicio en Kong para acceso	Completada
H6-T12	Registrar el servicio en Kong para acceso	Completada
H5-T12	Formulario de bajas con fehca de baja y fecha de registro de baja para manejo de la cesantia	(Nueva) Completada
H5-T12	Desarrollo e implementación de la lógica de backend para el almacenamiento y revisión de los archivos de bajas registrados en pdf.	(Nueva) Completada

Observaciones: Ninguna

Problemas encontrados: Existieron incongruencias en la definición de los requisitos lo que hiso que se incremente la duración del sprint. Los requerimientos entregados inicialmente no contemplaban la incorporación de la fecha de la baja y del documento, documento que tenía que ser almacenado en el sistema para futuras revisiones

Resultados

Definición de requisitos	Se identificaron y documentaron los campos y tipos de datos necesarios para registrar las altas y bajas de asegurados, así como los beneficiarios según los informes entregados (ver Anexo J), los procesos y procedimientos (ver Anexo K) y reuniones con los empleados.
Edición de base de datos	Se diseñaron las tablas para almacenar las altas y bajas de asegurados, y la información de beneficiarios, asegurando las relaciones adecuadas entre ambas.
Implementación en Laravel	Se crearon los modelos y migraciones en Laravel para las altas y bajas de asegurados, y para los beneficiarios, incluyendo validaciones necesarias.

Implementación de controladores y rutas	Se desarrollaron los controladores y rutas en Laravel para gestionar las operaciones CRUD de altas y bajas de asegurados, y de beneficiarios.
Desarrollo del frontend	Se implementaron los componentes en React para los formularios de altas y bajas de asegurados, así como para los beneficiarios.
Validación de formularios	Se añadieron validaciones en los formularios para asegurar la correcta entrada de datos, tanto para las altas y bajas de asegurados como para los beneficiarios.
Integración del frontend y backend	Se configuró la comunicación entre React y Laravel para enviar y recibir datos correctamente en las altas y bajas de asegurados, así como para los beneficiarios.
Pruebas y validación	Se realizaron pruebas funcionales que validaron el correcto registro, edición y eliminación de asegurados y beneficiarios, documentándose los resultados.

3.4.4.3 Sprint Retrospective

Tabla 35: Tabla resumen del sprint retrospective SP4.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Gestión de dependencias: Se manejaron adecuadamente las dependencias entre tareas, lo que permitió un flujo de trabajo continuo. - Implementación técnica: Las migraciones y modelos en Laravel se implementaron sin mayores contratiempos, asegurando la integridad de la base de datos. - Validaciones en formularios: Las validaciones implementadas en React garantizan la calidad de los datos ingresados por los usuarios.
Desafíos enfrentados
- Tiempo de documentación: La complejidad de los módulos de altas, bajas y beneficiarios hizo que la documentación requiriera más tiempo del previsto. - Implementación de bajas: En vista de nuevos requerimientos se procedió a desarrolla tanto la lógica en el backend como la interfaz de usuario lo que incremento el tiempo del sprint. - Integración frontend-backend: Aunque se completó, la conexión entre React y Laravel presentó desafíos menores en la sincronización de datos.
Mejoras para futuros sprints
- Revisión de backlog: Implementar una revisión más rigurosa del backlog para evitar duplicaciones y asegurar que todas las tareas estén correctamente referenciadas. - Estimación de tiempos: Ajustar las estimaciones de tiempo para la documentación, considerando la complejidad de las tareas. - Capacitación en herramientas: Proporcionar capacitación adicional sobre herramientas de gestión de proyectos para mejorar la organización y seguimiento de tareas.

Ilustración 37: *Diseño de la base de datos de beneficiarios con las tablas de clasificación.*

Fuente: Elaboración propia

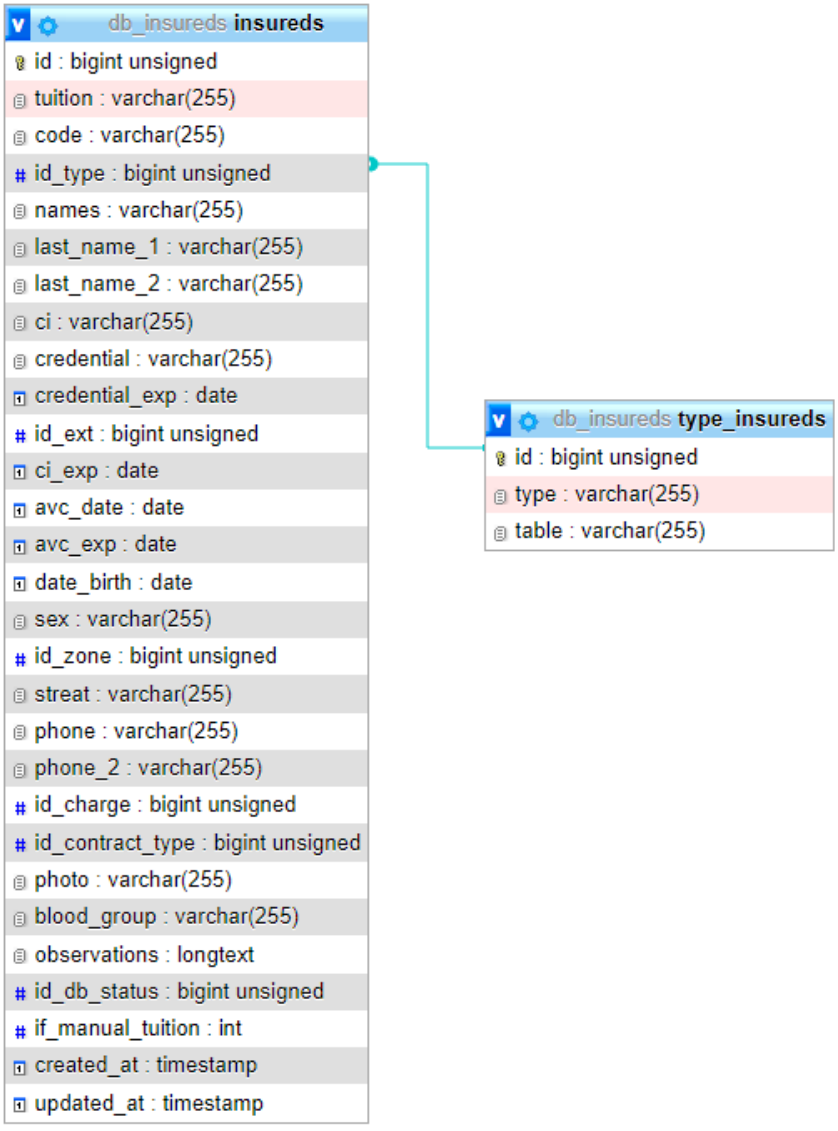


Ilustración 38: *Diseño de la base de datos de beneficiarios con las tablas de altas y bajas.*

Fuente: Elaboración propia

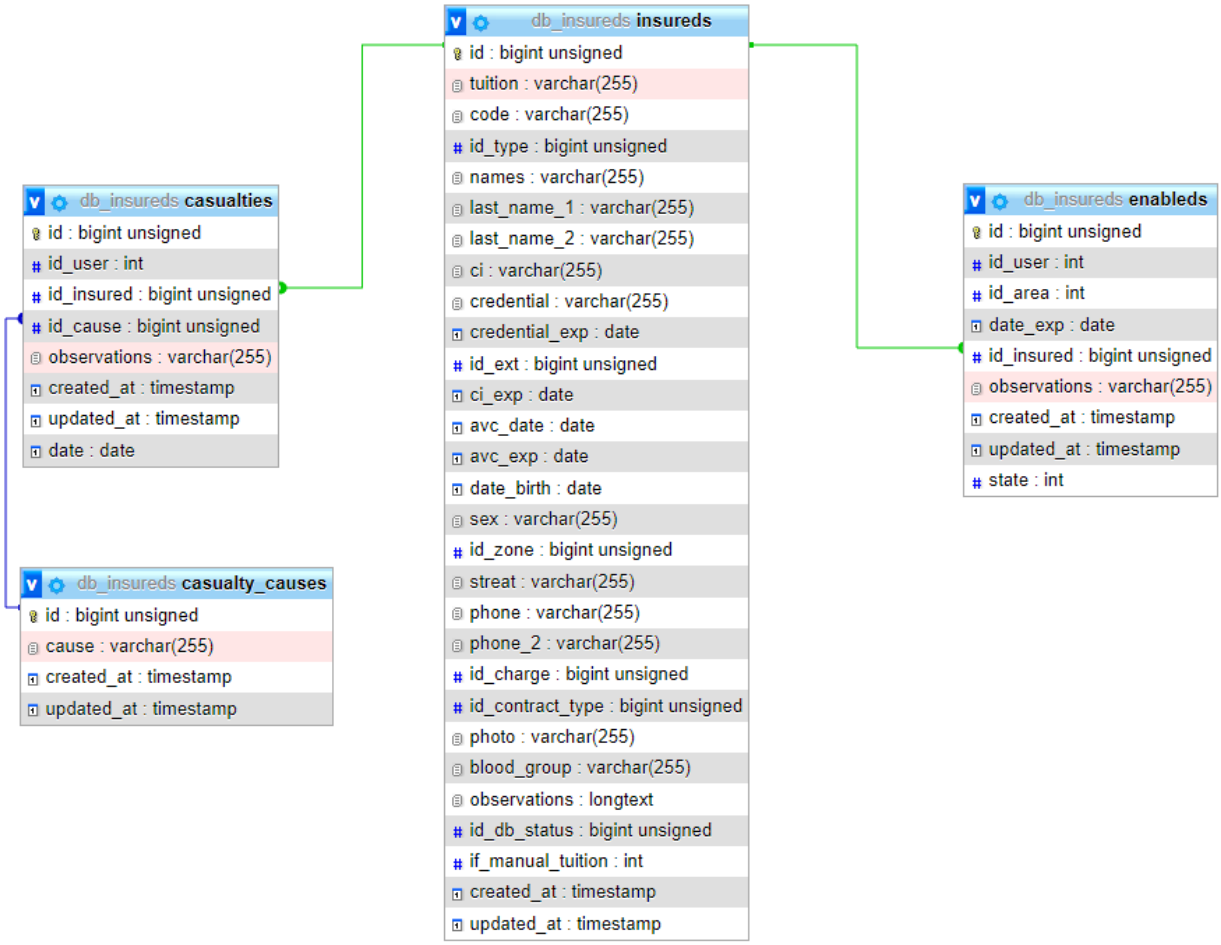
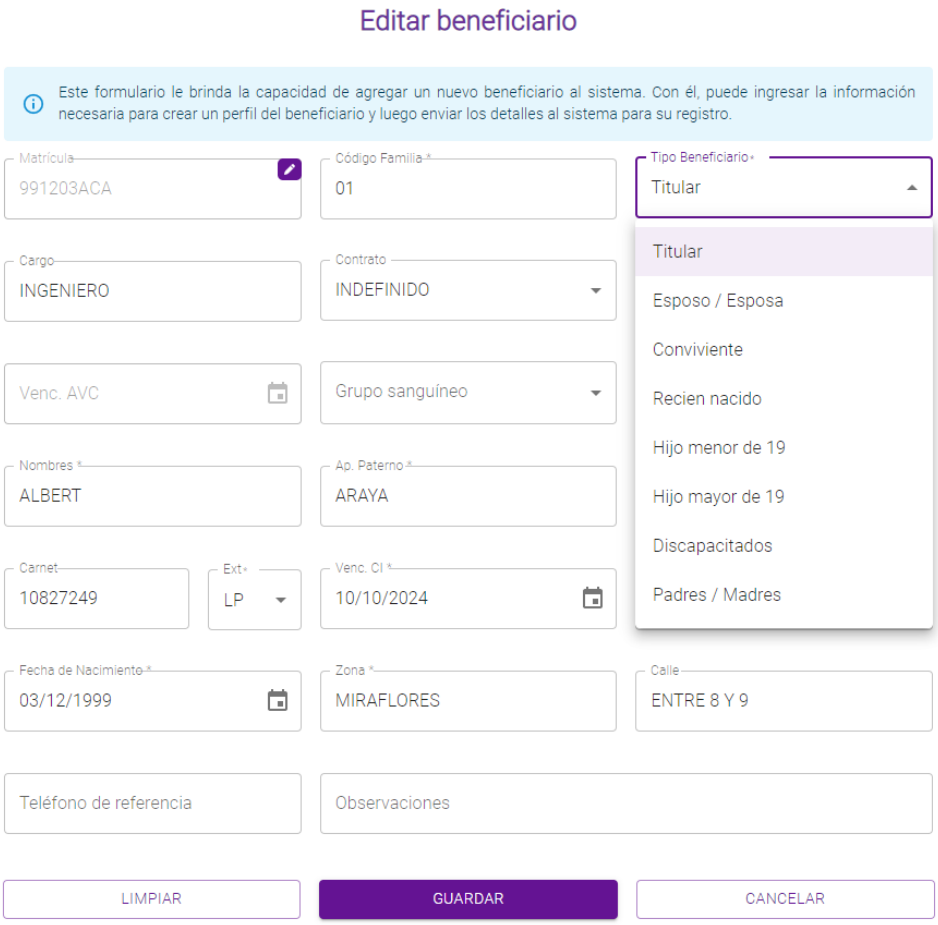


Ilustración 39: *Formulario de edición con la opción de cambio de tipo de beneficiario.*

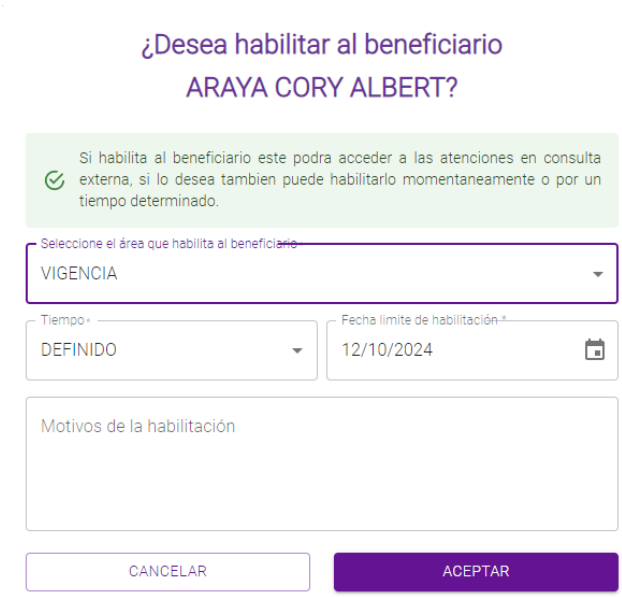
Fuente: Elaboración propia



Descripción: *Se agregaron los tipos de beneficiarios a los formularios para agregar y editar, para el registro de documentos, altas y bajas de beneficiarios*

Ilustración 40: *Formulario de habilitación de beneficiarios.*

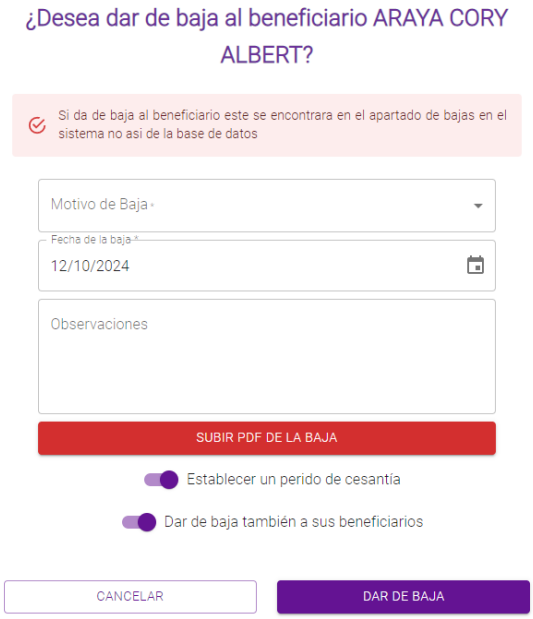
Fuente: Elaboración propia



Descripción: *Formulario empleado para el registro de habilitaciones de beneficiarios con sus respectivas validaciones en el frontend para una mejor experiencia de usuario, se abre haciendo clic en la opción de habilitar de la fila en la que se encuentre el beneficiario en la tabla de visualización de beneficiarios*

Ilustración 41: *Formulario de baja de beneficiarios.*

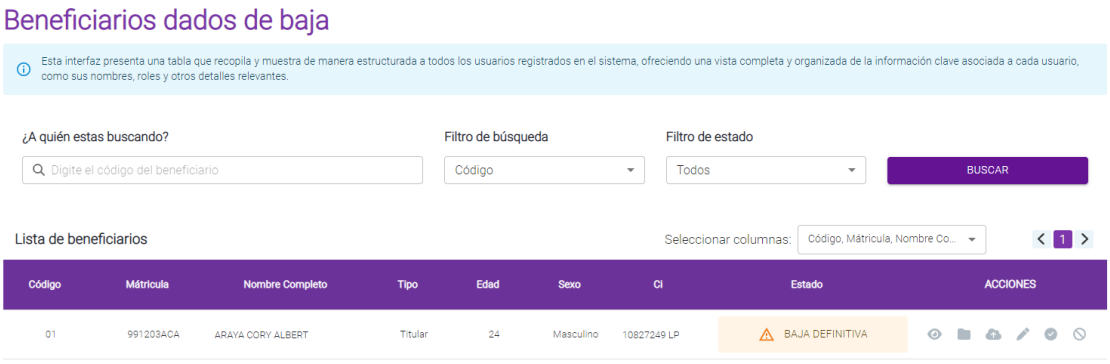
Fuente: Elaboración propia



Descripción: *Formulario empleado para el registro de bajas de beneficiarios con sus respectivas validaciones en el frontend para una mejor experiencia de usuario, se abre haciendo clic en la opción de dar de baja de la fila en la que se encuentre el beneficiario en la tabla de visualización de beneficiarios*

Ilustración 42: *Tabla de visualización de los beneficiarios dados de baja.*

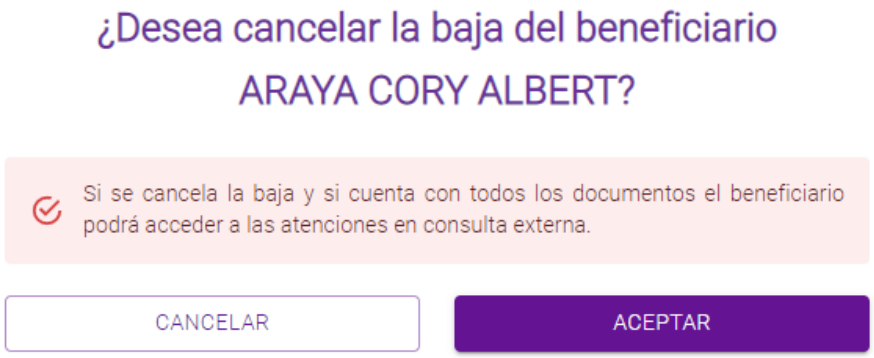
Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de los beneficiarios dados de baja, mostrando una serie de acciones en la parte derecha que abren modales de configuración y registro de acciones.*

Ilustración 43: *Modal de cancelación de bajas o habilitaciones.*

Fuente: Elaboración propia



Descripción: *Formulario empleado para la cancelación de bajas o habilitaciones de beneficiarios con sus respectivas validaciones en el frontend para una mejor experiencia de usuario, se abre haciendo clic en la opción de cancelar de la fila en la que se encuentre el beneficiario en la tabla de visualización de beneficiarios*

3.4.5 SP5: Implementar el escaneo y almacenamiento seguro de los documentos de los asegurados

3.4.5.1 Sprint Planning

a. **Objetivo del Sprint:** Diseñar e implementar un proceso automatizado para el escaneo y almacenamiento seguro de los documentos de los asegurados, utilizando tecnologías que aseguren la confidencialidad, integridad y disponibilidad de los documentos almacenados.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H6** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 36: Sprint Backlog SP5.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H6-T1	Definir los tipos de documentos a escanear y almacenar	Identificar los tipos de documentos necesarios y las reglas de almacenamiento.	Ninguno	Muy Alta
H6-T2	Diseñar las tablas de la base de datos para almacenamiento de documentos	Crear las tablas en la base de datos para almacenar la información relacionada con los documentos escaneados	H6-T1	Alta
H6-T3	Crear los modelos en Laravel para gestionar los documentos	Implementar los modelos de Laravel que gestionen la relación entre los asegurados y sus documentos, estableciendo las validaciones y relaciones necesarias.	H6-T2	Alta
H6-T4	Implementar las migraciones de la base de datos para documentos	Crear las migraciones de Laravel que reflejen la estructura de las tablas para almacenar los documentos de manera segura.	H6-T3	Alta
H6-T5	Implementar el backend para la subida segura de documentos escaneados	Desarrollar los controladores en Laravel para gestionar el proceso de subida de documentos escaneados al servidor de manera segura,	H6-T4	Alta

		incluyendo cifrado y validación.		
H6-T6	Definir las rutas para las operaciones de subida y consulta de documentos	Definir las rutas en Laravel que permitirán a los usuarios subir, consultar y descargar los documentos escaneados.	H6-T5	Alta
H6-T7	Crear el componente de React para la subida de documentos escaneados	Desarrollar el formulario de frontend en React que permita a los usuarios escanear (subir) documentos desde un escáner o su dispositivo y subirlos al sistema.	H6-T1	Media
H6-T8	Implementar la validación de los documentos en el frontend	Asegurar que los documentos subidos cumplan con los formatos y tamaños permitidos, validando estos aspectos desde el frontend antes de la subida.	H6-T7	Media
H6-T9	Conectar el frontend con el backend para la subida de documentos	Configurar la comunicación entre React y Laravel para gestionar la subida de los documentos escaneados mediante peticiones HTTP seguras.	H6-T6 H6-T7	Alta
H6-T10	Implementar el almacenamiento seguro de documentos en el servidor	Configurar un almacenamiento seguro en el servidor para almacenar los documentos escaneados de manera que estén protegidos contra accesos no autorizados.	H6-T5	Alta
H6-T11	Implementar cifrado para los documentos	Configurar el cifrado de los documentos tanto en tránsito como en reposo para asegurar que solo usuarios autorizados puedan acceder a ellos.	H6-T5 H6-T10	Alta
H6-T12	Implementar medidas de acceso seguro a los documentos	Configurar roles y permisos en Laravel para asegurar que solo los usuarios autorizados puedan visualizar y descargar los documentos almacenados.	H6-T11	Alta
H6-T13	Pruebas de subida y almacenamiento de documentos	Realizar pruebas funcionales para asegurar que los documentos se suban, almacenen y se recuperen de manera segura y eficiente.	H6-T9 H6-T10	Media
H6-T14	Pruebas de seguridad en el acceso a documentos	Verificar que las medidas de seguridad funcionen correctamente y que solo los usuarios autorizados puedan acceder a los documentos.	H6-T12	Media

H6-T15	Documentar el módulo de escaneo y almacenamiento de documentos	Crear la documentación técnica que describa la estructura de la base de datos, las rutas, controladores y los mecanismos de seguridad implementados para el manejo de documentos.	Todas	Baja

3.4.5.2 Sprint Review

Tabla 37: Tabla resumen del sprint review SP5.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H6-T1	Definir los tipos de documentos a escanear y almacenar	Completada
H6-T2	Diseñar las tablas de la base de datos para almacenamiento de documentos	Completada
H6-T3	Crear los modelos en Laravel para gestionar los documentos	Completada
H6-T4	Implementar las migraciones de la base de datos para documentos	Completada
H6-T5	Implementar el backend para la subida segura de documentos escaneados	Completada
H6-T6	Definir las rutas para las operaciones de subida y consulta de documentos	Completada
H6-T7	Crear el componente de React para la subida de documentos escaneados	Completada
H6-T8	Implementar la validación de los documentos en el frontend	Completada
H6-T9	Conectar el frontend con el backend para la subida de documentos	Completada
H6-T10	Implementar el almacenamiento seguro de documentos en el servidor	Completada
H6-T11	Implementar cifrado para los documentos	Completada
H6-T12	Implementar medidas de acceso seguro a los documentos	Completada
H6-T13	Pruebas de subida y almacenamiento de documentos	Completada
H6-T14	Pruebas de seguridad en el acceso a documentos	Completada
H6-T15	Documentar el módulo de escaneo y almacenamiento de documentos	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		

Definición de tipos de documentos	Se identificaron los tipos de documentos necesarios y se establecieron las reglas de almacenamiento para garantizar un manejo adecuado de la información.
Diseño de la base de datos	Se diseñaron las tablas necesarias para almacenar la información relacionada con los documentos escaneados, asegurando la integridad de los datos.
Implementación en Laravel	Se crearon los modelos y migraciones en Laravel para gestionar la relación entre los asegurados y sus documentos, estableciendo validaciones y relaciones necesarias.
Desarrollo del backend para la subida de documentos	Se desarrollaron los controladores en Laravel que gestionan el proceso de subida de documentos escaneados al servidor de manera segura, incluyendo cifrado y validación.
Definición de rutas para la gestión de documentos	Se definieron las rutas en Laravel que permiten a los usuarios subir, consultar y descargar los documentos escaneados, facilitando la interacción con el sistema.
Desarrollo del frontend	Se implementó un formulario en React que permite a los usuarios escanear y subir documentos desde un escáner o su dispositivo, asegurando la facilidad de uso del sistema.
Validación de documentos en el frontend	Se implementaron validaciones en el frontend para asegurar que los documentos subidos cumplan con los formatos y tamaños permitidos antes de la subida.
Integración del frontend y backend	Se configuró la comunicación entre React y Laravel para gestionar la subida de documentos escaneados mediante peticiones HTTP seguras, asegurando la correcta funcionalidad.
Almacenamiento seguro de documentos	Se configuró un almacenamiento seguro en el servidor para proteger los documentos escaneados contra accesos no autorizados, garantizando la seguridad de la información.
Implementación de cifrado	Se configuró el cifrado de los documentos tanto en tránsito como en reposo, asegurando que solo usuarios autorizados puedan acceder a ellos.
Medidas de acceso seguro	Se configuraron roles y permisos en Laravel para asegurar que solo los usuarios autorizados puedan visualizar y descargar los documentos almacenados.
Pruebas de funcionalidad de subida	Se realizaron pruebas funcionales para asegurar que los documentos se suban, almacenen y se recuperen de manera segura y eficiente, garantizando la calidad del proceso.
Pruebas de seguridad	Se verificó que las medidas de seguridad funcionen correctamente y que solo los usuarios autorizados puedan acceder a los documentos, asegurando la protección de la información.
Documentación del módulo	Se creó la documentación técnica que describe la estructura de la base de datos, las

	rutas, controladores y mecanismos de seguridad implementados para el manejo de documentos.

3.4.5.3 Sprint Retrospective

Tabla 38: Tabla resumen del sprint retrospective SP5.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Definición clara de requisitos: La identificación de los tipos de documentos a escanear y almacenar se llevó a cabo de manera efectiva, lo que permitió establecer reglas claras para el almacenamiento. - Diseño robusto de la base de datos: Las tablas diseñadas para el almacenamiento de documentos se implementaron sin problemas, asegurando que la estructura fuera adecuada para el manejo de los datos. - Validación eficaz en el frontend: La implementación de la validación de documentos en el frontend ayudó a reducir errores durante la subida de archivos, mejorando la experiencia del usuario.
Desafíos enfrentados
- Problemas de integración: Se presentaron dificultades en la comunicación entre el frontend y el backend, lo que llevó a retrasos en la implementación de la subida de documentos. - Requisitos de seguridad complejos: La configuración de medidas de acceso seguro y cifrado resultó más complicada de lo esperado, lo que aumentó el tiempo dedicado a estas tareas. - Pruebas de seguridad extensivas: Las pruebas de seguridad requerían más tiempo del planificado, ya que era necesario verificar minuciosamente que las restricciones de acceso funcionaran correctamente.
Mejoras para futuros sprints
- Revisión de integración temprana: Realizar revisiones más frecuentes de la integración entre el frontend y el backend durante el sprint para identificar y resolver problemas de comunicación rápidamente. - Capacitación en seguridad: Proporcionar formación adicional al equipo sobre las mejores prácticas de seguridad para manejar documentos, lo que podría acelerar la implementación de medidas de seguridad.

Ilustración 44: *Diseño de la base de datos de beneficiarios con las tablas de almacenamiento de documentos.*

Fuente: Elaboración propia

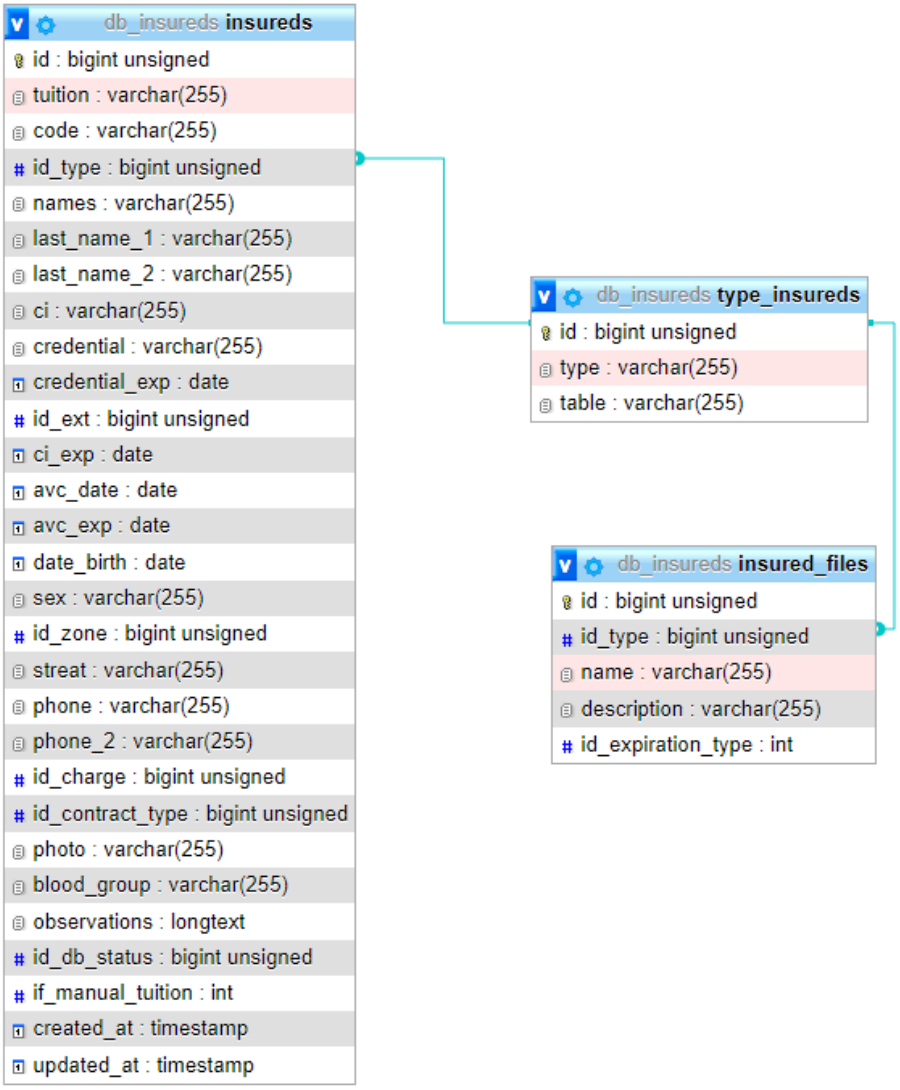


Ilustración 45: *Modal de subida de archivos.*

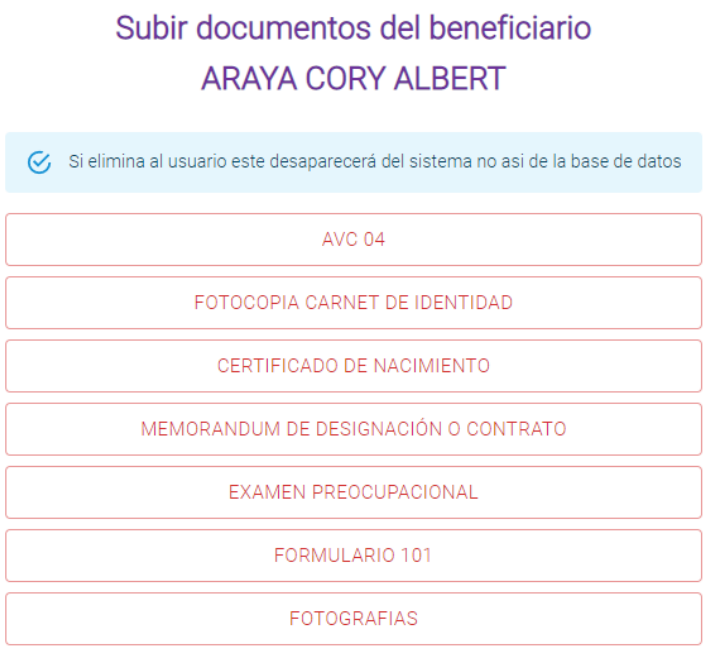
Fuente: Elaboración propia



Descripción: *Formulario empleado para el registro de documentos, este presenta una opción para colocar una fecha de vencimiento opcional para generar alertas si el documento se vence, se abre haciendo*

Ilustración 46: *Modal de ejemplo de titulares para elegir el documento a subir según el tipo de beneficiarios.*

Fuente: Elaboración propia

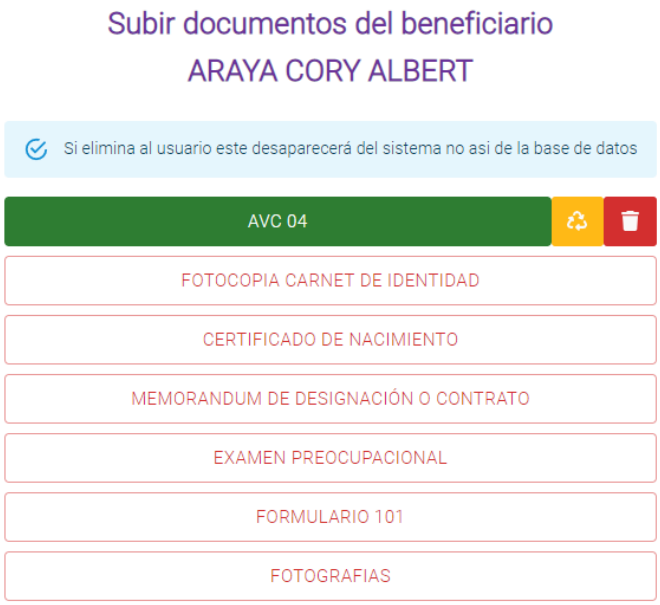


Descripción: *Formulario empleado para el registro de documentos de los beneficiarios con sus respectivas validaciones en el frontend para una mejor experiencia de usuario, se abre después de aceptar el modal de subida de archivos, ese formulario es dinámico ya que los documentos cambian según el tipo.*

clic en la opción de agregar documentos en la fila en la que se encuentra el beneficiario

Ilustración 47: *Modal de ejemplo con el archivo en el servidor.*

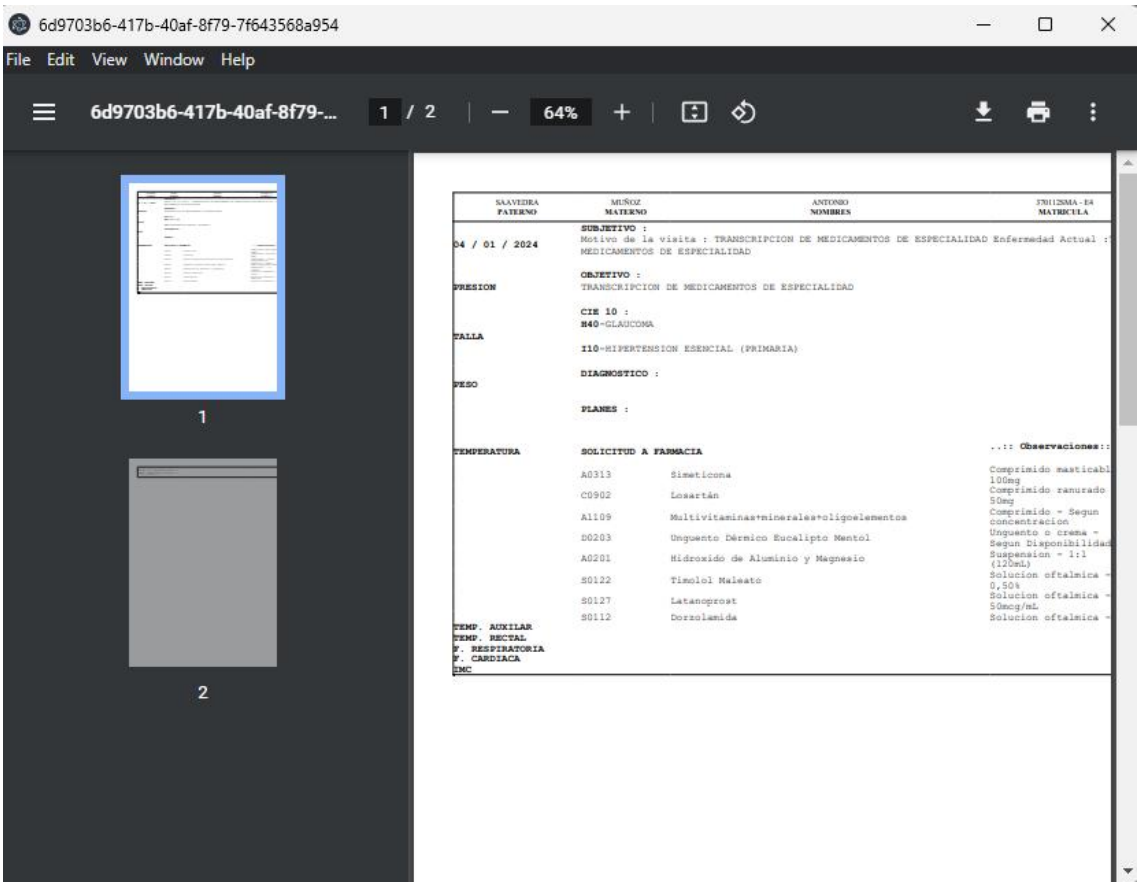
Fuente: Elaboración propia



Descripción: *Una vez se suba el documento en pdf, este aparecerá en verde, haciendo clic en la parte verde se abrirá una ventana con el documento escaneado para que se pueda observar, para eliminar se presiona el botón rojo y para actualizar el botón amarillo*

Ilustración 48: *Modal de visualización de documentos.*

Fuente: Elaboración propia



3.4.6 SP6: Configurar en el sistema los documentos correspondientes para cada tipo de beneficiario

3.4.6.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar la funcionalidad que permita configurar los documentos correspondientes para cada tipo de beneficiario (esposa, hijos, padres) en el sistema, asegurando que los documentos requeridos estén definidos, almacenados y gestionados adecuadamente para su uso en procesos administrativos y médicos.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H11** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 39: Sprint Backlog SP6.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H11-T1	Definir los tipos de documentos para cada tipo de beneficiario	Identificar y documentar los tipos de documentos requeridos para cada beneficiario.	Ninguno	Muy Alta
H11-T2	Diseñar las tablas de la base de datos para almacenar la configuración de documentos por beneficiario	Crear las tablas necesarias en la base de datos para almacenar la configuración de documentos asignados a cada tipo de beneficiario. Esto incluirá la relación entre los beneficiarios y los documentos requeridos.	H11-T1	Alta
H11-T3	Crear los modelos en Laravel para gestionar la configuración de documentos	Desarrollar los modelos en Laravel que gestionen la asociación entre los beneficiarios y los documentos asignados a ellos, estableciendo las relaciones necesarias.	H11-T2	Alta
H11-T4	Implementar las migraciones de la base de datos para la	Crear las migraciones en Laravel para reflejar las estructuras de las tablas necesarias para la	H11-T3	Alta

	configuración de documentos	configuración de documentos por tipo de beneficiario.		
H11-T5	Desarrollar el backend para la configuración de documentos	Implementar los controladores en Laravel que permitan agregar, editar y gestionar la configuración de los documentos asociados a cada tipo de beneficiario.	H11-T4	Alta
H11-T6	Crear las rutas para la gestión de documentos	Definir las rutas necesarias en Laravel para que los controladores puedan gestionar la configuración de documentos para los beneficiarios. Estas rutas permitirán crear, leer, actualizar y eliminar configuraciones.	H11-T5	Alta
H11-T7	Crear la interfaz de usuario para configurar los documentos	Desarrollar el componente en React que permita a los administradores del sistema configurar y visualizar los documentos necesarios para cada tipo de beneficiario.	H11-T1 H11-T6	Media
H11-T8	Conectar el frontend con el backend	Configurar la comunicación entre el frontend (React) y el backend (Laravel) para que los cambios realizados en la interfaz de usuario se reflejen en la base de datos mediante peticiones HTTP seguras.	H11-T6 H11-T7	Alta
H11-T9	Validar la configuración de documentos en el frontend	Implementar validaciones en el frontend para asegurar que los documentos configurados cumplan con los requerimientos para cada tipo de beneficiario.	H11-T8	Media
H11-T10	Pruebas funcionales del módulo de configuración de documentos	Realizar pruebas para asegurarse de que los administradores puedan agregar, editar y gestionar correctamente la configuración de los documentos por tipo de beneficiario.	H11-T9	Media
H11-T11	Documentar el módulo de configuración de documentos	Crear la documentación técnica que describa la estructura de la base de datos, las rutas, controladores y vistas relacionadas con la configuración de documentos por tipo de beneficiario.	Todas	Baja

3.4.6.2 Sprint Review

Tabla 40: Tabla resumen del sprint review SP6.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H11-T1	Definir los tipos de documentos para cada tipo de beneficiario	Completada
H11-T2	Diseñar las tablas de la base de datos para almacenar la configuración de documentos por beneficiario	Completada
H11-T3	Crear los modelos en Laravel para gestionar la configuración de documentos	Completada
H11-T4	Implementar las migraciones de la base de datos para la configuración de documentos	Completada
H11-T5	Desarrollar el backend para la configuración de documentos	Completada
H11-T6	Crear las rutas para la gestión de documentos	Completada
H11-T7	Crear la interfaz de usuario para configurar los documentos	Completada
H11-T8	Conectar el frontend con el backend	Completada
H11-T9	Validar la configuración de documentos en el frontend	Completada
H11-T10	Pruebas funcionales del módulo de configuración de documentos	Completada
H11-T11	Documentar el módulo de configuración de documentos	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición de tipos de documentos por beneficiario	Se identificaron y documentaron los tipos de documentos requeridos para cada beneficiario, asegurando que la información esté clara y accesible.	
Diseño de la base de datos	Se diseñaron las tablas necesarias para almacenar la configuración de documentos asignados a cada tipo de beneficiario, estableciendo la relación adecuada entre ambos.	
Implementación en Laravel	Se desarrollaron los modelos en Laravel para gestionar la asociación entre los beneficiarios y los documentos asignados a ellos, estableciendo las relaciones necesarias.	
Creación de migraciones	Se implementaron las migraciones en Laravel para reflejar las estructuras de las tablas necesarias para la configuración de documentos por tipo de beneficiario.	

Desarrollo del backend	Se implementaron controladores en Laravel que permiten agregar, editar y gestionar la configuración de los documentos asociados a cada tipo de beneficiario.
Definición de rutas	Se definieron las rutas necesarias en Laravel para que los controladores gestionen la configuración de documentos, permitiendo crear, leer, actualizar y eliminar configuraciones.
Desarrollo de la interfaz de usuario	Se creó un componente en React que permite a los administradores del sistema configurar y visualizar los documentos necesarios para cada tipo de beneficiario.
Integración del frontend y backend	Se configuró la comunicación entre el frontend (React) y el backend (Laravel) para que los cambios realizados en la interfaz de usuario se reflejen en la base de datos.
Validación de configuración en el frontend	Se implementaron validaciones en el frontend para asegurar que los documentos configurados cumplan con los requerimientos para cada tipo de beneficiario.
Pruebas funcionales del módulo	Se realizaron pruebas para asegurarse de que los administradores puedan agregar, editar y gestionar correctamente la configuración de los documentos por tipo de beneficiario.
Documentación del módulo	Se creó la documentación técnica que describe la estructura de la base de datos, las rutas, controladores y vistas relacionadas con la configuración de documentos por tipo de beneficiario.

3.4.6.3 Sprint Retrospective

Tabla 41: Tabla resumen del sprint retrospective SP6.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Identificación clara de requisitos: La definición de los tipos de documentos requeridos para cada beneficiario se realizó de manera efectiva, lo que facilitó el desarrollo posterior. - Estructura de base de datos bien diseñada: Las tablas para almacenar la configuración de documentos se crearon correctamente, asegurando que las relaciones entre beneficiarios y documentos fueran claras y eficientes. - Interfaz de usuario intuitiva: El componente de React para la configuración de documentos resultó ser amigable y fácil de usar para los administradores, permitiendo una gestión efectiva de los documentos necesarios.
Desafíos enfrentados
- Complejidad en la gestión de relaciones: Hubo dificultades para establecer las relaciones adecuadas entre los beneficiarios y los documentos en los modelos de Laravel, lo que causó retrasos en el desarrollo.

- Problemas en la validación del frontend: La implementación de validaciones en el frontend tomó más tiempo del esperado, ya que era necesario garantizar que los documentos configurados cumplieran con los requisitos específicos.
Mejoras para futuros sprints
- Reforzar las validaciones tempranas: Implementar validaciones básicas en el backend antes de la implementación en el frontend, lo que podría ayudar a identificar errores más rápidamente y reducir la carga de validación en el cliente..

Ilustración 49: *Diseño de la base de datos de beneficiarios con las tablas de configuración y asignación de documentos.*

Fuente: Elaboración propia

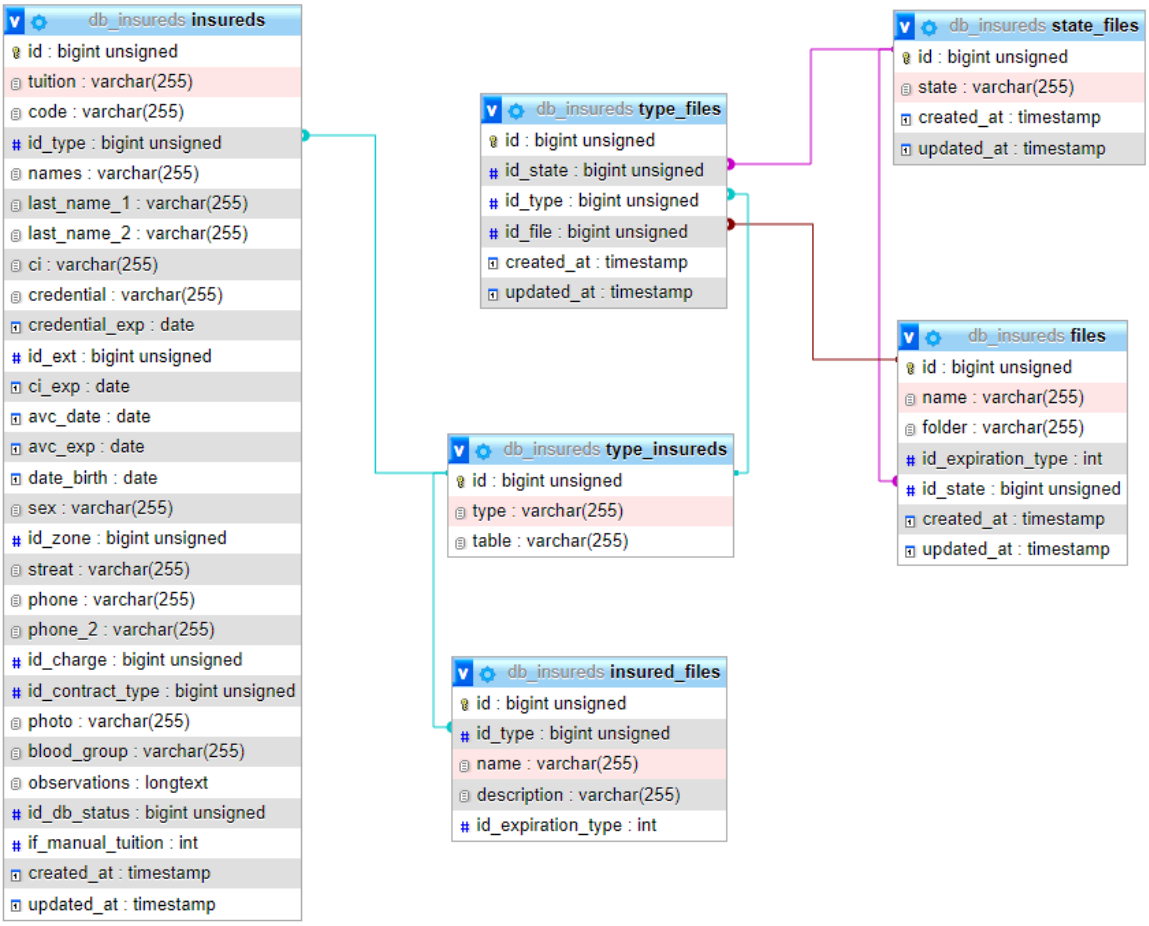


Ilustración 50: *Tabla de configuración y listado de documentos.*

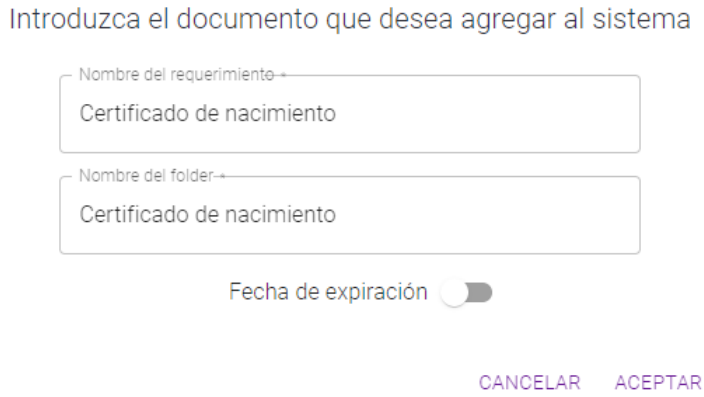
Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de los documentos de los beneficiarios registrados en el sistema.*

Ilustración 51: *Tabla de configuración y listado de documentos.*

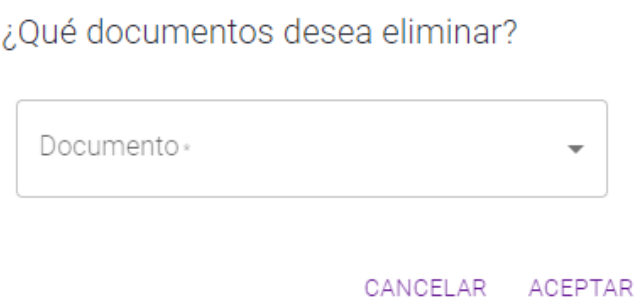
Fuente: Elaboración propia



Descripción: *Modal empleado para la edición de documentos, se abre haciendo clic en uno de los documentos*

Ilustración 52: *Tabla de configuración y listado de documentos.*

Fuente: Elaboración propia



Descripción: *Modal empleado para eliminar documentos, se abre haciendo clic en uno de los documentos*

Ilustración 53: *Tabla de asignación de documentos según el tipo de beneficiario.*

Fuente: Elaboración propia

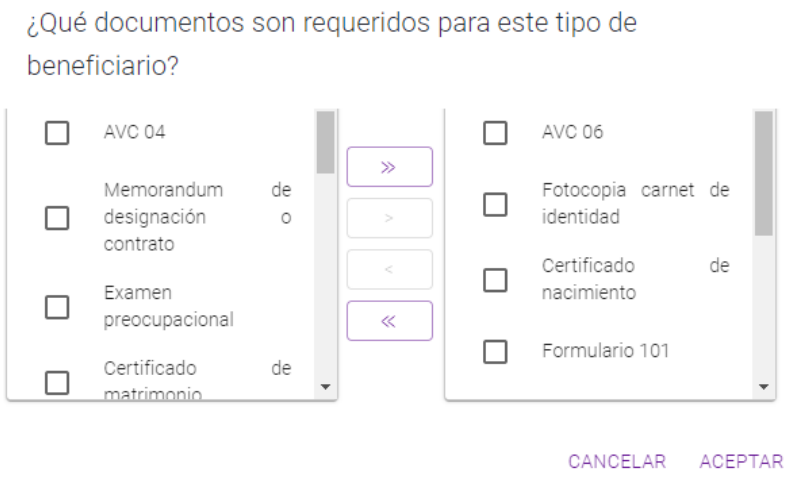
Asignación de requerimientos



Descripción: *Vista empleada para la visualización de los tipos de beneficiarios registrados en el sistema.*

Ilustración 54: *Modal de asignación de documentos según el beneficiario.*

Fuente: Elaboración propia



Descripción: *Modal empleado para la asignación de documentos según el tipo de beneficiario, se abre haciendo clic en uno de los tipos de beneficiarios*

3.4.7 SP7: Modificar, almacenar y actualizar la documentación periódica, asegurando la inclusión de nuevos documentos requeridos

3.4.7.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar una funcionalidad que permita modificar, actualizar y almacenar documentos de manera continua, asegurando que los nuevos requisitos documentales se integren sin problemas en el flujo de trabajo de gestión de documentos del sistema.

b. Sprint backlog

En base a que el sprint involucra a las historias de usuario **H7 y H9** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 42: Sprint Backlog SP7.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H7-T1	Definir los tipos de documentación periódica	Identificar y documentar los tipos de documentos que se deben modificar y almacenar periódicamente (renovaciones, certificados, etc.).	Ninguno	Muy Alta
H7-T2	Agregar los campos necesarios a la base de datos para el almacenamiento de documentos periódicos	Agregar los campos a la base de datos que permitan almacenar la información de los documentos periódicos, incluyendo la fecha de emisión, fecha de caducidad y tipo de documento.	H7-T1	Alta
H7-T3	Modificar los modelos en Laravel para gestionar los documentos periódicos	Editar los modelos de Laravel que gestionen la modificación, actualización y almacenamiento de documentos periódicos, estableciendo relaciones con los usuarios o asegurados.	H7-T2	Alta
H7-T4	Implementar las migraciones de la base de datos	Crear las migraciones necesarias para reflejar las estructuras de las tablas de almacenamiento de documentos periódicos y la relación con los asegurados.	H7-T3	Alta

H7-T5	Desarrollar el backend para modificar y almacenar documentación periódica	Implementar los controladores en Laravel que gestionen la creación, modificación y almacenamiento de documentos periódicos, asegurando que se puedan actualizar y guardar correctamente en la base de datos.	H7-T4	Alta
H7-T6	Crear las rutas para la gestión de documentación periódica	Definir las rutas necesarias en Laravel para permitir a los controladores gestionar las acciones CRUD (crear, leer, actualizar y eliminar) de los documentos periódicos.	H7-T5	Alta
H7-T7	Desarrollar la funcionalidad para la inclusión de nuevos documentos	Crear una funcionalidad en el backend que permita agregar nuevos tipos de documentos al sistema y actualice la base de datos para aceptar estos nuevos documentos.	H7-T5	Alta
H7-T8	Crear la interfaz de usuario para modificar y actualizar documentos	Desarrollar el componente de React que permita a los administradores modificar y actualizar la documentación periódica desde una interfaz gráfica fácil de usar.	H7-T1 H7-T6	Media
H7-T9	Conectar el frontend con el backend para gestión de documentos	Configurar la comunicación entre el frontend (React) y el backend (Laravel) para asegurar que los cambios y actualizaciones realizados en la interfaz de usuario se reflejen en la base de datos.	H7-T6 H7-T8	Alta
H7-T10	Validar la entrada y modificación de documentos	Implementar validaciones tanto en el frontend como en el backend para asegurarse de que los documentos almacenados y modificados cumplan con los requisitos.	H7-T9	Media
H7-T11	Pruebas funcionales del módulo de documentación periódica	Realizar pruebas para asegurarse de que los administradores puedan modificar, almacenar y actualizar documentos periódicos correctamente, y que se puedan incluir nuevos tipos de documentos en el sistema.	H7-T10	Media
H7-T12	Documentar el módulo de modificación y actualización de documentos	Crear la documentación técnica que describa el funcionamiento, la estructura de la base de datos, las rutas, los controladores y las vistas relacionadas con la gestión de la documentación periódica.	Todas	Baja

3.4.7.2 Sprint Review

Tabla 43: Tabla resumen del sprint review SP7.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H7-T1	Definir los tipos de documentación periódica	Completada
H7-T2	Agregar los campos necesarios a la base de datos para el almacenamiento de documentos periódicos	Completada
H7-T3	Modificar los modelos en Laravel para gestionar los documentos periódicos	Completada
H7-T4	Implementar las migraciones de la base de datos	Completada
H7-T5	Desarrollar el backend para modificar y almacenar documentación periódica	Completada
H7-T6	Crear las rutas para la gestión de documentación periódica	Completada
H7-T7	Desarrollar la funcionalidad para la inclusión de nuevos documentos	Completada
H7-T8	Crear la interfaz de usuario para modificar y actualizar documentos	Completada
H7-T9	Conectar el frontend con el backend para gestión de documentos	Completada
H7-T10	Validar la entrada y modificación de documentos	Completada
H7-T11	Pruebas funcionales del módulo de documentación periódica	Completada
H7-T12	Documentar el módulo de modificación y actualización de documentos	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición de tipos de documentación periódica	Se identificaron y documentaron los tipos de documentos que deben ser modificados y almacenados periódicamente, como renovaciones y certificados.	
Adición de campos a la base de datos	Se agregaron los campos necesarios a la base de datos para almacenar información sobre documentos periódicos, incluyendo fecha de emisión, fecha de caducidad y tipo de documento.	
Modificación de modelos en Laravel	Se editaron los modelos en Laravel para gestionar la modificación, actualización y almacenamiento de documentos periódicos, estableciendo relaciones con los usuarios o asegurados.	

Implementación de migraciones	Se crearon las migraciones necesarias para reflejar las estructuras de las tablas que almacenan documentos periódicos y su relación con los asegurados.
Desarrollo del backend	Se implementaron controladores en Laravel que gestionan la creación, modificación y almacenamiento de documentos periódicos, asegurando que se puedan actualizar y guardar correctamente.
Creación de rutas	Se definieron las rutas necesarias en Laravel para permitir a los controladores gestionar las acciones CRUD (crear, leer, actualizar y eliminar) de los documentos periódicos.
Desarrollo de funcionalidad para nuevos documentos	Se creó una funcionalidad en el backend que permite agregar nuevos tipos de documentos al sistema y actualiza la base de datos para aceptar estos nuevos documentos.
Desarrollo de la interfaz de usuario	Se desarrolló un componente en React que permite a los administradores modificar y actualizar la documentación periódica desde una interfaz gráfica fácil de usar.
Conexión entre frontend y backend	Se configuró la comunicación entre el frontend (React) y el backend (Laravel) para asegurar que los cambios y actualizaciones realizados en la interfaz de usuario se reflejen en la base de datos.
Validación de entrada y modificación de documentos	Se implementaron validaciones tanto en el frontend como en el backend para asegurarse de que los documentos almacenados y modificados cumplan con los requisitos establecidos.
Pruebas funcionales del módulo	Se realizaron pruebas para asegurarse de que los administradores puedan modificar, almacenar y actualizar documentos periódicos correctamente, y que se puedan incluir nuevos tipos de documentos.
Documentación del módulo	Se creó la documentación técnica que describe el funcionamiento, la estructura de la base de datos, las rutas, los controladores y las vistas relacionadas con la gestión de la documentación periódica.

3.4.7.3 Sprint Retrospective

Tabla 44: Tabla resumen del sprint retrospective SP7.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Identificación precisa de documentos: La fase inicial para definir los tipos de documentación periódica fue exitosa. Identificamos claramente los documentos necesarios, como renovaciones y certificados, lo que facilitó el desarrollo de funcionalidades específicas.

- Actualización de la base de datos efectiva: La adición de nuevos campos para almacenar información crucial, como fechas de emisión y caducidad, se realizó sin inconvenientes. Esto garantizó que la base de datos pudiera manejar adecuadamente los documentos periódicos. - Interfaz amigable y funcional: El componente en React diseñado para la modificación y actualización de documentos fue bien recibido por los administradores. Su diseño intuitivo permitió un manejo ágil y efectivo de los documentos.
Desafíos enfrentados
- Validaciones inconsistentes: La implementación de validaciones para la entrada y modificación de documentos reveló fallos en algunas entradas, lo que llevó a errores que fueron difíciles de rastrear. Esto nos enseñó que una validación más robusta desde el inicio podría haber mitigado estos problemas.
Mejoras para futuros sprints
- Definición clara de rutas en la etapa inicial: Incluir una revisión exhaustiva de las rutas desde el principio podría evitar malentendidos durante la fase de desarrollo. Esto aseguraría que todos los miembros del equipo estén alineados sobre cómo interactuarán las diferentes partes del sistema.

Ilustración 55: *Modal de carpetas con archivos en el servidor para almacenamiento periódico.*

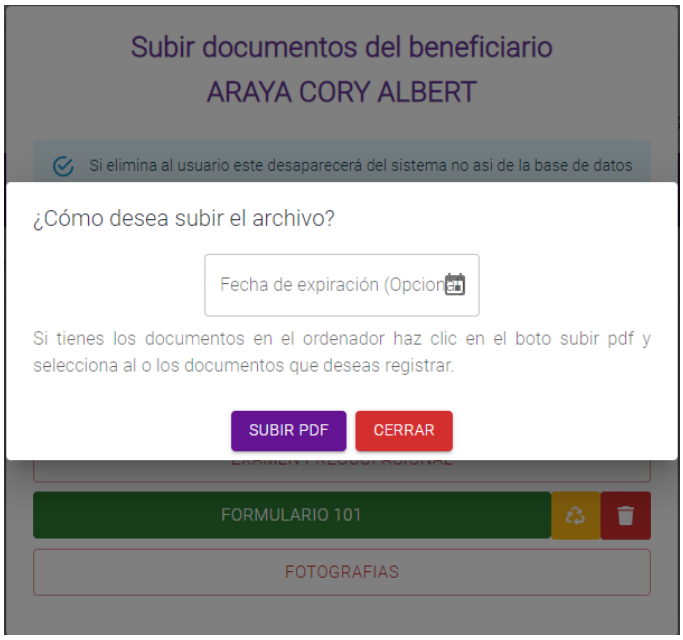
Fuente: Elaboración propia



Descripción: *Modal empleado para la visualización de documentos en formato carpeta, se abre haciendo clic en ver documentos de la tabla de beneficiarios*

Ilustración 56: *Modal de actualización de documentos después de seleccionar el botón amarillo.*

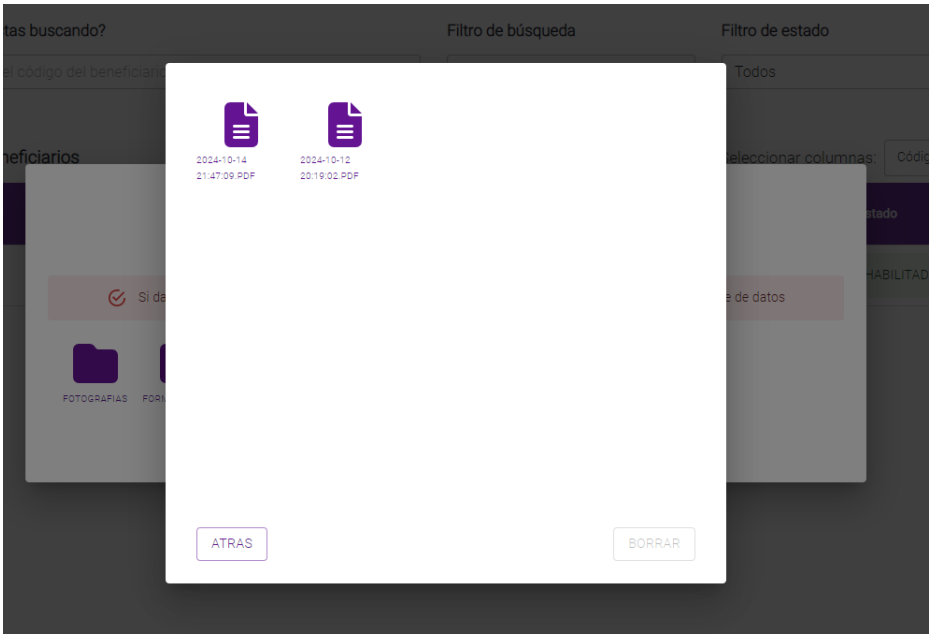
Fuente: Elaboración propia



Descripción: *Modal empleado para la actualización de documentos, cuenta con un campo fecha de vencimiento opcional para generar alertas cuando el documento haya caducado*

Ilustración 57: *Modal de verificación de documentos guardados por fecha de registro.*

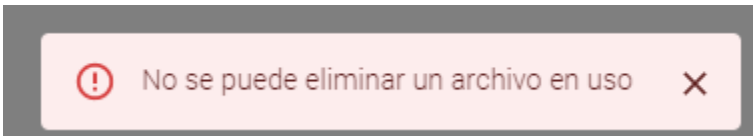
Fuente: Elaboración propia



Descripción: *Modal empleado para la visualización de documentos ordenados según fecha, para eliminar simplemente se debe hacer un clic en el archivo y luego presionar eliminar, si desea verlo se debe hacer dos clics en el documento*

Ilustración 58: *Modal de verificación de documentos guardados por fecha de registro.*

Fuente: Elaboración propia



Descripción: *Alerta que se muestra al intentar eliminar un documento que esta en uso, para eliminar el documento debe dirigirse a l modal de gestión de documentos.*

3.4.8 SP8: Implementar una funcionalidad de registro de usuarios seguro y eficiente.

3.4.8.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar una funcionalidad de registro de usuarios que permita crear, almacenar y gestionar cuentas de manera segura y eficiente, asegurando que los datos de los usuarios estén protegidos y que el proceso de registro sea fácil de usar.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H16** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 45: Sprint Backlog SP8.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H16-T1	Definir los requisitos de datos para el registro de usuarios	Identificar los campos necesarios para el registro de usuarios, como nombre, correo electrónico, contraseña, confirmación de contraseña, y cualquier dato adicional requerido (rol, teléfono, etc.).	Ninguno	Muy Alta
H16-T2	Diseñar la base de datos para usuarios	Crear las tablas necesarias en la base de datos para almacenar la información de los usuarios, incluyendo los campos definidos y asegurando que los datos estén debidamente estructurados y seguros.	H16-T1	Alta
H16-T3	Implementar modelos de usuarios en Laravel	Crear el modelo de usuario en Laravel que gestione la creación, almacenamiento y actualización de la información de los usuarios. Implementar relaciones y validaciones necesarias.	H16-T2	Alta

H16-T4	Implementar migraciones para la base de datos de usuarios	Crear las migraciones necesarias en Laravel para reflejar la estructura de la base de datos de usuarios, asegurando la consistencia entre el código y la base de datos.	H16-T3	Alta
H22 H16T5	Desarrollar controladores para el registro de usuarios	Implementar los controladores en Laravel que gestionen el flujo del registro de usuarios, asegurando la validación de los datos y la creación de la cuenta de usuario en la base de datos.	H16-T4	Alta
H16-T6	Implementar validación y encriptación de contraseñas	Utilizar los mecanismos de validación y encriptación de Laravel (bcrypt) para asegurar que las contraseñas se almacenen de manera segura y que las entradas de los formularios sean válidas.	H16-T5	Media
H16-T7	Crear las rutas de registro de usuarios	Definir las rutas en Laravel para gestionar las acciones de registro de nuevos usuarios. Estas rutas permitirán acceder al formulario de registro y procesar la creación de cuentas.	H16-T5	Alta
H16-T8	Desarrollar el formulario de registro de usuarios en React	Crear el componente de formulario en React que permita a los usuarios ingresar sus datos para registrarse, con validaciones básicas en el frontend (campos obligatorios, formato de email, etc.).	H16-T1	Media
H16-T9	Conectar el frontend con el backend para el registro de usuarios	Configurar la comunicación entre el frontend (React) y el backend (Laravel) para enviar los datos del formulario de registro y recibir respuestas del servidor.	H16-T7 H16-T8	Alta
H16-T10	Realizar pruebas del flujo de registro de usuarios	Ejecutar pruebas funcionales para asegurarse de que los usuarios puedan registrarse correctamente, sus datos se almacenen de manera segura, y se reciban los correos de verificación.	H16-T10	Media
H16-T11	Documentar el flujo de registro de usuarios	Crear la documentación técnica que describa el proceso de registro de usuarios, la estructura de la base de datos, los modelos,	Todas	Baja

		las rutas y los controladores relacionados.		
H16-T12	Implementar una verificación JWT en todos los servicios	Configurar la autenticación basada en tokens JWT en todos los servicios, asegurando que cada solicitud verifique la validez del token antes de procesar cualquier acción.	H16-T9	Alta
H16-T13	Crear las rutas protegidas de cada servicio	Definir rutas seguras en cada servicio que solo permitan acceso a usuarios autenticados, protegiendo los recursos y datos sensibles mediante la verificación del JWT.	H16-T12	Alta
H16-T14	Implementar rutas públicas para acceso sin necesidad de login	Configurar rutas accesibles sin autenticación para que los usuarios puedan acceder a determinados servicios o recursos sin necesidad de iniciar sesión.	H16-T13	Alta
H16-T15	Bloquear el acceso a los documentos solo para usuarios registrados	Restringir el acceso a los documentos o archivos confidenciales, asegurando que solo los usuarios autenticados y registrados puedan visualizarlos o descargarlos.	H16-T14	Alta

3.4.8.2 Sprint Review

Tabla 46: Tabla resumen del sprint review SP8.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H6-T1	Definir los tipos de documentos a escanear y almacenar	Completada
H6-T2	Diseñar las tablas de la base de datos para almacenamiento de documentos	Completada
H6-T3	Crear los modelos en Laravel para gestionar los documentos	Completada
H6-T4	Implementar las migraciones de la base de datos para documentos	Completada
H6-T5	Implementar el backend para la subida segura de documentos escaneados	Completada
H6-T6	Definir las rutas para las operaciones de subida y consulta de documentos	Completada

H6-T7	Crear el componente de React para la subida de documentos escaneados	Completada
H6-T8	Implementar la validación de los documentos en el frontend	Completada
H6-T9	Conectar el frontend con el backend para la subida de documentos	Completada
H6-T10	Implementar el almacenamiento seguro de documentos en el servidor	Completada
H6-T11	Implementar cifrado para los documentos	Completada
H16-T12	Implementar una verificación JWT en todos los servicios	Completada
H16-T13	Crear las rutas protegidas de cada servicio	Completada
H16-T14	Implementar rutas públicas para acceso sin necesidad de login	Completada
H16-T15	Bloquear el acceso a los documentos solo para usuarios registrados	Completada

Observaciones: Ninguna

Problemas encontrados: Ninguno

Resultados

Definición de los requisitos de datos para el registro de usuarios	Se identificaron los campos necesarios para el registro de usuarios, como nombre, correo electrónico, contraseña, confirmación de contraseña, y otros datos adicionales (rol, etc.).
Diseño de la base de datos para usuarios	Se crearon las tablas necesarias en la base de datos para almacenar la información de los usuarios, asegurando que los datos estén estructurados y sean seguros.
Implementación de modelos de usuarios en Laravel	Se creó el modelo de usuario en Laravel, gestionando la creación, almacenamiento y actualización de la información de los usuarios, con relaciones y validaciones necesarias.
Implementación de migraciones para la base de datos de usuarios	Se desarrollaron las migraciones en Laravel para reflejar la estructura de la base de datos de usuarios, asegurando consistencia entre el código y la base de datos.
Desarrollo de controladores para el registro de usuarios	Se implementaron los controladores en Laravel para gestionar el flujo del registro de usuarios, validando los datos y creando las cuentas de usuario en la base de datos.
Implementación de validación y encriptación de contraseñas	Se utilizaron los mecanismos de validación y encriptación de Laravel (bcrypt) para asegurar que las contraseñas se almacenen de manera segura y las entradas sean válidas.
Creación de las rutas de registro de usuarios	Se definieron las rutas en Laravel para gestionar las acciones de registro de nuevos usuarios, permitiendo acceder al formulario y procesar la creación de cuentas.
Desarrollo del formulario de registro de usuarios en React	Se creó el componente de formulario en React para permitir a los usuarios ingresar sus datos y registrarse, con validaciones básicas en el frontend.

Conexión del frontend con el backend para el registro de usuarios	Se configuró la comunicación entre React y Laravel para enviar los datos del formulario de registro y recibir respuestas del servidor.
Pruebas del flujo de registro de usuarios	Se realizaron pruebas funcionales para asegurar que los usuarios puedan registrarse correctamente, sus datos se almacenen de manera segura y reciban los correos de verificación.
Documentación del flujo de registro de usuarios	Se creó la documentación técnica que describe el proceso de registro de usuarios, la estructura de la base de datos, los modelos, rutas y controladores relacionados.
Implementación de verificación JWT en todos los servicios	Se configuró la autenticación basada en tokens JWT en todos los servicios, asegurando la validez de los tokens antes de procesar cualquier acción.
Creación de rutas protegidas de cada servicio	Se definieron rutas seguras para que solo los usuarios autenticados puedan acceder a servicios y recursos protegidos mediante la verificación del JWT.
Implementación de rutas públicas	Se configuraron rutas accesibles sin autenticación, permitiendo el acceso a determinados recursos sin necesidad de iniciar sesión.
Bloqueo del acceso a documentos solo para usuarios registrados	Se restringió el acceso a documentos confidenciales, permitiendo solo a usuarios autenticados visualizarlos o descargarlos.

3.4.8.3 Sprint Retrospective

Tabla 47: Tabla resumen del sprint retrospective SP8.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Definición de requisitos para el registro de usuarios (H16-T1): Se logró identificar de manera clara los campos necesarios para el registro de usuarios, como el nombre, correo electrónico, y contraseñas, asegurando que todas las necesidades de los usuarios fueran cubiertas desde el inicio. - Diseño de la base de datos (H16-T2): La base de datos fue diseñada eficientemente para almacenar de manera segura la información de los usuarios. Se implementaron correctamente las tablas necesarias, lo que facilitó la integración con el backend. - Validación y encriptación de contraseñas (H16-T6): La encriptación de contraseñas con bcrypt se implementó correctamente, asegurando que las contraseñas de los usuarios estén seguras y almacenadas de manera adecuada.
Desafíos enfrentados
- Implementación de rutas protegidas y JWT (H16-T12, H16-T13): La configuración de la autenticación JWT resultó más compleja de lo esperado. Algunas rutas protegidas no validaban correctamente los tokens, lo que causó problemas de seguridad que debieron ser corregidos. - Conexión del frontend con el backend para el registro de usuarios (H16-T9): Durante la conexión entre el frontend (React) y el backend (Laravel), se encontraron problemas con la gestión de respuestas del servidor. Estos errores hicieron que el registro de usuarios fallara intermitentemente.

- Formulario de registro de usuarios en React (H16-T8): El diseño del formulario fue bien ejecutado, pero algunos usuarios reportaron problemas con las validaciones del frontend, lo que llevó a realizar ajustes para asegurar que los campos requeridos y formatos se validaran correctamente.
Mejoras para futuros sprints
- Revisar la lógica JWT en las rutas protegidas: Se necesita una revisión más detallada de la implementación de JWT, especialmente en la validación de tokens. Una opción sería implementar pruebas automatizadas para validar que las rutas protegidas funcionen como se espera. - Optimización de la comunicación frontend-backend: Se sugiere mejorar el manejo de errores entre el frontend y el backend, añadiendo feedback más claro para los usuarios cuando el registro no se completa correctamente. También se podría incluir un mejor manejo de excepciones en el backend. - Documentación del flujo de registro de usuarios (H16-T11): Aunque se creó una documentación técnica, esta debe ser ampliada para cubrir casos de errores comunes y mejores prácticas para la integración JWT, además de ejemplos prácticos para otros desarrolladores.

Ilustración 59: *Diseño de la base de datos de usuarios.*

Fuente: Elaboración propia

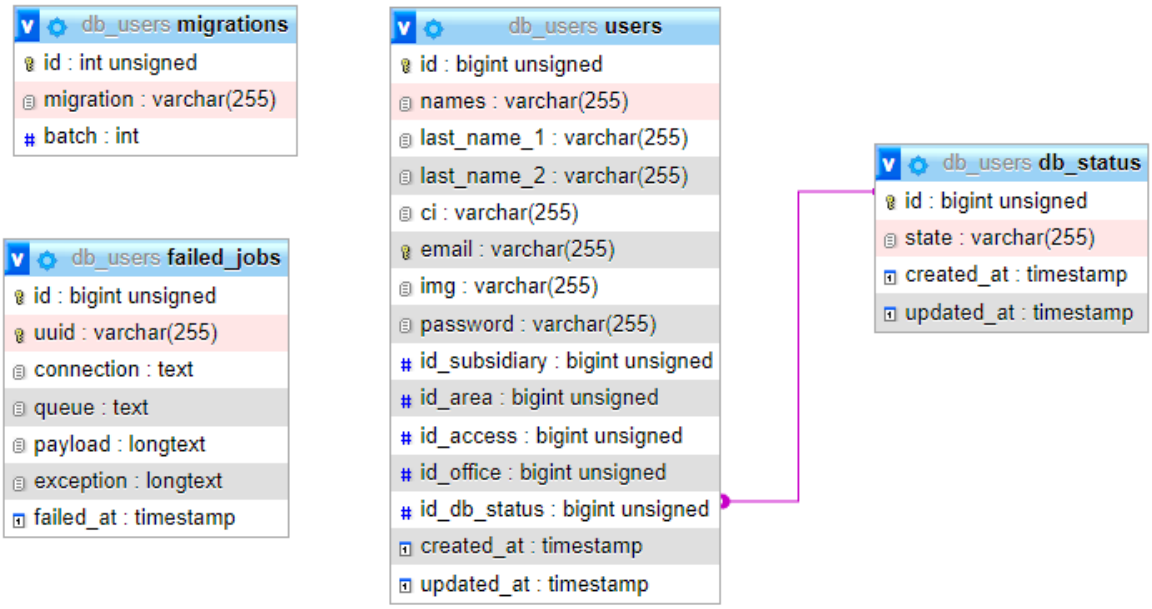
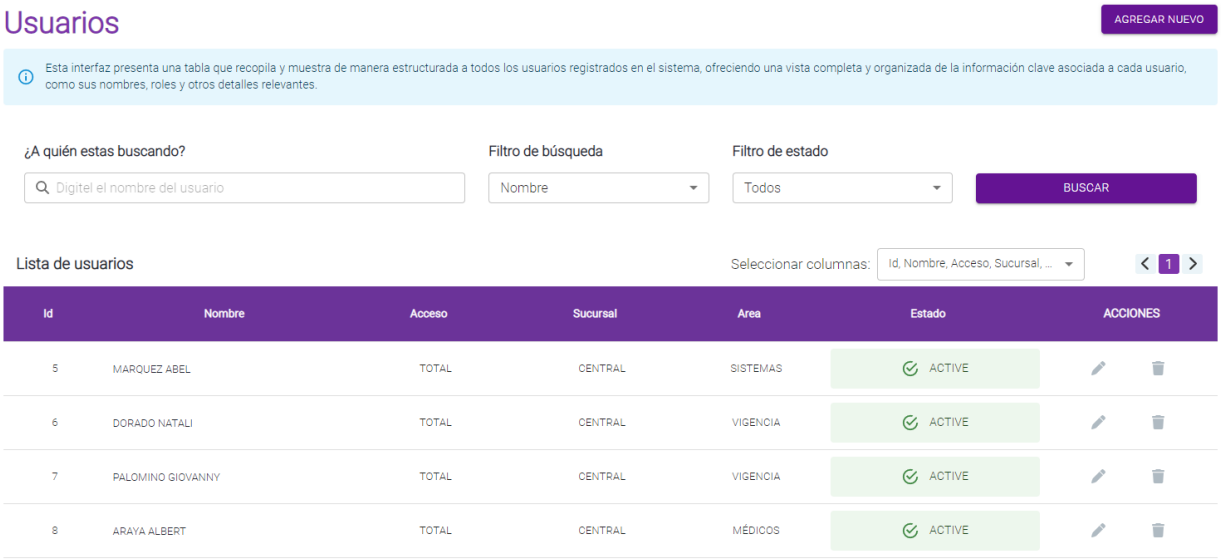


Ilustración 60: *Tabla de usuarios registrados en el sistema con sus respectivas opciones.*

Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de los usuarios, mostrando una serie de acciones en la parte derecha que abren modales de configuración y registro de acciones.*

Ilustración 61: *Modal para el registro de nuevos usuarios al sistema.*

Fuente: Elaboración propia



Descripción: *Modal empleado para el registro de nuevos usuarios con sus respectivas validaciones para una mejor experiencia de los usuarios, se abre haciendo clic en el botón agregar en la parte superior de la tabla de visualización de usuarios.*

Ilustración 62: *Modal para la edición de usuarios.*

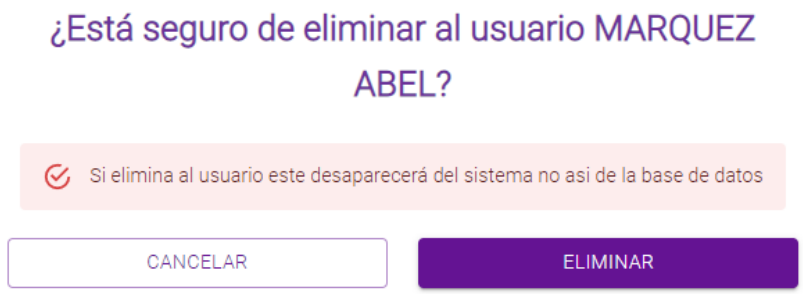
Fuente: Elaboración propia



Descripción: *Modal empleado para la edición de usuarios ya registrados en el sistema, se abre haciendo clic en la opción editar presente en la fila donde se encuentra el usuario en la tabla de visualización.*

Ilustración 63: *Modal de confirmación para la eliminación de usuarios.*

Fuente: Elaboración propia



Descripción: *Modal empleado para confirmar la eliminación de un usuario en el sistema, se abre haciendo clic en la opción eliminar presente en la fila donde se encuentra el usuario en la tabla de visualización.*

3.4.9 SP9: Implementar una funcionalidad de control de acceso basado en niveles de jerarquía de usuario

3.4.9.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar una funcionalidad de control de acceso basada en niveles de jerarquía de usuarios, asegurando que solo los usuarios autorizados puedan acceder a ciertas funcionalidades o áreas del sistema según su nivel de jerarquía, brindando una gestión efectiva de permisos y roles.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H10** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 48: Sprint Backlog SP9.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H10-T1	Definir los niveles de jerarquía de usuario	Identificar y documentar los niveles de jerarquía que se utilizarán en el sistema (por ejemplo, Administrador, Gerente, Empleado, Invitado) y las funcionalidades o áreas accesibles para cada nivel.	Ninguno	Muy Alta
H10-T2	Diseñar las tablas de la base de datos para almacenar niveles de jerarquía	Crear las tablas necesarias en la base de datos para asociar usuarios con sus niveles de jerarquía y los permisos correspondientes a cada nivel.	H10-T1	Alta
H10-T3	Crear los modelos en Laravel para gestionar jerarquías y permisos	Implementar los modelos de Laravel que gestionen los niveles de jerarquía y permisos, estableciendo las relaciones necesarias entre usuarios, roles y permisos.	H10-T2	Alta
H10-T4	Implementar las migraciones de la base de datos para jerarquías y permisos	Crear las migraciones de Laravel para reflejar las estructuras de las tablas relacionadas con los niveles de jerarquía y permisos, asegurando que la base de	H10-T3	Alta

		datos soporte la funcionalidad de control de acceso.		
H10-T5	Implementar el middleware para el control de acceso basado en jerarquías	Desarrollar un middleware en Laravel que verifique el nivel de jerarquía del usuario y restrinja el acceso a rutas o acciones que requieren un nivel específico.	H10-T3	Alta
H10-T6	Crear las rutas protegidas por niveles de jerarquía	Definir las rutas en Laravel que serán protegidas por el middleware de control de acceso según el nivel de jerarquía del usuario.	H10-T5	Media
H10-T7	Desarrollar el backend para la asignación de jerarquías a usuarios	Implementar los controladores en Laravel que permitan asignar y gestionar los niveles de jerarquía para cada usuario del sistema, asegurando que se puedan editar los roles y permisos.	H10-T3 H10-T4	Alta
H10-T8	Crear la interfaz de usuario para la gestión de jerarquías y permisos	Desarrollar el componente en React para que los administradores puedan asignar y gestionar los niveles de jerarquía de los usuarios a través de la interfaz gráfica.	H10-T7	Alta
H10-T9	Conectar el frontend con el backend para la asignación de jerarquías	Configurar la comunicación entre el frontend (React) y el backend (Laravel) para enviar y recibir los niveles de jerarquía asignados a los usuarios mediante peticiones HTTP seguras.	H10-T8	Alta
H10-T10	Implementar validaciones en el frontend para la asignación de jerarquías	Asegurar que en la interfaz de usuario se implementen validaciones al momento de asignar jerarquías, para evitar inconsistencias o errores en los roles y permisos.	H10-T9	Alta
H10-T11	Pruebas funcionales del control de acceso	Realizar pruebas funcionales para verificar que el control de acceso basado en niveles de jerarquía funciona correctamente, impidiendo el acceso no autorizado a áreas protegidas.	H10-T10	Media
H10-T12	Documentar la funcionalidad de control de acceso	Crear la documentación técnica que describa cómo funciona el control de acceso por jerarquía, incluyendo la estructura de la base de datos, middleware, rutas y vistas.	Todas	Baja

3.4.9.2 Sprint Review

Tabla 49: Tabla resumen del sprint review SP9.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H10-T1	Definir los niveles de jerarquía de usuario	Completada
H10-T2	Diseñar las tablas de la base de datos para almacenar niveles de jerarquía	Completada
H10-T3	Crear los modelos en Laravel para gestionar jerarquías y permisos	Completada
H10-T4	Implementar las migraciones de la base de datos para jerarquías y permisos	Completada
H10-T5	Implementar el middleware para el control de acceso basado en jerarquías	Completada
H10-T6	Crear las rutas protegidas por niveles de jerarquía	Completada
H10-T7	Desarrollar el backend para la asignación de jerarquías a usuarios	Completada
H10-T8	Crear la interfaz de usuario para la gestión de jerarquías y permisos	Completada
H10-T9	Conectar el frontend con el backend para la asignación de jerarquías	Completada
H10-T10	Implementar validaciones en el frontend para la asignación de jerarquías	Completada
H10-T11	Pruebas funcionales del control de acceso	Completada
H10-T12	Documentar la funcionalidad de control de acceso	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición de niveles de jerarquía de usuario	Se identificaron y documentaron los niveles de jerarquía que se utilizarán en el sistema y las funcionalidades o áreas accesibles para cada nivel.	
Diseño de tablas de la base de datos	Se crearon las tablas necesarias en la base de datos para asociar usuarios con sus niveles de jerarquía y los permisos correspondientes a cada nivel.	
Creación de modelos en Laravel	Se implementaron los modelos de Laravel que gestionan los niveles de jerarquía y permisos, estableciendo las relaciones necesarias entre usuarios, roles y permisos.	
Implementación de migraciones	Se crearon las migraciones de Laravel para reflejar las estructuras de las tablas relacionadas con los niveles de jerarquía y	

	permisos, asegurando que la base de datos soporte la funcionalidad de control de acceso.
Desarrollo de middleware	Se desarrolló un middleware en Laravel que verifica el nivel de jerarquía del usuario y restringe el acceso a rutas o acciones que requieren un nivel específico.
Creación de rutas protegidas	Se definieron las rutas en Laravel que serán protegidas por el middleware de control de acceso según el nivel de jerarquía del usuario.
Desarrollo del backend para asignación de jerarquías	Se implementaron controladores en Laravel que permiten asignar y gestionar los niveles de jerarquía para cada usuario del sistema, asegurando que se puedan editar los roles y permisos.
Desarrollo de la interfaz de usuario	Se desarrolló un componente en React para que los administradores puedan asignar y gestionar los niveles de jerarquía de los usuarios a través de la interfaz gráfica.
Conexión entre frontend y backend	Se configuró la comunicación entre el frontend (React) y el backend (Laravel) para enviar y recibir los niveles de jerarquía asignados a los usuarios mediante peticiones HTTP seguras.
Implementación de validaciones en el frontend	Se aseguraron implementaciones de validaciones en la interfaz de usuario al momento de asignar jerarquías, para evitar inconsistencias o errores en los roles y permisos.
Pruebas funcionales del control de acceso	Se realizaron pruebas funcionales para verificar que el control de acceso basado en niveles de jerarquía funciona correctamente, impidiendo el acceso no autorizado a áreas protegidas.
Documentación del control de acceso	Se creó la documentación técnica que describe cómo funciona el control de acceso por jerarquía, incluyendo la estructura de la base de datos, middleware, rutas y vistas.

3.4.9.3 Sprint Retrospective

Tabla 50: Tabla resumen del sprint retrospective SP9.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Definición clara de niveles de jerarquía: La identificación y documentación de los diferentes niveles de jerarquía, como Administrador, Gerente, Empleado e Invitado, fue exitosa. Esto proporcionó una base sólida para el desarrollo posterior y permitió a los miembros del equipo comprender las funcionalidades accesibles para cada nivel. - Interfaz de usuario intuitiva: El componente en React para gestionar jerarquías y permisos fue bien recibido por los administradores. Su diseño amigable permitió una rápida asignación y modificación de roles sin complicaciones.
Desafíos enfrentados

- Complejidad en el middleware de acceso: El desarrollo del middleware que verifica los niveles de jerarquía presentó desafíos. Inicialmente, hubo dificultades para implementar correctamente la lógica de restricción de acceso a las rutas, lo que llevó a algunos retrasos en la integración. - Confusión con las rutas protegidas: La definición de rutas protegidas por niveles de jerarquía fue un proceso complicado. La falta de claridad sobre cómo aplicar el middleware a las diferentes rutas causó malentendidos, lo que requirió varias revisiones.
Mejoras para futuros sprints
- Revisión más profunda de la lógica del middleware: Es recomendable realizar una planificación más detallada de la lógica de control de acceso desde el inicio. Esto ayudará a identificar posibles complicaciones antes de comenzar la implementación.

Ilustración 64: *Diseño de la base de datos de usuarios con las tablas de control de acceso jerárquico.*

Fuente: Elaboración propia

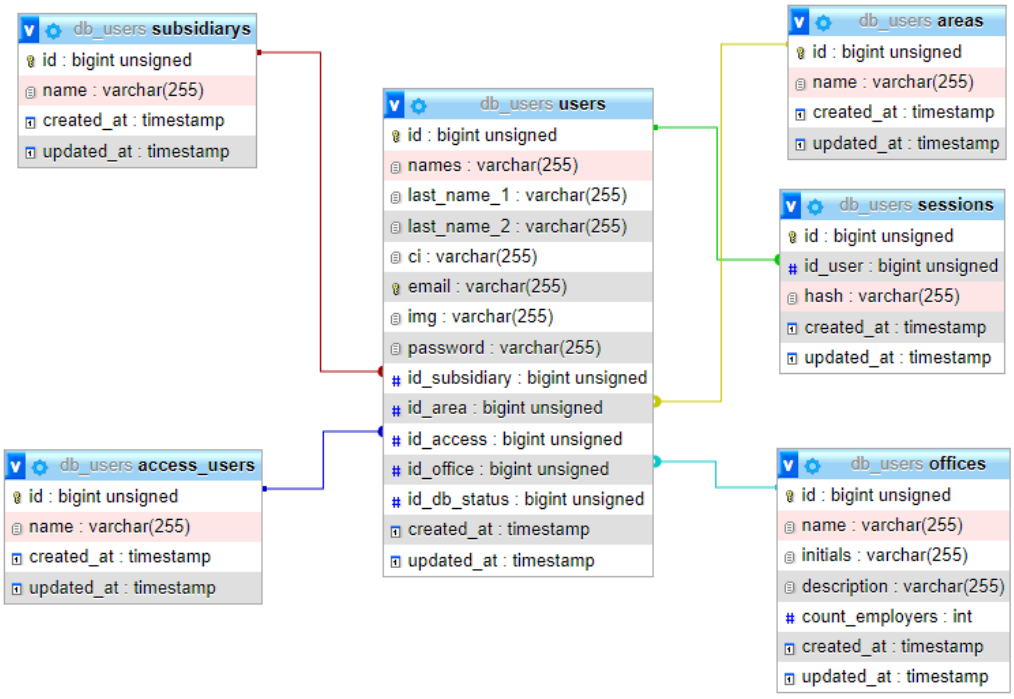


Ilustración 65: *Modal para la asignación de acceso al sistema.*

Fuente: Elaboración propia



Descripción: *Campo de nivel de acceso agregado a los modales de edición y registro de usuarios al sistema.*

Ilustración 66: *Modal para la asignación de acceso al sistema más la asignación de la sucursal.*

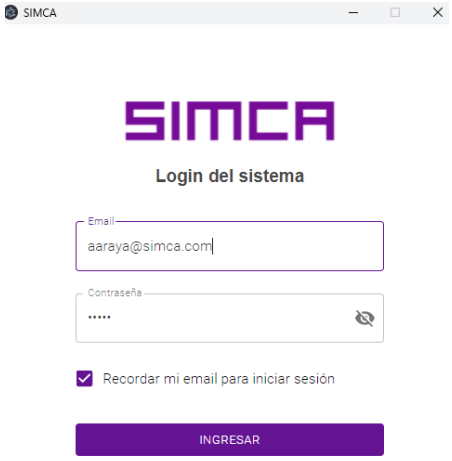
Fuente: Elaboración propia



Descripción: *Campo de sucursal agregado a los modales de edición y registro de usuarios al sistema.*

Ilustración 67: *Ventana de acceso al sistema*

Fuente: Elaboración propia



Descripción: *Ventana de acceso al sistema con sus respectivas validaciones, se abre al ejecutar el sistema.*

3.4.10 SP10: Implementar una funcionalidad para registrar solicitudes y gestionar la entrega de carnets de manera eficiente

3.4.10.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar la funcionalidad que permita registrar solicitudes de carnets y gestionar la entrega de los mismos de manera eficiente, asegurando un proceso organizado y trazable para el registro y seguimiento de las solicitudes y la posterior entrega de los carnets a los asegurados.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H17** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 51: Sprint Backlog SP10.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H17-T1	Definir los campos requeridos para las solicitudes de carnets	Identificar y documentar los datos necesarios para registrar una solicitud de carnet, como el nombre del asegurado, fecha de solicitud, número de identificación, y el estado de la solicitud.	Ninguno	Muy Alta
H17-T2	Diseñar las tablas de la base de datos para solicitudes y entrega de carnets	Crear las tablas necesarias para almacenar las solicitudes de carnets, con campos para el estado de la solicitud (pendiente, procesada, entregada) y fecha de entrega.	H17-T1	Alta
H17-T3	Crear los modelos en Laravel para gestionar solicitudes de carnets	Desarrollar los modelos de Laravel que gestionen el registro y seguimiento de las solicitudes y entrega de carnets, incluyendo validaciones y relaciones necesarias.	H17-T2	Alta
H17-T4	Implementar las migraciones de la base	Crear migraciones en Laravel para reflejar las tablas y relaciones	H17-T3	Alta

	de datos para solicitudes y entrega de carnets	relacionadas con las solicitudes y entrega de carnets.		
H17-T5	Desarrollar el backend para el registro y gestión de solicitudes de carnets	Implementar los controladores en Laravel que permitan registrar nuevas solicitudes, actualizar el estado de una solicitud y gestionar la entrega de los carnets.	H17-T4	Alta
H17-T6	Crear las rutas para el registro y gestión de solicitudes de carnets	Definir las rutas necesarias en Laravel para permitir la gestión de las solicitudes y la entrega de carnets mediante acciones CRUD (crear, leer, actualizar, eliminar).	H17-T5	Alta
H17-T7	Crear la interfaz de usuario para el registro de solicitudes de carnets	Desarrollar el componente en React para que los administradores puedan registrar solicitudes de carnets y actualizar el estado de entrega en el sistema.	H17-T1 H17-T6	Media
H17-T8	Conectar el frontend con el backend	Configurar la comunicación entre React y Laravel para registrar las solicitudes y gestionar la entrega de carnets mediante peticiones HTTP seguras.	H17-T6 H17-T7	Alta
H17-T9	Implementar la funcionalidad de seguimiento del estado de la solicitud	Permitir que los administradores del sistema puedan visualizar el estado actual de las solicitudes (pendiente, en proceso, entregada) y actualizarlas conforme se procesan.	H17-T8	Media
H17-T10	Pruebas del módulo de registro y entrega de carnets	Realizar pruebas funcionales para asegurar que las solicitudes de carnets puedan registrarse, actualizarse y entregarse correctamente, verificando la trazabilidad del proceso.	H17-T9	Media
H17-T11	Documentar el módulo de registro y entrega de carnets	Crear la documentación técnica que detalle el funcionamiento del módulo, la estructura de la base de datos, controladores, rutas y vistas relacionadas con el proceso de solicitud y entrega de carnets.	Todas	Baja

3.4.10.2 Sprint Review

Tabla 52: Tabla resumen del sprint review SP10.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H17-T1	Definir los campos requeridos para las solicitudes de carnets	Completada
H17-T2	Diseñar las tablas de la base de datos para solicitudes y entrega de carnets	Completada
H17-T3	Crear los modelos en Laravel para gestionar solicitudes de carnets	Completada
H17-T4	Implementar las migraciones de la base de datos para solicitudes y entrega de carnets	Completada
H17-T5	Desarrollar el backend para el registro y gestión de solicitudes de carnets	Completada
H17-T6	Crear las rutas para el registro y gestión de solicitudes de carnets	Completada
H17-T7	Crear la interfaz de usuario para el registro de solicitudes de carnets	Completada
H17-T8	Conectar el frontend con el backend	Completada
H17-T9	Implementar la funcionalidad de seguimiento del estado de la solicitud	Completada
H17-T10	Pruebas del módulo de registro y entrega de carnets	Completada
H17-T11	Documentar el módulo de registro y entrega de carnets	Completada
H17-T1	Definir los campos requeridos para las solicitudes de carnets	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición de campos requeridos para solicitudes de carnets	Se identificaron y documentaron los datos necesarios para registrar una solicitud de carnet, como el nombre del asegurado, fecha de solicitud, número de identificación y el estado de la solicitud.	
Diseño de tablas de la base de datos	Se crearon las tablas necesarias para almacenar las solicitudes de carnets, con campos para el estado de la solicitud (pendiente, procesada, entregada) y fecha de entrega.	
Creación de modelos en Laravel	Se desarrollaron los modelos de Laravel que gestionan el registro y seguimiento de las solicitudes y entrega de carnets, incluyendo validaciones y relaciones necesarias.	

Implementación de migraciones	Se crearon migraciones en Laravel para reflejar las tablas y relaciones relacionadas con las solicitudes y entrega de carnets.
Desarrollo del backend	Se implementaron controladores en Laravel que permiten registrar nuevas solicitudes, actualizar el estado de una solicitud y gestionar la entrega de los carnets.
Creación de rutas	Se definieron las rutas necesarias en Laravel para permitir la gestión de las solicitudes y la entrega de carnets mediante acciones CRUD (crear, leer, actualizar, eliminar).
Desarrollo de la interfaz de usuario	Se desarrolló un componente en React para que los administradores puedan registrar solicitudes de carnets y actualizar el estado de entrega en el sistema.
Conexión entre frontend y backend	Se configuró la comunicación entre React y Laravel para registrar las solicitudes y gestionar la entrega de carnets mediante peticiones HTTP seguras.
Implementación de seguimiento del estado de la solicitud	Se permitió que los administradores del sistema puedan visualizar el estado actual de las solicitudes (pendiente, en proceso, entregada) y actualizarlas conforme se procesan.
Pruebas del módulo	Se realizaron pruebas funcionales para asegurar que las solicitudes de carnets puedan registrarse, actualizarse y entregarse correctamente, verificando la trazabilidad del proceso.
Documentación del módulo	Se creó la documentación técnica que detalla el funcionamiento del módulo, la estructura de la base de datos, controladores, rutas y vistas relacionadas con el proceso de solicitud y entrega de carnets.

3.4.10.3 Sprint Retrospective

Tabla 53: Tabla resumen del sprint retrospective SP10.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Identificación de campos para solicitudes de carnets: La definición clara de los campos requeridos, como el nombre del asegurado, la fecha de solicitud, el número de identificación y el estado de la solicitud, facilitó la creación de un sistema organizado para gestionar las solicitudes. - Estructuración de la base de datos: El diseño de las tablas para almacenar las solicitudes y su estado se realizó sin contratiempos. Esto garantizó que la información estuviera bien estructurada y accesible para futuras consultas. - Interfaz de usuario funcional: El componente de React para registrar solicitudes y actualizar estados fue bien recibido. La interfaz es intuitiva, lo que permitió a los administradores manejar las solicitudes de manera efectiva.
Desafíos enfrentados

- Migraciones iniciales problemáticas: Algunas migraciones de la base de datos tuvieron que ser corregidas debido a errores en la definición de relaciones. Esto causó retrasos en la integración del sistema y obligó a revisar varias partes del código.
Mejoras para futuros sprints
- Documentación de migraciones: Mantener una documentación clara de las migraciones y su impacto en la base de datos puede ayudar a evitar errores. Es recomendable documentar cualquier cambio realizado en las estructuras de la base de datos durante el desarrollo.

Ilustración 68: *Diseño de la base de datos de beneficiarios con las tablas para solicitud y entrega de carnets.*

Fuente: Elaboración propia

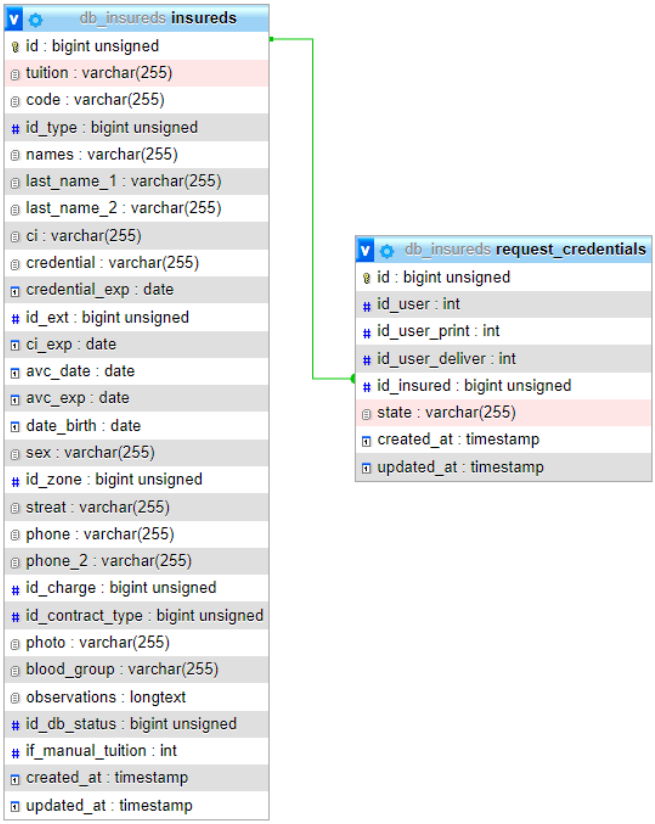


Ilustración 69: *Buscador para la verificación de credenciales.*

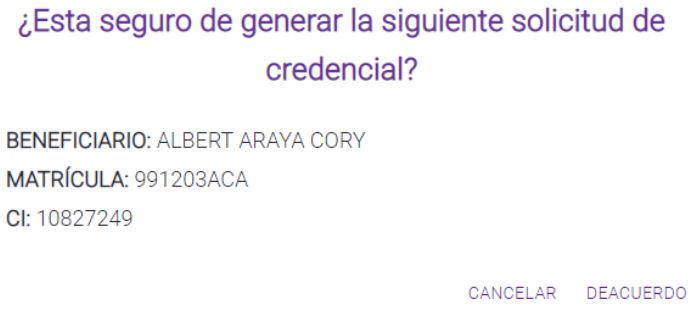
Fuente: Elaboración propia



Descripción: *Vista empleada para la verificación de los credenciales físicos entregados a los beneficiarios empleando un buscador general.*

Ilustración 70: *Modal de confirmación para la creación de una solicitud de credencial.*

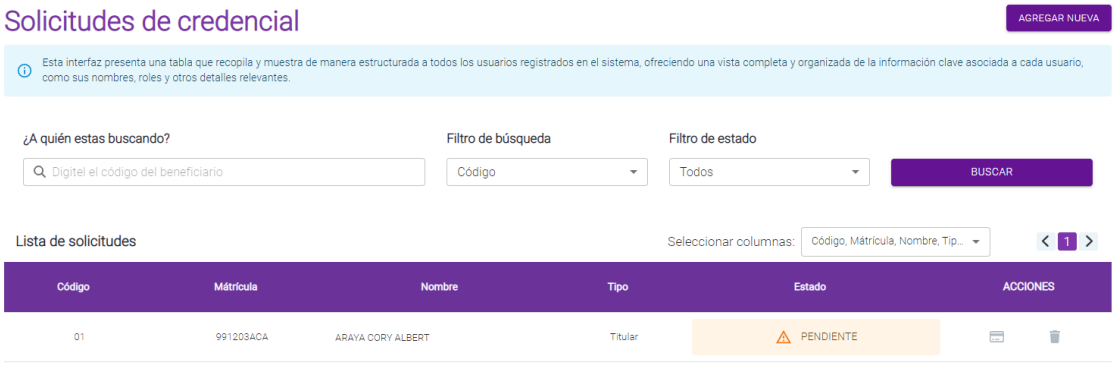
Fuente: Elaboración propia



Descripción: *Modal empleado para el registro de una solicitud de credencial, se abre haciendo clic sobre un beneficiario que no cuenta con credencial*

Ilustración 71: *Tabla de consulta de solicitudes de credenciales.*

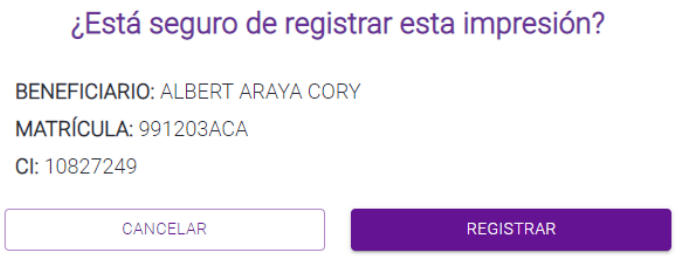
Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de las solicitudes de credencial, mostrando una serie de acciones en la parte derecha que abren modales de configuración y registro de acciones.*

Ilustración 72: *Modal para el registro de la impresión de credenciales.*

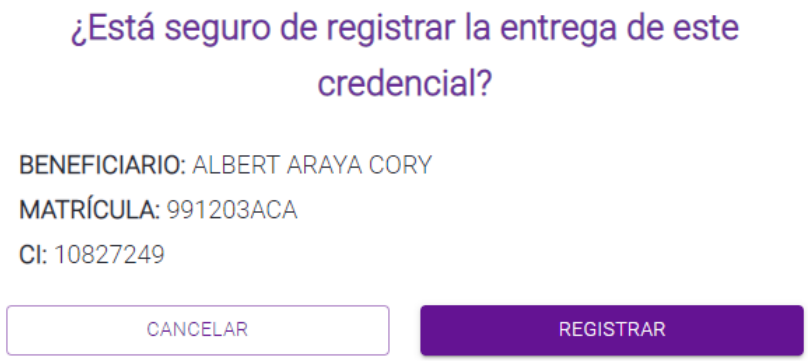
Fuente: Elaboración propia



Descripción: *Modal empleado para el registro de la impresión de una credencial, se abre haciendo clic en la opción imprimir de la tabla de consulta de solicitudes de credencial*

Ilustración 73: *Modal para el registro de la entrega de credenciales.*

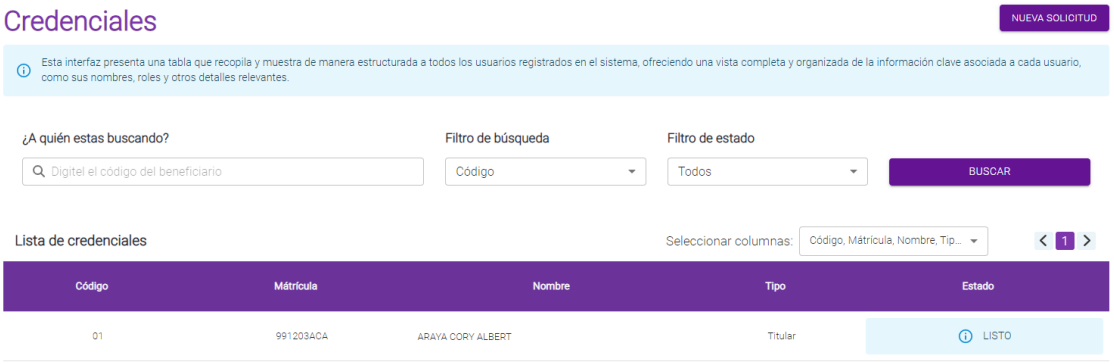
Fuente: Elaboración propia



Descripción: *Modal empleado para el registro de la impresión de una credencial, se abre haciendo clic en la opción entregar de la tabla de consulta de solicitudes de credencial*

Ilustración 74: *Tabla que lista todas las credenciales más su estado actual.*

Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de las credenciales, mostrando una serie de acciones en la parte derecha que abren modales de configuración y registro de acciones.*

3.4.11 SP11: Implementar la funcionalidad para registrar y gestionar los cambios en los horarios de atención de los médicos de manera eficiente.

3.4.11.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar la funcionalidad que permita registrar y gestionar eficientemente los cambios en los horarios de atención de los médicos, asegurando que el sistema maneje de manera adecuada las modificaciones en los turnos, fechas y horarios disponibles para la atención médica.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H18** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 54: Sprint Backlog SP11.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H18-T1	Definir los requerimientos para el registro de cambios de horarios	Identificar y documentar los detalles requeridos para el registro de los cambios de horarios, incluyendo médicos, días de la semana, horas, y cambios específicos.	Ninguno	Muy Alta
H18-T2	Diseñar las tablas de la base de datos para gestionar los horarios de los médicos	Crear las tablas en la base de datos para almacenar la información sobre los horarios de los médicos, los cambios aplicados y el historial de modificaciones.	H18-T1	Alta
H18-T3	Crear los modelos en Laravel para gestionar los horarios de atención	Implementar los modelos en Laravel para gestionar los horarios de los médicos y sus modificaciones, estableciendo relaciones entre los médicos y sus horarios de atención.	H18-T2	Alta
H18-T4	Implementar las migraciones de la base de datos	Crear las migraciones de Laravel que reflejen la estructura de las tablas diseñadas para gestionar los	H18-T3	Alta

		horarios de los médicos y sus cambios.		
H18-T5	Desarrollar los controladores para gestionar los cambios en los horarios	Implementar los controladores en Laravel que permitan registrar, modificar y eliminar los cambios en los horarios de atención de los médicos.	H18-T4	Alta
H18-T6	Crear las rutas para gestionar los cambios de horarios	Definir las rutas en Laravel para que los controladores puedan manejar las operaciones CRUD (crear, leer, actualizar y eliminar) para los horarios de los médicos y sus modificaciones.	H18-T5	Alta
H18-T7	Crear la interfaz de usuario para registrar y gestionar los cambios de horarios	Desarrollar el componente en React que permita a los usuarios del sistema (administradores o encargados) registrar y gestionar los cambios en los horarios de atención de los médicos.	H18-T1 H18-T6	Media
H18-T8	Conectar el frontend con el backend	Configurar la comunicación entre el frontend (React) y el backend (Laravel) para que los datos ingresados en la interfaz de usuario se reflejen en la base de datos mediante peticiones HTTP seguras.	H18-T6 H18-T7	Alta
H18-T9	Implementar validaciones en el registro de cambios de horarios	Implementar validaciones en el frontend y backend para asegurar que los cambios registrados en los horarios de los médicos sean correctos (por ejemplo, evitar solapamientos de turnos o días incorrectos).	H18-T8	Media
H18-T10	Realizar pruebas funcionales del módulo de cambios de horarios	Realizar pruebas para asegurarse de que los cambios en los horarios de atención de los médicos se registren y gestionen correctamente en el sistema.	H18-T9	Media
H18-T11	Documentar el módulo de gestión de horarios de atención	Crear la documentación técnica que describa la estructura de la base de datos, las rutas, controladores y vistas para la gestión de horarios y sus modificaciones.	Todas	Baja

3.4.11.2 Sprint Review

Tabla 55: Tabla resumen del sprint review SP11.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H18-T1	Definir los requerimientos para el registro de cambios de horarios	Completada
H18-T2	Diseñar las tablas de la base de datos para gestionar los horarios de los médicos	Completada
H18-T3	Crear los modelos en Laravel para gestionar los horarios de atención	Completada
H18-T4	Implementar las migraciones de la base de datos	Completada
H18-T5	Desarrollar los controladores para gestionar los cambios en los horarios	Completada
H18-T6	Crear las rutas para gestionar los cambios de horarios	Completada
H18-T7	Crear la interfaz de usuario para registrar y gestionar los cambios de horarios	Completada
H18-T8	Conectar el frontend con el backend	Completada
H18-T9	Implementar validaciones en el registro de cambios de horarios	Completada
H18-T10	Realizar pruebas funcionales del módulo de cambios de horarios	Completada
H18-T11	Documentar el módulo de gestión de horarios de atención	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición de requerimientos para el registro de cambios	Se identificaron y documentaron los detalles requeridos para el registro de los cambios de horarios, incluyendo médicos, días de la semana, horas y cambios específicos.	
Diseño de tablas de la base de datos	Se crearon las tablas en la base de datos para almacenar la información sobre los horarios de los médicos y las fichas disponibles.	
Creación de modelos en Laravel	Se implementaron los modelos en Laravel para gestionar los horarios de los médicos y sus modificaciones, estableciendo relaciones entre los médicos y sus horarios de atención.	
Implementación de migraciones	Se crearon las migraciones de Laravel que reflejan la estructura de las tablas diseñadas para gestionar los horarios de los médicos y sus cambios.	
Desarrollo de controladores	Se implementaron los controladores en Laravel que permiten registrar, modificar y eliminar los	

	cambios en los horarios de atención de los médicos.
Creación de rutas	Se definieron las rutas en Laravel para que los controladores puedan manejar las operaciones CRUD (crear, leer, actualizar y eliminar) para los horarios de los médicos y las fichas.
Desarrollo de la interfaz de usuario	Se desarrolló un componente en React que permite a los usuarios del sistema (administradores o encargados) registrar y gestionar los cambios en los horarios de atención de los médicos.
Conexión entre frontend y backend	Se configuró la comunicación entre el frontend (React) y el backend (Laravel) para que los datos ingresados en la interfaz de usuario se reflejen en la base de datos mediante peticiones HTTP seguras.
Implementación de validaciones	Se implementaron validaciones en el frontend y backend para asegurar que los cambios registrados en los horarios de los médicos sean correctos (por ejemplo, evitar solapamientos de turnos o días incorrectos).
Pruebas del módulo	Se realizaron pruebas funcionales para asegurarse de que los cambios en los horarios de atención de los médicos se registren y gestionen correctamente en el sistema.
Documentación del módulo	Se creó la documentación técnica que describe la estructura de la base de datos, las rutas, controladores y vistas para la gestión de horarios y sus modificaciones.

3.4.11.3 Sprint Retrospective

Tabla 56: Tabla resumen del sprint retrospective SP11.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Diseño efectivo de la base de datos: La creación de tablas específicas para gestionar los horarios de los médicos y los cambios aplicados resultó exitosa, asegurando un acceso rápido y eficiente a la información. - Interfaz de usuario intuitiva: El componente de React para registrar y gestionar cambios en los horarios fue bien recibido por los usuarios, lo que facilitó la adopción y el uso del sistema.
Desafíos enfrentados
Mejoras para futuros sprints

Ilustración 75: *Diseño de tablas para la asignación de fichas y horarios según él día.*

Fuente: Elaboración propia

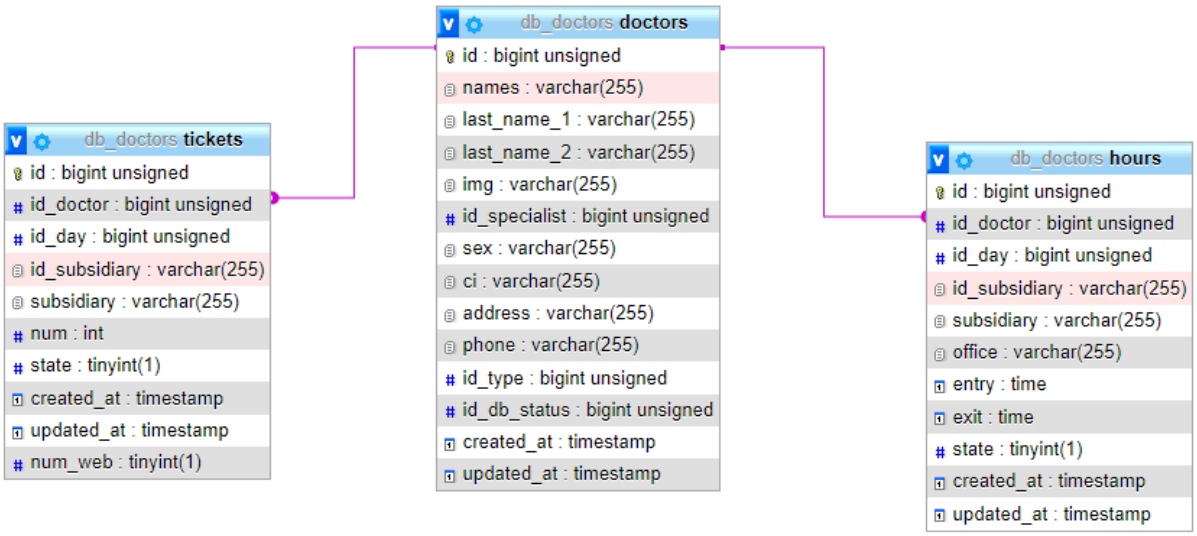
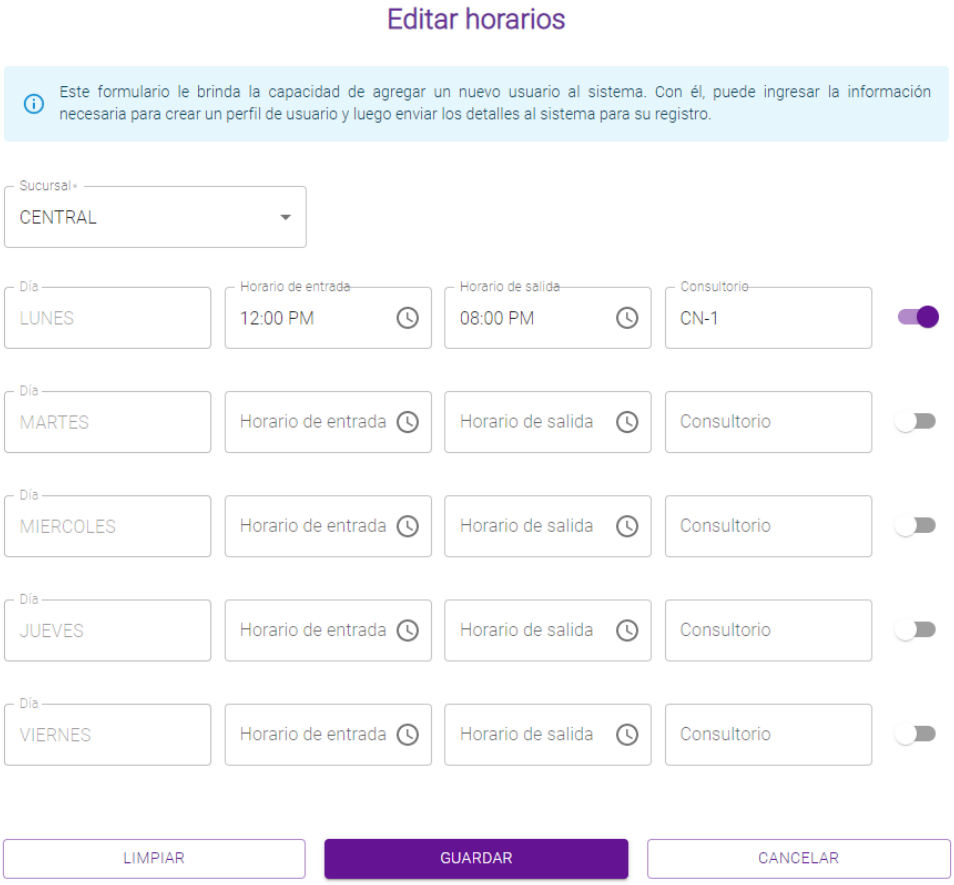


Ilustración 76: *Modal de registro de horarios según sucursal y día.*

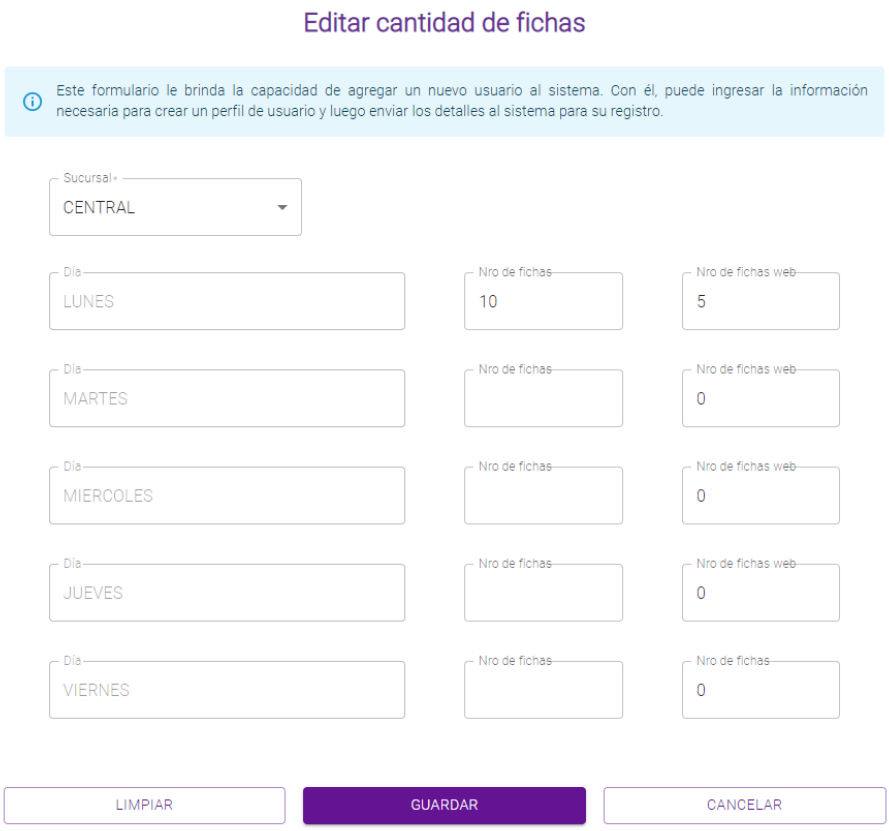
Fuente: Elaboración propia



Descripción: *Modal empleado para el registro de horarios según la sucursal, el dia y el doctor, se abre haciendo clic en la opción de horarios en la tabla de visualización de doctores.*

Ilustración 77: *Modal de registro de fichas para emitir según sucursal y día.*

Fuente: Elaboración propia



Descripción: *Modal empleado para el registro de fichas según la sucursal, el día y el doctor, se abre haciendo clic en la opción de fichas en la tabla de visualización de doctores.*

3.4.12 SP12: Implementar la visualización de turnos en TV según la ficha correspondiente

3.4.12.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar la funcionalidad que permita visualizar en pantallas de TV los turnos de los pacientes según las fichas correspondientes, asegurando que la información esté actualizada en tiempo real y se muestre de manera clara y organizada.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H19** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 57: Sprint Backlog SP12.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H19-T1	Definir el diseño y los requisitos de visualización en TV	Definir cómo se mostrará la información de los turnos en la pantalla de TV (formato, colores, tamaño de texto, etc.), y los datos que se deben incluir (número de turno, nombre del paciente, hora estimada, etc.).	Ninguno	Muy Alta
H19-T2	Desarrollar el backend para gestionar los turnos	Implementar los controladores en Laravel que gestionen la asignación, actualización y finalización de turnos para los pacientes, de acuerdo con las fichas correspondientes.	H19-T1	Alta
H19-T3	Crear las rutas para gestionar los turnos	Definir las rutas en Laravel para gestionar las operaciones CRUD (crear, leer, actualizar y eliminar) de los turnos, permitiendo la asignación y actualización de los mismos en tiempo real.	H19-T2	Alta
H19-T4	Desarrollar la interfaz para la visualización de turnos en TV	Crear una interfaz o componente frontend en React que se pueda mostrar	H19-T1	Alta

		en una pantalla de TV, y que reciba los turnos desde el backend. Esta interfaz debe actualizarse dinámicamente para reflejar los cambios en los turnos en tiempo real.		
H19-T5	Implementar WebSockets para actualizaciones en tiempo real	Configurar WebSockets o una tecnología similar para que los turnos se actualicen automáticamente en la pantalla de TV sin necesidad de recargar la página.	H19-T1	Alta
H19-T6	Conectar el frontend (visualización de turnos) con el backend	Configurar la comunicación entre el frontend (visualización en TV) y el backend para que los turnos sean enviados y actualizados en tiempo real a través de WebSockets.	H19-T5	Alta
H19-T7	Pruebas funcionales de la visualización de turnos en TV	Realizar pruebas para asegurarse de que los turnos se visualicen correctamente en la pantalla de TV, que se actualicen en tiempo real, y que la información mostrada sea clara y precisa.	H19-T6	Media
H19-T8	Documentar la funcionalidad de visualización de turnos en TV	Crear la documentación técnica que describa la implementación del módulo de visualización de turnos, incluyendo la estructura de la base de datos, las rutas y la configuración de WebSockets.	Todas	Baja

3.4.12.2 Sprint Review

Tabla 58: Tabla resumen del sprint review SP12.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H19-T1	Definir el diseño y los requisitos de visualización en TV	Completada
H19-T2	Desarrollar el backend para gestionar los turnos	Completada
H19-T3	Crear las rutas para gestionar los turnos	Completada
H19-T4	Desarrollar la interfaz para la visualización de turnos en TV	Completada

H19-T5	Implementar WebSockets para actualizaciones en tiempo real	Completada
H19-T6	Conectar el frontend (visualización de turnos) con el backend	Completada
H19-T7	Pruebas funcionales de la visualización de turnos en TV	Completada
H19-T8	Documentar la funcionalidad de visualización de turnos en TV	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición del diseño y requisitos de visualización en TV	Se definieron los requisitos para la visualización de información de turnos en la pantalla de TV, incluyendo formato, colores, tamaño de texto y datos a incluir como número de turno, nombre del paciente y hora estimada.	
Desarrollo del backend para gestionar los turnos	Se implementaron los controladores en Laravel que gestionan la asignación, actualización y finalización de turnos para los pacientes, asegurando que cumplan con las fichas correspondientes.	
Creación de rutas para gestionar los turnos	Se definieron las rutas en Laravel para permitir las operaciones CRUD (crear, leer, actualizar y eliminar) de los turnos, facilitando la asignación y actualización en tiempo real.	
Desarrollo de la interfaz para la visualización de turnos en TV	Se creó un componente frontend en React que se muestra en una pantalla de TV, recibiendo los turnos desde el backend y actualizándose dinámicamente para reflejar cambios en tiempo real.	
Implementación de WebSockets para actualizaciones en tiempo real	Se configuraron WebSockets o una tecnología similar para permitir actualizaciones automáticas de los turnos en la pantalla de TV, eliminando la necesidad de recargar la página.	
Conexión del frontend con el backend	Se configuró la comunicación entre el frontend (visualización en TV) y el backend para enviar y actualizar turnos en tiempo real a través de WebSockets.	
Pruebas funcionales de la visualización de turnos en TV	Se realizaron pruebas para verificar que los turnos se visualicen correctamente en la pantalla de TV, que se actualicen en tiempo real y que la información sea clara y precisa.	
Documentación de la funcionalidad de visualización de turnos en TV	Se elaboró la documentación técnica que describe la implementación del módulo de visualización de turnos, incluyendo la estructura de la base de datos, rutas y configuración de WebSockets.	

3.4.12.3 Sprint Retrospective

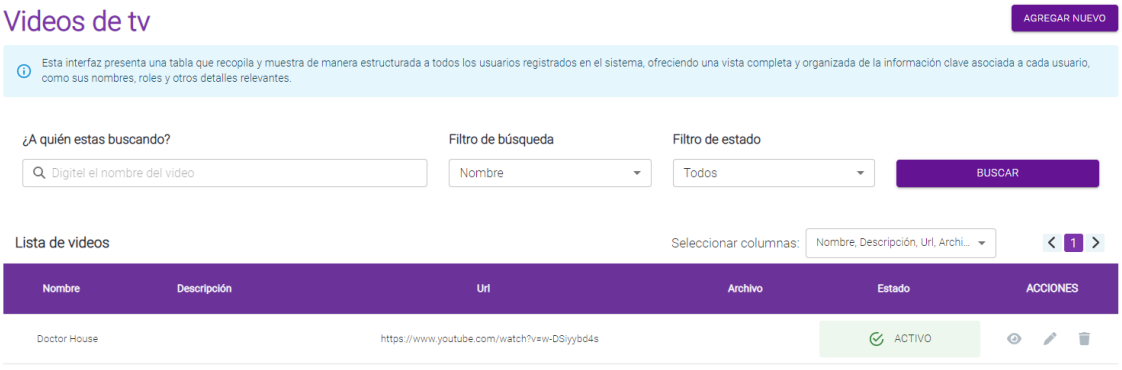
Tabla 59: Tabla resumen del sprint retrospective SP12.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Definición clara del diseño y requisitos: Se establecieron criterios específicos para la visualización de información en la pantalla de TV, lo que permitió un diseño coherente y atractivo. La inclusión de elementos como el número de turno, nombre del paciente y hora estimada fue clave para garantizar que la información fuera comprensible. - Desarrollo efectivo del backend: La implementación de controladores en Laravel para gestionar la asignación y actualización de turnos resultó eficiente. Esto permitió que los turnos se administraran de manera centralizada y se garantizaran actualizaciones en tiempo real. - Configuración exitosa de WebSockets: La implementación de WebSockets facilitó actualizaciones automáticas en la pantalla de TV sin recargar la página, lo que mejoró la experiencia del usuario al visualizar turnos en tiempo real.
Desafíos enfrentados
- Sincronización de datos en tiempo real: Aunque la tecnología de WebSockets fue configurada, hubo desafíos iniciales con la sincronización de datos, lo que llevó a algunos retrasos en la implementación de actualizaciones en tiempo real. Esto se debió a problemas en la gestión de conexiones en el servidor. - Documentación técnica incompleta: La documentación creada fue útil, pero se identificó que faltaban algunos detalles clave sobre la configuración de WebSockets y la estructura de la base de datos, lo que podría dificultar la comprensión para futuros desarrolladores.
Mejoras para futuros sprints
- Revisar la gestión de conexiones en WebSockets: Incluir pruebas más exhaustivas sobre la gestión de conexiones y desconexiones de WebSockets asegurará una experiencia más fluida y confiable al visualizar los turnos en tiempo real. - Mejorar la documentación técnica: Crear una documentación más detallada que incluya ejemplos de implementación, configuraciones específicas y mejores prácticas para el uso de WebSockets facilitará el trabajo de los futuros desarrolladores que trabajen en el sistema.

Ilustración 78: *Tabla para la administración de Videos para la TV.*

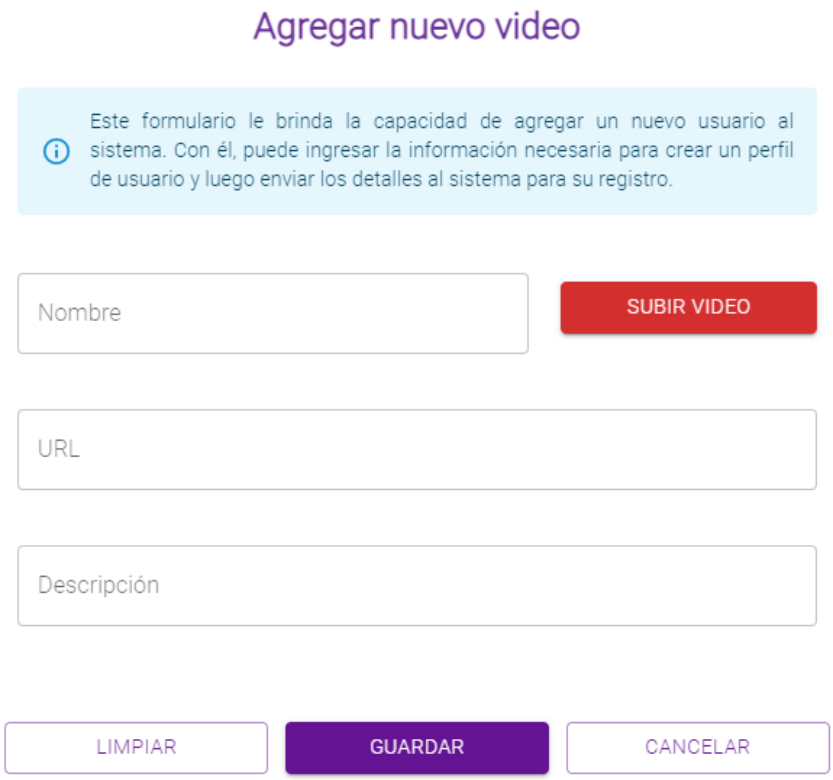
Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de los videos de TV, mostrando una serie de acciones en la parte derecha que abren modales de configuración y registro de acciones.*

Ilustración 79: *Modal para el registro de videos para la TV.*

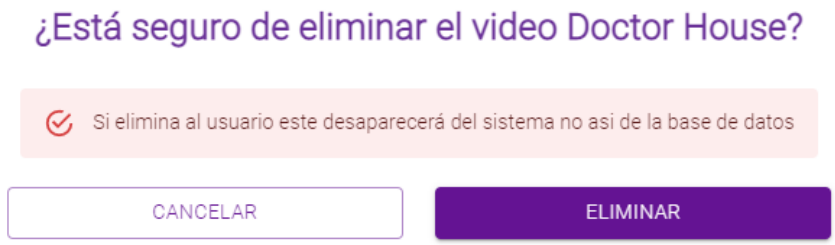
Fuente: Elaboración propia



Descripción: *Modal empleado para el registro de videos, se abre haciendo clic en la opción de agregar en la tabla de visualización de doctores.*

Ilustración 80: *Modal para eliminar videos de TV.*

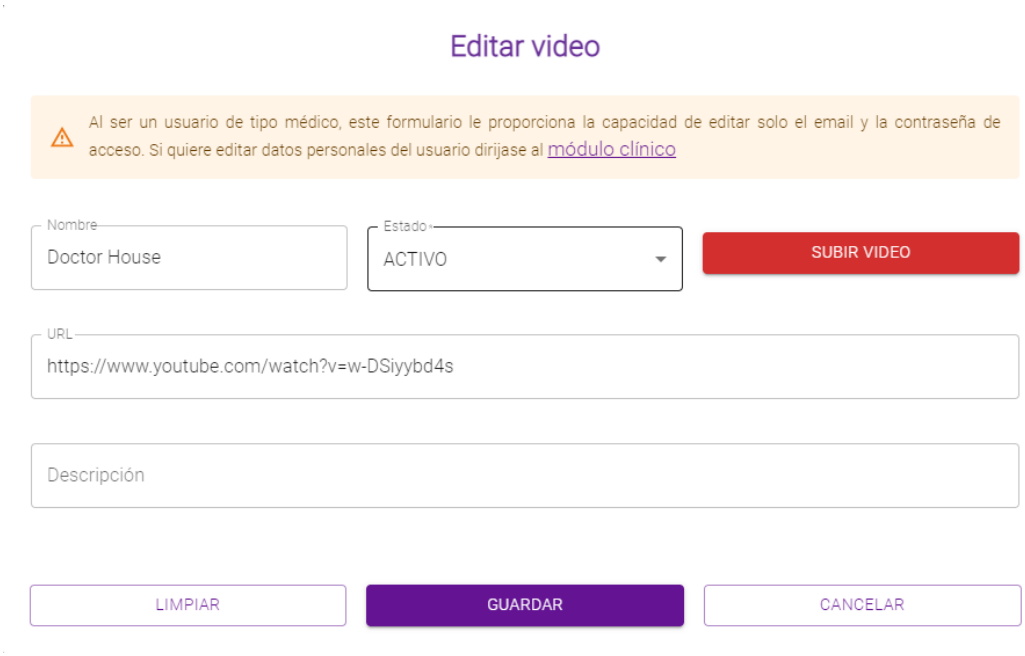
Fuente: Elaboración propia



Descripción: *Modal empleado para eliminar un video, se abre haciendo clic en la opción de eliminar en la tabla de visualización de doctores.*

Ilustración 81: *Modal para editar un video de TV.*

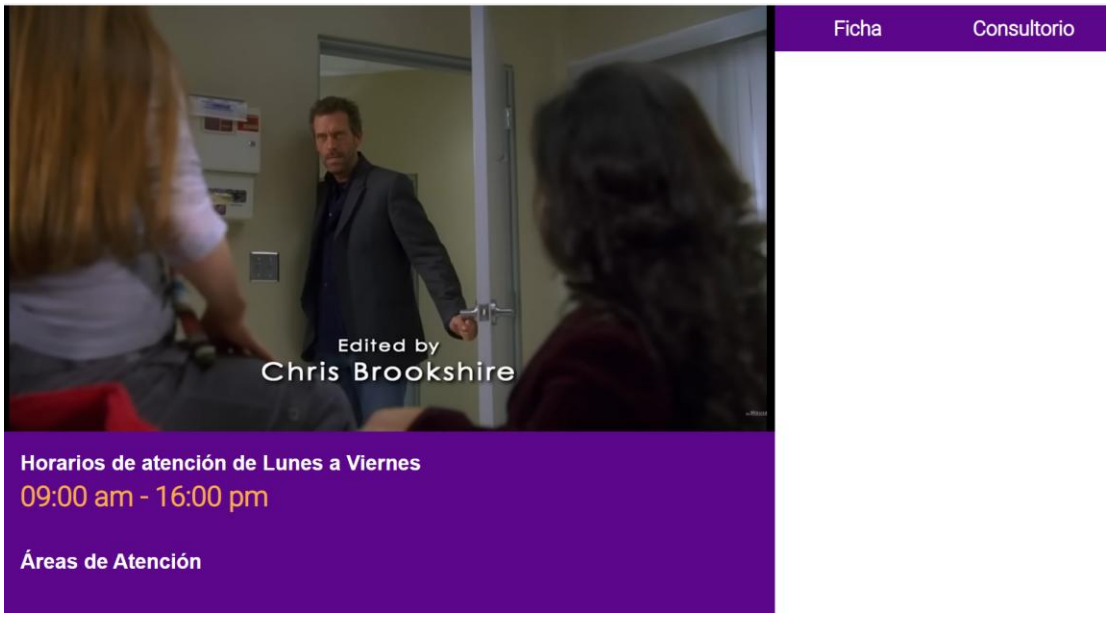
Fuente: Elaboración propia



Descripción: *Modal empleado para editar un video, se abre haciendo clic en la opción de editar en la tabla de visualización de doctores.*

Ilustración 82: *Pantalla de TV para visualización de fichas médicas.*

Fuente: Elaboración propia



Descripción: *Vista empleada para la visualización de fichas médicas las cuales se llaman desde enfermería.*

3.4.13 SP13: Desarrollar una plataforma funcional que permita a los usuarios agendar visitas en línea de manera sencilla

3.4.13.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar una plataforma web que permita a los usuarios agendar visitas en línea de manera sencilla, proporcionando una interfaz intuitiva para seleccionar fechas y horas, integrando la funcionalidad de calendario y asegurando que las visitas puedan gestionarse de manera eficiente tanto por los usuarios como por los administradores.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H21** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 60: Sprint Backlog SP13.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H21-T1	Definir los requisitos funcionales para agendar visitas	Identificar y documentar todos los requisitos del sistema, como los tipos de usuarios, opciones de horarios disponibles, y restricciones en la programación de visitas.	Ninguno	Muy Alta
H21-T2	Desarrollar la lógica del backend para gestionar la agenda	Implementar controladores en Laravel que permitan a los usuarios agendar visitas, cancelarlas o modificarlas. También permitir que los administradores gestionen la disponibilidad de horarios.	H21-T1	Alta
H21-T3	Crear las rutas para gestionar las citas	Definir las rutas en Laravel para que los controladores puedan manejar las operaciones CRUD para las citas (crear, leer, actualizar y eliminar).	H21-T2	Alta

H21-T4	Diseñar la interfaz de usuario en React	Desarrollar el frontend en React con una interfaz intuitiva que permita a los usuarios visualizar los horarios disponibles y seleccionar la fecha y hora para su visita.	H21-T1	Media
H21-T5	Conectar el frontend con el backend	Configurar la comunicación entre React y Laravel para que los usuarios puedan interactuar con la base de datos de citas y horarios disponibles, a través de peticiones HTTP seguras.	H21-T4 H21-T3	Alta
H21-T6	Validar los datos de entrada en los formularios	Implementar validaciones tanto en el frontend como en el backend para asegurar que los datos ingresados por los usuarios, como fechas y horas, sean correctos y que los horarios seleccionados estén disponibles.	H21-T4	Media
H21-T7	Implementar autenticación de asegurados	Configurar el sistema de autenticación en Laravel para asegurar que los asegurados registrados puedan acceder al sistema para agendar visitas, y los administradores puedan gestionar el sistema.	H21-T6	Muy Alta
H21-T8	Pruebas del módulo de agenda de visitas	Realizar pruebas funcionales del sistema de citas para asegurar que los usuarios puedan agendar, modificar y cancelar visitas, y que los administradores puedan gestionar la agenda.	H21-T7	Media
H21-T9	Documentar el módulo de agendamiento de citas	Crear la documentación técnica que detalle cómo funciona el sistema de agendamiento, incluyendo el modelo de datos, rutas, controladores, y la interfaz de usuario.	Todas	Baja

3.4.13.2 Sprint Review

Tabla 61: Tabla resumen del sprint review SP13.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H21-T1	Definir los requisitos funcionales para agendar visitas	Completada
H21-T2	Desarrollar la lógica del backend para gestionar la agenda	Completada
H21-T3	Crear las rutas para gestionar las citas	Completada
H21-T4	Diseñar la interfaz de usuario en React	Completada
H21-T5	Conectar el frontend con el backend	Completada
H21-T6	Validar los datos de entrada en los formularios	Completada
H21-T7	Implementar autenticación de asegurados	Completada
H21-T8	Pruebas del módulo de agenda de visitas	Completada
H21-T9	Documentar el módulo de agendamiento de citas	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición de requisitos funcionales para agendar visitas	Se identificaron y documentaron los requisitos del sistema, incluyendo tipos de usuarios, opciones de horarios disponibles y restricciones en la programación de visitas.	
Desarrollo de la lógica del backend para gestionar la agenda	Se implementaron controladores en Laravel que permiten a los usuarios agendar, cancelar o modificar visitas, además de permitir a los administradores gestionar la disponibilidad de horarios.	
Creación de rutas para gestionar las citas	Se definieron las rutas en Laravel para que los controladores puedan manejar las operaciones CRUD (crear, leer, actualizar y eliminar) para las citas.	
Diseño de la interfaz de usuario en React	Se desarrolló un frontend en React con una interfaz intuitiva que permite a los usuarios visualizar los horarios disponibles y seleccionar la fecha y hora para su visita.	
Conexión del frontend con el backend	Se configuró la comunicación entre React y Laravel para que los usuarios puedan interactuar con la base de datos de citas y horarios disponibles a través de peticiones HTTP seguras.	

Validación de datos de entrada en los formularios	Se implementaron validaciones tanto en el frontend como en el backend para asegurar que los datos ingresados por los usuarios (fechas y horas) sean correctos y que los horarios seleccionados estén disponibles.
Implementación de autenticación de asegurados	Se configuró el sistema de autenticación en Laravel para asegurar que los asegurados registrados puedan acceder al sistema para agendar visitas y que los administradores puedan gestionar el sistema.
Pruebas del módulo de agenda de visitas	Se realizaron pruebas funcionales del sistema de citas para asegurar que los usuarios puedan agendar, modificar y cancelar visitas, y que los administradores puedan gestionar la agenda.
Documentación del módulo de agendamiento de citas	Se creó la documentación técnica que detalla cómo funciona el sistema de agendamiento, incluyendo el modelo de datos, rutas, controladores y la interfaz de usuario.

3.4.13.3 Sprint Retrospective

Tabla 62: Tabla resumen del sprint retrospective SP13.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
- Desarrollo eficiente de la lógica del backend: La implementación de controladores en Laravel para gestionar la agenda funcionó sin problemas, permitiendo que los usuarios agendaran, cancelaran y modificaran visitas. Además, los administradores pudieron gestionar fácilmente la disponibilidad de horarios. - Interfaz intuitiva en React: La creación de un frontend en React resultó en una interfaz amigable que permitió a los usuarios visualizar horarios disponibles de manera clara y seleccionar fechas y horas para sus visitas sin complicaciones.
Desafíos enfrentados
- Integración entre frontend y backend: Aunque se logró la conexión, hubo algunos problemas menores en la configuración de las peticiones HTTP seguras, lo que ocasionó ciertos retrasos en las pruebas iniciales. Se necesitó un tiempo adicional para asegurar que las respuestas fueran correctas y seguras.
Mejoras para futuros sprints
- Revisar y mejorar la integración API: Se sugiere realizar una revisión exhaustiva de la integración entre el frontend y backend, asegurando que las peticiones y respuestas sean coherentes y seguras, y facilitar la depuración de errores.

Ilustración 83: *Ingreso al sitio web para agnación de fichas para los beneficiarios.*

Fuente: Elaboración propia

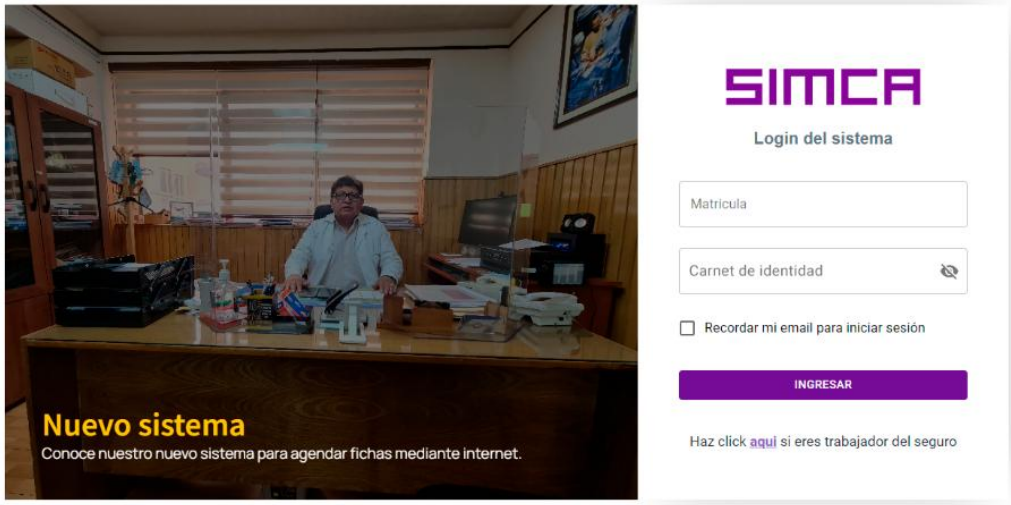
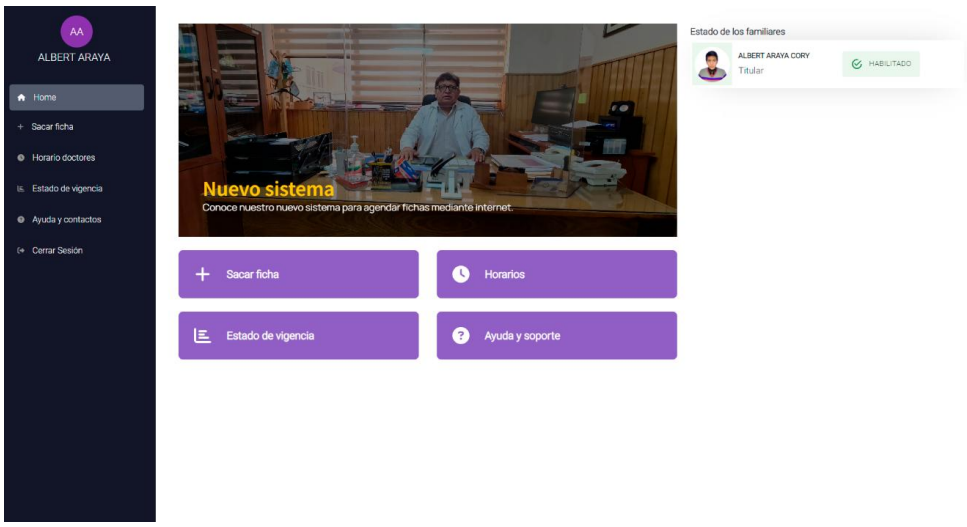


Ilustración 84: *Dashboard de los beneficiarios para agendar una ficha.*

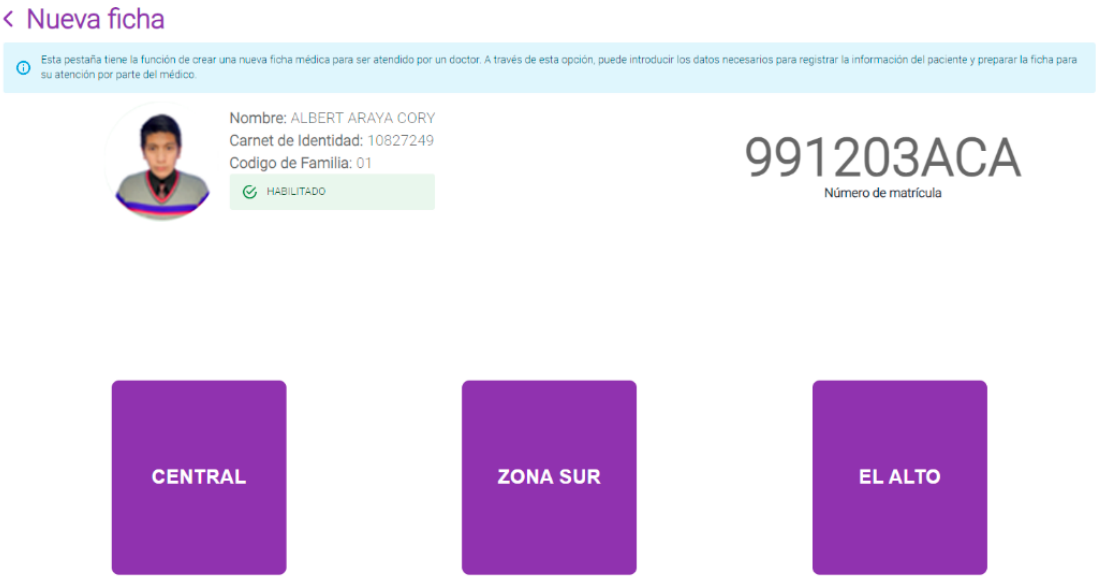
Fuente: Elaboración propia



Descripción: *Vista principal de los beneficiarios al iniciar sesión en el sistema mediante el sitio web*

Ilustración 85: *Vista para elegir la sucursal donde se registrará la ficha.*

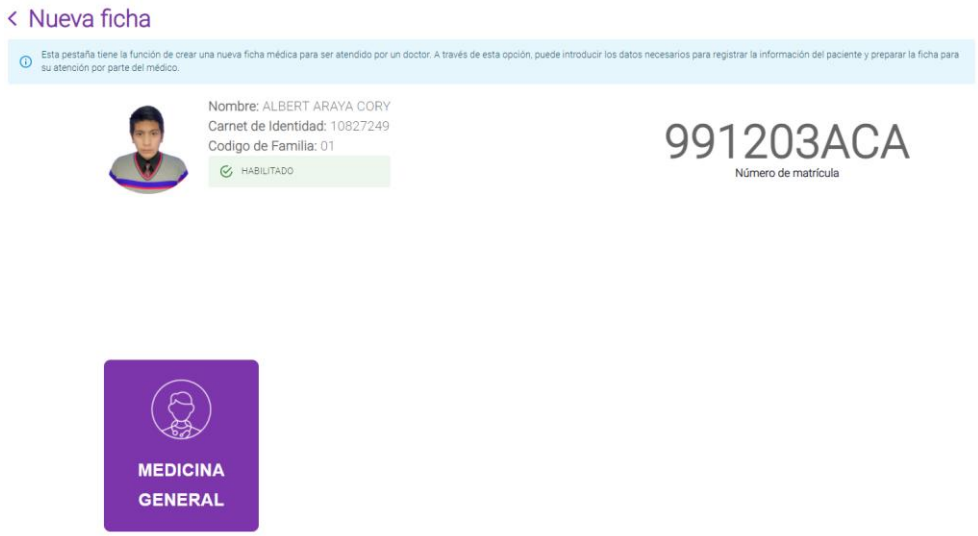
Fuente: Elaboración propia



Descripción: *Vista empleada para la elegir una sucursal para registrar una ficha*

Ilustración 86: *Ventana de asignación de especialidad.*

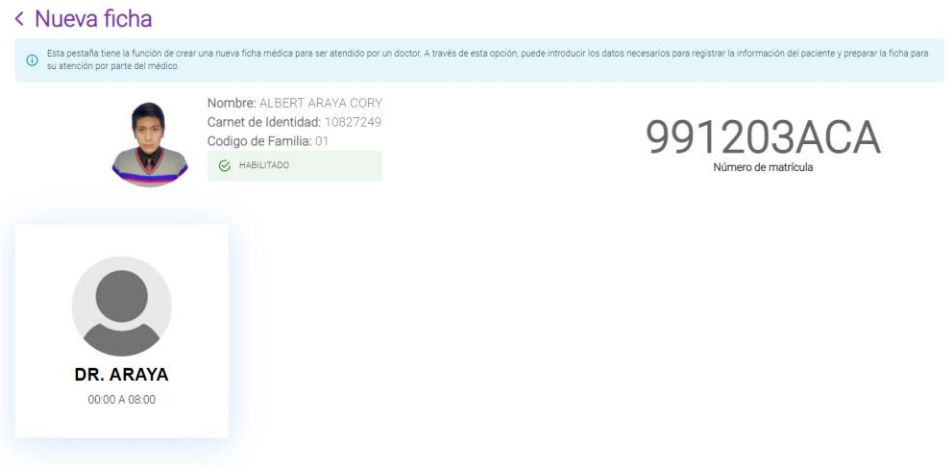
Fuente: Elaboración propia



Descripción: *Vista empleada para la elegir una especialidad la sucursal para registrar una ficha*

Ilustración 87: *Ventana de asignación de doctores.*

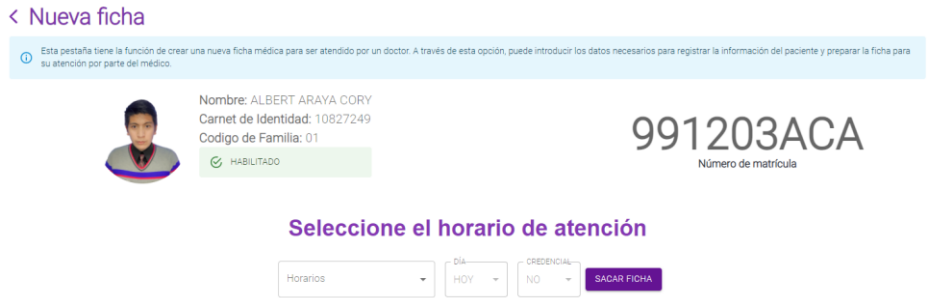
Fuente: Elaboración propia



Descripción: *Vista empleada para la elegir un doctor según la especialidad y la sucursal para registrar una ficha*

Ilustración 88: *Ventana de asignación de horarios.*

Fuente: Elaboración propia



Descripción: *Vista empleada para la elegir un horario según el doctor, la especialidad y la sucursal para registrar una ficha.*

***Ilustración 89**: Diagrama de procesos para generar una ficha como paciente*

Fuente: Elaboración propia

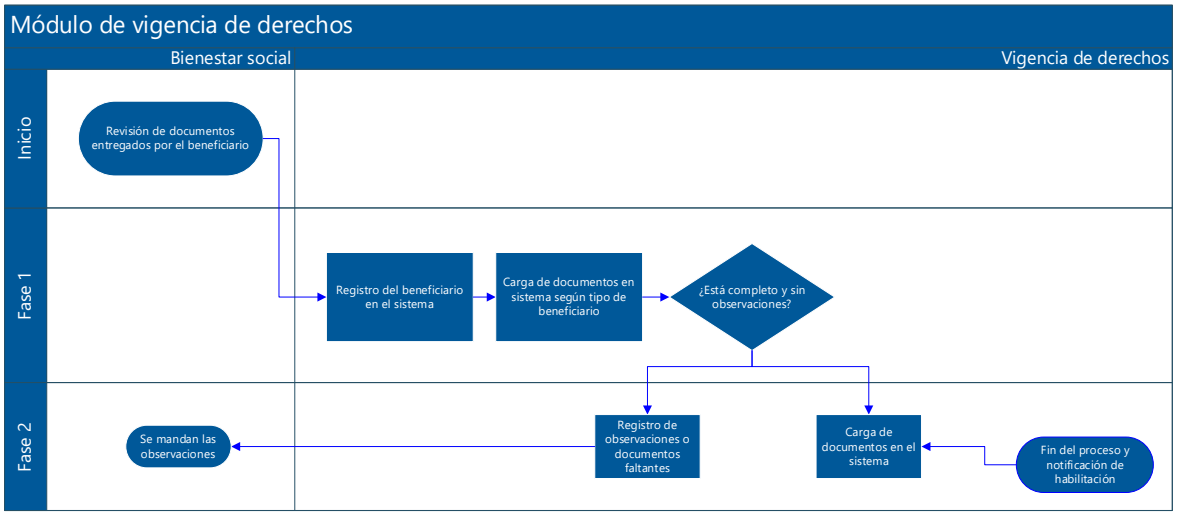
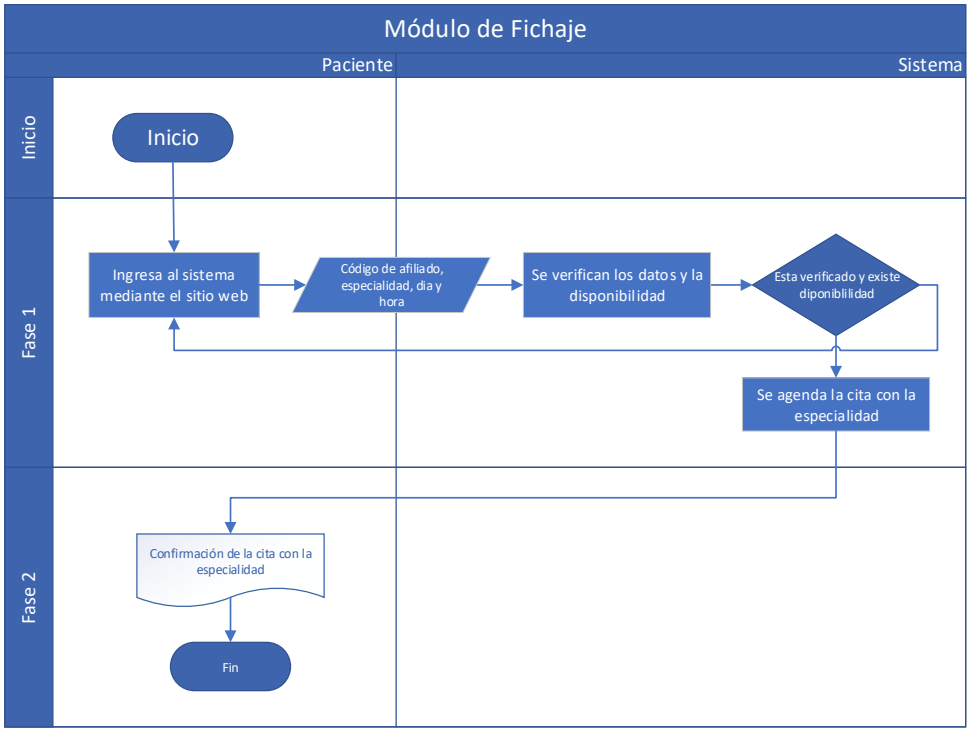


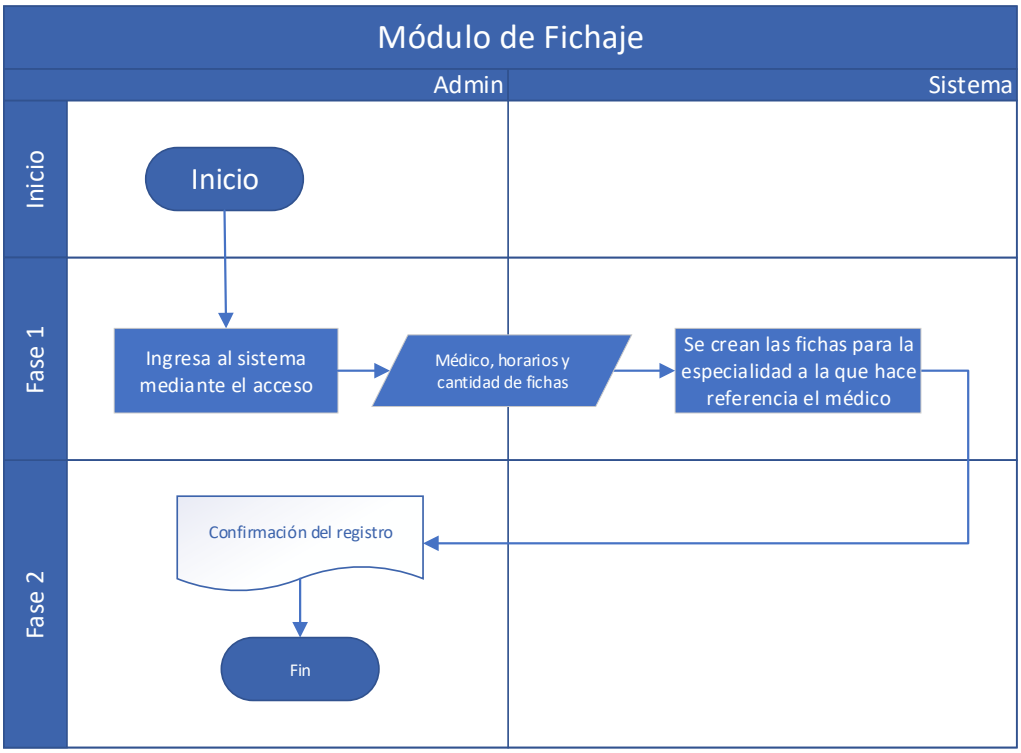
***Ilustración 90**: Diagrama de procesos para generar una ficha como paciente*

Fuente: Elaboración propia



***C16**: Diagrama de procesos para generar una ficha como administrador*

Fuente: Elaboración propia



3.4.14 SP14: Implementar una función de detección y visualización de alertas para documentos caducados

3.4.14.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar una funcionalidad que permita detectar y visualizar alertas para documentos caducados en el sistema, asegurando que los usuarios sean notificados de manera efectiva sobre la necesidad de renovación de documentos, mejorando así la gestión administrativa y de cumplimiento.

b. Sprint backlog

En base a que el sprint involucra a las historias de usuario **H8 y H15** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia,

descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 63: Sprint Backlog SP14.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H8-T1	Definir los criterios para la detección de documentos caducados	Establecer los criterios de caducidad para los diferentes tipos de documentos (por ejemplo, fecha de caducidad, periodo de gracia, etc.).	Ninguno	Muy Alta
H8-T2	Crear un servicio para la detección de documentos caducados	Desarrollar un servicio en Laravel que realice la detección automática de documentos caducados basándose en los criterios definidos.	H8-T1	Muy Alta
H8-T3	Implementar un cron job para la detección periódica	Configurar un cron job que ejecute el servicio de detección de documentos caducados de forma regular (diaria, semanal, etc.) para mantener la información actualizada.	H8-T2	Alta
H8-T4	Crear un modelo de datos para almacenar las alertas de documentos caducados	Diseñar e implementar un modelo en Laravel para almacenar las alertas generadas por la detección de documentos caducados.	H8-T2	Alta
H8-T5	Desarrollar un controlador para gestionar las alertas	Implementar un controlador en Laravel que gestione las alertas de documentos caducados, permitiendo crear, leer y eliminar alertas según sea necesario.	H8-T4	Alta
H8-T6	Crear las rutas para gestionar las alertas de documentos caducados	Definir las rutas en Laravel que permitan al controlador gestionar las alertas de documentos caducados.	H8-T5	Alta
H15-T1	Desarrollar la interfaz de usuario para visualizar alertas	Crear un componente en React que muestre las alertas de documentos caducados a los usuarios, incluyendo detalles como tipo de documento, fecha de caducidad y acciones recomendadas.	H8-T6	Media
H15-T2	Implementar lógica para marcar documentos como renovados	Desarrollar funcionalidad que permita a los usuarios marcar documentos como renovados en la interfaz de usuario, actualizando así el estado en la base de datos.	H15-T1	Alta

H8-T7 H15-T3	Conectar el frontend con el backend para la gestión de alertas	Configurar la comunicación entre React y Laravel para enviar y recibir datos sobre las alertas de documentos caducados, incluyendo la renovación.	H15-T2 H8-T6	Alta
H15-T4	Validar la entrada de datos y las acciones en la interfaz de usuario	Implementar validaciones para asegurar que las acciones realizadas en la interfaz (como la renovación de documentos) sean correctas y seguras.	H8-T7 H15-T3	Media
H8-T8	Pruebas funcionales del módulo de detección y visualización de alertas	Realizar pruebas para asegurar que el sistema detecte correctamente los documentos caducados y que las alertas se visualicen adecuadamente.	H15-T4	Media
H8-T9 H15-T5	Documentar el módulo de detección y visualización de alertas	Crear la documentación técnica que describa la estructura del módulo, incluyendo el servicio de detección, controlador, modelos de datos y la interfaz de usuario.	Todas	Baja

3.4.14.2 Sprint Review

Tabla 64: Tabla resumen del sprint review SP14.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H8-T1	Definir los criterios para la detección de documentos caducados	Completada
H8-T2	Crear un servicio para la detección de documentos caducados	Completada
H8-T3	Implementar un cron job para la detección periódica	Completada
H8-T4	Crear un modelo de datos para almacenar las alertas de documentos caducados	Completada
H8-T5	Desarrollar un controlador para gestionar las alertas	Completada
H8-T6	Crear las rutas para gestionar las alertas de documentos caducados	Completada
H15-T1	Desarrollar la interfaz de usuario para visualizar alertas	Completada
H15-T2	Implementar lógica para marcar documentos como renovados	Completada
H15-T3 H8-T7	Conectar el frontend con el backend para la gestión de alertas	Completada

H15-T4	Validar la entrada de datos y las acciones en la interfaz de usuario	Completada
H8-T8	Pruebas funcionales del módulo de detección y visualización de alertas	Completada
H8-T9 H15-T5	Documentar el módulo de detección y visualización de alertas	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición de criterios para la detección de documentos caducados	Se establecieron los criterios de caducidad para los diferentes tipos de documentos, incluyendo la fecha de caducidad y el período de gracia.	
Creación de un servicio para la detección de documentos caducados	Se desarrolló un servicio en Laravel que realiza la detección automática de documentos caducados basándose en los criterios definidos.	
Implementación de un cron job para la detección periódica	Se configuró un cron job que ejecuta el servicio de detección de documentos caducados de forma regular (diaria, semanal, etc.) para mantener la información actualizada.	
Creación de un modelo de datos para almacenar las alertas	Se diseñó e implementó un modelo en Laravel para almacenar las alertas generadas por la detección de documentos caducados.	
Desarrollo de un controlador para gestionar las alertas	Se implementó un controlador en Laravel que gestiona las alertas de documentos caducados, permitiendo crear, leer y eliminar alertas según sea necesario.	
Creación de rutas para gestionar las alertas de documentos caducados	Se definieron las rutas en Laravel que permiten al controlador gestionar las alertas de documentos caducados.	
Desarrollo de la interfaz de usuario para visualizar alertas	Se creó un componente en React que muestra las alertas de documentos caducados a los usuarios, incluyendo detalles como tipo de documento, fecha de caducidad y acciones recomendadas.	
Implementación de lógica para marcar documentos como renovados	Se desarrolló funcionalidad que permite a los usuarios marcar documentos como renovados en la interfaz de usuario, actualizando así el estado en la base de datos.	
Conexión del frontend con el backend para la gestión de alertas	Se configuró la comunicación entre React y Laravel para enviar y recibir datos sobre las alertas de documentos caducados, incluyendo la renovación.	
Validación de la entrada de datos y las acciones en la interfaz de usuario	Se implementaron validaciones para asegurar que las acciones realizadas en la interfaz (como la renovación de documentos) sean correctas y seguras.	
Pruebas funcionales del módulo de detección y visualización de alertas	Se realizaron pruebas para asegurar que el sistema detecte correctamente los documentos caducados y que las alertas se visualicen adecuadamente.	
Documentación del módulo de detección y visualización de alertas	Se creó la documentación técnica que describe la estructura del módulo, incluyendo	

	el servicio de detección, controlador, modelos de datos y la interfaz de usuario.

3.4.14.3 Sprint Retrospective

Tabla 65: Tabla resumen del sprint retrospective SP14.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
Definición de criterios de caducidad: Se establecieron criterios claros para la detección de documentos caducados, incluyendo fechas de caducidad y periodos de gracia. Esto proporcionó una base sólida para el desarrollo del servicio y facilitó la detección precisa de documentos. Desarrollo del servicio en Laravel: La implementación del servicio para la detección automática de documentos caducados fue exitosa. Se logró que el servicio funcionara de acuerdo a los criterios establecidos, permitiendo una identificación rápida y eficiente de documentos caducados. Interfaz de usuario en React: Se creó un componente funcional en React que muestra las alertas de documentos caducados. Este componente incluye detalles como el tipo de documento, la fecha de caducidad y acciones recomendadas, proporcionando a los usuarios una visualización clara y útil.
Desafíos enfrentados
Configuración del cron job: Hubo dificultades iniciales para configurar el cron job que ejecutaría el servicio de detección periódica. Algunos problemas en la programación llevaron a que no se ejecutara como se esperaba, lo que retrasó la actualización de las alertas. Implementación de la lógica para marcar documentos como renovados: La funcionalidad para permitir que los usuarios marcaran documentos como renovados presentó errores que provocaron problemas en la actualización del estado en la base de datos. Se requirió tiempo adicional para solucionar estos inconvenientes. Validación de la entrada de datos: Durante la implementación de las validaciones en la interfaz de usuario, se encontraron casos en los que los usuarios podían realizar acciones no válidas, lo que generó errores en el sistema. Esto evidenció la necesidad de reforzar la lógica de validación.
Mejoras para futuros sprints
Revisión y prueba del cron job: Es esencial realizar una revisión exhaustiva de la configuración del cron job, asegurando que esté correctamente programado para ejecutar el servicio de detección en los intervalos deseados. Realizar pruebas de la configuración podría evitar problemas futuros. Mejora en la funcionalidad de renovación: Se debe realizar una revisión detallada de la lógica de marcado de documentos como renovados para asegurar que actualice correctamente el estado en la base de datos. Considerar la implementación de mensajes de retroalimentación para los usuarios después de realizar la acción podría ser útil. Refuerzo de validaciones: Es fundamental fortalecer las validaciones en la interfaz de usuario para asegurar que todas las acciones sean correctas y seguras. Esto podría incluir el uso de mensajes claros que informen a los usuarios sobre errores o acciones no permitidas.

Ilustración 91: *Tabla de beneficiarios habilitados antes de la ejecución del job.*

Fuente: Elaboración propia

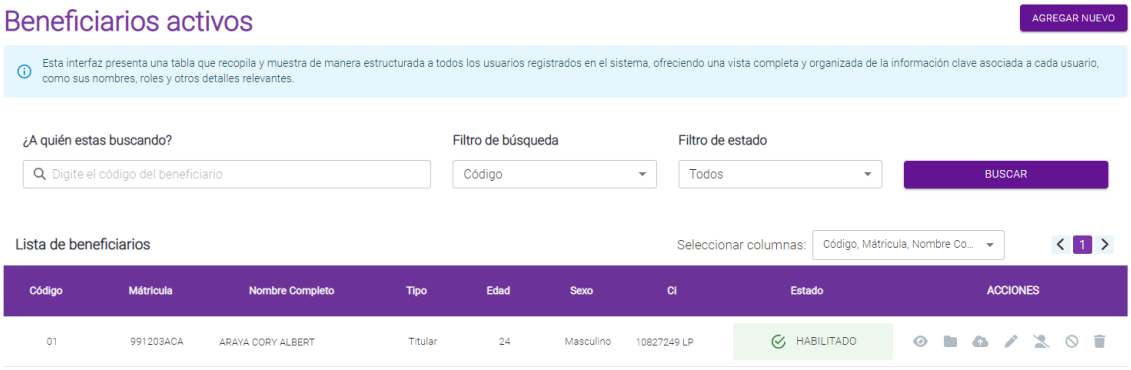


Ilustración 92: *Modal después de la ejecución del job.*

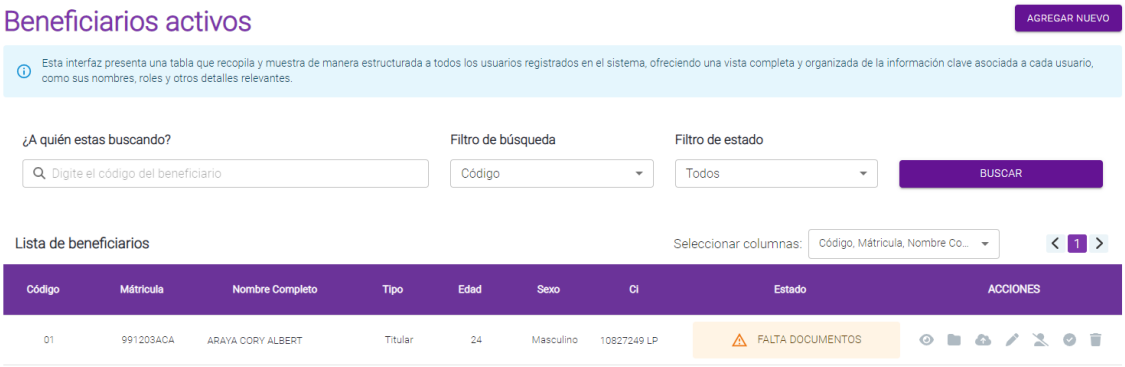
Fuente: Elaboración propia



Descripción: *Ejemplo de cambio de estado en la documentación después de la ejecución del cronjob*

Ilustración 93: *Tabla de beneficiarios después de la ejecución del job.*

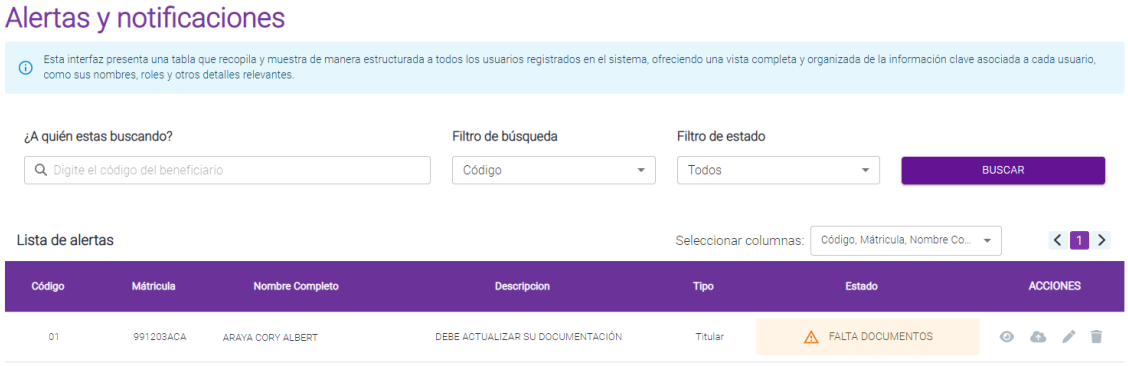
Fuente: Elaboración propia



Descripción: *Cambio presente en la tabla después de la ejecución del cronjob, puesto que el beneficiario ya no cuenta con los documentos al día por que uno de sus documentos caduco*

Ilustración 94: *Tabla de alertas generadas después de la ejecución del job.*

Fuente: Elaboración propia



Descripción: *Visualización de las alertas, una vez ejecutado el cronjob se genera una alerta la cual indica que beneficiario presenta falta de documentos u otras alertas*

3.4.15 SP15: Implementar la generación de informes en el formato establecido y permitir la exportación de datos a Excel

3.4.15.1 Sprint Planning

a. **Objetivo del Sprint:** Desarrollar la funcionalidad que permita la generación de informes en el formato establecido y la exportación de datos a Excel, asegurando que los usuarios puedan obtener información estructurada y accesible de manera eficiente para su análisis y gestión.

b. Sprint backlog

En base a que el sprint involucra a la historia de usuario **H1** (ver Tabla 24). Surgen un conjunto de tareas a realizar en este sprint, las cuales cuentan con una referencia, descripción, prerrequisitos, prioridad (definida según los criterios de prioridad redactados en el punto 3.3.5) y tiempo de desarrollo. Dichas tareas están descritas en la siguiente tabla.

Tabla 66: Sprint Backlog SP15.

Fuente: Elaboración propia.

Ref.	Tarea	Descripción	Pre Requisito	Prioridad
H1-T1	Definir el formato del informe	Especificar el formato de los informes que se generarán, incluyendo los campos y la estructura necesaria para que la información sea clara y fácil de entender.	Ninguno	Muy Alta
H1-T2	Crear una función en React para la generación de informes	Implementar una función en React que procese los datos y genere los informes en el formato establecido.	H1-T1	Muy Alta
H1-T3	Crear la interfaz de usuario para la generación de informes	Desarrollar un componente en React que permita a los usuarios solicitar la generación de informes y la exportación a Excel.	H1-T2	Alta
H1-T4	Implementar filtros y opciones de búsqueda	Agregar funcionalidades en la interfaz de usuario que permitan a los usuarios filtrar y buscar los datos que desean incluir en los informes antes de la generación.	H1-T2	Media
H1-T5	Validar la generación y exportación de informes	Implementar pruebas para asegurar que los informes generados y los archivos	H1-T4	Media

		Excel exportados contienen la información correcta y están en el formato adecuado.		
H1-T6	Pruebas funcionales del módulo de informes	Realizar pruebas funcionales para verificar que los usuarios puedan generar informes y exportar datos a Excel sin errores y que toda la funcionalidad esté operativa.	H1-T5	Media
H1-T7	Documentar el módulo de generación de informes	Crear la documentación técnica que detalle el funcionamiento del módulo de generación de informes, incluyendo ejemplos de formatos y descripciones de la API.	Todas	Baja

3.4.15.2 Sprint Review

Tabla 67: Tabla resumen del sprint review SP15.

Fuente: Elaboración propia.

SPRINT REVIEW		
Estado de tareas al finalizar el Sprint		
Ref.	**Tarea**	**Estado**
H1-T1	Definir el formato del informe	Completada
H1-T2	Crear una función en React para la generación de informes	Completada
H1-T3	Crear la interfaz de usuario para la generación de informes	Completada
H1-T4	Implementar filtros y opciones de búsqueda	Completada
H1-T5	Validar la generación y exportación de informes	Completada
H1-T6	Pruebas funcionales del módulo de informes	Completada
H1-T7	Documentar el módulo de generación de informes	Completada
Observaciones: Ninguna		
Problemas encontrados: Ninguno		
Resultados		
Definición del formato del informe	Se especificó el formato de los informes que se generarán, incluyendo los campos y la estructura necesaria para que la información sea clara y fácil de entender.	

Creación de una función en React para la generación de informes	Se implementó una función en React que procesa los datos y genera los informes en el formato establecido.
Creación de la interfaz de usuario para la generación de informes	Se desarrolló un componente en React que permite a los usuarios solicitar la generación de informes y la exportación a Excel.
Implementación de filtros y opciones de búsqueda	Se agregaron funcionalidades en la interfaz de usuario que permiten a los usuarios filtrar y buscar los datos que desean incluir en los informes antes de la generación.
Validación de la generación y exportación de informes	Se implementaron pruebas para asegurar que los informes generados y los archivos Excel exportados contienen la información correcta y están en el formato adecuado.
Pruebas funcionales del módulo de informes	Se realizaron pruebas funcionales para verificar que los usuarios puedan generar informes y exportar datos a Excel sin errores y que toda la funcionalidad esté operativa.
Documentación del módulo de generación de informes	Se creó la documentación técnica que detalla el funcionamiento del módulo de generación de informes, incluyendo ejemplos de formatos y descripciones de la API.

3.4.15.3 Sprint Retrospective

Tabla 68: Tabla resumen del sprint retrospective SP15.

Fuente: Elaboración propia.

SPRINT RETROSPECTIVE
Aspectos que funcionaron bien
Definición del formato del informe: Se logró especificar de manera clara el formato de los informes, incluyendo los campos y la estructura necesaria para que la información fuera comprensible. Esto facilitó el desarrollo de las siguientes tareas al tener un marco claro de referencia. Función en React para la generación de informes: La implementación de la función en React para procesar datos y generar informes fue exitosa. Se cumplió con el formato establecido y se logró que la función manejara adecuadamente diferentes tipos de datos. Interfaz de usuario para la generación de informes: Se desarrolló un componente de interfaz de usuario intuitivo que permite a los usuarios solicitar la generación de informes y la exportación a Excel. Los usuarios han encontrado la interfaz fácil de usar y funcional.
Desafíos enfrentados
Implementación de filtros y opciones de búsqueda: La adición de funcionalidades para filtrar y buscar datos en la interfaz resultó más complicada de lo anticipado. Algunos usuarios encontraron confusiones al utilizar los filtros, lo que sugiere que es necesario mejorar la usabilidad de esta funcionalidad. Validación de la generación y exportación de informes: Durante las pruebas, se detectaron algunos errores en la validación de los informes generados y en los archivos Excel exportados, que no siempre contenían la información correcta. Se requirió más tiempo para resolver estos inconvenientes. Pruebas funcionales del módulo: Aunque se realizaron pruebas funcionales, algunas situaciones no fueron cubiertas adecuadamente, lo que llevó a que se descubrieran errores en la generación

de informes durante el uso en producción. Se debe ampliar el enfoque de pruebas para incluir más casos de uso.
Mejoras para futuros sprints
Optimización de los filtros y opciones de búsqueda: Se sugiere realizar una revisión del diseño y la lógica detrás de los filtros y opciones de búsqueda para mejorar su usabilidad. Considerar agregar tutoriales o guías rápidas en la interfaz podría ayudar a los usuarios a familiarizarse con esta funcionalidad. Refuerzo de las pruebas de validación: Es fundamental establecer un proceso más riguroso para validar los informes generados y asegurarse de que los archivos Excel contengan la información correcta. Esto podría incluir la implementación de pruebas unitarias más exhaustivas. Documentación detallada del módulo: Al documentar el módulo de generación de informes, se debe incluir ejemplos claros y descripciones de la API utilizada. Esto facilitará la comprensión del sistema tanto para los desarrolladores actuales como para futuros colaboradores.

Ilustración 95: *Ejemplo de reporte individual de beneficiario para vigencia de derechos.*

Fuente: Elaboración propia

SEGURO MÉDICO DELEGADO DE COTEL - LISTADO DE DOCUMENTOS DEL TITULAR

DATOS PERSONALES



NOMBBRE: ALBERT ARAYA CORY
ZONA: MIRAFLORES
CÉDULA DE IDENTIDAD: 10827249 LP
CALLE: ENTRE 8 Y 9
FECHA NACIMIENTO: 1999-12-03
SEXO: MASCULINO
TELÉFONO:
GRUPO SANGUINEO:

DATOS AVC

CÓDIGO: 01
FECHA AVC: 2024-10-10
MATRICULA: 991203ACA
FECHA EXP AVC:
DIRECCIÓN:

REQUISITOS

AVC 04	CUMPLE
Fotocopia carnet de identidad	CUMPLE
Certificado de nacimiento	CUMPLE
Memorandum de designación o contrato	CUMPLE
Examen preocupacional	CUMPLE
Formulario 101	CUMPLE
Fotografias	CUMPLE

OBSERVACIONES

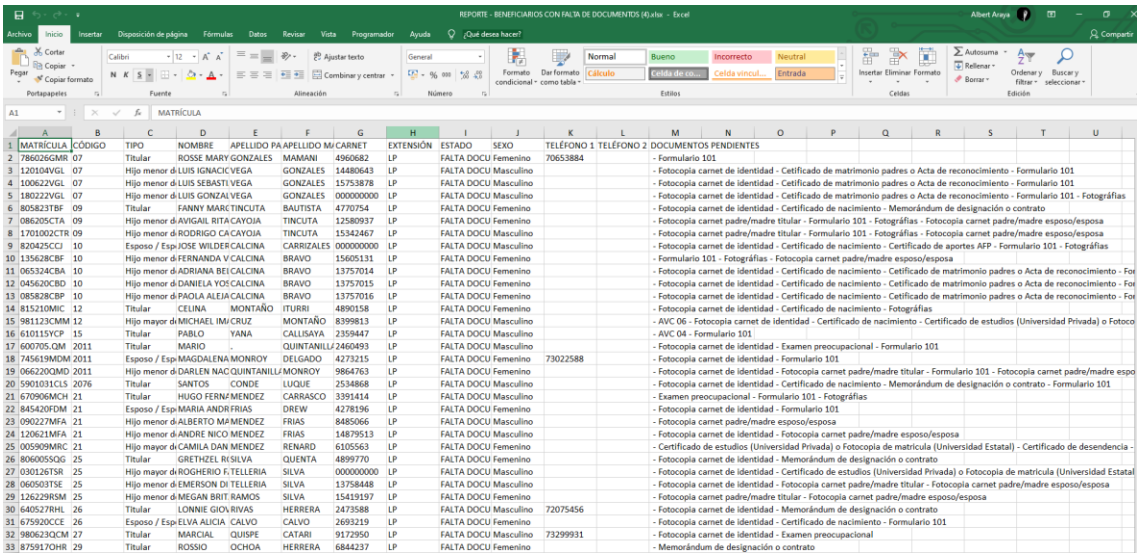
Ilustración 96: *Ejemplo de vista para obtener reportes para vigencia de derechos.*

Fuente: Elaboración propia



Ilustración 97: *Ejemplo de reporte de falta de documentos de beneficiarios para vigencia de derechos.*

Fuente: Elaboración propia



MATRÍCULA	CÓDIGO	TIPO	NOMBRE	APELLIDO PA	APELLIDO MA	CARNET	EXTENSIÓN	ESTADO	SEXO	TELÉFONO 1	TELÉFONO 2	DOCUMENTOS PENDIENTES
786026GMR	07	Titular	ROSSE MARY	GONZALES	MAMANI	4960682	LP	FALTA DOCU	Femenino	70653884		- Formulario 101
120104VGL	07	Hijo menor d	LUIS IGNACIO	VEGA	GONZALES	14480643	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Certificado de matrimonio padres o Acta de reconocimiento - Formulario 101
100622VGL	07	Hijo menor d	LUIS SEBASTI	VEGA	GONZALES	15753878	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Certificado de matrimonio padres o Acta de reconocimiento - Formulario 101
180222VGL	07	Hijo menor d	LUIS GONZAL	VEGA	GONZALES	000000000	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Certificado de matrimonio padres o Acta de reconocimiento - Formulario 101 - Fotográfias
805823TBF	09	Titular	FANNY MARC	TINCUTA	BAUTISTA	4770754	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Certificado de nacimiento - Memorándum de designación o contrato
086205CTA	09	Hijo menor d	AVIGAIL RITA	CAYOJA	TINCUTA	12580937	LP	FALTA DOCU	Femenino			- Fotocopia carnet padre/madre titular - Formulario 101 - Fotográfias - Fotocopia carnet padre/madre esposo/esposa
1701002CTR	09	Hijo menor d	RODRIGO CA	CAYOJA	TINCUTA	15342467	LP	FALTA DOCU	Masculino			- Fotocopia carnet padre/madre titular - Formulario 101 - Fotográfias - Fotocopia carnet padre/madre esposo/esposa
820425CCJ	10	Esposo / Esp	JOSE WILDER	CALCINA	CARRIZALES	000000000	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Certificado de nacimiento - Certificado de aportes AFP - Formulario 101 - Fotográfias
135628CBF	10	Hijo menor d	FERNANDA V	CALCINA	BRAVO	15605131	LP	FALTA DOCU	Femenino			- Formulario 101 - Fotográfias - Fotocopia carnet padre/madre esposo/esposa
065324CBA	10	Hijo menor d	ADRIANA BE	CALCINA	BRAVO	13757014	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Certificado de nacimiento - Cetificado de matrimonio padres o Acta de reconocimiento - For
045620CBD	10	Hijo menor d	DANIELA YOS	CALCINA	BRAVO	13757015	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Certificado de nacimiento - Certificado de matrimonio padres o Acta de reconocimiento - For
085828CBP	10	Hijo menor d	PAOLA ALEJA	CALCINA	BRAVO	13757016	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Certificado de matrimonio padres o Acta de reconocimiento - For
815210MIC	12	Titular	CELINA	MONTAÑO	ITURRI	4890158	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Certificado de nacimiento - Fotográfias
981123CMM	12	Hijo mayor d	MICHAEL IM	CRUZ	MONTAÑO	8399813	LP	FALTA DOCU	Masculino			- AVC 06 - Fotocopia carnet de identidad - Certificado de nacimiento - Certificado de estudios (Universidad Privada) o Fotoco
610115YCP	15	Titular	PABLO	YANA	CALLISAYA	2359447	LP	FALTA DOCU	Masculino			- AVC 04 - Formulario 101
600705.QM	2011	Titular	MARIO	.	QUINTANILLA	2460493	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Examen preocupacional - Formulario 101
745619MDM	2011	Esposo / Esp	MAGDALENA	MONROY	DELGADO	4273215	LP	FALTA DOCU	Femenino	73022588		- Fotocopia carnet de identidad - Formulario 101
066220QMD	2011	Hijo menor d	DARLEN NAC	QUINTANILLA	MONROY	9864763	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Fotocopia carnet padre/madre titular - Formulario 101 - Fotocopia carnet padre/madre espo
5901031CLS	2076	Titular	SANTOS	CONDE	LUQUE	2534868	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Certificado de nacimiento - Memorándum de designación o contrato - Formulario 101
670906MCH	21	Titular	HUGO FERNA	MENDEZ	CARRASCO	3391414	LP	FALTA DOCU	Masculino			- Examen preocupacional - Formulario 101 - Fotográfias
845420FDM	21	Esposo / Esp	MARIA ANDR	FRIAS	DREW	4278196	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Formulario 101
090227MFA	21	Hijo menor d	ALBERTO MA	MENDEZ	FRIAS	8485066	LP	FALTA DOCU	Masculino			- Fotocopia carnet padre/madre esposo/esposa
120621MFA	21	Hijo menor d	ANDRE NICO	MENDEZ	FRIAS	14879513	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Fotocopia carnet padre/madre esposo/esposa
005909MRC	21	Hijo mayor d	CAMILA DAN	MENDEZ	RENARD	6105563	LP	FALTA DOCU	Femenino			- Certificado de estudios (Universidad Privada) o Fotocopia de matricula (Universidad Estatal) - Certificado de desendencia -
806005SQG	25	Titular	GRETHZEL R	SILVA	QUENTA	4899770	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Memorándum de designación o contrato
030126TSR	25	Hijo mayor d	ROGHERIO F	TELLERIA	SILVA	000000000	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Certificado de estudios (Universidad Privada) o Fotocopia de matricula (Universidad Estatal
060503TSE	25	Hijo menor d	EMERSON DI	TELLERIA	SILVA	13758448	LP	FALTA DOCU	Masculino			- Fotocopia carnet de identidad - Fotocopia carnet padre/madre titular - Fotocopia carnet padre/madre esposo/esposa
126229RSM	25	Hijo menor d	MEGAN BRIT	RAMOS	SILVA	15419197	LP	FALTA DOCU	Femenino			- Fotocopia carnet padre/madre titular - Fotocopia carnet padre/madre esposo/esposa
640527RHL	26	Titular	LONNIE GIOV	RIVAS	HERRERA	2473588	LP	FALTA DOCU	Masculino	72075456		- Fotocopia carnet de identidad - Memorándum de designación o contrato
675920CCE	26	Esposo / Esp	ELVA ALICIA	CALVO	CALVO	2693219	LP	FALTA DOCU	Femenino			- Fotocopia carnet de identidad - Certificado de nacimiento - Formulario 101
980623QCM	27	Titular	MARCIAL	QUISPE	CATARI	9172950	LP	FALTA DOCU	Masculino	73299931		- Fotocopia carnet de identidad - Examen preocupacional
875917OHR	29	Titular	ROSSIO	OCHOA	HERRERA	6844237	LP	FALTA DOCU	Femenino			- Memorándum de designación o contrato

Ilustración 98: *Visualización para obtener el reporte personal de fichas para el área de fichaje.*

Fuente: Elaboración propia

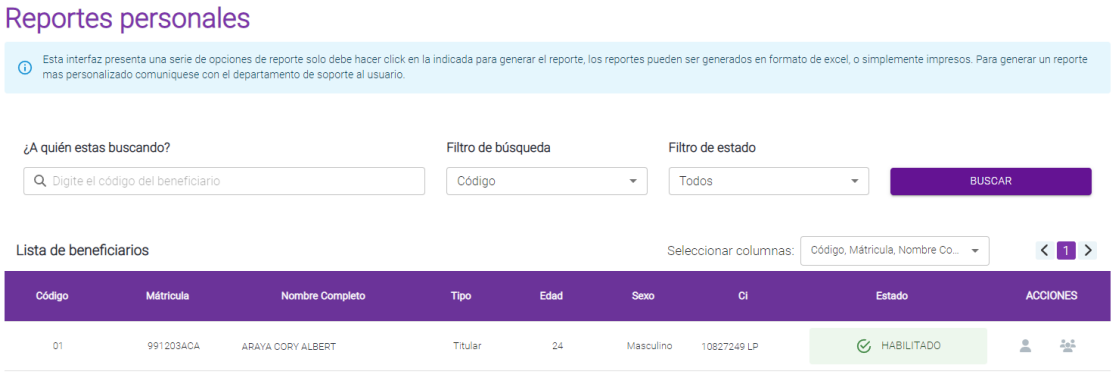


Ilustración 99: *Visualización para obtener el reporte de fichas por dr para el área de fichaje.*

Fuente: Elaboración propia

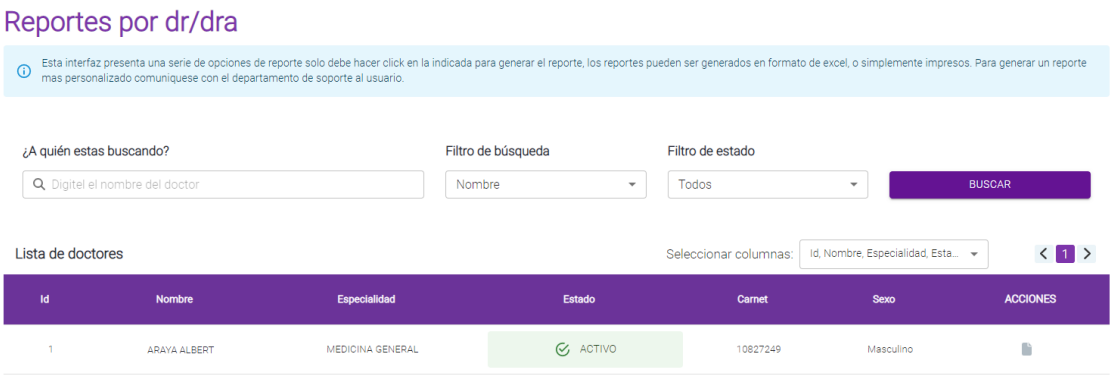
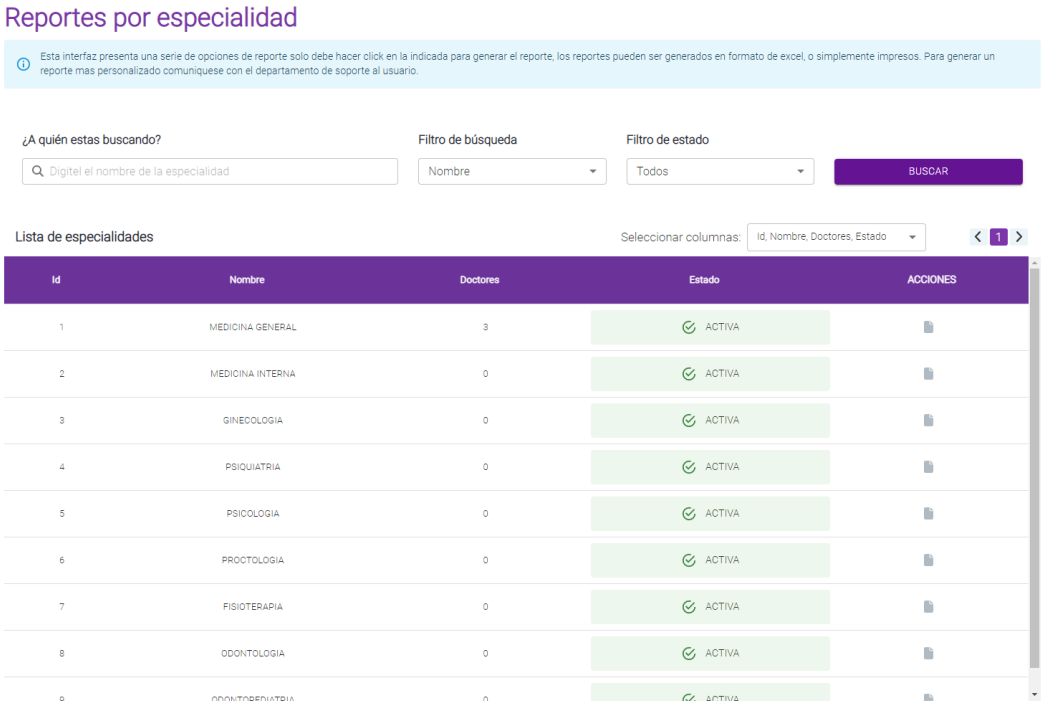


Ilustración 100: *Visualización para obtener el reporte de fichas por especialidad para el área de fichaje.*

Fuente: Elaboración propia



3.5 ANALISIS DE COSTOS

3.5.1 Determina el tamaño del sistema

El tamaño del sistema es una de las métricas más importantes en COCOMO II. Dicho tamaño fue determinado por SLOC (número de líneas de código) expresados en la siguiente tabla:

Tabla 69: Tabla con todas las estimaciones slog.
Fuente: Elaboración propia.

Módulo/Componente	Descripción	Estimación de SLOC
Módulo de Doctores	Backend en Laravel para gestión de doctores	2241
Módulo de Fichas	Backend en Laravel para manejo de fichas	2532
Módulo de Usuarios	Backend en Laravel para gestión de usuarios	3250
Frontend en React/Electron	Interfaz de usuario en React conectada a Electron	4321
Frontend en React (solo web)	Interfaz de usuario en React para web	2986
Cron en Express	Configuración y lógica del cron	837
Socket.io (alertas)	Servidor Socket.io para alertas en tiempo real	1428
Socket.io (fichas)	Servidor Socket.io para manejo de fichas en tiempo real	1569
Pruebas Unitarias y de Integración	Código para pruebas automatizadas	1489

Sumado todo lo desarrollado nos da como resultado un coste total en SLOC de 23500.

$$\mathbf{SLOC_{preliminar}: 18265}$$

$$\mathbf{SLOC_{final}: 20653}$$

3.5.2 Selecciona el nivel de COCOMO II

COCOMO II tiene tres niveles que corresponden a diferentes fases del desarrollo:

- Early Design Model: Ideal para estimaciones tempranas durante la fase de diseño.

- Post-Architecture Model: Usado cuando el diseño del sistema está completo.

- Application Composition Model: Para proyectos que involucran integración de software.

Para el cálculo de este punto se toma en cuenta que el sistema ya está terminado, **Post-Architecture Model** es la opción adecuada.

3.5.3 Identifica los factores de costo (Cost Drivers)

Para el cálculo de los factores de costo se empleó la siguiente tabla:

Tabla 70: Tabla factores de costo del sprint.

Fuente: Elaboración propia.

Categoría	Factor de Costo	Descripción	Bajo (0.85)	Medio (1.00)	Alto (1.15)	Muy Alto (1.30)
Personnel	Analyst Capability (ACAP)	Habilidad de los analistas.	0.85	1.00	1.15	1.30
	Programmer Capability (PCAP)	Habilidad de los programadores.	0.85	1.00	1.15	1.30
	Team Capability (TCAP)	Habilidad del equipo.	0.85	1.00	1.15	1.30
	Software Development Experience (EXP)	Experiencia del equipo en desarrollo de software.	0.90	1.00	1.10	-
Product	Required Software Reliability (RELY)	Fiabilidad requerida del software.	0.82	1.00	1.15	1.40
	Database Size (DATA)	Tamaño de la base de datos.	0.87	1.00	1.15	-
	Product Complexity (CPLX)	Complejidad del producto.	0.70	1.00	1.15	1.30
Project	Development Flexibility (RUSE)	Flexibilidad del desarrollo.	0.87	1.00	1.15	-
	Documentation (DOCU)	Extensión y calidad de la documentación.	0.87	1.00	1.15	-

	Time Constraints (TIME)	Restricciones de tiempo.	1.00	1.20	1.50	-
	Required Development Schedule (SCHED)	Programa de desarrollo requerido.	1.00	1.20	1.50	-
Tools	Tool Use (TOOL)	Uso de herramientas de software.	0.87	1.00	1.15	-
Environment	Multi-Site Development (SITE)	Desarrollo en múltiples sitios.	1.00	1.15	1.30	-
	Staff Turnover (TURN)	Tasa de rotación del personal.	1.00	1.10	1.20	-

Lo cual después de realizar el análisis obtenemos el siguiente resultado:

Tabla 71: Tabla factores de costo

Fuente: Elaboración propia.

Factor de Costo	Nivel	Valor
ACAP (Analyst Capability)	Medio	1,00
PCAP (Programmer Capability)	Alto	1,15
TCAP (Team Capability)	Medio	1,00
EXP (Software Development Experience)	Medio	0,90
RELY (Required Software Reliability)	Alto	1,00
CPLX (Product Complexity)	Muy Alto	1,00
RUSE (Development Flexibility)	Medio	1,00
DOCU (Documentation)	Medio	0,87
TIME (Time Constraints)	Alto	1,00
SCHED (Required Development Schedule)	Medio	1,00
TOOL (Tool Use)	Medio	1,00
SITE (Multi-Site Development)	Alto	1,15
TURN (Staff Turnover)	Medio	1,00

$$\prod_{EM} = 1.00 \times 1.15 \times 1.00 \times 0.90 \times 1.00 \times 1.00 \times 1.00 \times 0.87 \times 1.00 \times 1.00 \times 1.00 \times 1.15 \times 1.00$$

$$\mathbf{\prod_{EM} = 1{,}0355175}$$

3.5.4 Cálculo del esfuerzo en Persona-Meses

Ecuación 15

$$PM = A \times (KSLOC)^{B} \times \prod EM$$

Donde:

- A es una constante que depende del tipo de proyecto (usualmente **2.94** para sistemas orgánicos).

- KSLOC es el tamaño del software en miles de líneas de código (Kilo SLOC). El cual es **20.653**

- E es el exponente de escala, que varía según los factores del proyecto y tiene un valor base de 1.05 a 1.20. el cual es **1,1769**

- EM son los multiplicadores de esfuerzo o Effort Multipliers, basados en los 17 factores de costo. el cual es **1,0355175**

Remplazando:

$$PM = A \times (KSLOC)^{E} \times \prod EM$$

$$PM = 2.94 \text{ x } 20{,}653^{1{,}1769} \text{ x } 1{,}0355175$$

$$\mathbf{PM = 107{,}43 \text{ persona-mes}}$$

3.5.5 Determina el tiempo de desarrollo

Una vez calculado el esfuerzo en Persona-Meses, puedes estimar el tiempo de desarrollo total (TDEV) usando:

Ecuación 16

$$TDEV = C \times (PM)^{F}$$

Donde:

- C es una constante (usualmente 3.67).
- F es el exponente de tiempo, que usualmente varía de 0.28 a 0.91 según el tipo de proyecto. Como fue calculado previamente, simplemente usaremos el valor de:

$$\mathbf{F = 0,4697}$$

Remplazando:

Ecuación 17

$$TDEV = C \times (PM)^{F}$$

$$TDEV = 3.67 \times (107,43)^{0,4697}$$

$$\mathbf{TDEV = 33,01\ Meses}$$

3.5.6 Calcula el costo total

Finalmente, para obtener el costo del proyecto, se multiplica el esfuerzo estimado (en Persona-Meses) por el costo promedio de un desarrollador por mes.

Ecuación 18

$$Costo\ Total = PM \times Costo\ por\ Persona\text{-}Mes$$

$$Costo\ Total = 157,711062 \text{ x } 350$$

$$\mathbf{Costo\ Total = 37.599\ \$us.}$$

CAPÍTULO 4:

CONCLUCIONES Y RECOMENDACIONES

4 CONCLUCIONES Y RECOMENDACIONES

4.1 Conclusiones

El desarrollo del sistema integrado de gestión de documentos y emisión de fichas médicas para el Seguro Médico Delegado de COTEL alcanzó todos los objetivos planteados con éxito, consolidando un proyecto que integra innovación tecnológica y mejoras sustanciales en la operatividad diaria de las áreas de vigencia de derechos y fichaje.

En primer lugar, el análisis detallado de las actividades y procesos de las áreas involucradas permitió identificar de manera precisa los requerimientos funcionales y no funcionales necesarios para el diseño del sistema. Este estudio sirvió de base para el desarrollo de un sistema alineado con las necesidades específicas de los usuarios y con una arquitectura que responde a las expectativas de fiabilidad y eficiencia.

El diseño y desarrollo de la arquitectura del sistema y las APIs REST para los módulos de vigencia de derechos y fichaje resultaron en una solución escalable y modular. Cada módulo fue creado con su propia base de datos, garantizando una comunicación fluida entre los servicios y mejorando la capacidad de gestionar datos y documentos de los beneficiarios de forma precisa. Esta configuración también facilitó la implementación de medidas de seguridad, como el uso de JWT para la autenticación de usuarios, lo cual asegura un control de accesos eficiente.

La infraestructura de Kubernetes, desplegada en un servidor dedicado, permitió la orquestación efectiva de los contenedores de los módulos monolíticos, optimizando la gestión de recursos y la escalabilidad del sistema. Esta tecnología no solo mejoró la distribución de cargas de trabajo, sino que también facilitó el mantenimiento y actualización de los módulos sin interrupciones. A pesar de algunos retos iniciales en la configuración de Kubernetes y su integración con el sistema, estos fueron superados mediante pruebas exhaustivas y ajustes continuos.

El sistema fue validado mediante pruebas unitarias y funcionales, las cuales demostraron su correcto funcionamiento en escenarios reales. Estas pruebas permitieron corroborar que el sistema no solo cumple con los objetivos de eliminar la inconsistencia en los datos y

documentos de los beneficiarios, sino que también mejora la eficiencia operativa y reduce errores humanos.

Otro aspecto destacado fue la implementación de flujos de integración continua (CI) y entrega continua (CD). Estos flujos permitieron automatizar procesos de despliegue y pruebas, acelerando el ciclo de desarrollo y minimizando el tiempo de inactividad. La integración de herramientas de gestión ágil, como Notion, facilitó la organización de tareas y la planificación de sprints, mejorando la coordinación del equipo de desarrollo y optimizando la productividad en todas las etapas del proyecto.

El desarrollo del frontend enfrentó desafíos, particularmente en la gestión de rutas y la implementación de middleware para la seguridad de la aplicación. No obstante, se logró construir una interfaz funcional y amigable para los usuarios, cumpliendo con los requisitos de usabilidad y accesibilidad previstos. Este desarrollo contribuyó a una experiencia de usuario eficiente y satisfactoria, consolidando el sistema como una herramienta confiable y moderna para la gestión de fichas médicas y documentos.

En conclusión, el proyecto ha cumplido plenamente con el objetivo general de eliminar la inconsistencia en los datos y documentos de los beneficiarios mediante una solución integral y escalable. El uso de tecnologías modernas como Kubernetes y prácticas de desarrollo avanzadas garantizó un sistema robusto, adaptable y alineado con las necesidades del Seguro Médico Delegado de COTEL. Esta experiencia ofrece un marco sólido para futuros desarrollos y plantea un camino claro para la adopción de mejores prácticas en la gestión tecnológica de organizaciones similares.

4.2 Recomendaciones

4.2.1 Planificación e investigación detallada de Kong Gateway

Dada la complejidad observada en la implementación de Kong Gateway, se sugiere que futuros proyectos asignen más recursos para su estudio y configuración. La documentación de Kong es extensa y seguir las mejores prácticas es esencial para evitar problemas que afecten el rendimiento o la seguridad del sistema. Además, se recomienda evaluar otras soluciones de gestión de API que puedan ofrecer un mejor equilibrio entre costo y funcionalidad según el contexto del proyecto.

4.2.2 Continuidad en el uso de flujos de CI/CD

La implementación de flujos de CI/CD fue determinante para el éxito del proyecto, automatizando pruebas y despliegues y asegurando la calidad del código. Esta práctica debe mantenerse en futuros desarrollos, ya que mejora la eficiencia del equipo, reduce errores y permite un despliegue más ágil de nuevas funcionalidades y actualizaciones.

4.2.3 Infraestructura de servidores propios

Aunque se logró desplegar el sistema en DigitalOcean, la decisión de migrar a servidores propios subraya la importancia de evaluar los costos a largo plazo de la infraestructura en la nube. Para futuros proyectos, se recomienda analizar detenidamente el balance entre los costos operativos de la nube versus los beneficios de contar con infraestructura interna, tomando en cuenta las necesidades de escalabilidad y mantenimiento a largo plazo.

Con estas recomendaciones, futuros proyectos podrán beneficiarse del aprendizaje obtenido en este desarrollo, mejorando la eficiencia, la calidad del software y la experiencia del usuario final.

BIBLIOGRAFÍA

Abrisketa, J. (s.f.). *Diccionario de Acción Humanitaria*. Obtenido de Diccionario de Acción Humanitaria: https://www.dicc.hegoa.ehu.eus/listar/mostrar/210.html

Agile Alliance. (2001). *Manifiesto por el Desarrollo Ágil de Software.* Obtenido de https://agilemanifesto.org/iso/es/manifesto.html

Agudelo, A. (2017). Proceso de atención integral en salud: Un enfoque hacia la calidad. *Revista de Salud Pública.* Obtenido de Proceso de atención integral en salud: Un enfoque hacia la calidad. .

Albretch, A. (1979). *Measuring Application Development Productivity.* CA: IBM Application Development Symposium, Monterrey.

Amazon Web Services. (s.f.). *AWS*. Obtenido de https://aws.amazon.com/es/what-is/scrum/

Arimetrics. (2024). *Notion*. Obtenido de Notion: https://www.arimetrics.com/glosario-digital/notion

Armijos, I. L. (2019). *DESARROLLO DE UN SISTEMA WEB DE GESTIÓN DE FICHAS MÉDICAS PARA LA ORGANIZACIÓN "SALUD INTEGRAL PARA TODOS", BAJO LA METODOLOGÍA SCRUM.*

ASSUS. (2022). *REGLAMENTO DE AFILIACIÓN, DESAFILIACIÓN Y REAFILIACIÓN PARA SEGURIDAD SOCIAL A CORTO PLAZO.* Obtenido de https://www.asuss.gob.bo/wp-content/uploads/2022/06/REGLAMENTO-DE-AFILIACIONES.pdf

AWS. (2024). *¿Qué es Docker?* Obtenido de https://aws.amazon.com/es/docker/

Barrios, D. (2019). *MANUAL DE PROCESOS Y PROCEDIMIENTOS DEL SEGURO MEDICO DELEGADO DE COTEL.* La Paz.

Behrens, C. (1983). *Measuring the Productivity of Computer Systems.* IEEET Transactions on.

Boehm, B. W. (1995). *Cost estimation model for software development: COCOMO II. Addison-Wesley.*

Caxton, G. (s.f.). How Private Insurance Works: A Primer. *Institution for Health Care Research and Policy, Georgetown University*.

COCOMO II. (0). *Documentos de la ayuda del software COCOMO II.0.*

Columbusstate. (s.f.). *Software Size Measurement: A Framework for Counting Source Statements.* Obtenido de http://csc.columbusstate.edu/carroll/6180/measuringSoftwareSize/seiCodeCountingChecklist.pdf

Concepto. (s.f.). Obtenido de https://concepto.de/diagrama/

Cristancho, F. (26 de Julio de 2022). ¿Qué es Laravel?

Cuervo, V. (14 de Febrero de 2019). *¿Qué es Postman?* Obtenido de ¿Qué es Postman?: https://arquitectoit.com/postman/que-es-postman/

Developers Survey. (2024). *Developers Survey.* Obtenido de https://survey.stackoverflow.co/2024/technology

Electron. (s.f.). *Blog Electron*. Obtenido de https://www.electronjs.org/es/docs/latest

Figma. (s.f.). *Figma*. Obtenido de https://help.figma.com/hc/es-419/articles/14563969806359--Qu%C3%A9-es-Figma

García, I. J. (2021). *ORQUESTACIÓN DE CONTENEDORES CON KUBERNETES.* Madrid.

GitHub. (2024). *Entender las GitHub Actions.* Obtenido de Entender las GitHub Actions: https://docs.github.com/es/actions/about-github-actions/understanding-github-actions

GitHub. (s.f.). *Documentación de GitHub*. Obtenido de https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

GitHub. (s.f.). *Flujo de trabajo con Git*. Obtenido de Flujo de trabajo con Git: https://aulasoftwarelibre.github.io/taller-de-git/gitflow/

Goethert, W., Bailey, E., & Busby, M. (1992). *Software Effort and Cronograma.* Pittsburgh, PA.: CMU/SEI-92-TR-21, Software Engineering.

Gómez, A., C.López, M., Migani, S., & Otazú, A. (s.f.). *UN MODELO DE ESTIMACION DE PROYECTOS DE SOFTWARE.* Obtenido de UN MODELO DE ESTIMACION DE PROYECTOS DE SOFTWARE: https://blogadmi1.wordpress.com/wp-content/uploads/2010/11/cocom0llfull.pdf

Gonzales, C. (2020). *SISTEMA WEB DE ADMINISTRACION DE HISTORIAS CLÍNICAS.*

Green, A. (22 de Octubre de 2021). *GitMinf.* Obtenido de https://gitmind.com/es/ejemplos-diagrama-erd.html

IBM. (s.f.). *¿Qué son las pruebas de software?* Obtenido de https://www.ibm.com/es-es/topics/software-testing

IEPUG. (1994). *IFPUG Function Point Counting Practices: Manual Release 4.0,.* OH: Group, Westerville.

Infoleyes. (2 de Junio de 2000). Obtenido de Infoleyes: https://bolivia.infoleyes.com/norma/2369/decreto-supremo-25798

Isla, A. (24 de Diciembre de 2022). *Medium*. Obtenido de Medium: https://willywes.medium.com/por-qu%C3%A9-es-una-buena-idea-utilizar-gitflow-c92c5e7754e0

ISO. (2000). Obtenido de https://gestiondecalidadmpn.wordpress.com/wp-content/uploads/2012/02/iso-9000-2000-sistemas-de-gestic3b3n-de-la-calidad-conceptos-y-vocabulario.pdf

ISO/EIC 25010. (s.f.). Obtenido de https://iso25000.com/index.php/normas-iso-25000/iso-25010

ittgweb. (29 de Mayo de 2016). *Wordpress.* Obtenido de https://ittgweb.wordpress.com/2016/05/29/descomposicion-modular/

JWT. (2024). Obtenido de https://jwt.io/introduction

KeepCoding. (2024). Obtenido de https://keepcoding.io/blog/que-son-las-pruebas-de-caja-negra/

kong. (2024). *Kong Gateway.* Obtenido de Kong Gateway: https://docs.konghq.com/gateway/latest/#:~:text=Kong%20Gateway%20is%20a%20lightweight,extended%20through%20modules%20and%20plugins.

Kubernetes. (2024). *¿Qué es Kubernetes?* Obtenido de ¿Qué es Kubernetes?: https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

Kunkler, J. (1985). *Cooperative Industry Study on Software.* Rochester, NY 14644: Xerox Corporation, Xerox Square -.

Linkedin. (s.f.). *¿Cuáles son los beneficios y desafíos de usar COCOMO II para la estimación de costos de mantenimiento de software?* Obtenido de ¿Cuáles son los beneficios y desafíos de usar COCOMO II para la estimación de costos de mantenimiento de software?: https://www.linkedin.com/advice/1/what-benefits-challenges-using-cocomo?lang=es&originalSubdomain=es#:~:text=COCOMO%20II%20significa%20Modelo%20de,de%20los%20proyectos%20de%20software.

Microsoft. (s.f.). *Microsoft Learn*. Obtenido de https://learn.microsoft.com/es-es/devops/develop/git/what-is-git

Miro. (s.f.). Obtenido de https://miro.com/es/diagrama/que-es-diagrama-entidad-relacion/

Olsina, L. (2024). Obtenido de https://www.researchgate.net/publication/338660264_Ingenieria_Web_Evaluacion_de_Calidad_de_Sitios_y_Aplicaciones_Web_usando_WebQEM

OMS. (2020). *La atención médica de calidad y su impacto en la salud.* .

OMS. (2020). *La importancia del personal de enfermería en la atención de salud.* Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/nursing-and-midwifery

OMS. (2020). *Servicios sanitarios de calidad*. Obtenido de Servicios sanitarios de calidad: https://www.who.int/es/news-room/fact-sheets/detail/quality-health-services

Oracle. (2024). *¿Qué es MySQL?* Obtenido de ¿Qué es MySQL?: https://www.oracle.com/pe/mysql/what-is-mysql/

Organización Internacional del Trabajo. (2001). HECHOS CONCRETOS SOBRE LA SEGURIDAD SOCIAL., (pág. 1). Ginebra.

Park, R. (1992). *Software Size Measurement: A Framework for Counting Source.* Pittsburgh, PA: Software Engineering Institute.

Paz, I. C. (2020). *SISTEMA WEB DE ADMINISTRACION DE HISTORIAS CLÍNICAS.* El Alto.

Prado, I. D. (2021). *DESARROLLO E IMPLEMENTACION DE UN SISTEMA DE FICHAS MÉDICA, PARA EL CONSULTORIO DE LA DRA. SARA DELGADO.* Guayaquil.

RAE. (2023). Obtenido de https://dle.rae.es/calidad

React. (s.f.). *React.* Obtenido de https://es.react.dev/

Schwaber, K., & Sutherland, J. (Noviembre de 2020). *La Guía de Scrum.* Obtenido de La Guía de Scrum: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf

Simões, C. (14 de Julio de 2020). *MoSCoW. ¿Qué es y cómo priorizar en el desarrollo de tu aplicación?* Obtenido de MoSCoW. ¿Qué es y cómo priorizar en el desarrollo de tu aplicación?: https://www.itdo.com/blog/moscow-que-es-y-como-priorizar-en-el-desarrollo-de-tu-aplicacion/

Smartdraw. (s.f.). *Símbolos de diagramas de flujo.* Obtenido de https://www.smartdraw.com/flowchart/simbolos-de-diagramas-de-flujo.htm

Trejo, O. (24 de Abril de 2024). Obtenido de https://www.linkedin.com/pulse/tablas-de-decisi%C3%B3n-estructurando-la-l%C3%B3gica-tus-pruebas-oscar-trejo-ngcff/

Universidad Veracruzana. (s.f.). Obtenido de https://www.uv.mx/personal/aherrera/files/2020/05/DIAGRAMAS-DE-FLUJO.pdf

Universitat Carlemany. (21 de Marzo de 2024). *Metodologías de desarrollo de software*. Obtenido de https://www.universitatcarlemany.com/actualidad/blog/metodologias-de-desarrollo-de-software/

Venngage. (s.f.). Obtenido de https://es.venngage.com/blog/diagrama-de-actividades/

Venngage. (s.f.). *Venngage.* Obtenido de https://es.venngage.com/blog/diagrama-de-actividades/

Wikipedia. (2018). *DigitalOcean.* Obtenido de DigitalOcean: https://es.wikipedia.org/wiki/DigitalOcean

Wikipedia. (2024). *Socket.IO.* Obtenido de Socket.IO: https://en.wikipedia.org/wiki/Socket.IO#cite_note-3

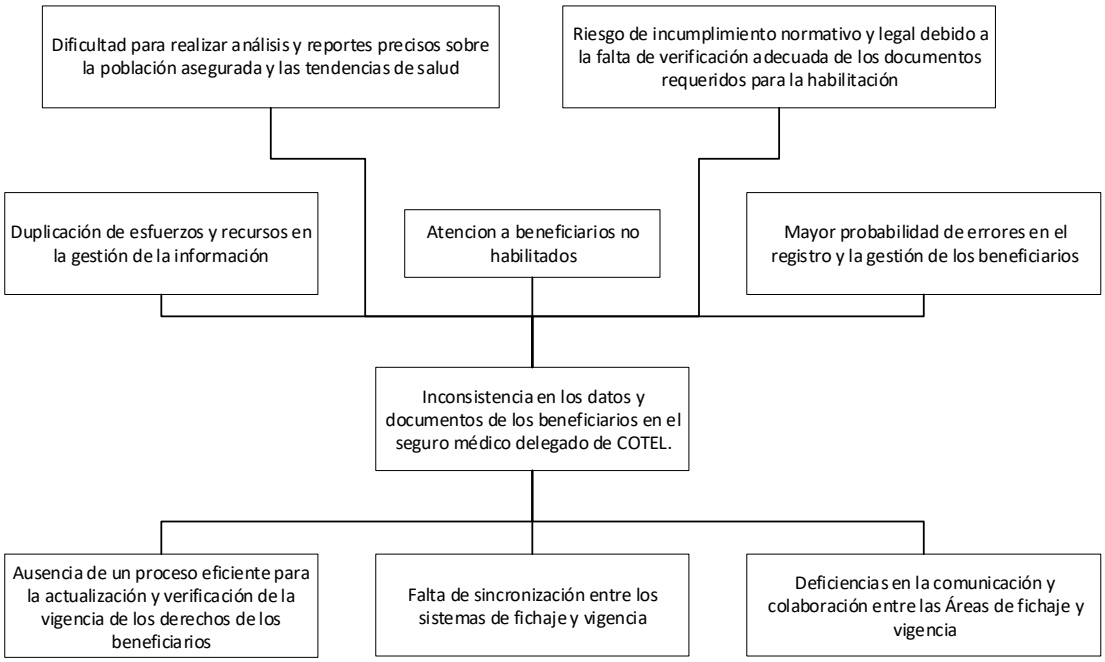
Wikipedia. (s.f.). *Arquitectura de software.* Obtenido de https://es.wikipedia.org/wiki/Arquitectura_de_software

Wikipedia. (s.f.). *Cliente-servidor.* Obtenido de https://es.wikipedia.org/wiki/Cliente-servidor

ANEXOS

ANEXOS A: *Árbol de problemas*

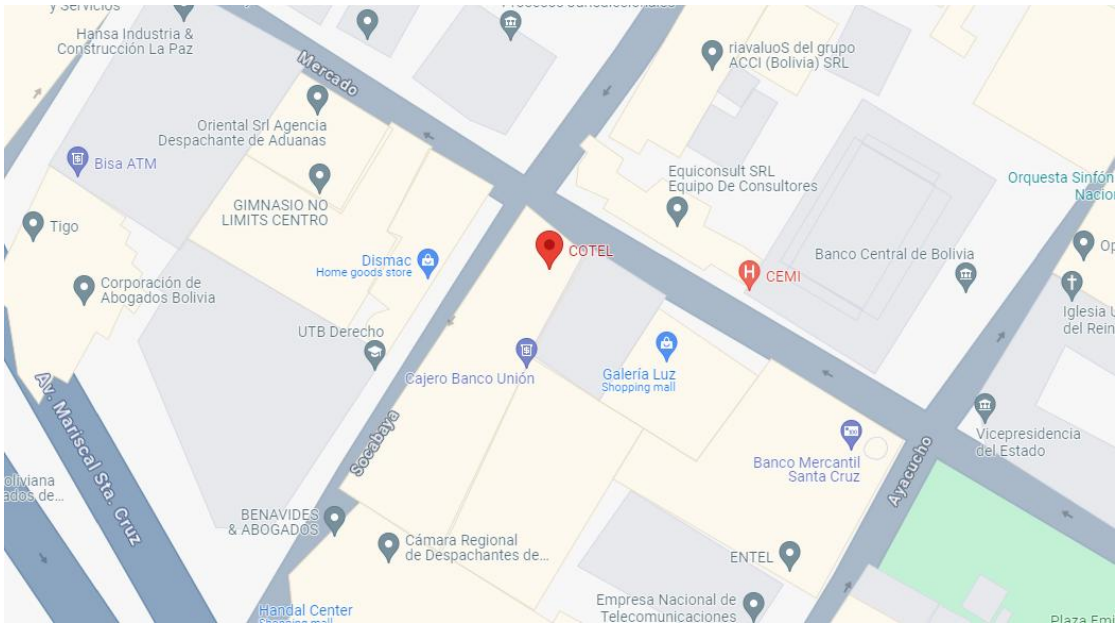
Fuente: Elaboración propia



Nota:En el presente anexo se muestra el árbol de problemas sus causas y efecto que se utilizará para el proyecto. Fuente: Elaboración Propia

ANEXOS B: *Ubicación del Seguro Médico Delegado de COTEL*

Fuente: Elaboración propia



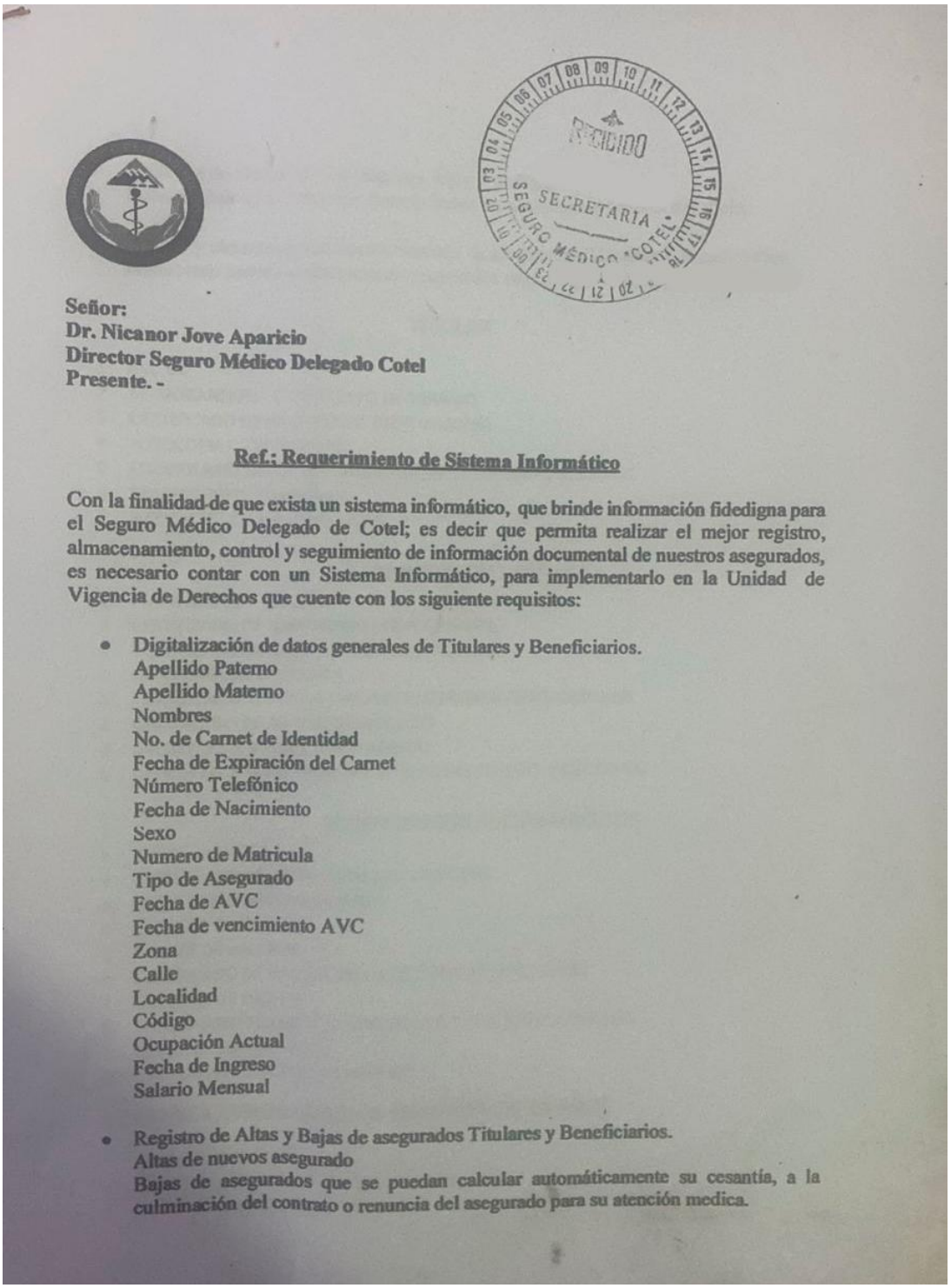
Nota:En el presente anexo se muestra la ubicación del policonsultorio central del seguro médico delegado de COTEL donde se desarrollará el proyecto. Fuente: Elaboración Propia

ANEXOS C: *Hojas de requerimientos firmadas por empleados del seguro médico*

Fuente: Empleados del seguro médico delegado de COTEL

C1*: Hoja 1 de Requerimientos área de vigencia de derechos*

Fuente: Elaboración propia



RECIBIDO
SECRETARIA
SEGURO MÉDICO COTEL

Señor:
Dr. Nicanor Jove Aparicio
Director Seguro Médico Delegado Cotel
Presente. -

Ref.: Requerimiento de Sistema Informático

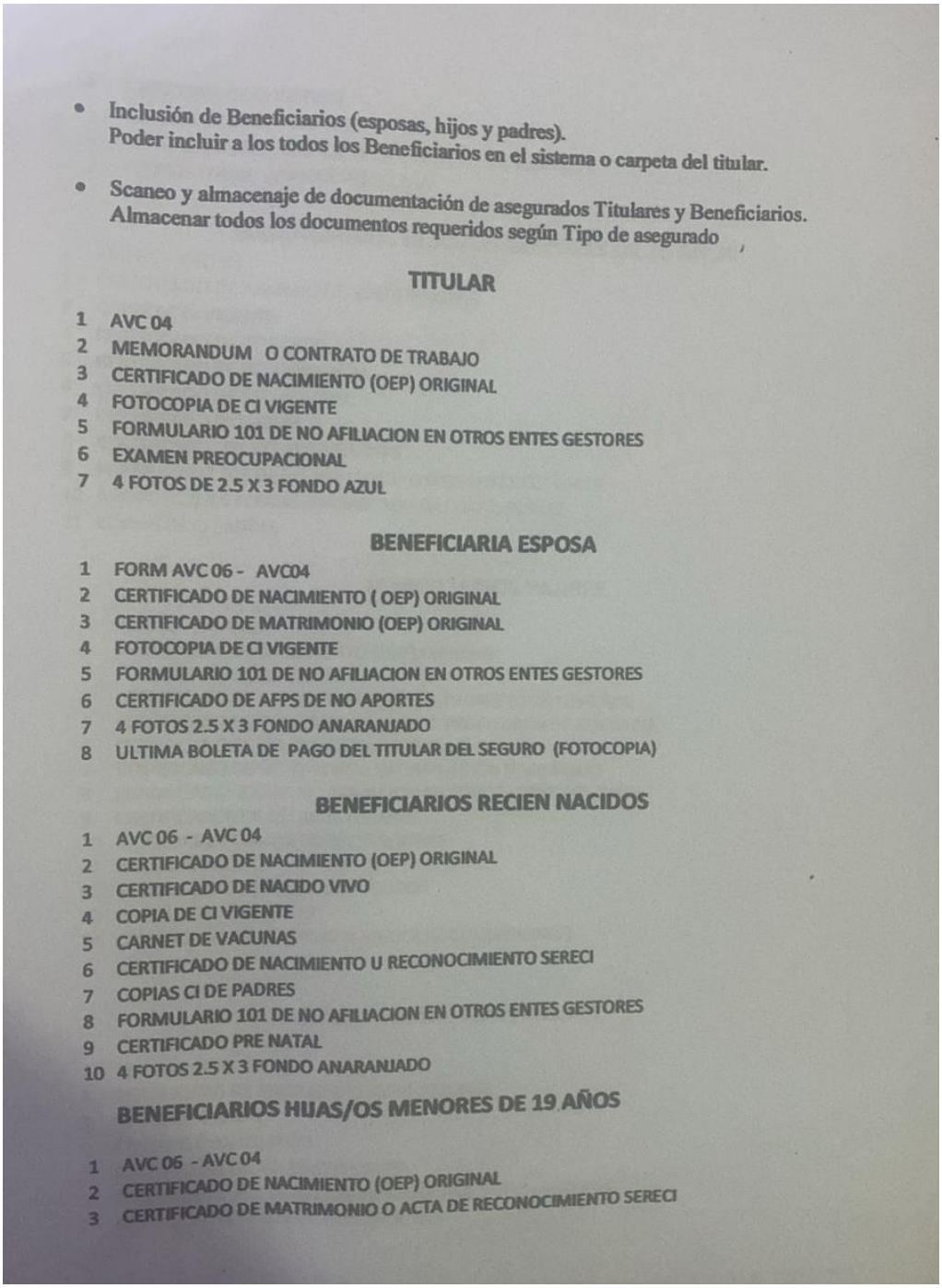
Con la finalidad de que exista un sistema informático, que brinde información fidedigna para el Seguro Médico Delegado de Cotel; es decir que permita realizar el mejor registro, almacenamiento, control y seguimiento de información documental de nuestros asegurados, es necesario contar con un Sistema Informático, para implementarlo en la Unidad de Vigencia de Derechos que cuente con los siguiente requisitos:

- **Digitalización de datos generales de Titulares y Beneficiarios.**
  Apellido Paterno
  Apellido Materno
  Nombres
  No. de Carnet de Identidad
  Fecha de Expiración del Carnet
  Número Telefónico
  Fecha de Nacimiento
  Sexo
  Numero de Matricula
  Tipo de Asegurado
  Fecha de AVC
  Fecha de vencimiento AVC
  Zona
  Calle
  Localidad
  Código
  Ocupación Actual
  Fecha de Ingreso
  Salario Mensual

- **Registro de Altas y Bajas de asegurados Titulares y Beneficiarios.**
  Altas de nuevos asegurado
  Bajas de asegurados que se puedan calcular automáticamente su cesantía, a la culminación del contrato o renuncia del asegurado para su atención medica.

***C2**: Hoja 2 de Requerimientos área de vigencia de derechos*

Fuente: Elaboración propia



- Inclusión de Beneficiarios (esposas, hijos y padres).
  Poder incluir a los todos los Beneficiarios en el sistema o carpeta del titular.

- Scaneo y almacenaje de documentación de asegurados Titulares y Beneficiarios.
  Almacenar todos los documentos requeridos según Tipo de asegurado

TITULAR

1 AVC 04
2 MEMORANDUM O CONTRATO DE TRABAJO
3 CERTIFICADO DE NACIMIENTO (OEP) ORIGINAL
4 FOTOCOPIA DE CI VIGENTE
5 FORMULARIO 101 DE NO AFILIACION EN OTROS ENTES GESTORES
6 EXAMEN PREOCUPACIONAL
7 4 FOTOS DE 2.5 X 3 FONDO AZUL

BENEFICIARIA ESPOSA

1 FORM AVC 06 - AVC04
2 CERTIFICADO DE NACIMIENTO (OEP) ORIGINAL
3 CERTIFICADO DE MATRIMONIO (OEP) ORIGINAL
4 FOTOCOPIA DE CI VIGENTE
5 FORMULARIO 101 DE NO AFILIACION EN OTROS ENTES GESTORES
6 CERTIFICADO DE AFPS DE NO APORTES
7 4 FOTOS 2.5 X 3 FONDO ANARANJADO
8 ULTIMA BOLETA DE PAGO DEL TITULAR DEL SEGURO (FOTOCOPIA)

BENEFICIARIOS RECIEN NACIDOS

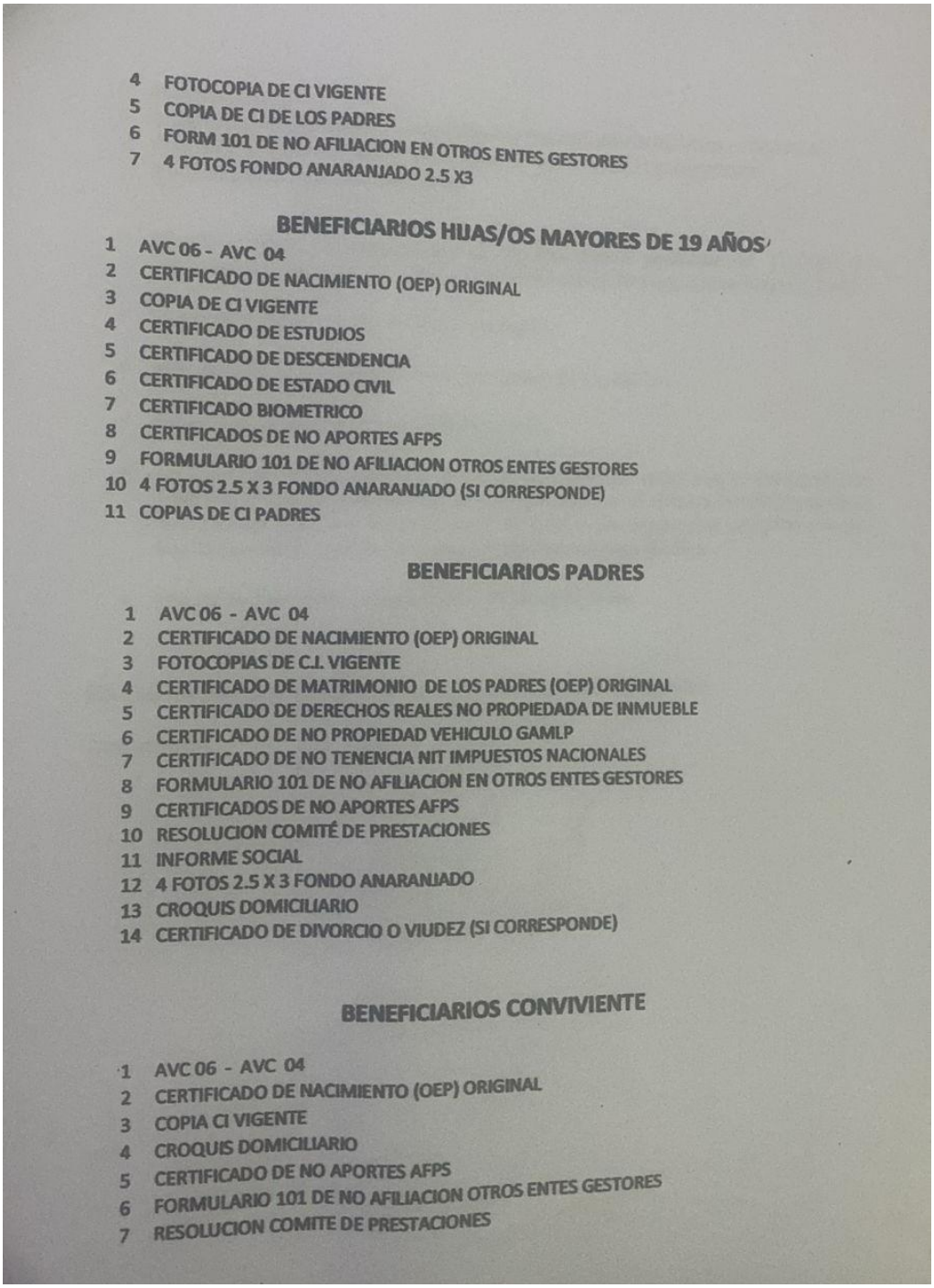
1 AVC 06 - AVC 04
2 CERTIFICADO DE NACIMIENTO (OEP) ORIGINAL
3 CERTIFICADO DE NACIDO VIVO
4 COPIA DE CI VIGENTE
5 CARNET DE VACUNAS
6 CERTIFICADO DE NACIMIENTO U RECONOCIMIENTO SERECI
7 COPIAS CI DE PADRES
8 FORMULARIO 101 DE NO AFILIACION EN OTROS ENTES GESTORES
9 CERTIFICADO PRE NATAL
10 4 FOTOS 2.5 X 3 FONDO ANARANJADO

BENEFICIARIOS HIJAS/OS MENORES DE 19 AÑOS

1 AVC 06 - AVC 04
2 CERTIFICADO DE NACIMIENTO (OEP) ORIGINAL
3 CERTIFICADO DE MATRIMONIO O ACTA DE RECONOCIMIENTO SERECI

***C3**: Hoja 3 de Requerimientos área de vigencia de derechos*

Fuente: Elaboración propia



4 FOTOCOPIA DE CI VIGENTE
5 COPIA DE CI DE LOS PADRES
6 FORM 101 DE NO AFILIACION EN OTROS ENTES GESTORES
7 4 FOTOS FONDO ANARANJADO 2.5 X3

BENEFICIARIOS HIJAS/OS MAYORES DE 19 AÑOS

1 AVC 06 - AVC 04
2 CERTIFICADO DE NACIMIENTO (OEP) ORIGINAL
3 COPIA DE CI VIGENTE
4 CERTIFICADO DE ESTUDIOS
5 CERTIFICADO DE DESCENDENCIA
6 CERTIFICADO DE ESTADO CIVIL
7 CERTIFICADO BIOMETRICO
8 CERTIFICADOS DE NO APORTES AFPS
9 FORMULARIO 101 DE NO AFILIACION OTROS ENTES GESTORES
10 4 FOTOS 2.5 X 3 FONDO ANARANJADO (SI CORRESPONDE)
11 COPIAS DE CI PADRES

BENEFICIARIOS PADRES

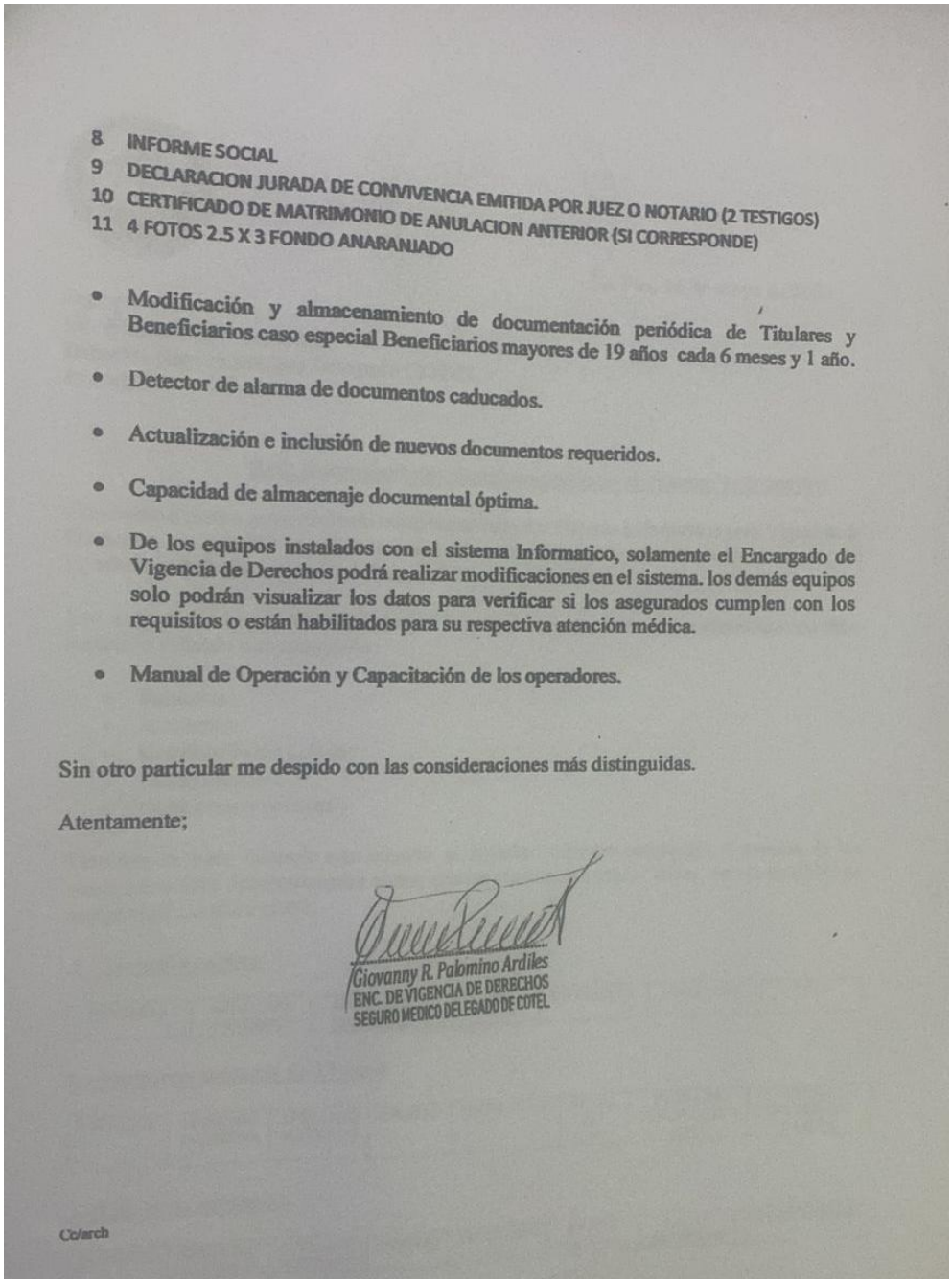
1 AVC 06 - AVC 04
2 CERTIFICADO DE NACIMIENTO (OEP) ORIGINAL
3 FOTOCOPIAS DE C.I. VIGENTE
4 CERTIFICADO DE MATRIMONIO DE LOS PADRES (OEP) ORIGINAL
5 CERTIFICADO DE DERECHOS REALES NO PROPIEDADA DE INMUEBLE
6 CERTIFICADO DE NO PROPIEDAD VEHICULO GAMLP
7 CERTIFICADO DE NO TENENCIA NIT IMPUESTOS NACIONALES
8 FORMULARIO 101 DE NO AFILIACION EN OTROS ENTES GESTORES
9 CERTIFICADOS DE NO APORTES AFPS
10 RESOLUCION COMITÉ DE PRESTACIONES
11 INFORME SOCIAL
12 4 FOTOS 2.5 X 3 FONDO ANARANJADO
13 CROQUIS DOMICILIARIO
14 CERTIFICADO DE DIVORCIO O VIUDEZ (SI CORRESPONDE)

BENEFICIARIOS CONVIVIENTE

1 AVC 06 - AVC 04
2 CERTIFICADO DE NACIMIENTO (OEP) ORIGINAL
3 COPIA CI VIGENTE
4 CROQUIS DOMICILIARIO
5 CERTIFICADO DE NO APORTES AFPS
6 FORMULARIO 101 DE NO AFILIACION OTROS ENTES GESTORES
7 RESOLUCION COMITE DE PRESTACIONES

***C4**: Hoja 4 de Requerimientos área de vigencia de derechos*

Fuente: Elaboración propia



8 INFORME SOCIAL
9 DECLARACION JURADA DE CONVIVENCIA EMITIDA POR JUEZ O NOTARIO (2 TESTIGOS)
10 CERTIFICADO DE MATRIMONIO DE ANULACION ANTERIOR (SI CORRESPONDE)
11 4 FOTOS 2.5 X 3 FONDO ANARANJADO

- Modificación y almacenamiento de documentación periódica de Titulares y Beneficiarios caso especial Beneficiarios mayores de 19 años cada 6 meses y 1 año.
- Detector de alarma de documentos caducados.
- Actualización e inclusión de nuevos documentos requeridos.
- Capacidad de almacenaje documental óptima.
- De los equipos instalados con el sistema Informatico, solamente el Encargado de Vigencia de Derechos podrá realizar modificaciones en el sistema. los demás equipos solo podrán visualizar los datos para verificar si los asegurados cumplen con los requisitos o están habilitados para su respectiva atención médica.
- Manual de Operación y Capacitación de los operadores.

Sin otro particular me despido con las consideraciones más distinguidas.

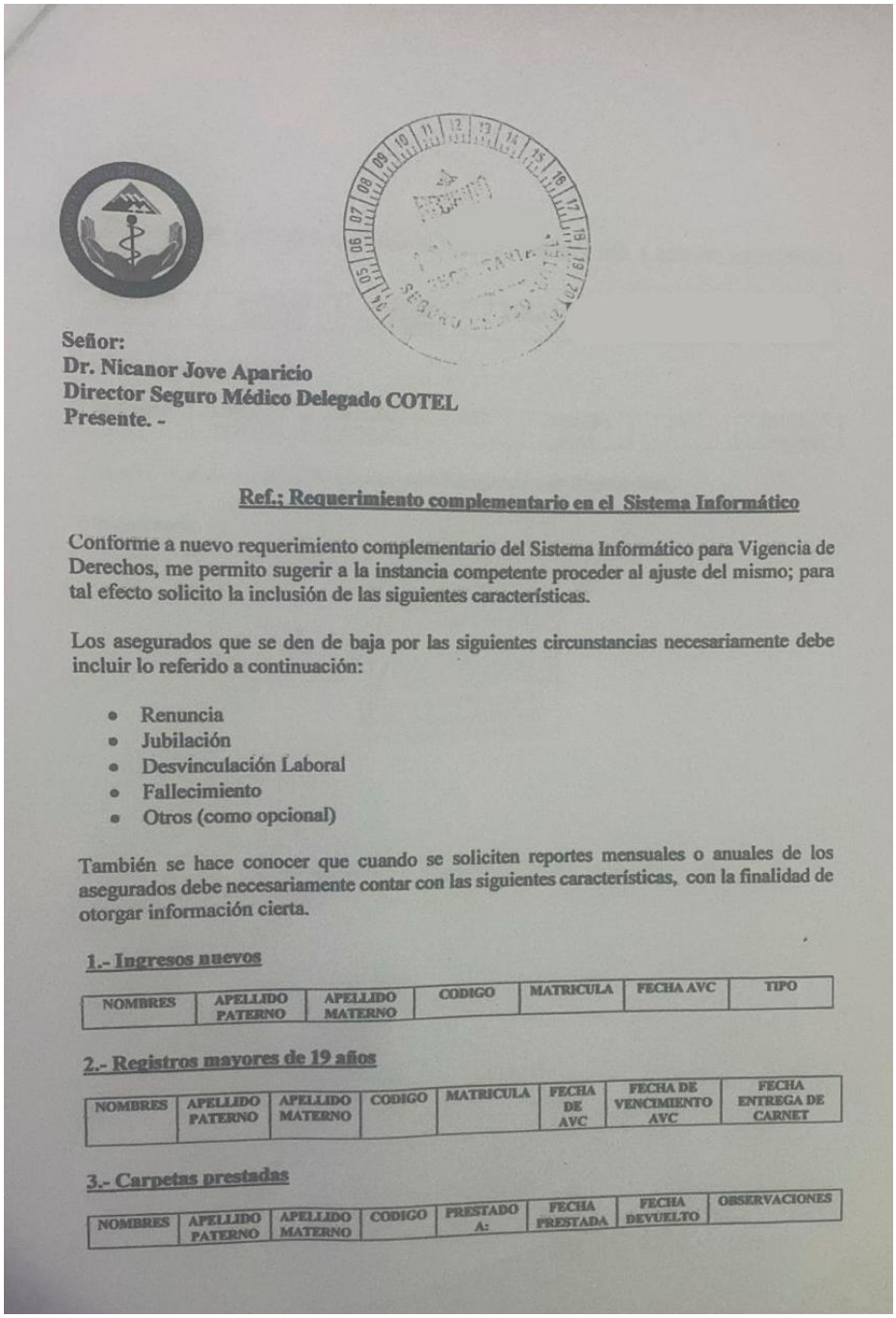
Atentamente;

Giovanny R. Palomino Ardiles
ENC. DE VIGENCIA DE DERECHOS
SEGURO MEDICO DELEGADO DE COTEL

Cc/arch

***C5**: Hoja 5 de Requerimientos área de vigencia de derechos*

Fuente: Elaboración propia



Señor:
Dr. Nicanor Jove Aparicio
Director Seguro Médico Delegado COTEL
Presente. -

Ref.; Requerimiento complementario en el Sistema Informático

Conforme a nuevo requerimiento complementario del Sistema Informático para Vigencia de Derechos, me permito sugerir a la instancia competente proceder al ajuste del mismo; para tal efecto solicito la inclusión de las siguientes características.

Los asegurados que se den de baja por las siguientes circunstancias necesariamente debe incluir lo referido a continuación:

- Renuncia
- Jubilación
- Desvinculación Laboral
- Fallecimiento
- Otros (como opcional)

También se hace conocer que cuando se soliciten reportes mensuales o anuales de los asegurados debe necesariamente contar con las siguientes características, con la finalidad de otorgar información cierta.

1.- Ingresos nuevos

NOMBRES	APELLIDO PATERNO	APELLIDO MATERNO	CODIGO	MATRICULA	FECHA AVC	TIPO

2.- Registros mayores de 19 años

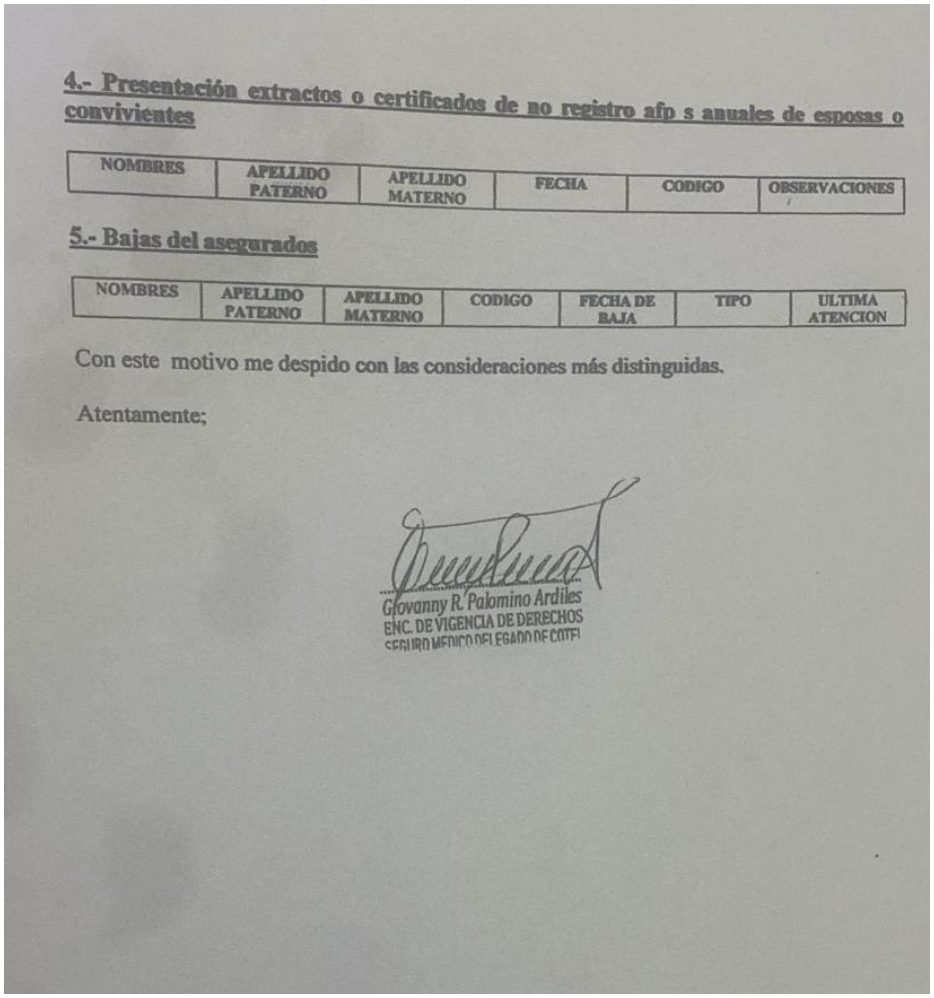
NOMBRES	APELLIDO PATERNO	APELLIDO MATERNO	CODIGO	MATRICULA	FECHA DE AVC	FECHA DE VENCIMIENTO AVC	FECHA ENTREGA DE CARNET

3.- Carpetas prestadas

NOMBRES	APELLIDO PATERNO	APELLIDO MATERNO	CODIGO	PRESTADO A:	FECHA PRESTADA	FECHA DEVUELTO	OBSERVACIONES

***C6**: Hoja 6 de Requerimientos área de vigencia de derechos*

Fuente: Elaboración propia



4.- Presentación extractos o certificados de no registro afp s anuales de esposas o convivientes

NOMBRES	APELLIDO PATERNO	APELLIDO MATERNO	FECHA	CODIGO	OBSERVACIONES

5.- Bajas del asegurados

NOMBRES	APELLIDO PATERNO	APELLIDO MATERNO	CODIGO	FECHA DE BAJA	TIPO	ULTIMA ATENCION

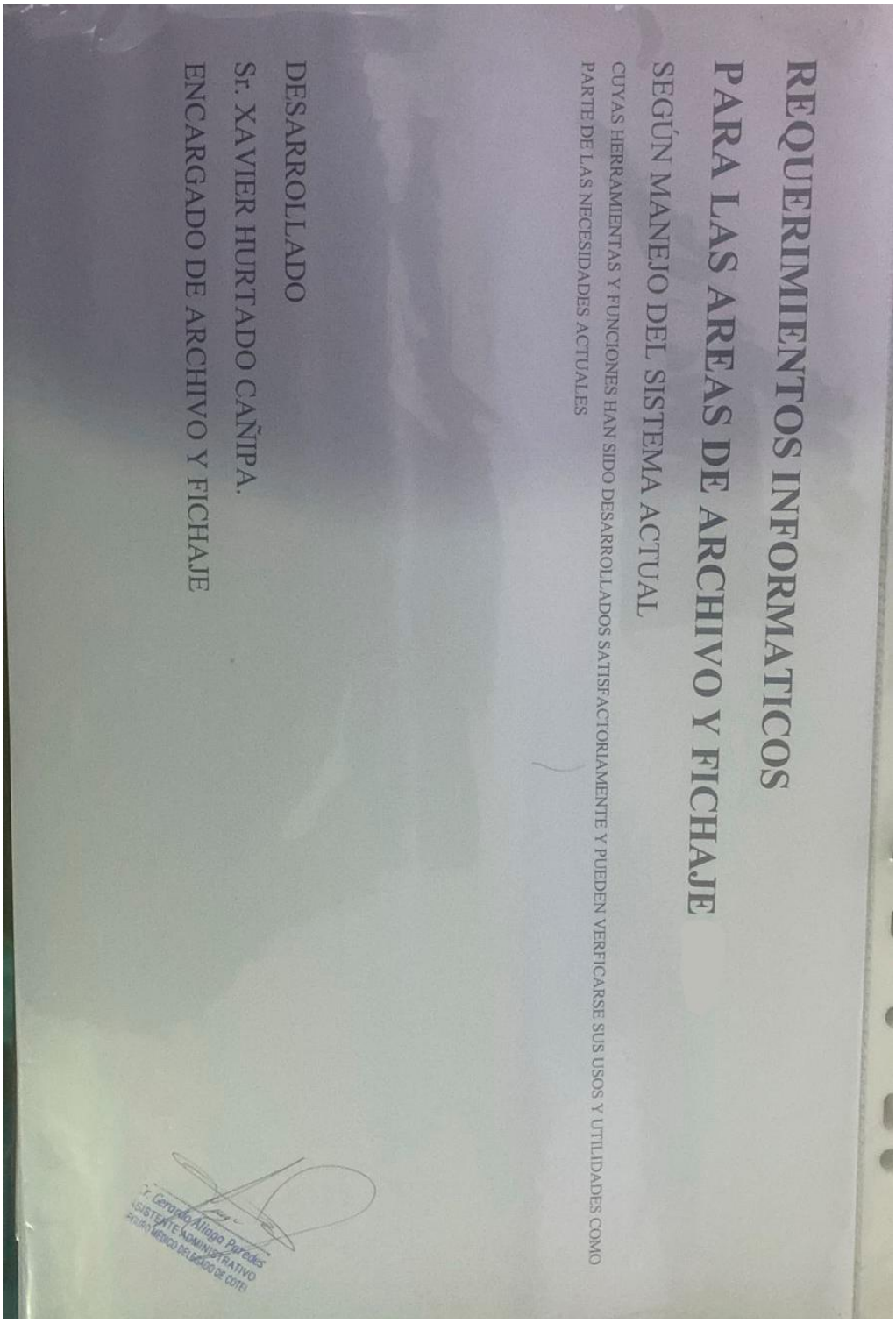
Con este motivo me despido con las consideraciones más distinguidas.

Atentamente;

Giovanny R. Palomino Ardiles
ENC. DE VIGENCIA DE DERECHOS
SEGURO MEDICO DELEGADO DE COTFI

***C7**: Hoja 1 de Requerimientos área de fichaje*

Fuente: Elaboración propia



REQUERIMIENTOS INFORMATICOS

PARA LAS AREAS DE ARCHIVO Y FICHAJE

SEGÚN MANEJO DEL SISTEMA ACTUAL

CUYAS HERRAMIENTAS Y FUNCIONES HAN SIDO DESARROLLADOS SATISFACTORIAMENTE Y PUEDEN VERFICARSE SUS USOS Y UTILIDADES COMO PARTE DE LAS NECESIDADES ACTUALES

DESARROLLADO

Sr. XAVIER HURTADO CAÑIPA.

ENCARGADO DE ARCHIVO Y FICHAJE

Ing. Gerardo Aliaga Paredes
ASISTENTE ADMINISTRATIVO
ARCHIVO MEDICO DEL HOSPITAL BASICO DE COTE

***C8**: Hoja 2 de Requerimientos área de fichaje*

Fuente: Elaboración propia

REQUERIMIENTOS INFORMATICOS ARCHIVO - FICHAJE

EN GENERAL

En general-, el sistema debe poder relacionar tablas, crear registros y modificarlos más adelante, si fuera necesario. También deberá poder exportar datos y tener campos de reserva para futuras consultas en base de datos es importante aclarar que esencialmente existen dos módulos en esta área que son ARCHIVO Y FICHAJE. Por lo que se quiere que un mismo módulo pueda manipularse ambos. En este punto se aclara, sin embargo, que deben también poder manejarse desde dos diferentes computadoras.

Idealmente el sistema corregirá los errores de agnación como ser agendar con un médico que ya esté en hora de salida. Esto puede hacerse de forma hablada por el sistema o bien con cuadros de diálogo.

El sistema deberá ser muy versátil porque en los hechos las situaciones por las que pasan los asegurados obligan al seguro a utilizar las diversas normas del código de salud y estas pueden no estar totalmente contempladas en los límites del sistema.

Aparecerán en el desarrollo más características

AREA FICHAJE

FICHAJE

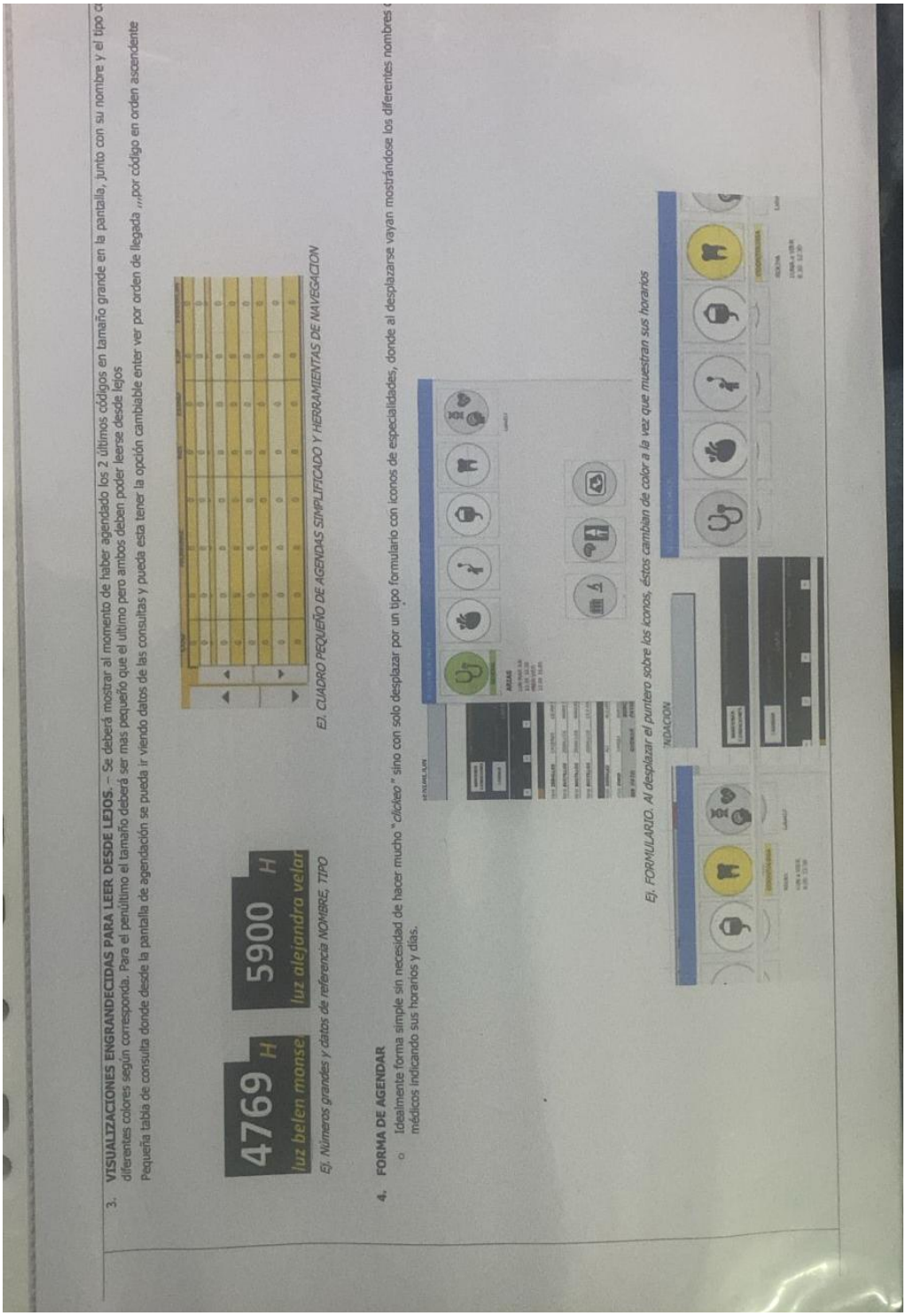
1. **LECTURA DE CREDENCIAL**. Además de las diferentes formas de agendar la de lectura de credencial en ventanilla deberá soportar lo siguiente:
   - Campo donde se registre el tipo de agenda ya sea vía WEB, PROGRAMADO, TELEFONO o PRESENTACION DE CREDENCIAL
   - Si el paciente no porta credencial deberá poder registrarlo indicando el sistema que no presenta credencial. Al momento de usar el lector de barras, deberá registrarlo automáticamente "con credencial" al momento de hacerlo manualmente el sistema deberá tener selector múltiple de documento utilizado como ser: ROBO, PERDIDA, OLVIDO, AVC, **PRESTADO NO LEE CREDENCIAL y deberá tener otro campo de selección múltiple para elegir quien lo autoriza.
   - En el caso de que no pueda enlazar con la base de datos deberá poder hacer un "registro Flotante" donde, en el caso de los mayores de 18 años incluya un 0campo donde se introduzca la fecha de expiración propia que lleva la credencial para verificar con la fecha actual y filtre si el posible realizar la atención.

   **IMPORTANTE. EL SISTEMA DEBERA PODER REALIZAR UNA AGENDA INCLUSO SI EL PACIENTE NO ESTA VIGENTE SOLO PARA CASO DE PRESTAMO A OTRAS UNIDADES COMO SE DIRECTION, CONTAGILIDAD ETC. Esto para cuando saber la ubicación del expediente cuando se pida desde otra unidad ADMINISTRATIVA O AREA PROPIA DEL SEGURO.
2. **TABLAS DE PACIENTES** Al realizar una búsqueda se deberán mostrar las coincidencias relacionadas con la búsqueda. Se deberá poder entonces tener la opción para realizar BUSQUEDAS AVANZADAS donde se podrá especificar o delimitar las coincidencias:
   - Los campos deben estar agrupados por colores; datos numéricos, personales, últimas consultas, últimos estudios como ser Rx, laboratorio, otros estudios, anterior consulta, nombre del médico. IMPORTANTE si no pudiese desplazar esta información podrá incluir un botón donde se relacionen estas tablas de ARCHIVO para así desplegar esta información aquí con marcador amarillo
   - Al momento de navegar dentro de los registros deberá el sistema cambiar el tamaño de letra, color resaltado y en negrilla DE LA FILA DONDE ESTA EL PUNTERO O PROMPT
   - Los datos de peligro deberán estar resaltados con color rojo; edad mayor de 18; próxima baja o fin de cesantía etc.

Ej: BASE DE DATOS. Se han agrupado Las coincidencias de búsqueda por coincidencias en CODIGO Agrupado en coincidencia de MATRICULA, etc

***C9**: Hoja 3 de Requerimientos área de fichaje*

Fuente: Elaboración propia



3. **VISUALIZACIONES ENGRANDECIDAS PARA LEER DESDE LEJOS.** – Se deberá mostrar al momento de haber agendado los 2 últimos códigos en tamaño grande en la pantalla, junto con su nombre y el tipo c
diferentes colores según corresponda. Para el penúltimo el tamaño deberá ser mas pequeño que el ultimo pero ambos deben poder leerse desde lejos
Pequeña tabla de consulta donde desde la pantalla de agendación se pueda ir viendo datos de las consultas y pueda esta tener la opción cambiable enter ver por orden de llegada ,,,por código en orden ascendente

4769 H luz belen monse

5900 H luz alejandra velar

Ej. Números grandes y datos de referencia NOMBRE, TIPO

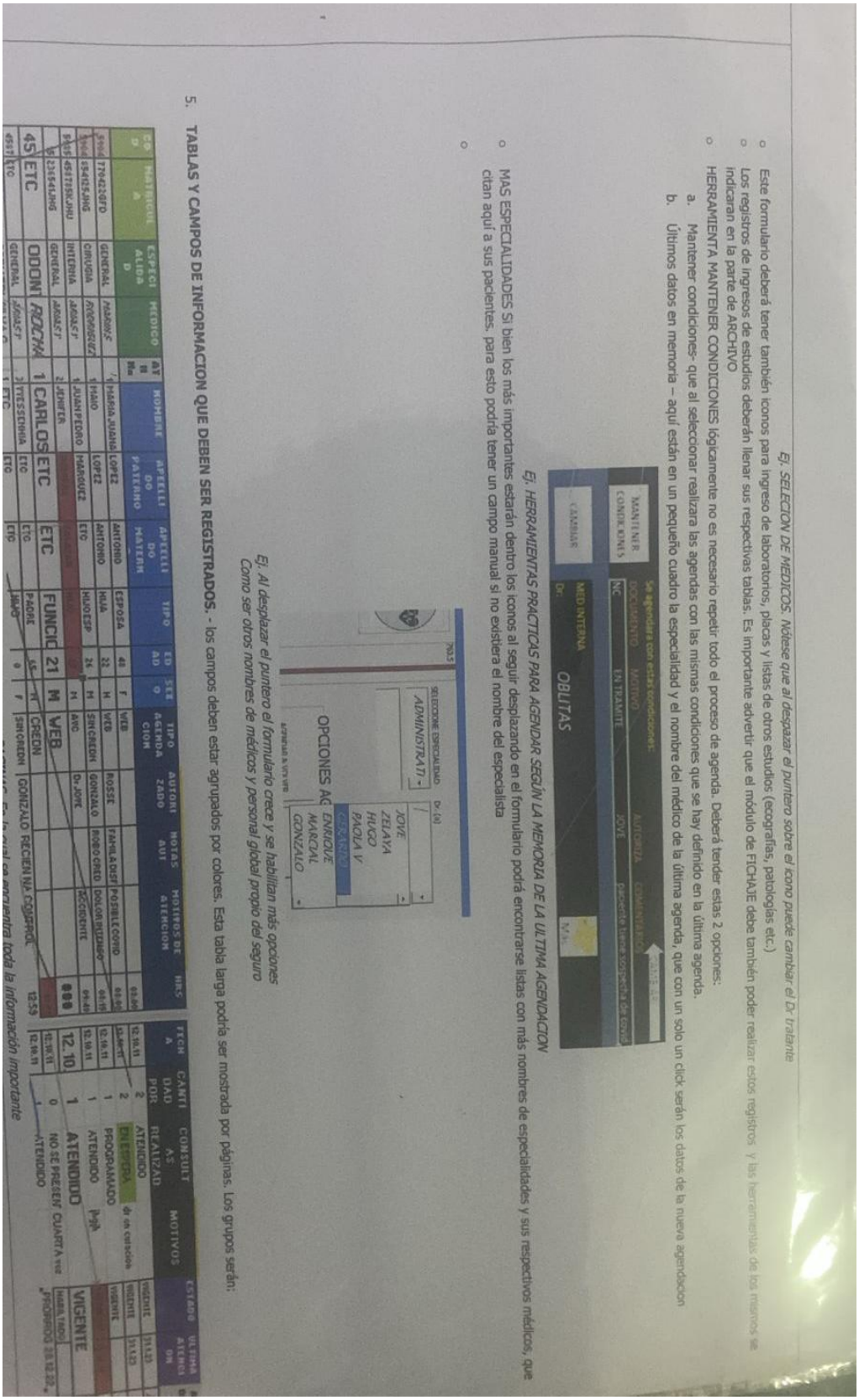
El. CUADRO PEQUEÑO DE AGENDAS SIMPLIFICADO Y HERRAMIENTAS DE NAVEGACION

4. **FORMA DE AGENDAR**
   - Idealmente forma simple sin necesidad de hacer mucho "*clickeo*" sino con solo desplazar por un tipo formulario con iconos de especialidades, donde al desplazarse vayan mostrándose los diferentes nombres
     médicos indicando sus horarios y días.

Ej. FORMULARIO. Al desplazar el puntero sobre los iconos, éstos cambian de color a la vez que muestran sus horarios

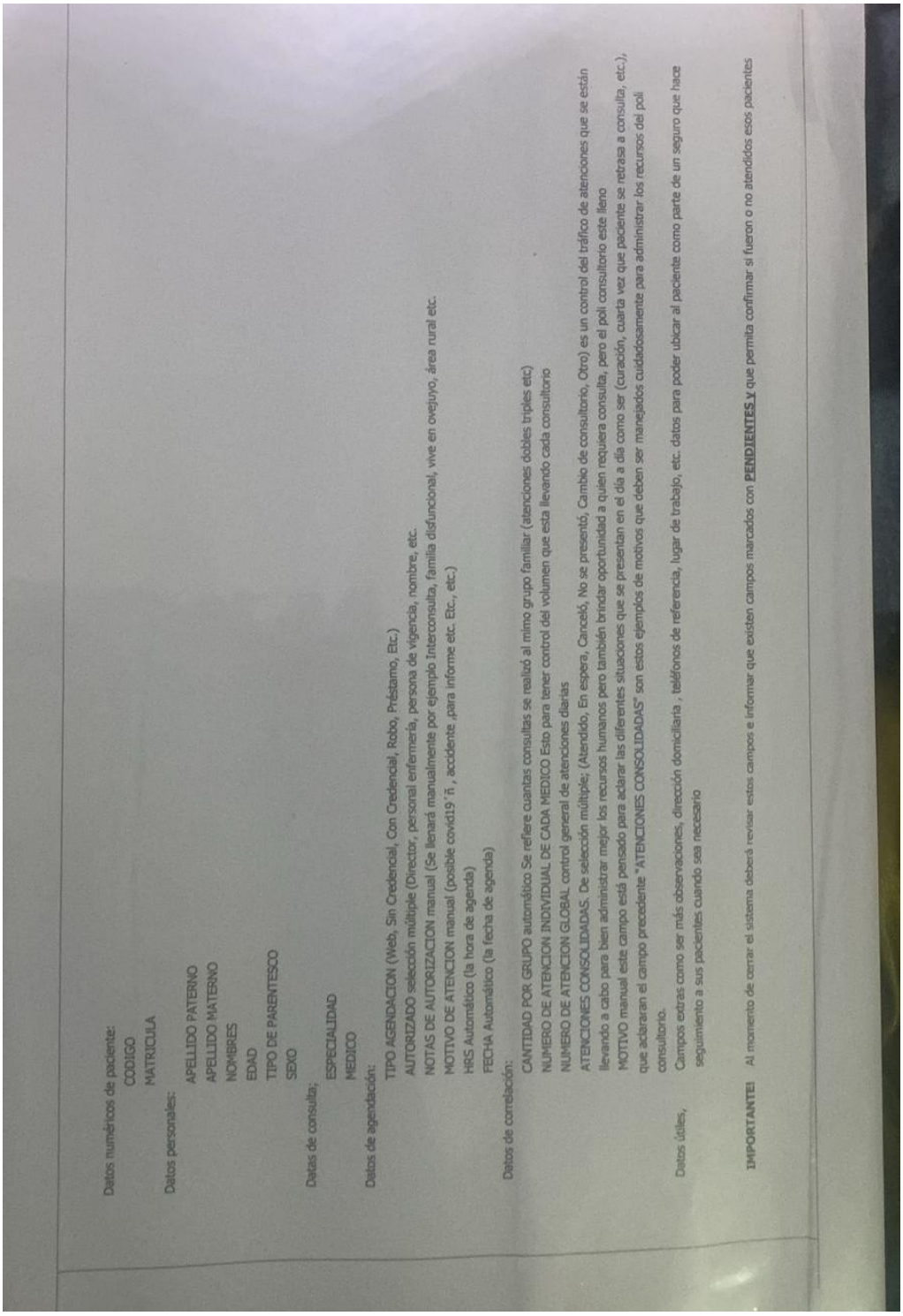
***C10**: Hoja 4 de Requerimientos área de fichaje*

Fuente: Elaboración propia



***C11**: Hoja 5 de Requerimientos área de fichaje*

Fuente: Elaboración propia



Datos numéricos de paciente:
- CODIGO
- MATRICULA

Datos personales:
- APELLIDO PATERNO
- APELLIDO MATERNO
- NOMBRES
- EDAD
- TIPO DE PARENTESCO
- SEXO

Datos de consulta;
- ESPECIALIDAD
- MEDICO

Datos de agendación:
- TIPO AGENDACION (Web, Sin Credencial, Con Credencial, Robo, Préstamo, Etc.)
- AUTORIZADO selección múltiple (Director, personal enfermería, persona de vigencia, nombre, etc.
- NOTAS DE AUTORIZACION manual (Se llenará manualmente por ejemplo Interconsulta, familia disfuncional, vive en ovejuyo, área rural etc.
- MOTIVO DE ATENCION manual (posible covid19 "ñ , accidente ,para informe etc. Etc., etc.)
- HRS Automático (la hora de agenda)
- FECHA Automático (la fecha de agenda)

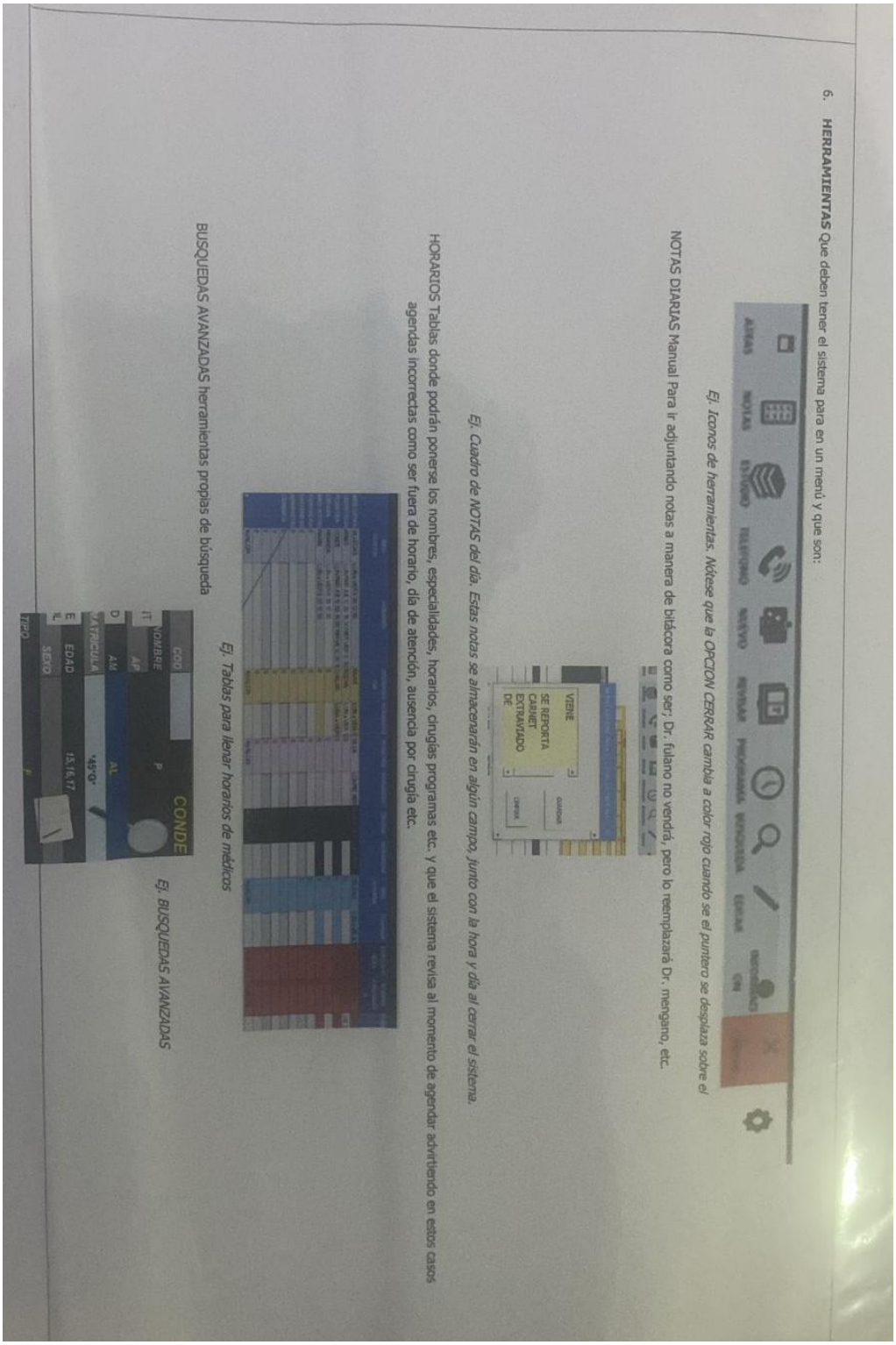
Datos de correlación:
- CANTIDAD POR GRUPO automático Se refiere cuantas consultas se realizó al mismo grupo familiar (atenciones dobles triples etc)
- NUMERO DE ATENCION INDIVIDUAL DE CADA MEDICO Esto para tener control del volumen que esta llevando cada consultorio
- NUMERO DE ATENCION GLOBAL control general de atenciones diarias
- ATENCIONES CONSOLIDADAS. De selección múltiple; (Atendido, En espera, Canceló, No se presentó, Cambio de consultorio, Otro) es un control del tráfico de atenciones que se están llevando a cabo para bien administrar mejor los recursos humanos pero también brindar oportunidad a quien requiera consulta, pero el poli consultorio este lleno
- MOTIVO manual este campo está pensado para aclarar las diferentes situaciones que se presentan en el día a día como ser (curación, cuarta vez que paciente se retrasa a consulta, etc.), que aclararan el campo precedente "ATENCIONES CONSOLIDADAS" son estos ejemplos de motivos que deben ser manejados cuidadosamente para administrar los recursos del poli consultorio.

Datos útiles, Campos extras como ser más observaciones, dirección domiciliaria , teléfonos de referencia, lugar de trabajo, etc. datos para poder ubicar al paciente como parte de un seguro que hace seguimiento a sus pacientes cuando sea necesario

IMPORTANTE! Al momento de cerrar el sistema deberá revisar estos campos e informar que existen campos marcados con **PENDIENTES y** que permita confirmar si fueron o no atendidos esos pacientes

***C12**: Hoja 6 de Requerimientos área de fichaje*

Fuente: Elaboración propia



6. **HERRAMIENTAS** Que deben tener el sistema para en un menú y que son:

Ej. Iconos de herramientas. Nótese que la OPCION CERRAR cambia a color rojo cuando se el puntero se desplaza sobre el

NOTAS DIARIAS Manual Para ir adjuntando notas a manera de bitácora como ser; Dr. fulano no vendrá, pero lo reemplazará Dr. mengano, etc.

Ej. Cuadro de NOTAS del día. Estas notas se almacenarán en algún campo, junto con la hora y día al cerrar el sistema.

HORARIOS Tablas donde podrán ponerse los nombres, especialidades, horarios, cirugías programas etc. y que el sistema revisa al momento de agendar advirtiendo en estos casos agendas incorrectas como ser fuera de horario, día de atención, ausencia por cirugía etc.

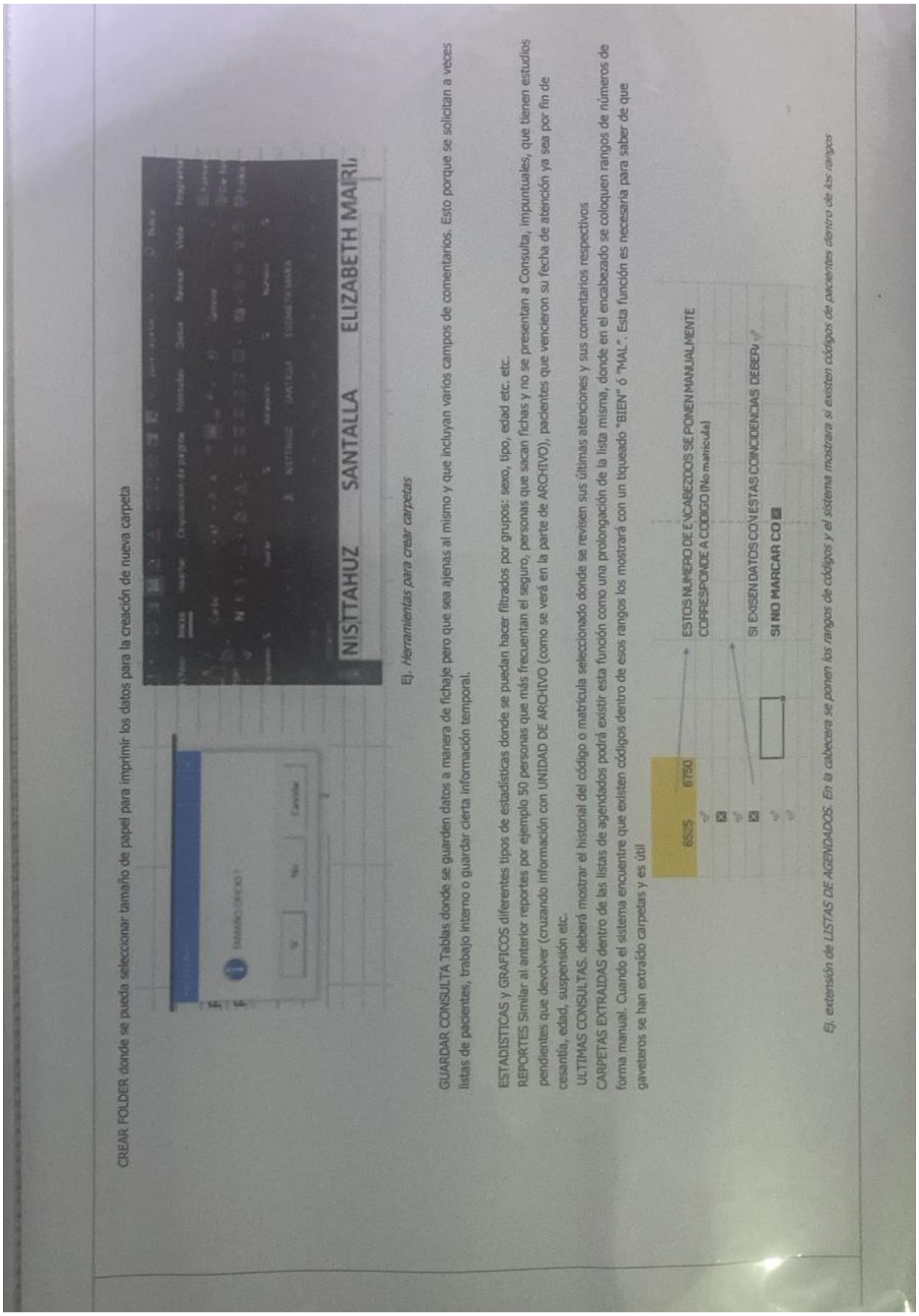
Ej. Tablas para llenar horarios de médicos

BUSQUEDAS AVANZADAS herramientas propias de búsqueda

Ej. BUSQUEDAS AVANZADAS

***C13**: Hoja 7 de Requerimientos área de fichaje*

Fuente: Elaboración propia



CREAR FOLDER donde se pueda seleccionar tamaño de papel para imprimir los datos para la creación de nueva carpeta

Ej. *Herramientas para crear carpetas*

GUARDAR CONSULTA Tablas donde se guarden datos a manera de fichaje pero que sea ajenas al mismo y que incluyan varios campos de comentarios. Esto porque se solicitan a veces listas de pacientes, trabajo interno o guardar cierta información temporal.

ESTADISTICAS y GRAFICOS diferentes tipos de estadísticas donde se puedan hacer filtrados por grupos: sexo, tipo, edad etc. etc.

REPORTES Similar al anterior reportes por ejemplo 50 personas que más frecuentan el seguro, personas que sacan fichas y no se presentan a Consulta, impuntuales, que tienen estudios pendientes que devolver (cruzando información con UNIDAD DE ARCHIVO (como se verá en la parte de ARCHIVO), pacientes que vencieron su fecha de atención ya sea por fin de cesantía, edad, suspensión etc.

ULTIMAS CONSULTAS. deberá mostrar el historial del código o matricula seleccionado donde se revisen sus últimas atenciones y sus comentarios respectivos

CARPETAS EXTRAIDAS dentro de las listas de agendados podrá existir esta función como una prolongación de la lista misma, donde en el encabezado se coloquen rangos de números de forma manual. Cuando el sistema encuentre que existen códigos dentro de esos rangos los mostrará con un tiqueado "BIEN" ó "MAL". Esta función es necesaria para saber de que gaveteros se han extraído carpetas y es útil

Ej. extensión de LISTAS DE AGENDADOS. En la cabecera se ponen los rangos de códigos y el sistema mostrara si existen códigos de pacientes dentro de los rangos

***C14**: Hoja 8 de Requerimientos área de fichaje*

Fuente: Elaboración propia

EXPORTAR IMPORTAR DATOS .Si por algún motivo no se pudo actualizar algún tipo de registro diario como ser fallas de internet, o cualquier otra razón, deberá poder importarse datos desde Excel. Igualmente exportar datos a Excel

PROGRAMAR Donde se pueda programar agendas desde días anteriores

RECORDATORIOS se utilizará para guardar recordatorios con un CALENDARIO que se combinará con la herramienta PROGRAMAR. También ocasionalmente se nos ha solicitado tener ciertas historias lista para ciertas fechas

CONDICIONES DE AGENDACION poder anclar desde aquí también condiciones de agenda para no tener que repetir todo el proceso de selección de médico y agendar

TELEFONOS poder almacenar nuevos números de personal que cambia constantemente. Pero también poder acceder a los teléfonos de vigencia (este es otro módulo y si requieren ideas para el mismo ofrezco apoyo)

Ej. Agenda telefónica de especialistas

CERRAR deberá revisar listas pendientes antes de cerrar. Indicar la fecha y hora que se está cerrando el sistema

7. **HERRAMIENTA REVISOR** de acceso directo en la pantalla de agenda
   - Mediante typeo o lector de código de barrar el sistema pedirá los datos de grupo familiar y leerá los datos encontrados que son: Nombre del paciente agendado y nombre del médico que lo-le atendió.
   - El cuadro permanecerá activo y tendrá las opciones de ir a la tabla de agendados si fuera necesario modificar su atención (puede que el paciente no se haya presentado en todo el día, pero solo al revisar la carp nos enteramos. Puede que se la haya cambiado de consultorio y un montón de situaciones que serán necesario cambiar)

Ej herramientas de revisión de carpetas. Cuando un código no existe el sistema informa "no esta registrado". Cuando se introducen códigos largos dice "reintentar", y esta activo hasta cerrar.

ANEXOS D: *Historias de usuario obtenidas a partir de requerimientos funcionales*

Fuente: Elaboración propia

D1: *Historia obtenida a partir del requerimiento RGN1*

H1	**Exportar datos a Excel**
Como usuarios del sistema, **queremos** exportar los datos del sistema a Excel **para** poder generar informes personalizados y gráficos en esta herramienta.	

D2: *Historia obtenida a partir del requerimiento RGN2*

H2	**Generación de informes según formato establecido por área**
Como usuarios del sistema, **queremos** generar informes según las plantillas que ya estamos acostumbrados a manejar **para** no tener inconvenientes con los formatos ya establecidos en el seguro.	

D3: *Historia obtenida a partir del requerimiento RV1*

H3	**Registro de datos generales de titulares y beneficiarios**
Como encargado de vigencia de derechos, **quiero** poder crear, leer, actualizar y eliminar los datos generales de titulares y beneficiarios **para** mantener la información siempre actualizada y precisa.	

D4: *Historia obtenida a partir del requerimiento RV2*

H4	**Registro de altas y bajas de los asegurados**
Como encargado de vigencia de derechos, quiero poder gestionar las altas y bajas de los asegurados para reflejar los cambios en la cobertura de los usuarios.	

D5: *Historia obtenida a partir del requerimiento RV3*

H5	**Inclusión de beneficiarios (esposas, hijos y padres)**
Como encargado de vigencia de derechos, **quiero** poder incluir beneficiarios, como esposas, hijos y padres, **para** asegurar que también tengan la cobertura correspondiente.	

D6: *Historia obtenida a partir del requerimiento RV4*

H6	**Escaneo y almacenaje de documentos de los asegurados**
Como encargado de vigencia de derechos, **quiero** escanear y almacenar los documentos de los asegurados **para** tener un registro digital accesible y organizado de toda la documentación.	

D7: *Historia obtenida a partir del requerimiento RV5*

H7	**Modificación y almacenamiento de documentación periódica**
Como encargado de vigencia de derechos, **quiero** modificar y almacenar documentación periódica **para** mantener actualizados los documentos requeridos y cumplir con las normativas.	

D8: *Historia obtenida a partir del requerimiento RV5*

H8	**Detector de alarmas de documentos caducados**
Como encargado de vigencia de derechos, **quiero** recibir alertas cuando los documentos de los asegurados estén próximos a caducar **para** poder solicitar actualizaciones de manera proactiva.	

D9: *Historia obtenida a partir del requerimiento RV7*

H9	**Actualización e inclusión de nuevos documentos requeridos**
Como encargado de vigencia de derechos, **quiero** poder actualizar e incluir nuevos documentos requeridos **para** asegurar el cumplimiento de las normativas vigentes.	

D10: *Historia obtenida a partir del requerimiento RV8*

H10	**Control de acceso según nivel de jerarquía de usuario**
Como encargado de vigencia de derechos, **quiero** que el sistema implemente control de acceso basado en niveles jerárquicos **para** garantizar que solo usuarios autorizados accedan a información sensible.	

D11: *Historia obtenida a partir del requerimiento RV9*

H11	**Configurar el documento asignado a cada tipo de beneficiario en el sistema**
Como encargado de vigencia de derechos, **quiero** poder configurar los tipos de documentos asignados a cada tipo de beneficiario **para** garantizar que se recabe la documentación correcta según su relación con el titular.	

D12: *Historia obtenida a partir del requerimiento RF1*

H12	**Leer credenciales con pistola láser**
Como encargado de fichaje, **quiero** poder leer las credenciales de los beneficiarios con una pistola láser **para** agilizar el proceso de registro y fichaje.	

D13: *Historia obtenida a partir del requerimiento RF2*

H13	**Búsqueda por carnet y nombre del beneficiario**
Como encargado de fichaje, **quiero** poder buscar a los beneficiarios por su carnet de identidad o nombre **para** identificar rápidamente su información en el sistema.	

D14: *Historia obtenida a partir del requerimiento RF3*

H14	**Registro de fichas emitidas**
Como encargado de fichaje, **quiero** poder crear, leer, actualizar y eliminar las fichas emitidas a los beneficiarios **para** llevar un control detallado y actualizado del proceso de fichaje.	

D15: *Historia obtenida a partir del requerimiento RF4*

H15	**Visualización de alertas**
Como encargado de fichaje, **quiero** poder visualizar alertas en el sistema **para** estar informado de situaciones que requieran mi atención, como errores en las fichas o documentos vencidos.	

D16: *Historia obtenida a partir del requerimiento RS1*

H16	**Registro de usuarios**
Como encargado de sistemas, **quiero** poder crear, leer, actualizar y eliminar usuarios en el sistema **para** gestionar el acceso y la administración de los usuarios registrados.	

D17: *Historia obtenida a partir del requerimiento RS2*

H17	**Registro de solicitudes y entrega de carnets**
Como encargado de sistemas, **quiero** registrar las solicitudes y el proceso de entrega de carnets **para** llevar un control detallado y organizado de este procedimiento.	

D18: *Historia obtenida a partir del requerimiento RG1*

H18	**Registro de cambios en horarios de atención de los médicos**
Como parte gerencial, **queremos** poder registrar los cambios en los horarios de atención de los médicos **para** asegurarnos de que los beneficiarios tengan la información actualizada y correcta.	

D19: *Historia obtenida a partir del requerimiento RG2*

H19	**Visualización de turnos según ficha en TV**
Como parte gerencial, **queremos** que los turnos de atención se muestren en pantallas de TV en las instalaciones, **para** mejorar la organización y la eficiencia en la atención de los beneficiarios.	

D20: *Historia obtenida a partir del requerimiento RG3*

H20	**Gaceta de emisión de fichas para beneficiarios**

Como parte gerencial, **queremos** contar con una gaceta que informe sobre la emisión de fichas **para** beneficiarios, de modo que podamos llevar un control claro y accesible de las citas otorgadas.

D21: *Historia obtenida a partir del requerimiento RG4*

H21	**Sitio web para agendar visitas por internet**
Como parte gerencial, **queremos** implementar un sitio web donde los beneficiarios puedan agendar sus visitas en línea, **para** mejorar la comodidad y accesibilidad en el proceso de citas.	

D21: *Historia obtenida a partir del requerimiento RGN4, RGN5 y RGN6*

H22	**Diseño y desarrollo de la arquitectura**
Como parte gerencial, **quiero** que se diseñe y desarrolle la arquitectura de la aplicación, **para** asegurar que los módulos de usuarios, beneficiarios y fichaje estén bien organizados y sean escalables. Garantizando un rendimiento óptimo y facilidad de mantenimiento a largo plazo.	

ANEXOS E: *Tabal de SLOC según funcionalidades y características para Backend (Laravel)*

Fuente: Elaboración propia

Funcionalidad	Descripción	Características	Complejidad	Estimación de SLOC
Crear (Create)	Operaciones para crear registros en la base de datos.	CRUD básico, validaciones de datos	Baja	40 - 60
		Uso de Eloquent o Query Builder	Media	60 - 100
		Transacciones opcionales	Alta	100 - 150
Leer (Read)	Consultas para obtener registros de la base de datos.	Filtros, paginación	Baja	30 - 50
		Uso de Eloquent o Query Builder	Media	50 - 80
		Joins complejos en consultas	Alta	80 - 120
Actualizar (Update)	Modificación de registros existentes en la base de datos.	CRUD básico, validaciones de datos	Baja	40 - 70
		Uso de Eloquent o Query Builder	Media	70 - 100
		Transacciones opcionales	Alta	100 - 150
Eliminar (Delete)	Eliminación de registros de la base de datos.	CRUD básico, validaciones de datos	Baja	20 - 40
		Uso de Eloquent o Query Builder	Media	40 - 60
		Soft deletes (eliminación suave)	Alta	60 - 100
Autenticación	Proceso de verificación de la identidad del usuario.	Uso de Laravel Passport o Sanctum	Media	80 - 150
		Token JWT, manejo de sesiones	Alta	150 - 300
Manejo de roles y permisos	Control de acceso a funcionalidades según roles de usuario.	Uso de middleware, gestión de políticas	Media	80 - 150
		Integración con Spatie Laravel Permission	Alta	150 - 300
Relaciones (1-N)	Manejo de relaciones uno a muchos entre entidades.	Definición de relaciones en modelos	Baja	30 - 70
		Eager loading para optimizar consultas	Media	70 - 120
		Validaciones en cascada	Alta	120 - 200
Relaciones (N-N)	Manejo de relaciones muchos a muchos entre entidades.	Uso de tablas intermedias, definiciones de relaciones	Baja	40 - 80
		Eager loading para optimizar consultas	Media	80 - 150
		Validaciones en cascada	Alta	150 - 250
Búsqueda avanzada	Implementación de funcionalidades de búsqueda compleja en la base de datos.	Búsqueda por múltiples criterios	Media	40 - 100
		Uso de Full Text Search en MySQL	Alta	100 - 150
Filtrado de datos	Posibilidad de filtrar resultados de	Filtros por múltiples atributos	Baja	40 - 80

	consultas según criterios.	Implementación de filtros complejos	Media	80 - 120
Paginación	Implementación de paginación en resultados de consultas.	Integración de paginación de Laravel	Baja	20 - 50
		Configuración de resultados por página	Media	50 - 80
Carga de archivos	Subida y gestión de archivos, incluyendo validaciones.	Validaciones de tipo y tamaño de archivos	Baja	30 - 70
		Uso de almacenamiento en la nube	Media	70 - 120
		Manejo de versiones de archivos	Alta	120 - 200
Notificaciones	Implementación de sistemas de notificación, ya sea por correo o en tiempo real.	Uso de Laravel Notifications	Media	50 - 100
		Integración con servicios de mensajería	Alta	100 - 200
Gestión de API Gateway (Kong, Apigee)	Manejo de un API Gateway para gestionar solicitudes a microservicios.	Configuración de rutas y seguridad	Alta	150 - 300
OAuth2 / Single Sign-On (SSO)	Implementación de autenticación unificada mediante OAuth2.	Uso de Laravel Passport, manejo de tokens	Alta	150 - 300
Manejo de eventos WebSockets	Implementación de WebSockets para comunicaciones en tiempo real.	Uso de Laravel Echo	Media	80 - 150
		Manejo de conexiones persistentes	Alta	150 - 250
Manejo de archivos grandes (chunking)	Implementación de carga y procesamiento de archivos grandes en fragmentos.	Uso de chunking para optimizar memoria	Alta	150 - 300
Gestión de microservicios	Diseño y gestión de una arquitectura basada en microservicios.	Uso de contenedores, orquestación	Alta	150 - 350
Test unitarios (Unit Tests)	Implementación de pruebas unitarias para garantizar la calidad del código.	Uso de PHPUnit, pruebas en aislamiento	Media	30 - 80
		Cobertura de pruebas	Alta	80 - 150
Test de integración (Integration Tests)	Pruebas que verifican la interacción entre diferentes componentes del sistema.	Uso de PHPUnit, validaciones de flujo	Media	50 - 120
		Pruebas de API y servicios	Alta	120 - 200
Automatización CI/CD (Jenkins, GitLab CI)	Implementación de pipelines de integración y entrega continua.	Uso de herramientas de automatización	Media	100 - 200
		Pruebas automáticas, despliegues continuos	Alta	200 - 400
Procesamiento asíncrono (jobs en background)	Ejecución de tareas en segundo plano utilizando Laravel Queues.	Manejo de tareas programadas	Media	50 - 100
		Uso de Redis o otros backends	Alta	100 - 200

Notificaciones en tiempo real	Implementación de notificaciones en tiempo real usando WebSockets (Pusher, etc.).	Conexiones persistentes, actualizaciones en tiempo real	Media	200 - 400
Gestión de archivos	Subida, procesamiento y almacenamiento de archivos grandes (imágenes, videos).	Uso de almacenamiento en la nube	Media	200 - 300

ANEXOS F: *Tabal de SLOC según funcionalidades y características para Frontend (ReactJs)*

Fuente: Elaboración propia

ANEXOS G: *Análisis desglosado de cálculo de SLOC según micro-funcionalidades de las historias de usuario*

Fuente: Elaboración propia

G1: *Calculo de SLOC de la historia H1*

H1	**Exportar datos a Excel**	**SLOC: 100**
Funcionalidades Backend		SLOC
Total SLOC Backend		**0**
Funcionalidades Frontend		SLOC
HF1.1	Función	100
Total SLOC Frontend		**100**

G2: *Calculo de SLOC de la historia H2*

H2	**Generación de informes según formato establecido por área**	**SLOC: 1295**
Funcionalidades Backend		SLOC
HB2.1	API de consulta 120 LOC x 7 Reportes	840
Total SLOC Backend		**840**
Funcionalidades Frontend		SLOC
HF2.1	Botón de consulta 5 LOC x 7 Reportes	35
HF2.2	Manejo de peticiones HTTP 60 SLOC x 7 Reportes	420

Total SLOC Frontend	**455**

G3: *Calculo de SLOC de la historia H3*

H3	**Registro de datos generales de titulares y beneficiarios**	**SLOC: 960**
Funcionalidades Backend		SLOC
HB3.1	API de adición	60
HB3.2	API de lectura	50
HB3.3	API de edición	70
HB3.4	API de borrado	40
Total SLOC Backend		**220**
Funcionalidades Frontend		SLOC
HF3.1	Formulario de adición	120
HF3.2	Tabla para listar los datos	150
HF3.3	Formulario de edición	150
HF3.4	Formulario de borrado	80
HF3.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240
Total SLOC Frontend		**740**

G4: *Calculo de SLOC de la historia H4*

H4	**Registro de altas y bajas de los asegurados**	**SLOC: 750**
Funcionalidades Backend		SLOC
HB4.1	API de adición	60
HB4.2	API de lectura	50
HB4.3	API de edición	70
HB4.4	API de borrado	40

Total SLOC Backend		**220**
Funcionalidades Frontend		SLOC
HF4.1	Formulario de adición	120
HF4.3	Formulario de edición	150
HF4.4	Formulario de borrado	80
HF4.5	Manejo de peticiones HTTP 60 SLOC x 3 FNs	180
Total SLOC Frontend		**530**

G5: *Calculo de SLOC de la historia H5*

H5	**Inclusión de beneficiarios (esposas, hijos y padres)**	**SLOC: 960**
Funcionalidades Backend		SLOC
HB5.1	API de adición	60
HB5.2	API de lectura	50
HB5.3	API de edición	70
HB5.4	API de borrado	40
Total SLOC Backend		**220**
Funcionalidades Frontend		SLOC
HF5.1	Formulario de adición	120
HF5.2	Tabla para listar los datos	150
HF5.3	Formulario de edición	150
HF5.4	Formulario de borrado	80
HF6.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240
Total SLOC Frontend		**740**

G6: *Calculo de SLOC de la historia H6*

H6	**Escaneo y almacenaje de documentos de los asegurados**	**SLOC: 1040**

Funcionalidades Backend		SLOC
HB6.1	Manejo de archivos grandes (chunking)	300
Total SLOC Backend		**300**
Funcionalidades Frontend		SLOC
HF6.1	Formulario de adición	120
HF6.2	Tabla para listar los datos	150
HF6.3	Formulario de edición	150
HF6.4	Formulario de borrado	80
HF6.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240
Total SLOC Frontend		**740**

G7: *Calculo de SLOC de la historia H7*

H7	**Modificación y almacenamiento de documentación periódica**	**SLOC: 430**
Funcionalidades Backend		SLOC
HB7.1	Manejo de archivos grandes (chunking)	200
Total SLOC Backend		**200**
Funcionalidades Frontend		SLOC
HF7.1	Tabla para listar los datos	150
HF7.2	Formulario de borrado	80
Total SLOC Frontend		**230**

G8: *Calculo de SLOC de la historia H8*

H8	Detector de alarmas de documentos caducados	**SLOC: 610**
Funcionalidades Backend		SLOC
HB8.1	API de adición	60

HB8.2	API de lectura	50
HB8.3	API de edición	70
HB8.4	API de borrado	40
HB8.5	Procesamiento asíncrono (jobs en background)	80
HB8.6	Notificaciones	80
Total SLOC Backend		**380**
Funcionalidades Frontend		SLOC
HF8.1	Implementación de notificaciones	80
HF8.2	Tabla para listar los datos	150
Total SLOC Frontend		**230**

G9: *Calculo de LOC de la historia H9*

H9	**Actualización e inclusión de nuevos documentos requeridos**	**SLOC: 1020**
Funcionalidades Backend		SLOC
HB9.1	API de adición	60
HB9.2	API de lectura	50
HB9.3	API de edición	70
HB9.4	API de borrado	40
HB9.5	Relaciones (N-N)	60
Total SLOC Backend		**280**
Funcionalidades Frontend		SLOC
HF9.1	Formulario de adición	120
HF9.2	Tabla para listar los datos	150
HF9.3	Formulario de edición	150
HF9.4	Formulario de borrado	80

HF9.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240
Total SLOC Frontend		**740**

G10: *Calculo de SLOC de la historia H10*

H10	**Control de acceso según nivel de jerarquía de usuario**	**SLOC: 1200**
Funcionalidades Backend		SLOC
HB10.1	Gestión de microservicios	350
HB10.2	Gestión de API Gateway (Kong, Apigee)	200
HB10.3	Autenticación	250
Total SLOC Backend		**800**
Funcionalidades Frontend		SLOC
HF10.1	Autenticación y autorización	400
Total SLOC Frontend		**400**

G11: *Calculo de SLOC de la historia H11*

H11	**Configurar el documento asignado a cada tipo de beneficiario en el sistema**	**SLOC: 340**
Funcionalidades Backend		SLOC
HB11.1	API de edición	70
HB11.2	Relaciones (N-N)	60
Total LOC Backend		**130**
Funcionalidades Frontend		SLOC
HF11.1	Formulario de edición	150
HF11.2	Manejo de peticiones HTTP 60 SLOC x 1 FNs	60
Total SLOC Frontend		**210**

G12: *Calculo de SLOC de la historia H12*

H12	**Leer credenciales con pistola láser**	**SLOC: 120**
Funcionalidades Backend		SLOC
Total SLOC Backend		**0**
Funcionalidades Frontend		SLOC
HF12.1	Función	120
Total SLOC Frontend		**120**

G13: *Calculo de SLOC de la historia H13*

H13	**Búsqueda por carnet y nombre del beneficiario**	**SLOC: 610**
Funcionalidades Backend		SLOC
HB11.2	Búsqueda avanzada	150
Total SLOC Backend		**150**
Funcionalidades Frontend		SLOC
HF13.1	Formulario de búsqueda	250
HF13.2	Tabla para listar los datos	150
HF13.3	Manejo de peticiones HTTP 60 SLOC x 1 FNs	60
Total SLOC Frontend		**460**

G14: *Calculo de SLOC de la historia H14*

H14	**Registro de fichas emitidas**	**SLOC: 990**
Funcionalidades Backend		SLOC
HB14.1	API de adición	60
HB14.2	API de lectura	50
HB14.3	API de edición	70
HB14.4	API de borrado	40

Total SLOC Backend		**220**
Funcionalidades Frontend		SLOC
HF14.1	Formulario de adición	150
HF14.2	Tabla para listar los datos	150
HF14.3	Formulario de edición	150
HF14.4	Formulario de borrado	80
HF14.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240
Total SLOC Frontend		**770**

G15: *Calculo deS LOC de la historia H15*

H15	**Visualización de alertas**	**SLOC: 840**
Funcionalidades Backend		SLOC
HB15.1	API de adición	60
HB15.2	API de lectura	50
HB15.3	API de borrado	40
HB15.4	Notificaciones	120
Total SLOC Backend		**270**
Funcionalidades Frontend		SLOC
HF15.1	Tabla para listar los datos	150
HF15.2	Formulario de borrado	80
HB15.3	Manejo de WebSockets	100
HB15.4	Implementación de notificaciones	120
HF15.5	Manejo de peticiones HTTP 60 LOC x 2 FNs	120
Total SLOC Frontend		**570**

G16: *Calculo de SLOC de la historia H16*

H16	**Registro de usuarios**	**LOC: 990**
Funcionalidades Backend		SLOC
HB16.1	API de adición	60
HB16.2	API de lectura	50
HB16.3	API de edición	70
HB16.4	API de borrado	40
Total SLOC Backend		**220**
Funcionalidades Frontend		SLOC
HF16.1	Formulario de adición	150
HF16.2	Tabla para listar los datos	150
HF16.3	Formulario de edición	150
HF16.4	Formulario de borrado	80
HF16.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240
Total SLOC Frontend		**770**

G17: *Calculo deS LOC de la historia H17*

H17	**Registro de solicitudes y entrega de carnets**	**SLOC: 990**
Funcionalidades Backend		SLOC
HB17.1	API de adición	60
HB17.2	API de lectura	50
HB17.3	API de edición	70
HB17.4	API de borrado	40
Total SLOC Backend		**220**

Funcionalidades Frontend		SLOC
HF17.1	Formulario de adición	150
HF17.2	Tabla para listar los datos	150
HF17.3	Formulario de edición	150
HF17.4	Formulario de borrado	80
HF17.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240
Total SLOC Frontend		**770**

G18: *Calculo de SLOC de la historia H18*

H18	**Registro de cambios en horarios de atención de los médicos**	**SLOC: 1190**
Funcionalidades Backend		SLOC
HB18.1	API de adición	60
HB18.2	API de lectura	50
HB18.3	API de edición	70
HB18.4	API de borrado	40
HB18.5	Relaciones (N-N)	200
Total SLOC Backend		**420**
Funcionalidades Frontend		SLOC
HF18.1	Formulario de adición	150
HF18.2	Tabla para listar los datos	150
HF18.3	Formulario de edición	150
HF18.4	Formulario de borrado	80
HF18.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240
Total SLOC Frontend		**770**

G19: *Calculo de SLOC de la historia H19*

H19	**Visualización de turnos según ficha en TV**	**SLOC: 930**
Funcionalidades Backend		SLOC
HB19.1	Manejo de eventos WebSockets	200
HB19.2	Notificaciones en tiempo real	250
HB19.3	API de borrado	40
Total SLOC Backend		**220**
Funcionalidades Frontend		SLOC
HF19.1	Componentes dinámicos	300
HF19.2	Tabla para listar los datos	150
HF19.3	Implementación de notificaciones	120
HF19.4	Formulario de borrado	80
HF19.5	Manejo de peticiones HTTP 60 SLOC x 1 FNs	60
Total SLOC Frontend		**710**

G20: *Calculo de SLOC de la historia H20*

H20	**Gaceta de emisión de fichas para beneficiarios**	**SLOC: 770**
Funcionalidades Backend		LOC
Total SLOC Backend		**0**
Funcionalidades Frontend		LOC
HF20.1	Formulario de adición	150
HF20.2	Tabla para listar los datos	150
HF20.3	Formulario de edición	150
HF20.4	Formulario de borrado	80
HF20.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	240

Total SLOC Frontend	**770**

G21: *Calculo de SLOC de la historia H21*

H21	**Sitio web para agendar visitas por internet**	**SLOC: 1640**
Funcionalidades Backend		LOC
HB21.1	API de adición	60
HB21.2	API de lectura	50
HB21.3	API de edición	70
HB21.4	API de borrado	40
Total SLOC Backend		**220**
Funcionalidades Frontend		SLOC
HF21.1	Formulario de adición	300
HF21.2	Tabla para listar los datos	300
HF21.3	Formulario de edición	300
HF21.4	Formulario de borrado	230
HF21.5	Manejo de peticiones HTTP 60 SLOC x 4 FNs	290
Total SLOC Frontend		**1420**

G22: *Calculo de SLOC de la historia H22*

H22	**Diseño y desarrollo de la arquitectura**	**SLOC: 490**
Funcionalidades Backend		LOC
HB22.1	Diseño de bases de datos para usuarios	80
HB22.2	Diseño de bases de datos para beneficiarios	80
HB22.3	Diseño de bases de datos para fichaje	80
HB22.4	Configuración de Docker para los módulos	100

HB22.5	Implementación en Kubernetes (manifiestos, servicios)	100
HB22.6	Implementación de Kong como api gateway	50
Total SLOC Backend		**490**
Funcionalidades Frontend		SLOC
Total SLOC Frontend		**0**

ANEXOS H: *Tabla guía para calcular el factor de influencia*

Fuente: Elaboración propia

Factor de Influencia	Descripción
Comunicación de datos	**0:** No se requiere comunicación. **5:** El sistema debe manejar grandes volúmenes de datos de múltiples fuentes.
Funciones distribuidas	**0:** El sistema es completamente local. **5:** El sistema se ejecuta en un entorno distribuido con múltiples servidores.
Rendimiento	**0:** No hay requisitos de rendimiento específicos. **5:** Se requieren tiempos de respuesta muy rápidos, incluso bajo carga alta.
Eficiencia de usuario final	**0:** Interfaz básica. **5:** La interfaz debe ser altamente intuitiva y permitir a los usuarios completar tareas de forma rápida.
Procesamiento complejo	**0:** Procesos simples. **5:** Requiere algoritmos complejos o procesamiento intensivo (ej. procesamiento de imágenes o cálculos matemáticos complejos).
Reusabilidad del código	**0:** No se requiere reusabilidad. **5:** El sistema debe permitir el uso de componentes en diferentes módulos o proyectos.
Facilidad de instalación	**0:** Instalación complicada. **5:** Instalación completamente automática y sin intervención del usuario.
Facilidad operativa	**0:** Muy difícil de operar. **5:** El sistema debe ser fácil de operar con un mínimo de entrenamiento.
Multiplataforma	**0:** Solo se ejecuta en un sistema operativo específico. **5:** El sistema debe funcionar en múltiples plataformas (Windows, Linux, macOS, etc.).
Transacciones automáticas	**0:** No se requieren transacciones automáticas. **5:** Debe manejar miles de transacciones simultáneas sin error.
Facilidad de cambio	**0:** No es necesario realizar cambios. **5:** El sistema debe adaptarse fácilmente a cambios frecuentes en los requisitos.
Facilidad de uso	**0:** Dificultad extrema para los usuarios. **5:** Extremadamente fácil de usar, incluso para usuarios no técnicos.

Seguridad	**0:** Sin requisitos de seguridad. **5:** Requiere altos niveles de seguridad, como cifrado y autenticación multifactor.
Portabilidad	**0:** No se requiere portabilidad. **5:** El sistema debe ser capaz de ejecutarse en diversos entornos de hardware y software.

ANEXOS I: *Criterios de aceptación por historias de usuarios*

Fuente: Elaboración propia

I1: *Criterios de aceptación de la historia H1*

H1	Exportar datos a Excel
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir la exportación de datos a Excel. - Los datos exportados deben incluir todos los campos relevantes. - Debe haber una opción para seleccionar el rango de datos a exportar.
Confiabilidad	- La exportación debe completarse sin errores. - Los archivos generados deben ser legibles y compatibles con Excel.
Usabilidad	- La opción de exportación debe ser accesible desde el menú principal. - Debe haber un mensaje de confirmación después de la exportación exitosa.
Eficiencia	- La exportación debe completarse en menos de 25 segundos para conjuntos de datos grandes. - El rendimiento del sistema no debe verse afectado durante la exportación.
Mantenibilidad	- El código relacionado con la exportación debe ser modular y fácil de actualizar.
Portabilidad	- Los archivos exportados deben ser compatibles con múltiples versiones de Excel.

I2: *Criterios de aceptación de la historia H2*

H2	Generación de informes según formato establecido por área
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe generar informes en formatos predefinidos. - Los informes deben incluir datos específicos por área.
Confiabilidad	- Los informes generados deben ser exactos y sin errores. - Los informes deben ser auditables para verificar la integridad de los datos.

Usabilidad	- La interfaz para generar informes debe ser intuitiva y fácil de usar.
Eficiencia	- La generación de informes debe realizarse en menos de 15 segundos.
Mantenibilidad	- El formato de informes deben ser PDF. - La lógica de generación de informes debe ser modular.
Portabilidad	- Los informes deben poder abrirse en diferentes aplicaciones de procesamiento de texto.

I3: *Criterios de aceptación de la historia H3*

H3	Registro de datos generales de titulares y beneficiarios
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir crear, leer, actualizar y eliminar datos de titulares y beneficiarios. - Debe validarse que los datos ingresados cumplan con las reglas establecidas.
Confiabilidad	- Las operaciones CRUD deben completarse sin errores y ser reversibles. - Los datos deben persistir en la base de datos después de las operaciones.
Usabilidad	- La interfaz debe ser fácil de navegar para realizar operaciones CRUD. - Debe haber mensajes de error claros para entradas inválidas.
Eficiencia	- Las operaciones CRUD deben completarse en menos de 5 segundos. - El sistema debe manejar múltiples solicitudes CRUD simultáneamente.
Mantenibilidad	- El código para las operaciones CRUD debe estar bien documentado. - Las estructuras de datos deben ser fácilmente modificables.
Portabilidad	- La lógica de CRUD debe ser compatible con futuras extensiones del sistema.

I4: *Criterios de aceptación de la historia H4*

H4	Registro de altas y bajas de los asegurados
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir el alta y baja de asegurados de manera sencilla. - Debe haber validaciones para evitar bajas incorrectas. - Se debe registrar la fecha y el motivo del alta/baja.
Confiabilidad	- Los registros de altas y bajas deben ser precisos y sin inconsistencias.

	- Las operaciones de alta y baja deben ser reversibles.
Usabilidad	- La interfaz de alta y baja debe ser fácil de usar.
Eficiencia	- Las altas y bajas deben procesarse en menos de 5 segundos. - El sistema debe manejar múltiples solicitudes simultáneamente.
Mantenibilidad	- Los procesos de alta y baja deben ser fácilmente modificables.
Portabilidad	- La lógica de CRUD debe ser compatible con futuras extensiones del sistema.

I5: *Criterios de aceptación de la historia H5*

H5	Inclusión de beneficiarios (esposas, hijos y padres)
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir la inclusión de beneficiarios como esposas, hijos y padres. - Debe existir validaciones para evitar duplicados en la inclusión. - Se debe registrar la relación de cada beneficiario con el titular.
Confiabilidad	- La inclusión debe ser precisa y sin errores. - Los beneficiarios incluidos deben ser accesibles para futuras consultas.
Usabilidad	- La interfaz de inclusión debe ser clara y fácil de usar. - Deben existir mensajes claros sobre el estado de la inclusión.
Eficiencia	- La inclusión de beneficiarios debe completarse en menos de 5 segundos. - El sistema debe manejar múltiples inclusiones simultáneamente.
Mantenibilidad	- La lógica para la inclusión debe ser modular y fácil de mantener. - Los campos de entrada deben ser fácilmente actualizables.
Portabilidad	- La lógica de CRUD debe ser compatible con futuras extensiones del sistema.

I6: *Criterios de aceptación de la historia H6*

H6	Escaneo y almacenaje de documentos de los asegurados
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir el escaneo y almacenamiento de documentos.

	- Los documentos deben estar organizados por tipo y fecha.
Confiabilidad	- Los documentos escaneados deben ser almacenados sin pérdida de calidad. - La recuperación de documentos debe ser rápida y sin errores.
Usabilidad	- La interfaz para escanear y almacenar debe ser intuitiva. - Los usuarios deben recibir confirmaciones después del escaneo.
Eficiencia	- El proceso de escaneo y almacenamiento debe completarse en menos de 5 segundos por documento.
Mantenibilidad	- El sistema de almacenamiento debe ser fácilmente ajustable.
Portabilidad	- Los documentos escaneados deben ser accesibles desde diferentes plataformas. - Deben ser compatibles con aplicaciones de visualización de documentos.

I7: *Criterios de aceptación de la historia H7*

H7	Modificación y almacenamiento de documentación periódica
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir la modificación y almacenamiento de documentos periódicos.
Confiabilidad	- Las modificaciones deben ser registradas con fecha y usuario. - El almacenamiento debe ser seguro y sin pérdidas de información.
Usabilidad	- La interfaz de modificación debe ser clara y fácil de navegar. - Los cambios deben reflejarse inmediatamente en el sistema.
Eficiencia	- La modificación de documentos debe completarse en menos de 5 segundos. - El sistema debe manejar múltiples modificaciones simultáneamente.
Mantenibilidad	- Los documentos deben ser fácilmente actualizables y configurables. - La lógica de almacenamiento debe ser modular y flexible.
Portabilidad	- Las modificaciones deben ser accesibles desde diversas plataformas.

I8: *Criterios de aceptación de la historia H8*

H8	Detector de alarmas de documentos caducados
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe detectar y alertar sobre documentos caducados. - Debe existir un registro de las fechas de vencimiento de los documentos. - Se deben enviar notificaciones automáticas sobre los documentos próximos a caducar.
Confiabilidad	- Las alertas deben ser precisas y oportunas. - El sistema debe manejar correctamente excepciones en los registros.
Usabilidad	- Las alertas deben ser visibles y fácilmente accesibles para los usuarios. - Debe haber una interfaz para gestionar las alertas.
Eficiencia	- La detección de documentos caducados debe realizarse en menos de 5 segundos.
Mantenibilidad	- La lógica de detección debe ser flexible y fácil de actualizar.
Portabilidad	- Las alertas deben ser accesibles desde diferentes dispositivos.

I9: *Criterios de aceptación de la historia H9*

H9	Actualización e inclusión de nuevos documentos requeridos
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir la inclusión de nuevos tipos de documentos. - Los documentos requeridos deben poder actualizarse según normativa.
Confiabilidad	- Las actualizaciones de documentos deben reflejarse de manera precisa en todo el sistema.
Usabilidad	- La interfaz para actualizar e incluir documentos debe ser fácil de usar y entender. - Los usuarios deben recibir confirmaciones de cambios exitosos.
Eficiencia	- Las actualizaciones deben reflejarse en menos de 5 segundos en todas las áreas del sistema. - La carga de nuevos documentos no debe afectar el rendimiento del sistema.
Mantenibilidad	- Los formatos y tipos de documentos deben ser fácilmente actualizables. - El código para la gestión de documentos debe ser modular y fácil de modificar.
Portabilidad	- La función de actualización debe funcionar en múltiples plataformas y dispositivos.

I10: *Criterios de aceptación de la historia H10*

H10	Control de acceso según nivel de jerarquía de usuario
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir asignar niveles de acceso basados en jerarquías de usuario. - Los usuarios deben tener acceso solo a las funciones y datos autorizados por su nivel. - Los niveles de acceso deben ser configurables por los administradores.
Confiabilidad	- El control de acceso debe ser fiable, impidiendo accesos no autorizados.
Usabilidad	- La interfaz de administración de jerarquías debe ser fácil de usar y comprender. - Los usuarios deben ser notificados si intentan acceder a algo sin permisos.
Eficiencia	- El sistema debe verificar el nivel de acceso de un usuario en menos de 2 segundos. - Los controles de acceso no deben afectar el rendimiento del sistema.
Mantenibilidad	- Los niveles de acceso deben ser fáciles de actualizar según nuevas políticas. - Las reglas de jerarquía deben estar bien documentadas y ser modificables sin afectar el código principal.
Portabilidad	- Los niveles de acceso deben mantenerse sin importar el dispositivo o plataforma utilizada. - El control de acceso debe integrarse fácilmente con futuras extensiones del sistema.

I11: *Criterios de aceptación de la historia H11*

H11	Configurar el documento asignado a cada tipo de beneficiario en el sistema
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir asignar un tipo de documento específico a cada tipo de beneficiario. - Los documentos asignados deben ser configurables según políticas de la organización.
Confiabilidad	- Los documentos asignados deben ser accesibles de manera precisa. - El sistema debe garantizar que los documentos no se asignen incorrectamente.
Usabilidad	- La interfaz de configuración debe ser clara y fácil de usar. - Debe haber mensajes de confirmación tras la asignación exitosa de documentos.
Eficiencia	- La asignación de documentos debe realizarse en menos de 5 segundos.
Mantenibilidad	- Las asignaciones deben ser fácilmente actualizables según nuevos requerimientos.

	- El código de asignación de documentos debe estar bien estructurado y ser fácilmente modificable.
Portabilidad	- La asignación de documentos debe funcionar correctamente en diversas plataformas. - El sistema debe ser compatible con futuros tipos de documentos que puedan surgir.

I12: *Criterios de aceptación de la historia H12*

H12	Leer credenciales con pistola láser
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe poder leer credenciales con una pistola láser. - Debe validar automáticamente las credenciales escaneadas. - La pistola debe ser compatible con el sistema sin necesidad de configuraciones adicionales.
Confiabilidad	- La lectura de las credenciales debe ser precisa y sin errores. - Los datos escaneados deben ser correctamente almacenados en el sistema.
Usabilidad	- La interfaz debe ser fácil de usar y las instrucciones para el uso de la pistola deben estar claras. - Debe haber retroalimentación visual y auditiva para confirmar la lectura exitosa.
Eficiencia	- Las credenciales deben ser leídas en menos de 1 segundo. - La lectura de múltiples credenciales no debe ralentizar el sistema.
Mantenibilidad	- La compatibilidad con diferentes modelos de pistola láser debe ser asegurada fácilmente. - Las configuraciones de la pistola deben ser actualizables sin afectar otras partes del sistema.
Portabilidad	- El sistema debe funcionar con varios tipos de lectores de credenciales. - La funcionalidad debe estar disponible en diferentes plataformas.

I13: *Criterios de aceptación de la historia H3*

H13	Búsqueda por carnet y nombre del beneficiario
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir la búsqueda de beneficiarios por carnet y nombre. - La búsqueda debe ser precisa y permitir filtros por otros parámetros adicionales. - Los resultados deben ser mostrados en tiempo real.

Confiabilidad	- Los resultados de la búsqueda deben ser exactos y relevantes. - El sistema debe manejar errores de entrada y mostrar mensajes claros.
Usabilidad	- La interfaz de búsqueda debe ser intuitiva y de fácil acceso. - Los resultados deben ser presentados de forma clara y ordenada.
Eficiencia	- La búsqueda debe devolver resultados en menos de 5 segundos. - El sistema debe manejar búsquedas simultáneas sin afectar el rendimiento.
Mantenibilidad	- La funcionalidad de búsqueda debe ser fácil de ajustar para agregar nuevos parámetros. - El código de búsqueda debe ser modular y fácilmente modificable.
Portabilidad	- La búsqueda debe ser accesible desde diversos dispositivos y plataformas. - La funcionalidad debe integrarse con futuros módulos de búsqueda avanzada.

I14: *Criterios de aceptación de la historia H14*

H14	Registro de fichas emitidas
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir crear, leer, actualizar y eliminar fichas emitidas. - Las fichas deben incluir información relevante de los beneficiarios.
Confiabilidad	- Las operaciones CRUD deben ser precisas y sin errores. - El sistema debe garantizar la integridad de los datos de las fichas.
Usabilidad	- La interfaz de gestión de fichas debe ser clara y fácil de navegar. - Los usuarios deben recibir mensajes de confirmación para las acciones CRUD.
Eficiencia	- Las operaciones CRUD deben completarse en menos de 5 segundos. - El sistema debe manejar múltiples solicitudes CRUD sin afectar el rendimiento.
Mantenibilidad	- La lógica de gestión de fichas debe ser modular y fácil de actualizar. - El código debe estar bien documentado para facilitar modificaciones.
Portabilidad	- Las operaciones CRUD deben funcionar correctamente en múltiples plataformas. - Las fichas deben ser accesibles desde cualquier dispositivo.

I15: *Criterios de aceptación de la historia H15*

H15	Visualización de alertas
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir la visualización de alertas para eventos críticos. - Las alertas deben actualizarse en tiempo real.
Confiabilidad	- Las alertas deben mostrarse sin errores y de manera oportuna. - El sistema debe garantizar que las alertas se mantengan hasta ser gestionadas.
Usabilidad	- Las alertas deben ser claramente visibles en la interfaz de usuario. - Los usuarios deben recibir notificaciones cuando ocurra una nueva alerta.
Eficiencia	- Las alertas deben aparecer en menos de 1 segundo después de ser generadas. - El sistema debe poder manejar múltiples alertas sin disminuir su rendimiento.
Mantenibilidad	- La lógica de alertas debe ser fácil de ajustar y configurar. - Los tipos de alertas deben poder ser modificados según los requisitos del negocio.
Portabilidad	- Las alertas deben ser visibles desde cualquier dispositivo y plataforma. - Las notificaciones de alertas deben enviarse a través de diferentes canales.

I16: *Criterios de aceptación de la historia H16*

H16	Registro de usuarios
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir la definición del nivel de acceso para cada usuario. - Los usuarios deben tener acceso a funcionalidades según su nivel de jerarquía. - Los administradores deben poder modificar fácilmente los niveles de acceso.
Confiabilidad	- El sistema debe prevenir accesos no autorizados de usuarios.
Usabilidad	- La interfaz para definir niveles de acceso debe ser intuitiva y fácil de usar. - Los usuarios deben ser notificados si se les asigna un nuevo nivel de acceso.
Eficiencia	- La asignación de niveles de acceso debe reflejarse en menos de 2 segundos. - El sistema debe manejar múltiples cambios de nivel sin afectar su rendimiento.
Mantenibilidad	- Los niveles de acceso deben ser fácilmente ajustables conforme cambien las políticas de seguridad.

	- La lógica para la definición de niveles debe ser modular y fácilmente actualizable.
Portabilidad	- Los niveles de acceso deben ser compatibles en todos los dispositivos y plataformas. - Las modificaciones a los niveles de acceso deben reflejarse en tiempo real en todas las plataformas.

I17: *Criterios de aceptación de la historia H17*

H17	Registro de solicitudes y entrega de carnets
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir registrar las solicitudes de carnets. - Debe permitir registrar la entrega de carnets a los beneficiarios. - Se debe mantener un historial de las solicitudes y entregas realizadas.
Confiabilidad	- Las solicitudes y entregas deben ser registradas sin errores. - El sistema debe garantizar la integridad de los datos durante el registro.
Usabilidad	- La interfaz de registro de solicitudes y entregas debe ser fácil de usar.
Eficiencia	- Las solicitudes y entregas deben ser registradas en menos de 3 segundos. - El sistema debe manejar múltiples solicitudes y entregas sin problemas de rendimiento.
Mantenibilidad	- La lógica de registro de solicitudes y entregas debe ser modular y fácil de actualizar. - Las configuraciones de solicitudes deben ser ajustables según nuevas políticas.
Portabilidad	- El registro de solicitudes y entregas debe ser accesible desde cualquier dispositivo. - La funcionalidad debe ser compatible con futuras extensiones del sistema de gestión de carnets.

I18: *Criterios de aceptación de la historia H18*

H18	Registro de cambios en horarios de atención de los médicos
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir registrar cambios en los horarios de atención de los médicos. - Los horarios actualizados deben reflejarse en todas las áreas del sistema.
Confiabilidad	- Los cambios de horario deben registrarse sin errores. - El sistema debe garantizar que los horarios actualizados se muestren de manera correcta.

Usabilidad	- La interfaz para registrar cambios de horario debe ser intuitiva y fácil de usar. - Los usuarios deben ser notificados si hay cambios en el horario de su médico.
Eficiencia	- Los cambios de horario deben reflejarse en el sistema en menos de 5 segundos. - El sistema debe manejar múltiples cambios de horario sin problemas de rendimiento.
Mantenibilidad	- La lógica de horarios debe ser modular y fácil de actualizar. - Las configuraciones de horarios deben ser ajustables según las políticas internas.
Portabilidad	- Los cambios de horario deben reflejarse correctamente en cualquier dispositivo o plataforma. - La funcionalidad debe ser compatible con futuras integraciones de calendario o agenda.

I19: *Criterios de aceptación de la historia H19*

H19	Visualización de turnos según ficha en TV
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir la visualización de turnos asignados en una TV en el establecimiento. - Los turnos deben actualizarse en tiempo real conforme se emitan nuevas fichas.
Confiabilidad	- La visualización de turnos debe ser precisa y sin errores. - El sistema debe garantizar que los turnos mostrados correspondan con el orden correcto de atención.
Usabilidad	- La interfaz de visualización en TV debe ser clara y legible a distancia. - Los turnos deben estar organizados de manera intuitiva para los usuarios.
Eficiencia	- Los turnos deben actualizarse en menos de 2 segundos después de cada emisión. - El sistema debe poder manejar múltiples actualizaciones de turnos sin afectar su rendimiento.
Mantenibilidad	- La lógica de visualización de turnos debe ser fácilmente ajustable. - La interfaz de visualización debe ser fácilmente modificable según nuevos requisitos.
Portabilidad	- La visualización de turnos debe ser compatible con diversas marcas y modelos de TV. - La funcionalidad debe ser accesible también en dispositivos móviles y ordenadores.

I20: *Criterios de aceptación de la historia H20*

H20	Gaceta de emisión de fichas para beneficiarios
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe generar una gaceta de emisión de fichas para los beneficiarios. - La gaceta debe incluir información relevante sobre las fichas emitidas. - Los beneficiarios deben poder consultar la gaceta en tiempo real.
Confiabilidad	- La gaceta debe generarse de manera precisa y sin errores. - El sistema debe garantizar la exactitud de los datos presentados en la gaceta.
Usabilidad	- La gaceta debe ser clara y fácil de entender por los beneficiarios. - Los beneficiarios deben poder acceder a la gaceta sin dificultades técnicas.
Eficiencia	- La gaceta debe generarse en menos de 5 segundos después de la emisión de cada ficha. - El sistema debe manejar múltiples emisiones de fichas sin afectar el rendimiento.
Mantenibilidad	- La lógica de generación de la gaceta debe ser modular y fácilmente actualizable. - La configuración de la gaceta debe ser ajustable según nuevos requerimientos.
Portabilidad	- La gaceta debe ser accesible desde cualquier dispositivo o plataforma. - La funcionalidad debe ser compatible con futuras integraciones de software de gestión de fichas.

I21: *Criterios de aceptación de la historia H21*

H21	Sitio web para agendar visitas por internet
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe permitir a los beneficiarios agendar visitas por internet. - El sistema debe enviar confirmaciones automáticas de las visitas agendadas. - Los usuarios deben poder modificar o cancelar sus visitas agendadas.
Confiabilidad	- El sistema debe garantizar que las visitas agendadas se registren de manera precisa. - Las cancelaciones y modificaciones deben reflejarse correctamente en el sistema.
Usabilidad	- La interfaz para agendar visitas debe ser clara y fácil de usar. - Los beneficiarios deben recibir instrucciones claras para modificar o cancelar visitas.
Eficiencia	- La reserva de visitas debe completarse en menos de 2 minutos.

	- El sistema debe manejar múltiples solicitudes de reserva simultáneas sin problemas.
Mantenibilidad	- La funcionalidad de agendamiento debe ser fácilmente actualizable. - Los parámetros de agendamiento deben poder ajustarse según los requerimientos de la organización.
Portabilidad	- El sitio web debe ser accesible desde cualquier dispositivo o plataforma. - La funcionalidad de agendamiento debe integrarse con futuros módulos de gestión de visitas.

I22: *Criterios de aceptación de la historia H22*

H22	Diseño y desarrollo de la arquitectura
Dimensión	**Criterio de aceptación**
Funcionalidad	- El sistema debe incluir bases de datos para los módulos de usuarios, beneficiarios y fichaje. - La configuración de Docker debe permitir la fácil implementación de cada módulo. - La arquitectura debe permitir el escalado de los módulos en el futuro.
Confiabilidad	- Las bases de datos deben garantizar la integridad y consistencia de los datos. - Las configuraciones de Docker y Kubernetes deben ser robustas y recuperables en caso de fallo.
Usabilidad	- La estructura de la arquitectura debe facilitar la implementación de nuevas funcionalidades.
Eficiencia	- La configuración de Docker y Kubernetes debe permitir la implementación en menos de 10 minutos. - El sistema debe manejar múltiples solicitudes de manera eficiente sin degradar el rendimiento.
Mantenibilidad	- La arquitectura debe ser fácilmente modificable para adaptarse a cambios en los requisitos. - La estructura de las bases de datos debe facilitar la actualización y mantenimiento sin tiempo de inactividad.
Portabilidad	- La infraestructura debe ser compatible con diferentes entornos de despliegue (local, staging, producción). - Las configuraciones de Docker y Kubernetes deben permitir la migración entre diferentes servidores o nubes.

ANEXOS J: *Resultados del Sprint 1*

J3: *Manifiesto Deployment de Kubernetes para despliegue de Kong.*

Fuente: Elaboración propia

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kong
  labels:
    name: kong
spec:
  replicas: 1
  selector:
    matchLabels:
      name: kong
  template:
    metadata:
      name: kong
      labels:
        name: kong
    spec:
      containers:
        - name: kong-db
          image: postgres:15.0
          env:
          - name: POSTGRES_DB
            value: kong
          - name: POSTGRES_PASSWORD
            value: PASSWPRD
          - name: POSTGRES_USER
            value: sagitario
          ports:
            - containerPort: 5432
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
              name: kong-pvc
              subPath: kongdb

        - name: kong
          image: kong:latest
          ports:
            - containerPort: 8000
            - containerPort: 8001
            - containerPort: 8443
          env:
            - name: KONG_DATABASE
              value: postgres
            - name: KONG_PG_DATABASE
              value: kong
            - name: KONG_PG_PASSWORD
              value: PASSWPRD
            - name: KONG_PG_USER
              value: sagitario
            - name: KONG_ADMIN_LISTEN
              value: 0.0.0.0:8001
            - name: KONG_TRUSTED_IPS
              value: 0.0.0.0/0

        - name: konga
          image: pantsel/konga
          env:
```

```
            - name: NODE_ENV
              value: production
          ports:
            - containerPort: 1337

      volumes:
        - name: kong-pvc
          persistentVolumeClaim:
            claimName: kong-pvc

      restartPolicy: Always
```

J4: *Manifiesto Serivice External de Kubernetes para despliegue de Kong.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: Service
metadata:
  name: kong-external
  labels:
    name: kong
  annotations:
    service.beta.kubernetes.io/do-loadbalancer-name: "simca.space"
    service.beta.kubernetes.io/do-loadbalancer-protocol: "https"
    service.beta.kubernetes.io/do-loadbalancer-certificate-id: "dasdasasd-
8fjgfg06-4e4b-b855-esdasdas"
    service.beta.kubernetes.io/do-loadbalancer-disable-lets-encrypt-dns-
records: "false"
    service.beta.kubernetes.io/do-loadbalancer-redirect-http-to-https: "true"
spec:
  type: LoadBalancer
  ports:
    - name: kong-proxy-listen
      port: 8000
      targetPort: 8000
    - name: kong-proxy-listen-http
      port: 80
      targetPort: 8000
    - name: kong-proxy-listen-https
      port: 443
      targetPort: 8000
    - name: konga-proxy-listen-http
      port: 1337
      targetPort: 1337
  selector:
    name: Kong
```

J5: *Manifiesto Serivice Internal de Kubernetes para despliegue de Kong.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: Service
metadata:
  name: kong-internal
  labels:
    name: kong
spec:
```

```
  ports:
    - name: kong-proxy-listen-ssl
      port: 8443
      targetPort: 8443
    - name: kong-admin-api-listen
      port: 8001
      targetPort: 8001
    - name: kong-db
      port: 5432
      targetPort: 5432
  selector:
    name: kong
```

J6: *Manifiesto Persistent Volume Claim de Kubernetes para despliegue de Kong.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: kong-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100M
status: {}
```

J7: *Manifiesto Deployment de Kubernetes para despliegue del módulo de sistemas.*

Fuente: Elaboración propia

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: systems
spec:
  replicas: 1
  selector:
    matchLabels:
      role: systems
  template:
    metadata:
      labels:
        role: systems
    spec:
      containers:
        - image: webdevops/php-apache:8.0
          name: web
          ports:
            - containerPort: 80
            - containerPort: 443
          resources: {}
          volumeMounts:
            - mountPath: /app
              name: systems
              subPath: services/users
        - image: mysql
          name: db
```

```
        env:
          - name: MYSQL_DATABASE
            value: db_users
          - name: MYSQL_PASSWORD
            value: PASSWPRD
          - name: MYSQL_ROOT_PASSWORD
            value: PASSWPRD
          - name: MYSQL_USER
            value: sagitario
        args:
          - --default-authentication-plugin=mysql_native_password
        ports:
          - containerPort: 3306
        volumeMounts:
          - mountPath: /var/lib/mysql/
            name: systems
            subPath: databases/users
        resources: {}
      - image: node
        name: socket
        workingDir: /home
        command: ["/bin/sh"]
        args: ["-c", "npm install && npm run dev"]
        ports:
          - containerPort: 3000
        volumeMounts:
          - mountPath: /home
            name: systems
            subPath: services/users/socket
        resources: {}
      - image: phpmyadmin/phpmyadmin
        name: phpmyadmin
        env:
          - name: MYSQL_DATABASE
            value: db_fixed_assets
          - name: MYSQL_PASSWORD
            value: PASSWPRD
          - name: MYSQL_ROOT_PASSWORD
            value: PASSWPRD
          - name: MYSQL_USER
            value: sagitario
          - name: PMA_HOST
            value: HOST0
          - name: PMA_PORT
            value: '30000'
          - name: APACHE_HTTP_PORT_NUMBER
            value: '9010'
        ports:
          - containerPort: 9010
        command: ["/bin/bash"]
        args: ["-c", "sed -i \"s/80/$(APACHE_HTTP_PORT_NUMBER)/g\"
/etc/apache2/sites-available/000-default.conf /etc/apache2/ports.conf &&
/docker-entrypoint.sh apache2-foreground"]
        resources: {}
    restartPolicy: Always
    volumes:
      - name: systems
        persistentVolumeClaim:
          claimName: systems
```

J8: *Manifiesto Service Internal de Kubernetes para despliegue del módulo de sistemas.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: Service
metadata:
  name: internal-systems
spec:
  ports:
    - name: "users-db"
      port: 30000
      targetPort: 3306
    - name: "users-api"
      port: 30001
      targetPort: 80
    - name: "users-ssl"
      port: 30002
      targetPort: 443
    - name: "users-socket"
      port: 30003
      targetPort: 3000
    - name: "users-phpmyadmin"
      port: 30004
      targetPort: 9010
  selector:
    role: systems
```

J9: *Manifiesto Persisten Volume Claim de Kubernetes para despliegue del módulo de sistemas.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: systems
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100M
status: {}
```

J10: *Manifiesto Deployment de Kubernetes para despliegue del módulo de beneficiarios.*

Fuente: Elaboración propia

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: insureds
spec:
  replicas: 1
  selector:
    matchLabels:
      role: insureds
  template:
    metadata:
```

```
      labels:
        role: insureds
    spec:
      containers:
        - image: webdevops/php-apache:8.0
          name: web
          ports:
            - containerPort: 80
            - containerPort: 443
          resources: {}
          volumeMounts:
            - mountPath: /app
              name: web-insureds
        - image: mysql
          name: db
          env:
            - name: MYSQL_DATABASE
              value: db_insureds
            - name: MYSQL_PASSWORD
              value: PASSWPRD
            - name: MYSQL_ROOT_PASSWORD
              value: PASSWPRD
            - name: MYSQL_USER
              value: sagitario
          args:
            - --default-authentication-plugin=mysql_native_password
          ports:
            - containerPort: 3306
          volumeMounts:
            - mountPath: /var/lib/mysql/
              name: web-insureds
              subPath: mysql
          resources: {}
        - image: phpmyadmin/phpmyadmin
          name: phpmyadmin
          env:
            - name: MYSQL_DATABASE
              value: db_fixed_assets
            - name: MYSQL_PASSWORD
              value: PASSWPRD
            - name: MYSQL_ROOT_PASSWORD
              value: PASSWPRD
            - name: MYSQL_USER
              value: sagitario
            - name: PMA_HOST
              value: HOST
            - name: PMA_PORT
              value: '30000'
            - name: APACHE_HTTP_PORT_NUMBER
              value: '9010'
          ports:
            - containerPort: 9010
          command: ["/bin/bash"]
          args: ["-c", "sed -i \"s/80/$(APACHE_HTTP_PORT_NUMBER)/g\"
/etc/apache2/sites-available/000-default.conf /etc/apache2/ports.conf &&
/docker-entrypoint.sh apache2-foreground"]
          resources: {}
      restartPolicy: Always
      volumes:
        - name: web-insureds
          persistentVolumeClaim:
            claimName: web-insureds
```

J11: *Manifiesto Service Internal de Kubernetes para despliegue del módulo de beneficiarios.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: Service
metadata:
  name: insureds-internal
spec:
  ports:
    - name: "insureds-db"
      port: 30000
      targetPort: 3306
    - name: "insureds-api"
      port: 30001
      targetPort: 80
    - name: "insureds-ssl"
      port: 30002
      targetPort: 443
    - name: "insureds-phpmyadmin"
      port: 30004
      targetPort: 9010
  selector:
    role: insureds
```

J12: *Manifiesto Persisten Volume Claim de Kubernetes para despliegue del módulo de beneficiarios.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: web-insureds
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 15Gi
status: {}
```

J13: *Manifiesto de Kubernetes para despliegue del módulo de fichaje.*

Fuente: Elaboración propia

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tokens
spec:
  replicas: 1
  selector:
    matchLabels:
      role: tokens
  template:
    metadata:
      labels:
        role: tokens
```

```
  spec:
    containers:
      - image: webdevops/php-apache:8.0
        name: web
        ports:
          - containerPort: 80
          - containerPort: 443
        resources: {}
        volumeMounts:
          - mountPath: /app
            name: web-tokens
      - image: mysql
        name: db
        env:
          - name: MYSQL_DATABASE
            value: db_tokens
          - name: MYSQL_PASSWORD
            value: PASSWPRD
          - name: MYSQL_ROOT_PASSWORD
            value: PASSWPRD
          - name: MYSQL_USER
            value: sagitario
        args:
          - --default-authentication-plugin=mysql_native_password
        ports:
          - containerPort: 3306
        volumeMounts:
          - mountPath: /var/lib/mysql/
            name: web-tokens
            subPath: mysql
        resources: {}
      - image: node
        name: socket
        workingDir: /home
        command: ["/bin/sh"]
        args: ["-c", "npm install && npm run dev"]
        ports:
          - containerPort: 3000
        volumeMounts:
          - mountPath: /home
            name: web-tokens
            subPath: socket
        resources: {}
      - image: phpmyadmin/phpmyadmin
        name: phpmyadmin
        env:
          - name: MYSQL_DATABASE
            value: db_fixed_assets
          - name: MYSQL_PASSWORD
            value: PASSWPRD
          - name: MYSQL_ROOT_PASSWORD
            value: PASSWPRD
          - name: MYSQL_USER
            value: sagitario
          - name: PMA_HOST
            value: HOST
          - name: PMA_PORT
            value: '30000'
          - name: APACHE_HTTP_PORT_NUMBER
            value: '9010'
        ports:
          - containerPort: 9010
        command: ["/bin/bash"]
```

```
          args: ["-c", "sed -i \"s/80/$(APACHE_HTTP_PORT_NUMBER)/g\"
/etc/apache2/sites-available/000-default.conf /etc/apache2/ports.conf &&
/docker-entrypoint.sh apache2-foreground"]
          resources: {}
      restartPolicy: Always
      volumes:
        - name: web-tokens
          persistentVolumeClaim:
            claimName: web-tokens
```

J14: *Manifiesto Persisten Volume Claim de Kubernetes para despliegue del módulo de fichaje.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: Service
metadata:
  name: tokens-internal
spec:
  ports:
    - name: "tokens-db"
      port: 30000
      targetPort: 3306
    - name: "tokens-api"
      port: 30001
      targetPort: 80
    - name: "tokens-ssl"
      port: 30002
      targetPort: 443
    - name: "tokens-socket"
      port: 30003
      targetPort: 3000
    - name: "tokens-phpmyadmin"
      port: 30004
      targetPort: 9010
  selector:
    role: tokens
```

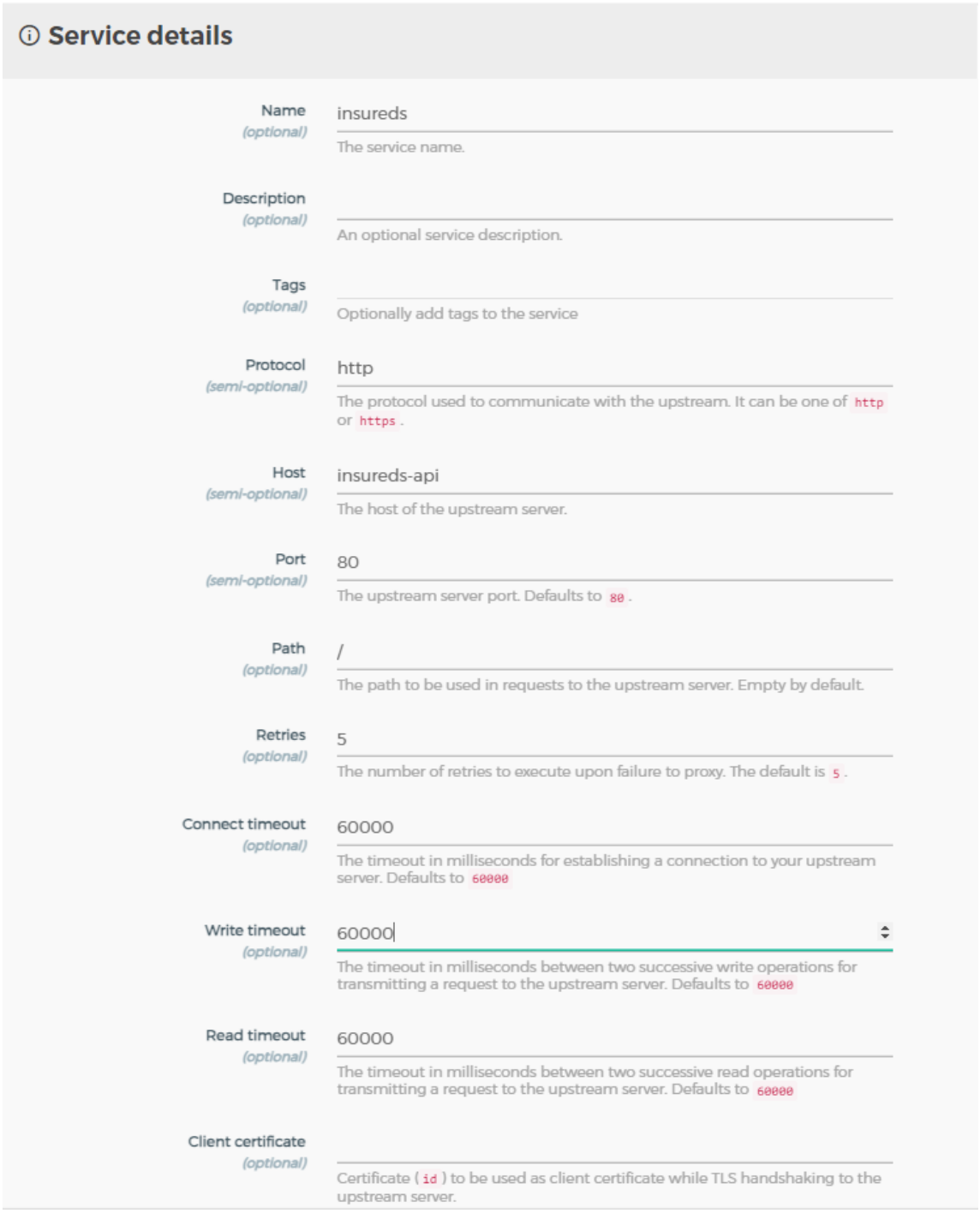
J15: *Manifiesto Persisten Volume Claim de Kubernetes para despliegue del módulo de fichaje.*

Fuente: Elaboración propia

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: web-tokens
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100M
status: {}
```

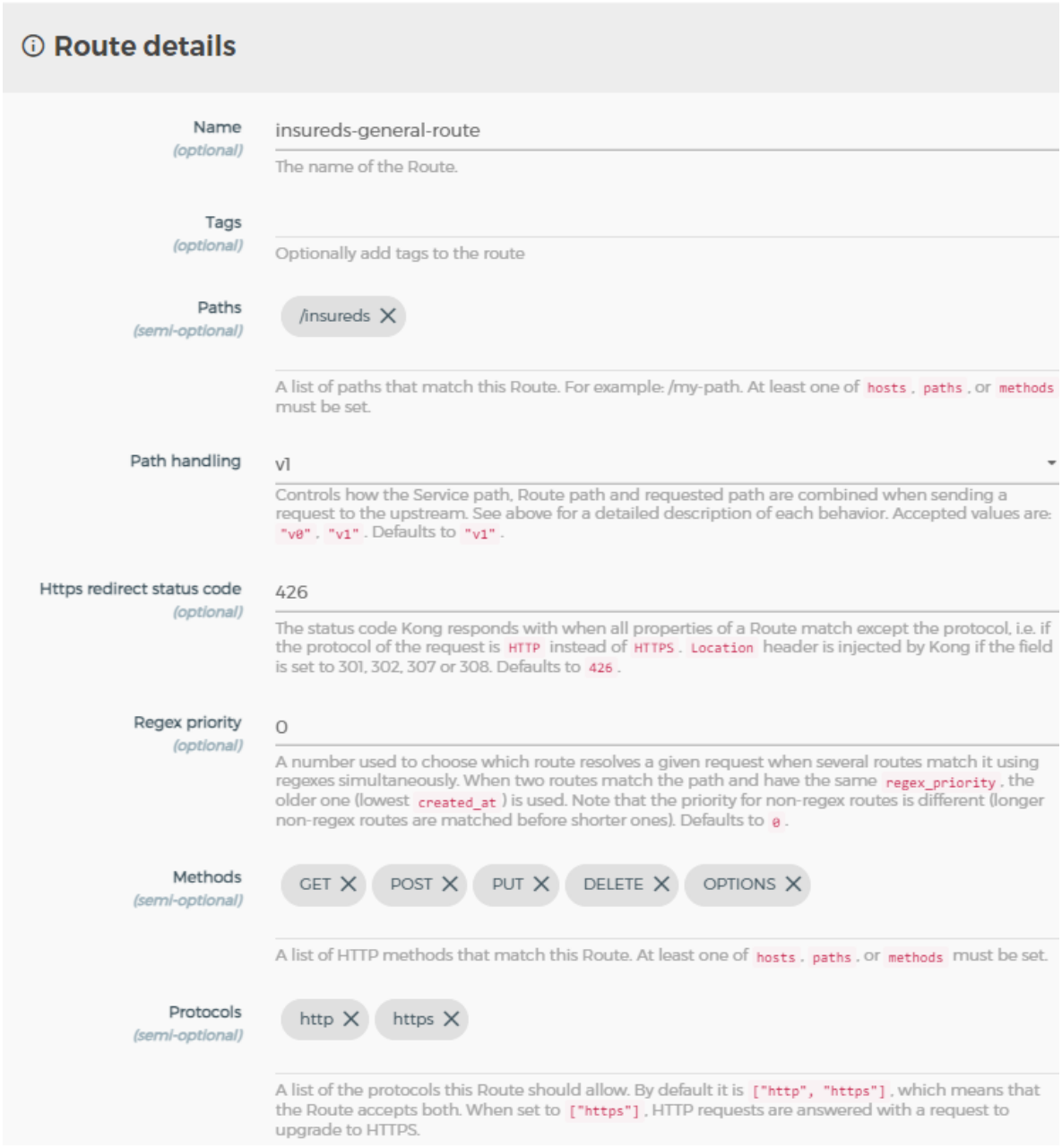
K1: *Configuración del servicio de beneficiarios en Kong.*

Fuente: Elaboración propia



K2: *Configuración de la ruta general de beneficiarios en Kong.*

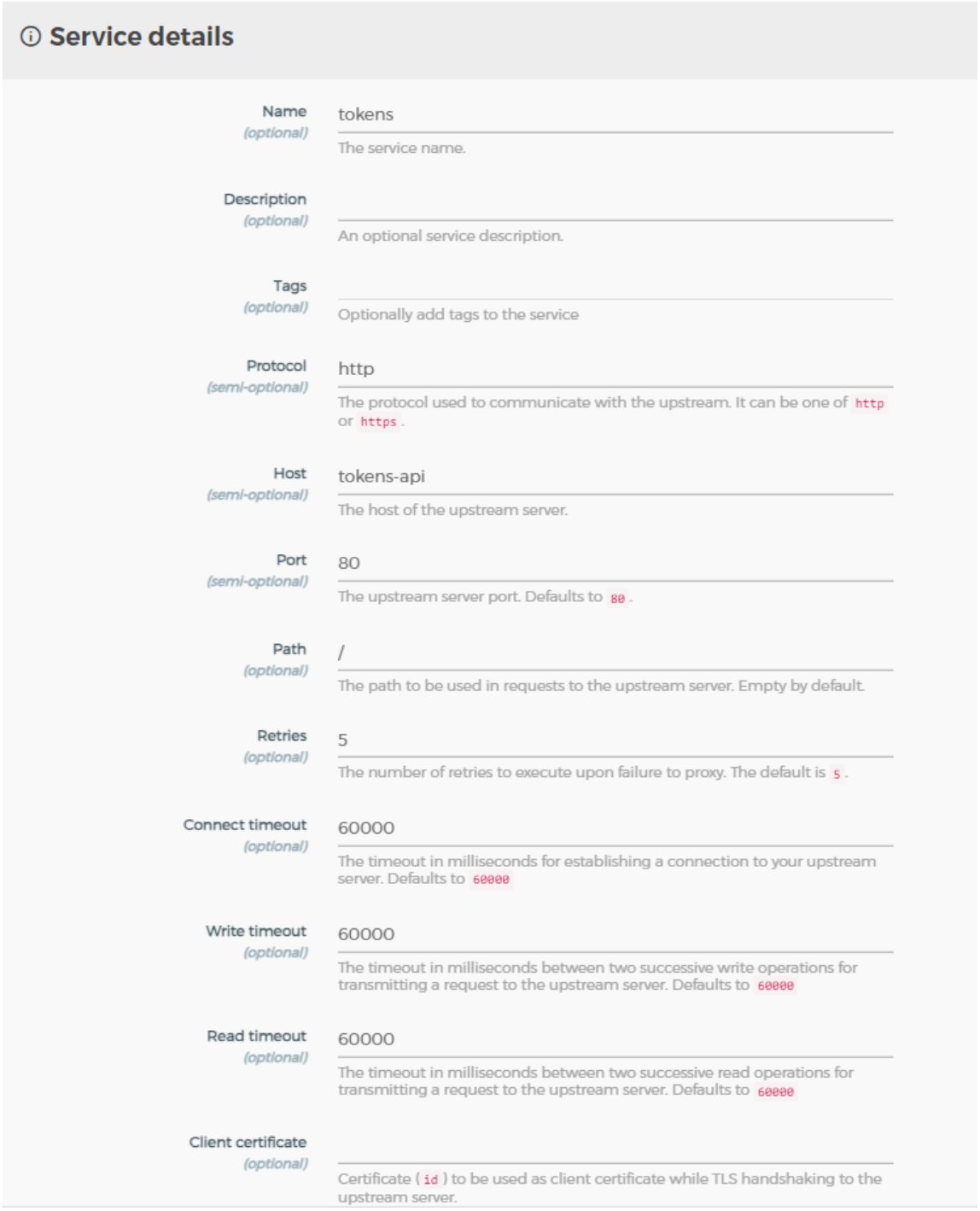
Fuente: Elaboración propia



ANEXOS L: *Resultados del Sprint 3*

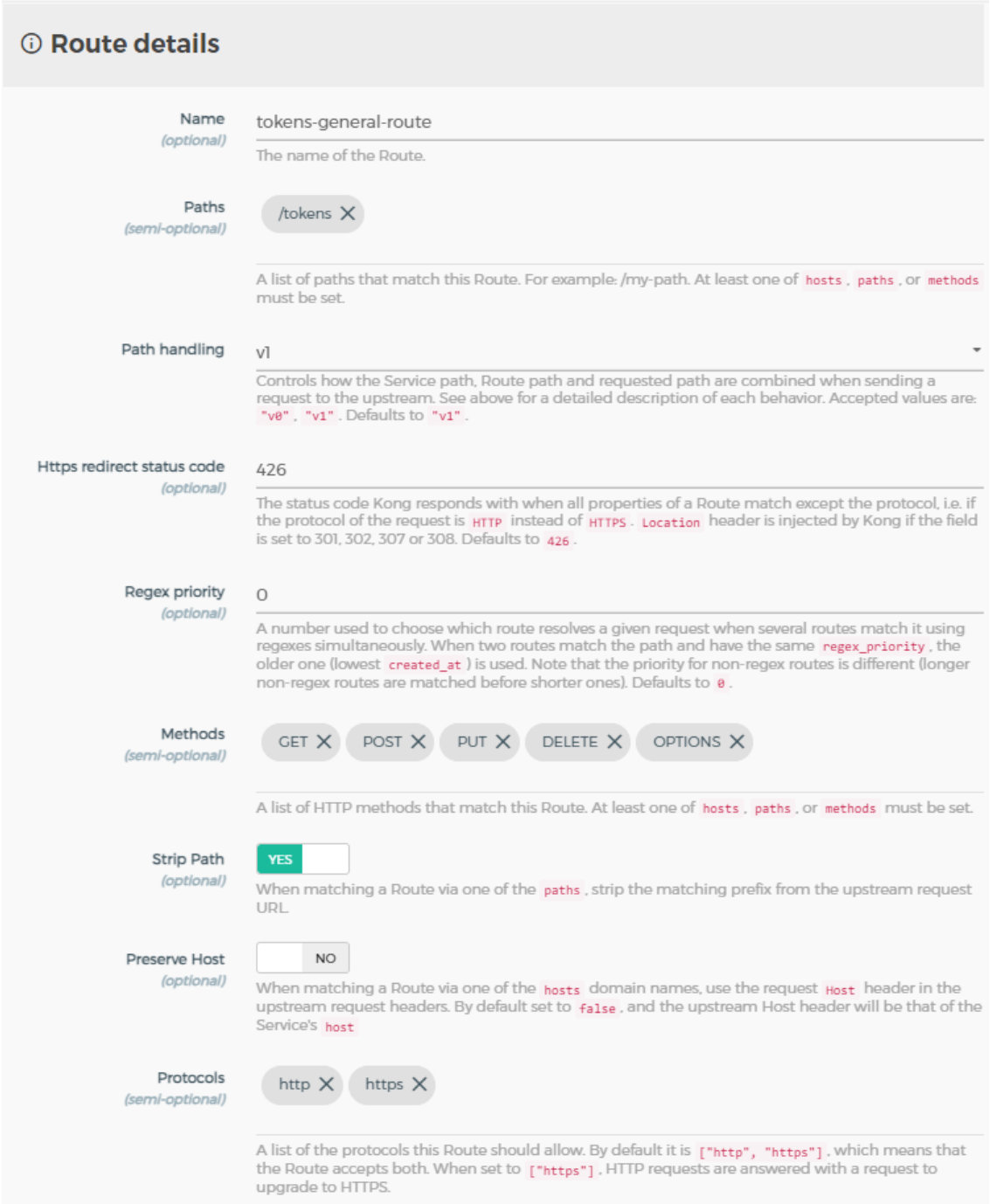
L1: *Configuración del servicio de fichaje en Kong.*

Fuente: Elaboración propia



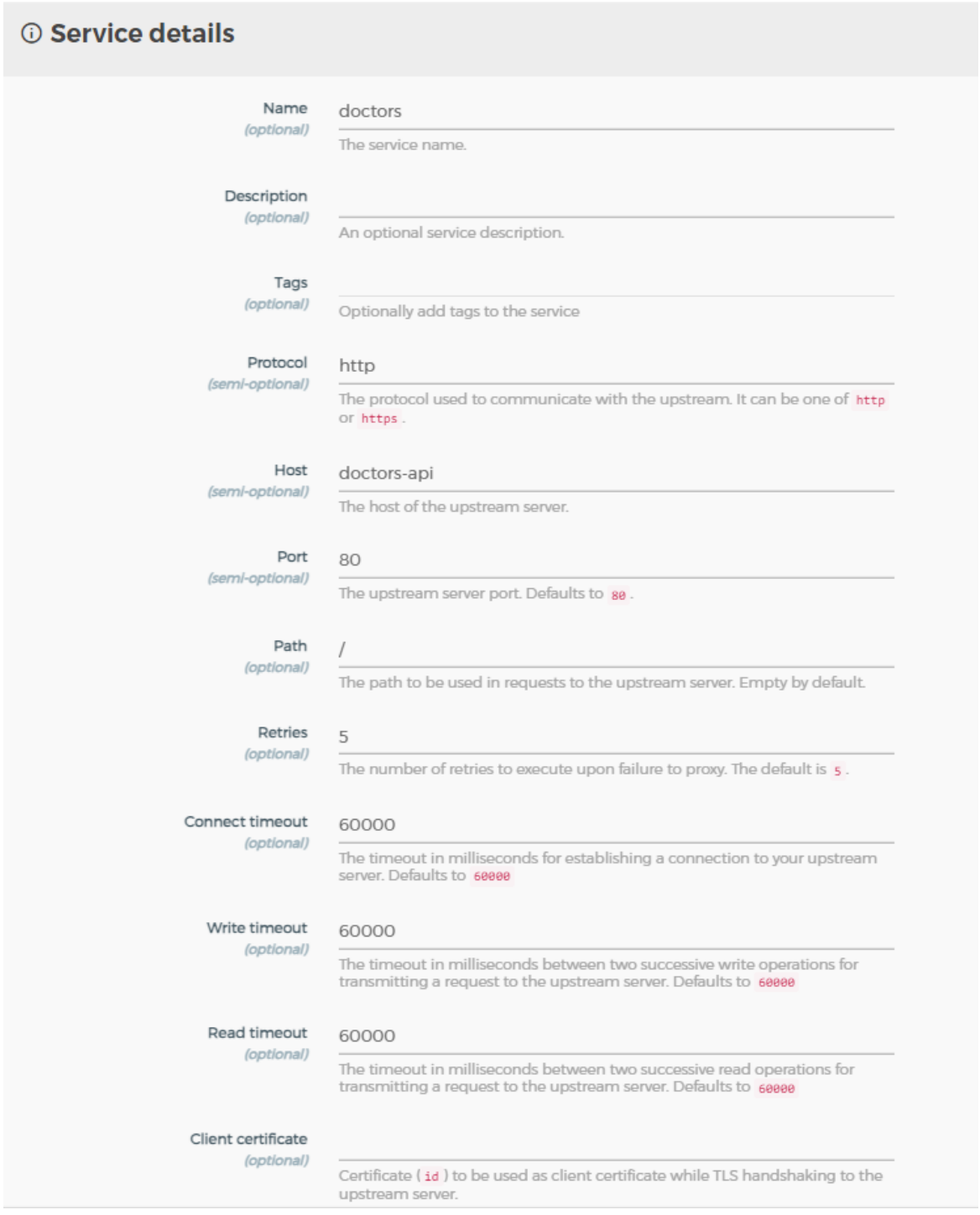
L2: *Configuración de la ruta general de fichaje en Kong.*

Fuente: Elaboración propia



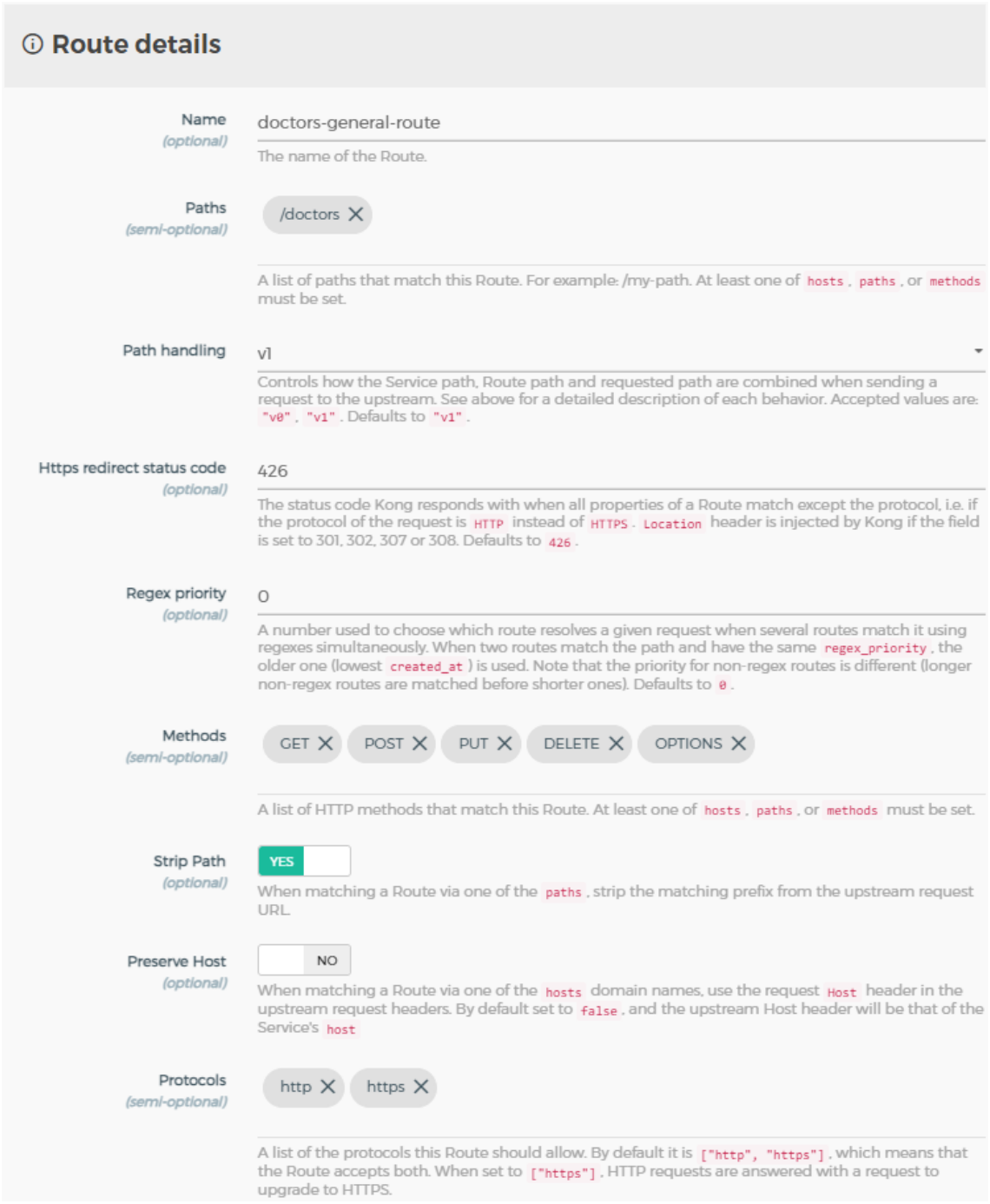
L3: *Configuración del servicio de doctores en Kong.*

Fuente: Elaboración propia



L4: *Configuración de la ruta general de doctores en Kong.*

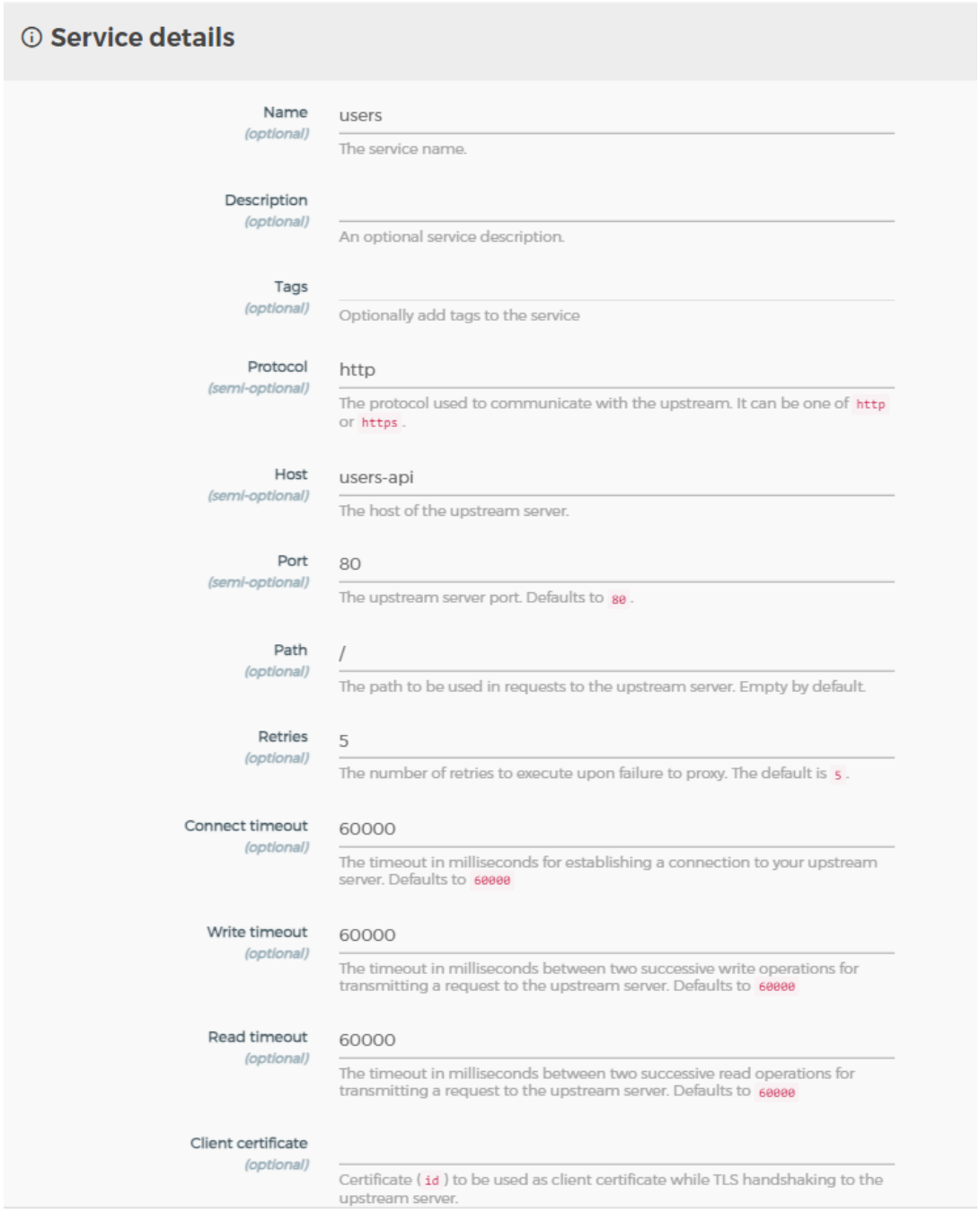
Fuente: Elaboración propia



ANEXOS M: *Resultados del Sprint 4*

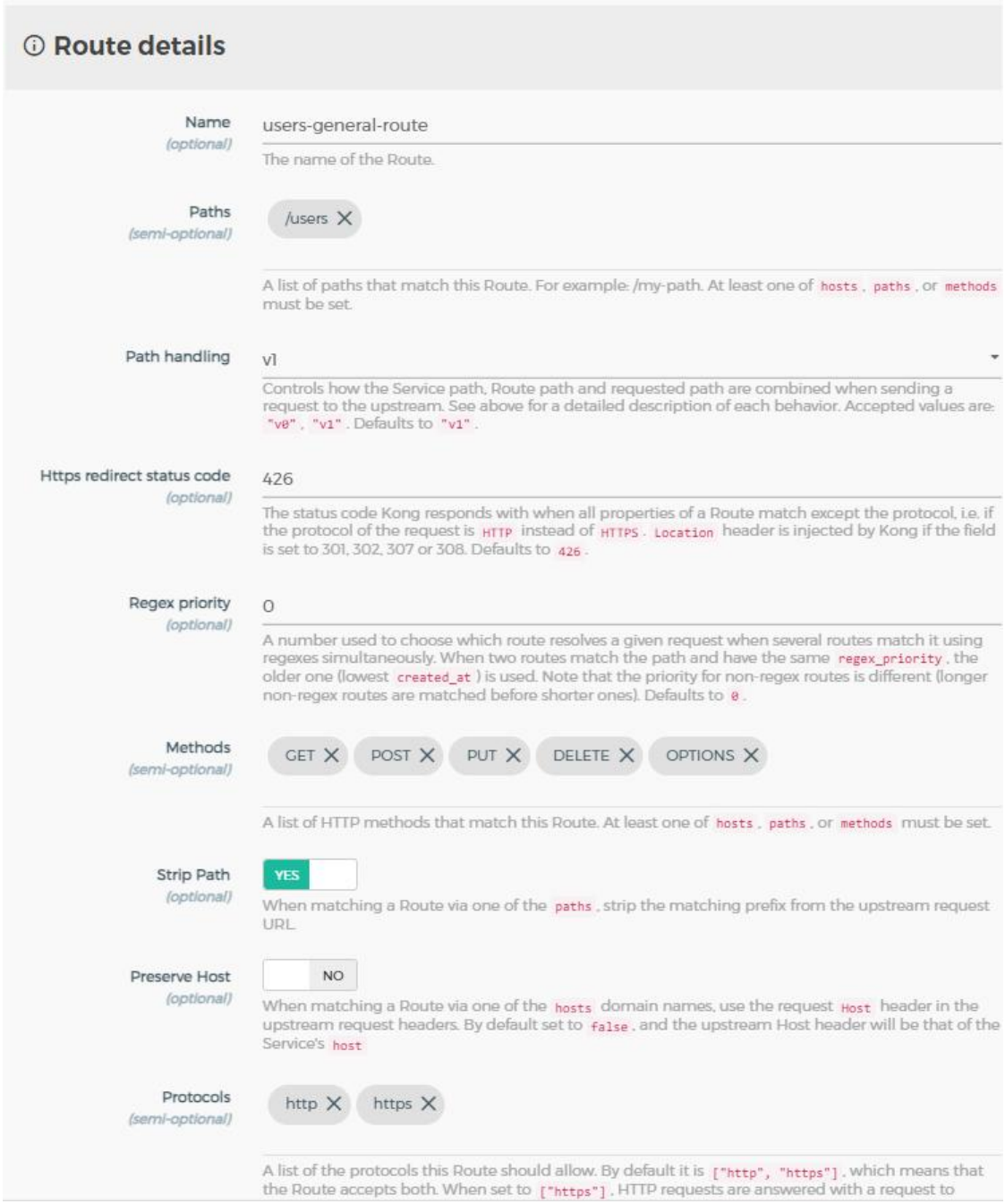
M1: *Configuración del servicio de usuarios en Kong.*

Fuente: Elaboración propia



M2: *Configuración de la ruta general de usuarios en Kong.*

Fuente: Elaboración propia



ANEXOS N: *Resultados del Sprint 8*

N1: *Ejemplo de middleware implementado.*

Fuente: Elaboración propia

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Tymon\JWTAuth\Facades\JWTAuth;

class JwtMiddleware
{
    public function handle($request, Closure $next)
    {
        try {
            JWTAuth::parseToken()->authenticate();
        } catch (\Exception $e) {
            // Manejar error de autenticación (token inválido, caducado, etc.)
            return response()->json([
                'status' => '401',
                'error' => 'Unauthorized'
            ], 401);
        }

        // Puedes hacer más verificaciones aquí según tus necesidades

        return $next($request);
    }
}
```

N2: *Definición de rutas.*

Fuente: Elaboración propia

```php
Route::prefix('private')->group(function () {
    Route::middleware(['jwt.auth'])->group(function () {

    });
});
```

N3: *Registro de la base de datos para usarse..*

Fuente: Elaboración propia

```php
        'db_user' => [
            'driver' => 'mysql',
            'url' => env('DATABASE_URL'),
            'host' => env('DB_USER_HOST', '127.0.0.1'),
            'port' => env('DB_USER_PORT', '3306'),
            'database' => env('DB_USER_DATABASE', 'forge'),
```

```
            'username' => env('DB_USER_USERNAME', 'forge'),
            'password' => env('DB_USER_PASSWORD', ''),
            'unix_socket' => env('DB_SOCKET', ''),
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'prefix_indexes' => true,
            'strict' => true,
            'engine' => null,
            'options' => extension_loaded('pdo_mysql') ? array_filter([
                PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
            ]) : [],
        ],
```

N4 *Ejemplo de modelo implementado en cada servicio.*

Fuente: Elaboración propia

```
<?php
namespace App\Models;

use Tymon\JWTAuth\Contracts\JWTSubject;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable implements JWTSubject{
    use HasFactory, Notifiable;

    protected $connection = 'db_user';

    protected $fillable = [
        'names',
        'last_name_1',
        'last_name_2',
        'email',
        'ci',
        'id_ext',
        'password',
        'id_access',
        'id_db_status',
        'id_area',
        'id_subsidiary',
        'id_office',
        'img',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
```

```
    ];

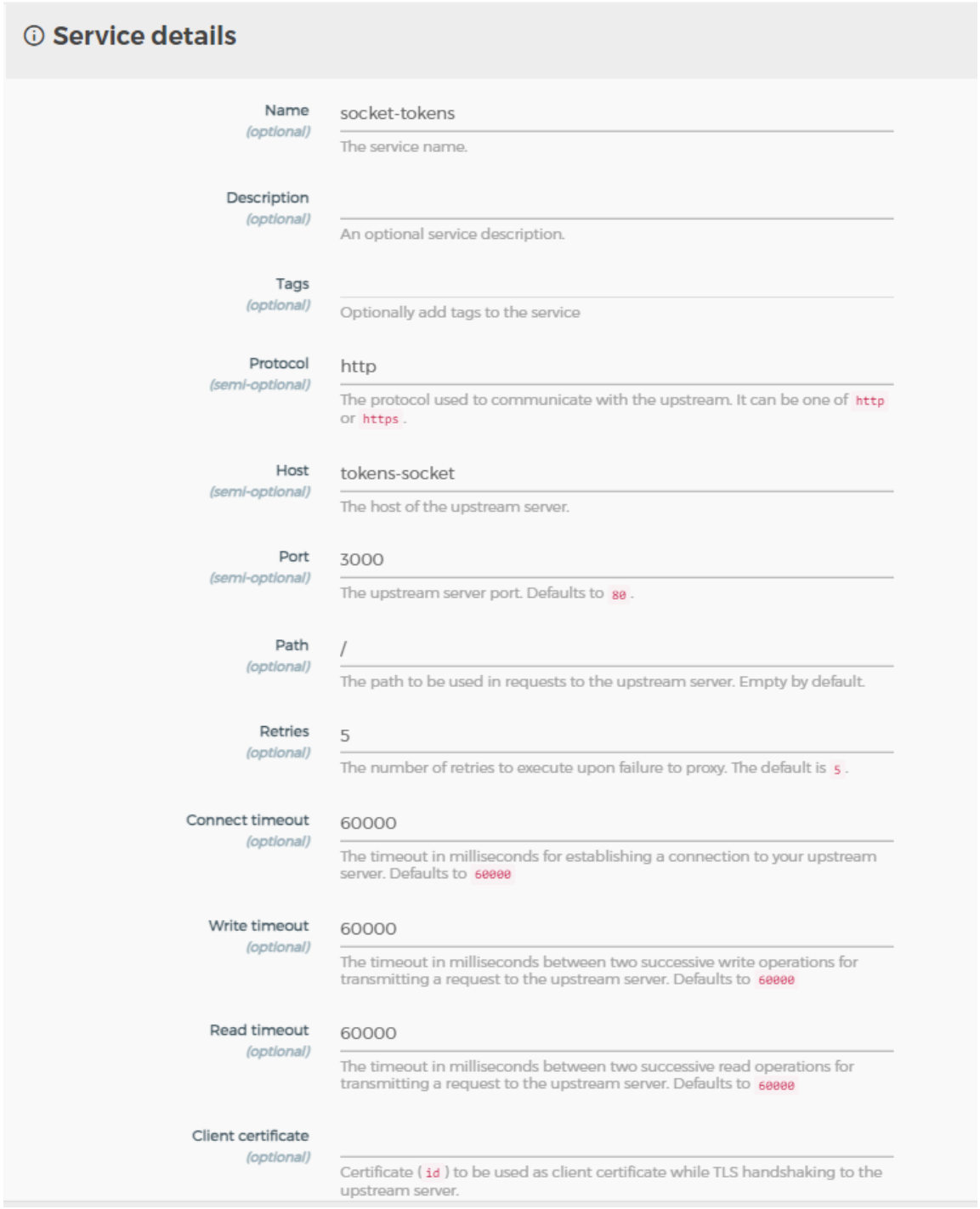
    public function getJWTIdentifier()
    {
        return $this->getKey();
    }

    public function getJWTCustomClaims()
    {
        return [];
    }
}
```

ANEXOS O: *Resultados del Sprint 13*

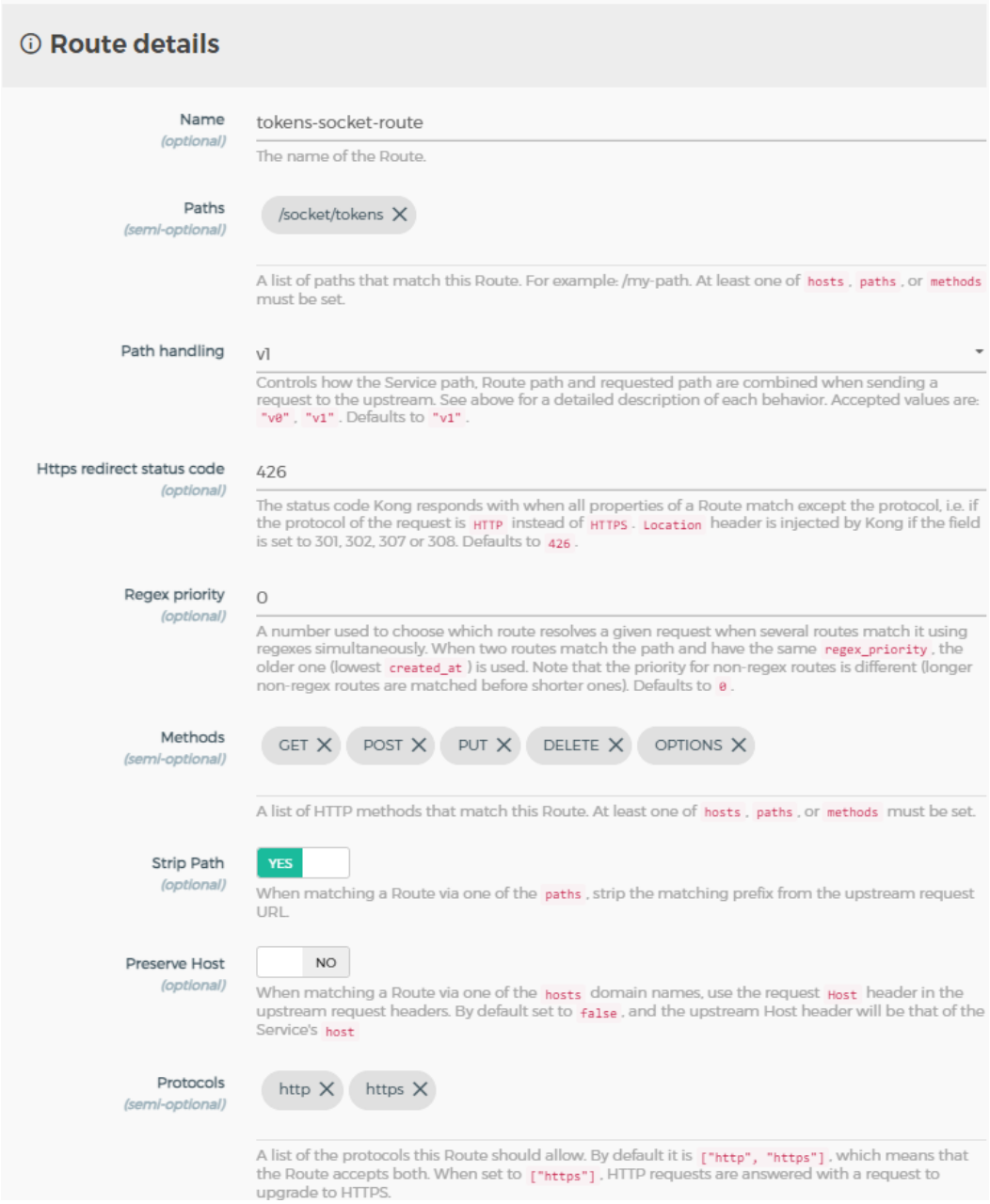
O1: *Configuración del socket de fichaje en Kong.*

Fuente: Elaboración propia



O2: *Configuración de la ruta general del socket de fichaje en Kong.*

Fuente: Elaboración propia



ANEXOS P: *Resultados del Sprint 14*

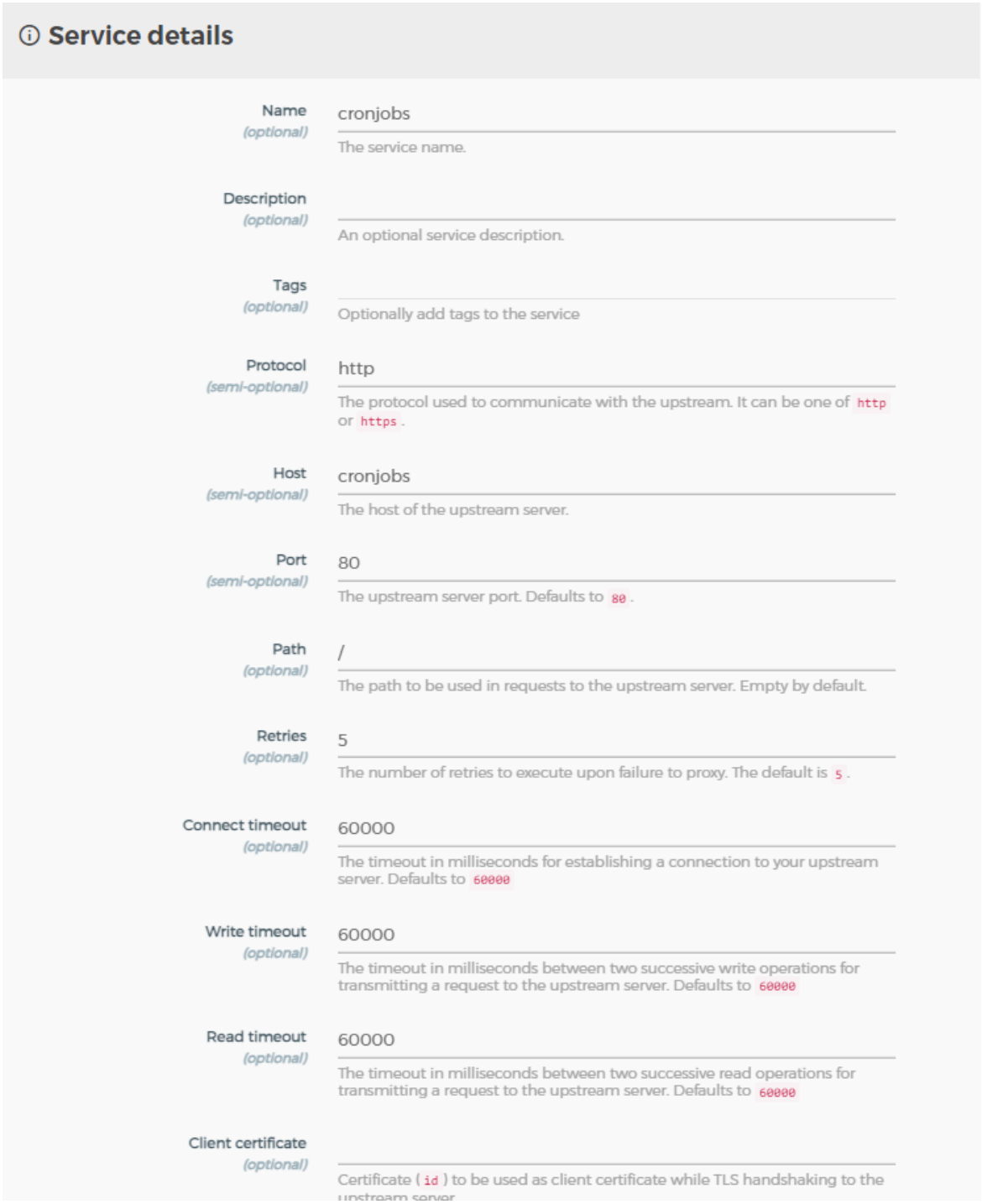
P1: *Configuración del socket de fichaje en Kong.*

Fuente: Elaboración propia

```
cronjobs:
  container_name: cronjobs
  image: node
  restart: always
  working_dir: /home/app
  command: /bin/sh -c "npm install && npm start"
  expose:
    - 80
  volumes:
    - ./cronjobs:/home
```

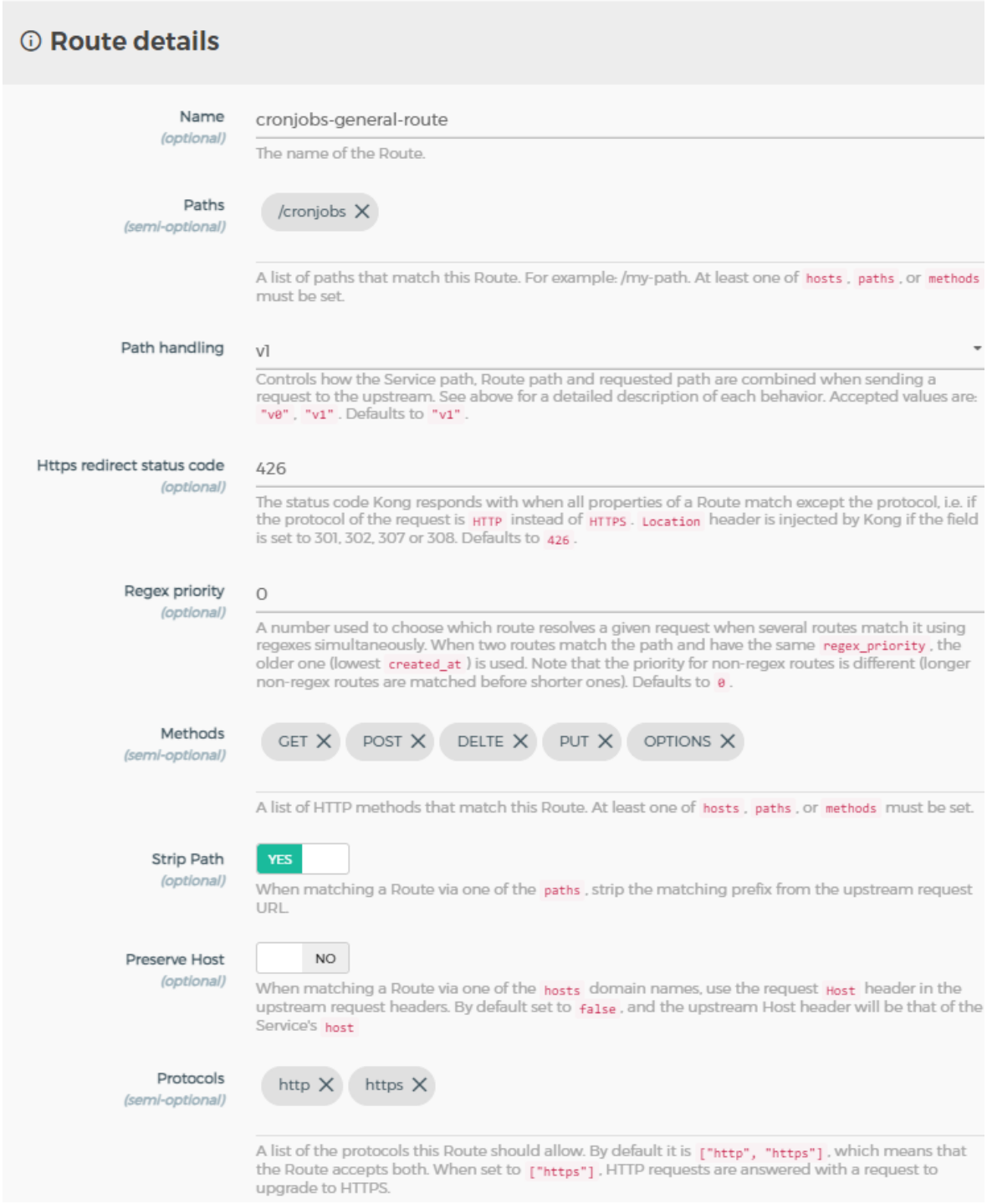
P2: *Configuración del servicio de cronjobs en Kong.*

Fuente: Elaboración propia



P3: *Configuración de las rutas del cronjob en Kong.*

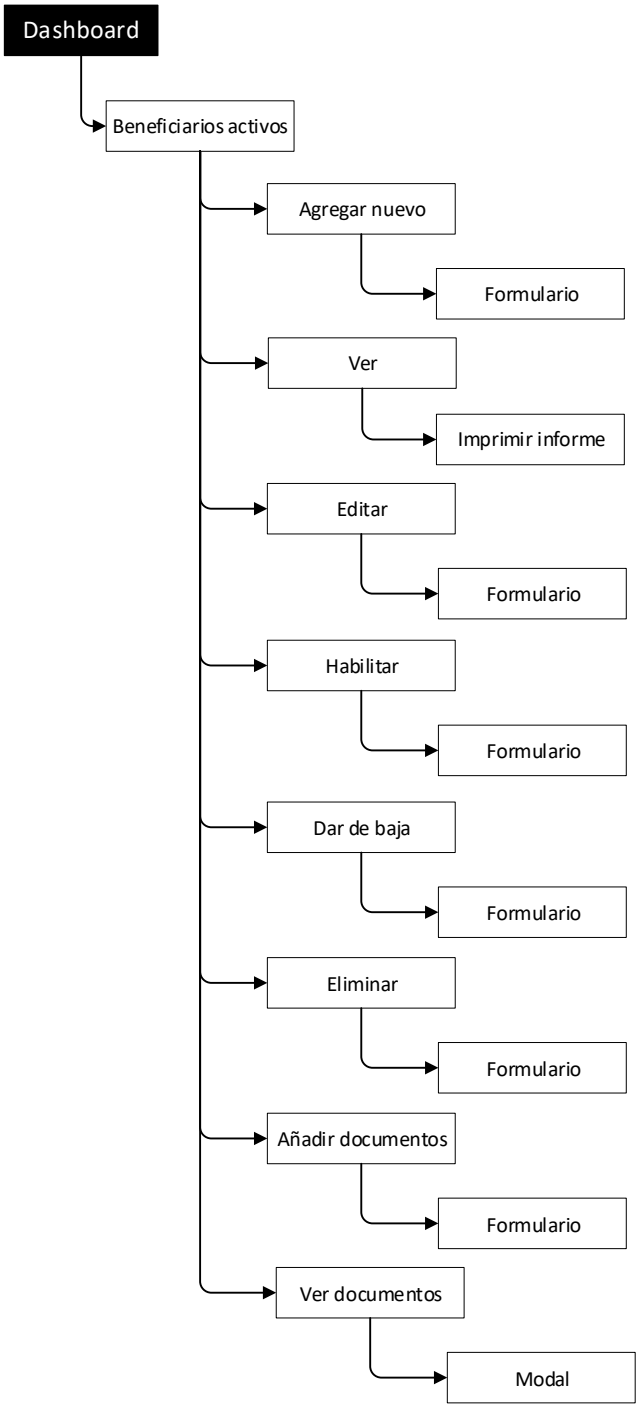
Fuente: Elaboración propia



ANEXOS Q: *Mapas de navegación del módulo de vigencia*

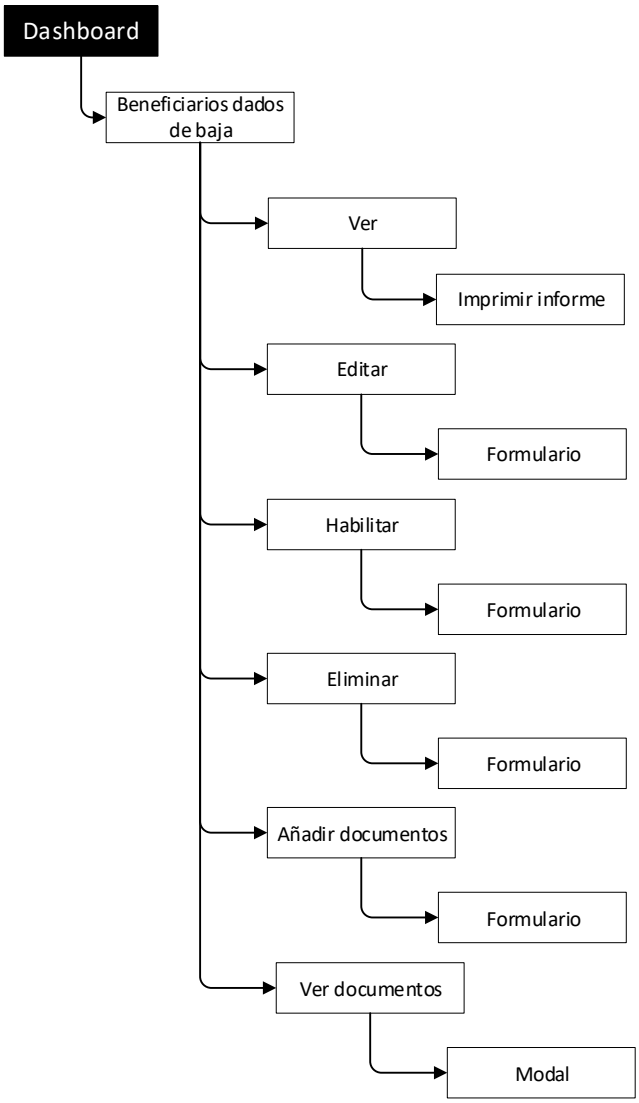
Q1: *Beneficiarios activos*

Fuente: Elaboración propia



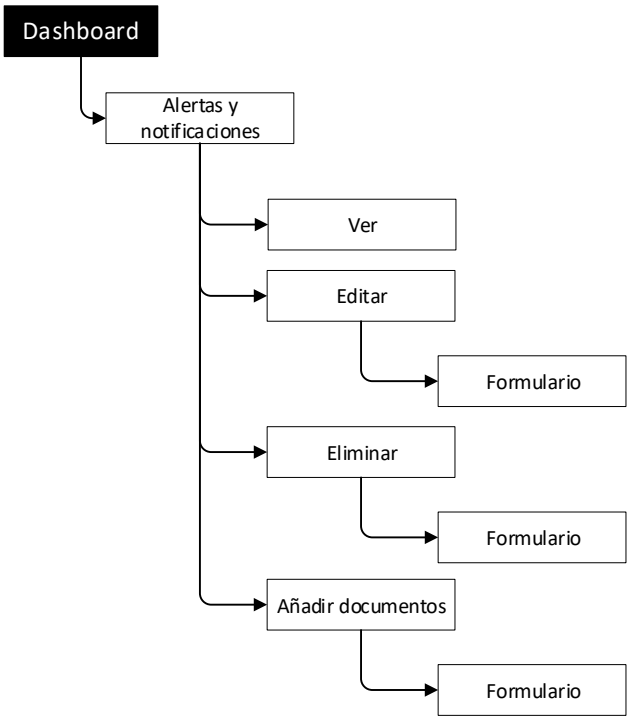
Q2: *Beneficiarios dados de baja*

Fuente: Elaboración propia



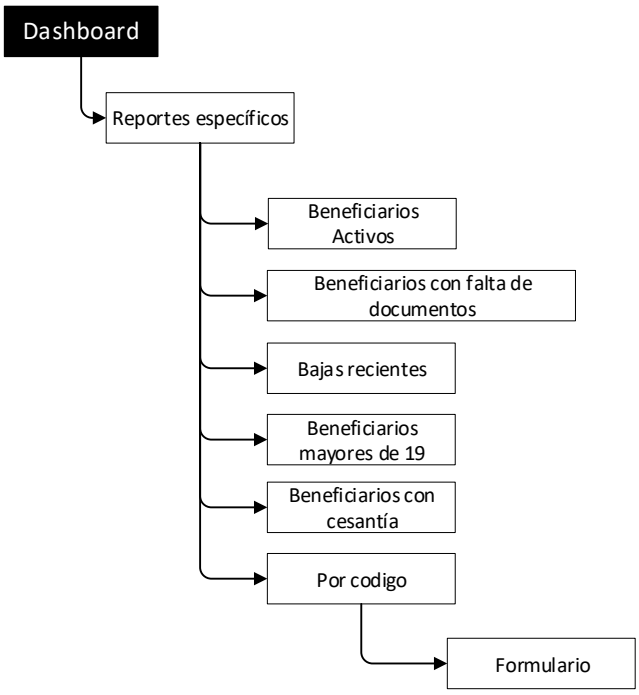
Q3: *Alertas y notificaciones*

Fuente: Elaboración propia



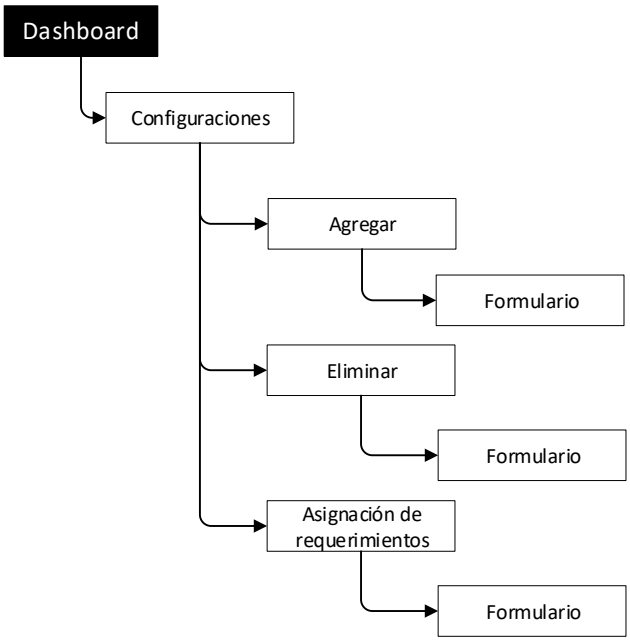
Q4: *reportes específicos*

Fuente: Elaboración propia



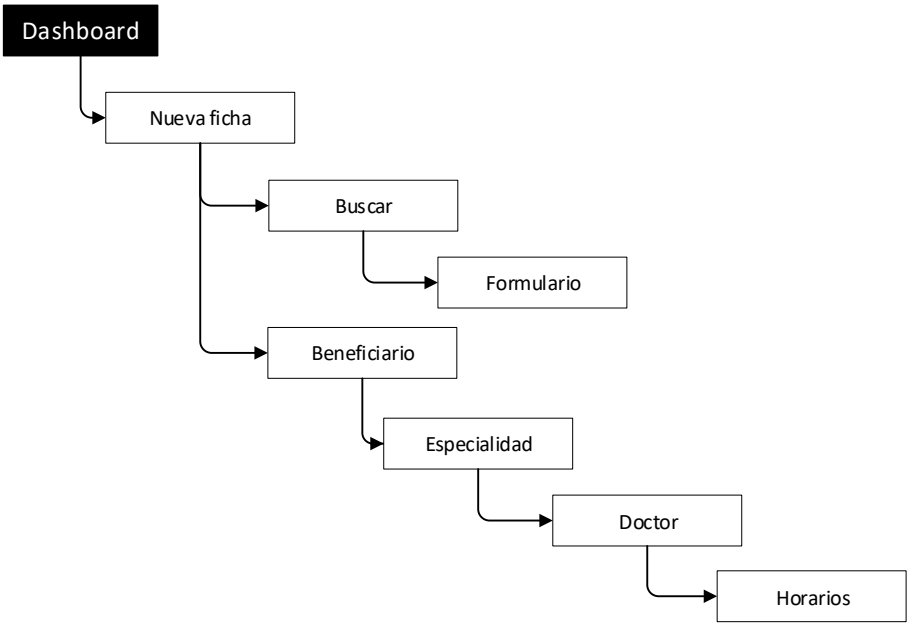
Q5: *Configuraciones*

Fuente: Elaboración propia



Q6: *Alertas y notificaciones*

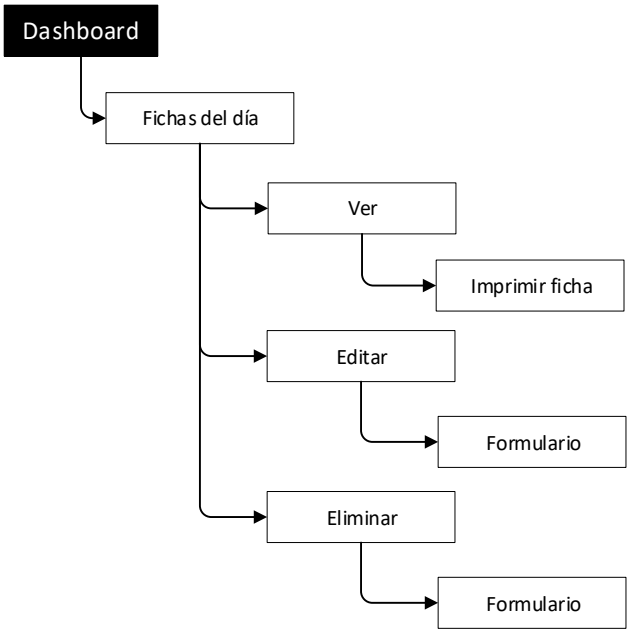
Fuente: Elaboración propia



ANEXOS R: *Mapas de navegación del módulo de fichaje*

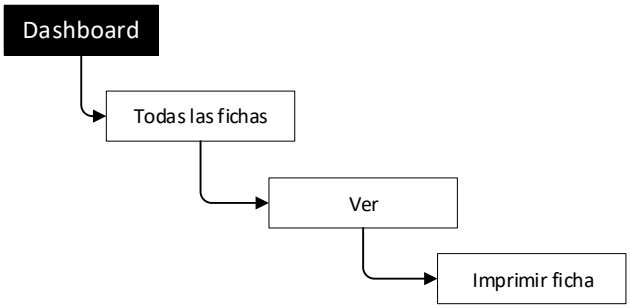
R1: *Fichas del día*

Fuente: Elaboración propia



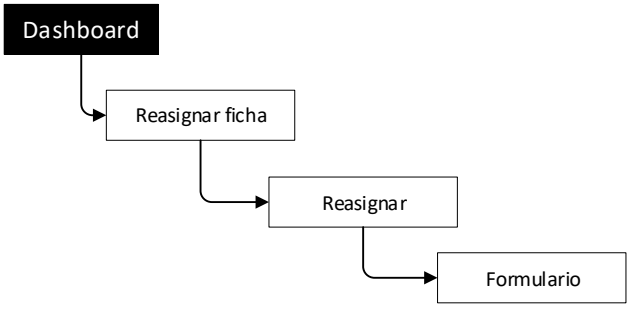
R1: *Todas las fichas*

Fuente: Elaboración propia



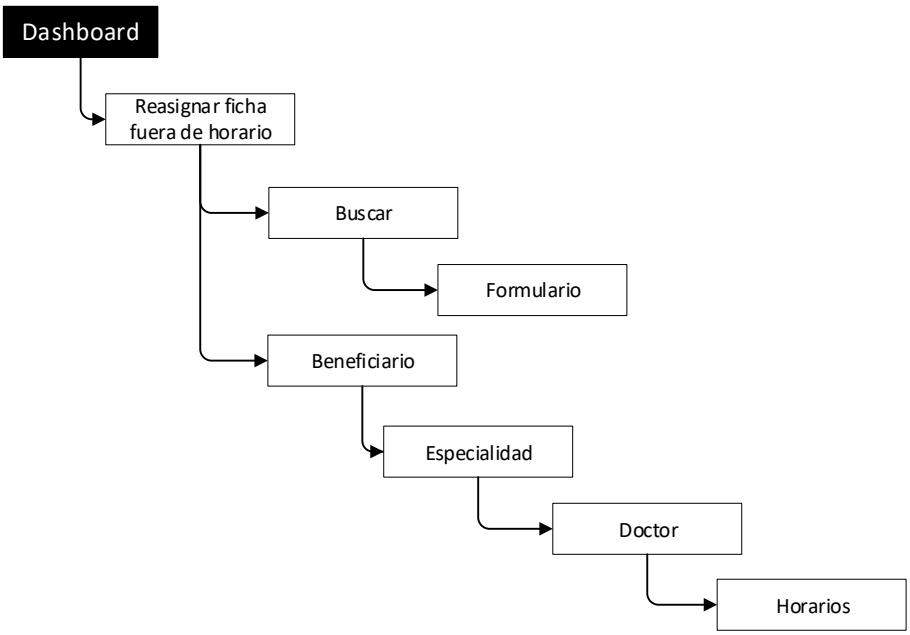
R1: *Reasignar fichas*

Fuente: Elaboración propia



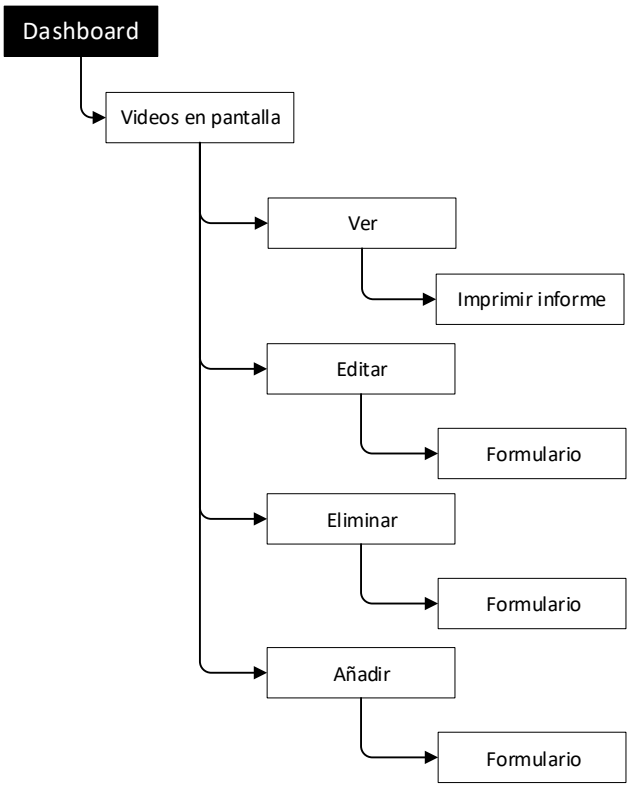
R1: *Reasignar fichas fuera de horario*

Fuente: Elaboración propia



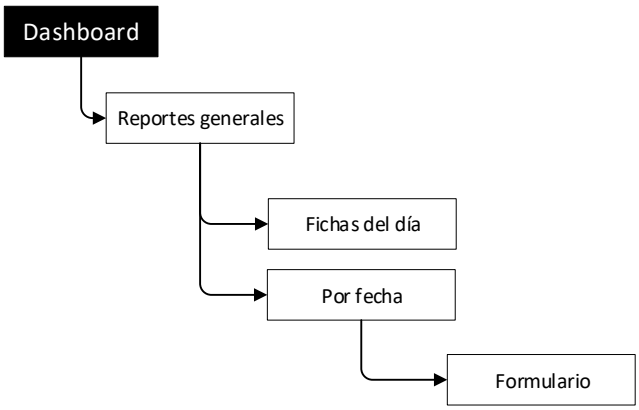
R2: *Videos en pantalla*

Fuente: Elaboración propia



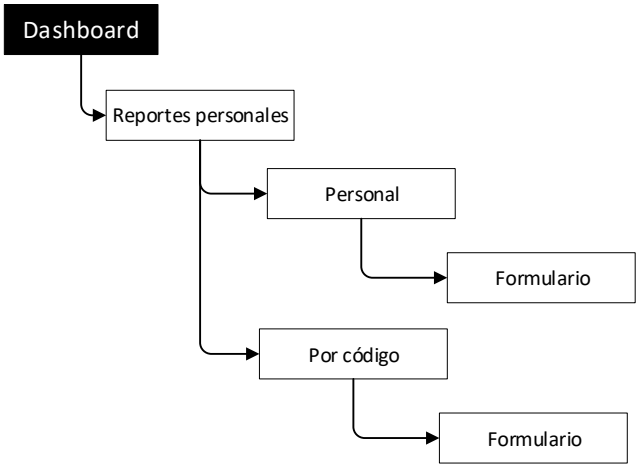
R3: *Reportes generales*

Fuente: Elaboración propia



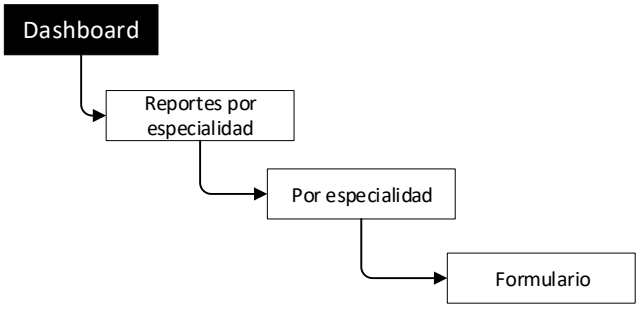
R4: *Reportes personales*

Fuente: Elaboración propia



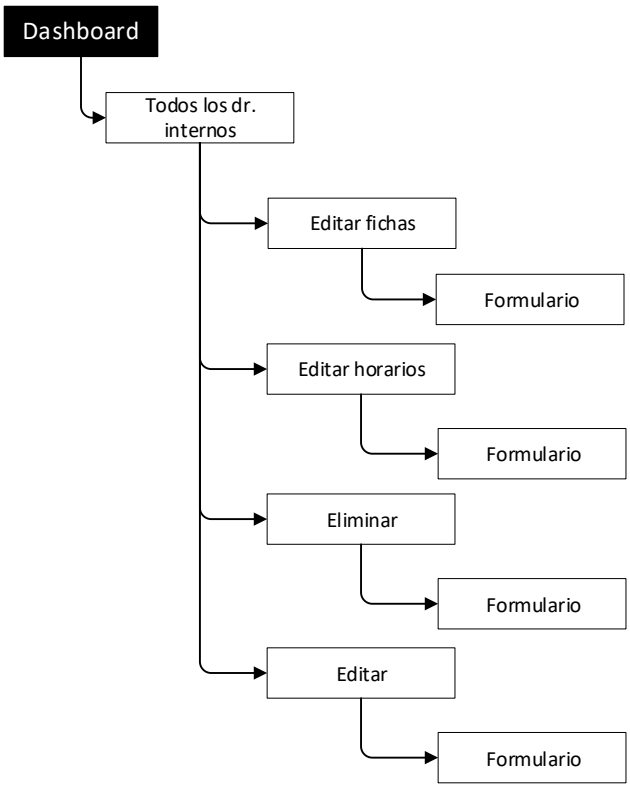
R5: *reportes por especialidad*

Fuente: Elaboración propia



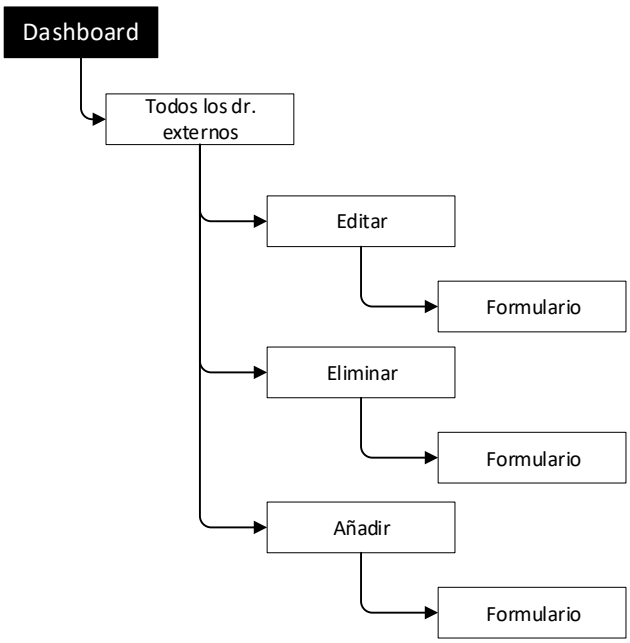
R6: *Todos los drs. internos*

Fuente: Elaboración propia



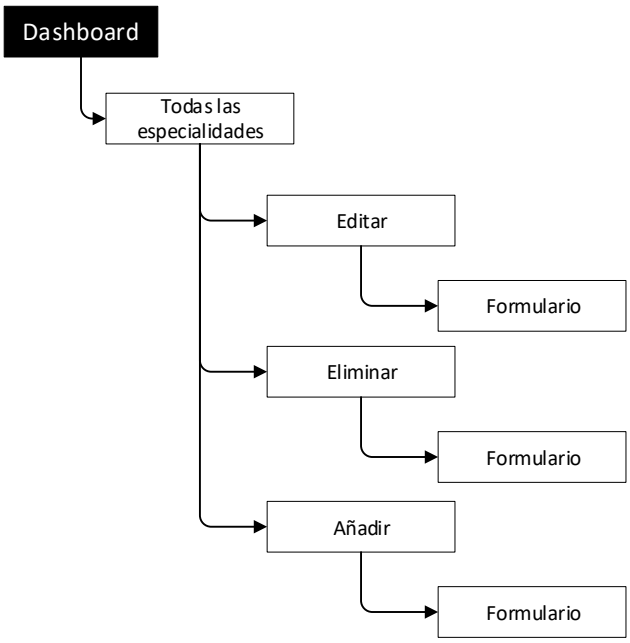
R6: *Todos los drs. externos*

Fuente: Elaboración propia



R7: *Todas las especialidades*

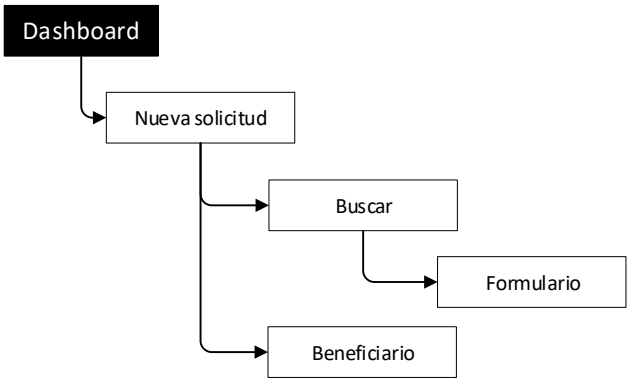
Fuente: Elaboración propia



ANEXOS S: *Mapas de navegación del módulo de sistemas*

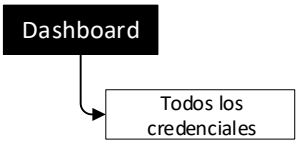
S1: *Nueva solicitud*

Fuente: Elaboración propia



S2: *Todas las credenciales*

Fuente: Elaboración propia



S3: *Todas las solicitudes*

Fuente: Elaboración propia

