UNIVERSIDAD LA SALLE

CARRERA DE INGENIERÍA DE SISTEMAS



Automatización de los casos de prueba aplicando la herramienta de JMeter para el sistema Material Flow"

Caso: "New Vision Data"

Por

Nombre: Clemencia Mariluna Alavi Choque

Tutor: M. Sc. Walter Carvajal Mamani

Proyecto de grado presentado para la obtención del Grado de Licenciatura en Ingeniería de Sistemas

La Paz - Bolivia

2019

Agradecimientos

Agradecimientos A Dios, por permitirme llegar a este momento de mi vida tan especial, por los triunfos y los momentos más difíciles que me han enseñado a valorarlo cada día más.

A mi madre Eduarda por todo el apoyo y sacrificio para darme la oportunidad de estudiar hasta obtener una profesión y su infinito amor por todos sus hijos y nietos.

A mi hermana Marisol por todos los consejos que me dio a lo largo de la carrera y por el gran ejemplo de dedicación que siempre mostró.

A todos los profesores del colegio que siempre nos impartieron sus conocimientos con mucha paciencia.

A Michell Angello Ramírez y José Quintanilla quienes me enseñan y enseñaron con el ejemplo que todos los días se debe aprender algo nuevo.

A Walter Carvajal por ser mi tutor y ayudarme con el proceso de titulalación y enseñarme la vida debe vivirse como vacaciones.

A Lic. Martha Heredia por todo el apoyo y su conocimiento impartido durante la carrera en la universidad.

Por ultimo a todos mis amigos del colegio y universidad por el aporyo y la confianza hacia mi persona.

¡MUCHAS GRACIAS A TODOS!

Dedicatoria

A Dios:

Por haberme permitido llegar a hasta este punto de mi vida con mucha salud, prosperidad y rodeada de personas maravillosas.

A mi amada madre Eduarda:

Por todos sus consejos y haber servido de ejemplo como persona y mujer que me acompaño durante mi vida universitaria.

A mi familia:

Por que ser el pilar fundamental en vida en todo sentido y todo el apoyo y confianza depositada en mi.

¡Este trabajo fue posible por todos ellos!

RESUMEN

En el proceso de control de calidad para el sistema Material Flow, donde se debe realizar diferentes pruebas, seguimientos de errores encontrados, verificación de requerimientos, validación si el sistema cumple con las espectativas e informar del estado del sistema y teniendo en cuenta que el tiempo que se emplea en este proceso es de 16 a 24 horas aproximadamente, se pensó en la automatización de los casos de prueba de las pantallas Management View, Request History, Internal y Exception con el fin de corroborar que su funcionamiento es el esperado, cumple con los requerimientos de los clientes y también reducir el tiempo que se emplea en la ejecución de los casos de prueba.

Para iniciar con el proyecto se utilizó la metodología Ágil Scrum para el desarrollo y proceso del software; en la misma se basó la planificación de los sprints, identificando las fechas de cada tarea y su respectivo responsable. Para el desarrollo de las pruebas automatizadas (scripts) se utilizó la herramienta JMeter con una integración con Jenkins, la última herramienta permite almacenar los casos de pruebas automatizados y ejecutarlas por períodos determinados, después de cada ejecución de las pruebas se obtiene un reporte con el nombre de caso de prueba, el nombre del usuario, el estado (si falló o pasó) y el tiempo que tardo en ejecutarse y para determinar el costo del proyecto se aplicó el modelo de costos COCOMO.

Contenido

I. MARCO REFERENCIAL 1

1.1. Introducción 1

1.2. Antecedentes 2

1.3. Planteamiento del Problema 3

1.4. Formulación del Problema 4

1.5. Fines y propósito 4

1.6. Objetivo General 5

1.6.1. Objetivos Específicos 5

1.7. Justificaciones 6

1.7.1. Justificación Práctica 6

1.7.2. Justificación Ingenieril 6

1.7.3. Justificación Económica 6

1.7.4. Justificación Social 7

1.8. Límites, Alcances y aportes 7

1.8.1. Límites 7

1.8.2. Alcances: 8

1.8.3. Aportes 8

1.9. Modelos y Herramientas 9

II. Marco Teórico 11

2.1. Ingeniería de Software 12

2.1.1. Defectos en el software 13

2.1.2. Pruebas de Software 14

2.2. Metodología para el desarrollo del sistema 16

2.2.1. Metodología Scrum 16

2.3. Automatización de software 17

2.4. Calidad 19
2.4.1. Control de calidad 20
2.4.2. Calidad de software 20
2.4.3. Aseguramiento de la Calidad (QA) 21
2.4.4. Los tipos de Aseguramiento de la Calidad (QA) 22
2.5. Herramientas para el desarrollo del sistema 23
2.5.1. Herramienta para la estimación de costos Cocomo 23
2.5.2. Herramientas de modelado JMeter 26
2.5.3. Herramienta de ejecución de Test case (Jenkins) 32
2.5.4. Herramienta de modelado para el sistema UML 33
III. Marco Institucional 36
3.1. Pantalla de Internal del software Material Flow 39
3.2. Pantalla de Excepciones 42
3.3. Pantalla de Management View 44
3.4. Pantalla de Request History 46
IV. Marco Práctico 49
4.1. Introducción 49
4.2. Análisis de requerimientos 50
4.2.1. Requerimientos Funcionales 50
4.2.2. Requerimientos No Funcionales 51
4.3. Modelado de Negocio 51
4.3.1. Diagrama de casos de uso de las pantallas de automatización del Test Plan 51
4.4. Planificación del sistema 59
4.5. Sprint Product Backlog para el desarrollo de la automatización 60
4.5.1. Primer Sprint – Análisis de los casos de uso para la automatización 64
4.5.2. Segundo Sprint - Script para la Pantalla de Internal 67
4.5.3. Tercer Sprint - Script para la Pantalla de Excepciones 88
4.5.4. Cuarto Sprint - Script para la Pantalla de Management View 104

4.5.5. Quinto Sprint - Script para la Pantalla de Request History 122

4.6. Integración de los script de JMeter con Jenkins 126

4.7. Aplicación de la estimación de coste COCOMO 135

4.7.1. Costo de Implementación 138

4.7.2. Costo de elaboración del proyecto 138

4.7.3. Costo total del Software 139

V. Conclusiones 141

VI. Recomendaciones 144

VII. Bibliografía 145

VIII. Webgrafia 147

Glosario 148

Anexos 149

Índice de Ilustraciones

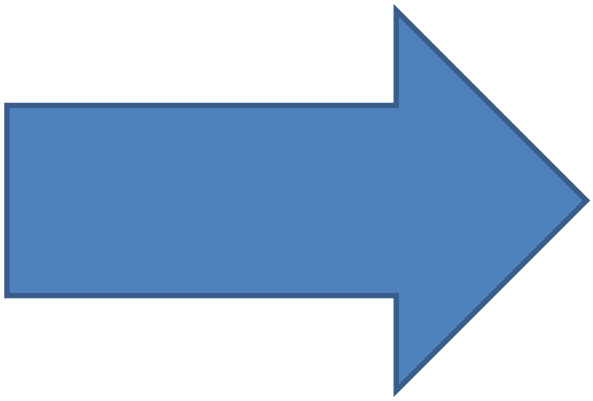
Ilustración 1: Costo de defectos 14
Ilustración 2: Prueba de software 15
Ilustración 3: QA Constructivo 22
Ilustración 4: QA Analítico 23
Ilustración 5: Tabla para la estimación KDLC 25
Ilustración 6: Tabla para el modelo 25
Ilustración 7: Tabla para conductores de coste 26
Ilustración 8: Funcionamiento de JMeter 27
Ilustración 9: Estructura de JMeter 28
Ilustración 10: Activar JMeter por el archivo de lotes 29
Ilustración 11: Configuración de Explore 29
Ilustración 12: Como grabar las llamadas al servidor 30
Ilustración 13: Llamas al servidor 31
Ilustración 14: Componente BeanShell 32
Ilustración 15: Pantalla de internal - primera parte 41
Ilustración 16: Pantalla de Excepción 43
Ilustración 17: Pantalla de Management View 45
Ilustración 18: Pantalla de Request History 47
Ilustración 19 Construcción del script para el proceso de entrega de pedidos 71
Ilustración 20: Archivo de configuración (configuration.properties) base del script 75
Ilustración 21: GetValueOfPropertie (incialización de variables) 76
Ilustración 22: Consulta SQL para obtener lista de pedidos 77
Ilustración 23: Resultado de la consulta 77
Ilustración 24: Procesar variables de la consulta 77
Ilustración 25: Consulta para los request por begin delivery 78
Ilustración 26: Procesar datos en formatos determinados. 79
Ilustración 27: Consulta pare excepción de notificación 79
Ilustración 28: Consulta para excepción de cantidad 80
Ilustración 29: Consulta para los request en la lista de Delivery 80
Ilustración 30: Proceso de datos para la es de delivery y distintos test case 81
Ilustración 31: Tiempo inicial de los test case 81
Ilustración 32: Regex para el análisis de la página 82
Ilustración 33: Consulta para contar los request con excepciones 82
Ilustración 34: Tiempo final y validación de los test case 82
Ilustración 35: Generar el reporte del test case 84
Ilustración 36: Estado inicial de Internal 85
Ilustración 37: Request a completar con excepción de cantidad 86
Ilustración 38: IDs de los request 86
Ilustración 39: Estado final del área de Internal 87
Ilustración 40: Diseño del script para la Pantalla de Excepciones 93

Ilustración 41: Consulta para request en la lista de excepciones 94
Ilustración 42: Request para ser procesados por next request 94
Ilustración 43: Request para procesarlos en cantidades 95
Ilustración 44: Control de tiempo de ejecución inicial 96
Ilustración 45: Control de tiempo y validación del test case 96
Ilustración 46: Regex para obtener la cantidad de la lista de exception 98
Ilustración 47: Generación del reporte 99
Ilustración 48: Configuración para el script de excepciones 100
Ilustración 49: Estado inicial de request en la Pantalla de Excepciones 101
Ilustración 50 :Pick/Delivery begin delivery - Consulta a la base de datos 102
Ilustración 51: IDs de request para pick/delivery 102
Ilustración 52: Estado final de la lista de excepción 103
Ilustración 53: Diseño de script para la Pantalla de Management View 109
Ilustración 54: Configuración Management View 110
Ilustración 55: Lógica para los rangos de fechas 110
Ilustración 56: Lógica para el cambio de prioridades 111
Ilustración 57: Consulta para los test case de búsqueda 112
Ilustración 58: Control de la cantidad de request 112
Ilustración 59: Obtener valores de las consultas 113
Ilustración 60: Regex para el análisis de la página 115
Ilustración 61: Consultas para datos archived 116
Ilustración 62: Guardar los datos obtenidos 116
Ilustración 63: Análisis de la página con los datos de la consulta 117
Ilustración 64: Consulta para el Id's de los requets 118
Ilustración 65: Procesamiento de datos 118
Ilustración 66: Procesamiento de datos, separado por comas 119
Ilustración 67: Generar el reporte 119
Ilustración 68: Consulta a la base datos de los request para cambiar su estado 120
Ilustración 69: Id’s de request para cambiar su estado 121
Ilustración 70: Estado final de Management View 121
Ilustración 71: Request History - Búsqueda por todos los días 125
Ilustración 72: Archivos csv exportados de las pantallas 126
Ilustración 73: Panel de control de Jenkins 127
Ilustración 74: Configuración de conexión 128
Ilustración 75: Proyecto para el script de Internal 128
Ilustración 76: Proyecto para el script de Exception 129
Ilustración 77: Proyecto para la configuración de Management View 129
Ilustración 78: Proyecto para Request History 130
Ilustración 79: Vista de los reportes 130

Índice de Tablas

Tabla 1: Proceso de control de calidad en Material Flow 38
Tabla 2: Casos de prueba para la Pantalla Internal 41
Tabla 3: Casos de prueba para la pantalla Exception 43
Tabla 4: Casos de prueba para Management View 45
Tabla 5: Casos de prueba para Request History 47
Tabla 6: Requerimientos Funcionales 50
Tabla 7: Requerimientos no funcionales 51
Tabla 8: Caso de uso del Pantalla Internal 53
Tabla 9: Tabla del proceso de ejecución de Management View 55
Tabla 10: Tabla del proceso de Management View 57
Tabla 11: Caso de uso de Request History 58
Tabla 12: Usuarios para el sistema de automatización 59
Tabla 13: Sprint Product Backlog 60
Tabla 14: Tabla de Sprint Backlog 61
Tabla 15: Primer sprint 64
Tabla 16: Primer sprint - Requerimientos para los scripts 64
Tabla 17: Análisis de requerimientos para el primer sprint 65
Tabla 18: Lista de los casos de prueba 65
Tabla 19: Segundo sprint 67
Tabla 20: Segundo Script - Pantalla Internal 67
Tabla 21: Descripción de caso de uso para la pantalla de Internal 69
Tabla 22: Lista de componentes de JMeter para los scripts 72
Tabla 23: Tercer sprint 88
Tabla 24: Tercer Sprint - Pantalla de Excepciones 88
Tabla 25: Descripción del diagrama de caso de uso 90
Tabla 26: Cuarto Sprint 104
Tabla 27: Cuarto Sprint - Pantalla Management View 104
Tabla 28: Descripción de los casos de uso 106
Tabla 29: Quinto Sprint 122
Tabla 30: Quinto Sprint - Pantalla Request History 122
Tabla 31: Descripciones de los test case para automatizar 122
Tabla 32: Lista de test case y comparación de tiempos 132
Tabla 33: Test case automatizados por Pantalla 134
Tabla 34: Comparación de tiempos para las pruebas 135
Tabla 35: Tiempo reducido con la automatización 135
Tabla 36: Costo de elaboración del proyecto 138
Tabla 37: Costo total del software 139
Tabla 38: Tiempos de test case automatizado 141
Tabla 39: Cumplimiento de requerimientos 142

CAPÍTULO I



"La gente habla de conseguir golpes de suerte en sus carreras. Yo soy la prueba de que la teoría de los golpes de suerte es simplemente errónea. Uno consigue su propia suerte. El mundo está dirigido por los que se dan a conocer, no por los que esperan a que se pregunte por ellos."

Steve Blank

Capítulo I

En el presente capítulo se detallara las razones por las cuales se llevo acabo el proyecto de grado también se especifica que es el sistema Material Flow y su respectivo proceso de control de calidad.

I. MARCO REFERENCIAL

1.1. Introducción

Según la ISTQB (International Software Testing Qualifications Board) que es una organización encargada de la certificación internacional de control de software, establece guías para la acreditación y evaluación de los profesionales de testing, hasta el 2017 se ha certificado a más de 535000 testers en 120 países. También nos muestran los procesos de pruebas, razones del porque se realizan las pruebas para la calidad de software, los resultados que se obtienen se esperan que cumplan con los requerimientos de funcionamiento, seguridad o integración según sea el caso.

Material Flow al ser un sistema con 10 años de desarrollo llego a convertirse en un sistema complejo, al momento de realizar el control de calidad los casos de prueba son alrededor de 261 para la version 3.88, sin embargo se incrementaran en una siguiente versión por las nuevas funcionalidades según requerimientos al igual que podrian llegar a ser dependientes de otras o que afecten a funcionalidades anteriores. Al tener muchos casos de uso para el respectivo testing y al ser ejecutados de forma manual se convierte en un proceso ineficiente por que este proceso no es constante ni consistente, ademas que se

invierte tiempo en la capacitación de un nuevo integrante del área de control de calidad y la ejecución del test por parte del mismo puede llegar a ser diferente teniendo como consecuencia resultados no certeros al 100%, una ejecución de baja calidad, un retrazo de software o bien se tenga un alto costo de desarrollo del mismo sistema.

1.2. Antecedentes

Existen otros proyectos parecidos o realizados bajo el mismo concepto.

“Automatización del proceso de codificación y funcionalidad del sistema SAS basado en el uso de la herramienta rational functional tester, aplicado a la empresa aseguradora Rimac seguros y reaseguros” por Luis Miguel Peralta, Lima – Peru 2014 Facultad de Ingenieria de Sistemas y Electronica, el proyecto consiste en una herramienta en la que se automatice el ciclo de vida de aplicación de tal manera que se pueda reducir el tiempo de ejecución. (Peralta, 2014)

“Sistema web de gestión academica y repositorio virtual caso: Unidad Educativa Principe de Luz” por Edwin Ruben Cachaga Villegas La Paz – Bolivia 2015 Universidad Mayor de San Andres, el proyecto consiste en efectivizar el tiempo en el que se procesa la información de Kardex y la libreta digital de tal forma que el docente tenga el control de llenado de las calificaciones y el seguimiento académico. (Villegas, 2015)

“Automatizacion de casos de prueba y pruebas de stress” por Alvaro Rolando Pedregal Guzman La Paz – Bolivia 2014 Universidad Mayor de San Andres, el proyecto consiste certificacion de software por medio de la automatizacion de los casos de pruebas con agentes inteligentes, teniendo en cuenta tambien un test de stress para el software. (Guzman, 2014)

1.3. Planteamiento del Problema

El sistema Material Flow es desarrollado por equipo Flow en NVD (New Vision Data), este equipo se encuentra dividido en dos áreas internas las cuales son: áreas de desarrollo quienes están encargados del desarrollo de las nuevas funcionalidades, del soporte para todo el sistema para corrección de fallas funcionales del sistema Material Flow y el área de control de calidad que tiene como principal objetivo llevar acabo de las respectivas pruebas (testing) funcional que consiste en probar el funcionamiento correcto de todas las pantallas, ingresando al sistema datos de entrada y examinando los de salida, el propósito de la prueba funcional es mostrar discrepancias con los requerimientos sobre alguna nueva funcionalidad, luego de que el sistema pase esta etapa se aplica la prueba de estrés y su objetivo es mostrar el comportamiento del sistema y la base de datos con varios usuarios concurrentes.

Los del área de control de calidad elaboran el plan de pruebas en el que se detallan todos los casos de prueba que deben ser probados en el sistema Material Flow, estos casos de pruebas son en base a las nuevas y antiguas funcionalidades, al ser un sistema complejo la forma de control que tienen es por medio de un repositorio con las aplicaciones tortoise[1] y subversion[2], después del desarrollo e implementación de un nuevo requerimiento se realizan una nueva versión para el respectivo testeo el cual dura entre 16 a 24 horas (tiempo aproximado en ejecutar todo el test plan, en reportar errores de funcionalidad (bugs[3]), en obtener los resultados con una persona capacitada pero con poca experiencia y conocimiento el sistema Material Flow, y de 10 a 16 horas (tiempo aproximado en ejecutar todo el plan de pruebas, en reportar bugs, en obtener los resultados). La funcionalidad más importante es la entrega de los pedidos necesarios para el ensamblaje de los automóviles, en la pantalla de Internal es

[1] Tortoise: es un cliente Subversion de licencia libre, ayuda a enviar archivos al repositorio.
[2] Subversion: es un sistema para el control de versiones de un sistema.
[3] Bugs: error encontrado en una funcionalidad o diseño

donde el usuario puede ver los pedidos que debe entregar, en la pantalla de Exception se muestran los pedidos que por alguna razón no podrá ser entregado completamente, en las pantallas de Management View y Requests history también se ve los pedidos pero se observa toda la información del pedido (mayor información se encuentra en el capítulo de marco institucional), por lo tanto si estas pantallas dejaran de funcionar la planta de ensamblaje podría detenerse teniendo como consecuencia la perdida de registros, al igual que recurso económicos, se estima que si en una planta de Jaguar Rover Land[4] se detiene se pierde aproximadamente 10385.57 dolares por hora, el cual es una perdida cuantiosa para una empresa de esta magnitud vale decir que no es una opción el de reiniciar el servidor.

1.4. Formulación del Problema

¿La automatización del plan de pruebas, para el software Material Flow, utilizando la herramienta JMeter, permite reducir el tiempo de procesamiento de los casos de prueba en las pantallas de “internal, request history, management view y exceptions”?

1.5. Fines y propósito

El fin por el cual se realizara la automatización de los casos de prueba con la herramienta JMeter (la cual graba las llamadas al servidor que el sistema realiza en el momento en que se lo usa) en las pantallas “internal, request history, management view y exceptions”, es para su ejecución y eliminar su proceso de testing [5]manual con el propósito de aumentar la calidad de ejecución, evitar los retrasos de entrega de software de manera que se tenga una mayor certeza en los resultados que se obtendrán después de la ejecución de los scripts.

[4] Jaguar Rover Land: Compañia inglesa de uutomoción con sede en Whitley, Conventry, West Midelands e Inglaterra, con la actividad de diseño, desarrollo y fabricación de automoviles.
[5] Testing: ejecución de casos de prueba

1.6. Objetivo General

Automatizar los casos de prueba de Material Flow, en New Vision Data, para reducir el tiempo de ejecución de los casos de pruebas en al menos 60%, en las pantallas de “internal, request history, management view y exception”, aplicando las herramientas JMeter[6] y Jenkins[7].

1.6.1. Objetivos Específicos

- Identificar los casos de prueba de las diferentes pantallas Management View, Request History, Internal y Exception, tomando en cuenta la estructura de desarrollo del script para que su respectivo mantenimiento sea rápido y eficaz.
- Elaboración de los scripts de automatización con la herramienta JMeter para los casos de prueba.
- Implementar los script para los casos de prueba en Material Flow para observar su comportamiento.
- Elaborar un archivo con extención csv[8] el que sea el reporte de cada script, con la información si los casos de prueba fueron exitosos o fallidos.

[6] JMeter: herramienta para la grabación de llamadas al servidor para la automatización de casos de prueba
[7] Jenkins: software de integración continua open source escrito en java.
[8] CSV: archivo sencillo donde las columnas se separan por comas.

1.7. Justificaciones

1.7.1. Justificación Práctica

El desarrollo del proyecto de grado se efectuara porque existe la necesidad de mejorar la ejecución del test plan y reducir el tiempo en el que se efectúa el testing de los casos de uso de las pantallas de internal, request history, management view y exceptions.

1.7.2. Justificación Ingenieril

Con la utilización de Scrum y automatización de las pantallas de internal, exception, management view y request history se reducirá el tiempo de ejecución del plan de pruebas, también el testeo será mucho más efectivo, consistente y requiera menor esfuerzo de un nuevo e incluso para un antiguo integrante en el área de control de calidad, de tal forma que ellos puedan concentrarse en otras tareas, por lo tanto se estaría optimizando el proceso de prueba al sistema de tal manera que se ayuda al área de control de calidad.

1.7.3. Justificación Económica

El proyecto de grado se justifica económicamente por que permitirá optimizar el tiempo de prueba del sistema Flow de tal manera que se simplifique este proceso para el encargado de control de calidad, ayudando a almacenar los resultados que proveerá la ejecución de los script para los casos de uso de las pantallas de internal, request history, management view y exception.

En cuanto a las herramientas que se aplicaran para el desarrollo de los scripts son de licencia libre por lo cual reduce los costos, sin embargo el costo de personal para la creación de los scripts de las pantallas se explica en el marco teórico la aplicación de COCOMO[9] para la estimación de costos.

[9] COCOMO: herramienta para la estimación de costos del sistema a desarrollar.

1.7.4. Justificación Social

Con la automatización de los casos de prueba de las pantallas de internal, request history, management view y exception, pertenecientes al sistema Material Flow es de tener un constante y optimo control de calidad del sistema completo teniendo en cuenta que se añaden funcionalidades nuevas. De igual forma que facilite al encargado de control de calidad para obtener resultados más exactos en cuanto a la ejecución de los casos de prueba. El personal se beneficiara por que no tendrá que realizar trabajos monótonos y este tiempo podrá ser empleado en funciones que sean prioritarias.

1.8. Límites, Alcances y aportes

1.8.1. Límites

- Para poder realizar la automatización de los casos de prueba en las pantallas Management View, Request History, Internal y Exception del sistema Material Flow deben de tener una funcionalidad madura y si sufren cambios estos no deben influir en su funcionalidad funcionalidad, caso contrario el script deberá ser modificado o crear uno nuevo.
- Los resultados obtenidos, podrían no ser los esperados por una mala configuración previa a la ejecución de los scripts.
- Restricciones de ejecución de los scripts, podrán ser usados siempre y cuando no se cambie el caso de prueba, si cambian los casos de prueba el script deberá ser actualizado para la versión de Material Flow correspondiente.

1.8.2. Alcances:

Con el presente proyecto de grado se estable:

- Cumplir con el objetivo principal de realizar la automatización de los casos de prueba de las pantallas internal, request history, management view, y exception aplicando como herramienta principal a JMeter, para la creación de los scripts se debe conocer el funcionamiento y ejecución de correcto de los casos de prueba.
- Con la elaboración de la automatización de los casos prueba se quiere tener una plataforma tecnológica que ayude al aumento de calidad de ejecución del plan de pruebas en las pantallas Management View, Request History, Internal y Exception, en el sentido de monitoreo y con control del sistema Material Flow.
- Para la automatización de las pruebas será en base a la versión estable de sistema Material Flow.
- Con la finalización de la automatización de los casos de prueba deberán ayudar al testeo de nuevas versiones de Material Flow con las pantallas ya automatizadas.
- Se tiene 79 casos de prueba en total de las cuatro pantallas de los cuales es prioridad automatizar por ahora 48 casos de prueba.

1.8.3. Aportes

Para poder realizar la automatización respectiva de los casos de prueba se aplicará la herramienta JMeter para simular el funcionamiento respectivo y llegar al mismo resultado si la prueba se ejecutara manualmente, tomando en cuenta los siguientes puntos:

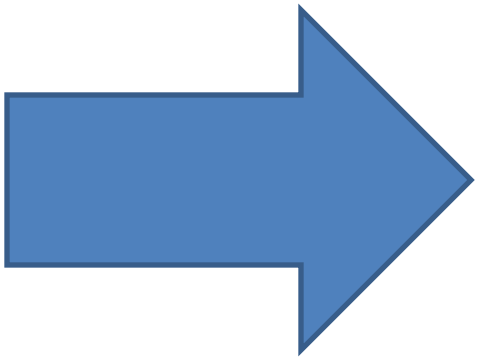
- Reducción del tiempo de manera que sea mucho más efectivo la ejecución del test plan de forma global.
- Control de calidad del software agil y eficiente.

- Generación de reportes de manera automática después de cada ejecución.
- Fusión de dos herramientas Jmeter y Jenkins para que sea posible la automatización.

1.9. Modelos y Herramientas

- La metodología de investigación para el proyecto es el Analítico que nos ayudara a desglosar el caso de estudio y llegas a puntos específicos que permitirán la automatización.
- La metodología de desarrollo del proyecto es el Scrum ya que al ser una metodología ágil se acopla a los requisitos del sistema a desarrollar,
- Las herramientas que se utilizaran son JMeter que nos permite grabar los scripts necesarios para los pasos de prueba y también se usara Jenkins, es una herramienta con la que podremos ejecutar los scripts de manera automática, y para la estimación de costes será el COCOMO.

CAPÍTULO II



"La muerte es probablemente el único buen invento de la vida. Es el agente de cambio de la vida; borra lo antiguo para dejar paso a lo nuevo. Y ahora lo nuevo son ustedes. Pero algún día, no dentro de mucho, se convertirán en los viejos y serán reemplazados. Siento ser tan dramático, pero es la verdad. Su tiempo es limitado así que no lo malgasten viviendo la vida de otra persona."

Steve Jobs

Capítulo II

En el presente capítulo se detalla y especifica en que consiste el proceso de control de calidad, para que sirve la automatización de casos de pruebas, las herramientas que se utilizo para el desarrollo del proyecto.

II. Marco Teórico

En este capítulo se mostraran los conceptos de calidad, control de calidad, calidad de software y automatización, actualmente es posible aplicar control de calidad a cualquier software y el encargado de control de calidad es quien juzga si el software cumple con las necesidades del cliente, para llegar a un resultado confiable es necesario llevar un proceso estricto por lo que vale decir si se quiere aplicar control de calidad a cualquier software es necesario seguir los siguientes pasos:

- ✓ Conocer el sistema que se quiere probar.
- ✓ Tener en cuenta los requerimientos.
- ✓ Identificar los casos de pruebas y cuál es el impacto de cada uno de ellos en el sistema.
- ✓ Crear un plan de pruebas con los casos de prueba.
- ✓ Ejecutar los casos de prueba.
- ✓ Si se encuentra fallos o si las pruebas devuelven resultados que no se esperan, se deben de reportar y dar a conocer como replicar esos errores.
- ✓ Dar seguimientos a los errores encontrados.
- ✓ Dar a conocer el estado del sistema al finalizar la ejecución del plan de pruebas. Duarte (2015)

Después de aplicar el proceso de calidad se debe evaluar si es adecuado aplicar una automatización de los casos de prueba al sistema en cuestión y que tipos de automatización seria adecuada, el tipo en ocasiones se define en función de los casos de pruebas y requerimientos del cliente, es decir para pruebas de funcionalidad, performance o stress, pruebas de visualización de componentes, para todos esos tipos de pruebas existen diferentes herramientas que se pueden emplear. Esteve Ambrosito (2014 - 2015)

- JMeter con esta herramienta se pueden realizar pruebas de funcionalidad y de stress. Apache (2017)
- Robot framework con esta herramienta es posible realizar pruebas de funcionalidad y visualización. Robot Framework (2018)
- Appium herramienta para pruebas de funcionalidad para aplicaciones Android. Appium (2018)

Como anteriormente se detallo que no todos los sistemas son aptos para una automatización, es decir es recomendable para sistemas que están constante desarrollo y se mantiene la funcionalidad base (para el encargado de control de calidad este proceso se vuelve moroso y molestoso por que técnicamente debería estar funcionado) y lo que se hace es añadir nuevas funcionalidades lo que es primordial probar en la nueva versión, también es bueno mencionar que no todos los casos de pruebas son aptos para una automatización por las características que tenga es decir que para su ejecución depende a otra herramienta (el cual tal vez no pueda ser emulado de forma automática).

2.1. Ingeniería de Software

La ingeniería de software es el proceso de:

- Analizar las necesidades del usuario
- Diseño
- Construcción

- Probar aplicaciones

Satisfacer las necesidades mediante el uso del lenguaje de programación de software. Es la aplicación de principios de ingeniería de desarrollo de software, a diferencia de la programación simple, es que se utiliza para la creación de sistemas grandes y más complejos que se usan como sistemas críticos para empresas y organizaciones Software Testing Help (2018), el concepto de software es el siguiente:

- ✓ John W. Tukey – 1957: Software es toda la información procesada por los sistemas informáticos programas y datos.
- ✓ Charles Babbage: Leer diferentes secuencias de instrucciones desde una memoria de un dispositivo para controlar los cálculos.
- ✓ Alan Turing – 1936: Una aplicación al problema de decisión.

2.1.1. Defectos en el software

Los fallos en el momento de la ejecución del software provocan que disminuya el éxito del sistema por lo tanto la confiabilidad y calidad del mismo es comprometido, sin embargo las causas para que ocurran son las siguientes Bubano (2010):

- Error humano el cual suelen ser un defecto que es introducido en el código de software en caso de los desarrolladores, o defectos en los datos o parámetros de configuración.
- Cambios en las condiciones ambientales, como ser radiación, magnetismo, campos electromagnéticos y polución, manchas solares, fallos de disco duro, fluctuaciones en el suministro de energía eléctrica.

Definiciones de error, defecto o fallo son los siguientes:

- ✓ Error (IEEE 610): Acción humana que produce un resultado incorrecto por ej. Un error de programación.

- ✓ Defecto (Defect): Desperfecto en un componente o sistema que puede ser una causa por la cual el sistema o componente no logre llevar o cabo su función específica, por ej. Sentencia o definición de datos incorrectas.
- ✓ Fallo (Failure): Manifestación física o funcional de un defecto. Si un defecto es encontrado durante la ejecución de una aplicación puede producir un fallo. Desviación de un componente o sistema respecto de la prestación, el servicio o resultado esperados.

Ilustración 1: Costo de defectos



Fuente: ISTQB El costo de los defectos

2.1.2. Pruebas de Software

Las pruebas de software son una parte importante el del desarrollo del software ya que es un conjunto de condiciones o valores bajo los cuales se determina si el software es cumple con las especificaciones o requisitos funcionales y no funcionales, es una serie de pasos que comprueban el

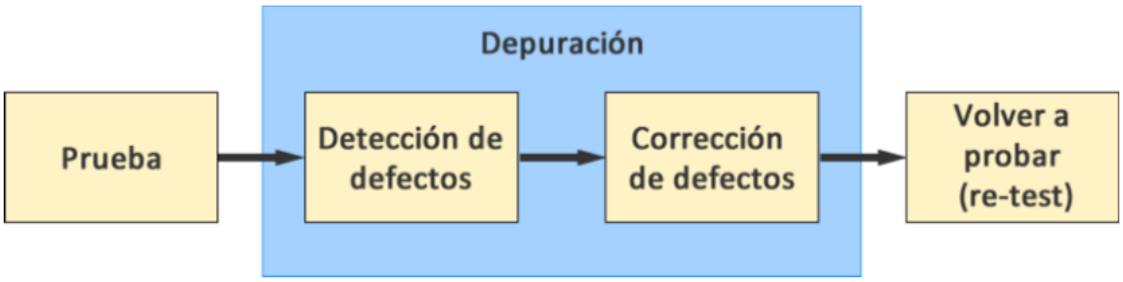
comportamiento correcto de una pantalla en específico, Boris Bezer 1995 define una prueba como:

> "Una secuencia de uno o más subtests ejecutados como una secuencia por que el resultado y/o el estado final de una sub-prueba es la entrada y/o o el estado inicial de la siguiente. La palabra "prueba" se usa para incluir subtests, pruebas propias y suites de prueba."

Los tipos de pruebas de software según la ISTQB son las siguientes:

- ✓ **Pruebas funcionales**, las cuales pueden estar escritas o acordadas en los requerimientos, características en el comportamiento del sistema, es considerado como pruebas de caja negra.
- ✓ **Pruebas no funcionales,** son pruebas de carga, usabilidad, mantenibilidad, confiabilidad, portabilidad el tiempo de respuesta de igual manera pruebas de caja negra.
    - Pruebas de arquitectura de software
    - Pruebas de regresión y re-prueba
    - Pruebas de mantenimiento

Ilustración 2: Prueba de software



Fuente: ISTQB prueba de software

2.2. Metodología para el desarrollo del sistema

2.2.1. Metodología Scrum

Scrum es una metodología en el que se aplican buenas prácticas para poder trabajar de manera colaborativa en un equipo teniendo como resultado un trabajo de equipo altamente productivo, ya que se realiza entregas parciales del producto final Peralta A. (s.f.). Actualmente en Material Flow se tiene aplicada de cierta manera esta metodología, en la que todas las mañanas se realizan reuniones sobre los últimos trabajos que cada integrante del equipo realizo día anterior y lo que hará en el transcurso del día presente.

Con la metodología Scrum el cliente especifica los requisitos o características del sistema que debe desarrollar, las personas encargadas de su desarrollo deben entregar adelantos funcionales del sistema con fin de que el cliente pueda observar o tener en cuenta que sus especificaciones se estén cumpliendo o puedan cambiarse, adaptarse o añadirse Ingeniería industría innovación (s.f.). Los beneficios de la metodología Scrum según Gallego, Metodología Scrum (s.f.) son las siguientes:

- ✓ Cumplimiento de expectativas
- ✓ Flexibilidad a cambios
- ✓ Reducción de time to Market
- ✓ Mayor calidad de software
- ✓ Mayor productividad
- ✓ Maximizar el retorno de la inversión
- ✓ Predicciones de tiempos
- ✓ Reducción de riesgos

Los roles de los que se componen la metodología Scrum son tres según Kniberg (2007) :

- **Product Owner:** es el representante de todos los interesados en el producto final, quien marca las prioridades del producto, tiene el control de las estimaciones y el retorno de la inversión.
- **ScrumMaster:** es quien transforma las tareas del Sprint Backlog en un incremento funcional del software, de tal manera que sea de calidad, autogestionado, auto-organizado, multifuncional y no mayor a 8 elementos.
- **Scrum Team:** es el encargado de incorporar la cultura de Scrum a la organización, asegurar el cumplimiento de los roles y resposabilidades, formación y entrenamiento en el proceso. (pág. 8)

2.3. Automatización de software

En nuestra actualidad, en el cual se quiere adelantarse a la competencia, en las exigencias para el usuario final en cuestión de que sea rápido, muestre información necesaria y concisa o que los negocios quieren ser más agiles de tal manera que se pueda sacar al mercado productos más rápidos y de buena calidad, para todo ellos es necesario aplicar procesos de automatización en las áreas industriales o sistema que usen pero no significa que se elimine por completo la intervención del recurso humano.

El concepto de automatización “Es un sistema donde transfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.” Canto (s.f.)

Las ventajas de la automatización además del ahorro del tiempo del tiempo, suele favorecer la precisión en el desarrollo de tareas como aspecto negativo, al menos en el plano social las maquinas pueden reemplazar a personas que podrían perder su empleo o deben ser movidas a otras pantallas y capacitarlos, pero otras ventajas y/o objetivos según Alvares, (s.f.) son las siguientes:

- ✓ Sirve para que los trabajos más complicados o peligrosos no sean realizados por los trabajadores sino por las maquinas.
- ✓ Mejorar y aumenta notablemente la producción.
- ✓ Permite que tareas por su minuciosidad o complejidad, no puedan ser desarrolladas por personas sean acometidas por maquinas.
- ✓ Contribuye a mejorar lo que es la seguridad de los empleados.
- ✓ Minimiza el esfuerzo del ser humano. Automatización del test plan para los casos de prueba

La automatización del plan de pruebas es recomendable según sea el caso o la magnitud del sistema, si es grande y se tienes muchas pantallas un cambio podría provocar que ocurran errores de diseño y funcionamiento, con el uso adecuado de una herramienta de automatización es posible grabar la secuencia de los casos de prueba y que en el futuro puedan ser usados de manera consecutiva y secuencial en función a los cambios ocurridos.

Las pruebas automatizadas de software son importantes debido a las siguientes razones según Guru99 (2018); Software Testing Help (2018):

- ✓ Prueba manual de todos los flujos de trabajo, todos los campos, todos los escenarios negativos son el tiempo y el costo que consume.
- ✓ Es difícil probar sitios multilingües manualmente.
- ✓ La automatización no requiere intervención humana. Puede ejecutar una prueba automatizada sin supervisión (durante la noche).
- ✓ La automatización aumenta la velocidad de ejecución de prueba.
- ✓ Las pruebas manuales pueden volverse aburridas y por lo tanto propensas a los errores.

Un punto importante y también crucial es elegir con cuidado que pruebas son necesarios y aptos para ser automatizados, de tal manera que retorne la inversión de la automatización según Guru99 (2018); Software Testing Help (2018):

- ✓ Casos de prueba de alto riesgo – Business critical
- ✓ Casos de prueba que se ejecutan repetidamente.
- ✓ Casos de prueba que son muy tediosos o difíciles de realizar manualmente.
- ✓ Casos de prueba que consumen mucho tiempo.

2.4. Calidad

Calidad es una herramienta básica y una propiedad importante que nos permite comparar el servicio o producto con los de su misma especie, el significado de calidad puede llegar a ser distintos según un ámbito específico en algunos casos es satisfacer necesidades implícitas o explicitas, en otro aspecto desde de la percepción cliente final si está conforme o satisface sus necesidades. Maria Constanza Cubillos Rodriguez (s.f.)

Algunas otras definiciones más formales sobre calidad son las siguientes:

- ✓ Según la norma **ISO 9000**: "Calidad: grado en el que un conjunto de características inherentes a un objeto (producto, servicio, proceso, persona, organización, sistema o recurso) cumple con los requisitos."
- ✓ **Real Academia de la Lengua Española**: "Propiedad o conjunto de propiedades inherentes a una cosa que permiten apreciarlas como igual, mejor o peor que las restantes de su especie".
- ✓ **Philip Crosby**: "Calidad es cumplimiento de requisitos".
- ✓ **Joseph Juran**: "Calidad es adecuación al uso del cliente".
- ✓ **Armand V. Feigenbaum**: "Satisfacer las necesidades del cliente".
- ✓ **Genichi Taguchi**: "Calidad es la perdida (monetaria) que el producto o servicio ocasiona a la sociedad desde que es expedido".
- ✓ **Walter A. Shewhart**: "Calidad es como resultado de la interacción de dos dimensiones subjetiva lo que el cliente quiere y dimensión objetiva (lo que se ofrece)".

- ✓ **Instituto Centroamericano de tecnología Industrial – ICAITI-COPANT-ISO 8402-1995:** “La totalidad de las características de una entidad que le confieren la aptitud para satisfacer las necesidades establecidas y las implícitas”.

2.4.1. Control de calidad

Es el seguimiento detallado de los procesos dentro de una empresa para mejorar la calidad del producto o servicio que se provee a un cliente, todo consiste en la implantación de programas, mecanismos, herramientas o técnicas en una empresa para la mejora de la calidad de sus productos, servicios y productividad. Blas Zurro (2014)

El control de calidad es una estrategia para asegurar el cuidado y la mejora continuidad de la calidad ofrecida. Besterfield (2009)

Establecer un control de calidad significa ofrecer y satisfacer a los clientes al máximo y conseguir los objetivos de las empresas, por lo tanto debe aplicarse a todos los procesos de la empresa, teniendo en cuenta los estándares de calidad que el mercado espera se podrá controlar cada proceso hasta la obtención del producto o servicio incluso hasta la distribución. Besterfield (2009)

2.4.2. Calidad de software

Hoy en día el rendimiento es uno de los principales determinantes de la calidad de software, ya que es uno de los principales factores que la experiencia del usuario sea exitoso, todo es una consecuencia de los avances en tecnología y la complejidad en la arquitectura de datos, lo que se debería tener en cuenta son los siguientes puntos:

- La importancia de una correcta gestión de los requerimientos con el fin de proveer un software de calidad, la cual no solo es el cumplimiento de las expectativas del usuario sobre lo que debería hacer el software, si no que

también es el análisis en todo lo que deberías tener en cuenta a la hora de definirlos y como se gestionaran de manera colaborativa con todos los miembros del equipo. Ingenieria de Sistemas (2004)

- Las pruebas que se aplican a un software generalmente se situaba al final del proceso de desarrollo o cuando ya se tiene un parte funcional. También control de calidad no solo trata de encontrar errores, si no que debe tener un perfil más técnico ya que está implicado en la definición de todo proceso de integración continua, del control de versiones, pasando por los reléase y acabando por despliegues. Castro (2008)
- Las pruebas son necesarias para la mejora de calidad del software al igual que en el proceso de desarrollo. Hernandez (2008)

La definición de calidad de software son los siguientes:

- ✓ Definición: Software (según IEEE 610): Programas de computadora, procedimientos, y posiblemente documentación asociada así como los datos relacionados con la operación de un sistema Informático.
- ✓ Definición: Calidad software (según ISO/IEC 9126): La totalidad de la funcionalidad y características de un producto software que contribuyen a su habilidad de satisfacer necesidades especificadas o implícitas
- ✓ Definición: Calidad (según IEEE Std 610): Grado en el cual un componente, sistema o proceso satisface los requisitos especificados y/o necesidades del usuario/cliente y sus expectativas.

2.4.3. Aseguramiento de la Calidad (QA)

El aseguramiento de la calidad (QA) consiste en el seguimiento de unas líneas de actuación planificada y sistemática, implantadas dentro del Sistema de gestión de Calidad de la empresa. Estas acciones deben ser demostrables con el objeto de proporcionar la confianza adecuada, tanto a la propia empresa como a los clientes y proveedores. Bubano (2010)

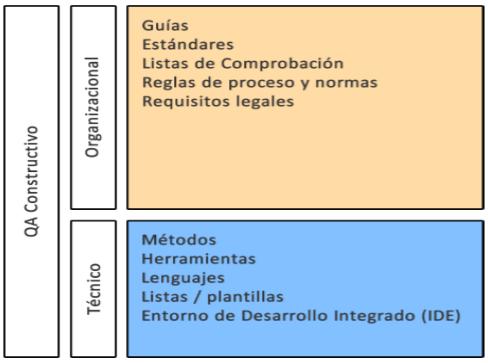
Según la norma ISO/IEC 9126 la calidad de software está constituida por:

- ✓ Funcionalidad
- ✓ Fiabilidad
- ✓ Usabilidad
- ✓ Eficiencia
- ✓ Mantenibilidad
- ✓ Portabilidad

2.4.4. Los tipos de Aseguramiento de la Calidad (QA)

Actividades constructivas con el objeto de prevenir defectos, por ejemplo a través de la aplicación de métodos apropiados de ingeniería de software. Bubano (2010)

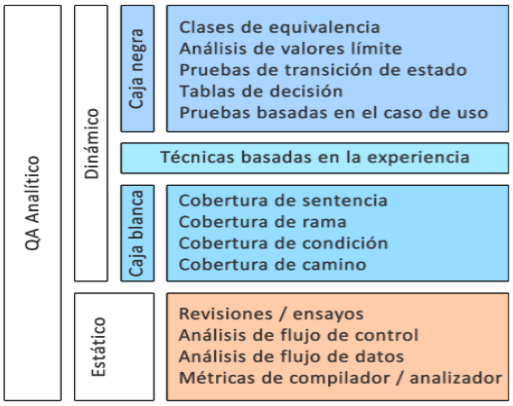
Ilustración 3: QA Constructivo



Fuente: ISTQB QA Constructivo

Actividades analíticas con el objeto de prevenir defectos. Por ejemplo a través de pruebas que conducen a la corrección de defectos y prevención de fallos, incrementando así la calidad del software.Burtano, N. (2010). *ISTQB (traducido).*

Ilustración 4: QA Analítico



Fuente 1: ISTQB QA Analítico

2.5. Herramientas para el desarrollo del sistema

2.5.1. Herramienta para la estimación de costos Cocomo

El modelo de COCOMO (Constructive Cost Model) desarrollado por Barry M. Boehm, se engloba en el grupo de los modelos algorítmos se tratan de establecer una relación matemática la cual permite estimar el esfuerzo y tiempo requerido para desarrollar un producto. Valerdi (2007)

En el modelo de COCOMO define 3 modos de desarrollo o tipos de proyectos Gutierrez (2010):

- Orgánico: proyectos relativamente sencillos, menores de 50 KDLC (KDLC=Cálculo de esfuerzo en función del tamaño estimado) líneas de códigos, en los cuales se tiene experiencia de proyectos similares y se encuentran en entornos estables.
- Semi-acoplado: proyectos intermedios en complejidad y tamaño (menores de 300 KDLC), donde la experiencia en este tipo de proyectos es variable, y las restricciones intermedias.

- Empotrado: proyectos bastante complejos, en los que apenas se tien experiencia y se engloban en un entorno de gran innovación técnica. Además se trabaja con unos requisitos muy restrictivos y de gran volatilidad.

Por otro lado existen diferentes modelos que define COCOMO Gutierrez (2010):

- Modelo básico: Se basa exclusivamente en el tamaño expresado en LDC (línea de código orientada al tamaño).
- Modelo intermedio: Además del tamaño del programa incluye un conjunto de medidas subjetivas llamadas conductores de costes.
- Modelo avanzado: Incluye todo lo del modelo intermedio además del impacto de cada conductor de costo en las distintas fases de desarrollo.

Las fórmulas que se usaremos para el modelo intermedio que se tiene son las siguientes:

- **E=Esfuerzo= aKLDC^{e} * FAE(**persona por mes**)**
- **T=Tiempo de duración del desarrollo=cEsfuerzo^{d}(**meses**)**
- **P=Personal=E/T(**personas**)**

La tabla que usaremos para determina el esfuerzo es la siguiente:

Ilustración 5: Tabla para la estimación KDLC

LENGUAJE	LDC/PF
Ensamblador	320
C	150
COBOL	105
Pascal	91
Prolog/LISP	64
C++	64
Visual Basic	32
SQL	12

Fuente: estimación COCOMO

Para determinar el modelo del proyecto usaremos la siguiente tabla:

Ilustración 6: Tabla para el modelo

PROYECTO SOFTWARE	a	e	c	d
Orgánico	3,2	1,05	2,5	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	2,8	1,20	2,5	0,32

Fuente: estimación COCOMO

Y por otro lado también hemos de hallar la variable FAE, la cual se obtiene mediante la multiplicación de los valores evaluados en los diferentes 15 conductores de coste que se observan en la siguiente tabla:

Ilustración 7: Tabla para conductores de coste

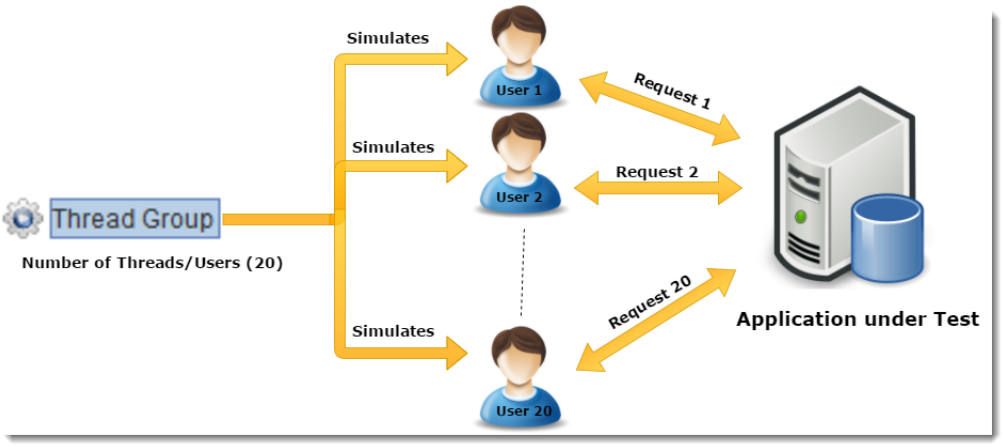
CONDUCTORES DE COSTE	VALORACIÓN					
	Muy bajo	***Bajo***	***Nominal***	***Alto***	***Muy alto***	***Extr. alto***
Fiabilidad requerida del software	0,75	0,88	1.00	**1,15**	1,40	-
Tamaño de la base de datos	-	0,94	**1.00**	1,08	1,16	-
Complejidad del producto	0,70	**0,85**	1.00	1,15	1,30	1,65
Restricciones del tiempo de ejecución	-	-	1.00	**1,11**	1,30	1,66
Restricciones del almacenamiento principal	-	-	**1.00**	1,06	1,21	1,56
Volatilidad de la máquina virtual	-	0,87	**1.00**	1,15	1,30	-
Tiempo de respuesta del ordenador	-	0,87	1.00	**1,07**	1,15	-
Capacidad del analista	1,46	1,19	1.00	**0,86**	0,71	-
Experiencia en la aplicación	1,29	1,13	1.00	0,91	**0,82**	-
Capacidad de los programadores	1,42	1,17	1.00	0,86	**0,70**	-
Experiencia en S.O. utilizado	1,21	1,10	**1.00**	0,90	-	-
Experiencia en el lenguaje de programación	1,14	1,07	1.00	**0,95**	-	-
Prácticas de programación modernas	1,24	1,10	**1.00**	0,91	0,82	-
Utilización de herramientas software	1,24	1,10	1.00	**0,91**	0,83	-
Limitaciones de planificación del proyecto	1,23	**1,08**	1.00	1,04	1,10	-

Fuente: estimación COCOMO

2.5.2. Herramientas de modelado JMeter

Es la principal herramienta para la automatización de los casos de prueba, con la cual se puede realizar pruebas funcionales es decir que el sistema responda de manera correcta o pruebas de performance donde se observara el comportamiento de Matrial Flow cuando tiene 5, 10, 20 o más usuarios ingresan al sistema y realizan una misma actividad. Guru 99 (s.f.)

Ilustración 8: Funcionamiento de JMeter



Fuente: JMeter ToolsQA

JMeter es un programa en base Java diseñada para hacer pruebas de estrés de servidores, aplicaciónes web también puede ser usada para pruebas unitarias, tiene la posibilidad de realizar conexiones de base de datos, requisitos en FTP (Protocolo de transferencia), se puede programar en java para ciertas acciones específicas. Además posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de requisitos que permite diagnosticar el comportamiento de una aplicación en condiciones de producción, en el momento de la su ejecución simula todas las funcionalidades de un navegador u otro cliente, siendo capaz de manipular resultados en determinado requisito obteniéndolos de una secuencia con un extractor y reutilizarlos para ser empleados en una nueva secuencia. Apache (2017)

Es importante tener en cuenta la siguiente definición de JMeter:

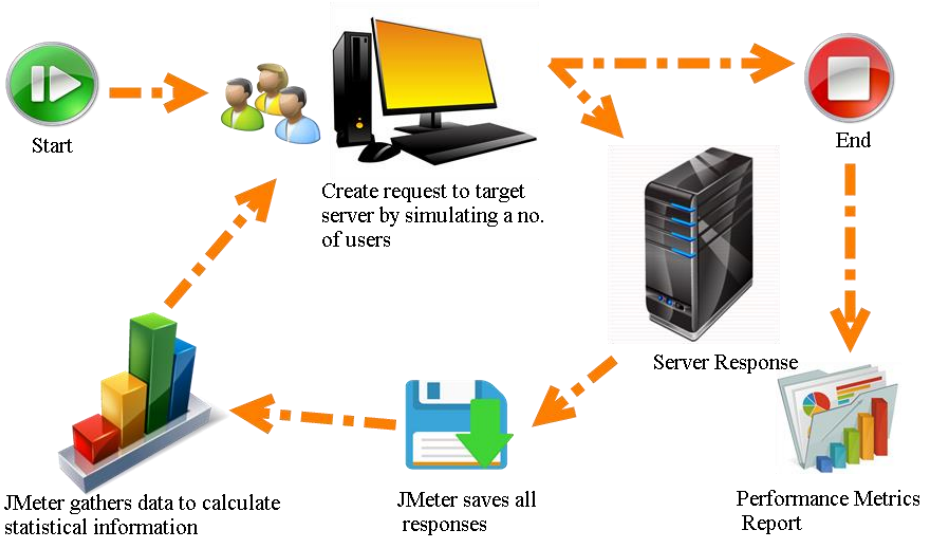
“JMeter no es un navegador, funciona a nivel de protocolo. En lo que respecta a servicios web y servicios remotos, JMeter se parece a un navegador (o más bien, varios navegadores); sin embargo, JMeter no realiza todas las acciones admitidas por los navegadores. En particular, JMeter no ejecuta el JavaScript encontrado en las páginas HTML. Tampoco representa las páginas HTML como

lo hace un navegador (es posible ver la respuesta como HTML, etc., pero los tiempos no se incluyen en ninguna muestra, y solo se muestra una muestra en un hilo a la vez)" Apache (2017)

Apache JMeter funciona de la siguiente manera:

Envía solicitudes a un servidor de destino simulando un grupo de usuarios y luego recopila datos para calcular estadísticas y mostrar métricas de rendimiento a través de varios formatos. Sattar (2016)

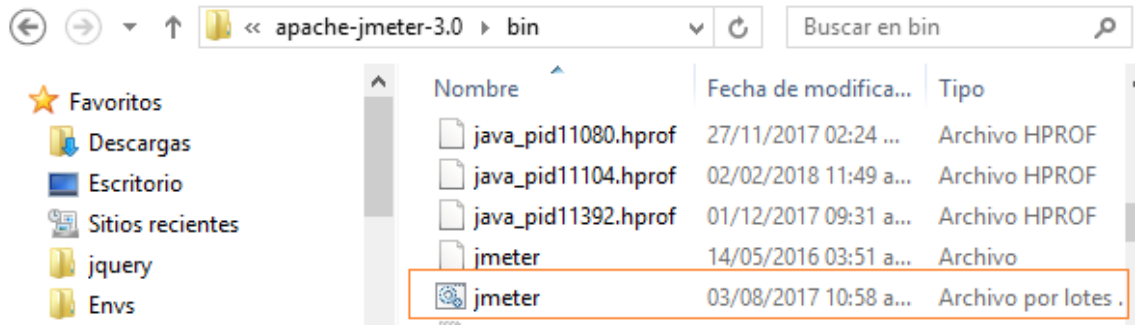
Ilustración 9: Estructura de JMeter



Fuente: JMeter – ToolsQA

Los pasos a realizar un script es necesario descargar una versión estable de la herramienta JMeter, en este caso usaremos la versión 3.0, y la versión de java es la 7, luego debe ser copiado en una carpeta e ingresar a la carpeta bin y clic en el archivo de jmeter.bat(archivo por lotes), como se muestra en la ilustración 10:

Ilustración 10: Activar JMeter por el archivo de lotes



Fuente: Elaboración propia

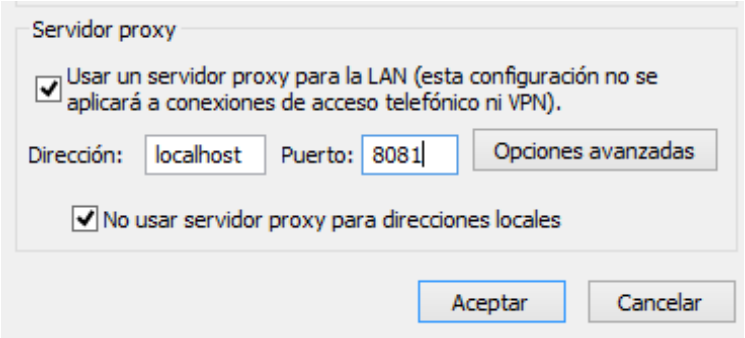
Los siguientes pasos a realizar la grabación de las llamadas al sistema son:

Paso1: Añadir un grupo de hilos, click derecho en plan de pruebas> add> thread user> thread group

Paso 2: Click derecho en banco de prueba> add> Non-Test elements> HTTP(S) Test Script Recorder.

Paso 3: Ir al buscador (explore para Material Flow), cambiar el proxi para iremos a opciones de internet>Conexiones>Configuraciones de red LAN> en el servidor proxi colocaremos local host y el puerto que usaremos para que JMeter haga su trabajo.

Ilustración 11: Configuración de Explore

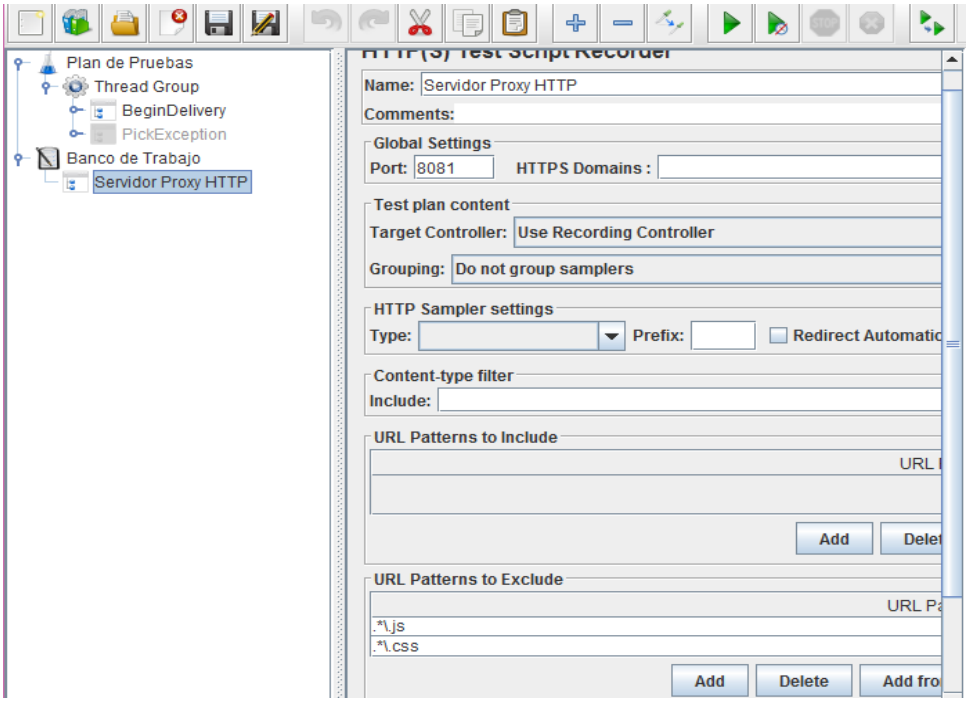


Fuente: Elaboración propia

Paso 4: Ahora colocaremos el mismo puerto en el componente servidor proxi HTTP o HTTP(S) Test Script Recorder, y en URL partterns to exclude

detallamos las extenciones que solamanete podrian provocar ruido en el script y no son relevantes, luego de ello click en el botón start (se mostrara un mensaje colocar aceptar) .

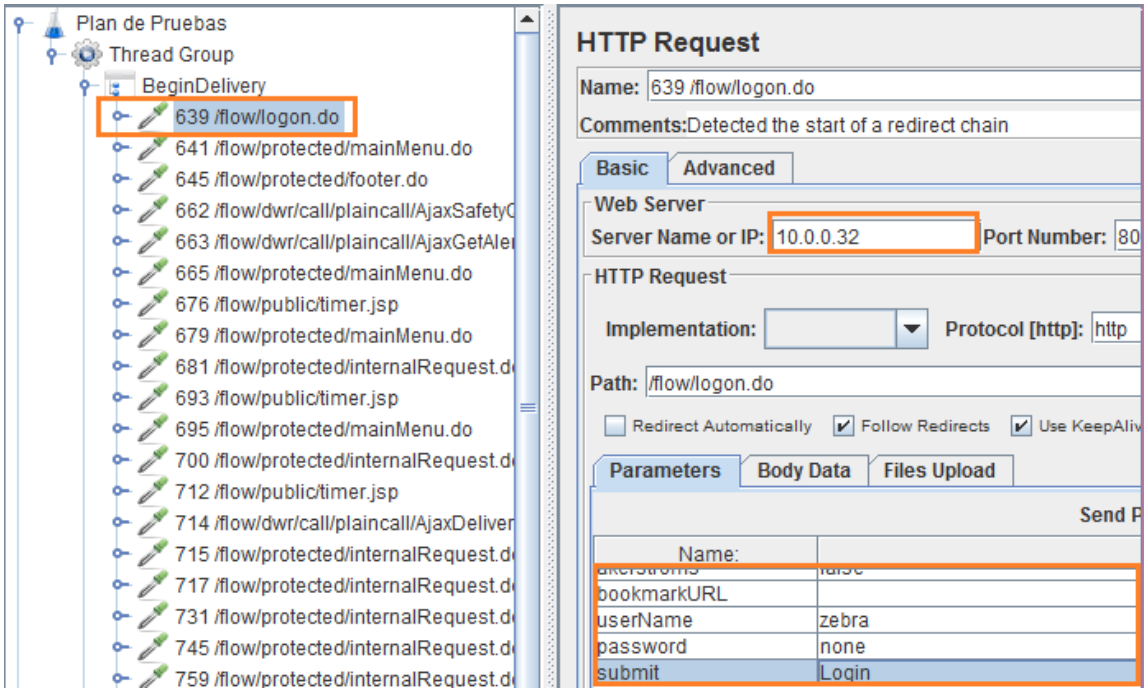
Ilustración 12: Como grabar las llamadas al servidor



Fuente: Elaboración propia

Paso 5: Ahora solo nos queda hacer login o realizar algun caso de prueba y todo ello se grabará y cuando acabe presione el botón stop, como se muestra en la ilustración 13:

Ilustración 13: Llamas al servidor

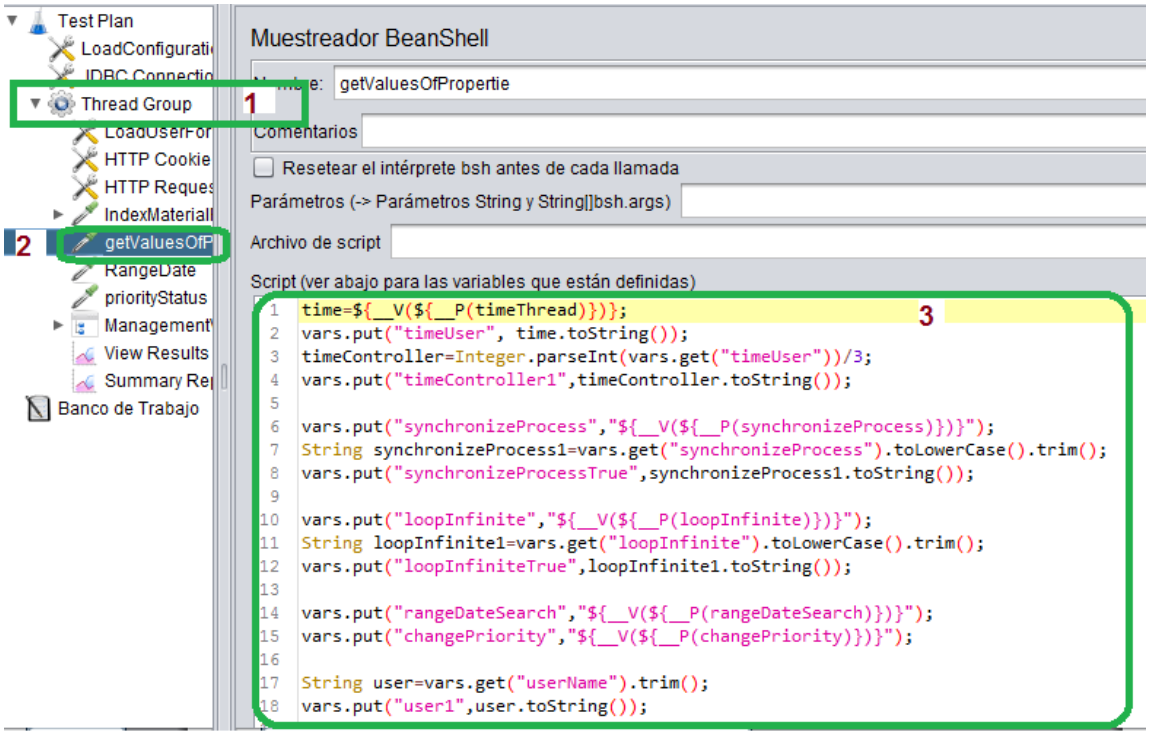


Fuente: Elaboración propia

Paso 6: Después de tener las llamadas al servidor es necesario adaptarlo para que pueda ejecutarse varias pruebas, en la ilustracion 14 se muestra el componente de JMeter que se usa para programar.

- Thread Group: Es el componente donde se manipula la cantidad de usuarios, el tiempo en el que ingresaran y la cantidad de veces que un usuario repetirá la prueba.
- getValuesOfPropertie: es un componente BeanShell en el mismo se puede programar sentencias que el script cumplira cuando se ejecute.
- El codigo mostrado corresponde a la recuperar los valores de las variables en el archivo de propiedades.

Ilustración 14: Componente BeanShell



Fuente: Elaboración propia

2.5.3. Herramienta de ejecución de Test case (Jenkins)

Jenkins será la herramienta con la cual se integrará las pruebas automatizadas con JMeter, la facilidad que nos dará será la programación de tiempos en los que se ejecutaran los scripts.

“Jenkins es un servidor autónomo de código abierto que se puede usar para automatizar todo tipo de tareas relacionadas con la construcción, prueba y entrega o implementación de software.” Jenkins (2017)

“Jenkins es un servidor de integración continua, gratuito, open-source y actualmente uno de los más empleados para esta función y fácil de usar.” Garzas (2014)

Las características que tiene Jenkins son las siguientes Jenkins (2017):

- ✓ **Fácil instalación:** se puede ejecutar con la línea de comando java –jar Jenkins.war.
- ✓ **Configuración sencilla:** puede configurarse directamente desde el modo GUI.
- ✓ **Compatibilidad con cambios de conjuntos:** puede generar una lista de cambios realizados en la compilación de Subversion/CSV. Esto también se hace de una manera bastante eficiente, para reducir la carga en el repositorio.
- ✓ **Enlaces permanentes:** proporciona URL limpias y legibles para la mayoría de sus páginas, como la última compilación, ultima compilación exitosa.
- ✓ **RSS/ E-mail/Integración de IM:** para la supervisión de resultados de compilación por RSS en tiempo real.
- ✓ **Informes de pruebas de JUnit/ TestNG:** con el informe de JUnit se puede tener un registro sobre desde cuando se empezó a romper el software.
- ✓ **Compatibilidad con complementos:** se puede extender por medio de complementos o plugins para el apoyo de los procesos de su equipo.

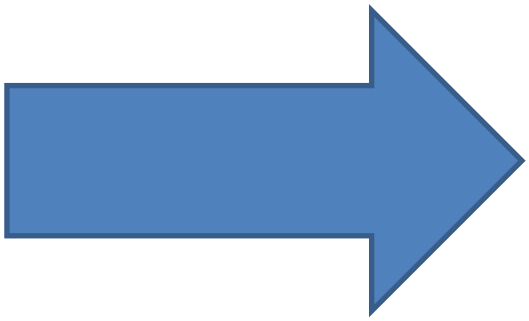
2.5.4. Herramienta de modelado para el sistema UML

UML tiene la capacidad de diagramación, sus esquemas de apoyo de diseño, documentación, construcción e implantación de sistema. Así mismo, su flexibilidad para admitir cambios no previstos durante el diseño o el rediseño. En resumen, la herramienta ideal, es aquella que admite diseño desde inicio a fin, diseño inverso (o rediseño), con esquemas amplios para documentar detalladamente los procesos. Orallo (s.f.)

Los distintos tipos de diagramas son los siguientes Pinelo (2009):

- Diagramas de Casos de Uso para modelar los procesos ’business’.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.

CAPÍTULO III



"Utiliza en la vida los talentos que poseas: el bosque estaría muy silencioso si sólo cantasen los pájaros que cantan mejor."

Henry Van Dike

Capítulo III

En este capítulo se dará a conocer para que sirven las 4 pantallas seleccionadas para la automatización correspondiente, se responderán a tres preguntas importantes, las cuales son: ¿Qué es?, ¿Para qué sirve? y ¿Cómo funciona? De tal manera que la información sea sencilla y fácil de comprender, la fuente de donde se obtuvo la información se mostrará en la parte de anexos.

III. Marco Institucional

Dentro de New Vision Data se encuentra el equipo de Flow donde el proyecto Material Flow se desarrolla, donde se brinda el mantenimiento del mismo, el equipo Flow tiene un líder de proyecto, líder del área de desarrollo y líder del área de control de calidad y es ahí donde el proyecto de automatización ayudara con la ejecución de pruebas.

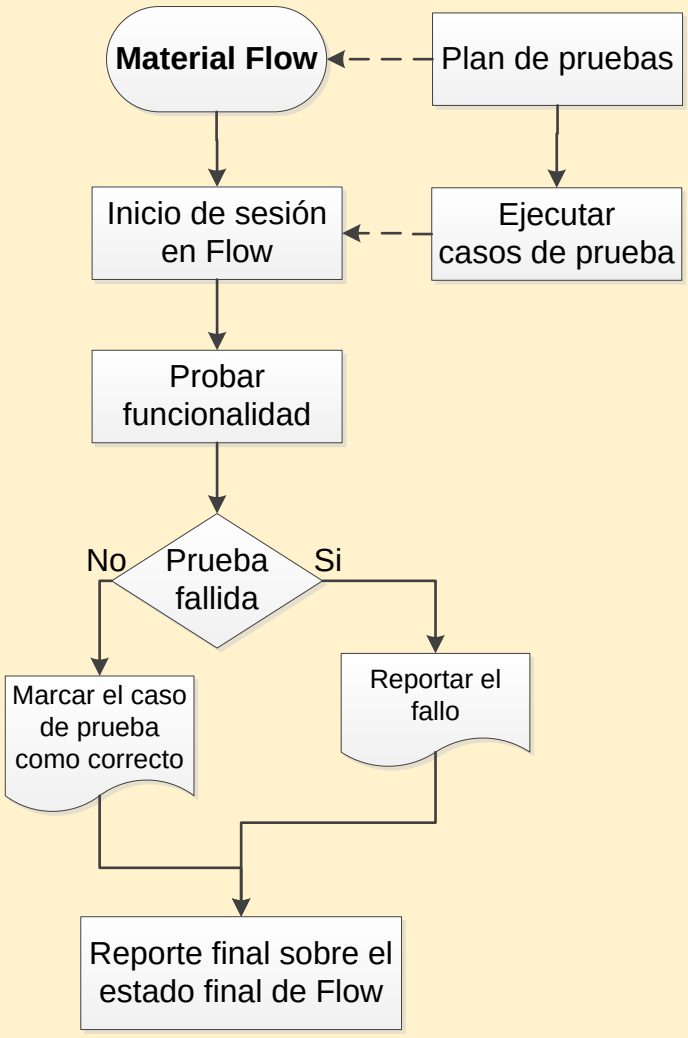
El proceso de control de calidad al sistema Material Flow (todos estos procesos son internos y la información que se manejan no es compartida con personas ajenas al equipo). Flow es un sistema que cuenta varias versiones de forma que se adecue a las necesidades y requerimientos del usuario o los diferentes clientes, es decir el cliente compra las licencias de las diferentes funcionalidades que Flow posee.

Para poder realizar control de calidad a Material Flow es necesario crear un entorno:

Paso 1: Contar con la última versión de Material Flow.
Paso 2: Una base de datos que pertenezcan al cliente (en este caso es el cliente quien proporciona una base de datos cada cierto tiempo para que las pruebas sean lo más cercanos a la realidad)
Paso 3: Preparar el plan de pruebas adecuado a la funcionalidad que se probara.
Paso 4: Conocer los requerimientos que la versión de Flow debe cumplir.
Paso 5: Una vez instalado y adecuado el entorno de testing se debe ejecutar de los casos de prueba.
Paso 6: Como resultado final se adjunta al plan de pruebas los resultados de los casos de prueba, las fallas encontradas, observaciones.

*Observar la tabla 1: proceso de control de calidad de Material Flow

Tabla 1: Proceso de control de calidad en Material Flow

Diagrama de flujo del proceso de control de Calidad	Descripción
	Para el proceso de calidad es necesario tener a Material Flow configurado correctamente con la versión y base de datos. Ingresar al sistema y tener a la mano el plan de pruebas. Una vez en el sistema se inicia con la ejecución de casos de prueba. Si la prueba falla es decir no se obtiene el resultado esperado, se lo reporta y marca con el resultado. Al final se actualiza el plan de pruebas con la información de si el sistema es estable.

Fuente: Elaboración propia

La automatización atacara en la parte de la ejecución de los casos de prueba de tal manera que el encargado de control de calidad administrar y emplear de mejor manera su tiempo, ya que algunos de los casos de prueba son monótonos y se esperar solo un resultado.

3.1. Pantalla de Internal del software Material Flow

En el Pantalla de internal se muestran los pedidos que un usuario haya realizado, sin embargo el usuario que haya iniciado sesión antes de poder visualizar los pedidos en la pantalla de Internal debe asignarse a una ruta en la pantalla de “Selección de ruta”, solo siguiendo este paso el usuario podrá observar todos los pedidos activos que estén en esta ruta sin importar el tiempo que tengan.

La información que se muestra en esta pantalla está relacionada con el pedido, sin embargo antes aclaremos que es un pedido, que está especificado en los siguientes puntos:

- Un pedido es el producto final de la relación de una parte con rutas y locaciones por el cual el pedido debe entregarse.
- La parte es el objeto para el ensamblaje del automóvil (por ejemplo vidrios, llantas…) y es una relación compleja de un nombre único, una descripción, la forma en que se entrega y realiza el pedido es decir la forma si de manera unitaria, en paquetes o por lotes, también se especifica el peso de la parte, cuantos pedidos se entregaran por cada pedido y por último se lo relaciona con ubicaciones lo cual nos sirve para sabes de donde a donde se puede transportar dicha parte (esta configuración se realiza en la pantalla de “Part” o “Parte”).
- Después de tener configurado una parte se debe relacionar la misma con una ruta (a esta configuración se la llama “Part Route”) y según la ruta con se puede realizar los pedidos en la pantalla de manual, es decir se pueden hacer n pedidos en un determinado tiempo.
- La mayoría de esta información se muestra en la pantalla de Internal como se puede observar en la ilustración 15.

- Todos los pedidos pueden ser completados por usuario, forma unitaria, de muchos a muchos, estos procesos se realizan con los botones que se observan en la ilustración 15, que tienen por nombre:
  Botón 1: Assigned sirve para que un usuario se asigne a un pedido.
  Botón 2: Begin delivery sirve para el proceso de entrega de los pedidos, el usuario puede entregarlos desde 2 o más pedidos.
  Botón 3: Next Request de igual forma es para el proceso de entrega de pedidos, pero forma unitaria.
  Botón 4: Refresh sirve para refrescar la pantalla.

Para el proceso de entrega el usuario se asigna al pedido en el momento en que se lo recoge la parte de la primera ubicación, luego se conoce la ubicación de destino donde el usuario entrega el pedido.

En la lista de pedidos se muestra la siguiente información (ilustración 15):

- **Campo Priority (prioridad)** estas se las idéntifica por medio de los colores (plomo si el pedidos es reciente, verde, amarillo, naranja y por ultimo rojo) los colores cambian en función del tiempo que transcurre.
- **Campo Part (parte)** es el nombre de la parte del pedido que debe ser entregado.
- **Campo Quantity (cantidad)** es la cantidad de partes que debe entregarse en el pedido.
- **Campo From location (locación de origen)** es la ubicación inicial de donde se debe de recoger la cantidad determina de partes para que el pedido se entregue de forma correcta.
- **Campo To location (locación de destino)** es la ubicación de destino del pedido a ser entregado.

Ilustración 15: Pantalla de internal - primera parte



Fuente: Elaboración propia

Tabla 2: Casos de prueba para la Pantalla Internal

Casos de prueba para la pantalla Internal
Completar entrega de uno a uno
Completar entrega de manera multiple
Asignarse a multiples pedidos por el botón begin delivery
Asignarse a pedidos uno a uno por el botón next request
Asignarse a un pedido con mensaje de excepción
Completar el pedido con mensaje de excepción
Asignarse a un pedido con excepción de cantidad en la pantalla de Delivered Quantity
Completar el pedido con excepción de cantidad en la pantalla de Delivered Quantity
Exportar la lista CSV de internal
Cancelar un pedido en la lista de Delivery
Cancelar un pedido en la pantalla de Delivery Request Part
Pedido con excepción de cantidad en el proceso de delivery complete
Sortear la lista de pedidos con los filtros
Refrescar pantalla

Fuente: Elaboración propia

3.2. Pantalla de Excepciones

El Pantalla de Excepciones funciona de manera similar en cuanto a los procesos del Pantalla Internal, es decir así como se muestra la información correspondiente de un pedido, en esta pantalla se podrá observa cual fue la razón para que el pedido tenga una excepción y no haya sido completado.

Existen diferentes maneras para realizar un pedido con excepción, las cuales son a la hora de realizar el proceso de pick, en la lista de delivery o en la pantalla flotante de delivery. Sin embargo la razón principal para que un request tenga una excepción es que no pueda ser completa es decir que suceda algo cuando se esté transportando el pedido de la ubicación A a la ubicación B, estas razones podrían ser que no es la cantidad pedida, las partes del pedido se hayan caído o perdido. De tal manera que todos estos pedidos con excepciones se mostraran en la pantalla de excepción y la información que se muestra es quien realizo la excepción a ese pedido, cual es la razón, cual es la cantidad que no encontró. En la pantalla de excepciones se podrá realizar los procesos son:

Botón 1: Begin Delivery (Inicio de entrega) con el cual se podrá completar el pedido si aún es posible, de forma masiva.
Botón 2: Next Request (Siguiente pedido) completar el pedido de uno a uno.
Botón 3: Void (vaciar) con este botón se vacía el pedido es decir se anulara el pedido por que no se tiene la cantidad necesaria o se perdió por completo.
Botón 4: Cancel (cancelar) el proceso de cancelación en caso de que ya tenga la cantidad necesaria o el pedido se haya encontrado, una vez que se cancela la excepción, el pedido se muestra en la lista de delivery.
Botón 5: Refresh (refrescar) botón para actualizar la pantalla.

Ilustración 16: Pantalla de Excepción



Fuente: Elaboración propia

Tabla 3: Casos de prueba para la pantalla Exception

Casos de prueba para la panatalla Exception
Completar el pedido con excepción de uno a uno
Completar el pedido con excepción de manera múltiple
Cancelar una excepción
Completar la excepción
Exportar la lisa CSV de la pantalla de excepción

Fuente: Elaboración propia

3.3. Pantalla de Management View

En esta siguiente pantalla se puede observar alrededor de 80 campos de información sobre todos los request que aún no fueron completados o los que se encuentran en la lista de excepciones, desde esta pantalla el usuario puede observar todos estos request sin importar este o no asignado a una ruta específica, de igual manera podrá completarlos o realizar algún proceso, los cuales son:

Botón 1: Excepción este proceso de igual forma se aplican las reglas para que un pedido tenga o se le asigne una razón de excepción, la cual se puede realizar cambiando su estado en la ventana flotante o de eligiendo una o más pedidos y presionando el botón amarillo con el signo de interrogación, después de realizar esta acción en la columna de estados se mostrar el mismo icono que del botón.

Botón 2: Void (vaciar) este proceso se puede cumplir de 2 diferentes maneras, una de forma más unitaria dentro de una ventana flotante y la otra eligiendo una o más pedidos y presionando en el botón azul, después de realizar este proceso la cantidad de request mostrados disminuirá.

Botón 3: Cancel (cancelar) este botón sirve para que el usuario cancelar la entrega de un pedido.

- El proceso de convertir en hold un pedido el cual se realizar cambiando su estado en la ventana flotante, después de esta acción en la columna de estado se mostrara un icono de pause de color rojo.
- En la lista de pedidos se muestra toda su información, es decir qué tipo de pedio es, su ruta, tipo de ruta, que usuario realizo el pedido, la fecha y la hora en la que se pidió, a que usuario esta asignado un determinado request.

Ilustración 17: Pantalla de Management View



Fuente: Elaboración propia

Tabla 4: Casos de prueba para Management View

Casos de prueba para la pantalla Management View
Exportar la lista CSV de management view
Vaciar la lista de pedidos
Vaciar pedidos con excepción
Cancelar un pedido
Cancelar un pedido con excepción
Cambiar las prioridades de un request (overdue, requested, need, need son, critical)
Cambiar los estados de un request (Exception, Hold, Void)
Buscar un pedido por cada campo
Verificar información para los campos de scanner LPN
Verificar información para los campos de asignación LPN
Verificar información para el campo de asignación a usuario
Seleccionar el checbox para mostrar datos archivados

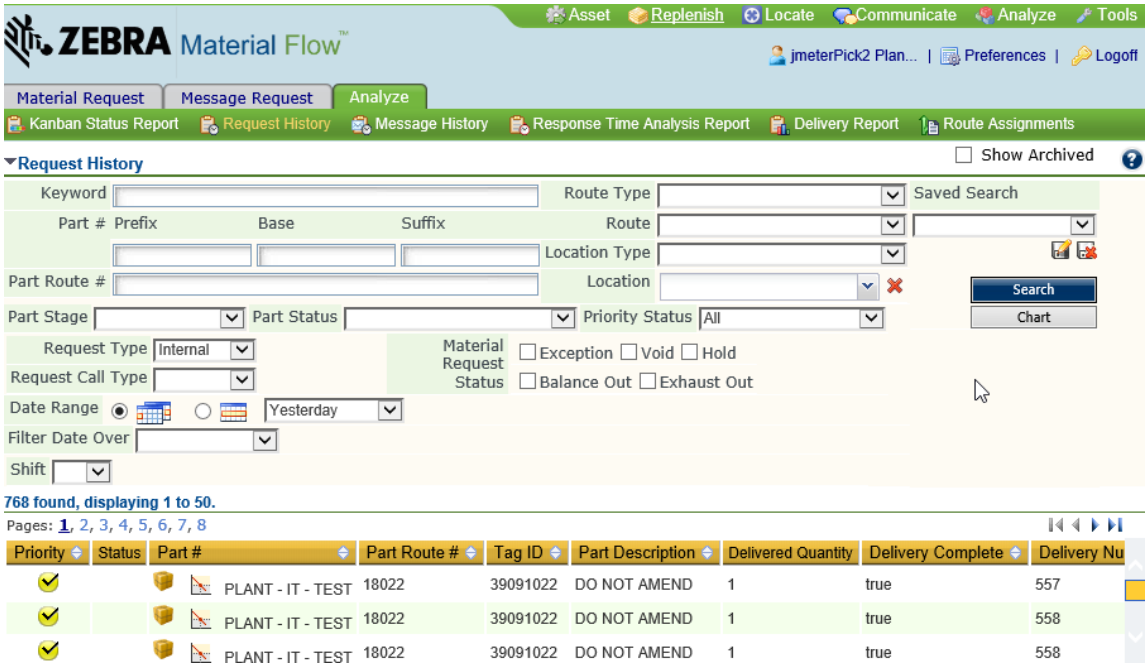
Fuente: Elaboración propia

3.4. Pantalla de Request History

En la pantalla de Request History la información que se muestra es similar a la de Management View sin embargo es posible ver la información completa de un pedido de forma mucho más clara, extensa y completa sobre todo los procesos en que se realizó con dicho request, pero el usuario no podrá realizar más que el proceso de búsqueda. Toda la información que se muestra es la siguiente (ver la ilustración 18):

- El estado de los request, su prioridad si cuales fueron o no completados según sea el caso.
- A que usuario fue asignado, la fecha y hora en la que sucedió.
- Quien realizo el pedido y a la hora respectiva.
- A que usuario fue asignado, por medio de qué proceso se completó, ya sea pick/delivery, excepción, quien lo convirtió en excepción, hold o void.
- Cuál es su tipo de ruta y su ruta, cuales es su destino de entrega y desde donde se deben recoger.
- Cuál es el proceso por el cual se creó el pedido.

Ilustración 18: Pantalla de Request History



Fuente: Elaboración propia

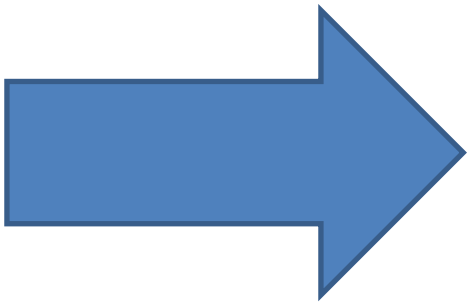
Tabla 5: Casos de prueba para Request History

Casos de prueba para Request History
Buscar un pedido por cada campo
Buscar pedios por rango de fechas
Exportar archivo csv
Verificar información para los campos de scanner LPN
Verificar información para los campos de asignación LPN
Verificar información para el campo de asignación a usuario
Seleccionar el checbox para mostrar datos archivados

Fuente: Elaboración propia

CAPíTULO

IV



"Lo que cuenta no es la cantidad de horas que dedicamos al trabajo, sino la calidad del trabajo en esas horas"

Sam Ewing

Capítulo IV

En este capítulo se muestra a detalle todo el proceso de desarrollo de la automatización de los casos de prueba para el sisema Material Flow en las pantallas respectivas, teniendo en cuenta la metodología de desarrollo para la identificación de tareas, el diseño y la implementación.

IV. Marco Práctico

4.1. Introducción

Cada punto que se tocara en este capítulo es de importancia ayuda al análisis, entendimiento y alcance de la solución del problema, en el modelado de negocio con los casos de prueba, se muestra diagramas de la ejecución actual de los casos de prueba, la evaluación y la manera en la que se obtienen los resultados.

En la parte de análisis de requerimientos se detallará todos los componentes que son necesarios para la elaboración de los reportes, parámetros necesarios para su funcionalidad.

4.2. Análisis de requerimientos

Con los siguientes requerimientos se describirá las características sobre cómo se espera que sea el comportamiento de sistema de automatización del test plan para los casos de uso en las pantallas especificadas.

4.2.1. Requerimientos Funcionales

Todos los requerimientos funcionales se describen en la tabla 6:

Tabla 6: Requerimientos Funcionales

Nro.	Requerimiento	Descripción	Prioridad
RF1	Cada script debe ser la ejecución de un caso de de prueba de la lista en el plan de pruebas.	Cada script automatizado debe cumplir adecuadamente con la ejecución de los casos de prueba.	Alta
RF2	Reporte de la ejecución de cada script.	Al finalizar la ejecución de cada script debe exportar resultado sobre si el caso de prueba se ejecuto correctamente.	Alta
RF3	Ejecución de los scripts periódicamente.	Al finalizar con la creación de los scripts los mismos deben ejecutarse forma periódica.	Alta
RF4	Notificación sobre la ejecución de los scripts.	Existe posibilidades de que los script fallen en su ejecución por lo tanto deben notificarse los encargados de control de calidad.	Alta
RF5	Plan de prueba automatizado.	Con los reportes que se tendrán después de cada ejecución de los scripts deben servir para el test plan general	Alta

Fuente: Elaboración propia

4.2.2. Requerimientos No Funcionales

Tabla 7: Requerimientos no funcionales

Nro.	No requerimiento	Descripción
NFR1	Estandarización de los reportes	No se tiene actualmente una forma de presentación de los reportes, de manera que debe proponer la más adecuada.
NFR2	Almacenar los planes de prueba automatizados	Cada uno de los test plan son el resultado de la ejecución de los script tengan o no test que no pasaron.
NFR3	Pruebas de strees para Material Flow	Los scripts deben ayudar a probar el performance del sistema Flow, en cada nuevo build.
NFR4	Integración continua de la automatización	Todos los script debe ejecutarse continuamente de tal manera que se tenga una revisión continua de las versiones de Material Flow

Fuente: Elaboración propia

4.3. Modelado de Negocio

4.3.1. Diagrama de casos de uso de las pantallas de automatización del Test Plan

En los siguientes diagramas se muestra la forma en la que se trabaja con el sistema Material Flow el momento de realizar el testeo de una nueva versión en las pantallas de Internal, Management View, Exception y Request History. Los diagramas se mostraran por cada pantalla y se tendrá una tabla en donde se explicara de forma más detallada.

4.3.1.1. Diagrama casos de uso: Entrega de pedidos (Pantalla de Internal-Matrial Flow)

Diagrama 1: Caso de uso de la Pantalla de Internal



Fuente: Elaboración propia

Tabla 8: Caso de uso del Pantalla Internal

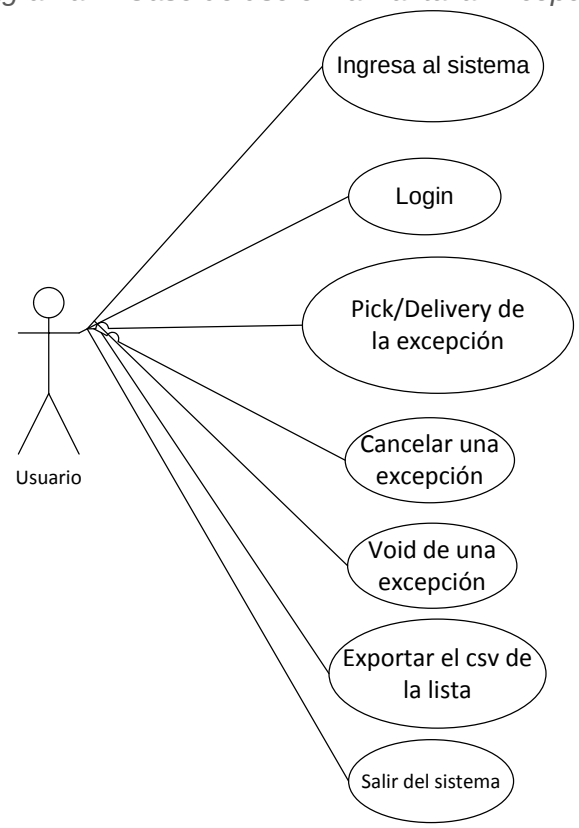
CU-1	**Proceso ejecución de casos de prueba en la pantalla Internal**
Descripción	En la pantalla de Internal se realiza el proceso de pick y delivery, para relizar la entrega de las partes en las distintas locaciones.
Secuencia normal	1. Para ingresar a la aplicación se debe pasar por el login. 2. Ir a la Pantalla de Replenish/Material Request. 3. Ir a la Pantalla Internal con la lista de request que tenemos podemos realizar varios procesos de manera paralela o secuencial. 4. Pick por medio del next requet, el usuario se assiganara a una sola parte, para luego ser entregado. 5. Pick por medio del botón beguin delivery, click en 2 o mas check box de los pedidos, click en el botón beguin delivery el usuario se assignara a todas las partes y podra ser entregado. 6. Asignarse a un pedido, el usuario que se asigna a un pedio, el pedido solo podra ser entregado por ese mismo usuario. 6.1 En el proceso de pick el usuario puede realizar una excepción en cuanto a cantidad del pedido. 6.2 El usuario se asigna al pedido por medio del botón next request. 6.3 El usuario se asigna a muchos pedidos con el botón beguin delivery. 7. En la lista de deliverys se muestran todos los pedidos que el usuario debe entregar. 7.1 El usuario se asigna al pedido con el botón delivery complete 7.2 Para desasignarse del pedido, se debe selecionar el peido y presionar el botón cancel 7.3 Antes de completar el pedido es posible realizar una excepción. 8. Exportar csv en la parte inferior izquierda.

	9. El usuario debe verificar que la lista pueda ser ordenada correctament por el número de la locación. 10. La lista debe ser ordenada por la prioridad del pedido 11. En la parte inferior de la pantalla internal se encuentra el botón refresh al darle un click la lista debe actuaizarse
Excepciones	
Prioridad	Alta
Roles	QA.
Datos	Request generados de manera manul o automatica.

Fuente: Elaboración propia

4.3.1.2. Diagrama caso de uso: Revisión de pedido (Pantalla Excepción – Material Flow)

Diagrama 2: Caso de uso en la Pantalla Excepción



Fuente: Elaboración propia

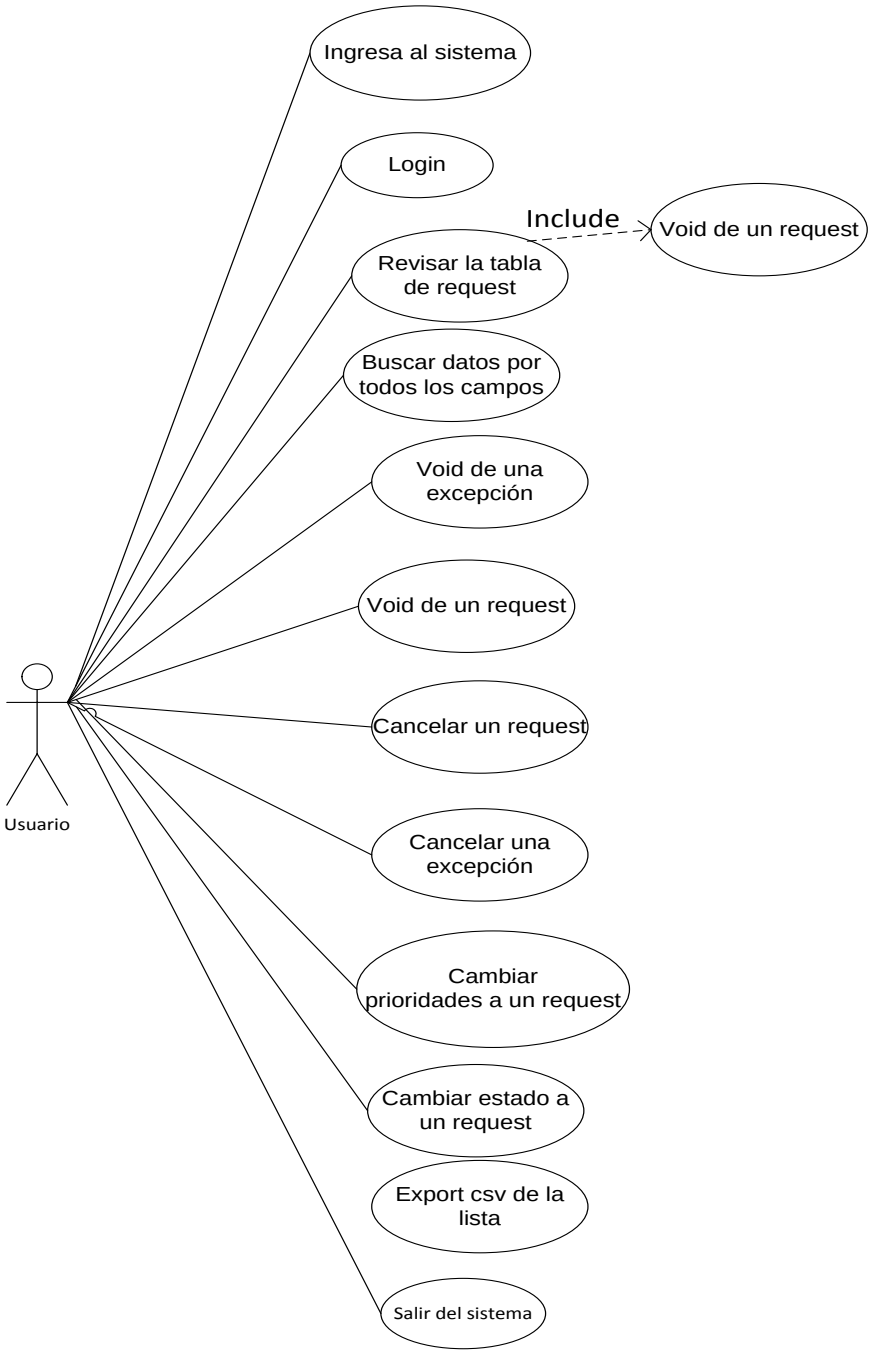
Tabla 9: Tabla del proceso de ejecución de Management View

CU-2	**Proceso ejecución de casos de prueba en la pantalla Exceptions**
Descripción	Este proceso de excepciones de igual manera es realizado manualmente, en esta Pantalla tenemos la posibilidad de terminar el proceso de Pick y Delivery de manera forzada, de un request que no se pudo haber completado o no se puede completar.
Secuencia normal	1. Para ingresar a la aplicación se debe pasar por el login. 2. Despues del login exitoso, se debe a la area de Replenish, Internal. 3. En la lista de excepciones se puede realizar los siguientes procesos. - El pedido con excepción es posible cancelarla. - Uno o mas pedidos pueden ser eliminados con el botón void. - En la lista los pedidos se pueden completar por pick/delivery por medio del botón next request. - En la lista los pedidos se pueden completar por pick/delivery por medio del botón begin delivery. - La lista de exception puede ser exportada.
Excepciones	
Prioridad	Alta
Roles	QA.
Datos	Request enviados desde pick o delivery.

Fuente: Elaboración propia

4.3.1.3. Diagrama caso de uso: Revisión General de todos los pedidos (Pantalla de Management View-Material Flow)

Diagrama 3: Caso de uso de la Pantalla Management View



Fuente: Elaboración propia

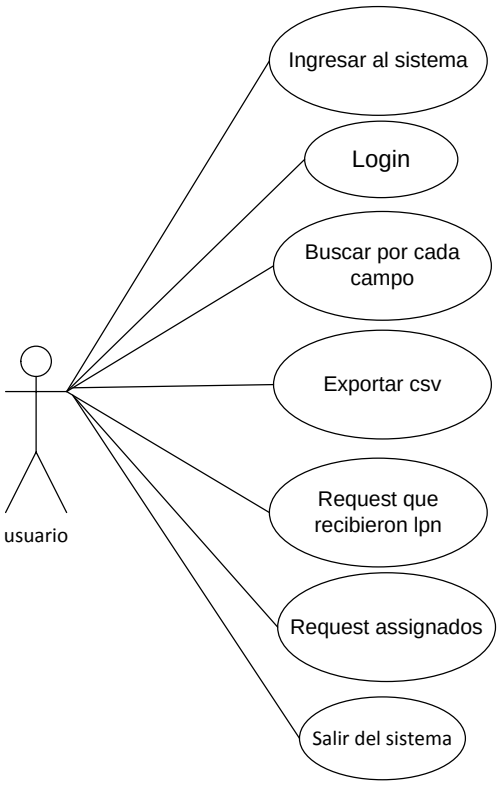
Tabla 10: Tabla del proceso de Management View

CU-3	**Proceso ejecución de casos de pruebas en la pantalla Management View**
Descripción	El Pantalla de Management View, es donde nos muestra la lista de todos los request que estan por completarse, en esta Pantalla tenemos el control total de los datos. Los procesos no deben realizar de manera secuencial no tienen un orden especifico.
Secuencia normal	1. Para ingresar a la aplicación se debe pasar por el login. 2. Despues del login exitoso, se debe a la area de replenishment, Management View, y buscar los request. 3. Al seleccionar un request podemos realizar void del mismo. 4. De la lista que nos muesta, elegimos un request con excepción y tambien podemos realizar void. 5. De aquellos request que esten en la lista de delivery, podemos cancelarlo ya sea seleccionando uno o muchos. 6. De la lista manera de la lista de excepciones podemos cancelarlos de uno o mas. 7. Al seleccionar un request podemos cambiar su prioridad segun sea necesario. 8. Al seleccionar un request podemos cambiar su estado igual sea necesario (void, hold, excepción). 9. La lista de csv puede ser exportada presionando el icono de una hoja en la parte inferior izquierda.
Excepciones	
Prioridad	Alta
Roles	QA.
Datos	Request generados de manera manual o automatica.

Fuente: Elaboración propia

4.3.1.4. Diagrama de caso de uso: Reporte de todos los pedidos (Request History-Material Flow)

Diagrama 4: Caso de uso de Request History



Fuente: Elaboración propia

Tabla 11: Caso de uso de Request History

CU-3	**Proceso ejecución de casos de prueba en la pantalla Request History**
Descripción	El Pantalla de Request History, es donde nos muestra la lista de todos los request que estan por completarse o que fueron completados, se muestra el seguimiento de todos los request es decir cuando fue realizado el pedido, que usuario lo completo, por medio de que proceso.
Secuencia normal	1. Para ingresar a la aplicación se debe pasar por el login. 2. Despues del login exitoso, se debe a la Pantalla de replenishment, Request History.

	3. Buscar los request por todos los campos. 4. Exportar la lista en archivo csv. 5. Request que se completaron con lpn. 6. Asignacion de usuarios a un pedido. 7. Buscar request con datos archived.
Excepciones	
Prioridad	Alta
Roles	QA.
Datos	Request generados de manera manual o automatica.

Fuente: Elaboración propia

4.4. Planificación del sistema

En el equipo donde se desarrolla el sistema “Material Flow” se identificaron los siguientes usuarios, como se puede observar en la tabla12.

Tabla 12: Usuarios para el sistema de automatización

Usuario	Función
Proyect Manager	El líder del proyecto.
Control de calidad	Es el encargado de verificar todos los requerimientos se cumplan, que el sistema funcione correctamente con los cambios que se realicen.
Líder de equipo	Es el encargado de organizar todas las actividades del equipo con respecto al desarrollo de “Material Flow”.
Desarrolladores	Encargados de realizar diferentes tareas que el líder de equipo designe.

Fuente: Elaboración propia

Ahora después de exponer los usuarios para el uso de la automatización, en la tabla 13 se mostrará la lista de los sprints.

Tabla 13: Sprint Product Backlog

ID	Historia	Prioridad	Tiempo estimado (días)	Fecha de inicio	Dependencia
1	**Requerimientos para los scripts**	Alta	3	23/04/2018	Ningúna
2	**Script para la pantalla de Internal en los casos de prueba**	Alta	1	25/04/2018	1
3	**Script para la pantalla de Excepciones en los casos de prueba**	Alta	1	07/05/2018	1
4	**Script para la pantalla de Management View en los casos de prueba**	Alta	1	14/05/2018	1
5	**Script para el Pantalla de Request History en los casos de prueba**	Alta	1	23/05/2018	1

Fuente: Elaboración propia

4.5. Sprint Product Backlog para el desarrollo de la automatización

En la tabla 14 se observa las tareas que se llevaran a cabo, cabe mencionar que es una lista general y se podrá ampliar en función se vaya acabando o trabajando en los Sprint, cada una de las tareas tiene su respectivo nombre, la prioridad que se merece, el estado en el que se encuentra actualmente, la persona encargada de realizar las tareas y una fecha de entrega e inicio.

Tabla 14: Tabla de Sprint Backlog

Sprint Backlog para automatización de Plan de pruebas						
Nombre de tareas		**Prioridad**	**Owen**	**Fecha inicio**	**Fecha final**	**Estado**
Sprint #1 Requerimientos para los scripts						
Rq-1	Identificar pantallas para la automatización	**Alta**	M. Alavi	23/04/2018	24/04/2018	Acabado
Rq-2	Identificar los casos de prueba en base a las pantallas del test plan aptos para ser automatizados	**Alta**	M. Alavi	24/04/2018	25/04/2018	Acabado
Rq-3	Requerimientos para la ejecución y funcionalidad de los scripts	**Alta**	M. Alavi	25/04/2018	26/04/2018	Acabado
Sprint #2 Script para el área de Internal en los casos de prueba						
In-1	Completar entrega de uno a uno	**Alta**	M. Alavi	25/04/2018	27/04/2018	Acabado
In-2	Completar entrega de manera multiple	**Alta**	M. Alavi	26/04/2018	28/04/2018	Acabado
In-3	Asignarse a multiples pedidos por el botón begin delivery	**Medio**	M. Alavi	27/04/2018	29/04/2018	Acabado
In-4	Asignarse a pedidos uno a uno por el botón next request	**Medio**	M. Alavi	28/04/2018	30/04/2018	Acabado
In-5	Asignarse a un pedido con mensaje de excepción	**Alta**	M. Alavi	29/04/2018	01/05/2018	Acabado
In-6	Completar el pedido con mensaje de excepción	**Alta**	M. Alavi	30/04/2018	02/05/2018	Acabado
In-7	Asignarse a un pedido con excepción de cantidad en la pantalla de Delivered Quantity	**Alta**	M. Alavi	01/05/2018	03/05/2018	Acabado
In-8	Completar el pedido con excepción de cantidad en la pantalla de Delivered Quantity	**Alta**	M. Alavi	02/05/2018	04/05/2018	Acabado
In-9	Exportar la lista CSV de internal	**Medio**	M. Alavi	03/05/2018	05/05/2018	Acabado

Nombre de tareas		Prioridad	Owen	Fecha inicio	Fecha final	Estado
In-10	Cancelar un pedido en la lista de Delivery	**Bajo**	M. Alavi	04/05/2018	06/05/2018	Acabado
In-11	Cancelar un pedido en la pantalla de Delivery Request Part	**Bajo**	M. Alavi	05/05/2018	07/05/2018	Acabado
In-12	Pedido con excepción de cantidad en el proceso de delivery complete	**Alta**	M. Alavi	05/05/2018	07/05/2018	Acabado
In-13	Pedido con excepción de notificación en el proceso de delivery complete	**Alta**	M. Alavi	05/05/2018	07/05/2018	Acabado
In-14	Diseño del script para los casos de prueba.	**Alta**	M. Alavi	10/05/2018	12/05/2018	Acabado
In-15	Detallar los componentes de JMeter que se usaran.	**Alta**	M. Alavi	10/05/2018	12/05/2018	Acabado
In-16	Implementación de los casos de prueba automatizados con Jenkins	**Alta**	M. Alavi	10/05/2018	12/05/2018	Acabado
Sprint #3 Script para el Pantalla de Exceptions en los casos de prueba						
Exp-1	Completar el pedido con excepción de uno a uno	**Medio**	M. Alavi	13/05/2018	15/05/2018	Acabado
Exp-2	Completar el pedido con excepción de manera múltiple	**Medio**	M. Alavi	14/05/2018	16/05/2018	Acabado
Exp-3	Cancelar una excepción	**Alta**	M. Alavi	15/05/2018	17/05/2018	Acabado
Exp-4	Completar la excepción	**Alta**	M. Alavi	16/05/2018	18/05/2018	Acabado
Exp-5	Exportar la lisa CSV de la pantalla de excepción	**Medio**	M. Alavi	18/05/2018	20/05/2018	Acabado
Exp-6	Diseño del script para los casos de prueba.	**Alta**	M. Alavi	21/05/2018	23/05/2018	Acabado
Exp-7	Detallar los componentes de JMeter que se usaran.	**Alta**	M. Alavi	21/05/2018	23/05/2018	Acabado
Exp-8	Implementación de los casos de prueba automatizados con Jenkins	**Alta**	M. Alavi	21/05/2018	23/05/2018	Acabado
Sprint #4 Script para el Pantalla de Management View en los casos de prueba						

Nombre de tareas		Prioridad	Owen	Fecha inicio	Fecha final	Estado
Mv-1	Exportar la lista CSV de management view	**Alta**	M. Alavi	24/05/2018	26/05/2018	Acabado
Mv-2	Vaciar la lista de pedidos	**Alta**	M. Alavi	25/05/2018	27/05/2018	Acabado
Mv-3	Vaciar pedidos con excepción	**Alta**	M. Alavi	26/05/2018	28/05/2018	Acabado
Mv-4	Cancelar un pedido	**Medio**	M. Alavi	27/05/2018	29/05/2018	Acabado
Mv-5	Cancelar un pedido con excepción	**Medio**	M. Alavi	28/05/2018	30/05/2018	Acabado
Mv-6	Cambiar las prioridades de un request (overdue, requested, need, need son, critical)	**Alta**	M. Alavi	01/06/2018	03/06/2018	Acabado
Mv-7	Cambiar los estados de un request (Exception, Hold, Void)	**Alta**	M. Alavi	02/06/2018	04/06/2018	Acabado
Mv-8	Buscar un pedido por cada campo	**Alta**	M. Alavi	03/06/2018	05/06/2018	Acabado
Mv-9	Diseño del script para los casos de prueba.	**Alta**	M. Alavi	04/06/2018	07/06/2018	Acabado
Mv-10	Detallar los componentes de JMeter que se usaran.	**Alta**	M. Alavi	08/06/2018	10/06/2018	Acabado
Mv-11	Implementación de los casos de prueba automatizados con Jenkins	**Alta**	M. Alavi	08/06/2018	10/06/2018	Acabado
Sprint #5 Script para el Pantalla de Request History en los casos de prueba						
Rh-1	Exportar la lista CSV de Request history	**Alta**	M. Alavi	11/06/2018	13/06/2018	Acabado
Rh-2	Diseño del script para los casos de prueba.	**Alta**	M. Alavi	14/06/2018	16/06/2018	Acabado
Rh-3	Detallar los componentes de JMeter que se usaran.	**Alta**	M. Alavi	14/06/2018	16/06/2018	Acabado
Rh-4	Implementación de los casos de prueba automatizados con Jenkins	**Alta**	M. Alavi	14/06/2018	16/06/2018	Acabado

Fuente: Elaboración Propia

4.5.1. Primer Sprint – Análisis de los casos de uso para la automatización

En el primer sprint se realizará el análisis de los casos de prueba del test plan, aquellos que sean los adecuados para la automatización en relación de las pantallas mencionadas anteriormente.

Tabla 15: Primer sprint

Sprint	Historia	Fecha de inicio	Duración (días)
1	Requerimientos para los scripts	23/04/2018	3

Fuente: Elaboración propia

En la tabla 16 se especificará la lista de las tareas que son necesarias para concluir el primer sprint, las tareas de la primera iteración constara en el análisis de casos de prueba para luego poder automatizarlos.

Tabla 16: Primer sprint - Requerimientos para los scripts

ID	Nombre	Tipo	Prioridad	Owen	Duración (días)	Estado
Rq-1	Identificar pantallas para la automatización	Análisis	**Alta**	M. Alavi	1	Terminado
Rq-2	Identificar los test case en base a las pantallas del test plan aptos para ser automatizados	Análisis	**Alta**	M. Alavi	1	Terminado
Rq-3	Requerimientos para la ejecución y funcionalidad de los scripts	Análisis	**Alta**	M. Alavi	1	Terminado

Fuente: Elaboración propia

4.5.1.1. Diseño-Identificación de requerimientos para la construcción de scripts

Tabla 17: Análisis de requerimientos para el primer sprint

Nro.	Requerimiento	Descripción	Prioridad
R1	Archivo de configuración global.	El usuario antes de ejecutar debe de configurar los parámetros necesarios para llevar acabo el test.	Alta
R2	Archivo de configuración por cada script.	Cada script deberá tener un archivo según sea necesario, con su propia configuración.	Alta
R3	Sincronización de los hilos que simularan a los usuarios.	Al ser script de performance es necesario que los hilos estén sincronizados para evitar bloqueos de la base de datos.	Alta
R4	Archivo de usuarios.	Cada script deberá tener su propio archivo de usuarios con los que ejecutaran el test.	Alta

Fuente: Elaboración propia

Lista de casos de pruebas para ser construidos

Tabla 18: Lista de los casos de prueba

Nro.	Nombre de caso de uso
1	Completar entrega de uno a uno
2	Completar entrega de manera multiple
3	Asignarse a multiples pedidos por el botón begin delivery
4	Asignarse a pedidos uno a uno por el botón next request
5	Asignarse a un pedido con mensaje de excepción

Nro.	Nombre de caso de uso
6	Completar el pedido con mensaje de excepción
7	Asignarse a un pedido con excepción de cantidad en la pantalla de Delivere Quantity
8	Completar el pedido con excepción de cantidad en la pantalla de Delivere Quantity
9	Exportar la lista CSV de internal
10	Cancelar un pedido en la lista de Delivery
11	Cancelar un pedido en la pantalla de Delivery Request Part
12	Pedido con excepción de cantidad en el proceso de delivery complete
13	Pedido con excepción de notificación en el proceso de delivery complete
14	Completar el pedido con excepción de uno a uno
15	Completar el pedido con excepción de manera múltiple
16	Cancelar una excepción
17	Completar la excepción
18	Exportar la lista CSV de management view
19	Vaciar la lista de pedidos
20	Vaciar pedidos con excepción
21	Cancelar un pedido
22	Cancelar un pedido con excepción
23	Cambiar las prioridades de un request (overdue, requested, need, need son, critical)
24	Cambiar los estados de un request (Exception, Hold, Void)
25	Buscar un pedido por cada campo
26	Exportar la lista CSV de management view
27	Exportar la lista en un archivo csv

Fuente: Elaboración propia

4.5.2. Segundo Sprint - Script para la Pantalla de Internal

En el segundo sprint se dará a conocer la cantidad de script para la pantalla de internal, después de realizar el análisis respectivo, en la tabla 19 se muestra una información general del segundo sprint.

Tabla 19: Segundo sprint

Sprint	Historia	Fecha de inicio	Duración (días)
2	Script para el Pantalla de Internal en los casos de prueba	25/04/2018	17

Fuente: Elaboración propia

En la tabla 20 se especificara la lista de las tareas que son necesarias para concluir el segundo sprint, las tareas de la segunda iteración constará en el desarrollo de las siguientes tareas.

Tabla 20: Segundo Script - Pantalla Internal

ID	Nombre	Tipo	Prioridad	Owen	Tiempo (días)	Estado
In-1	Completar entrega de uno a uno	Desarrollo	**Alta**	M. Alavi	1 ½	Acabado
In-2	Completar entrega de manera multiple	Desarrollo	**Alta**	M. Alavi	1 ½	Acabado
In-3	Asignarse a multiples pedidos por el botón begin delivery	Desarrollo	**Medio**	M. Alavi	1 ½	Acabado
In-4	Asignarse a pedidos uno a uno por el botón next request	Desarrollo	**Medio**	M. Alavi	1 ½	Acabado

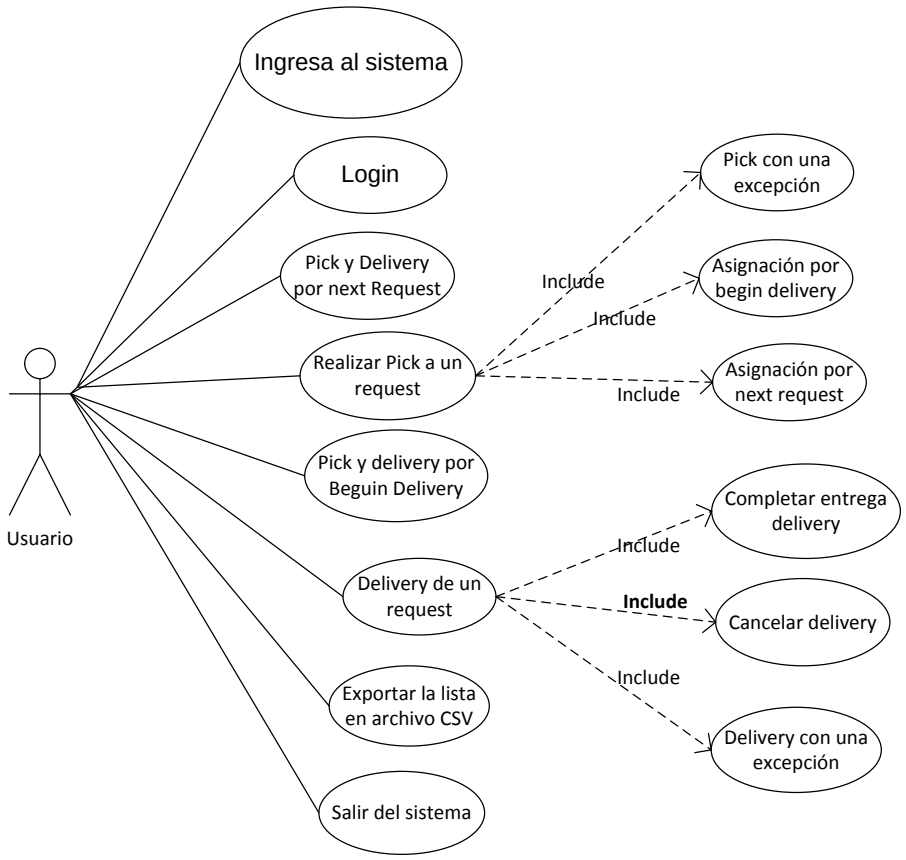
ID	Nombre	Tipo	Prioridad	Owen	Tiempo (días)	Estado
In-5	Asignarse a un pedido con mensaje de excepción	Desarrollo	**Alta**	M. Alavi	1 ½	Acabado
In-6	Completar el pedido con mensaje de excepción	Desarrollo	**Alta**	M. Alavi	1 ½	Acabado
In-8	Completar el pedido con excepción de cantidad en la pantalla de Delivere Quantity	Desarrollo	**Alta**	M. Alavi	1 ½	Acabado
In-9	Exportar la lista CSV de internal	Desarrollo	**Medio**	M. Alavi	1 ½	Acabado
In-10	Cancelar un pedido en la lista de Delivery	Desarrollo	**Bajo**	M. Alavi	1 ½	Acabado
In-11	Cancelar un pedido en la pantalla de Delivery Request Part	Desarrollo	**Bajo**	M. Alavi	1 ½	Acabado
12	Pedido con excepción de cantidad en el proceso de delivery complete	Desarrollo	**Medio**	M. Alavi	1 ½	Acabado
13	Pedido con excepción de notificación en el proceso de delivery complete	Desarrollo	**Medio**	M. Alavi	1 ½	Acabado

Fuente: Elaboración propia

4.5.2.1. Análisis de los casos de prueba para el scripts

En el siguiente diagrama que se muestran todos los casos de prueba que son aptos de la automatización de tal manera que debe reducir el tiempo de testing manual, fueron elegidos porque se espera solo un resultado es decir no es suceptible a cambios.

Diagrama 5: Casos de uso para la automatización en el Pantalla de Internal



Fuente: Elaboración propia

Tabla 21: Descripción de caso de uso para la pantalla de Internal

CU-1	**Internal – Scripts**
Descripción	Para la pantalla de internal se eligió los siguientes casos de uso, por cada uno de ellos se creara un script para su respectuva ejecución. El flujo principal para cada script son los primeros 3 pasos.

Secuencia normal	1. Ingreso a la aplicación por el login. 2. Ir a la Pantalla de Replenish/Material Request. 3. Ir a la pantalla Internal 4. Pick/Delivery de un pedido por el botón next request. 5. Pick/Delivery de uno o mas pedidos por medio del botón begin delivery. 6. Pick de un pedido con excepción. 6.1 Pick con una excepción. 6.2 Pick de un pedido por el botón begin delivery. 6.3 Pick de un pedido por el botón next request. 7. En la lista de delivery se realizara los siguientes procesos: 7.1 Delivery por medio del botón delivery complete 7.2 Cancelar un pedido por el botón cancel 7.3 Delivery con una excepción. 8. Exportar csv de la lista internal.
Excepciones	
Prioridad	Alta
Roles	
Datos	Request generados de manera manul o automatica.

Fuente: Elaboración propia

4.5.2.2. Diseño de los scripts para el Pantalla de Internal

Para la construscion de los scripts para el Pantalla de Internal se tomo en cuenta todos los querimientos especificados y el orden de los pasos son los siguientes:

Paso 1: Conexión a la base de datos para realizar consultas necesarias.

Paso 2: Configuración de los usuarios.

Paso 3: Uso del componente *HTTP Request Defaults* para realizar la conexión con la IP donde se encuentra la aplicación Flow.

Paso 4: El paso de inicio de sesión por medio de login.

Paso 5: Validar que el usuario existe para completar el test, de no existir el usuario el test no se ejecutara.

Paso 6: Realizar el proceso necesario (pick, delivery, cancelar request, excepción de un request).

Paso 7: Para completar alguno de los anteriores procesos sera necesrio realizar una consulta en especifico para cada caso del area de internal, si la consulta no devuele ningun dato el test no llevará acabo y el usuario cerrara sesión.

Paso 8: Según sea la configuración del archivo respectivo del script y su respectivo proceso podra ser ejecutado de forma infinita o definida.

Paso 9: Cerrar sesión despues de terminar con la ejecución del proceso.

Componentes de JMeter usuados en los scripts

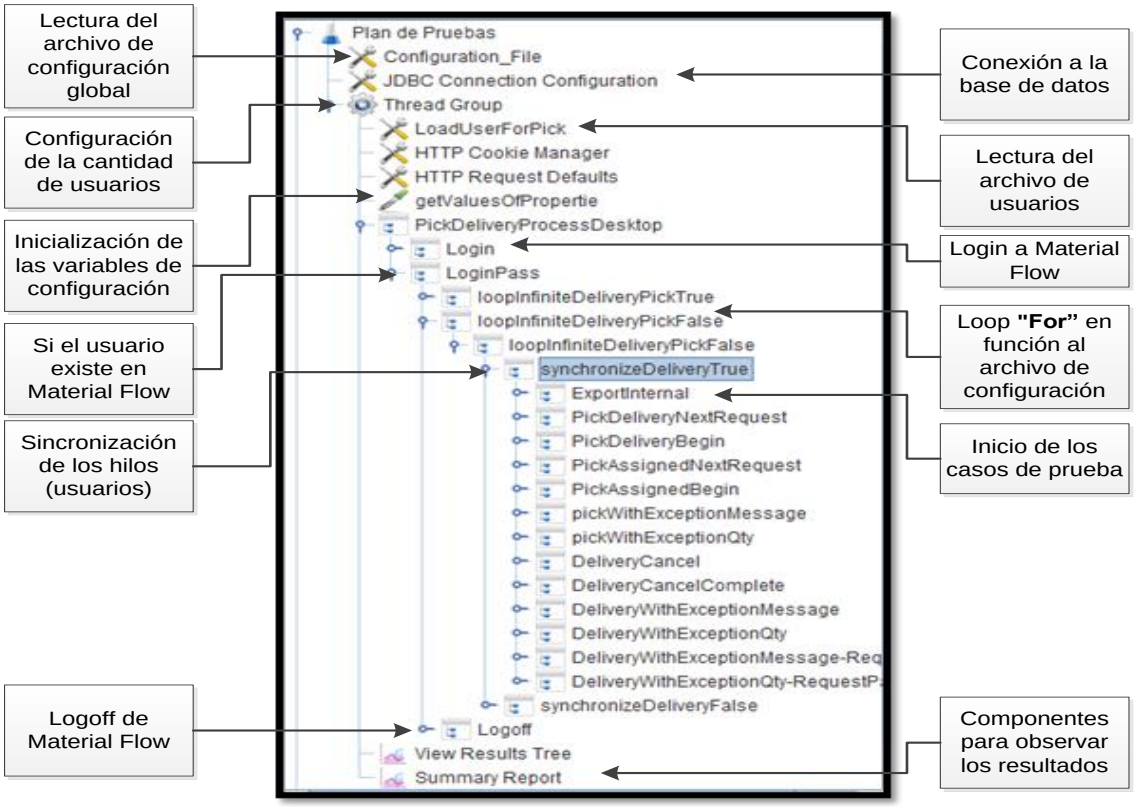
Los componentes usuados se muestran en la Ilustración 19 (Construcción del script para el proceso de entrega de pedidos en la pantalla Internal) los componentes se ejecutarán de forma secuencial en la que se muestra en la ilustración. En la tabla 22 (Lista de componentes de JMeter para los scripts) se mencionan todos los componentes que se usuaran para la construcción de todos los script de automatización.

Tabla 22: Lista de componentes de JMeter para los scripts

Nombre	Función
Group thread	Componente en el que se configura la cantidad de hilos que emularan a los usuarios.
Property fiel reader	Para poder leer archivos de extención properties, en las cueles se declara variables o propiedades para la configuración.
CSV data confi	Ayuda con la lectura de archivos de extención csv, en este caso se usará para declarar a los usuarios para completar el test.
BeanShell Sampler	Componente para realizar programación en java.
Recordin Controller	Componente en el cual se guarda los pasos de la grabación del test.
Transaction Controller	Para la agrupación de los pasos grabados.
If Controller	Para poder delimitar que pasos ejecutará el usuario.
Regex Controller	Nos ayudará obtener datos del html.
Loop Controller	Con el cual se realizará la emulación de un for para ejecutar una cantidad de veces un paso.
Foreach Controller	Se usará con relación a una consulta a la base de datos.
JDBC Request	Componente con el que se realizarán las consultas a la base datos
JDBC Connection Configuration	Componente con el que se realizará la conexion con una respectiva base de datos, por medio de una ip, usuario de DB y contraseña.
View results tree	Es el componente en el que se visualizarán el comportamiento de la ejecución del script, es decir si los pason estan correctos se mostrarán en verde y si no se ejecutaron correctamente se mostrarán en rojo

Fuente: Elaboración propia

Ilustración 19: Construción del script para el proceso de entrega de pedidos



Fuente: Elaboración propia

Archivo de configuración global (Configuration.properties)

La ejecución del script se realizo con la siguiente configuración (como se muestra en la Ilustración 20: Archivo de configuración “configuration.properties”.

- **Primera parte:** se configura todo lo relacionado con la conexión de base de datos, ip de Flow con que se testeara:
    - ipDb=10.0.0.122 -> IP de la base de datos con la cual trabaja Flow.
    - passwordDB=control123!-> es password de la base datos de Flow
    - nameDB=JRL386->Nombre de la base de datos
    - userDB=sa-> el usuario de la base datos
    - ipFlow=10.0.0.149-> IP de la maquina donde se encuentra Flow

- timeThread=10-> es el tiempo que deberia tardar en ejecutarse el test para un usuario (el tiempo esta en milisegundos)

- **Segunda parte:** se configura la cantidad de usuario que ingresarán a la aplicación a realizar el test.
  - totalUser=1-> la cantidad de usuarios que ingresarán al sistema Flow, aquí definimos si se hara un test funcional o de stres, en esta oportunidad esta configurado para un test funcional. (Si se quiere un test de stress no se recomienda usar mas de 30 hilos por que ocurrira el error de memoria)
  - rampUpPeriodo=0-> el tiempo en el que ingresaran los usuarios, como solo estamos usuando 1 hilo el tiempo por ahora no influye, sin embargo si usaramos mas de 2 el tiempo o ramup si influye, la fórmula es:

tiempo=totalUser/ rampUpPeriodo

- **Tercera parte:** configuraremos si el test sera sincronizado, cuantas veces el usuario realizará el test, la cantidad de request que procesaremos en casos de completarlos, cancelarlos o completarlos con excepciones.
  - synchronizeProcess=true->Definimos si el processo si el test sera sincronizado como true=verdadero o false=Falso
  - loopInfinite=false-> Definimos si sera un loop infinito true=verdadero o false=Falso
  - numberLoops=1-> Definimos la cantidad de iteraciones que cada usuario realizara, es decir el numero de veces que repetira el tes, en esta oportunida al ser un test funcional solo sera una iteración.
  - quantityRequest=30-> Es la cantida de request que se procesará con Pick/Delivery y en el proceso de pick, (es impotante que este número sea mucho mayor y 30 es el numero adecuado para un test exitoso).
  - cancelDelivery=2-> De los 30 request 2 de ellos se cancelarán.

- quantityException=3-> Es la cantidad de request que se procesaran con exceptiones en el proceso de Pick y 3 de los 30 en el proceso de delivery.

Ilustración 20: Archivo de configuración (configuration.properties) base del script

```
#Configuration of Flow  (Primera Parte)
ipDb=10.0.0.122
passwordDB=control123!
nameDB=JRL386
userDB=sa
ipFlow=10.0.0.149
timeThread=10

#Configure quantity users (Segunda Parte)
totalUser=1
rampUpPeriodo=0

#Configuration excute script for HandHeld and
Desktop (Tercera Parte)
synchronizeProcess=true
loopInfinite=false
numberLoops=1
quantityRequest=30
cancelDelivery=2
quantityException=3
```

Fuente: Elaboración propia

GetValueOfPropertie (Obtener valores del archivo de propiedades): Componente BeanSeshll

En este componente se realizará la lógica de extraer los valores declarados en el archivo de configuración en la ilustracion 21:

- **Primera Parte:** Recuadro de color naranja se está escribiendo en los log de JMeter para corroborar los datos introducidos en el archivo properties.
- **Segunda Parte:** Recuadro del color verde se realiza la lógica de cuanto tardara la ejecución del test, se dividió ente 4 por que el test consta de las partes login, pick, delivery y logoff.
- **Tercera Parte:** En el recuadro de color celeste obtenemos el valor de si el test es sincronizado y si es loop infinito.

- **Cuarta Parte:** Recuadro de color morado obtenemos el valor del usuario y password, también eliminamos los espacios incensarios.

Ilustración 21: GetValueOfPropertie (incialización de variables)

```
Log.info(“IP DB: “+”${__P(ipDb)}”);    (Primera Parte)
log.info("Name DB: "+"${__P(nameDB)}");
log.info("User db: "+"${__P(userDB)}");
log.info("IP Flow: "+"${__P(ipFlow)}");
log.info("Total user: "+"${__P(totalUser)}");

time=${__V(${__P(time_thread)})}; (Segunda Parte)
vars.put("timeUser", time.toString());
timeController=Integer.parseInt(vars.get("timeUser"))/4;
vars.put("timeController1",timeController.toString());
timeDelay=${__V(${__P(timeDelay)})};
vars.put("timeDelay1", timeDelay.toString());

vars.put("synchronizeProcess","${__V(${__P(synchronizeProcess)})}");(Tercera Parte)
String synchronizeProcess1=vars.get("synchronizeProcess").toLowerCase().trim();
vars.put("synchronizeProcessTrue",synchronizeProcess1.toString());

vars.put("loopInfinite","${__V(${__P(loopInfinite)})}");
String loopInfinite1=vars.get("loopInfinite").toLowerCase().trim();
vars.put("loopInfiniteTrue",loopInfinite1.toString());

String user=vars.get("userName").trim();(Cuarta Parte)
vars.put("user1",user.toString());
String password=vars.get("password").trim();
vars.put("password1",password.toString());
log.info("User name: "+vars.get("user1")+" and password: "+vars.get("password1"));
```

Fuente: Elaboración propia

Pick y Delivery, asignación a un pedido con el botón next request

Para los test case de pick/delivery, asignación a un request por next request se debe realizar consultas a la base de datos para conocer los Id’s de los request que estén asociados a una determinada ruta y al usuario con el que se ingresó al sistema Flow, también se tienen otras condiciones como ser que los requests no estén asociados a una excepción y también estén activos, al final nos devolverá los datos del ID del request y el Id del usuario, como se muestra en la ilustración 22 y 23:

Ilustración 22: Consulta SQL para obtener lista de pedidos

```
select top 1 d.JDOID as delivery, b.JDOID as userID, d.fromLocation
 from DeliveryJDOBase d, RouteJDOBase r,(select ra.route, u.JDOID from
ResourceScheduleJDO rs with (NOLOCK)
 left join ResourceAssignJDO ra with (NOLOCK)  on rs.resourceAssign =
ra.JDOID
 left join UserJDOBase u  with (NOLOCK)  on u.JDOID = ra.user0
 where u.userName like '${user1}'
 and ra.unassigned = 0
 and rs.active  = 1
 and rs.dayOfWeek = (SELECT DATEPART(dw,GETDATE()) a) )b
 where r.JDOID = d.route AND r.JDOID= b.route AND d.initialSegmentReqTime
IS nOT NULL and d.deliveryNumber = 0
 and d.exceptionNotificationBy is null
 group by d.JDOID, b.JDOID, d.fromLocation
 having d.JDOID >1
```

Fuente: Elaboración propia

Ilustración 23: Resultado de la consulta



Fuente: Elaboración propia

Después de realizar la consulta a la base datos, los valores que nos devolvió la consulta los procesamos de la siguiente manera, primero determinamos si los valores no son nulos, después si el tamaño del array es mayor a 0, guardamos los valores en variables que puedan ser usado en todo el test case (ver la ilustración 24).

Ilustración 24: Procesar variables de la consulta

```
ArrayList deliveryPickId=vars.getObject("deliveryID");
if(deliveryPickId==null){
      log.info("The delivery is null");
      }else{
      if(deliveryPickId.size()>0){
            vars.put("requestExists","true");

      vars.put("userID",vars.getObject("deliveryID").get(0).get("userID").t
oString());
```

```
      vars.put("requestPickID",vars.getObject("deliveryID").get(0).get("del
ivery").toString());
            log.info("request Id: "+vars.get("requestPickID"));
      }else{
            vars.put("requestExists","false");
            vars.put("testResult","Bloqued");
            log.info("The delivery no exist");
      }}
```

Fuente: Elaboración propia

Pick/Delivery, Asignación a un pedido por el botón Begin Delivery

Para los test case de pick/delivery, asignación a un request por begin delivery se debe realizar consultas a la base de datos para conocer los Id's de los request que estén asociados a una determinada ruta y al usuario con el que se ingresó al sistema Flow, también se tienen otras condiciones como ser que los requests no estén asociados a una excepción y también estén activos, al final nos devolverá los datos del ID del request, el Id del usuario y también la cantidad que se especificó en el archivo propertie (ilustración 25):

Ilustración 25: Consulta para los request por begin delivery

```
select top ${__P(quantityRequest)} d.JDOID as delivery, b.JDOID as userID
 from DeliveryJDOBase d, RouteJDOBase r,(select ra.route, u.JDOID from
ResourceScheduleJDO rs with (NOLOCK)
 left join ResourceAssignJDO ra with (NOLOCK)  on rs.resourceAssign =
ra.JDOID
 left join UserJDOBase u  with (NOLOCK)  on u.JDOID = ra.user0
 where u.userName like '${user1}'
 and ra.unassigned = 0
 and rs.active  = 1
 and rs.dayOfWeek = (SELECT DATEPART(dw,GETDATE()) a) )b
 where r.JDOID = d.route AND r.JDOID= b.route AND d.initialSegmentReqTime
IS nOT NULL and d.deliveryNumber = 0
 and d.exceptionNotificationBy is null
 group by d.JDOID, b.JDOID
 having d.JDOID >1
```

Fuente: Elaboración propia

Para este test case la forma de procesar los datos serán distintos, ya que en ciertos lugares del script nos pide los datos o id's estén separados por % y en otro por comas, también validamos que realmente se tenga datos antes de realizar el test case (ilustración 26).

Ilustración 26: Procesar datos en formatos determinados.

```
ArrayList listOfDeliveryIds=vars.getObject("deliveryID");
String requestID="";
String requestIDajax="";
int size=listOfDeliveryIds.size();
int total=0;
for(Object deliveryId:listOfDeliveryIds )
{
	if(total+1==size){
		requestID=requestID+deliveryId.get("delivery");
		requestIDajax=requestIDajax+deliveryId.get("delivery");
		vars.put("allDelivery",requestID.toString());
		vars.put("ajaxAllDelivery",requestIDajax.toString());
		log.info("Request ids: " + vars.get("allDelivery"));
		log.info("Request ids ajax: " + vars.get("ajaxAllDelivery"));
	}
	else  {
		requestID=requestID+deliveryId.get("delivery")+",";
		requestIDajax=requestIDajax+deliveryId.get("delivery")+"%2C";
	}
total = total + 1;
}

if(requestID.length()>0){
	vars.put("existsRequest","true");
}
else{
	vars.put("existsRequest","false");
	vars.put("testResult","Bloqued");
	log.info("Does no exists request");
	}
```

Fuente: Elaboración propia

Consulta para obtener el ID de excepciones

Para los test case que necesitarán completarse con excepciones, es necesario obtener el id de la excepción que también esté relacionado al mismo grupo del usuario, por lo tanto se realizar dos tipos de consulta con una obtenemos una excepción de cantidad y otra de notificación (ilustración 27 y 28).

Ilustración 27: Consulta pare excepción de notificación

```
SELECT TOP 1 e.JDOID
FROM ExceptionConditionJDOBase e, UserJDOBase u
where exceptionNotification = 1 and u.userName like
'${user1}' and e.assignedGroup=u.homeGroup
order by e.JDOID asc
```

Fuente: Elaboración propia

Ilustración 28: Consulta para excepción de cantidad

```
SELECT TOP 1 e.JDOID, e.name0
FROM ExceptionConditionJDOBase e, UserJDOBase u
where e.deliveredQuantity = 1 and e.exceptionNotification=0 and u.userName
like '${user1}' and e.assignedGroup=u.homeGroup
order by e.JDOID asc
```

Fuente: Elaboración propia

Test case relacionado al proceso de delivery

Para los test case que completaran por medio del proceso de delivery la consulta consistirá en obtener la cantidad especificada en el archivo de configuración, id's de los requets, id del usuario, las condiciones son que deben estar activos, asignados a la ruta y usuario, no relacionado a una excepción (ver la ilustración 29).

Ilustración 29: Consulta para los request en la lista de Delivery

```
select top ${__P(cancelDelivery)} d.JDOID as delivery, b.JDOID as userID
 from DeliveryJDOBase d, RouteJDOBase r,(select ra.route, u.JDOID from
ResourceScheduleJDO rs with (NOLOCK)
 left join ResourceAssignJDO ra with (NOLOCK)  on rs.resourceAssign =
ra.JDOID
 left join UserJDOBase u  with (NOLOCK)  on u.JDOID = ra.user0
 where u.userName like '${user1}'
 and ra.unassigned = 0
 and rs.active  = 1
 and rs.dayOfWeek = (SELECT DATEPART(dw,GETDATE()) a) )b
 where r.JDOID = d.route AND r.JDOID= b.route AND d.initialSegmentReqTime
IS nOT NULL and d.deliveryNumber > 0 and d.exceptionCondition is null
 group by d.JDOID, b.JDOID
 having d.JDOID >1
```

Fuente: Elaboración propia

Después de obtener todos los datos necesarios como el Id del request, el Id del usuario asignado y también el Id de la excepción si es necesario o según el test case, se guardaran los datos en variables que puedan ser usadas en el script, de tal manera que de debemos validar los datos antes de que se usen, por ejemplo si enviamos datos vacíos o erróneos el script fallará y tendremos como resultado un "Fail" en el reporte (ver la ilustración 30).

Ilustración 30: Proceso de datos para la es de delivery y distintos test case

```
ArrayList listOfDeliveryIds=vars.getObject("deliveryID");
String requestID = "";
int size = listOfDeliveryIds.size();
int total = 0;
for(Object  deliveryId:listOfDeliveryIds )
{
	if(total+1==size){
		requestID= requestID+deliveryId.get("delivery");
		vars.put("deliveryIdList1",requestID.toString());
		log.info("Request ids: " +
vars.get("deliveryIdList1"));
		vars.put("sizeList", size.toString());
	}
	else{
		requestID=requestID+deliveryId.get("delivery")+",";
	}
	total=total+1;
}

if(requestID.length()>0){
	vars.put("existsRequest","true");
}
else{
	vars.put("existsRequest","false");
	vars.put("testResult","Bloqued");
	log.info("Does no exists request");
}
```

Fuente: Elaboración propia

Control de tiempos de ejecución y validación de los casos de prueba

Para llevar un control de los tiempos en los que ejecutan los test case de tal manera con los datos obtenidos podamos compararlos con los tiempos de un test case manual, para este control se obtiene el tiempo del sistema y también se inicia con la sincronización según la configuración del archivo propertie (ver la ilustración 31).

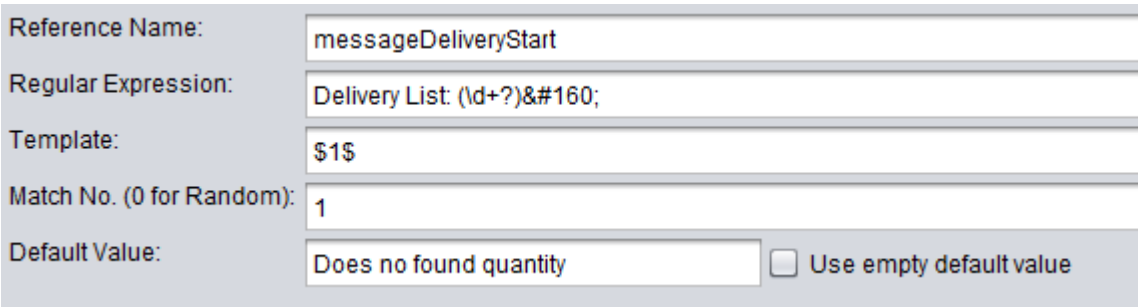
Ilustración 31: Tiempo inicial de los test case

```
import SyncronizeObject.*;
Long start = System.nanoTime();
vars.put("startPickDelivery", start.toString());
SyncronizeObject.writeLock.lock();
```

Fuente: Elaboración propia

En la parte donde está el “try - catch” es el lugar donde se valida si el test case se ejecutó el correctamente, teniendo como parámetros un valor inicial y otro valor final, esto producto de un análisis de la página por medio del componente de regex y la siguiente cadena, en algunos casos esta validación se hace con una consulta para contar los request inicial y otra al finalizar el test case, como se muestran en las ilustraciones 32, 33 y 34:

Ilustración 32: Regex para el análisis de la página



Fuente: Elaboración propia

Ilustración 33: Consulta para contar los request con excepciones

```
select count (d.JDOID) as countRequest
 from DeliveryJDOBase d, RouteJDOBase r,(select ra.route, u.JDOID from
ResourceScheduleJDO rs with (NOLOCK)
 left join ResourceAssignJDO ra with (NOLOCK)  on rs.resourceAssign =
ra.JDOID
 left join UserJDOBase u  with (NOLOCK)  on u.JDOID = ra.user0
 where u.userName like '${user1}'
 and ra.unassigned = 0
 and rs.active  = 1
 and rs.dayOfWeek = (SELECT DATEPART(dw,GETDATE()) a) )b
 where r.JDOID = d.route AND r.JDOID= b.route AND d.initialSegmentReqTime
IS not NULL
 and d.deliveryNumber > 0 and d.exceptionCondition is not null
```

Fuente: Elaboración propia

Ilustración 34: Tiempo final y validación de los test case

```
import java.util.*;
import java.text.*;
import SyncronizeObject.*;

NumberFormat formatter = new DecimalFormat("#0.00");
Long endPickDelivery=System.nanoTime();
long elapsedTimePickDelivery = endPickDelivery - Long.parseLong(
vars.get("startPickDelivery"));
```

```
Double secondsPickDelivery = ((Double) elapsedTimePickDelivery) /
1000000000.0;
vars.put("secondsPickDelivery",secondsPickDelivery.toString());
log.info("Real Time Pick Delivery : "+secondsPickDelivery);
timeController=Integer.parseInt(vars.get("timeController1"));

      if(secondsPickDelivery<timeController){
            Long secondsDelayPickDelivery=(long)(timeController-
secondsPickDelivery) + (long)secondsPickDelivery;

      vars.put("secondsDelayPickDelivery1",secondsDelayPickDelivery.toStrin
g());
            vars.put("Pick_Delivery_Delay_Taken",
secondsDelayPickDelivery.toString());
            }
      else{
            log.info("Time taken Pick Delivery : "+secondsPickDelivery);
            vars.put("secondsPickDelivery1","0");
      Long secondsPickDelivery1=(long) secondsPickDelivery;
            vars.put("Pick_Delivery_Delay_Taken",
secondsPickDelivery1.toString());
            }
try{
      int numberStart=Integer.parseInt(vars.get("countRequestqty"));
      int
numberInternal=Integer.parseInt(vars.get("messageInternalStart"));
      vars.put("countRequestFinishqty",vars.getObject("countRequestFinsh").
get(0).get("countRequestFinish").toString());
      int numberFinsh=Integer.parseInt(vars.get("countRequestFinishqty"));
      if(numberInternal>0 && numberFinsh>numberStart){
            vars.put("testResult","Pass");
            vars.put("quantityFinsh", numberFinsh.toString());
      }
      else{
            if(numberInternal>0 && numberFinsh>numberStart){
                  vars.put("testResult","Fail");
                  vars.put("quantityFinsh", numberFinsh.toString());
            }
            else{
                  vars.put("testResult","Bloqued");
                  vars.put("quantityFinsh", numberFinsh.toString());
            }
      }
}
catch(NumberFormatException e){
      vars.put("quantityFinsh","Does no found quantity");
      vars.put("error",e.toString());
      log.error(vars.get("error"));
      }
SyncronizeObject.writeLock.unlock();
```

Fuente: Elaboración propia

Generar el reporte después de la ejecución del test case

Después de ejecutarse el test case sin importar si fallo o fue exitoso se debe generar un reporte con los resultados, por esa razón en esta parte generamos el reporte en formato csv, el nombre del archivo consta de un nombre predefinido concatenado con la fecha actual ejemplo “PickDeliveryDesktop2018-06-28”, el reporte consta del número build del sistema Flow, el nombre del test case, resultado, tiempo de ejecución y la fecha de ejecución (ver la ilustración 35).

Ilustración 35: Generar el reporte del test case

```
import org.apache.jmeter.services.FileServer;
import java.text.SimpleDateFormat;

//Date for CSV
SimpleDateFormat formatDate = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
java.util.Date date = new Date();
String format=formatDate.format(date);
//Date for name CSV
SimpleDateFormat formatDateCSV = new SimpleDateFormat("yyyy-MM-dd");
java.util.Date datecsv = new Date();
String formatcsv=formatDateCSV.format(datecsv);
quantityFinsh1=vars.get("quantityFinsh");
testResult1=vars.get("testResult");
secondsPickDelivery1=vars.get("secondsPickDelivery");
messageInternalStart1=vars.get("countRequestqty");
userTest=vars.get("user1");
versionMessage1=vars.get("versionMessage");

file = new FileOutputStream (FileServer.getFileServer().getBaseDir() + "\\
PickDeliveryDesktop"+formatcsv+".csv", true);
print = new PrintStream (file);

print.println (versionMessage1+",Test case: Pick with exception message"
+",User: "+userTest+",Quantity request
initial:"+messageInternalStart1+",Quantity request final:
"+quantityFinsh1+","+testResult1+",Date: "+format+",Time:
"+secondsPickDelivery1);

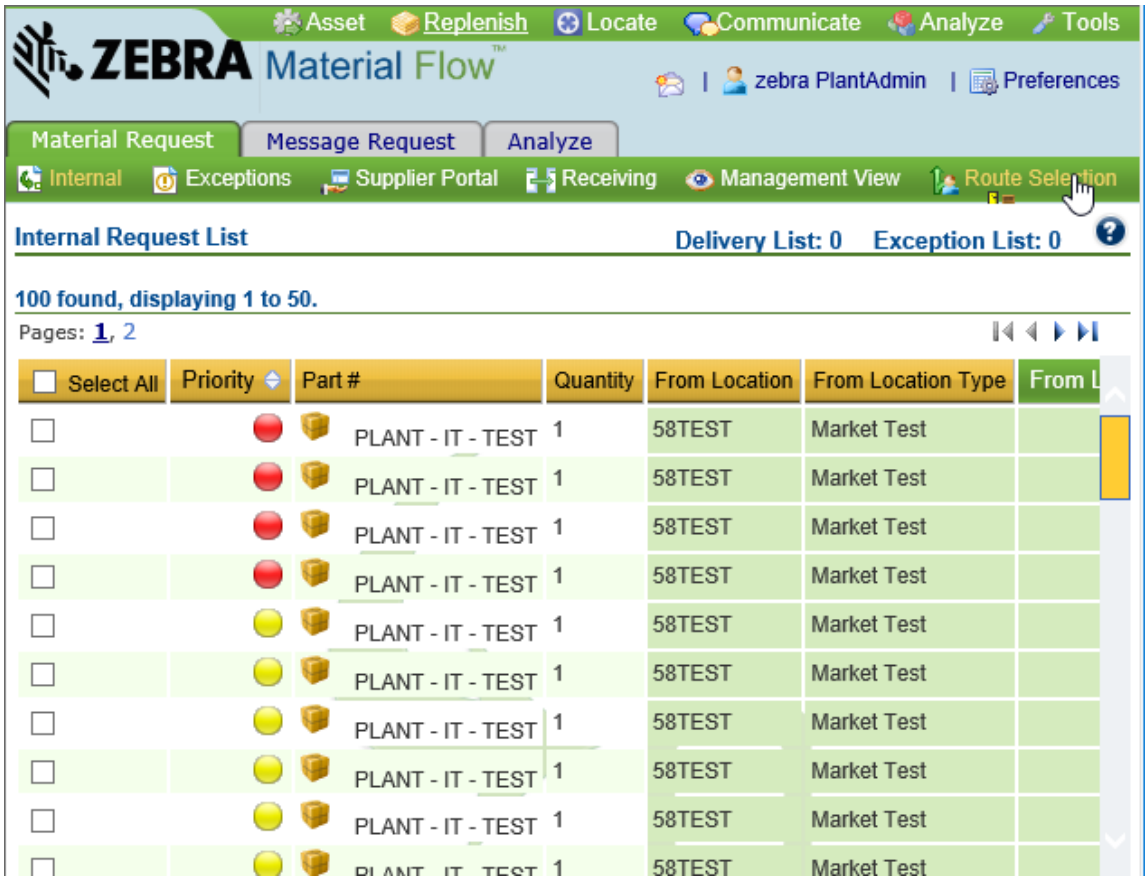
// Cerrar archivo (s)
print.close ();
file.close ();
```

Fuente: Elaboración propia

4.5.2.3. Implementación de los scripts automatizados para la Pantalla de Internal

La ejecucion del script para el Pantalla de Internal se inicio con 100 request para ser procesados teniendo en cuenta todos los test case del Test plan que fueron automatizados (ver la ilustración 36).

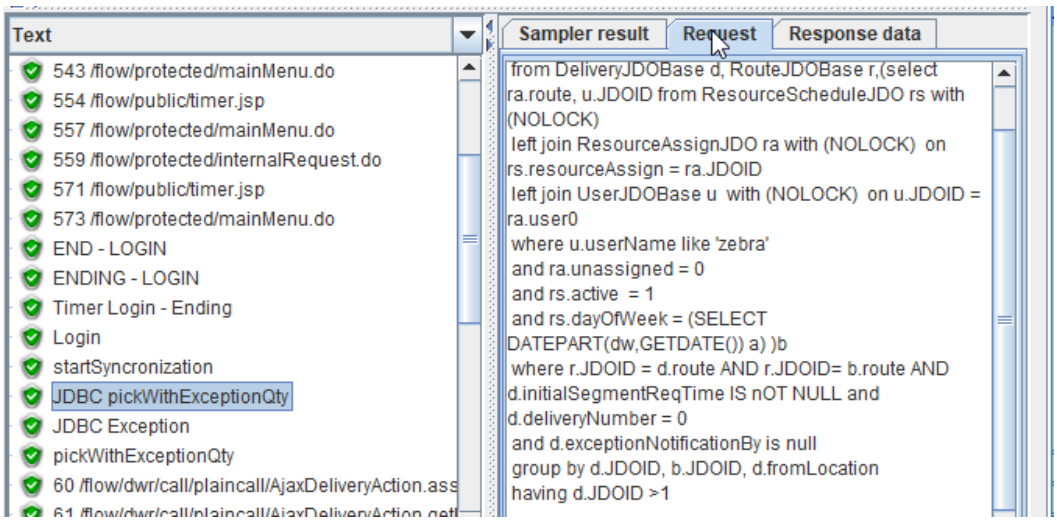
Ilustración 36: Estado inicial de Internal



Fuente: Elaboración propia

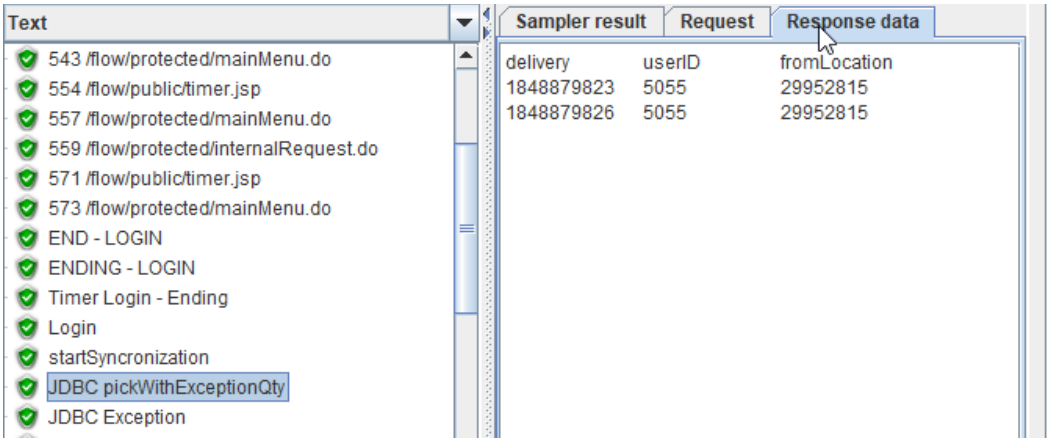
En las ilustraciones 37 y 38 se mostrarán la ejecución del script y uno de los querys a la base de datos, los “IDs” de los request para completa con éxito cada test case automatizado, la última imagen mostrará el estado en que quedo Internal después de la ejecución del script.

Ilustración 37: Request a completar con excepción de cantidad



Fuente: Elaboración propia

Ilustración 38: IDs de los request



Fuente: Elaboración propia

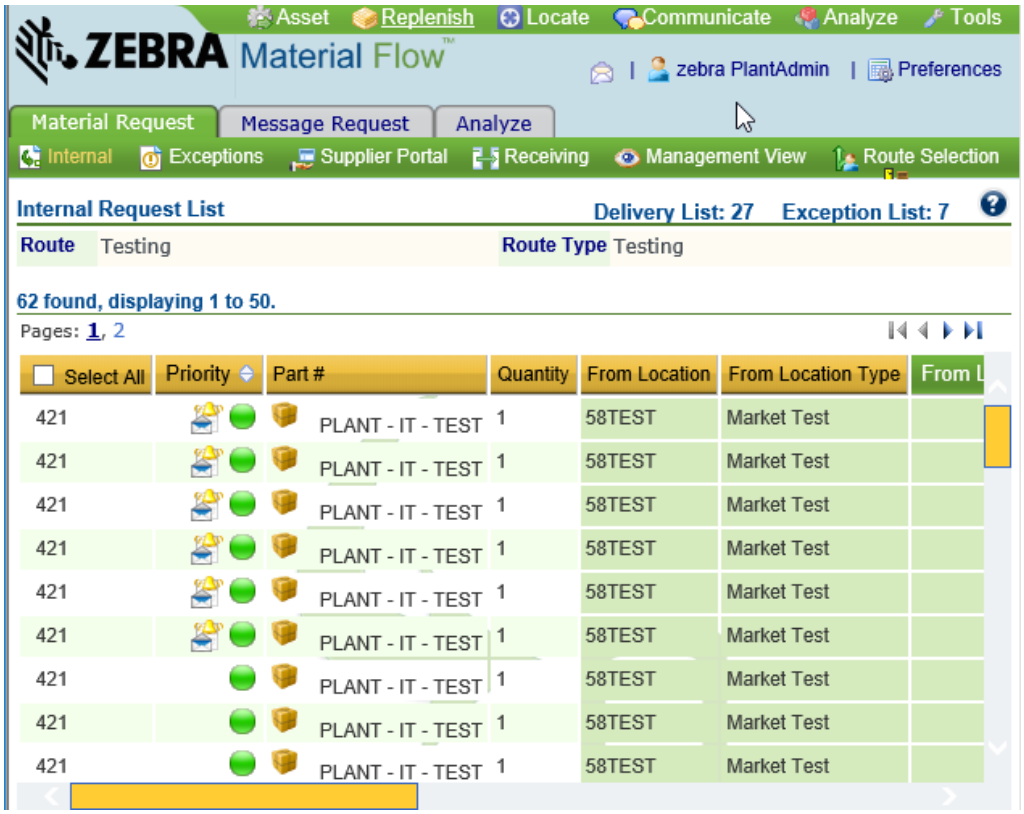
Estado final después de la ejecución del script

Los request en el Pantalla de Internal se redujo notablemente, con la configuración especifico los resultados son (ver la ilustración 39):

- La cantidad de request completados por pick/delivery son 31.
- Existen 69 request entre la lista de delivery, excepción y pick

- En la lista de excepción existen 7 requets, 3 de ellos son producto del test case de delivery excepción con cantidad, 3 por medio de test case delivery complete y 1 por pick con excepción de cantidad
- En la lista de delivery existen 27 requets, de los cuales 7 request son request con excepcion message completados por los test case de pick, delivery y delivery complete con excepction message.

Ilustración 39: Estado final del área de Internal



Fuente: Elaboración propia

El reporte que se creo fue para un mejor control de la ejecución del script, pues ocurren los casos que en el script nos muestra que todo se cumplio al 100% sin errores pero al revisar la apliacación nos encontramos que no se cumplieron los test case y no se tiene una razón o en donde ocurrio el fallo. Los datos que tenermos es en que version se ejecuto el script, cual es el nombre del test case, que usuario la cumplio, las cantidades de antes y despues de que un test case

se ejecute, el resultado se paso o fallo, la fecha y hora y cuanto tiempo tardo en cumplir con la ejecución del test case, los reportes se puden observar en la parte de anexos como "**Pick/Delivery por desktop**".

4.5.3. Tercer Sprint - Script para la Pantalla de Excepciones

En el tercer sprint se dará a conocer la cantidad de script para la pantalla de excepciones, después de realizar el análisis respectivo, en la tabla 23 se muestra la información general del tercer sprint.

Tabla 23: Tercer sprint

Sprint	Historia	Fecha de inicio	Duración (días)
3	Script para el Pantalla de Excepciones en los casos de prueba	13/05/2018	10

Fuente: Elaboración propia

En la tabla 24 se especificara la lista de las tareas que son necesarias para concluir el tercer sprint, las tareas de la tercera iteración constara en el desarrollo de las siguientes tareas.

Tabla 24: Tercer Sprint - Pantalla de Excepciones

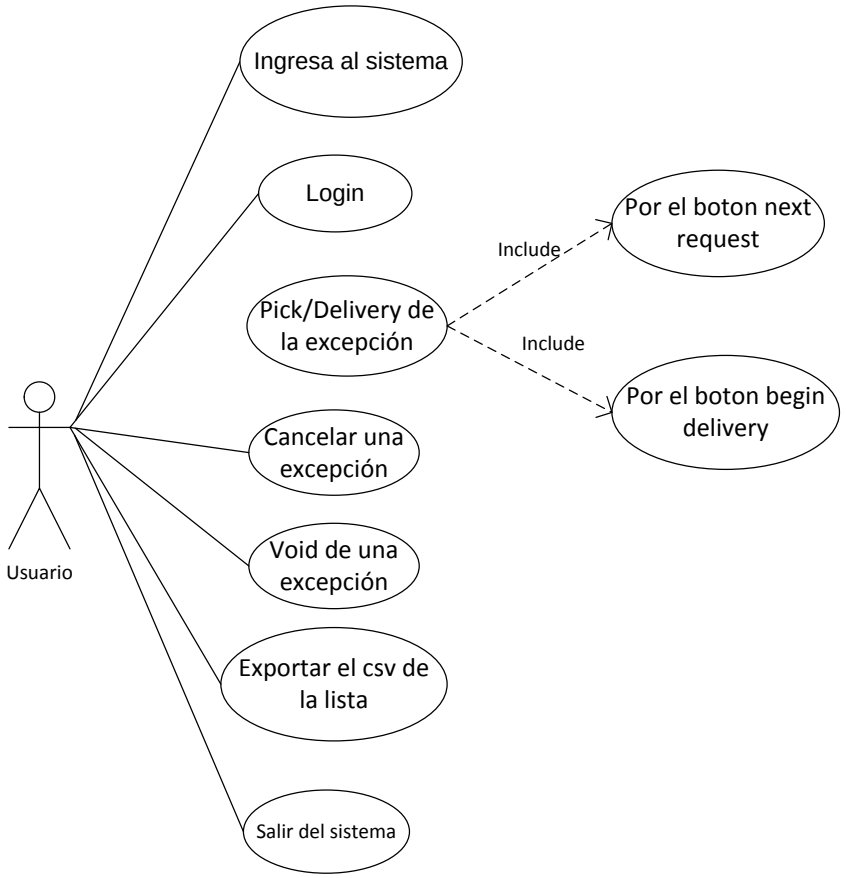
ID	Nombre	Tipo	Prioridad	Owen	Duración (días)	Estado
Exp-1	Completar el pedido con excepción de uno a uno	Desarrollo	**Medio**	M. Alavi	1 ½	Terminado
Exp-2	Completar el pedido con excepción de manera múltiple	Desarrollo	**Medio**	M. Alavi	1 ½	Terminado

Exp-3	Cancelar una excepción	Desarrollo	**Alta**	M. Alavi	1 ½	Terminado
Exp-4	Completar la excepción	Desarrollo	**Alta**	M. Alavi	1 ½	Terminado
Exp-5	Exportar la lisa CSV de la pantalla de excepción	Desarrollo	**Medio**	M. Alavi	1 ½	Terminado

Fuente: Elaboración propia

4.5.3.1. Análisis de los casos de uso para los scripts

Diagrama 6: Casos de uso para el Pantalla de excepción



Fuente: Elaboración propia

Tabla 25: Descripción del diagrama de caso de uso

CU-2	Excepciones – Scripts
Descripción	Para el Pantalla y pantalla de excepciones se crearan scripts para los siguientes casos de uso. Los 3 primeros pasos seran replicados en todos los scripts.
Secuencia normal	1) Ingreso a la apliacion por el login 2) Ir a Replenish/Material Request 3) Ir a la lista de excepciones 4) En esta lista se se ejecutaran los siguientes scripts: - Cancelar una excepción. - Uno o mas pedidos con excepciones seran completodos por el proceso void. - Pick/Delivery de una excepción por medio del botón next request. - Pick/Delivery de una o mas excepción por medio del botón begin delivery. - Exportación de la lista de excepciones en csv
Excepciones	
Prioridad	Alta
Roles	QA.
Datos	Request enviados desde pick o delivery.

Fuente: Elaboración propia

4.5.3.2. Diseño de los scripts para la Pantalla de Excepciones

Para la construscion de los scripts para la pantalla de excepciones se tomo en cuenta todos los querimientos especificados y el orden de los pasos son los siguientes:

Paso 1: Conexión a la base de datos para realizar consultas necesarias.

Paso 2: Configuración de los usuarios.

Paso 3: Uso del componente *HTTP Request Defaults* para realizar la conexión con la IP donde se encuentra la apliacion Flow.

Paso 4: El paso de inicio de sesión por medio de login.

Paso 5: Validar que el usuario existe para completar el test, de no existir el usuario el test no se ejecutara.

Paso 6: El test ira a la pantalla de excepciones donde realizará alguno de los procesos.

- cancelar una excepción
- void de una excepción
- exportación de la lista
- pick/delivery de excepciones por next request o begin delivery

✓ Para completar alguno de los anteriores procesos sera necesario realizar una consulta en especifico para cada caso del Pantalla de excepciones, si la consulta no devuele ningun dato el test no llevara acabo y el usuario cerrara sesión.

✓ Según sea la configuracion del archivo respectivo del script y su respectivo proceso podra ser ejecutado de forma infinita o definida.

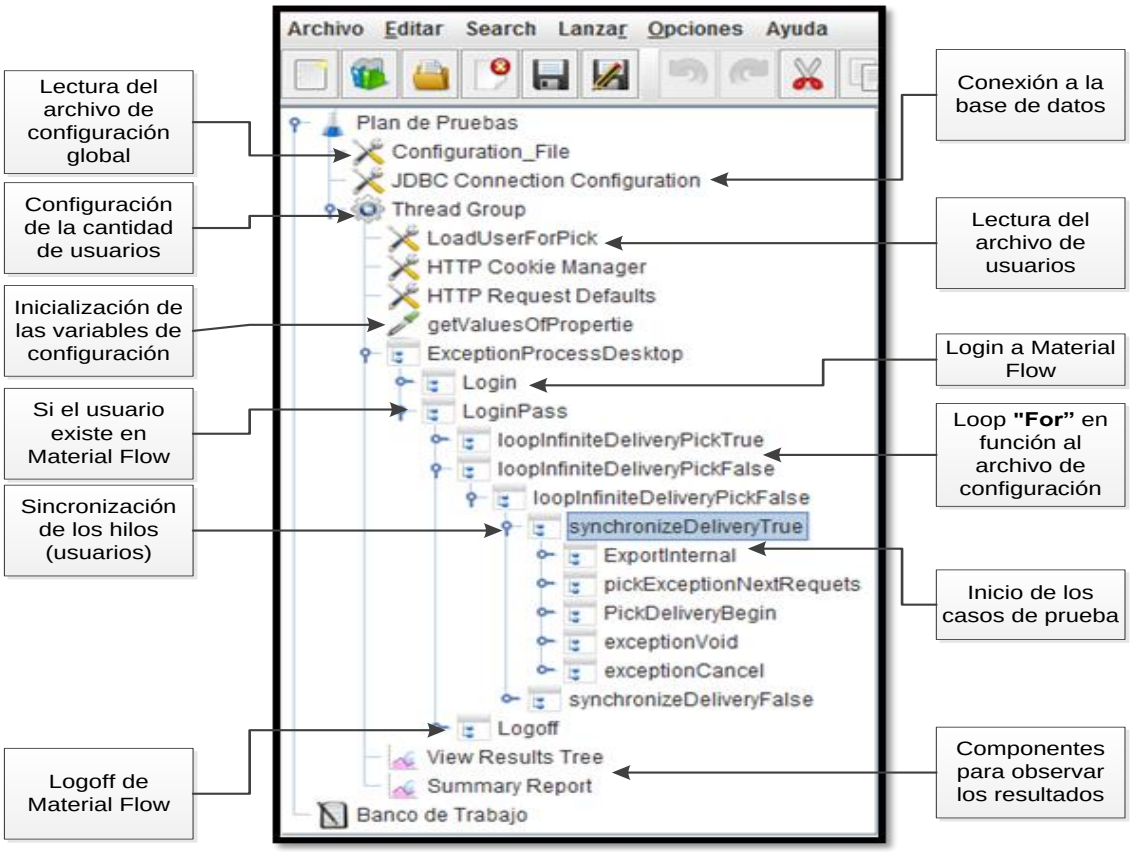
✓ Cerrar sesión despues de terminar con la jecución del proceso.

Componentes de JMeter usuados en los scripts

Nombre	Función
Group thread	Componente en el que se configura la cantidad de hilos que emularan a los usuarios.
Property fiel reader	Para poder leer archivos de exención properties, en las cueles se declara variables o propiedades para la configuración.
CSV data confi	Ayuda con la lectura de archivos de extención csv, en este caso se usara para declarar a los usuarios para completar el test.
BeanShell Sampler	Componente para realizar programacion en java.

Nombre	Función
Recordin Controller	Componente en el cual se guarda los pasos de la grabación del test.
Transaction Controller	Para la agrupación de los pasos grabados.
If Controller	Para poder delimitar que pasos ejecutará el usuario.
Regex Controller	Nos ayudará obtener datos del html.
Loop Controller	Con el cual se realizará la emulación de un for para ejecutar una cantidad de veces un paso.
Foreach Controller	Se usará con relación a una consulta a la base de datos.
JDBC Request	Componente con el que se realizaran las consultas a la base datos
JDBC Connection Configuration	Componente con el que se realizara la conexión con una respectiva base de datos, por medio de una ip, usuario de DB y contraseña.
View results tree	Es el componente que se usará para el seguimiento de la ejecución del script de tal manera que podamos controlar que pasos se ejecutaron correctamente y cuales no.

Ilustración 40: Diseño del script para la Pantalla de Excepciones



Fuente: Elaboración propia

La configuración para este script y test case es la misma del archivo properti que se creo para el primer sprint, por lo tanto no es necesario explicarlo de nuevo, sin embargo se mostrara las consultas a las base datos que de igual manera son necesarios para que pueda funcionar para varios test case y varios request.

Test case para los pedidos en la lista de Exception

Para poder ejecutar y procesar los request de la lista de Exception es necesario realizar una consulta donde le cambio es la cantidad de datos obtendremos en relación a la configuración del archivo propertie, la condicion

fundamental es que esten relacionado con una ruta y usuario especifico tambien que su excepción no sea de notificación (ver la ilustración 41).

Ilustración 41: Consulta para request en la lista de excepciones

```
select top ${__P(quantityException)}  d.JDOID as delivery, b.JDOID as
userID
 from DeliveryJDOBase d, RouteJDOBase r,(select ra.route, u.JDOID from
ResourceScheduleJDO rs with (NOLOCK)
 left join ResourceAssignJDO ra with (NOLOCK)  on rs.resourceAssign =
ra.JDOID
 left join UserJDOBase u  with (NOLOCK)  on u.JDOID = ra.user0
 where u.userName like '${user1}'
 and ra.unassigned = 0
 and rs.active  = 1
 and rs.dayOfWeek = (SELECT DATEPART(dw,GETDATE()) a) )b
 where r.JDOID = d.route AND r.JDOID= b.route AND d.initialSegmentReqTime
IS not NULL and d.deliveryNumber = 0
 and d.exceptionNotificationBy =b.JDOID and d.deliveryStatus=10
 group by d.JDOID, b.JDOID
 having d.JDOID >1
```

Fuente: Elaboración propia

Despues de que la consulta se haya ejecutado y nos devuelva los datos necesarios, estos los guardaremos en variables, realizaremos las validacionces necesarias si es nulo, si realmente tiene datos y dependiendo del test case lo guardaremos en un formato determinado, es decir los Id's separados por comas o porcentajes para los pasos que son ajax (ver las ilustraciones 42 y 43).

Ilustración 42: Request para ser procesados por next request

```
ArrayList listOfDeliveryIds =  vars.getObject("deliveryExceptionID");

if(listOfDeliveryIds==null){
      log.info("Does no exists request ");
      }
else{
      if(listOfDeliveryIds.size()>0){
            vars.put("requestExists","true");
            vars.put("requestExceptionId",
vars.getObject("deliveryExceptionID").get(0).get("delivery").toString());
            log.info("Request ID: "+vars.get("requestExceptionId"));
            }
      else{
```

```
            log.info("Does no exists request");
            vars.put("testResult","Bloqued");
            vars.put("requestExists","false");
            }
      }
```

Fuente: Elaboración propia

Ilustración 43: Request para procesarlos en cantidades

```
ArrayList listOfDeliveryIds =  vars.getObject("deliveryID");
String requestID = "";
String requestIDs = "";
int size = listOfDeliveryIds.size();
int total = 0;
for(Object  deliveryId:listOfDeliveryIds )
{
      if(total+1==size){
            requestID= requestID+deliveryId.get("delivery");
            requestIDs= requestIDs+deliveryId.get("delivery");
            vars.put("pickID",requestID.toString());
            log.info("user name: "+vars.get("user1")+" Request ids: " +
vars.get("pickID"));
            vars.put("sizeList", size.toString());
      }else{
            requestID=requestID+deliveryId.get("delivery")+",";
      }
      total=total+1;}
if(requestID.length()>0){
      vars.put("existsRequest","true");
}else{
      vars.put("existsRequest","false");
      vars.put("testResult","Bloqued");
      log.info("Does no exists request");
      }
```

Fuente: Elaboración propia

Control de tiempos de ejecución y validación de los casos de prueba

Para llevar un control de los tiempos en los que ejecutan los test case de tal manera con los datos obtenidos podamos compararlos con los tiempos de un caso de prueba manual, para este control se obtiene el tiempo del sistema en

nano tiempo y también se inicial con la sincronización según la configuración del archivo property (ver la ilustración 44).

Ilustración 44: Control de tiempo de ejecución inicial

```
import SyncronizeObject.*;
Long start = System.nanoTime();
vars.put("startPickDelivery",
start.toString());
SyncronizeObject.writeLock.lock();
```

Fuente: Elaboración propia

Al finalizar la ejecución del test case se obtendrá nuevamente el tiempo del sistema de tal forma que restando con el tiempo inicial se tendrá un tiempo de ejecución, también se valida si el test case fue exitoso o fallido después de comparar datos obtenidos de un análisis de la página con el componente regex con el resultado de la comparación de cantidades de determina si es Pass o Fail, como se observa en las ilustraciones 45 y 46.

Ilustración 45: Control de tiempo y validación del test case

```
import java.util.*;
import java.text.*;
import SyncronizeObject.*;

NumberFormat formatter = new DecimalFormat("#0.00");
Long endPickDelivery=System.nanoTime();
long elapsedTimePickDelivery = endPickDelivery - Long.parseLong(
vars.get("startPickDelivery"));
Double secondsPickDelivery = ((Double) elapsedTimePickDelivery) /
1000000000.0;
```

```
vars.put("secondsPickDelivery",secondsPickDelivery.toString());
log.info("Real Time Pick Delivery : "+secondsPickDelivery);
timeController=Integer.parseInt(vars.get("timeController1"));

      if(secondsPickDelivery<timeController){
            Long secondsDelayPickDelivery=(long)(timeController-
secondsPickDelivery) + (long)secondsPickDelivery;

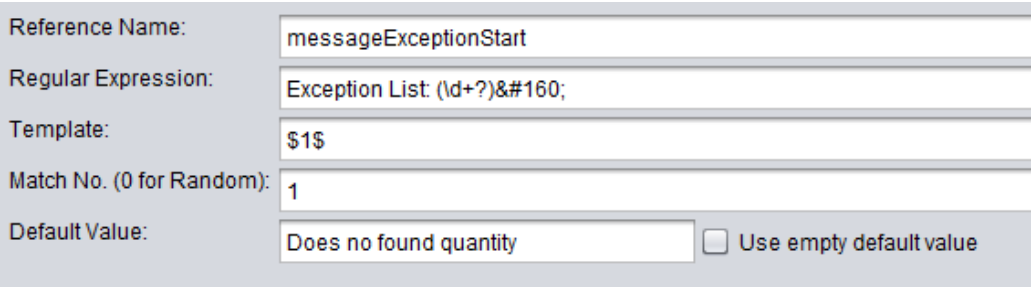
      vars.put("secondsDelayPickDelivery1",secondsDelayPickDelivery.toStrin
g());
            vars.put("Pick_Delivery_Delay_Taken",
secondsDelayPickDelivery.toString());
            }
      else{
            log.info("Time taken Pick Delivery : "+secondsPickDelivery);
            vars.put("secondsPickDelivery1","0");
      Long secondsPickDelivery1=(long) secondsPickDelivery;
            vars.put("Pick_Delivery_Delay_Taken",
secondsPickDelivery1.toString());
            }
try{
      int numberStart=Integer.parseInt(vars.get("messageExceptionStart"));
      int numberFinsh=Integer.parseInt(vars.get("messageExceptionFinish"));
      if(numberStart>0 && numberStart>numberFinsh){
            vars.put("testResult","Pass");
            vars.put("quantityFinsh", numberFinsh.toString());
      }
      else{
            if(numberStart==0){
                  vars.put("testResult","Bloqued");
                  vars.put("quantityFinsh", numberFinsh.toString());
                  }
            else{
                  vars.put("testResult","Fail");
```

```
                    vars.put("quantityFinsh", numberFinsh.toString());
                }
        }
}
catch(NumberFormatException e){
        vars.put("quantityFinsh","Does no found quantity");
        vars.put("error",e.toString());
        log.error(vars.get("error"));
}

SyncronizeObject.writeLock.unlock();
```

Fuente: Elaboración propia

Ilustración 46: Regex para obtener la cantidad de la lista de exception



Fuente: Elaboración propia

Generar el reporte después de la ejecución del test case

Después de ejecutarse el test case sin importar si fallo o fue exitoso se debe generar un reporte con los resultados, el reporte en formato csv, el nombre del archivo consta de un nombre predefinido concatenado con la fecha actual ejemplo “ExceptionProcessDesktop2018-06-28”, el reporte consta del número build del sistema Flow, el nombre del test case, resultado, tiempo de ejecución y la fecha de ejecución (ver la ilustración 47).

Ilustración 47: Generación del reporte

```
import org.apache.jmeter.services.FileServer;
import java.text.SimpleDateFormat;

//Date for CSV
SimpleDateFormat formatDate = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
java.util.Date date = new Date();
String format=formatDate.format(date);
//Date for name CSV
SimpleDateFormat formatDateCSV = new SimpleDateFormat("yyyy-MM-dd");
java.util.Date datecsv = new Date();
String formatcsv=formatDateCSV.format(datecsv);
quantityFinsh1=vars.get("quantityFinsh");
testResult1=vars.get("testResult");
secondsPickDelivery1=vars.get("secondsPickDelivery");
messageInternalStart1=vars.get("messageExceptionStart");
userTest=vars.get("user1");
versionMessage1=vars.get("versionMessage");

file = new FileOutputStream (FileServer.getFileServer().getBaseDir() + "\\
ExceptionProcessDesktop"+formatcsv+".csv", true);
print = new PrintStream (file);
print.println (versionMessage1+",Test case: Pick delivery exception by
begin delivery" +",User: "+userTest+",Quantity request
initial:"+messageInternalStart1+",Quantity request final:
"+quantityFinsh1+","+testResult1+",Date: "+format+",Time:
"+secondsPickDelivery1);
// Cerrar archivo (s)
print.close ();
file.close ();
```

Fuente: Elaboración propia

4.5.3.3. Implementación de los scripts automatizados para la Pantalla de Excepciones

La ejecucion del script para la pantalla de Exceptions se inicio con 53 request para ser completados teniendo en cuenta todos los test case del Test plan que fueron automatizados. La ejecución del script se realizo con la siguiente configuración (ver la ilustración 48):

Primera parte: Se configura todo lo relacionado con la conexión de base de datos, ip de Flow con que se testeara

Segunda parte: Se configura la cantidad de usuario que ingresaran a la aplicación a realizar el test.

Tercera parte: Configuraremos unicamente si el test sera sincronizado, cuantantas veces el usuario realizara el test y el valor de quantityException.

Ilustración 48: Configuración para el script de excepciones

```
#Configuration of Flow (Primera Parte)
ipDb=10.0.0.122
passwordDB=control123!
nameDB=JRL386
userDB=sa
ipFlow=10.0.0.149
timeThread=10

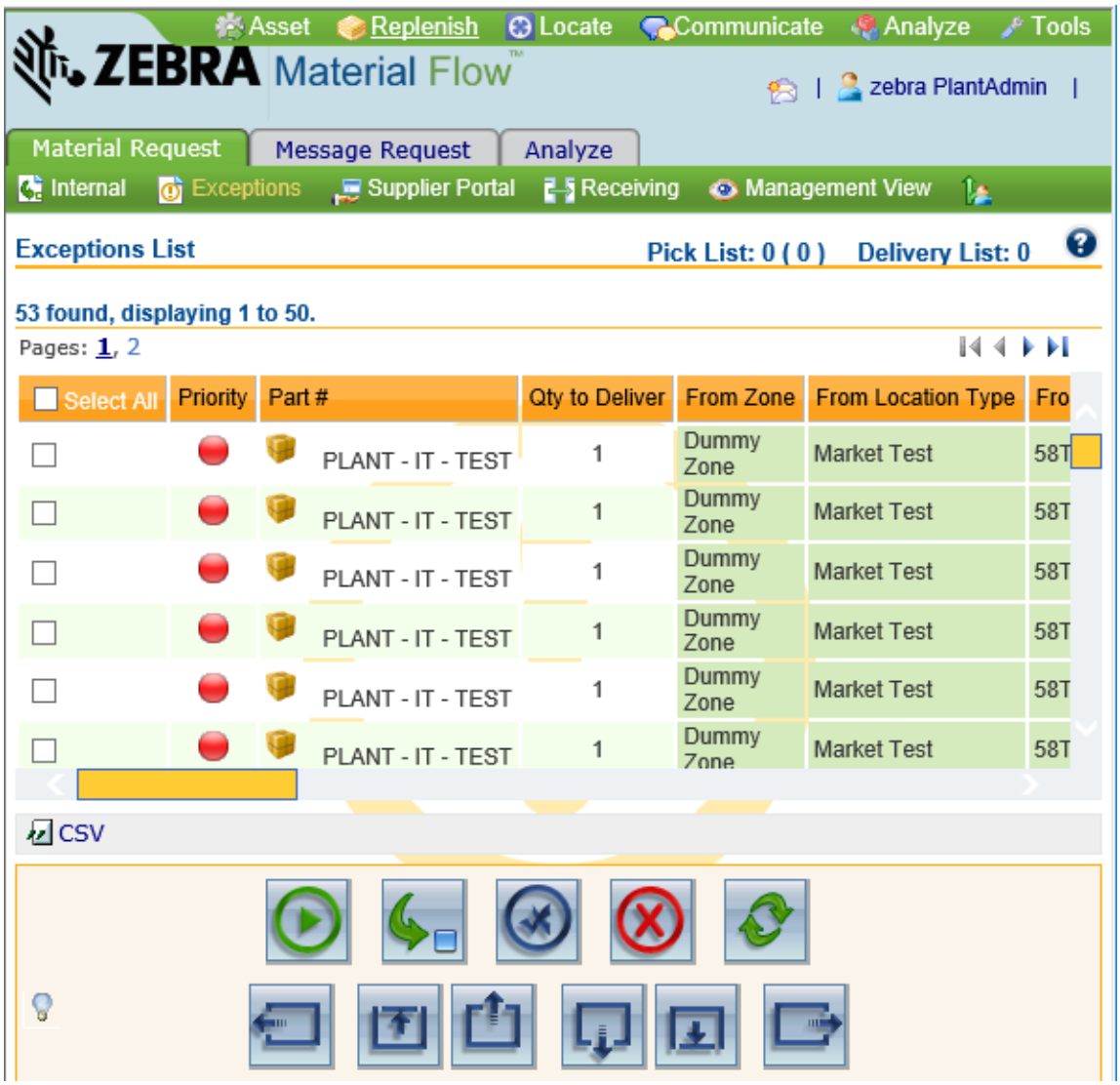
#Configure quantity users (Segunda Parte)
totalUser=1
rampUpPeriodo=0

#Configuration excute script for HandHeld and
Desktop (Tercera Parte)
synchronizeProcess=true
loopInfinite=false
numberLoops=1
quantityRequest=30
cancelDelivery=2
quantityException=3
```

Fuente: Elaboración propia

La ilustracion 49 muestra el estado inicial con el cual se iniciara el testing correspondiente al Pantalla.

Ilustración 49: Estado inicial de request en la Pantalla de Excepciones

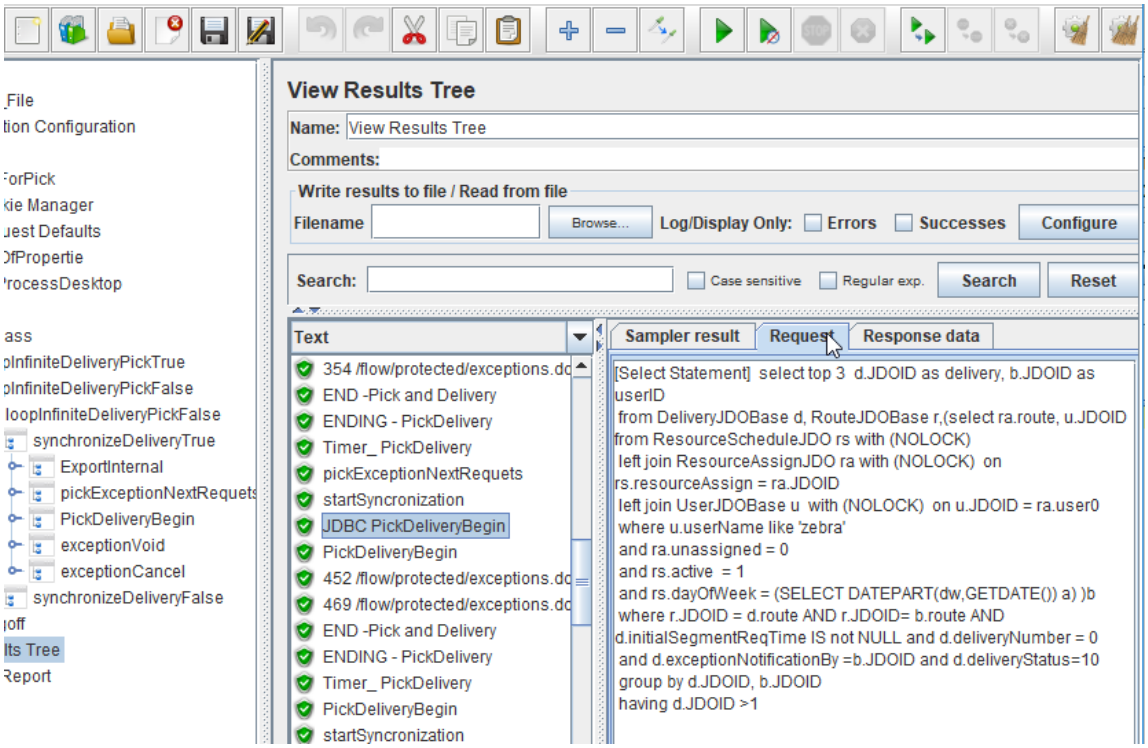


Fuente: Elaboración propia

Las ilustraciones 50 y 51 mostraran la ejecución del script para esta Pantalla y como se recuperan los datos después de ejecutarse una consulta en un test case determinado

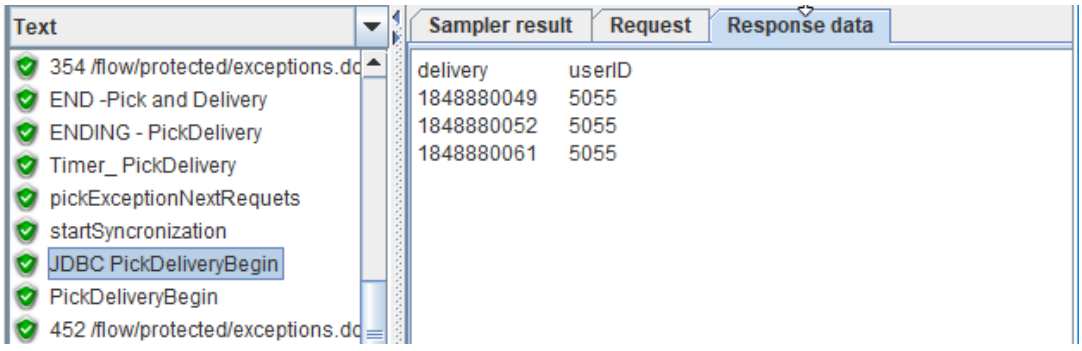
Pick / Delivery por begin delivey

Ilustración 50 :Pick/Delivery begin delivery - Consulta a la base de datos



Fuente: Elaboración propia

Ilustración 51: IDs de request para pick/delivery



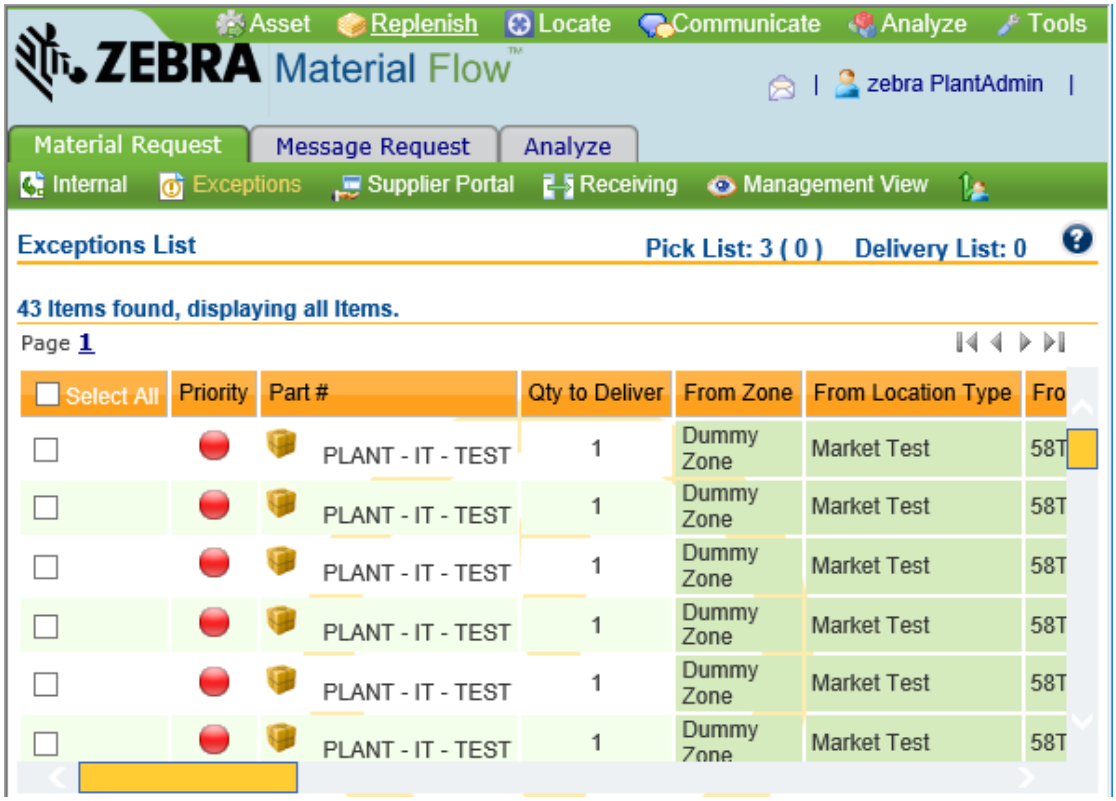
Fuente: Elaboración propia

Estado final de la lista de excepciones

El estado final en la lista se observa que se quedo con 43 request de 53 que se tenia inicialmente, la razon es (ver la ilustración 52):

- 1 request fue completado por pick/delivery por next request.
- 3 fueron completados con pick/delivery por begin delivery.
- 3 fueron completados por void.
- 3 se cancelaron razon por la cal se observa en “Pick List”.

Ilustración 52: Estado final de la lista de excepción



Fuente: Elaboración propia

Reporte después de la ejecución del script

El reporte que se creo fue para un mejor control de la ejecución del script, pues ocurren los casos que en el script nos muestra que todo se cumplio al 100% sin

errores pero al revisar la aplicación nos encontramos que no se cumplieron los test case y no se tiene una razón o en donde ocurrio el fallo. Los datos que tenermos es en que version se ejecuto el script, cual es el nombre del test case, que usuario la cumplio, las cantidades de antes y despues de que un test case se ejecute, el resultado se paso o fallo, la fecha y hora y cuanto tiempo tardo en cumplir con la ejecución del test case, el reporte podra verlo en la parte de anexos con el nombre de Reporte de Exception.

4.5.4. Cuarto Sprint - Script para la Pantalla de Management View

En el cuarto sprint se dará a conocer la cantidad de script para la Pantalla de Management View, después de realizar el análisis respectivo, en la tabla 26 se muestra una información general del cuarto sprint.

Tabla 26: Cuarto Sprint

Sprint	Historia	Fecha de inicio	Duración (días)
4	Script para el Pantalla de Management View en los casos de prueba	24/05/2018	16

Fuente: Elaboración propia

En la tabla 27 se especificara la lista de las tareas que son necesarias para concluir el tercer sprint, las tareas de la tercera iteración constará en el desarrollo de las siguientes tareas.

Tabla 27: Cuarto Sprint - Pantalla Management View

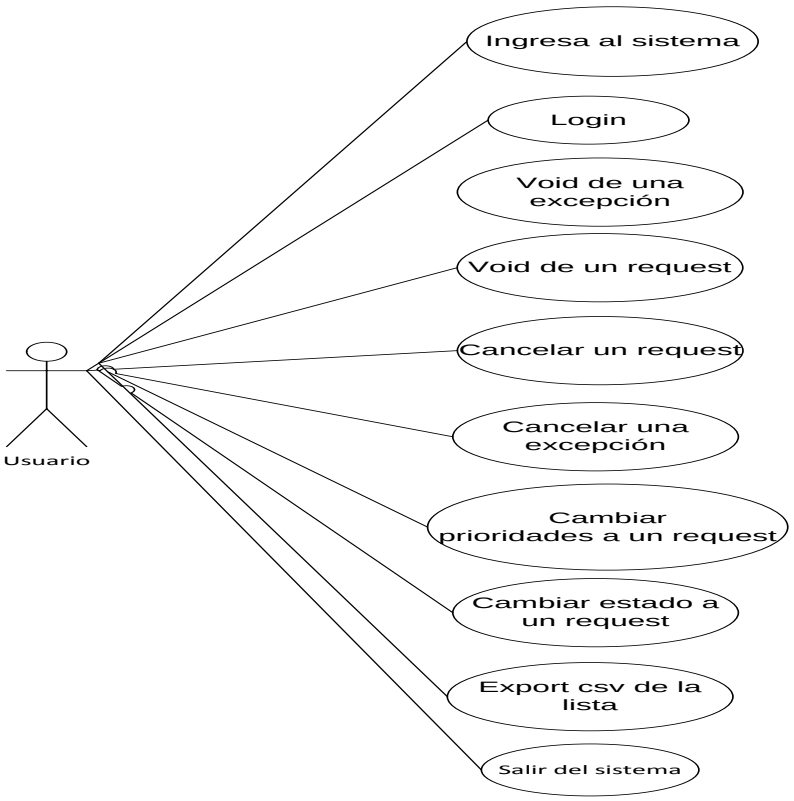
ID	Nombre	Tipo	Prioridad	Owen	Duración (días)	Estado
Mv-1	Exportar la lista CSV de management view	Desarrollo	**Alta**	M. Alavi	1 ½	Terminado
Mv-2	Vaciar la lista de pedidos	Desarrollo	**Alta**	M. Alavi	1 ½	Terminado
Mv-3	Vaciar pedidos con excepción	Desarrollo	**Alta**	M. Alavi	1 ½	Terminado
Mv-4	Cancelar un pedido	Desarrollo	**Medio**	M. Alavi	1 ½	Terminado

Mv-5	Cancelar un pedido con excepción	Desarrollo	**Medio**	M. Alavi	1 ½	Terminado
Mv-6	Cambiar las prioridades de un request (overdue, requested, need, need son, critical)	Desarrollo	**Alta**	M. Alavi	1 ½	Terminado
Mv-7	Cambiar los estados de un request (Exception, Hold, Void)	Desarrollo	**Alta**	M. Alavi	1 ½	Terminado

Fuente: Elaboración propia

4.5.4.1. Análisis de los casos de uso para los scripts

Diagrama 7: Casos de uso para la automatización de Management View



Fuente: Elaboración propia

Tabla 28: Descripción de los casos de uso

CU-3	Management View – Script
Descripción	En el Pantalla de Management View se desarrollaran un script para cada caso de uso y se dividiran aquellos que sean necesarios, los pasos que se repitaran en los script son los 2 primeros.
Secuencia normal	1. Ingresar a la apliación por medio del login. 2. Ir a Replenish/Material Request/Management View. 3. Void de un pedido 4. Void de un pedido con excepción. 5. Cancelar de un pedido que se encuentra en la lista delivery. 6. Cancelar de un pedido con excepción. 7. Cambiar la prioridad de un pedido. 8. Cambiar el estado de un pedido (void, hold, excepción). 9. Exportar la lista de management view en archivo csv.
Excepciones	
Prioridad	Alta
Roles	QA.
Datos	Request generados de manera manual o automatica.

Fuente: Elaboración propia

4.5.4.2. Diseño de los scripts para la Pantalla de Management View

Para la construcción de los scripts para la Pantalla de Management View se tomo en cuenta todos los querimientos especificados y el orden de los pasos son los siguientes:

Paso 1: Conexión a la base de datos para realizar consultas necesarias.

Paso 2: Configuracion de los usuarios.

Paso 3: Uso del componente *HTTP Request Defaults* para realizar la conexión con la IP donde se encuentra la aplicación Flow.

Paso 4: El paso de inicio de sesión por medio de login.

Paso 5: Validar que el usuario existe para completar el test, de no existir el usuario el test no se ejecutara.

Paso 7: El test pasara a la pantalla de Management View y buscara por el rango de tiempo que se especifico: allDays, today, yesterday, week, 1month, 3month, 6moths y year.

Paso 8: Despues de buscar los datos se ejecutara el test de:

- Void de un request
- Void de una excepción
- Cancelar un request que se encuentre en la lista delivery
- Cancelar un request con excepción
- Cambiar la prioridad de un request
- Cambiar el estado del request
- Exportar la lista en archivo csv

Paso 9: Para completar alguno de los anteriores procesos será necesario realizar una consulta en específico para cada caso del Pantalla de excepciones, si la consulta no devuele ningun dato el test no llevara acabo y el usuario cerrara sesión.

Paso 10: Según sea la configuración del archivo respectivo del script y su respectivo proceso podra ser ejecutado de forma infinita o definida.

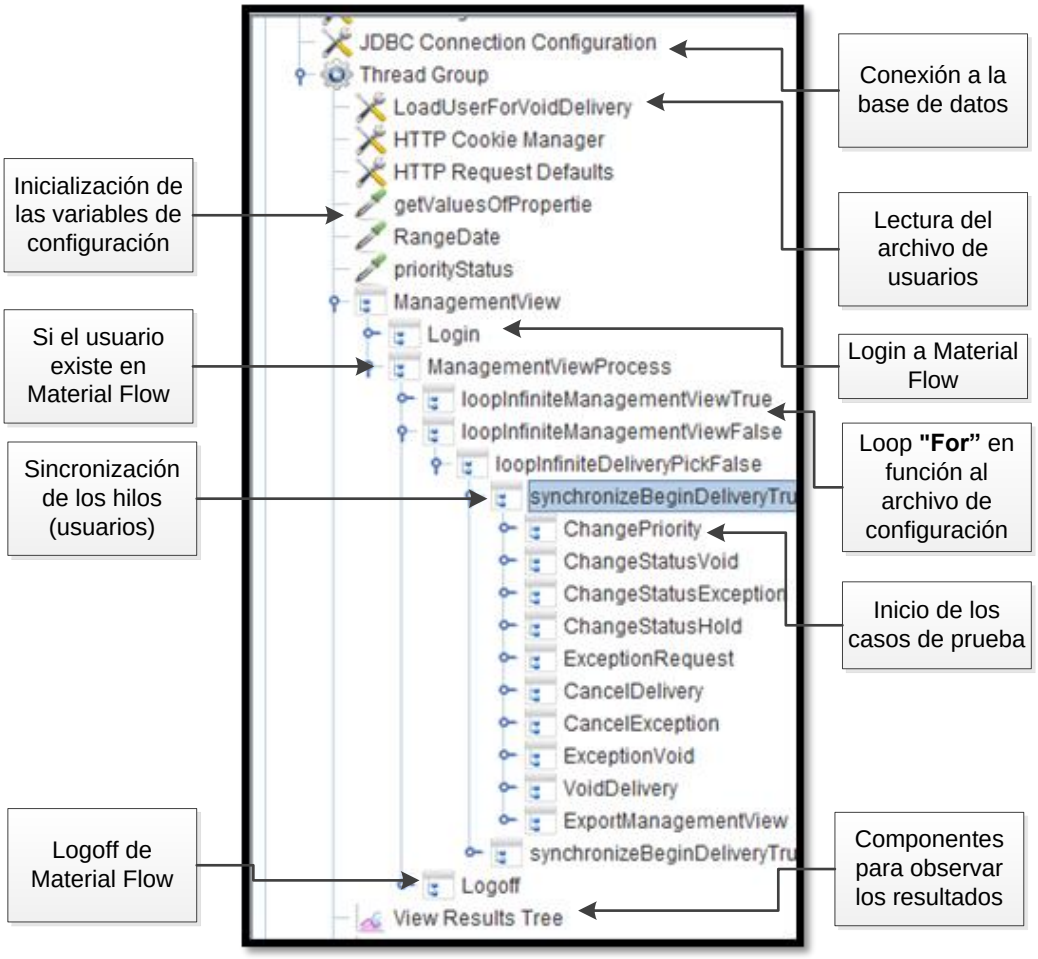
Paso 11: Cerrar sesión despues de terminar con la ejecución del proceso.

Componentes de JMeter usuados en los scripts

Nombre	Función
Group thread	Componente en el que se configura la cantidad de hilos que emularán a los usuarios.
Property fiel reader	Para poder leer archivos de exención properties, en las cueles se declara variables o propiedades para la configuración.
CSV data confi	Ayuda con la lectura de archivos de extención csv, en este caso se usara para declarar a los usuarios para completar el test.
BeanShell Sampler	Componente para realizar programación en java.
Recordin Controller	Componente en el cual se guarda los pasos de la grabación del test.
Transaction Controller	Para la agrupación de los pasos grabados.
If Controller	Para poder delimitar que pasos ejecutara el usuario.
Regex Controller	Nos ayudará obtener datos del html.
Loop Controller	Con el cual se realizará la emulación de un for para ejecutar una cantidad de veces un paso.
Foreach Controller	Se usara con relación a una consulta a la base de datos.
JDBC Request	Compnente con el que se realizarán las consultas a la base datos
JDBC Connection Configuration	Componente con el que se realizará la conexión con una respectiva base de datos, por medio de una ip, usuario de DB y contraseña.

En la ilustración 53 (Diseño de script para la Pantalla de Management View) se muestra la forma en la que se utilizo los componentes, para la construcción del script de Management View a sus lados se específica como se uso.

Ilustración 53: Diseño de script para la Pantalla de Management View



Fuente: Elaboración propia

Antes de la ejecución del script es necesario aclarar la configuración necesaria en el property file la primera parte de la configuración de la misma que en los dos anteriores casos, sin embargo para la configuración específica de Management View es la siguiente (ver la ilustración 54):

- Se debe dar un valor para cambiar las prioridades a los request.
- Especificar el rango de fechas por el cual se buscara requests para luego exportarlos.
- A cuantos request se cambiará el estado ya sea void, hold o excepción.

- Cuantos requests de la lista delivery se cancelaran.
- Cuantos request de golpe se cambiara a void.
- NOTA: la cantidad de excepciones debe ser mayor al número de cancelDeliveryQty.

Ilustración 54: Configuración Management View

```
#Configuration for management view
# requested, need, need soon, critical, overdue
# alldays, today, yesterday, week, month, 3month, 6month,
year
partRouteNumberRequest=18022
changePriority=critical
rangeDateSearch=today
quantityChangeStatus=3
requestExceptionQty=10
cancelDeliveryQty=3
DeliveryVoidQty=3
```

Fuente: Elaboración propia

Obtener valores para el rango y cambio de prioridades especificado en el archivo de configuración

En ilustración 55 se muestra el código para los rango de tiempo ya sea por el día actual, día anterior, hace una semana entre otros, para ello obtenemos la fecha actual en el formato necesario para la búsqueda en el script y otra para realizar las consultas a las base de datos.

Ilustración 55: Lógica para los rangos de fechas

```
import java.util.Date;
import java.text.SimpleDateFormat;

String rangeDateSearch1=vars.get("rangeDateSearch");
String rangeSearch=rangeDateSearch1.toLowerCase().trim();
log.info("range searchs: "+rangeSearch);
if(rangeSearch.equals("alldays")){
	vars.put("rangeForSearchDelivery","0");
	Date date = new Date();
	date.setDate(date.getDate());
	SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("MM/dd/yyyy");
	String formattedDate = simpleDateFormat.format(date);
```

```
	vars.put("startDateInput",formattedDate);
	vars.put("endDateInput",formattedDate);
	SimpleDateFormat simpleDateFormatDB = new
SimpleDateFormat("yyyy/MM/dd");
	String formattedDateDB = simpleDateFormatDB.format(date);
	vars.put("startDateInputDB",formattedDateDB);
	vars.put("endDateInputDB",formattedDateDB);
	}
```

Fuente: Elaboración propia (el resto del código se encuentra en la parte de anexos con el nombre "Range Date")

Para el cambio de prioridades es neceario asignarle un número, a travéz de este rango el sistema Flow cambiará a un request su prioridad (ver la ilustración 56).

Ilustración 56: Lógica para el cambio de prioridades

```
String changePriority1=vars.get("changePriority");
String statusPriority=changePriority1.toLowerCase().trim();
log.info("Priority status: "+statusPriority);
if(statusPriority.equals("requested")){
	vars.put("priorityStatus","1");
	log.info("Change priority to: "+vars.get("changePriority")+" range
"+vars.get("priorityStatus"));
		}
if(statusPriority.equals("need")){
	vars.put("priorityStatus","2");
	log.info("Change priority to: "+vars.get("changePriority")+" range
"+vars.get("priorityStatus"));
	}
if(statusPriority.equals("need soon")){
	vars.put("priorityStatus","3");
	log.info("Change priority to: "+vars.get("changePriority")+" range
"+vars.get("priorityStatus"));
	}
if(statusPriority.equals("critical")){
	vars.put("priorityStatus","4");
	log.info("Change priority to: "+vars.get("changePriority")+" range
"+vars.get("priorityStatus"));
	}
if(statusPriority.equals("overdue")){
	vars.put("priorityStatus","5");
	log.info("Change priority to: "+vars.get("changePriority")+" range
"+vars.get("priorityStatus"));  }
```

Fuente: Elaboración propia

Consulta a la base datos para validar los test case busqueda

Con la siguiente a la base de datos obtendremos todos los datos necesarios para la validación de los test case de busqueda, estos request son aquellos que son activos, no existe la necesidad de que este asignado al usuario con el que se realiza el login, sin embargo es necesario tener el valor de part route ya que este valor es muy importante para tener un control sobre el número de datos que se debe mostar en la lista de management view, la cosulta se muestran en las ilustraciones 57 y 58:

Ilustración 57: Consulta para los test case de búsqueda

```
select top 1 d.JDOID, d.status,
r.requestCallType,p.cleanedPartNumber,p.prefix, p.base,
p.suffix,p.partStatus, ps.name, ps.partStage, d.route,
rt.routeType,rt.name0 as routeName, rty.name0 as
routeTypeName,d.fromLocation, l.name0 as locName,d.toLocation, l.type0,
lt.name0 locTypeName
from DeliveryJDOBase d, RequestJDOBase r, PartRouteJDOBase pr, PartJDOBase
p, RouteJDOBase rt, LocationJDOBase l, LocationTypeJDOBase lt,
PartStatusJDOBase ps, RouteTypeJDOBase rty
where r.JDOID=d.request and r.active=1
and d.external1=0 and d.deliveryComplete=0 and d.deliveryStatus != 11
and pr.partRouteNumber=${__P(partRouteNumberRequest)}  and
r.partRoute=pr.JDOID and p.JDOID=pr.part and p.partStatus=ps.JDOID
and d.route=rt.JDOID and d.fromLocation=l.JDOID and l.type0=lt.JDOID and
rt.routeType=rty.JDOID
and d.initialSegmentReqTime between '${searchDateDB} 00:00:00.999' and
'${searchDateDB}  23:59:59.999'
```

Fuente: Elaboración propia

Ilustración 58: Control de la cantidad de request

```
select COUNT(d.JDOID) as countRequest
from DeliveryJDOBase d, RequestJDOBase r, PartRouteJDOBase pr
where r.JDOID=d.request
and d.external1=0
and d.deliveryComplete=0
and d.deliveryStatus != 11
and pr.partRouteNumber=${__P(partRouteNumberRequest)}
and r.partRoute=pr.JDOID
and d.initialSegmentReqTime between  '${searchDateDB}  00:00:00.999' and
'${searchDateDB}  23:59:59.999'
```

Fuente: Elaboración propia

Despues de obtener los valores con la consulta según el rango especificado en el archivo de propiedades, todos ellos se validaran y también se las guardaran en variables que puedan ser referenciadas en todo el script y los test case necesarios, la variables se guardan como se muestra en la ilustración 59 (Obtener valores de las consultas):

Ilustración 59: Obtener valores de las consultas

```
ArrayList listSearchID=vars.getObject("searchIds");

if(listSearchID==null){
	log.info("request list is null");
	}
else{
	if(listSearchID.size()>0){
		vars.put("requestIDs","true");

	vars.put("requestId",vars.getObject("searchIds").get(0).get("JDOID").
toString());

	vars.put("cleanedPartNumber1",vars.getObject("searchIds").get(0).get(
"cleanedPartNumber").toString());
		log.info("Cleand part number:
"+vars.get("cleanedPartNumber1"));

	vars.put("prefix1",vars.getObject("searchIds").get(0).get("prefix").t
oString());
		log.info("prefix: "+vars.get("prefix1"));

	vars.put("base1",vars.getObject("searchIds").get(0).get("base").toStr
ing());
		log.info("base: "+vars.get("base1"));

	vars.put("suffix1",vars.getObject("searchIds").get(0).get("suffix").t
oString());
		log.info("suffix: "+vars.get("suffix1"));

	vars.put("partStatus1",vars.getObject("searchIds").get(0).get("partSt
atus").toString());
		log.info("partStatus: "+vars.get("partStatus1"));

	vars.put("name1",vars.getObject("searchIds").get(0).get("name").toStr
ing());
		log.info("name: "+vars.get("name1"));
```

```
      vars.put("partStage1",vars.getObject("searchIds").get(0).get("partSta
ge").toString());
             log.info("partStage: "+vars.get("partStage1"));

      vars.put("route1",vars.getObject("searchIds").get(0).get("route").toS
tring());

      vars.put("routeName1",vars.getObject("searchIds").get(0).get("routeNa
me").toString());
             log.info("route: "+vars.get("routeName1")+"
ID:"+vars.get("route1"));

      vars.put("routeType1",vars.getObject("searchIds").get(0).get("routeTy
pe").toString());

      vars.put("routeTypeName1",vars.getObject("searchIds").get(0).get("rou
teTypeName").toString());
             log.info("routeType: "+vars.get("routeTypeName1")+ "ID:
"+vars.get("routeType1"));

      vars.put("fromLocation1",vars.getObject("searchIds").get(0).get("from
Location").toString());

      vars.put("locName1",vars.getObject("searchIds").get(0).get("locName")
.toString());
             log.info("fromLocation: "+vars.get("locName1")+" Id:
"+vars.get("fromLocation1"));

      vars.put("toLocation1",vars.getObject("searchIds").get(0).get("toLoca
tion").toString());
             log.info("toLocation: "+vars.get("toLocation1"));

      vars.put("type01",vars.getObject("searchIds").get(0).get("type0").toS
tring());

      vars.put("locTypeName1",vars.getObject("searchIds").get(0).get("locTy
peName").toString());
             log.info("loccation Type Name: "+vars.get("locTypeName1")+"
Id: "+vars.get("type01"));

      vars.put("requestCallType1",vars.getObject("searchIds").get(0).get("r
equestCallType").toString());
             log.info("request call type: "+vars.get("requestCallType1"));

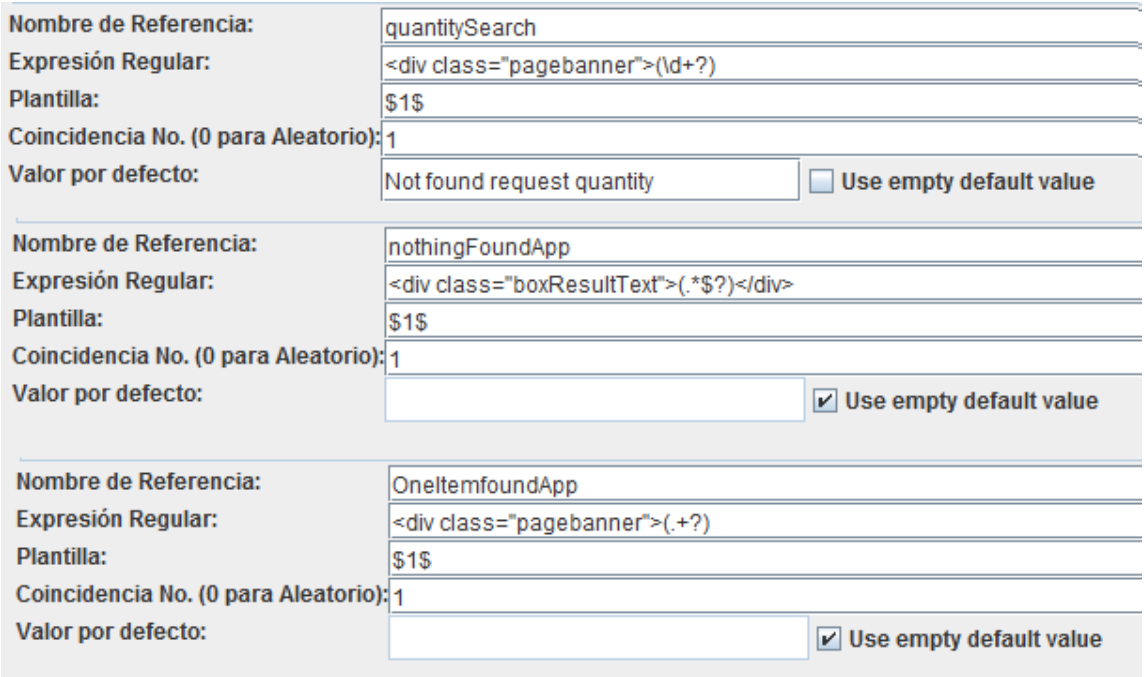

      vars.put("status1",vars.getObject("searchIds").get(0).get("status").t
oString());
             log.info("status: "+vars.get("status1"));
```

Fuente: Elaboración propia (el resto del código está en la parte de Anexos con el nombre Obetner valores de las consultas, searchRequestByForeachField*)*

Análisis de la página por medio del componente regex

Para el control y validación del test case es necesario saber el número inicial al momento en el que se ejecutará el test case, sin embargo para los test case de búsqueda este número se comparara con el número que devuelva la consulta para contar los registros, para el análisis de la página se usaron los siguientes regex como se observa en la Ilustración 60:

Ilustración 60: Regex para el análisis de la página



Fuente: Elaboración propia

Test case para archived

Para este tipo de test es necesario realizar 3 tipos de consultas a la base datos y después de obtener y guardarlos en variable para referenciar es necesario realiza un análisis de la página, con los componentes de regex, como se muestra en las ilustraciones 61, 62 y 63.

Ilustración 61: Consultas para datos archived

```
select JDOID, name0
from RouteJDOBase
with (nolock)
where archived =1
```

```
select JDOID, name0
from
LocationTypeJDOBase
with (nolock)
  where archived =1
```

```
select JDOID, name0
from RouteTypeJDOBase
with (nolock)
where archived =1
```

Fuente: Elaboración propia

Ilustración 62: Guardar los datos obtenidos

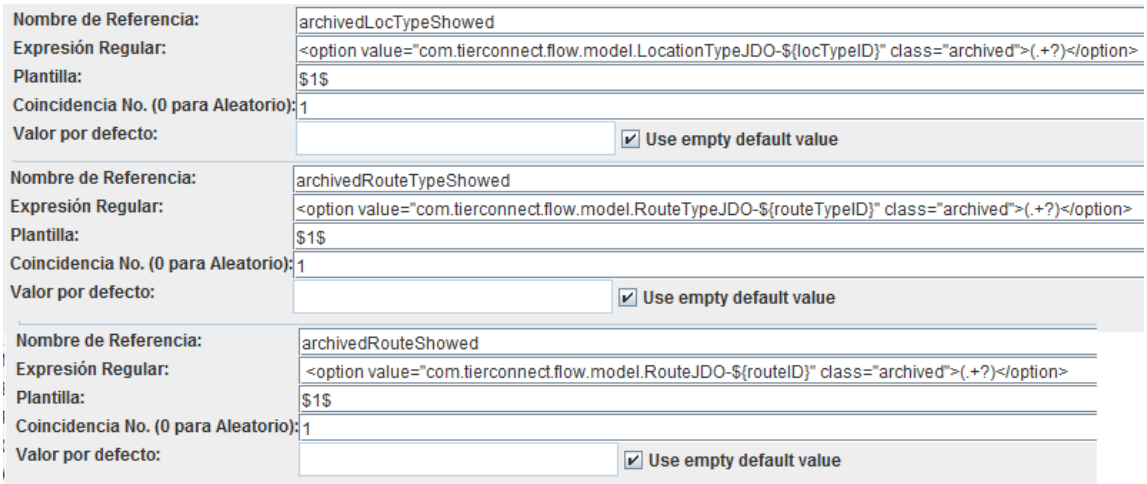
```
ArrayList archivedRoute=vars.getObject("archivedRoute");
ArrayList archivedLocType=vars.getObject("archivedLocationType");
ArrayList archivedRouteType=vars.getObject("archivedRouteType");

if(archivedRoute==0 ){
	log.info("No found archived route");
}
else{
	if(archivedRoute.size()>0){
		vars.put("existeArchivedRoute","true");
		vars.put("routeID",
vars.getObject("archivedRoute").get(0).get("JDOID").toString());
		vars.put("routeName",
vars.getObject("archivedRoute").get(0).get("name0").toString());
		log.info("Route name: "+vars.get("routeID")+" routeID:
"+vars.get("routeName"));
	}
	else{
		vars.put("existeArchivedRoute","false");
		log.info("No found archived route");
	}
}
if(archivedLocType==0){
	log.info("No found archived location type");
}else{
	if(archivedLocType.size()>0){
		vars.put("existeArchivedLocType","true");
		vars.put("locTypeID",
vars.getObject("archivedLocationType").get(0).get("JDOID").toString());
		vars.put("locTypeName",
vars.getObject("archivedLocationType").get(0).get("name0").toString());
```

```
            log.info("Location type name: "+vars.get("routeID")+" routeID:
"+vars.get("routeName"));
      }
      else{
            vars.put("existeArchivedLocType","false");
            log.info("No found archived route");
      }
}
if(archivedRouteType==0){
      log.info("No found archived route");
}
else{
      if(archivedRouteType.size()>0){
            vars.put("existeArchivedRouteType","true");
            vars.put("routeTypeID",
vars.getObject("archivedRouteType").get(0).get("JDOID").toString());
            vars.put("routeTypeName",
vars.getObject("archivedRouteType").get(0).get("name0").toString());
            log.info("Route type name: "+vars.get("routeTypeID")+" route
type ID: "+vars.get("routeTypeName"));
      }     else{
            vars.put("existeArchivedRouteType","false");
            log.info("No found archived route");
      }
}
```

Fuente: Elaboración propia

Ilustración 63: Análisis de la página con los datos de la consulta



Fuente: Elaboración propia

Test case de cambios de prioridad, estados y otros procesos

La consulta que se debe realizar antes de la ejecución de cada uno de los test case para el procesamiento de request consta de las condiciones de que debe ser activo, su estado de prioridad debe ser distinta a lo que especifico en el archivo de propiedades, deben ser request sin completar y deben estar dentro del rango especificado (ver la ilustración 64).

Ilustración 64: Consulta para el Id's de los requets

```
select top ${__P(DeliveryVoidQty)} d.JDOID
from DeliveryJDOBase d, RequestJDOBase r
where r.JDOID=d.request
and d.external1=0
and d.deliveryComplete=0
and d.deliveryStatus != 11
and r.active=1
and d.status !=${priorityStatus}
and d.initialSegmentReqTime between  '${startDateInputDB}  00:00:00.999'
and '${endDateInputDB}  23:59:59.999'
```

Fuente: Elaboración propia

Una vez obtenido los Id's de los request estos se procesaran de las siguiente formas ya sea para procesarlos uno por uno o procesaralos por cantidades por lo tanto tendran un formato extra, en este caso separados por comas (ver la ilustración 65 y 66).

Ilustración 65: Procesamiento de datos

```
ArrayList listRequestID=vars.getObject("requestID");
if(listRequestID==null){
       log.info("request list is null");
       }else{
       if(listRequestID.size()>0){
              vars.put("requestIDs","true");
       vars.put("requestId",vars.getObject("requestID").get(0).get("JDOID").
toString());
              log.info("Request change priority: "+vars.get("requestId"));
              }else{
              vars.put("requestIDs","false");
              log.info("Does exists request");
              }}
```

Fuente: Elaboración propia

Ilustración 66: Procesamiento de datos, separado por comas

```
ArrayList listDeliveryId=vars.getObject("deliveryIDVoids");
String deliveryIdForVoid="";
int sizeDeliveryID=listDeliveryId.size();
int total=0;

for(Object deliveryId : listDeliveryId){
      if(total+1==sizeDeliveryID){
            deliveryIdForVoid=deliveryIdForVoid+deliveryId.get("JDOID");
            vars.put("deliveryIdForVoid1",deliveryIdForVoid.toString());
         log.info("user name: "+vars.get("user1")+" "+"Delivery id:
"+vars.get("deliveryIdForVoid1"));
            }
      else{

      deliveryIdForVoid=deliveryIdForVoid+deliveryId.get("JDOID")+",";
            }
      total=total+1;
}
if(deliveryIdForVoid.length()>0){
      vars.put("hasDeliveryID","1");
      }
else{
      vars.put("hasDeliveryID","0");
      }
```

Fuente: Elaboración propia

Generación del Reporte

Después de una validación de los test case con los respectivos análisis de la página, se generará el reporte en formato csv, con las mismas reglas que los casos anteriores (ver la ilustración 67).

Ilustración 67: Generar el reporte

```
import org.apache.jmeter.services.FileServer;
import java.text.SimpleDateFormat;

//Date for CSV
SimpleDateFormat formatDate = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
java.util.Date date = new Date();
String format=formatDate.format(date);
//Date for name CSV
SimpleDateFormat formatDateCSV = new SimpleDateFormat("yyyy-MM-dd");
java.util.Date datecsv = new Date();
```

```
String formatcsv=formatDateCSV.format(datecsv);
versionMessage1=vars.get("versionMessage");
userTest=vars.get("user1");
quantityStart1=vars.get("quantityStart");
quantityFinsh1=vars.get("quantityFinsh");
objectSearch="";
testResult1=vars.get("testResult");
secondsexportManagementView1=vars.get("secondsexportManagementView");

file = new FileOutputStream (FileServer.getFileServer().getBaseDir() + "\\
ManagemntView"+formatcsv+".csv", true);
print = new PrintStream (file);

print.println (versionMessage1+",Test case: Void a delivery " +",User:
"+userTest+", Object search test: "+objectSearch+", Quantity at the start
of the test:"+quantityStart1+",Quantity at the finish of the test:
"+quantityFinsh1+","+testResult1+",Date: "+format+",Time:
"+secondsexportManagementView1);

// Cerrar archivo (s)
print.close ();
file.close ();
```

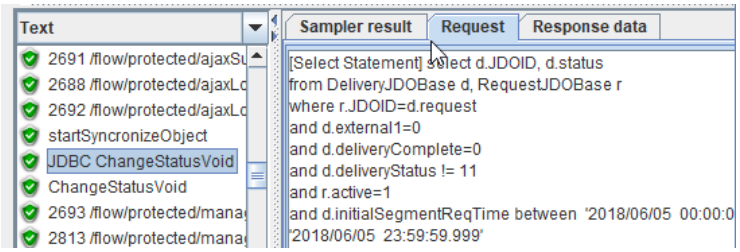
Fuente: Elaboración propia

4.5.4.3. Implementación de los scripts automatizados para la Pantalla de Management View

En la pantalla de Management View se puede observar todos los request pendientes sin importar que usuario este logueado en la aplicación, en donde es posible completarlos, cambiar el estado del request, su prioridad, de igual manera observamos cual es el origen del request, que usuario la creo.

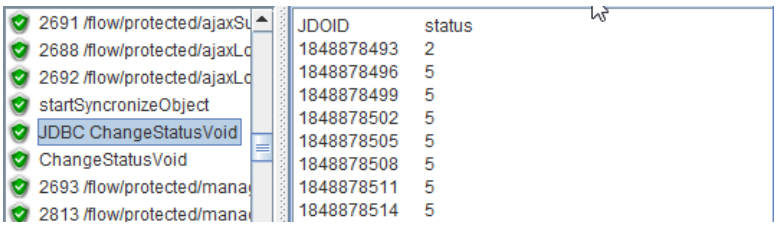
Cambiar el estado de un request a void

Ilustración 68: Consulta a la base datos de los request para cambiar su estado a void



Fuente: Elaboración propia

Ilustración 69: Id's de request para cambiar su estado

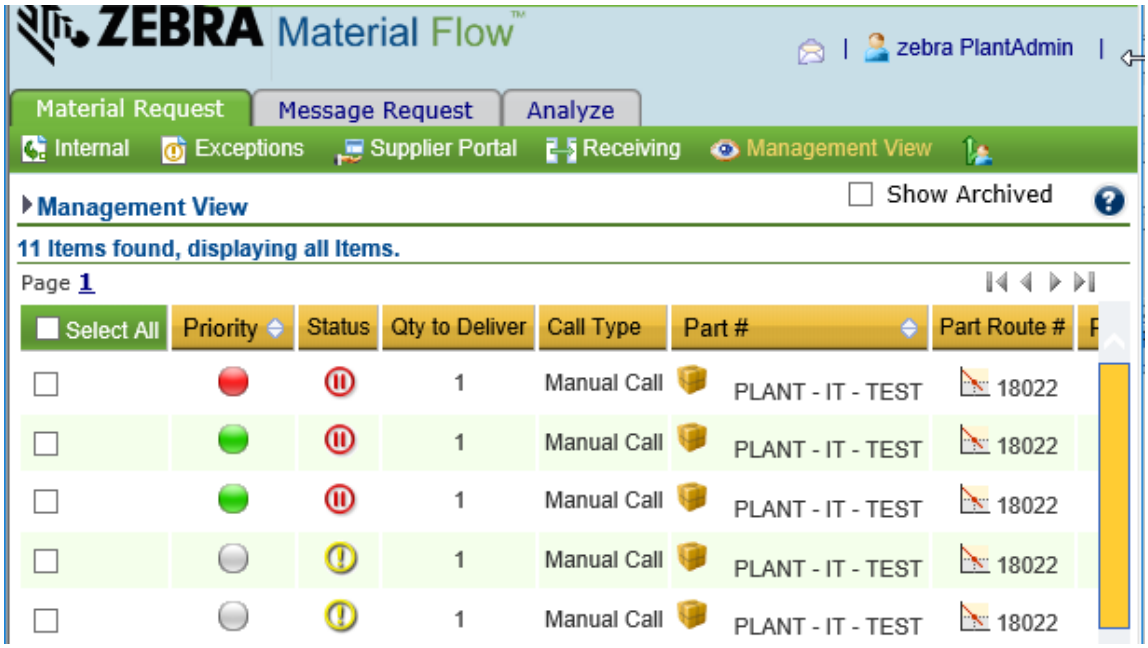


Fuente: Elaboración propia

Estado final de la lista de Management View

Como se puede observar los pedidos sufrieron modificaciones según la ejecución del script y los test case, los request que se les cambio su estado a hold son aquellos que tienen el icono de pause en rojo, los que se cambio a excepción tienen el icono de un signo de admiración de color amarillo, el cambio de prioridad se observa en los colores de la columna de "Priority", el reporte de la ejecución del script se lo podra ver en la parte de anexos (ver la ilustración 70).

Ilustración 70: Estado final de Management View



Fuente: Elaboración propia

4.5.5. Quinto Sprint - Script para la Pantalla de Request History

En el quinto sprint se dará a conocer la cantidad de script para la Pantalla de Request History, después de realizar el análisis respectivo, en la tabla 29 se muestra una información general del quinto sprint.

Tabla 29: Quinto Sprint

Sprint	Historia	Fecha de inicio	Duración (días)
5	Script para el Pantalla de Request History en los casos de prueba	11/06/2018	5

Fuente: Elaboración propia

En la tabla 30 se especificara la lista de las tareas que son necesarias para concluir el tercer sprint, las tareas de la tercera iteración constara en el desarrollo de las siguientes tareas.

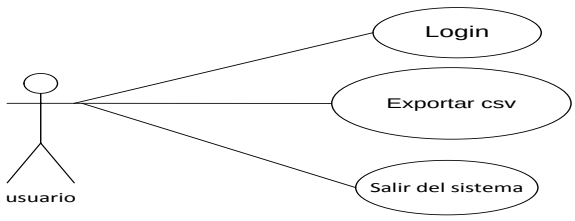
Tabla 30: Quinto Sprint - Pantalla Request History

ID	Nombre	Tipo	Prioridad	Owen	Duración (días)	Estado
Rh-1	Buscar un request por cada campo	Desarrollo	**Alta**	M. Alavi	2	Terminado
Rh-3	Exportar la lista CSV de Request history	Desarrollo	**Alta**	M. Alavi	2	Terminado

Fuente: Elaboración propia

4.5.5.1. Análisis de los casos de uso para los scripts

Diagrama 8: Casos de uso en Request History



Fuente: Elaboración propia

Tabla 31: Descripciones de los test case para automatizar

CU-3	Request History – Script
Descripción	Para el Pantalla de request history se realizar 2 script ya que son los adecuados para ser automatizados.
Secuencia normal	1. Ingreso a la apliacion por el login 2. Ir al Pantalla de replenishment, Request History. 3. Buscar los request por todos los campos. 4. Exportar la lista en archivo csv.
Excepciones	
Prioridad	Alta
Roles	QA.
Datos	Request generados de manera manual o automatica, request completados, excepciones y void.

Fuente: Elaboración propia

4.5.5.2. Diseño de los scripts para el Pantalla de Request History

Para la construcción de los scripts para el Pantalla de Request History se tomo en cuenta todos los requerimientos especificados y el orden de los pasos son los siguientes:

Paso 1: Conexión a la base de datos para realizar consultas necesarias.
Paso 2: Configuración de los usuarios.
Paso 3: Uso del componente *HTTP Request Defaults* para realizar la conexión con la IP donde se encuentra la aplicación Flow.
Paso 4: El paso de inicio de sesión por medio de login.
Paso 5: Validar que el usuario existe para completar el test, de no existir el usuario el test no se ejecutara.
Paso 6: El test pasara a la pantalla de Management View y buscará por el rango de tiempo que se especifico:

- allDays, today, yesterday, week, 1month, 3month, 6moths y year.

- Prioridades y estados.
- Part stage o nombre de la parte

Paso 7: Despues de buscar los datos se ejecutará el test de:

- Por cada busqueda se exportara un archivo csv comprimido en un zip.

Paso 8: Según sea la configuración del archivo respectivo del script y su respectivo proceso podra ser ejecutado de forma infinita o definida.

Paso 9: Cada busqueda sera aproximadamente cada 2 minutos.

Paso 10: Cerrar sesión despues de terminar con la ejecución del proceso.

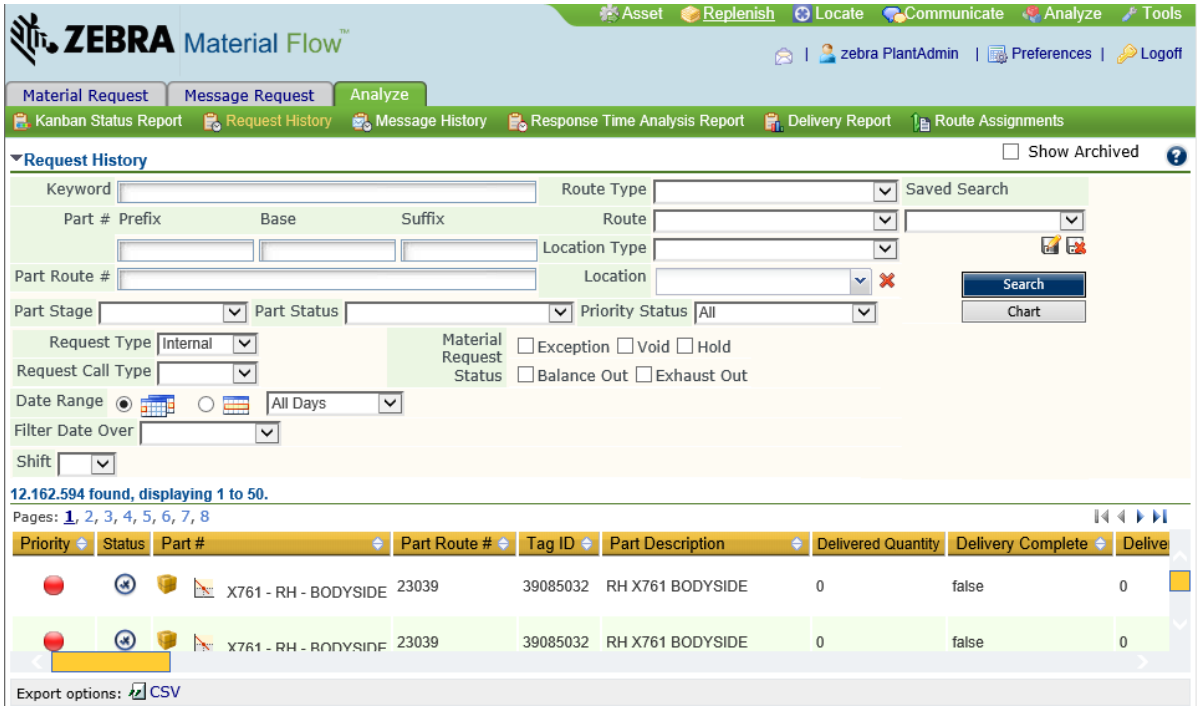
Componentes de JMeter usuados en los scripts

Nombre	Función
Group thread	Componente en el que se configura la cantidad de hilos que emularan a los usuarios.
Property fiel reader	Para poder leer archivos de exención properties, en las cuales se declara variables o propiedades para la configuración.
CSV data confi	Ayuda con la lectura de archivos de extención csv, en este caso se usara para declarar a los usuarios para completar el test.
BeanShell Sampler	Componente para realizar programación en java.
Recordin Controller	Componente en el cual se guarda los pasos de la grabación del test.
Transaction Controller	Para la agrupación de los pasos grabados.
If Controller	Para poder delimitar que pasos ejecutara el usuario.
Regex Controller	Nos ayudara obtener datos del html.
Loop Controller	Con el cual se realizara la emulación de un for para ejecutar una cantidad de veces un paso.
Foreach Controller	Se usara con relación a una consulta a la base de datos.
JDBC Request	Componente con el que se realizaran las consultas a la base datos
JDBC Connection Configuration	Componente con el que se realizará la conexión con una respectiva base de datos, por medio de una ip, usuario de DB y contraseña.

4.5.5.3. Implementación de los scripts automatizados para la Pantalla de Request History

En la ilustración 71 muestra la pantalla de request history después de una búsqueda de todos los días hasta la fecha actual, el script está diseñado para la exportación de los datos luego de una búsqueda por el campo Date Range.

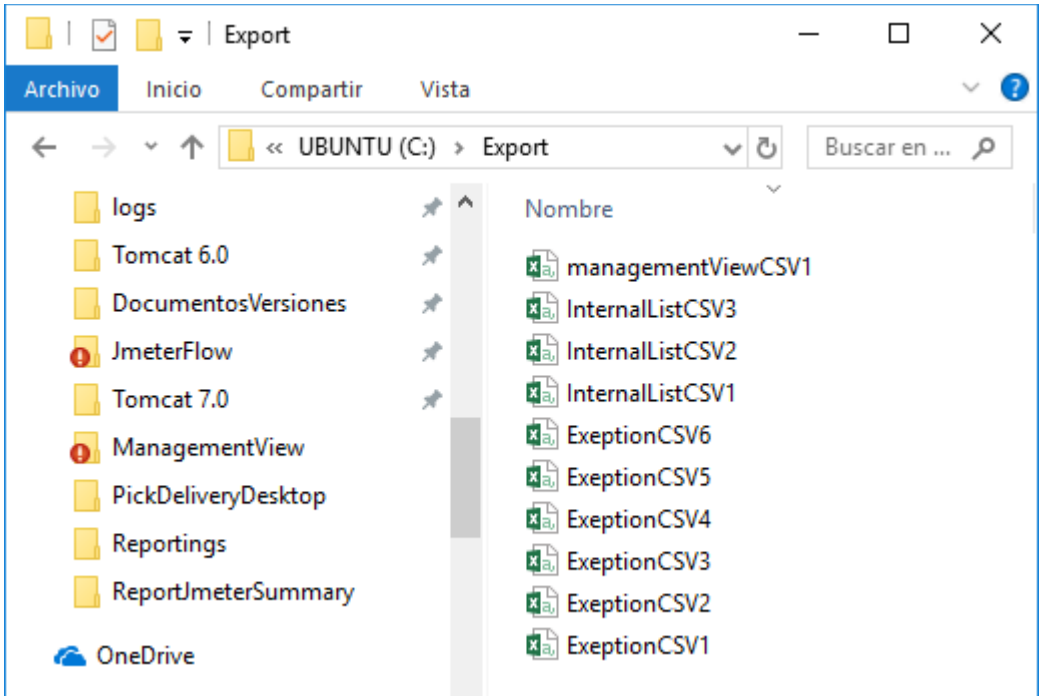
Ilustración 71: Request History - Búsqueda por todos los días



Fuente: Elaboración propia

Exportación de los archivos csv de las 4 pantallas

Ilustración 72: Archivos csv exportados de las pantallas



Fuente: Elaboración propia

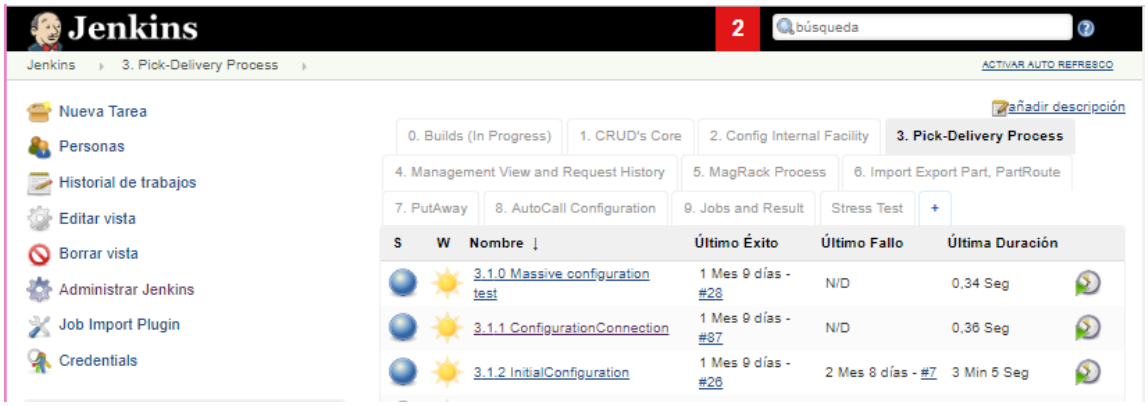
4.6. Integración de los script de JMeter con Jenkins

Después de terminar con la elaboración de los script para las pantallas management view, request history, internal y exception, es decir que los mismos ya pueden ser usados para la ejecución de testing de los casos de prueba y después de ellos los scripts devuelven un reporte mostrando los resultados de su ejecución. Sin embargo la configuración de los mismos es compleja para el usuario, para las ejecuciones deben ser manuales, por lo tanto como solución se integrara con Jenkins para tener una interfaz de configuración y las ejecuciones de testing sean periódicas, con cada ejecución fallida se enviará un correo electrónico a los integrantes del área de control de calidad, las características de la herramienta son las siguientes:

- Versión 2.73.2
- El componente para la ejecución del script fue el comando de línea CMD
- Ejecución parametrizada para las variables del script
- Exportación de parámetros en un archivo de propiedades (propertie file)
- Enviará email cuando la ejecución falle

En la ilustración 73 se muestra esta integración, en la herramienta es posible agrupar por pestañas todos los proyectos o tareas (como en Jenkins se los llama) que se ejecutaran de manera manual o periódica si se lo configura, el nombre, cuando fue la última ejecución exitosa, cuando fue la última fallida, cuanto duro la última ejecución y por último el botón para ejecutar el proyecto.

Ilustración 73: Panel de control de Jenkins

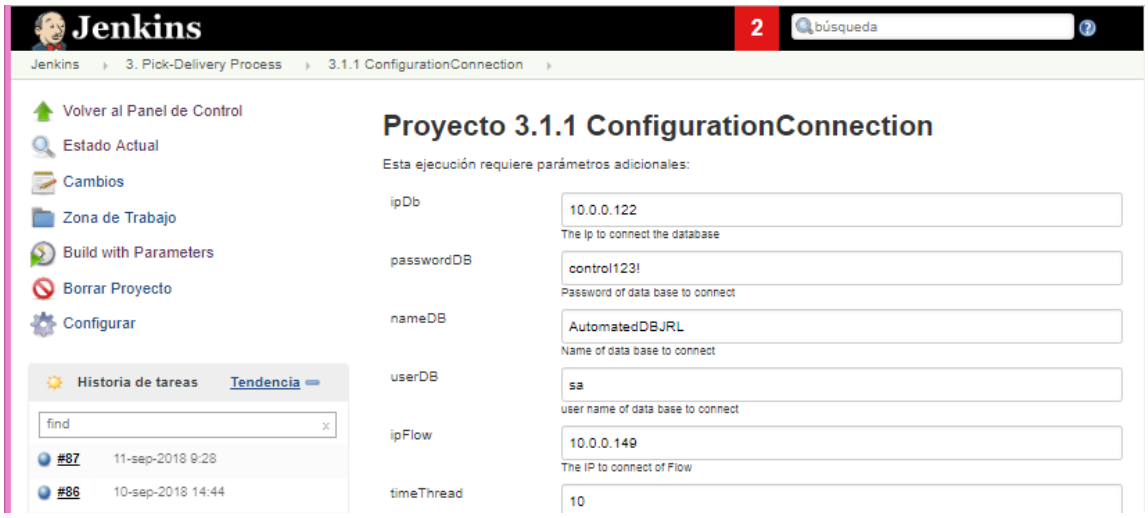


Fuente: Elaboración propia

Antes de la ejecución de los siguientes scripts para las pantallas es necesario realizar configuración en el archivo de propiedad como se mostró en las anteriores ilustraciones, ahora la configuración es más visual, lo que ayuda al usuario ubicarse mucho mejor, cada variable tiene una descripción para que sirve

y cómo influirá al momento de la ejecución de los scripts (ver las Ilustraciones 74, 75, 76, 77, 78 y 79):

Ilustración 74: Configuración de conexión



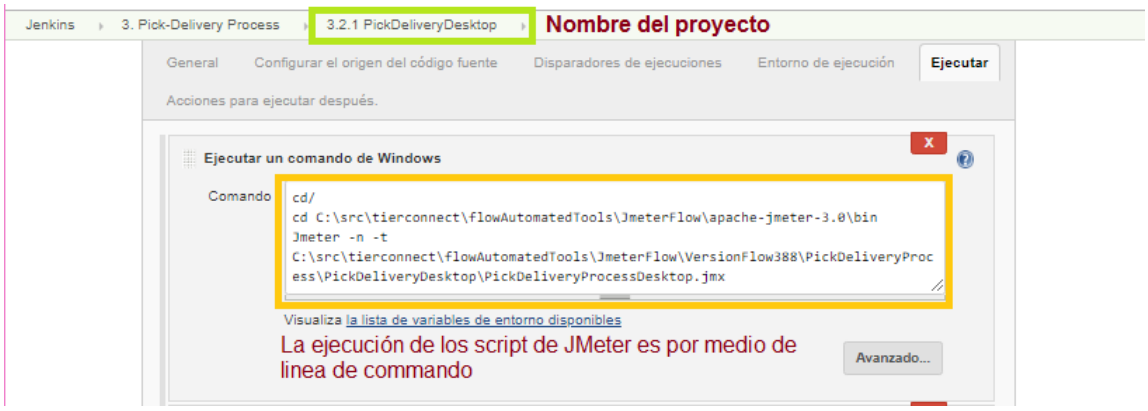
Fuente: Elaboración propia

Ilustración 75: Proyecto para el script de Internal



Fuente: Elaboración propia

Ilustración 76: Proyecto para el script de Exception



Fuente: Elaboración propia

Ilustración 77: Proyecto para la configuración de Management View



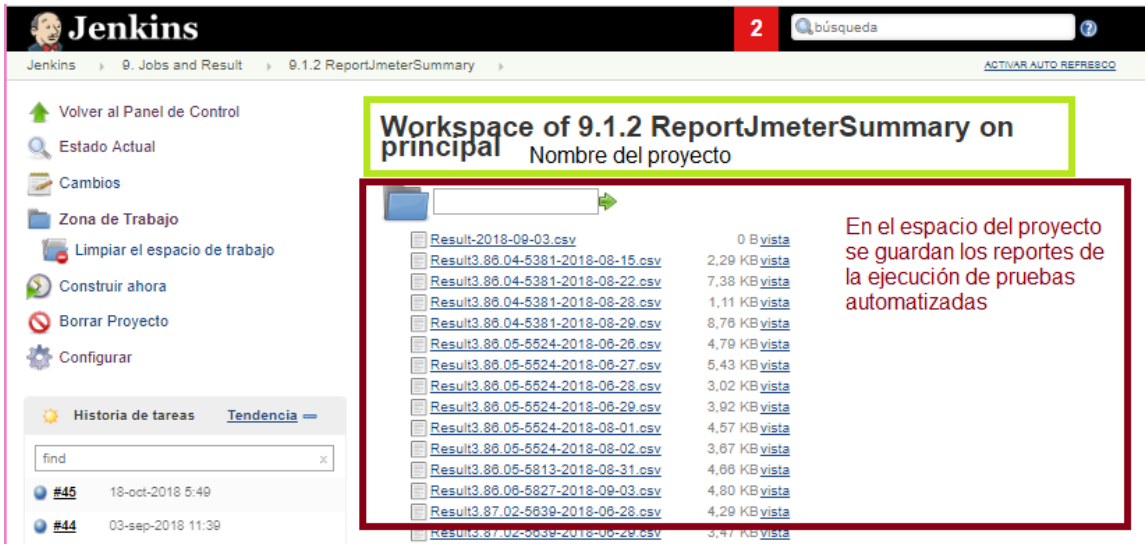
Fuente: Elaboración propia

Ilustración 78: Proyecto para Request History



Fuente: Elaboración propia

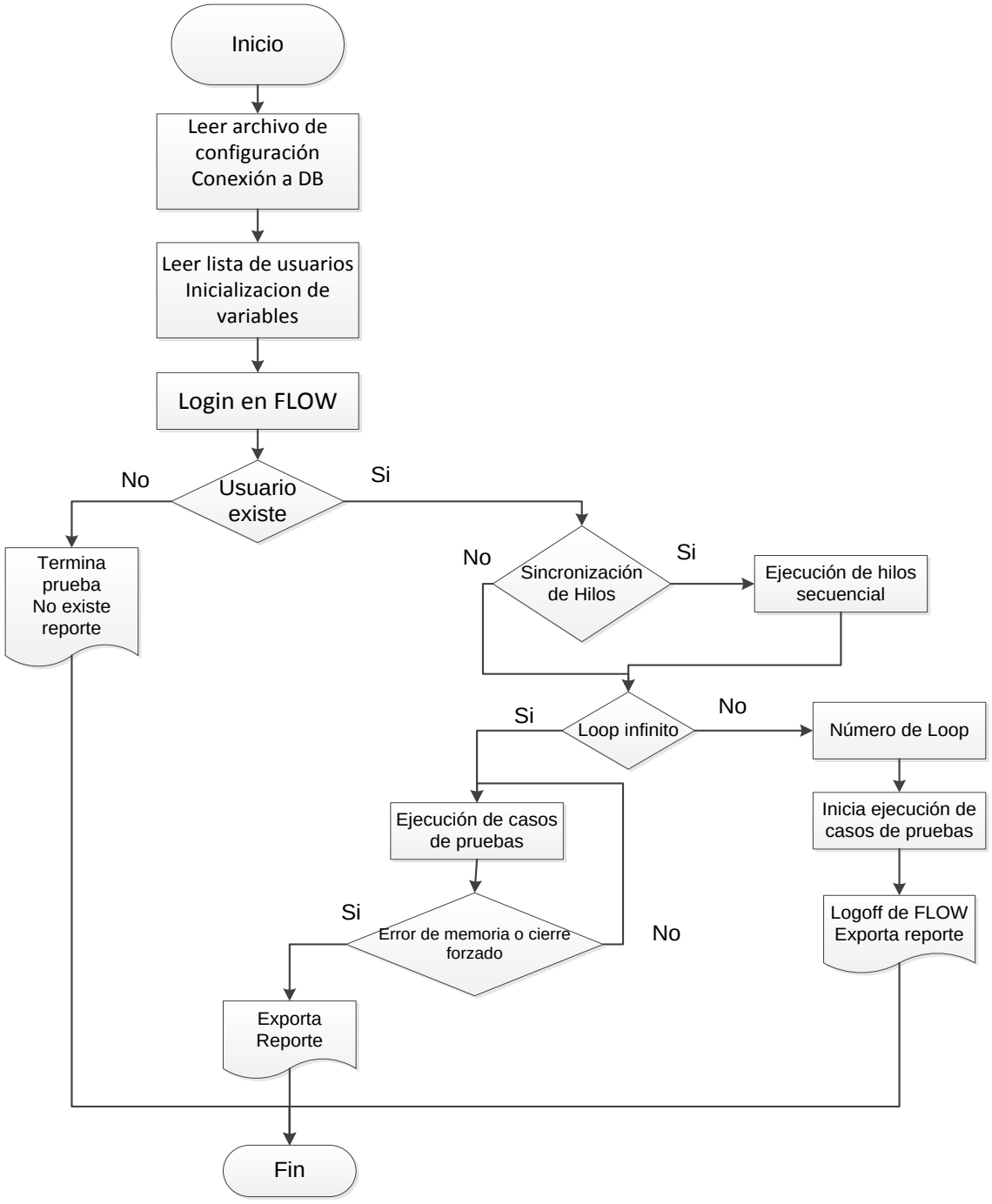
Ilustración 79: Vista de los reportes



Fuente: Elaboración propia

El diagrama 9 muestra el flujo del proceso de la automatización:

Diagrama 9: Proceso de ejecución de script de automatización



Fuente: Elaboración propia

En cada uno de los proyectos integrados con Jenkins se ejecutan 3 tipos de archivos con extensión jar, el primero es el “ReadAndWrite” el cual nos ayuda leer el reporte y contar los resultados de “Pass y Fail” para tener un dato más claro, después de ellos se ejecuta el jar de copyFile con este jar copiamos el reporte actual a una carpeta para que luego se ejecute el jar de JMeterRerports para tener un reporte con el resumen de todos los resultados finales de los reportes, estos jar se basan en la fecha de modificación para que no se confundan con otros archivos csv, el código de los archivos se los podrá observar en anexos.

Implementación de la Automatización

Una vez concluida la automatización con los script desarrollados e integrados con Jenkins se puedo obtener los siguientes resultados con los cuales se realizó una comparación con un testing manual.

Tabla 32: Lista de test case y comparación de tiempos

Pantalla	Casos de pruebas	Tiempo Manual (Min)	Tiempo Automatizado (seg)
Internal	Completar entrega de uno a uno	17	0,57
	Completar entrega de manera multiple	17	2
	Asignarse a multiples pedidos por el botón begin delivery	17	0,28
	Asignarse a pedidos uno a uno por el botón next request	17	0,43
	Asignarse a un pedido con mensaje de Excepción	17	0,72
	Completar el pedido con mensaje de Excepción	17	0,41
	Asignarse a un pedido con excepción de cantidad en la pantalla de Delivered Quantity	17	0,47
	Completar el pedido con excepción de cantidad en la pantalla de Delivered Quantity	17	0,57
	Exportar la lista CSV de internal	17	300

	Cancelar un pedido en la lista de Delivery	17	No automatizado
	Cancelar un pedido en la pantalla de Delivery Request Part	17	No automatizado
	Pedido con excepción de cantidad en el proceso de delivery complete	17	0,28
	Sortear la lista de pedidos con los filtros	17	0,18
	Refrescar pantalla	17	
Exception	Completar el pedido con excepción de uno a uno	17	0,51
	Completar el pedido con excepción de manera múltiple	17	0,4
	Cancelar una Excepción	17	0,21
	Completar la Excepción	17	0,39
	Exportar la lisa CSV de la pantalla de excepción	17	120
Manegement View	Exportar la lista CSV de management view	17	300
	Vaciar la lista de pedidos	17	90
	Vaciar pedidos con excepción	17	1,31
	Cancelar un pedido	17	1,81
	Cancelar un pedido con Excepción	17	1,24
	Cambiar las prioridades de un request (overdue, requested, need, need son, critical)	17	1,41
	Cambiar los estados de un request (Exception, Hold, Void)	17	2,05
	Buscar un pedido por cada campo	17	3

	Verificar información para los campos de scanner LPN	17	1
	Verificar información para los campos de asignación LPN	17	1
	Verificar información para el campo de asignación a usuario	17	1
	Seleccionar el checbox para mostrar datos archivados	17	30
Request History	Buscar un pedido por cada campo	17	No automatizado
Request History	Buscar pedios por rango de fechas	17	No automatizado
	Exportar archivo csv	17	300
	Verificar información para los campos de scanner LPN	17	No automatizado
	Verificar información para los campos de asignación LPN	17	No automatizado
	Verificar información para el campo de asignación a usuario	17	No automatizado
	Seleccionar el checbox para mostrar datos archivados	17	No automatizado

Fuente: Elaboración propia

Después de una comparación de tiempos manuales con el tiempo de la automatización se tiene los siguientes resultados en resumen, en la tabla 33 muestra todos los casos de prueba manuales y automatizados de las 4 pantallas.

Tabla 33: Test case automatizados por Pantalla

	Internal	Excepción	Managemnt View	Request History
Total test case	14	5	32	28
Total test case automatizado	11	5	31	1

Fuente: Elaboración propia

En la tabla 34 se muestra una comparación de tiempo que se invierte realizando las pruebas manuales contra las pruebas automatizadas.

Tabla 34: Comparación de tiempos para las pruebas

		Tiempo total (Hrs)	Tiempo día
Total casos de prueba	79	10,7667	1,3458333
Test automatizados	48	0,23923	0,0299042

Fuente: Elaboración propia

En la tabla 35 se demuestra el tiempo que se ahorra a los integrandes del área control de calidad con la implementación de la automatización al memento de realizar la ejecución de los casos de prueba.

Tabla 35: Tiempo reducido con la automatización

Total de casos de prueba	Tiempo total Manual (seg)	Tiempo total automatico (seg)	Elaboración test plan (Seg)	% Reducido
48	30600	1161,24	7200	72,6756863

Fuente: Elaboración propia

4.7. Aplicación de la estimación de coste COCOMO

Para realizar la estimación de costos en cuanto a la elaboración de los scripts y la integración con Jenkins usaremos las siguientes formulas:

Lineas de código=5721/53

KLDC= (PF * Linea de código por cada PF)/1000

KLDC= (107.94*53)/1000=5.72

- El caso del modelo es del tipo orgánico

Por otro lado hallaremos la variable FAE, la cual será la siguiente:

FAE=1.15*1.15*1.11*1.06*1.00*1.00*1.00*0.91*1.00*1.00*1.00*1.00*0.91*1.04

FAE=1.34

Justificación de los valores:

- **Fiabilidad requerida del software:** Si se enviará un dato erróneo en la configuración los script no ejecutarían correctamente (Valoración Alta).
- **Tamaño de la base de datos:** No se usa una base datos de almacenamiento, solo se realizan consultas (Valoración Muy Baja).
- **Complejidad del producto:** Los script realizan consultas de con cada usuario que lo ejecuta por lo tanto deben de ser sincronizados para no colapsar la base de datos (Valoración Alta).

Atributos de hardware

- **Restricciones del tiempo de ejecución:** En los requerimientos se exige alto rendimiento (Valoración Alta).
- **Restricciones del almacenamiento principal:** No hay restricciones al respecto (Valoración Nominal).
- **Volatilidad de la máquina virtual:** Se usarán sistemas de la "Familia Windows" (Valoración Nominal).
- **Tiempo de respuesta del ordenador:** El usuario solo interactuara con al momento de la ejecución y la obtención de los reportes (Valoración Nominal).

Atributos del personal

- **Capacidad del analista:** Capacidad nominal, debido a la experiencia en scripts de similar desarrollo y configuración (Valoración Nominal)
- **Experiencia en la aplicación:** Se tiene cierta experiencia en el desarrollo de otros scripts (Valoración alta).

- **Capacidad de los programadores:** Teóricamente deberá tenerse una capacidad nominal en el lenguaje java y el uso de la herramienta JMeter (Valoración nominal).
- **Experiencia en S.O. utilizado:** Con Windows 10 la experiencia es a nivel usuario (Valoración Nominal).
- **Experiencia en el lenguaje de programación:** Es nominal, dado que se controlan las nociones básicas y las propias del proyecto (Valoración nominal).

Atributos del proyecto

- **Prácticas de programación modernas:** Se usarán prácticas de programación mayormente convencional (Valoración Nominal).
- **Utilización de herramientas software:** Se usara la herramienta JMeter que no exigirán apenas formación, de las cuales se tiene cierta experiencia (Valoración Alta).
- **Limitaciones de planificación del proyecto:** Existen muchos límites de planificación. (Valoración Alta).

Cálculo del esfuerzo del desarrollo

$E=aKLDL^{e} * FAE = 3.2 * (5.72)\ ^{\wedge}1.05 *1.34 =26.76$ persona/mes

Cálculo tiempo de desarrollo

$T=c\ Esfuerzo^{d} = 2.5 *(26.76)\ ^{\wedge}0.38 =8.71$ meses

Productividad

PR= LDC/Esfuerzo = 5721/26.76= 213.79 LDC/ persona mes

Personal promedio

PR= E / T= 26.76/8.71=3.07 personas

Así tenemos los resultados que son necesarios 3 personas para la creación de los scripts.

El salario de un desarrollador junior o senior como sueldo mínimo es de $289.52, por lo tanto la estimación que se tiene es la siguiente:

Costo del software por persona: 3*289.52= 868.56 $US

Costo total del software desarrollado: 8 *868.56 = 6948.48 $US

4.7.1. Costo de Implementación

En el área del desarrollo del proyecto cuenta con soporte técnico y una red interna con acceso a internet, por lo tanto el único costo será la implementación y configuración del servidor donde se instalara Jenkins y JMeter, el mismo tendrá un costo de 100 $US.

4.7.2. Costo de elaboración del proyecto

Los costos de elaboración del proyecto se refieren principalmente a los gastos que se realizan a lo largo de las diferentes fases de la metodología Ágil SCRUM y esta se explica en la siguiente tabla 35:

Tabla 36: Costo de elaboración del proyecto

Detalle	Importe ($US)
Análisis y desarrollo del proyecto	150
Material de escritorio	20
Internet	70
Otros	20
Total	260

Fuente: Elaboración propia

4.7.3. Costo total del Software

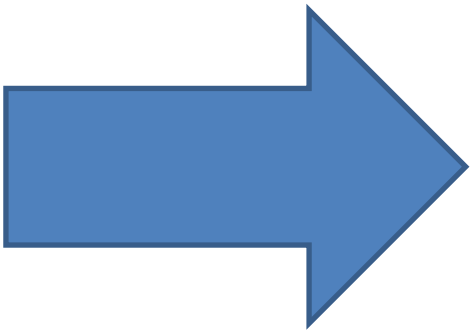
El costo total del software se lo obtiene de la sumatoria del costo de: desarrollo, implementación y elaboración del proyecto.

Tabla 37: Costo total del software

Detalle	Importe ($US)
Costo de desarrollo	6948,48
Costo de implementación	100
Costo de elaboración del proyecto	260
Total	7308,48

Fuente: Elaboración propia

CAPÍTULO V



"La educación formal tiene sus luces y sombras. Nuestro esfuerzo, más allá de que hoy seamos graduados, pueden servir para dar energía a las luces y hacer desaparecer las sombras"

Grant Smith

Capítulo V

En el presente capítulo se detalla las conclusiones en función al objetivo principal y los objetivos específicos y la forma en la que se cumplieron.

V. Conclusiones

Se desarrolló e implementó la automatización para las pantallas de Internal, Excepción, Management View y Request History en el sistema Material Flow para reducir tiempo de prueba en dichas pantallas, de tal manera que ayude al área de control de calidad.

La tabla 38 muestra el tiempo total de pruebas manuales y pruebas automatizadas, y el porcentaje que se redujo.

Tabla 38: Tiempos de test case automatizado

Total de casos de prueba	Tiempo total Manual (seg)	Tiempo total automatico (seg)	Elaboración test plan (Seg)	% Reducido
48	30600	1161,24	7200	72,6756863

Fuente: Elaboración propia

Para la automatización de los casos de prueba se hizo un análisis para determinar si los casos de prueba eran aptos para una automatización como se mostraron en tabla 32 (Lista de los casos de prueba para Internal), donde de 79 casos de prueba 48 fueron los automatizados.

Para la elaboración de los script fueron cuatro para cada pantalla teniendo en cuenta los casos de prueba para la automatización, aplicando la

herramienta de JMeter por lo cual se uso componentes propios de JMeter para las consultas sql, programación para que los script sean lo más adaptativo posible.

Para la implementación de la automatización se cumplió con los requerimientos, por lo cual se realizo la integración con Jenkins donde los usuarios pueden acceder al sistema de automatización a la dirección **10.0.0.149:9090**.

Después de cada ejecución de las pruebas automáticas se obtiene un archivo csv en la cual se muestra si los casos de pruebas fueron exitosos o fallidos, este archivo se muestra en la parte de anexos con el nombre Pick/Delivery por desktop, en la tabla 39 se muestra el cumplimiento de los requerimientos.

Tabla 39: Cumplimiento de requerimientos

Requerimientos Generales		
Requerimiento	**Como se cumplio**	**Página**
Cada script debe ser la ejecución de un caso de de prueba de la lista en el plan de pruebas.	Los casos de prueba que fueron elegidos se detalla en las tabla 32	Pag. 132
Reporte de la ejecución de cada script.	Son los archivos csv una muestra esta en la parte de Anexos	Anexo del D a F
Ejecución de los scripts periódicamente.	Se cumplio cuando se integro los script con Jenkins.	Pag. 126
Notificación sobre la ejecución de los scripts.	Con la integración de Jmeter con Jenkins, cada vez que se ejecuta un caso de prueba y si este falla se envia una notificación a los integrantes de control de calida	Pag. 129

Plana de prueba automatizada.	Despues de cada ejecución del test automatizado se devuelve un archivo csv en el cual esta la informacion de general de todos los casos de prueba.	Pag. 130
Requerimientos para los scripts		
Archivo de configuración global.	Todos los script tienen su archivo de configuración	Pag. 75
Sincronización de los hilos que simularan a los usuarios.	Es una librería con la cual es posible sincronizar los hilos.	Ver los script
Archivo de usuarios.	Es el archivo csv donde se especifican los nombres de los usuarios	Ver los script

Fuente: Elaboración propia

VI. Recomendaciones

Todos aquellos casos de prueba que no se puedon automatizar seria recomentable que se aplique un nuevo análisis de tal manera sean aptos para una automatización.

Los script si bien devuelven los reportes sobre si los casos de prueba automatizados fueron o no exitosos, los mismos datos no se muestran de forma gráfica automáticamente es decir sería bueno implementar un módulo de estadísticas con los datos de los reportes.

Los otros casos de pruebas del test plan aún podrían ser aptos para una siguiente y futura automatización en el sistema Material Flow.

VII. Bibliografía

Alexander Menzinsky, Gertrudis Lopez, Juan Palacios. (2016). *Scrum Manager.* Scrum Manager.

Alvares, A. R. (s.f.). *Introduccion a la Automatizacion de procesos.*

Besterfield, D. H. (2009). *Control de Calidad .* México: Pearson Prentice Hall.

Blas Zurro, J. I. (2014). *Control de calidad en la producción Industrial.* Valladolid: Iniversidad de Valladolid.

Bubano, N. (2010). *ISTQB certificate test.* Estados Unidos : Fundation Level Syllabus.

Canto, C. (s.f.). *Automatización: Conceptos generales.*

Castro, J. L. (2008). *Calidad de software.* Guatemala.

Duarte, P. M. (2015). *Estandares de Prueba.* Coacalco de Berriozabal - estado de Mexico .

Esteve Ambrosito, D. A. (2014 - 2015). *Implantación de un proceso de automatización de pruebas para una aplicación software.* Valencia.

Gallego, M. T. (s.f.). *Metodología Scrum.* TCF - Gestion de proyectos informáticos.

Gutierrez, D. (2010). *Estimación de costos y planificación de proyectos.* Universidad de Los Andes.

Guzman, A. R. (2014). *Automatizacion de casos de prueba y pruebas de stress.* La Paz: Repositorio institucional UMSA.

Hernandez, G. L. (2008). *Importancia de la calidad de desarrollo de productos de software.* México.

Ingenieria de Sistemas. (2004). *Calidad de software y procesos.* ORT Software Factory.

Ingeniería industría innovación. (s.f.). *Metodología agile y scrum.* sin lugar: INCUBICON.

Jaime Gonzalez V., David Morales Peake. (1997). *UML Gota a Gota.* México: Pearson Eduacción.

James Rumbaugh, Ivan Jacobson, Grandy Booch. (2000). *El lenguaje Unificado de Modelado. Manual de referencia.* Castilla y Leon - Salamanca: Rational software corporation.

Jonas, L. P. (s.f.). *Informatica aplicada a la Automatización.*

Jordi Escriu Paradell, J. L. (2010). *Normas para la gestion de calidad ISO 9000.* Cataluña: Instituto catografico de Cataluña.

Kniberg, H. (2007). *Scrum y XP desde las trincheras.* Estados Unidos: INFO-enterprise software development series.

Maria Constanza Cubillos Rodriguez, D. R. (s.f.). El concepto de calidad. *Revista Universidad De La Salle*.

Orallo, E. H. (s.f.). *El lenguaje unificado de modelo (UML).*

Peralta, A. (s.f.). *Metodología Scrum.* Uruguay: Universidad ORT Uruguay.

Peralta, L. M. (2014). *Automatización del proceso de codificación y funcionalidad del sistema sas basdo en el uso de la herramienta rational functiona tester, aplicado la empresa aseguradora Rimac.* Lima - Peru.

Pinelo, D. (2009). *Introducción a UML.* Object management group (OMG).

Valerdi, D. R. (2007). *Estimación de costos de proyectos de software.* Massachusetts: CIMAT.

Villegas, E. R. (2015). *Sistema Web de gestion academica y repositotio virtual caso: Unidad Educativa Principe de Luz.* La Paz - Bolivia: Repositorio UMSA.

VIII. Webgrafia

Apache. (4 de Septiembre de 2017). *The Apache software fundation*. Obtenido de The Apache software fundation: jmeter.apache.org

Appium. (2 de Octubre de 2018). *Appium*. Obtenido de Appium: appium.io

Garzas, J. (9 de mayo de 2014). *¿Que es Jenkins?* Obtenido de www.javiergarzas.com/2014/05/jenkins-en-menos-de-10-min.html

Guru 99. (s.f.). *Guru 99 página principal*. Obtenido de Guru 99 página principal: https://www.guru99.com/

Guru99. (5 de mayo de 2018). *Guru 99*. Obtenido de Pruebas de automatizacio: tutorial proceso, planificación y herramientas: www.guru99.com/automation-testing.html

Jenkins. (15 de Noviembre de 2017). *Jenkins.* Obtenido de Jenkins: https://jenkins.io/doc/

Robot Framework. (16 de septiembre de 2018). *Robot Framework*. Obtenido de Robot Framework: robotframework.org

Sattar, R. (29 de Agosto de 2016). *ToolsQA*. Obtenido de ToolsQA: toolsqa.com/jmeter/what-is-jmeter/

Software Testing Help. (4 de marzo de 2018). *Software Testing Help*. Obtenido de Software Testing Help: softwaretestinghelp.com

Sutherland, K. S. (2013). *La guia de Scrum.* Scrum.org and Scruminc.

Glosario

Tortoise: Es un cliente de Apache “Subversion (svn)”, implementado como una extensión de Shell de Windows. Es intuitivo y fácil de usar, ya que no requiere la ejecución del cliente de la línea de comando de suversion y es de uso gratuito.

Suversion: Es un sistema de control de versiones de código abierto.

QA: Quality Assurance o Aseguramiento de calidad.

Script: Es un archivo de órdenes o archivo de procesamiento por lotes.

Material Flow: Sistema de reabastecimiento de pedidos en tiempo real.

Test Plan: Plan de pruebas en el cual se detallan los casos de prueba.

Test case: Casos de prueba para el testing.

Pass: Término usado si un test case o caso de prueba se ejecutó exitosamente.

Fail: Término usado si un test case o caso de prueba fallo en su ejecución.

JMeter: herramienta de desarrollo de casos de prueba automatico.

Jenkins: herramienta para la ejecución de casos de prueba automatico y automatización de procesos.

CSV: archivo de lectura y escritura separada por comas.

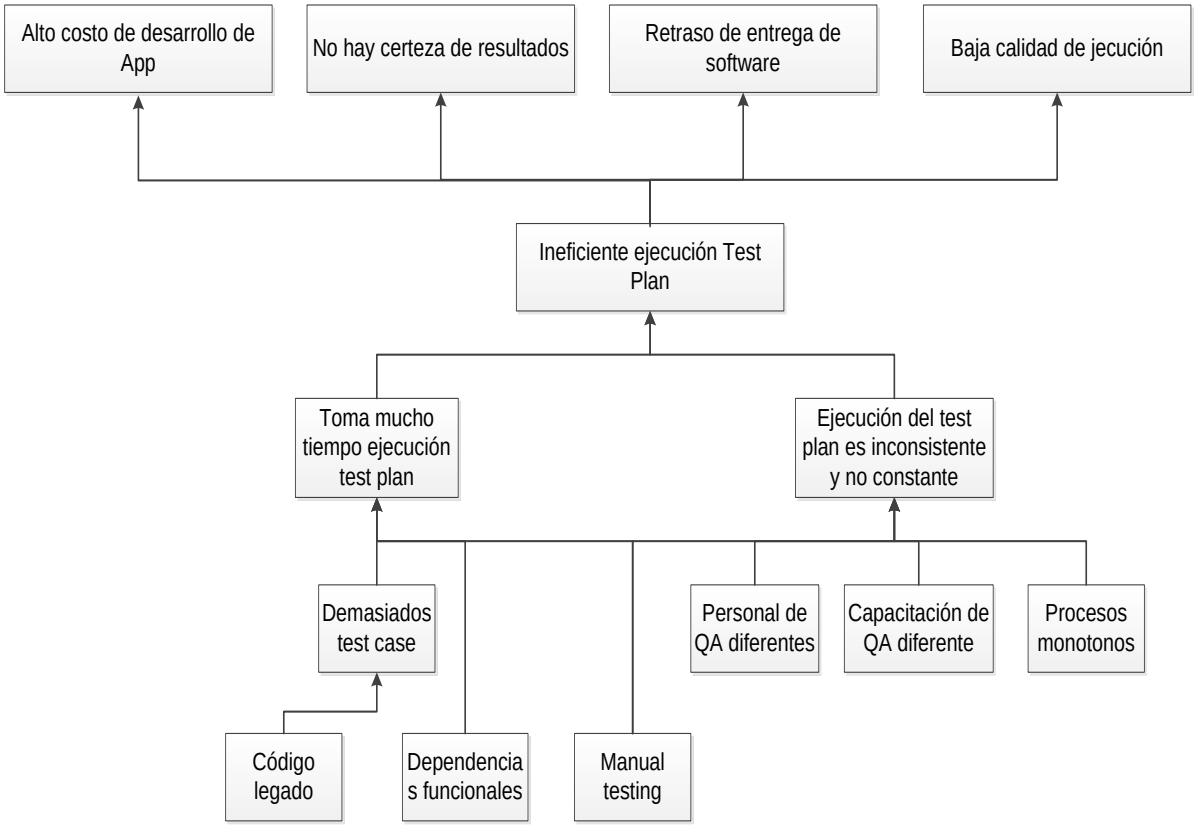
Cocomo: método para la estimación de costos.

Testing: ejecución de casos de prueba.

Anexos

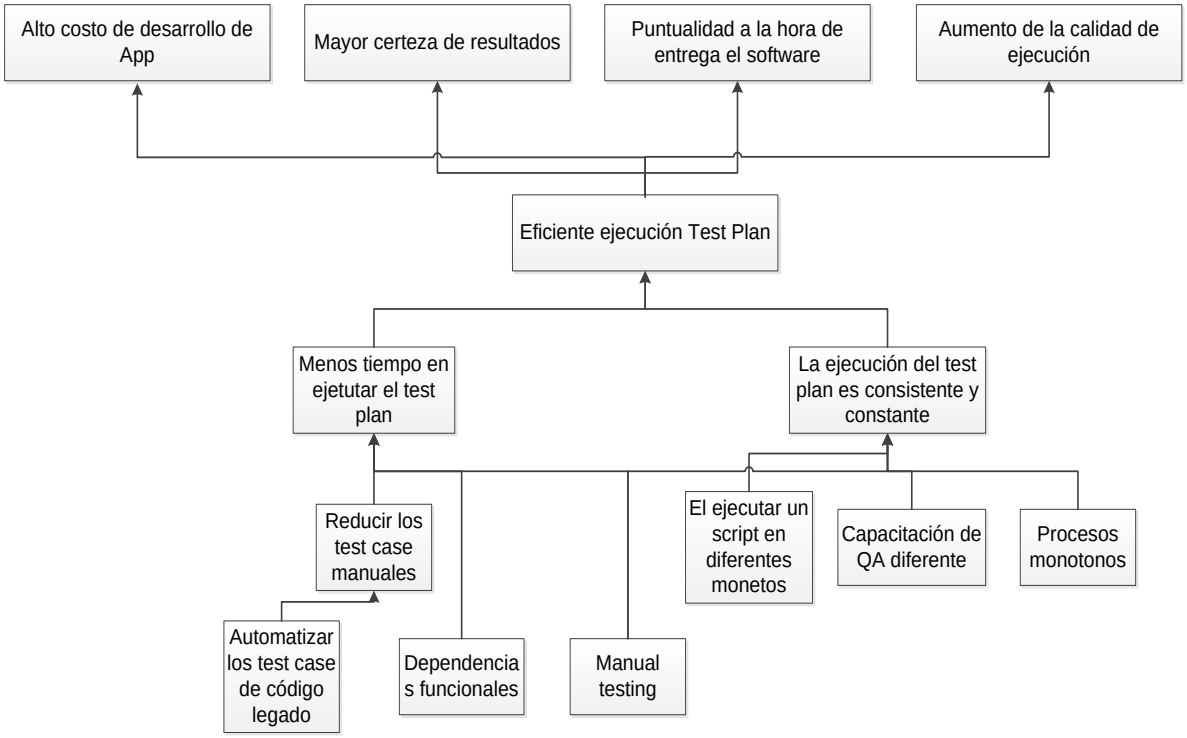
Anexos A

Árbol de Problemas



Anexo B

Árbol de Soluciones



Anexo C

Información de las Pantallas para automatización

La siguiente información es un pequeño manual de usuario que en la empresa se tiene y se le provee a los clientes.

Replenish: Request History

The history can show trends and timings for routes, shifts, users, part classes, etc. The users are encouraged to leverage the data to improve the efficiency of the facility.

Management View

The Management View allows managers to view all requests both internal and externally. Managers are also able to change the priority status, void, or cancel requests.

Set the filter conditions and click 

to execute the report.

Changing Priority Status

1. Select Request by clicking anywhere on the row except the checkbox.
2. Modify the Priority Status
3. Click 

Setting a Material Request to Exception or Void

1. Choose checkbox of the material request(s) you want to move to Exception of Void.
2. Click  or 

3. The Material Request will now have changed.

Internal

The Internal request is used for part route segments that are between internal locations and assigned to an internal route. Only, users that have been assigned to internal requests will be able to view requests from this screen. The screen will only show requests for the routes that the user is assigned to.

1. Click into the internal screen.
2. The user will be displayed a list of requests that have been prioritized by the settings found in the associated route.

Pick Process for Small Lot and Bulk Parts

1. Click 
    a. The system will bring a Pick Request Pop Window with the key information to fulfill the material request.
2. After picking the part the user clicks the following:
    a. **Small Lot Parts:** Click , if there is room or the delivery window has not been reached
    b. **Small Lot Pick Process Complete:** Click 
    c. **Bulk Parts:** Click 

Tip: **Choose Request**: Click  when multiple picked parts need to be delivered with the same delivery. Typically, intermediate staging locations with 3 carts with many picked items can leverage this feature.

Not all users will have the Choose Request feature. It is a permission associated to a role. Only, give this permission to material handling that will deliver many picked parts from many pickers.

Delivery Process for Small Lot and Bulk Parts

1. The application will display the Delivery List.
2. Select the part being delivered, anywhere on the row.
    a. Select Parts for delivery can by one by one or many at a time.
3. Click 
4. Repeat process until all parts are delivered

Exceptions

The Exception screen works the same way the internal screen works but is designed to highlight issues in the material flow. Material Request that are moved to the Exception screen need to be addressed and determined the correct course of action.

Tip: **Exceptions**: Click  if the material is not available when trying to deliver this will move the material request to the exception screen.

Tip: **Void**: Click  when you want to cancel the request. Void will cancel the request and place the request into history with a Void status.

Anexo D

Pick/Delivery por desktop. - Es el archivo csv que se obtiene después de la ejecución de los casos de prueba automatizados, se muestra la version de Material Flow, el nombre del caso de prueba, el usuario, la cantidad inicial de los pedidos, cantidad final de los pedidos, el estados si paso (Pass) o fallo (Fail), la fecha y el tiempo que tardo en ejecutarse respectivamente.

Build Versión: low-3.87.main-5532	Test case: Pick delivery by next request	User: jmeterPick2	Quantity request initial:100	Quantity request final: 99	Pass	Date: 2018-06-11 06:33:57	Time: 0.578943613
Build Versión: flow-3.87.main-5532	Test case: Pick delivery by begin delivey	User: jmeterPick2	Quantity request initial:99	Quantity request final: 69	Pass	Date: 2018-06-11 06:33:59	Time: 2.00462876
Build Versión: flow-3.87.main-5532	Test case: Pik assigned by next request	User: jmeterPick2	Quantity request initial: 0	Quantity request final: 1	Pass	Date: 2018-06-11 06:33:59	Time: 0.281653999
Build Versión: flow-3.87.main-5532	Test case: Pick assigned by begin delivery	User: jmeterPick2	Quantity request initial:1	Quantity request final: 31	Pass	Date: 2018-06-11 06:34:00	Time: 0.430761342
Build Versión: flow-3.87.main-5532	Test case: Pick with exception message	User: jmeterPick2	Quantity request initial:26	Quantity request final: 27	Pass	Date: 2018-06-11 06:34:01	Time: 0.728953751
Build Versión: flow-3.87.main-5532	Test case: Pick with exception quantity	User: jmeterPick2	Quantity request initial:100	Quantity request final: 101	Pass	Date: 2018-06-11 06:34:01	Time: 0.470652735
Build Versión: flow-3.87.main-5532	Test case: Delivery cancel in delivery list	User: jmeterPick2	Quantity request initial:32	Quantity request final: 30	Pass	Date: 2018-06-11 06:34:01	Time: 0.263902047

Build Versión: flow-3.87.main-5532	Test case: Cancel delivery by delivery complete	User: jmeterPick2	Quantity request initial:30	Quantity request final: 30	Pass	Date: 2018-06-11 06:34:02	Time: 0.184738115
Build Versión: flow-3.87.main-5532	Test case: Delivery with exception message	User: jmeterPick2	Quantity request initial:27	Quantity request final: 32	Pass	Date: 2018-06-11 06:34:02	Time: 0.419993498
Build Versión: flow-3.87.main-5532	Test case: Delivery with exception quantity	User: jmeterPick2	Quantity request initial:101	Quantity request final: 106	Pass	Date: 2018-06-11 06:34:03	Time: 0.569835613
Build Versión: flow-3.87.main-5532	Test case: Delivery with exception message-Delivey complete	User: jmeterPick2	Quantity request initial:32	Quantity request final: 37	Pass	Date: 2018-06-11 06:34:03	Time: 0.480938031
Build Versión: flow-3.87.main-5532	Test case: Delivery with exception quantity-Delivery complete	User: jmeterPick2	Quantity request initial:106	Quantity request final: 111	Pass	Date: 2018-06-11 06:34:04	Time: 0.561703659

Anexo E

Reporte para la pantalla Exception. - Es el archivo csv que se obtiene después de la ejecución de los casos de prueba automatizados, se muestra la version de Material Flow, el nombre del caso de prueba, el usuario, la cantidad inicial de los pedidos, cantidad final de los pedidos, el estados si paso (Pass) o fallo (Fail), la fecha y el tiempo que tardo en ejecutarse respectivamente.

Build Versión: flow-3.87.main-5532	Test case: Pick delivery exception by next request	User: jmeterPick2	Quantity request initial:58	Quantity request final: 57	Pass	Date: 2018-06-11 06:51:23	Time: 0.515697348
Build Versión: flow-3.87.main-5532	Test case: Pick delivery exception by begin delivery	User: jmeterPick2	Quantity request initial:57	Quantity request final: 52	Pass	Date: 2018-06-11 06:51:24	Time: 0.401501109
Build Versión: flow-3.87.main-5532	Test case: Void of request with exception	User: jmeterPick2	Quantity request initial:52	Quantity request final: 47	Pass	Date: 2018-06-11 06:51:24	Time: 0.218891416
Build Versión: flow-3.87.main-5532	Test case: Cancel of request with exception	User: jmeterPick2	Quantity request initial:47	Quantity request final: 42	Pass	Date: 2018-06-11 06:51:25	Time: 0.398524641

Anexo F

Reporte para la pantalla Management View.- Es el archivo csv que se obtiene después de la ejecución de los casos de prueba automatizados, se muestra la version de Material Flow, el nombre del caso de prueba, el usuario, la cantidad de registros que devuel la consulta sql o la cantidad inicial de los pedidos, la cantidad de registros que muestra en la aplicación o cantidad final de los pedidos, el estados si paso (Pass) o fallo (Fail), la fecha y el tiempo que tardo en ejecutarse respectivamente.

Build Versión: flow-3.87.main-5532	Test case: Search by field keyword with prefix name	User: zebra	Object searched: PLANT	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:30	Time: 0.589529463
Build Versión: flow-3.87.main-5532	Test case: Search by field keyword with base name	User: zebra	Object searched: TEST	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:30	Time: 1.072284412
Build Versión: flow-3.87.main-5532	Test case: Search by field keyword with base name	User: zebra	Object searched: IT	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:31	Time: 1.55846364

Build Versión: flow-3.87.main-5532	Test case: Search by field Part number	User: zebra	Object searched: PLANT-IT-TEST	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:31	Time: 2.025483671
Build Versión: flow-3.87.main-5532	Test case: Search by fields Part stage with part route number	User: zebra	Object searched: Production	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:32	Time: 2.495869079
Build Versión: flow-3.87.main-5532	Test case: Search by fields Part status with part route number	User: zebra	Object searched: Production-Approved	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:32	Time: 2.96581083
Build Versión: flow-3.87.main-5532	Test case: Search by fields route type with part route number and route type	User: zebra	Object searched: Testing	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:33	Time: 3.433379864
Build Versión: flow-3.87.main-5532	Test case: Search by fields route with part route	User: zebra	Object searched: Testing	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:33	Time: 3.86290765

	number and route							
Build Versión: flow-3.87.main-5532	Test case: Search by fields location type with part route and location type name	User: zebra	Object searched: Market Test	Request counted with DB:197	Request counted with application: 197	Pass	Date: 2018-06-12 12:04:34	Time: 4.436377099
Build Versión: flow-3.87.main-5532	Test case: Search by fields location with part route and location name	User: zebra	Object searched: 58TEST	Request counted with DB:197	Request counted with application: 0	Fail	Date: 2018-06-12 12:04:34	Time: 4.857288495
Build Versión: flow-3.87.main-5532	Test case: Search by fields priority status	User: zebra	Object searched: Need	Request counted with DB:10	Request counted with application: 10	Pass	Date: 2018-06-12 12:04:35	Time: 5.308649126
Build Versión: flow-3.87.main-5532	Test case: Search by fields Material request status	User: zebra	Object searched: Excepción	Request counted with DB:10	Request counted with application: 10	Pass	Date: 2018-06-12 12:04:35	Time: 5.762873706

Build Versión: flow-3.87.main-5532	Test case: Search by field request call type	User: zebra	Object searched:Manual_Call	Request counted with DB:7	Request counted with application: 7	Pass	Date: 2018-06-12 12:04:35	Time: 6.198645165
Build Versión: flow-3.87.main-5532	Test case: Search by field keyword with cleaned part number	User: zebra	Object searched: PLANTITTEST	Request counted with DB:197	Request counted with application: 0	Fail	Date: 2018-06-12 12:04:36	Time: 6.653733649
Build Versión: flow-3.87.main-5532	Test case: Change priority status	User: zebra	Object search: need	Quantity at the start of the test:10	Quantity at the finish of the test: 13	Pass	Date: 2018-06-12 12:04:38	Time: 2.056821731
Build Versión: flow-3.87.main-5532	Test case: Change status to void	User: zebra	Object search test: Void	Quantity at the start of the test:197	Quantity at the finish of the test: 196	Pass	Date: 2018-06-12 12:04:39	Time: 1.316234721
Build Versión: flow-3.87.main-5532	Test case: Change priority status to excepción	User: zebra	Object search test: Excepción	Quantity of excepción at the start of the test:10	Quantity of excepción at the finish of the test: 11	Pass	Date: 2018-06-12 12:04:41	Time: 1.427241913
Build Versión: flow-	Test case: Change priority	User: zebra	Object search test: Hold	Quantity of hold at the	Quantity of hold at the	Pass	Date: 2018-06-12	Time: 1.222452756

3.87.main -5532	status to hold			start of the test:3	finish of the test: 4		12:04:42	
Build Versión: flow-3.87.main -5532	Test case: Change priority status to excepción in list	User: zebra	Object search test: Excepción	Quantity of excepción at the start of the test:11	Quantity of excepción at the finish of the test: 21	Pass	Date: 2018-06-12 12:04:43	Time: 1.391275849
Build Versión: flow-3.87.main -5532	Test case: Cancel request with excepción	User: zebra	Object search test: Excepción	Quantity of excepción at the start of the test:21	Quantity of excepción at the finish of the test: 18	Pass	Date: 2018-06-12 12:04:45	Time: 1.411773191
Build Versión: flow-3.87.main -5532	Test case: Void a request with excepción	User: zebra	Object search test: Excepción	Quantity at the start of the test:18	Quantity at the finish of the test: 15	Pass	Date: 2018-06-12 12:04:47	Time: 1.245260891
Build Versión: flow-3.87.main -5532	Test case: Void a delivery	User: zebra	Object search test:	Quantity at the start of the test:193	Quantity at the finish of the test: 190	Pass	Date: 2018-06-12 12:04:48	Time: 1.816084632

Anexo G

Range Date. - Es el código que se encuentra implementado en el script para management view, con este es possible calcular la fecha inicial y final para los casos de prueba de búsqueda por cada campo.

```
import java.util.Date;
import java.text.SimpleDateFormat;

String rangeDateSearch1=vars.get("rangeDateSearch");
String rangeSearch=rangeDateSearch1.toLowerCase().trim();
log.info("range searchs: "+rangeSearch);
if(rangeSearch.equals("alldays")){
vars.put("rangeForSearchDelivery","0");
Date date = new Date();
date.setDate(date.getDate());
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");
String formattedDate = simpleDateFormat.format(date);
vars.put("startDateInput",formattedDate);
vars.put("endDateInput",formattedDate);

SimpleDateFormat simpleDateFormatDB = new SimpleDateFormat("yyyy/MM/dd");
String formattedDateDB = simpleDateFormatDB.format(date);
vars.put("startDateInputDB",formattedDateDB);
vars.put("endDateInputDB",formattedDateDB);
}
if(rangeSearch.equals("today")){
vars.put("rangeForSearchDelivery","2");
Date startDate = new Date();
Date endDate = new Date();
startDate.setDate(startDate.getDate());
```

```
endDate.setDate(endDate.getDate());
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");
String formattedStartDate = simpleDateFormat.format(startDate);
String formattedEndDate= simpleDateFormat.format(endDate);
vars.put("startDateInput",formattedStartDate);
vars.put("endDateInput",formattedEndDate);

SimpleDateFormat simpleDateFormatDB = new SimpleDateFormat("yyyy/MM/dd");
String formattedStartDateDB = simpleDateFormatDB.format(startDate);
String formattedEndDateDB = simpleDateFormatDB.format(endDate);
vars.put("startDateInputDB",formattedStartDateDB);
vars.put("endDateInputDB",formattedEndDateDB);
}
if(rangeSearch.equals("yesterday")){
vars.put("rangeForSearchDelivery","1");
log.info("Range for search delivery: "+vars.get("rangeForSearchDelivery"));
Date dateYesterday = new Date();
dateYesterday.setDate(dateYesterday.getDate()-1);
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");
String formattedDateYesterday = simpleDateFormat.format(dateYesterday);
vars.put("startDateInput",formattedDateYesterday);

Date dateToday = new Date();
dateToday.setDate(dateToday.getDate());
String formattedDateToday = simpleDateFormat.format(dateToday);
vars.put("endDateInput",formattedDateToday);

SimpleDateFormat simpleDateFormatDB = new SimpleDateFormat("yyyy/MM/dd");
String formattedStartDateDB = simpleDateFormatDB.format(dateYesterday);
String formattedEndDateDB = simpleDateFormatDB.format(dateToday);
vars.put("startDateInputDB",formattedStartDateDB);
```

```
vars.put("endDateInputDB",formattedEndDateDB);
}
if(rangeSearch.equals("week")){
vars.put("rangeForSearchDelivery","3");
log.info("Range for search delivery: "+vars.get("rangeForSearchDelivery"));
Date dateWeek= new Date();
dateWeek.setDate(dateWeek.getDate()-7);
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");
String formattedDateWeek = simpleDateFormat.format(dateWeek);
vars.put("startDateInput",formattedDateWeek);

Date dateToday = new Date();
dateToday.setDate(dateToday.getDate());
String formattedDateToday = simpleDateFormat.format(dateToday);
vars.put("endDateInput",formattedDateToday);

SimpleDateFormat simpleDateFormatDB = new SimpleDateFormat("yyyy/MM/dd");
String formattedStartDateDB = simpleDateFormatDB.format(dateWeek);
String formattedEndDateDB = simpleDateFormatDB.format(dateToday);
vars.put("startDateInputDB",formattedStartDateDB);
vars.put("endDateInputDB",formattedEndDateDB);
}
if(rangeSearch.equals("month")){
vars.put("rangeForSearchDelivery","4");
log.info("Range for search delivery: "+vars.get("rangeForSearchDelivery"));

Date dateMonth= new Date();
dateMonth.setDate(dateMonth.getDate()-30);
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");
String formattedDateMonth = simpleDateFormat.format(dateMonth);
vars.put("startDateInput",formattedDateMonth);
```

```
Date dateToday = new Date();
dateToday.setDate(dateToday.getDate());
String formattedDateToday = simpleDateFormat.format(dateToday);
vars.put("endDateInput",formattedDateToday);

SimpleDateFormat simpleDateFormatDB = new SimpleDateFormat("yyyy/MM/dd");
String formattedStartDateDB = simpleDateFormatDB.format(dateMonth);
String formattedEndDateDB = simpleDateFormatDB.format(dateToday);
vars.put("startDateInputDB",formattedStartDateDB);
vars.put("endDateInputDB",formattedEndDateDB);
}
if(rangeSearch.equals("3month")){
vars.put("rangeForSearchDelivery","4");
log.info("Range for search delivery: "+vars.get("rangeForSearchDelivery"));

Date date3Month= new Date();
date3Month.setDate(date3Month.getDate()-90);
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");
String formattedDate3Month = simpleDateFormat.format(date3Month);
vars.put("startDateInput",formattedDate3Month);

Date dateToday = new Date();
dateToday.setDate(dateToday.getDate());
String formattedDateToday = simpleDateFormat.format(dateToday);
vars.put("endDateInput",formattedDateToday);

SimpleDateFormat simpleDateFormatDB = new SimpleDateFormat("yyyy/MM/dd");
String formattedStartDateDB = simpleDateFormatDB.format(date3Month);
String formattedEndDateDB = simpleDateFormatDB.format(dateToday);
vars.put("startDateInputDB",formattedStartDateDB);
```

```
vars.put("endDateInputDB",formattedEndDateDB);
}
if(rangeSearch.equals("6month")){
vars.put("rangeForSearchDelivery","5");
log.info("Range for search delivery: "+vars.get("rangeForSearchDelivery"));

Date date6Month= new Date();
date6Month.setDate(date6Month.getDate()-180);
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");
String formattedDate6Month = simpleDateFormat.format(date6Month);
vars.put("startDateInput",formattedDate6Month);

Date dateToday = new Date();
dateToday.setDate(dateToday.getDate());
String formattedDateToday = simpleDateFormat.format(dateToday);
vars.put("endDateInput",formattedDateToday);

SimpleDateFormat simpleDateFormatDB = new SimpleDateFormat("yyyy/MM/dd");
String formattedStartDateDB = simpleDateFormatDB.format(date6Month);
String formattedEndDateDB = simpleDateFormatDB.format(dateToday);
vars.put("startDateInputDB",formattedStartDateDB);
vars.put("endDateInputDB",formattedEndDateDB);
}
if(rangeSearch.equals("year")){
vars.put("rangeForSearchDelivery","6");
log.info("Range for search delivery: "+vars.get("rangeForSearchDelivery"));

Date dateYear= new Date();
dateYear.setDate(dateYear.getDate()-365);
SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MM/dd/yyyy");
String formattedDateYearh = simpleDateFormat.format(dateYear);
```

```
vars.put("startDateInput",formattedDateYearh);


Date dateToday = new Date();
dateToday.setDate(dateToday.getDate());
String formattedDateToday = simpleDateFormat.format(dateToday);
vars.put("endDateInput",formattedDateToday);


SimpleDateFormat simpleDateFormatDB = new SimpleDateFormat("yyyy/MM/dd");
String formattedStartDateDB = simpleDateFormatDB.format(dateYear);
String formattedEndDateDB = simpleDateFormatDB.format(dateToday);
vars.put("startDateInputDB",formattedStartDateDB);
vars.put("endDateInputDB",formattedEndDateDB);
}
```

Anexo H

Obetner valores de las consultas, searchRequestByForeachField.- cuando se realiza la consulta sql el resultado que devuelve es en un objecto por lo cual es necesario guarder cada registro en una variable para que se pueda utilizar en todo el script.

```
ArrayList listSearchID=vars.getObject("searchIds");
if(listSearchID==null){
        log.info("request list is null");
        }
else{
        if(listSearchID.size()>0){
                vars.put("requestIDs","true");
        vars.put("requestId",vars.getObject("searchIds").get(0).get("JDOID").toString());
        vars.put("cleanedPartNumber1",vars.getObject("searchIds").get(0).get("cleanedPartNu
mber").toString());
                log.info("Cleand part number: "+vars.get("cleanedPartNumber1"));
                vars.put("prefix1",vars.getObject("searchIds").get(0).get("prefix").toString());
                log.info("prefix: "+vars.get("prefix1"));
                vars.put("base1",vars.getObject("searchIds").get(0).get("base").toString());
                log.info("base: "+vars.get("base1"));
                vars.put("suffix1",vars.getObject("searchIds").get(0).get("suffix").toString());
                log.info("suffix: "+vars.get("suffix1"));

        vars.put("partStatus1",vars.getObject("searchIds").get(0).get("partStatus").toString());
                log.info("partStatus: "+vars.get("partStatus1"));
                vars.put("name1",vars.getObject("searchIds").get(0).get("name").toString());
                log.info("name: "+vars.get("name1"));
```

```
	vars.put("partStage1",vars.getObject("searchIds").get(0).get("partStage").toString());
		log.info("partStage: "+vars.get("partStage1"));
		vars.put("route1",vars.getObject("searchIds").get(0).get("route").toString());

	vars.put("routeName1",vars.getObject("searchIds").get(0).get("routeName").toString())
;
		log.info("route: "+vars.get("routeName1")+" ID:"+vars.get("route1"));
	vars.put("routeType1",vars.getObject("searchIds").get(0).get("routeType").toString());
	vars.put("routeTypeName1",vars.getObject("searchIds").get(0).get("routeTypeName").
toString());
		log.info("routeType: "+vars.get("routeTypeName1")+ "ID:
"+vars.get("routeType1"));
	vars.put("fromLocation1",vars.getObject("searchIds").get(0).get("fromLocation").toStri
ng());
	vars.put("locName1",vars.getObject("searchIds").get(0).get("locName").toString());
		log.info("fromLocation: "+vars.get("locName1")+" Id:
"+vars.get("fromLocation1"));
	vars.put("toLocation1",vars.getObject("searchIds").get(0).get("toLocation").toString());
		log.info("toLocation: "+vars.get("toLocation1"));
	vars.put("type01",vars.getObject("searchIds").get(0).get("type0").toString());
	vars.put("locTypeName1",vars.getObject("searchIds").get(0).get("locTypeName").toStri
ng());
		log.info("loccation Type Name: "+vars.get("locTypeName1")+" Id:
"+vars.get("type01"));
	vars.put("requestCallType1",vars.getObject("searchIds").get(0).get("requestCallType").t
oString());
		log.info("request call type: "+vars.get("requestCallType1"));
	vars.put("status1",vars.getObject("searchIds").get(0).get("status").toString());
		log.info("status: "+vars.get("status1"));
		if(vars.get("status1").equals("1")){
			vars.put("priorityStatusDB","requested");
```

```
                log.info("Priority status: "+vars.get("priorityStatusDB"));
                        }
        if(vars.get("status1").equals("2")){
                vars.put("priorityStatusDB","Need");
                log.info("Priority status: "+vars.get("priorityStatusDB"));
                }
        if(vars.get("status1").equals("3")){
                vars.put("priorityStatusDB","Need Soon");
                log.info("Priority status: "+vars.get("priorityStatusDB"));
                }
        if(vars.get("status1").equals("4")){
                vars.put("priorityStatusDB","Critical");
                log.info("Priority status: "+vars.get("priorityStatusDB"));
                }
        if(vars.get("status1").equals("5")){
                vars.put("priorityStatusDB","Overdue");
                log.info("Priority status: "+vars.get("priorityStatusDB"));
                }
        }
else{
        vars.put("requestIDs","false");
        log.info("Does exists request");
        }
}
```

Anexo I

ReadAndWrtiteJMeter.jar .- Este archivo se lo aplica a los archivos csv es para obtener los resultados de paso o fallo en un format numeral, este archivo jar analizará un archvio csv si su fecha de modificación es igual al del día actual.

```
package com.company;

import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

public class Main {

    public static void main(String[] args) {

        Date date=new Date();
        SimpleDateFormat simpleDateFormat=new SimpleDateFormat("MM/dd/yyyy");
        String dateActual=simpleDateFormat.format(date);
        String dateFile="";
        String nameFile="";

        File jarPath = new
File(Main.class.getProtectionDomain().getCodeSource().getLocation().getPath()
);
        String pathFile = jarPath.getParentFile().getAbsolutePath();

        File dir = new File(pathFile);
        String extensions = "csv";
        File[] files = dir.listFiles(new FilenameFilter() {
            @Override
            public boolean accept(File dir, String name) {
                return name.endsWith(".csv");
            }
        });
        for(File csvFile:files){
            nameFile=csvFile.toString();
            File fileName = new File(nameFile);
            dateFile=simpleDateFormat.format(fileName.lastModified());
            if(dateFile.equals(dateActual)){
                String getNameFile = fileName.getName();
                System.out.println(getNameFile);
                File origen = new File(nameFile);
                getReport(origen,origen);
            }
        }
```

```
    }
    public static void getReport(File fileToBeProcessed,File fileToBeWrite){
        BufferedReader bufferedReader;
        BufferedWriter bufferedWriter = null;
        try{
            String readLine="";
            int pass=0,fail=0, bloqued=0;

            StringBuilder temp= new StringBuilder();
                if(fileToBeProcessed.exists()) {
                    bufferedReader = new BufferedReader(new
FileReader(fileToBeProcessed));
                    while ((readLine = bufferedReader.readLine()) != null) {
                        if (readLine.contains("Pass")) {
                            pass = pass + 1;
                        }
                        if (readLine.contains("Fail")) {
                            fail = fail + 1;
                        }
                        if (readLine.contains("Bloqued")) {
                            bloqued = bloqued+1;
                        }
                    }
                    File outFile = fileToBeWrite;
                    if(!outFile.exists()){
                        outFile.createNewFile();
                    }
                    FileWriter fileWriter= new
FileWriter(outFile.getAbsoluteFile(),true);
                    bufferedWriter = new BufferedWriter(fileWriter);
            }
            bufferedWriter.write(temp.toString());
            bufferedWriter.write("\nResults ");
            bufferedWriter.write("\nTest Pass ,"+pass);
            bufferedWriter.write("\nTest Fail ,"+fail);
            bufferedWriter.write("\nTest Bloqued ,"+bloqued);
        } catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            if (bufferedWriter != null){
                try{
                    bufferedWriter.close();
                }catch (IOException e){
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Anexo J

CopyFile.jar.- Con este archive java se copiara a un repositorio de reportes después del analísis del archivo csv.

```
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Properties;
import java.io.File;

public class Main {

    public static void main(String[] args) {
        Date date=new Date();
        SimpleDateFormat simpleDateFormat=new SimpleDateFormat("MM/dd/yyyy");
        String dateActual=simpleDateFormat.format(date);
        String dateFile="";
        Properties properties = new Properties();
        InputStream inputStream = null;
        String propFileName = "";
        String pathCopy = "";
        String nameFile = "";

        propFileName = "configuration.properties";
        File jarPath = new
File(Main.class.getProtectionDomain().getCodeSource().getLocation().getPath()
);
        String pathProperties = jarPath.getParentFile().getAbsolutePath();

        File dir = new File(pathProperties);
        String extensions = "csv";
        File[] files = dir.listFiles(new FilenameFilter() {
            @Override
            public boolean accept(File dir, String name) {
                return name.endsWith(".csv");
            }
        });

        try {
            inputStream = new FileInputStream(pathProperties + "..\\..\\..\\"
+ propFileName);
            //System.out.println(pathProperties + "..\\..\\..\\" +
propFileName);
            if (inputStream != null) {
                properties.load(inputStream);
                if (properties.getProperty("pathCopy") != null) {
```

```
                    pathCopy = properties.getProperty("pathCopy");
                } else {
                    properties = null;
                }
            }
        } catch (FileNotFoundException e) {
            e.fillInStackTrace();
            properties = null;
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
        for(File csvFile:files){
            nameFile=csvFile.toString();
            File fileName = new File(nameFile);
            dateFile=simpleDateFormat.format(fileName.lastModified());
            //System.out.println(dateFile);
            if(dateFile.equals(dateActual)){
                String getNameFile = fileName.getName();
              //  System.out.println(getNameFile);
                File origen = new File(nameFile);
                File destino = new File(pathCopy + "\\"+getNameFile);

                try {
                    InputStream in = new FileInputStream(origen);
                    OutputStream out = new FileOutputStream(destino);

                    byte[] buf = new byte[1024];
                    int len;

                    while ((len = in.read(buf)) > 0) {
                        out.write(buf, 0, len);
                    }

                    in.close();
                    out.close();
                } catch (IOException ioe) {
                    ioe.printStackTrace();
                }
            }
        }
    }
}
```