UNIVERSIDAD JOSÉ ANTONIO PÁEZ

DESARROLLO DE UN FRAMEWORK PARA LA AUTOMATIZACIÓN DE PRUEBAS FUNCIONALES EN ARQUITECTURAS ORIENTADAS A SERVICIOS EN APLICACIONES WEB

Autor: Jesús Pacheco
C.I.: 23.631.741
Tutor Académico: Ing. Mariosca Carpio

Urb. Yuma II, calle Nº 3. Municipio San Diego
Teléfono: (0241) 8714240 (master) – Fax: (0241) 8712394

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE COMPUTACIÓN

DESARROLLO DE UN FRAMEWORK PARA LA AUTOMATIZACIÓN DE PRUEBAS FUNCIONALES EN ARQUITECTURAS ORIENTADAS A SERVICIOS EN APLICACIONES WEB

Trabajo de grado presentado como requisito para optar al título de
INGENIERO COMPUTACIÓN

Autor: Jesús Pacheco
C.I.: 23.631.741
Tutor Académico: Ing. Mariosca Carpio

San Diego, Octubre de 2017

ÍNDICE

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE COMPUTACIÓN

DESARROLLO DE UN FRAMEWORK PARA LA AUTOMATIZACIÓN DE PRUEBAS FUNCIONALES EN ARQUITECTURAS ORIENTADAS A SERVICIOS EN APLICACIONES WEB

Autoras: Jesús Eduardo Pacheco Álvarez
Tutora: Ing. Mariosca Carpio
Fecha: Octubre, 2017

RESUMEN INFORMATIVO

En la siguiente investigación se realizó un estudio de las herramientas que se llevan a cabo para el desarrollo del Framework de automatización de pruebas funcionales. A raíz de este estudio por la incorrecta elección de herramientas o al prescindir de una, las pruebas funcionales deben hacerse de forma manual validando gran cantidad de ventanas, campos y procesos dependiendo de la complejidad de la aplicación web, en consecuencia, hay un incremento de horas hombres, costos, equipos y personal que deben interactuar para cumplir con validaciones y análisis para garantizar un software de calidad. Por esta razón, se planteó el desarrollo de Framework de automatización de pruebas funcionales que agilice los procedimientos dentro de las empresas de desarrollo de software, de forma que optimice la capacidad de respuesta a cualquier problemática que se presentó, disminuyendo las horas hombres. Para el logro de los objetivos planteados se usó la metodología XP (Extreme Programming) que está basado en cuatro fases para obtener como resultado un Framework que cumplió con los dichos objetivos. La aplicación se desarrolló con el lenguaje de programación Java y utilizando las herramientas Selenium y Fitnesse. Así mismo, el estudio llevado acabo obedece un proyecto especial, donde se utilizaron técnicas como: observación documental.

Descriptores: Framework, aplicación web, XP.

Universidad José Antonio Páez
Facultad de Ingeniería

FI-SE-C-012-2017-2

Valencia, 04 de Octubre de 2017.

Ciudadano:
Jesús Pacheco
C.I: 23.631.741
Presente.-

Cumplo con informarle que la Comisión de Trabajo de Grado y Pasantías de la Facultad de Ingeniería en su reunión **Nº 3-2017** de fecha **04/10/2017** aprobó el proyecto de trabajo de grado titulado "**DESARROLLO DE UN FRAMEWORK PARA LA AUTOMATIZACIÓN DE PRUEBAS FUNCIONALES EN ARQUITECTURAS ORIENTADAS A SERVICIOS EN APLICACIONES WEB**" Presentado por usted como requisito para optar al título de Ingeniero en Computación.

Se ratifica la designación de la Ing. Mariosca Carpio, C.I. 21.480.875 y la Ing. Alicia Pizzella, C.I. 4.598.880 como Tutores Académicos que lo asesorarán en el desarrollo de este proyecto.

Atentamente,

UNIVERSIDAD JOSE ANTONIO PAEZ
UJAP
FACULTAD DE INGENIERIA

Prof. Zulay Salcedo
Decana (E) de la Facultad de Ingeniería

c. c. Coordinación de Pasantías y Trabajo de Grado (2).

ZS/fr

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE COMPUTACIÓN

ACEPTACIÓN DEL TUTOR

Quien suscribe, Ingeniero Mariosca Teresa Carpio Rivero portador de la cédula de identidad N° 21480875, en mi carácter de tutor del trabajo de grado presentado por el(los) ciudadano(s) Jesús Eduardo Pacheco Álvarez, portador(es) de la cédula de identidad N° 23631741, (respectivamente), titulado Desarrollo de un Framework para la automatización de pruebas funcionales en arquitecturas orientadas a servicios en aplicaciones web presentado como requisito parcial para optar al título de Ingeniero de Computación, considero que dicho trabajo reúne los requisitos y méritos suficientes para ser sometido a la presentación pública y evaluación por parte del jurado examinador que se designe.

En San Diego, a los ocho días del mes de septiembre del año dos mil diecisiete

Ing. Mariosca Carpio
C.I.: 21480875

INTRODUCCIÓN

El desarrollo de software en empresas especializadas para tal fin debe estar enfocado en la imperiosa realidad de garantizar calidad tanto en todo su proceso como en resultados (producto) para así traducirse en una estrategia empresarial y lograr a través de este, las ventajas competitivas en un entorno altamente globalizado.

Las empresas desarrolladoras de software, para garantizar calidad en sus productos deben realizar pruebas de software construido. Las pruebas tienen una gran importancia y representan un gran volumen de actividades ya que se sugiere que en un proyecto de desarrollo de software el 75% de las pruebas sean automatizadas y el resto manual; esto evidencia que la estrategia de automatizar pruebas es una decisión primordial.

La automatización de las pruebas funcionales reduce significativamente el esfuerzo dedicado a las pruebas de regresión en productos que se encuentran en continuo mantenimiento. La automatización de las pruebas debe ser considerada un proyecto en sí mismo con objetivos definidos.

Ahora bien, a pesar del impacto positivo que ofrece la implementación de las pruebas automatizadas, todavía existen organizaciones con necesidad de esta tecnología. En este proyecto de investigación se presenta una metodología y el conjunto de herramientas open source utilizado para la automatización de las pruebas funcionales en productos con interfaz web. Este conjunto de herramientas está compuesto por: Selenium, Fitnesse, Eclipse y el Explorador Google Chrome. Se describe la experiencia de utilizar la metodología en un proyecto de prueba específico y se demostrará la factibilidad para la automatización de las pruebas siguiendo las actividades y el conjunto de herramientas definidos. Si bien las herramientas asisten en las pruebas automatizadas, no brindan soporte para la organización de los artefactos del proyecto: scripts, documentos y reportes de ejecución. Como trabajo a futuro se propone integrar al conjunto de herramientas definido, la herramienta FitNesse para gestionar los artefactos, buscando mejorar la organización de las pruebas junto con la comunicación y colaboración del equipo de pruebas.

A continuación, se presenta la estructura general de la investigación definida en cuatro (4) capítulos que abarcan todo el proceso de desarrollo, y una breve descripción de cada uno de ellos:

Capítulo I: En este capítulo se describirá el problema con más detalles, realizando énfasis sobre la problemática existente, factores que influyen en el desarrollo de un Framework multiplataforma para la automatización de pruebas funciones en una empresa. Una vez expuesta la situación de contexto en la que se circunscribe el problema de investigación y precisado el problema. Se formula la pregunta que se espera responder con el desarrollo de la investigación. Luego el objetivo general representa el propósito global del proyecto y prosigue con los objetivos específicos son los fines que persigue la investigación en un tiempo determinado y por ultimo con la justificación se pretende fundamentar la importancia del problema que aborda la investigación.

Capítulo II: Dentro de este marco abarca las investigaciones previas y bases teóricas que dan base a la investigación, así como las definiciones de términos.

Capítulo III: Se presentan todas las características de la investigación que rigen el trabajo de grado, también se define la población y muestra en donde se realizara el estudio por medio de las técnicas y herramientas que facilitaran la investigación, y se describirán las fases metodológicas donde se va a detallar en forma precisa el desarrollo de cada fase de la metodología.

Capítulo IV: Se presentara todas las actividades que se debe realizar, para la construcción del Framework, en donde se presentará las diferentes herramientas que se usaron durante el desarrollo del mismo.

Capítulo V: Dentro de este marco se visualizara las conclusiones de la investigación y los objetivos alcanzados. Asimismo, se anexará las recomendaciones que se puede realizar a largo y corto plazo.

CAPÍTULO I

1.1. Planteamiento del problema

Dentro del marco de los Sistemas de Información (SI) son múltiples las aplicaciones que pueden desarrollarse para las empresas en sus diversos ámbitos de acción. Actualmente las Tecnologías de Información (TI) impactan cualquier ámbito del desarrollo tecnológico y científico del ser humano; su aplicación a numerosas áreas cotidianas se ha vuelto indispensable, aportando beneficios al poder administrar, entre otras cosas el conocimiento y su gestión, que actualmente es considerado parte del capital más importante de las organizaciones.

Estudiar la garantía de la calidad en el desarrollo de software, tanto en su proceso de desarrollo como en el producto resultante, se ha constituido en un tópico de importancia para la Ingeniería del Software. La calidad esperada en el producto puede determinarse a través de una disciplina definida por RUP (2005) como "una colección de actividades correspondidas que a su vez se relacionan con un motivo de preocupación importante" dentro del proyecto total donde estén enmarcadas.

Dentro de estas organizaciones se incluye la industria del software en Venezuela, actualmente adolecen de estrategias, métodos y modelos que afiancen sus procesos y las hagan competitiva dentro de su entorno nacional con miras al ámbito internacional. De forma específica, las escasas investigaciones nacionales enfocadas en la disciplina de pruebas dentro del proceso de desarrollo de software nacional hacen aún más difícil que dichas empresas tengan una estrategia que las permita realizar una evaluación de las diversas herramientas que presenta el mercado en cuanto a pruebas se refiere; razón por la cual, la decisión de optar por una herramienta que satisfaga las necesidades del proyecto de software, se tornan con frecuencia imprecisa. En la mayoría de los casos, esta elección se ve restringida a escoger herramientas que no son las más acordes a la naturaleza del proyecto, y en el peor de los casos, se niega el uso de estas herramientas imposibilitando su cooperación en el proceso de desarrollo de software. Esta carencia en el uso de herramientas de pruebas igualmente refleja la poca importancia y la informalidad que a lo largo del proyecto de software se le da a quien

proporciona las directrices para obtener calidad de software. La Disciplina de Pruebas que debe ser vista como la unión de roles, actividades, artefactos y su orden interno más que como un simple requisito para la aprobación final del producto.

De este modo por la incorrecta elección de herramientas o al prescindir de una, las pruebas funcionales deben hacerse de forma manual validando gran cantidad de ventanas, campos y procesos dependiendo de la complejidad de la aplicación web, en consecuencia, hay un incremento de horas hombres, costos, equipos y personal que deben interactuar para cumplir con validaciones y análisis para garantizar un software de calidad.

Así mismo, involucrarse en el tema de las pruebas del software con todas sus etapas es una realidad que debe estar presente en cualquier empresa que desee adentrarse en el mundo del desarrollo del software y, por supuesto la calidad. En la actualidad existen variadas herramientas de automatización y planificación del esfuerzo de pruebas que facilitan el desarrollo de esta disciplina.

1.2. Formulación del problema

De acuerdo a lo planteado se formula la siguiente interrogante ¿Cómo se puede optimizar las pruebas funcionales en todo el proceso de desarrollo de software en una empresa?

1.3 Objetivos de la investigación

1.3.1. Objetivo General

Desarrollar un Framework para la automatización de pruebas funcionales en arquitecturas orientadas a servicios en aplicaciones web

1.3.2. Objetivos Específicos

1.4 Justificación

El desarrollo del Framework multiplataforma realiza un estudio de todo lo concerniente a la disciplina de pruebas del software en el ciclo de desarrollo, dando importancia al aseguramiento de la calidad del software como promotor y gerente de calidad en el proceso de desarrollo del software y a la selección de herramientas de pruebas que soporten esta disciplina.

El modelo propuesto, está dirigido a las organizaciones desarrolladoras de software y prioritariamente hacia las empresas venezolanas, las cuales están obteniendo popularidad y aceptación dentro de este tiempo de cambio y transformación que transita nuestro país. Dicho modelo será validado y documentado de acuerdo a las normas establecidas en la Ingeniería del Software, con el propósito especial de disminuir drásticamente el tiempo dedicado a pruebas funcionales de un software, para a su vez ser invertido en diferentes tareas para avanzar en el desarrollo y la calidad de la aplicación, además, disminuir los costos empresariales al requerir menos equipo y personal necesario para la ejecución de pruebas funcionales manuales. En consecuencia, de esto la empresa podrá disminuir su tiempo de producción de software haciéndola más eficiente y con mejor calidad a menor costo.

Por último, el presente trabajo una vez implementado sirve como un medio eficaz para aumentar de promoción tanto del estudiante como de la Universidad.

1.5 Alcance de la Investigación:

El presente proyecto de investigación tiene como objetivo principal la elaboración de un Framework para la automatización de pruebas funcionales en arquitecturas orientadas a servicios en aplicaciones web para el desarrollo de software adaptados al contexto nacional y fundamentados en la Ingeniería de Software para que contribuyan a mejorar los procesos productivos de empresas venezolanas que desarrollan software para uso interno o comercial; y elevar la calidad de sus productos.

Bajo esta premisa, la investigación se orienta a buscar y aportar solución al problema de pruebas manuales en el proceso de desarrollo de software que tienen las

empresas en Venezuela y que por consiguiente, pueda ser aplicada de forma eficaz en este contexto, específicamente en las organizaciones.

Las principales características que destacaran en el Framework es que será multiplataforma y entre sus principales funciones estará los comportamientos más comunes en los sistemas de información tales como inicio de sesión, selección de rol, hacer clic en un botón, colocar un texto en un campo, entre otros. Este proyecto se realizará utilizando la metodología de desarrollo de software basada en Metodología XP.

CAPÍTULO II

2.1. Antecedentes

Para efectos del desarrollo de este proyecto se tomaron en cuenta una serie de trabajos realizados previamente por diversos autores para poseer una referencia que contribuya al logro de los resultados esperados.

Esmite I (2013) en su trabajo de grado **"Automatización y Gestión de las Pruebas Funcionales usando Herramientas Open Source"**, realizado en la Universidad de la República Montevideo, Uruguay; cuyo objetivo fue proponer una metodología y un conjunto de herramientas open source utilizado para la automatización de las pruebas funcionales de productos con interfaz web. Este conjunto de herramientas está compuesto por: Selenium y Eclipse. Se describe la experiencia de utilizar la metodología en un proyecto de automatización específico y se concluye la factibilidad para la automatización de las pruebas siguiendo las actividades y el conjunto de herramientas definidos. Si bien las herramientas asisten en las pruebas automatizadas, no brindan soporte para la organización de los artefactos del proyecto: scripts, documentos y reportes de ejecución. Como resultado integrar al conjunto de herramientas definido, la herramienta FitNesse para gestionar los artefactos, buscando mejorar la organización de las pruebas junto con la comunicación y colaboración del equipo de pruebas.

Este antecedente, muestra un conjunto grande de detalles que sustentaran a la investigación presente.

Por otro parte, Mendoza M (2014) en su trabajo de grado **"Modelo de referencia para la selección de herramientas de pruebas como soporte al proceso de desarrollo de software en PYMES Venezolanas"**, realizado en la Universidad Católica Andrés Bello, cuyo resultado fue es proponer un modelo de referencia dirigido a PYMES venezolanas que permitió apoyar la selección de herramientas de prueba adaptada a las características de sus proyectos y del contexto organizacional, para asegurar la calidad de sus productos. Los resultados de esta investigación se lograron

siguiendo cada uno de los pasos y/o actividades enmarcados en el Framework metodológico.

El estudio consultado y señalado en líneas anteriores refleja la mayoría de la base en la cual se sustenta a esta investigación como motor principal.

Pérez B (2015), en su trabajo de grado titulada **"Proceso de Testing funcional Independiente"**, realizado en el Ministerio del Poder Popular para la Educación para la Catedra Seminario Especial de Grado, cuyo objetivo fue realizar un estudio del estado del arte en lo referente a las pruebas y al proceso de pruebas de software. Esta información fue organizada y resumida para la realización de las pruebas funcionales de un producto. Se concluyó que el proceso es una guía útil para realizar pruebas funcionales de productos de software. Actualmente, es usado en los proyectos de pruebas del Laboratorio de Testing Funcional del Centro de Ensayos de Software.

Este antecedente sirve de aporte en la definición del objeto de estudio así como para evidenciar la existencia de estudios previos enfocados en el Desarrollo de pruebas automatizadas.

2.2. Bases Teóricas

Las bases teóricas son el sustento de la investigación, permitiendo describirla de forma precisa y exacta, de esta manera se observó una visión más amplia sobre la investigación y esto sirve como punto de partida de la misma. Para que los analistas puedan dar una solución acertada al caso estudio. En atención a ello se consideró necesario reforzar algunos conocimientos los cuales se describen a continuación.

2.2.1 Aplicación Web

Es un sistema de información (SI) donde una gran cantidad de datos volátiles, altamente estructurado, van a ser consultados, procesados y analizados mediante navegadores. Una de las principales características va a ser su alto grado de interacción con el usuario, y el diseño de su interfaz debe ser claro, simple y debe estar estructurado de tal manera que sea orientativo para cada tipo de usuario.

2.2.2 Framework.

El concepto framework se emplea unos muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Podemos encontrar framework para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, y para cualquier ámbito que pueda ocurrir. En general, con el término framework, se refiriere a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

2.2.3 Pruebas de software

Consiste en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo.

Es un elemento crítico para la garantía del correcto funcionamiento del software. Entre sus objetivos están detectar defectos en el software, verificar la integración adecuada de los componentes, verificar que todos los requisitos se han implementados correctamente, identificar y asegurar que los defectos encontrados sean corregidos antes de entregar el software al cliente.

Tipos de Pruebas

a) Prueba unitaria: Es la prueba realizada sobre pequeñas porciones de códigos individuales o unidades. Una unidad es la pieza de software más pequeña que se puede probar. Normalmente refiere al trabajo de un programador. En el

paradigma de orientación a objetos (OOLP) la unidad de trabajo se refiere a una clase, a un método de una clase o a un conjunto pequeño de clases. Se busca con su aplicación el beneficio de la detección temprana de defectos. En este sentido resulta oportuno recordar la frase atribuida a Boris Beizer según se cita acerca que "las fallas más notorias en la historia del desarrollo del software fueron todas debidas a defectos en las unidades, defectos que podrían haber sido encontrados con apropiadas pruebas unitarias". En el alcance de este trabajo las pruebas unitarias adquieren capital importancia, por lo cual se volverá reiteradas veces sobre su análisis.

b) Prueba de Integración: Estas pruebas se realizan una vez que los componentes individuales están prontos y debe realizarse la integración con otros. La prueba de integración es efectuada para mostrar que la combinación de componentes es incorrecta o inconsistente. Su objetivo apunta a encontrar fallas en las interacciones entre los componentes.

c) Prueba del Sistema: Estas son las pruebas que se realizan sobre el sistema completo. Abarcan tanto pruebas funcionales como no funcionales del sistema. La mayoría de las faltas deben haber sido identificadas durante las pruebas de unidad e integración. La prueba del sistema generalmente se considera apropiada para probar requerimientos no funcionales del sistema tales como seguridad, desempeño, exactitud y confiabilidad. Las pruebas del sistema, a su vez, pueden clasificarse según diferentes tipos:

    Prueba de Volumen: La estrategia consiste en someter el sistema a grandes volúmenes de datos. Su propósito es demostrar que no puede manejar el volumen de datos especificados en sus objetivos.

    Pruebas de Esfuerzo: La estrategia consiste en someter el sistema a cargas pesadas consistentes en un volumen máximo de datos o actividad durante un lapso corto.

Pruebas de Usabilidad: Su objetivo es encontrar problemas con la facilidad de uso, claridad de presentación, y otros, en su interacción con las personas.

Pruebas de Seguridad: Intentan violentar las comprobaciones de seguridad del sistema.

Pruebas de Desempeño: Su objetivo es demostrar que el sistema no satisface sus objetivos de desempeño o de eficiencia. Estos objetivos generalmente indican tiempos de repuesta y rendimiento.

Pruebas de Almacenamiento: Su propósito es demostrar que ciertos objetivos de almacenamiento, como ser la cantidad de memoria o espacio en disco a utilizar, no se han resuelto.

Pruebas de Conversión: Muchos sistemas sustituyen a otros o surgen como evolución de sistemas anteriores, por lo cual es necesaria una conversión de datos. El objetivo de estas pruebas es demostrar que los procedimientos de conversión no funcionan.

Pruebas de Instalación: Se prueban aspectos relacionados con la instalación, especialmente cuando esta es automatizada.

Pruebas de Confiabilidad: Su objetivo es probar las declaraciones específicas de confiabilidad establecidas para el sistema.

d) Pruebas de Aceptación: Son las pruebas que realizan los clientes o usuarios para conocer si el sistema funciona de acuerdo a los requerimientos y necesidades que establecieron para el mismo. Pueden o no involucrar a los desarrolladores del sistema y a los verificadores (testers). Son de fundamental importancia ya que brindan directamente al cliente elementos para su aprobación del sistema.
e) Pruebas de Regresión: Su objetivo es verificar que no ocurrió una regresión en la calidad del producto luego de un cambio, probando que el cambio ha causado que algo que funcionaba dejo de hacerlo. La estrategia consiste en ejecutar

nuevamente las pruebas ya realizadas. Deben permitir obtener confianza en que los cambios realizados no afectaron el software de maneras no deseadas.

f) Pruebas de Humo: Una prueba de humo es una versión condensada de un conjunto de pruebas de regresión. Se focaliza en probar, a un nivel alto, la funcionalidad más crítica. Las pruebas de humo buscan inestabilidades grandes o elementos clave defectuosos o faltantes, que harían imposible efectuar las pruebas tal como fueron planificadas, actuando como filtro previo a la realización de otras pruebas. Su nombre deriva de la prueba inicial que realizan técnicos de hardware al encender un dispositivo.

2.2.4 Automatización de pruebas

Gao, Tsao y Wu (2013) definen la automatización de las pruebas de software como:

> Aquellas actividades y esfuerzos realizados con la intención de automatizar tareas de ingeniería y operaciones en un proceso de pruebas utilizando estrategias bien definidas y soluciones sistemáticas. Es preciso contar para esto con una infraestructura que como mínimo permita ocultar las complejidades brindando a sus usuarios la posibilidad de elaboración, mantenimiento y ejecución de las pruebas de forma sencilla; así como la obtención y registro de resultados, mantener la documentación, trazabilidad y el seguimiento de incidentes. (p. 40)

Automatizar puede significativamente reducir el esfuerzo requerido para efectuar las pruebas en forma adecuada, de formas más eficientes, o significativamente incrementar la cantidad de pruebas que pueden llevarse a cabo en un lapso determinado. Se han logrado ahorros de hasta un 80% del esfuerzo manual. Se ha ahorrado en los costos o generado productos de mejor calidad y más rápidamente. Sin embargo, las ventajas de la automatización en general son alcanzables a mediano y largo plazo, dependiendo de cada tipo de herramienta, no estando garantizado el éxito solo por su mera utilización.

2.2.5 Selenium

Selenium es un conjunto de herramientas que permiten desarrollar scripts para pruebas de aplicaciones Web en diversos lenguajes de programación como C#, Groovy, Java, .Net, Perl, PHP, Python o Ruby y donde las pruebas pueden ejecutarse usando la mayoría de navegadores web. Además, permite realizar pruebas funcionales en dispositivos móviles iPhone y Android. Selenium se compone de dos herramientas:

Selenium IDE: Es un entorno de desarrollo implementado como una extensión de Firefox y permite grabar, editar y depurar pruebas funcionales. Una vez ejecutada una prueba, se desarrollan una serie de scripts en un lenguaje de scripting especial para Selenium el cual provee comandos que ejecutan acciones sobre objetos en el navegador. Esta herramienta permite una fácil grabación y reproducción de scripts mediante la selección inteligente de campos usando el identificador, el nombre o el xpath según se disponga o se necesite. Esta herramienta es muy útil para el desarrollo de casos de pruebas donde se tienen que realizar gran cantidad de verificación de no conformidades cada vez que es desarrollado un producto de software o se realizan modificaciones de éste. Selenium automatiza el proceso de pruebas y permite ejecutar un conjunto de pruebas completo si es necesario o pruebas particulares.
Selenium WebDriver: Es un entorno de automatización de pruebas que opera a partir de los lenguajes de programación mencionados en la definición de Selenium. La principal contribución de WebDriver son los controladores nativos que dan soporte a distintos navegadores (Internet Explorer, Mozilla Firefox, Google Chrome, Opera y Safari). Debido a que Selenium tiene las restricciones propias de Javascript (lenguaje con el que está hecho), WebDriver va más allá y dependiendo del navegador que queramos probar utiliza el mecanismo más apropiado, por ejemplo, en Mozilla Firefox se implementa como una extensión, para Internet Explorer hace uso de los objetos propios de automatización. Incluso puede hacer uso de las características de los navegadores desde el punto de vista del sistema operativo. Gracias a WebDriver

ya no es necesario de un navegador web real para lanzar los test, sino que utiliza una aplicación basada en HtmlUnit para simular el navegador. A partir de la versión 2 de Selenium se dispone de toda la funcionalidad que se tenía hasta ahora y adicionalmente, si es requerida, las capacidades de WebDriver.

POM: Page Object Model

La creación de casos de prueba de Selenium puede resultar en un proyecto no sostenible. Una de las razones es que se usan demasiados códigos duplicados. La desventaja del código duplicado es que el proyecto es menos sostenible. Si algún localizador cambiará, deberá recorrer todo el código de prueba para ajustar los localizadores donde sea necesario. Al usar el modelo de objeto de página, podemos crear un código de prueba no frágil y reducir o eliminar el código de prueba duplicado. Además de eso, mejora la legibilidad y nos permite crear documentación interactiva. Por último, pero no menos importante, podemos crear pruebas con menos pulsaciones de teclas. Se puede lograr una implementación del modelo de objetos de página separando la abstracción del objeto de prueba y los scripts de prueba.

2.2.6 FIT: Framework for integrated tests

FIT utiliza un mecanismo dirigido por comandos y datos. Este mecanismo implementa los siguientes pasos básicos:

Lee tablas que contienen tanto los parámetros para la prueba (datos, comandos) como los resultados esperados (ejemplos).

Interpreta cada tabla con una clase conocida como “fixture”.

Compara el resultado de correr el programa bajo prueba con los resultados esperados, resaltando coincidencias, discrepancias y excepciones.

Si bien fue concebida para pruebas de aceptación, puede considerarse agnóstica con relación al tipo de prueba.

Tablas de decisiones

Las tablas que contienen los casos de prueba deben estar en formato HTML y el resultado se expresa en el mismo formato. FIT fue diseñado

originalmente para trabajar con archivos HTML. La especificación original del programa motor de ejecución de pruebas, conocido como FileRunner, indica que se le puede suministrar tanto un solo archivo como un camino, de forma que lea todos los archivos contenidos en una estructura de directorios. Todo contenido que no sean tablas HTML lo ignora. FIT define formatos para las tablas, aplicables a diferentes escenarios, estableciendo también diversas convenciones, entre otras para la expresión de los resultados de las pruebas.

Fixtures

Las fixtures ejecutan objetos del sistema a probar (SUT). Implementan las pruebas a partir de la información que obtienen desde las tablas y se programan en el mismo lenguaje que el SUT. Esto último explica la necesidad de portar FIT a diferentes lenguajes para ampliar su campo de aplicación. La versión original fue escrita en Java; habiendo sido portado a diferentes lenguajes y plataformas.

Las fixtures puedan interpretarse como 'el pegamento' que une las tablas con los objetos a probar. El código que los programadores deben escribir generalmente es breve, simple y utiliza clases que suministra FIT, también llamadas fixtures. Implementan el patrón "Adapter".

Metodología de trabajo

Los usuarios y verificadores expresan los casos de prueba mediante tablas con formatos preestablecidos, pudiendo combinarlas. Una vez que dichas tablas han sido pobladas con los datos necesarios, los desarrolladores escriben el código que las vincula con los objetos a probar.

Para viabilizar la utilización de FIT es necesario establecer una estrecha colaboración entre los usuarios, verificadores y desarrolladores.

Resultados de las pruebas

FIT expresa el resultado utilizando el mismo formato de tabla que contiene el caso de prueba correspondiente. Utiliza colores aplicados sobre los elementos de la tabla que contienen los resultados esperados y los resultados obtenidos, siendo su significado el siguiente: si es verde el resultado coincide con el esperado, si es rojo el

resultado no coincide con el esperado, con amarillo se denota que ocurrió una excepción y con gris se indica que la prueba no se ejecutó. En los casos de no coincidencia muestra junto al valor esperado el valor obtenido. Si el valor esperado no fue especificado simplemente despliega el valor obtenido. Si ocurrieron excepciones brinda información pertinente sobre las mismas.

Column Fixture

Define un formato de tabla específico del mismo nombre y una fixture (clase) que la interpreta. La prueba consiste en ejecutar el programa bajo prueba una vez por cada fila de la tabla, suministrando para cada ejecución un juego de parámetros de entrada y resultados esperados, obtenidos desde sus celdas. Al finalizar cada ejecución compara los resultados obtenidos con los esperados y expone el resultado de esa comparación, implementando un oráculo de entrada/salida.

ColumnFixture Jesús Pacheco (2017)

2.2.7 Fitnesse

FitNesse es una herramienta para especificar y verificar los criterios de aceptación de la aplicación (requisitos). Actúa como un puente entre las diferentes partes interesadas (disciplinas) en un proceso de entrega de software. Su servidor wiki hace que sea fácil documentar el software. Su capacidad de testexecution le permite

verificar la documentación contra el software, asegurando que la documentación permanezca actualizada y el software no esté enfrentando una regresión.

Para que esto funcione, las pruebas deben definirse en un nivel de negocios, en conjunto con representantes de negocios. Son básicamente requisitos de negocio, presentados de una manera fácil de entender por todas las partes interesadas. Cuando sus requisitos no son ambiguos, pueden verificarse automáticamente con su aplicación.

2.2.8 Calidad de Software

Pressman (Pressman, 1998) define la calidad del software como:

> La concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente.

En la definición de la calidad del software pueden estar involucrados aspectos como la ausencia de defectos, aptitud para el uso, seguridad, confiabilidad y reunión de especificaciones. Sin embargo, hay algo importante que se debe tener presente, la calidad del software debe ser construida desde el comienzo, no es algo que puede ser añadido después.

Para que el producto final sea de calidad, el proceso por medio del cual éste es elaborado debe ser también de calidad. Es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario.

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

2.3 Definición de Términos

Automatización: Se refiere a la ejecución automática de tareas industriales, administrativas o científicas haciendo más rápido y efectivo el trabajo ayudando de este modo al ser humano.

Casos de Uso: Capturan requerimientos funcionales, suministrando una narrativa acerca de cómo se usa el sistema mediante la especificación de una secuencia de acciones entre un actor y el sistema.

CSS (Cascade Style Sheet): Es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación.

Datos de prueba: Define una recopilación de valores de entrada de prueba que se consumen durante la ejecución de una prueba, y los resultados esperados que sirven de referencia para objetivos comparativos durante la ejecución de una prueba.

Errores, Defectos (Faltas) y Fallas: Un paso, proceso o definición de dato incorrecto en un programa de computadora. Puede ser el resultado de una equivocación. (Potencialmente origina una falla).

Framework: es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Herramientas de pruebas: Aquellas que se pueden utilizar para aumentar la eficacia de las pruebas.

HTML (HyperText Markup Language): Es una herramienta para que el ordenador conectado a Internet interprete como visualizar el documento.

Internet: Es un conjunto de computadores desplegados por todo el mundo y conectados entre sí intercambiándose información.

Pruebas de Regresión: Su objetivo es verificar que no ocurrió una regresión en la calidad del producto luego de un cambio, probando que el cambio ha causado que algo que funcionaba dejo de hacerlo.

Requerimientos: Corresponden a lo que debe hacer el software y a las características que debe presentar. Los requerimientos del software tienen como característica esencial que deben ser comprobables (mediante experimentos).

Software Testing: Son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada. Es una actividad más en el proceso de control de calidad.

Tecnología: Es el conjunto de saberes, destrezas y medios necesarios para llegar a un fin predeterminado mediante el uso de objetos artificiales o artefactos.

Verificación y Validación: conjunto de procesos y técnicas de evaluación de software.

Web dinámica: se generan a partir de lo que el usuario introduce en un Web o formulario y que utiliza el servidor para construir una Web personalizada que envían al cliente.

CAPÍTULO III

3.1 Tipo de Investigación

Se considera que ésta investigación se encuentra dentro de un proyecto especial ya que se orienta a la elaboración de un software para la resolución de un problema.

Según el Colegio Universitario Fermín Toro (2008), señala que "Se concibe como un estudio o trabajo con objetivos y enfoques novedosos con un resultado tangible, susceptible de ser evaluado o utilizado y que no requiere responder a una necesidad institucional o social. (p. 60)."

3.2 Diseño de la Investigación

En cuanto al diseño de la investigación, es un diseño documental, porque se concreta exclusivamente en la recopilación de información en diversas fuentes a fin de desarrollar un Framework para la automatización de pruebas funcionales en una aplicación Web.

Considerando las posiciones teóricas sobre metodología de la investigación, Fidias G. Arias (2012) define "la investigación documental es un proceso basado en la búsqueda, recuperación, análisis, crítica e interpretación de datos secundarios, es decir, obtenidos y registrados por otros investigadores en fuentes documentales: impresas, audiovisuales o electrónicas. Como en toda investigación, el propósito de este diseño es el aporte de nuevos conocimientos (pág.27)".

3.3 Nivel de la Investigación

En relación con la investigación descriptiva, Hurtado (2000) señala que la misma “tiene como objetivo central lograr la descripción o caracterización del evento de estudio dentro de un contexto particular” (p.213).

Así mismo Prieto (1994) señala, que la investigación descriptiva “va a la búsqueda de aquellos aspectos que se desean conocer y de los que se pretende obtener respuesta”.

Con respecto al nivel de la investigación, se encuentra ubicada dentro de los parámetros de investigación descriptiva, ya que permite realizar un análisis del comportamiento de los diferentes procesos especificando los diferentes elementos que lo conforman. Atendiendo al problema en estudio, la investigación conllevara a conocer las posibles causas de fallas y/o errores en los procesos de pruebas manuales.

3.4 Población y Muestra

3.4.1 Población

Tamayo (2010) la define como “la totalidad del fenómeno a investigar poseedora de características comunes que proporcionan datos a la investigación” (p. 114).

Así mismo, el caso que ocupa este proceso investigativo tiene una población distribuida en los departamentos de calidad de una empresa de desarrollo de software en Latinoamérica.

3.4.2 Muestra

Para Hernández, Fernández y Baptista (2009), la muestra es “en esencia un subgrupo de la población. Es decir, la muestra es una parte de la población que se toma para realizar los estudios respectivos, la misma debe ser representativa para, de esta manera, poder generalizar los resultados alcanzados” (p.312).

Para el caso del presente estudio, la totalidad de la población es un número probabilístico manejable por el investigador, por lo que no se hace necesario trabajar con muestras, sino con la totalidad de la población.

Es de señalar, que según Balestrini (1997) “el estudiar la totalidad de la población permite la obtención de un alto grado de confiabilidad en la información que se pretende obtener por cuanto se disminuye el sesgo en lo posible” (p. 110).

3.5 Técnicas e Instrumentos de Recolección de Datos

Con el fin de alcanzar los conocimientos necesarios que permitirán alcanzar el objetivo del estudio, se empleará la técnica documental para la recolección u obtención de información y datos. Ésta tiene por objeto elaborar un marco teórico conceptual para formar un cuerpo de ideas sobre el objeto de estudio, a través de las fuentes primarias de información tales como libros, revistas, monografías, tesis, entre otras, así como de las fuentes secundarias como enciclopedias, manuales y otros. Con el uso de ésta técnica se sentarán las bases teóricas fundamentales para lograr el manejo del tema con propiedad.

Para lograr un análisis profundo de las fuentes documentales se utilizarán las técnicas denominadas observación documental.

Por una parte, la observación documental, como punto de partida en el análisis de las fuentes documentales, mediante una lectura general de los textos, se iniciará la búsqueda y observación de los hechos presentes en los materiales escritos consultados que son de interés para esta investigación. Esta lectura inicial, será seguida de varias lecturas más detenidas y rigurosas de los textos, a fin de captar sus planteamientos esenciales y aspectos lógicos de sus contenidos y propuestas, a propósito de extraer los datos bibliográficos útiles para el estudio que se está realizando.

Por su parte Hurtado (2007) define la observación documental como “Este tipo de observación está basado en la obtención de información de los libros, revistas, biografías, informes, actas -entro otros-. La técnica que se utiliza para tener un buen reporte de este tipo de observación son las fichas bibliográficas, las cuales sirven para citar y tener presentes las diversas fuentes que se han utilizado durante la observación. (p. 40)”

3.6 Fases Metodológicas

La metodología que se va a utilizar para la realización de este proyecto es XP (Extreme Programming). Es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Fase I: Establecer las herramientas de automatización para implementación de Framework

Inicialmente, se describen las herramientas a utilizar en el Framework. Por consiguiente, se llevaron a cabo las actividades referentes al levantamiento de información y el análisis de los resultados obtenidos del instrumento de recolección de datos aplicado, logrando así las características en los cuales se basaría el Framework a construir.

Actividad 1: Aplicar las técnicas e instrumentos de recolección de datos con respecto a las herramientas necesarias para identificar aquellas que aporten utilidad para la construcción del Framework.

Actividad 2: Se definirá las herramientas previamente investigadas para el avance de Framework.

Fase II: Diseñar el Framework de automatización para pruebas funcionales

En esta fase se empieza a moldear el Framework mediante el diseño de arquitectura y procedimientos, además se deben considerar como base la metodología XP para la fase de esta investigación. Se tiene que analizar y diagnosticar los factores que permitan la operatividad del Framework propuesto, mediante uso del Lenguaje de Modelo Unificado (UML). De este modo, se obtendrán los diagramas que permitirán el levantamiento del marco de trabajo.

Actividad 1: Aplicar el diagrama de clases, para tener una visión de la estructura del Framework.

Actividad 2: Diseñar los modelos conceptuales para visualizar la organización del Framework.

Fase III: Construir el Framework para automatización de pruebas utilizando herramientas de desarrollo computarizado

En esta fase se pretende construir el Framework utilizando el lenguaje de programación Java y las herramientas establecidas en la primera fase de la investigación, siguiendo las pautas de la metodología XP y el diseño propuesto hasta lograr el objetivo deseado.

Actividad 1: Descargar las librerías y herramientas definidas en la primera fase.

Actividad 2: Se construirá los módulos que permiten la funcionalidad del Framework.

Fase IV Realizar pruebas funcionales de caja negra: Se desarrollará una serie de Scripts utilizando el Framework para realizar pruebas funcionales en aplicaciones web.

Actividad 1: Definir los casos de pruebas a automatizar.

Actividad 2: Construir los scripts utilizando el Framework.

Actividad 3: Construir tabla de decisiones.

Actividad 4: Ejecutar las pruebas correspondientes.

CAPÍTULO IV

En este capítulo se analizó la información obtenida en la investigación con la cual puede lograr los objetivos trazados. Para lograr dichos objetivos se realizó las siguientes acciones:

En primer lugar, por medio de la revisión documental bibliográfica y electrónica, se pudo diagnosticar de la situación actual que se encuentra la automatización de pruebas funcionales con la finalidad de cumplir con las acciones se procedió a la evaluación y análisis de los datos recaudados.

4.1 Fase I: Establecer las herramientas de automatización para implementación de Framework.

Inicialmente se realizó la revisión del material bibliográfico relacionado con las herramientas de pruebas, aseguramiento de la calidad y calidad del software. En primer lugar, conocer los principales lenguajes de programación ayudará a determinar el grado de dificultad que puede acarrear las pruebas automatizadas, debido a esto se eligió el lenguaje de programación Java en su versión Java Standard Edition 8, debido a su facilidad de aprendizaje, por ser un lenguaje interpretado y multiparadigma. Además, como herramienta de edición se escogió Eclipse IDE en su versión 4.7 (Oxygen).

Así mismo, para la ejecución de tareas básicas sobre cualquier aplicación Web se escogió Selenium WebDriver API, la misma accede directamente al navegador para utilizarlo, manejando las funciones propias de automatización de cada uno de ellos y FitNesse el cual está diseñado originalmente como una interfaz altamente utilizable en torno al marco Fit. En este sentido, su intención es apoyar un estilo ágil de aceptación de pruebas de caja negra y pruebas de regresión. En este estilo de prueba, los probadores funcionales de un proyecto de desarrollo de software colaboran con los desarrolladores de software para desarrollar un conjunto de pruebas.

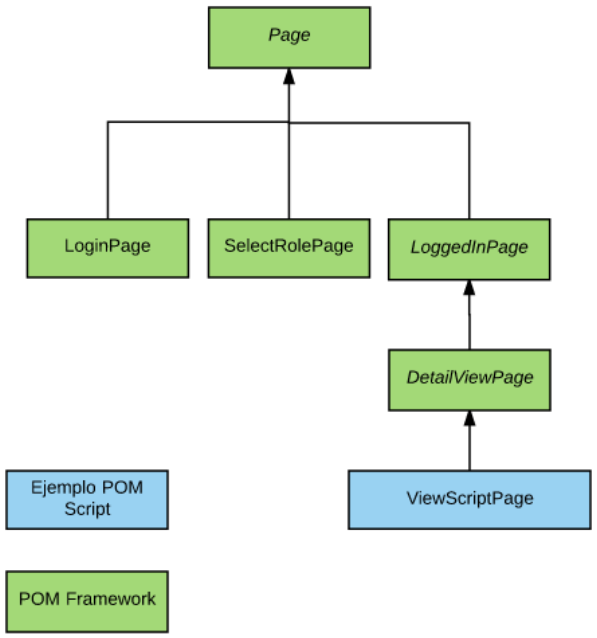
4.2 Fase II: Diseñar el Framework de automatización para pruebas funcionales

En esta fase se empieza a moldear el Framework mediante el diseño de arquitectura y procedimientos, el Framework está basado en el patrón Modelo – Vista

– Controlador siendo compatible con el mayor número de lenguajes y sistemas operativos posibles (multiplataforma).

4.2.1 Diseño de modelo conceptual

En esta sección se muestra las relaciones entre las clases POM (Page Object Model) del Framework sobre las operaciones más comunes sobre un navegador, a fin de tener una estructura organizativa en el mismo.



Modelo Conceptual Jesús Pacheco (2017)

4.2.2 Diseño de diagrama de clases

Se describe la estructura del marco de trabajo mostrando las clases del mismo, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

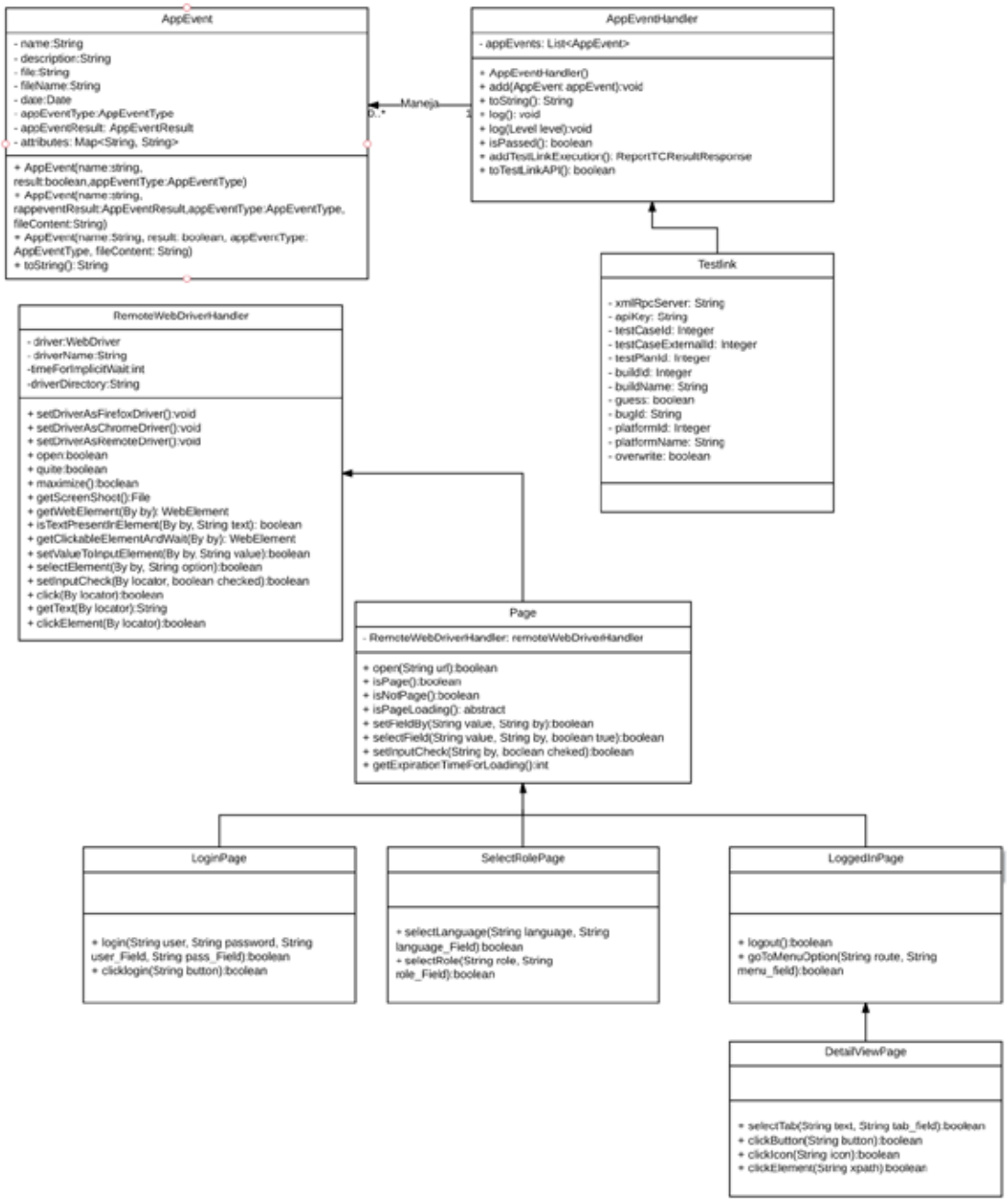


Diagrama de Clases Jesús Pacheco (2017)

4.3 Fase III Construir el Framework para automatización de pruebas utilizando herramientas de desarrollo computarizado

En esta fase se pretende desarrollar el Framework utilizando el lenguaje de programación Java y las herramientas establecidas en la primera fase de la investigación, siguiendo las pautas de la metodología XP y el diseño propuesto hasta lograr el objetivo deseado.

Ahora bien, para poder construir el Framework primero se procedió a crear un proyecto que contenga las dependencias de Selenium y Fitnesse para poder codificar las clases diseñadas en la fase anterior.

4.3.1 Codificación del Framework

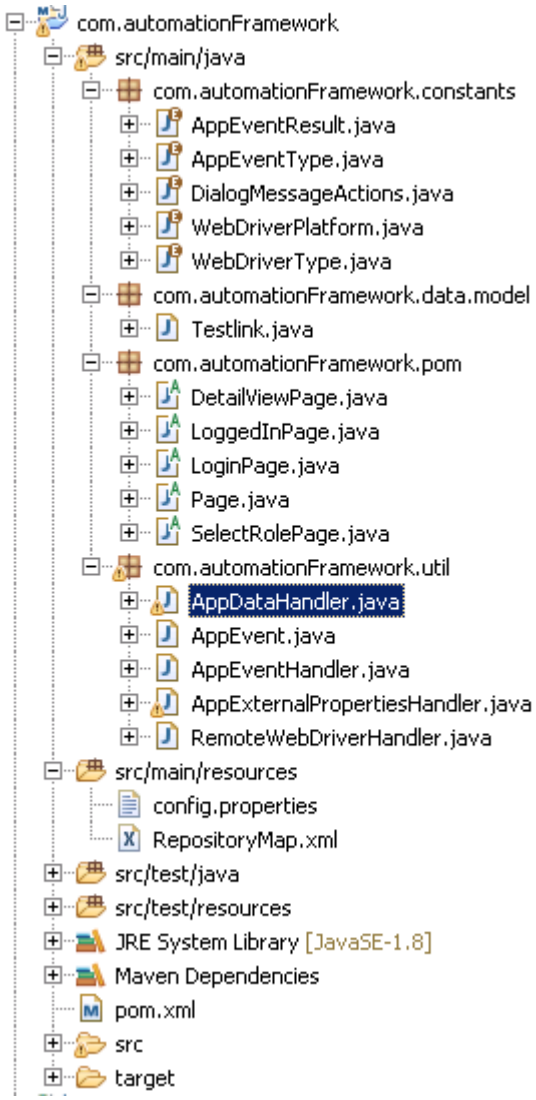
En primer lugar, se estructuro las clases por paquetes. Así mismo, se creó el paquete Constants, el cual contiene las clases AppEventResult para determinar los tipos de resultados que puede tener un determinado evento en la aplicación de pruebas; AppEventType contiene un listado de Tipos de Eventos (AppEvent), se utiliza para poder clasificar los eventos (Instancias de AppEvent) de una forma común con términos como Evidencia, Log, Error, etc; WebDriverPlatform Determina el tipo de plataforma a utilizar por determinada instancia de WebDriver, se asigna en el objeto DesiredCaptabilities recibido como parámetro durante la creación de una instancia de WebDriver y por último la clase WebDriverType determina el listado de browsers disponibles para utilizar en DesiredCaptabilities de una instancia de Webdriver.

En segundo lugar, se creó el paquete data.model el cual contiene la clase testlink para obtener datos relacionados con TestLink.

En tercer lugar, se creó la clase pom donde estarán las acciones más comunes que realiza un usuario en el navegador. Primero, la clase DetailViewPage que representa las acciones como seleccionar un tab, un elemento en pantalla, etc; seguido por la clase LoggedInPage que representa, según el patrón Page Object Model, páginas del de cuando el usuario ha iniciado sesión como menú, botón de cierre de sesión, ayuda, etc; la clase LoginPage representa la página de inicio de sesión de cualquier aplicación; Page Clase que encapsula los comportamientos comunes realizados por

aplicaciones Web como ingresar a la URL, escribir datos en un campo, etc; y por último, la clase SelectRolePage representa la selección de rol en una aplicación web.

Finalmente, se creó el paquete útil el cual contiene la clase AppEvent Clase para crear objetos tipo evento o incidencia, que representan determinadas acciones o resultados que se quieren plasmar en los resultados de las pruebas automatizadas; AppEventHandler Clase realizara para funcionar como agrupador y manejador de distintas instancias AppEvent o eventos registrados durante una prueba o set de pruebas y por ultimo RemoteWebDriverHandler clase utilizada para definir un RemoteWebDriver a utilizar en las pruebas con Selenium y agrupar los métodos utilizados para la creación de instancias de WebDriver. Agrupa también comportamientos comunes de acciones realizadas sobre aplicaciones Web.



Script de Prueba Jesús Pacheco (2017)

4.4 Fase IV Realizar pruebas funcionales de caja negra

Se desarrollará una serie de Scripts utilizando el Framework para realizar pruebas funcionales en aplicaciones web.

4.4.1 Definir los casos de pruebas a automatizar

Inicialmente se utilizó como aplicación de pruebas la web http://s3.sofoscorp.com/solicitudes/index.php. En efecto se procedió a crear un caso de

prueba para verificar el correcto funcionamiento del módulo de carga de horas. Para este caso se realizó las siguientes acciones:

Iniciar Sesión.

Ingresar a la ventana de carga de horas.

Hacer clic en el botón de guardar con los campos en blanco (Intento fallido).

Aseverar que los datos ingresados se hayan guardado satisfactoriamente

Cerrar Sesión.

4.4.2 Desarrollar Script para pruebas

En primer lugar, el entorno de desarrollo debe ser configurado, para ello se creó un proyecto para realizar las pruebas y luego se agregó como dependencia el JAR del Framework. Luego, se procedió a codificar el caso de prueba escogido, utilizando los métodos importados del Framework desarrollado.

```
public boolean openServer() throws Exception{
    server = args[0];
    driverType = args[1];
    remoteWebDriver = Boolean.valueOf(args[2]).booleanValue();
    remoteWebDriverHost = args[3];

    if (remoteWebDriver)
    {
      driverHandler = new RemoteWebDriverHandler(remoteWebDriverHost, WebDriverType.valueOf(driverType));
    }
    else {
      driverHandler = new RemoteWebDriverHandler(WebDriverType.valueOf(driverType));
    }
    if (loginPage.open(url)) {
      driverHandler.getAppEventHandler().add(new AppEvent("Abriendo AplicaciÃ³n en " + server, loginPage
        .isPage() ? AppEventResult.PASSED : AppEventResult.FAILED, AppEventType.EVIDENCE, driverHandler
        .getScreenshotAsString()));
      return true;
    }
    return false;
  }

  public boolean test() throws Exception {
      openServer();
      if(view.login(username, password)) {
          if(view.menu()) {
              if (!view.hoursCharge(null, null, null, null, null, null, null, null, null)) {
                  if (view.hoursCharge(date, client, project, fase, Activity, support, hours, deliverable, date_number)) {
                      return true;
                  }
              }
              view.logOut();
          }
      }

      return false;
  }
```

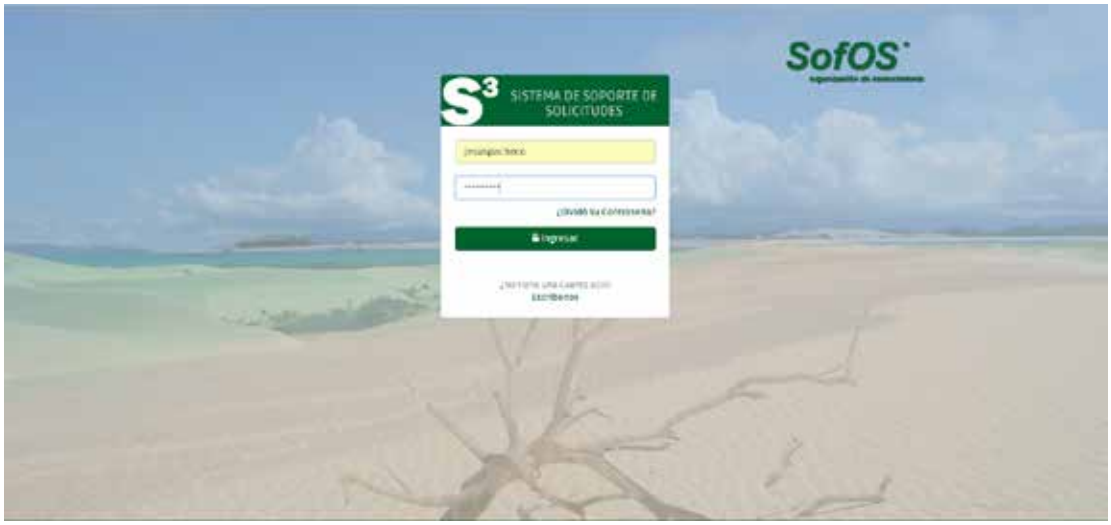
Script de Prueba Jesús Pacheco (2017)

4.4.3 Construir tabla de decisiones

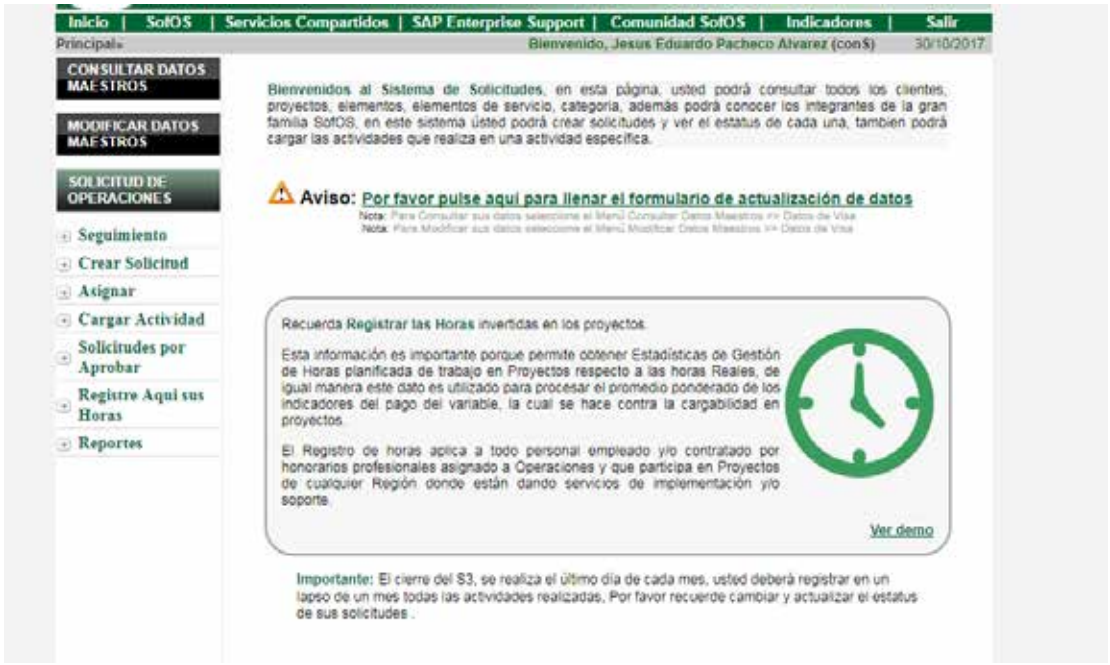
Dentro de este orden de ideas, para la lectura de datos de entrada se construyó una tabla HTML con sus respectivos valores de prueba.

4.4.4 Ejecutar las pruebas correspondientes

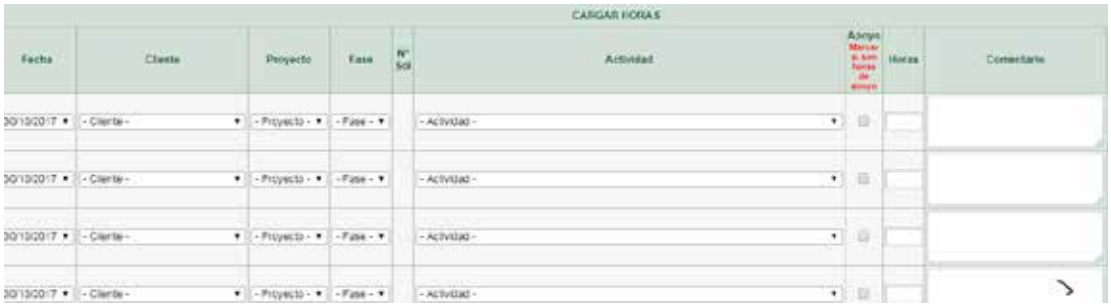
Por último, se realizo



Inicio de Sesión Jesús Pacheco (2017)



Ingresar a la Ventana Registre Aquí sus horas Jesús Pacheco (2017)



Carga de horas Jesús Pacheco (2017)

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

De la revisión literaria realizada en esta investigación se observa que las pruebas de software tienen gran importancia en la actualidad del desarrollo, que existen diversas técnicas para aplicarlas y que en algunas metodologías estas permiten dar un alto grado de calidad en los aplicativos desarrollados.

Para concluir, se han cumplido todos los objetivos marcados al principio del proyecto. Se pudo recabar información sobre la problemática de la continuidad en el desarrollo, manejo e implementación que aborda apreciaciones sobre el tema, tales como:

Todas las herramientas que se han instalado no han tenido absolutamente ningún coste en licencias, todas han sido OpenSource, por tanto, el ahorro es sustancial si comparamos lo que cuesta la licencia de otras herramientas destinas a la misma tarea.

La adopción de una metodología ágil de desarrollo que abarca todas las etapas del ciclo de vida del software ha permitido que en todas trabajemos bajo la misma disciplina, pese a que cada una ejecute tareas independientes, al 'hablar' el mismo lenguaje ha provocado que todo el proceso se desarrolle de forma más ágil.

Las pruebas realizadas han servido para demostrar que el tiempo destinado a las pruebas manuales son mucho mayor que el destinado a pruebas automatizadas, todo y que al principio la curva de aprendizaje pueda resultar un poco elevada y requiera más tiempo, a medio-largo plazo los beneficios se notan.

5.2 RECOMENDACIONES

Con el fin de mantener el Framework en un óptimo funcionamiento y ampliar el alcance del mismo se conciben las siguientes recomendaciones:

Completar implementación de Fixtures (DoFixture) para pruebas.

REFERENCIAS

Bibliográficas

Arias, F. (2006): **El proyecto de la Investigación: Introducción a la Metodología Científica**. Quinta Edición. Caracas: Episteme.

Hurtado, J. (2007): **El Proyecto de la Investigación: Metodología de la Investigación Holística**. Segunda Edición. Caracas: Quirón.

Kerlinger, F. (1985**): Investigación del Comportamiento: Métodos de Investigación en Ciencias Sociales**. Cuarta Edición. Long Beach: McGraw-Hill.

Meliá, Santiago (2013): **Un método de desarrollo dirigido por modelos de arquitectura para aplicación web.** Tesis doctoral. Universidad alicante.

Mijares, h. y García (2007). **Normas para la elaboración y presentación de los anteproyectos, proyectos y trabajos de grado.**

Sabino, C (2006): **Como se Elabora el Proyecto de Investigación**. Sexta Edición. Caracas: BL Consultores Asociados.

Universidad Pedagógica Experimental Libertador (2003): **Manual de Tesis de Grado y Especialización y Maestría y Tesis Doctorales**. Tercera Edición. Caracas: FEDUPEL.

Esmite, I (2013) **Automatización y Gestión de las Pruebas Funcionales Herramientas Open Source**,

Mendoza, M (2014) **Modelo de referencia para la selección de herramientas de pruebas como soporte al proceso de desarrollo de software en PYMES venezolanas.**

Pérez, B (2015) **Proceso de Testing funcional Independiente.**

Electrónicas

Ávila, C (2009):

http://www.javaworld.com/article/2071778/testing-debugging/fit-for-analysts-and-developers.html

Carrasco, B (2012):

https://dzone.com/refcardz/getting-started-fitnesse

León, A. (2011):

http://realsearchgroup.org/SEMaterials/tutorials/fit/

Martin, R. (2016):

http://fitnesse.org/FitNesse.UserGuide

Martin, N. (2012):

http://toolsqa.com/selenium-tutorial/

Méndez, cristina. (2014).

http://www.nubelo.com/blog/que-son-los-frameworks/

Mocholi, Ana (2015):

https://helpx.adobe.com/es/dreamweaver/using/web-applications.html

Sproul, N. (2010):

http://www.seleniumhq.org/

Zepeda, A. (2007):

http://fit.c2.com/