OPTIMIZACIÓN DE PRUEBAS DE SOFTWARE EN LA DIRECCIÓN DE GESTIÓN DE LA INFORMACIÓN DE LA AGENCIA NACIONAL DE DEFENSA JURÍDICA DEL ESTADO (ANDJE)

ANGE MELISSA GÓMEZ ESCOBAR

DICIEMBRE 2017

INSTITUCIÓN UNIVERSITARIA POLITÉCNICO GRANCOLOMBIANO
FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS

PROYECTO DE GRADO

BOGOTÁ D.C.

OPTIMIZACIÓN DE PRUEBAS DE SOFTWARE EN LA DIRECCIÓN DE GESTIÓN DE LA INFORMACIÓN DE LA AGENCIA NACIONAL DE DEFENSA JURÍDICA DEL ESTADO (ANDJE)

ANGE MELISSA GÓMEZ ESCOBAR

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERA DE SISTEMAS

ASESOR

Diego Iván Oliveros Acosta

Ingeniero electrónico e ingeniero de sistemas

INSTITUCIÓN UNIVERSITARIA POLITÉCNICO GRANCOLOMBIANO

FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS

PROYECTO DE GRADO

BOGOTÁ D.C.

Agradecimientos

A mi familia por apoyarme, estar siempre presentes y darme ánimos en todo momento, amor gracias por acompañarme durante toda la carrera, apoyarme y ayudarme en todo momento, a mi tutor gracias a su conocimiento y apoyo.

Resumen

El presente proyecto propone el desarrollo e implementación de una herramienta para la automatización de pruebas funcionales que cubra las necesidades en el desarrollo de software en el contexto particular de la Agencia Nacional de Defensa Jurídica del Estado en Colombia. En este sentido el trabajo aborda la importancia e impacto de las pruebas funcionales dentro del marco del ciclo de desarrollo de software y realiza un análisis comparativo de las herramientas para la automatización de software existentes en la actualidad que se adapten a las necesidades particulares en la organización escogida. De acuerdo con los resultados del estudio realizado se encontró que existen varias metodologías para la realización de pruebas automatizada, sin embargo, se evidenció una carencia de una metodología para la automatización. En consecuencia, se propone una metodología de automatización de pruebas basada en la experiencia y procesos de pruebas. Adicionalmente se presenta una herramienta para el grabado de pruebas funcionales basado en código abierto y se realizó un desarrollo especializado que permite guardar la documentación de los casos de prueba que se ejecutan en el proceso de pruebas en la entidad.

Abstract

This project proposes the development and implementation of a tool for the automation of functional tests that covers the needs in the software development in the context of the (Agencia Nacional de Defensa Jurídica del Estado) in Colombia. In this sense, the work addresses the importance and impact of functional tests within the framework of the software development cycle and performs a comparative analysis of existing software automation tools that are adapted to the needs of the organization chosen According to the results of the study carried out, it was found that there are several methodologies for automated testing, however, there was a lack of a methodology for automation. Consequently, a test automation methodology based on experience and testing processes is proposed. Additionally, a tool for the recording of functional tests based on open source is presented, and a specialized development was carried out to save the documentation of the test cases that are executed in the testing process in the entity.

Glosario

Automatización: Aplicación de máquinas o de procedimientos automáticos en la realización de un proceso o en una industria.

Daño antijurídico: Es fundamento de la responsabilidad patrimonial del Estado.

Pruebas: Hecho realizado para demostrar una acción, en sistemas es el hecho de comprobar los resultados de un sistema.

Selenium: Entorno de pruebas de software que permite la automatización de las mismas.

Software: Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

Contenido

Índice de figuras

Índice de tablas

1. Introducción

La ejecución de pruebas de software cumple un papel fundamental en el ciclo de vida de los proyectos informáticos, que permite garantizar la calidad de un software, Prevenir posibles defecto, garantizar que el producto final cumpla con los requerimientos del usuario final y mejorar los tiempos de puesta en producción entre otros y agilizar las pruebas ayudando a las empresas a que un software salga a producción más rápido y con mejor calidad (Collins, Dias-Neto, & de Lucena Jr., 2012). Por lo anterior se decidió realizar este trabajo de grado que se enfoca en la automatización de pruebas de software aplicada en el entorno particular de la Agencia Nacional de Defensa Jurídica del Estado en Colombia. En este sentido el trabajo aborda la automatización de pruebas desde las metodologías de prueba más relevantes, las herramientas de automatización existentes en el mercado y los procesos y procedimientos de pruebas utilizados en el entorno en el cual se desarrolla el trabajo. Es importante tener en cuenta que una herramienta no resuelve los problemas propios a la automatización de pruebas, para ello es necesario implementar conjuntamente una metodología para la agilización de los procesos las mismas.

Para este proyecto se realizará la aplicación de una metodología de desarrollo de software para la automatización de pruebas, si bien actualmente existen diversas metodologías sobre este campo, para efectos de este proyecto se abordará una metodología de pruebas desde la automatización de procesos y codificación de pruebas. Por lo tanto, para empezar, se utilizarán herramientas de código abierto para pruebas de automatización de aplicaciones web: complementos de Mozilla Firefox como Selenium IDE 2.9.1 y macros de Firefox, como motor de desarrollo se utilizará Eclipse. (Jústiz-Núñez, y otros, 2014)

Además, se realizará la documentación con la descripción del proyecto en un caso específico como lo es en la organización escogida y se concluirá con el impacto que tiene la automatización de pruebas con los procesos y herramientas establecidas.

Como trabajo futuro se propone realizar una herramienta para organizar los artefactos para mejorar el aprovechamiento de los mismos, mejorando la comunicación entre testers y desarrolladores y generando mejores reportes de los mismos.

2. Justificación

La tecnología nos ha facilitado la vida automatizando procesos en diferentes áreas, permitiendo a las personas aprovechar más su tiempo y conocimientos al no realizar tareas repetitivas, no por menos existen herramientas en pruebas de desarrollo de software que permiten automatizarlas, facilitando procesos de pruebas en diferentes ciclos de un proyecto, es por esto que se pensó en la aplicación de una metodología de automatización de pruebas de software en un sistema en evolución de desarrollo como lo es EKOGUI un sistema de gestión de información litigiosa del estado que se encuentra en la ANDJE el cual se describirán el (4.1 Entorno). En este sistema antes mencionado necesita garantizar la calidad constante y para mejorar el proceso de prueba se decide realizar este proyecto, aprovechando distintas herramientas que fortalecen diversas partes de la prueba como hace Selenium.

Con la automatización de pruebas de software podemos garantizar la calidad de un software que al final pueden denotar la discrepancia entre un proyecto de éxito y uno fracasado, permitiendo el aprovechamiento de recursos humanos y trabajar de forma ágil.

Dada la importancia de la universidad Politécnico Grancolombiano en desarrollar habilidades de innovación y el uso de las tecnologías de la información y la comunicación, se realizará un proyecto tecnológico aplicado a una entidad del estado, para la automatización del proceso de pruebas el cual permitirá ser el punto de partida para que otras personas implementen metodologías agiles para la automatización de pruebas, garantizando la calidad del software.

3. Objetivos

3.1. Objetivo General

Optimizar pruebas funcionales y de carga de desarrollo de software realizado en la Agencia Nacional de Defensa Jurídica del Estado(ANDJE)

3.2. Objetivos específicos

Aplicar una metodología de desarrollo de software para la automatización de pruebas en la ANDJE

Identificar posibles fallos y oportunidades de mejora para la automatización del proceso realizado en la ANDJE de la dirección de gestión de la información.

Evaluar las mejoras del proceso- de automatización de pruebas- mediante el uso de herramientas y estrategias que deriven de esta investigación.

4. Marco teórico

4.1. Entorno

4.1.1. Agencia Nacional de Defensa Jurídica del Estado

La Agencia Nacional de Defensa Jurídica del Estado es una entidad pública del orden nacional creada en el año 2011 con la misión de liderar la defensa jurídica de la Nación a través de la generación de conocimiento que permita al Estado, representado en las entidades públicas que lo componen, prevenir la ocurrencia de hechos que puedan derivar en futuros litigios en contra de la Nación y a su vez fortalecer la manera en la que el Estado ejerce la defensa judicial de los procesos que cursan actualmente en contra de él.

Dentro de las funciones otorgadas a la Agencia de Defensa Jurídica del Estado se destacan las siguientes:

- Diseñar y proponer estrategias, planes y acciones de resolución de conflictos,
- participación en procesos judiciales,
- formular políticas públicas de prevención del daño antijurídico,
- coordinar la defensa jurídica del Estado,
- desarrollar, implementar y administrar el Sistema único de Gestión de Información (Presidencia de la República, 2011)

Organigrama

La Agencia Nacional de Defensa Jurídica del Estado se divide en nueve áreas misionales organizadas jerárquicamente de acuerdo con la siguiente representación gráfica:

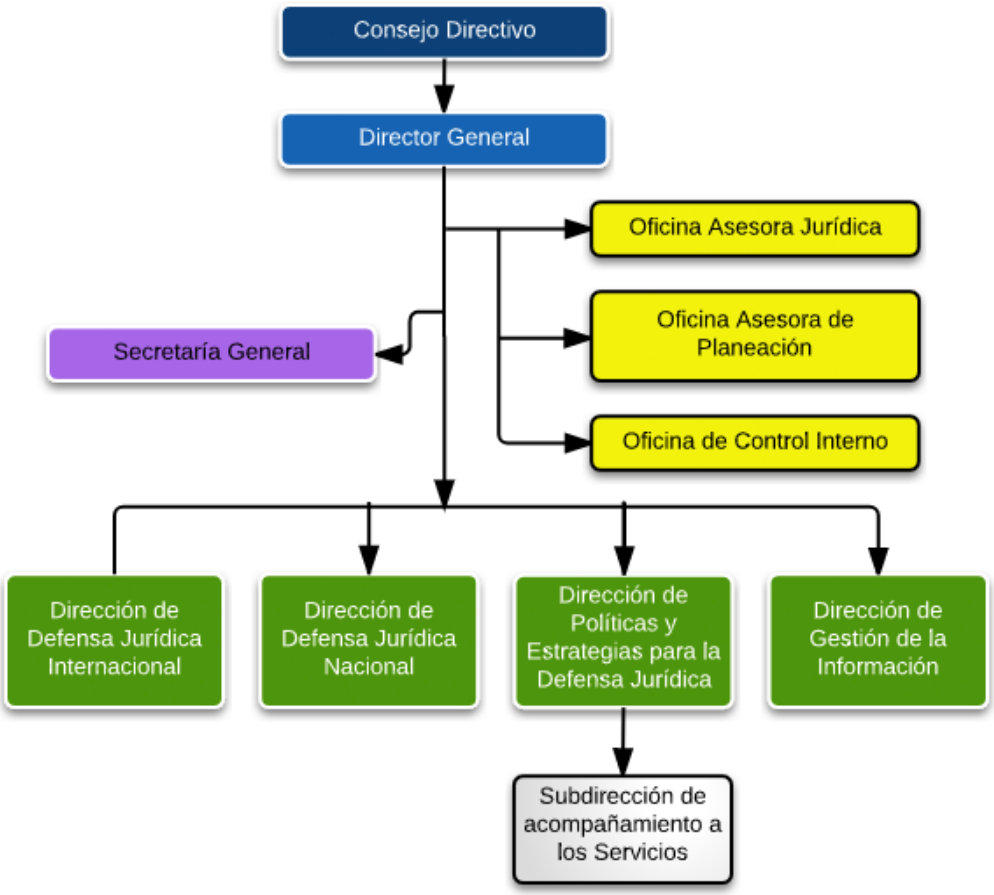


Figura 1 Organigrama de la ANDJE (Agencia Nacional de la Defensa jurídica de Estado, 2017)

4.1.2. Dirección de Gestión de la Información

Dentro de esta organización se destaca la labor de la Dirección de Gestión de la Información cuyas funciones enumeran a continuación:

- Construye, administra y garantiza la sostenibilidad del Sistema Único de Gestión de Información Litigiosa, el cual administra los procesos para identificar variables y tendencias que detecten de manera temprana y efectiva conflictividad asociada a la generación del Daño Antijuridico, así como proveer a las entidades de un sistema integral que les permita la óptima y oportuna gestión del ciclo de defensa jurídico.

- Construye, en conjunto con el MHCP, una herramienta que permita calcular el pasivo contingente de la Nación, lo cual contribuirá con el propósito de lograr una adecuada gestión de riesgos fiscales.
- Diseña mecanismos que garanticen la calidad y confiabilidad de la información.
- Diseña indicadores e instrumentos que permitan monitorear, hacer seguimiento y evaluar la gestión del ciclo de defensa jurídica del Estado. (Agencia Nacional de la Defensa jurídica de Estado, 2017)

4.1.3. Sistema Único de Gestión e Información Litigiosa del Estado (eKogui)

En el año 2012 la Agencia Nacional de Defensa Jurídica del Estado, y en particular la Dirección de Gestión de la Información, recibió del Ministerio de Justicia y del Derecho el sistema de información conocido como LITIGOB el cual para ese momento permitía la incorporación de información mínima sobre las características de los procesos judiciales.

A partir de este momento la Dirección de Gestión de la Información dio inicio a la estructuración del proyecto de evolución y transformación del sistema de información en un horizonte de ocho años. Dicho proyecto tiene como finalidad fortalecer el Sistema Único de Gestión e Información litigiosa del Estado, tanto en el aspecto técnico como funcional. En el desarrollo de este proyecto se destaca el desarrollo de una primera fase que tenía como objetivo el diseño de nuevos Front-end así como el cambio conceptual respecto a la información a capturar, facilitando la usabilidad por parte de los usuarios. Esta primera etapa culminó con el relanzamiento del Sistema Único de Gestión e Información litigiosa del Estado bajo el nombre de eKogui (anteriormente LITIGOB) en junio de 2015.

En enero de 2016 se dio inicio a la segunda fase del proyecto que busca la identificación, diseño y construcción de herramientas requeridas para la recolección

y adecuación de la información requerida por la Agencia para el cumplimiento de su misión, reflejada en el desarrollo de nuevas funcionalidades dentro del sistema.

En el contexto anteriormente descrito, el Sistema único de Gestión de Información (eKogui) contempla dos objetivos principales:

- Permitir a las entidades del Estado registrar y gestionar la información de los procesos judiciales que cursan en su contra
- Generar conocimiento a partir de la información registrada con el fin de fortalecer las políticas y estrategias de prevención del daño jurídico y ejercer una defensa judicial más eficientes

En este sentido el sistema está diseñado para asegurar la disponibilidad de la información con estándares de calidad y oportunidad, contar con funcionalidades que permitan determinar con certeza el número de procesos y el valor monetario de los procesos que cursan en contra de la Nación, las causales más frecuentes de la litigiosidad, el tipo de procesos más frecuentes, las entidades en las que se concentran los procesos más costosos y el riesgo o probabilidad de perder el caso entre otros factores; de modo que permita a la agencia la toma de decisiones estratégicas que lleven a mejorar la defensa del Estado.

- **EKOGUI 1.0** que es utilizada para gestionar los procesos judiciales y extrajudiciales de las entidades, ayuda a la generación de indicadores para la prevención del daño antijuridico, a hacer seguimiento y evaluar la gestión del ciclo de defensa jurídica del Estado.
- **EKOGUI 2.0** que actualmente permite al MHCP calcular el pasivo contingente de la nación, gestionar los usuarios y entidades registradas en el sistema, generar información jurisprudencial para genera estadísticas e indicadores en materia de gestión de las oficinas jurídicas de las entidades del estado colombiano.

Adicional a EKOGUI se encuentra **La Comunidad Jurídica del Conocimiento** una herramienta creada con ayuda de la dirección de defensa jurídica para la consulta de las tendencias jurisprudenciales y documentos especializados en temas de alta

relevancia para el Estado, lo cual permitirá fortalecer la defensa jurídica de las entidades, esta herramienta también permite a los defensores o abogados, generar y compartir información, participar en foros entre colegas, crear grupos, contactar con otros a través de un chat, acceder a cursos virtuales y capacitaciones virtuales. (Agencia nacional de defensa jurídica del estado, 2013)

4.2. Conceptos sobre pruebas de software

Las pruebas de software se crearon con la necesidad de mejorar la calidad de un sistema, convirtiéndose así en parte esencial del desarrollo de software. A continuación, se detallan algunos conceptos de pruebas de software.

4.2.1. Principios de pruebas

Para el proceso de planeación y ejecución de pruebas es necesario tener en cuenta los 7 principios de pruebas que han sido sugeridos por años como soluciones y directrices generales de pruebas aunque algunos otros autores sugieran más o menos principios, se trabajará como los describe la guía “Certified Tester Foundation Level Syllabus” (International Software Testing Qualifications Board, 2017) con los 7 principios de pruebas los cuales se citan a continuación:

Las pruebas muestran la presencia de defectos

Las pruebas muestran la presencia de defectos mas no muestran la ausencia de ellos. Las pruebas reducen la posibilidad de que queden errores en el software, pero aun si no se descubre ningún fallo en el sistema no significa que no lo exista. (Meyer, 2008)

Las pruebas exhaustivas son imposibles

Realizar las pruebas de todo el sistema, realizando todos los caminos posibles no es factible excepto en casos triviales, para los casos complejos no es recomendable escribir tantos casos de prueba como sea posible. En vez de realizar pruebas exhaustivas se sugiere realizar un plan de pruebas teniendo en cuenta los riegos y prioridades para enfocar así los esfuerzos necesarios para pruebas y no desgastar los recursos. (Myers, 2004)

Pruebas tempranas

Para encontrar errores lo antes posible, se recomienda realizar las pruebas del sistema en cada parte del ciclo de vida del desarrollo los más pronto que sea posible, incorporando las pruebas inclusive en el análisis y diseño del sistema para así tomar las medidas necesarias para su pronta solución.

Agrupación de defectos

Los defectos suelen encontrarse en grupo es probable que donde se encontraron más incidentes haya más, es por esto por lo que se debe enfocar el esfuerzo proporcionalmente a los errores que se vayan presentando, en los lugares que se encontraron más errores dedicar más esfuerzos ya que son módulos que se encuentran propensos a errores.

Paradoja del pesticida

Se debe ir cambiando los casos de pruebas a medida que se van avanzando en ellas ya que según esta paradoja si se realizan repetidamente las mismas pruebas no se van a encontrar más defectos. Para esto es necesario revisar periódicamente y escribir nuevos casos de prueba en diferentes partes del software con el fin de

encontrar otros defectos. (Jústiz-Núñez, y otros, 2014). Es necesario exploran toda la creatividad para crear nuevos casos

Las pruebas dependen del contexto

Las pruebas deben hacer diferentes en diferentes contextos todo depende de la necesidad que tiene, se debe verificar que tipo de recurso o qué tipo de herramienta o sistema utilizar para realizar la prueba entendiendo que función tiene la aplicación si es de internet probar con los navegadores más utilizados si es móvil probar en diferentes herramientas.

La ausencia de errores son una falacia

Encontrar y corregir errores no garantiza que el sistema va a satisfacer completamente todas las necesidades y expectativas del usuario. Para Myers un error está claramente presente si un programa no hace lo que se supone que debe hacer, pero los errores también están presentes si un programa hace lo que no debe hacer. Es por esto por lo que él propone realizar los casos de prueba para las condiciones de entrada que son inválidas e inesperadas. (Myers, 2004)

4.2.2. Tipos de pruebas

Caja negra

Son pruebas que se hacen al sistema sin tener un conocimiento específico del trabajo interno del sistema, sin tener acceso al código y ni a la arquitectura.

Caja blanca:

La prueba de la caja blanca es conocida como la prueba de caja transparente, se refiere a probar todos los casos con conocimiento pleno del código y arquitectura

Caja gris:

Se tratan de pruebas que se hacen cuando el tester al menos algún conocimiento de los componentes internos del sistema.

Para los diferentes ciclos de vida de desarrollo de software existe un rol responsable de realizar la validación como se muestra en la siguiente tabla:

Tipo de prueba	Rol responsable
Pruebas unitarias	Desarrollador
Pruebas integrales	Diseñador
Pruebas de validación	Analista de requisitos
Pruebas funcionales	Tester
Pruebas de regresión	Tester
Pruebas de carga	Tester/Arquitecto/administrador de bases de datos
Pruebas de rendimiento	Tester/Arquitecto
Pruebas de seguridad	Tester/Arquitecto

Tabla 1 rol responsable para pruebas

Como se muestra en la tabla anterior podemos definir:

Pruebas unitarias

Son pruebas que se realizan al código fuente como se muestra en la Tabla 1 rol responsable para pruebas, es responsabilidad del desarrollador y se concentran en cada componente, clase u objeto del desarrollo. Las pruebas unitarias están

limitadas ya que solo enfocan su esfuerzo de verificación en la parte más pequeña de diseño de software, se examinan con una estructura de datos locales verificando que los datos se almacenen correctamente y que la ejecución del algoritmo por todas sus rutas este correcta. (Pressman & Maxim, 2015)

Pruebas integrales

Son las pruebas que se hacen a todos los componentes a una parte del software verificando la comunicación entre los diferentes compontes, Se hacen probando los escenarios más destacados entre los casos de uso con un flujo rápido verificando que la interacción entre los componentes funcione, las evidencias se escriben o se pegan en un formato Word, donde se escribe que flujos y el caso de uso impactan.

Pruebas funcionales

Son pruebas que se realizan a la aplicación de acuerdo con las funcionalidades previamente diseñadas para desarrollo, se centra en probar que funcione, se realizan con los escenarios descritos en los casos de prueba y se diligencian las evidencias en el mismo.

Pruebas de regresión

Se realiza a partir de un cambio de código en la aplicación ya sea por un control de cambio o por un ajuste a un incidente, se toma como referencia las pruebas realizadas en las pruebas funcionales o de integración a discreción del tester.

Pruebas de carga

Es una prueba técnica enfocada al desempeño del sistema cuando este se somete a altas cargas y altos volúmenes transaccionales durante su la ejecución del sistema en un ambiente de producción. (Pressman & Maxim, 2015)

Pruebas de rendimiento

Estas pruebas técnicas son realizadas para verifica la calidad del producto con respecto a la disponibilidad midiendo la velocidad, escalabilidad y estabilidad.

Pruebas de estrés

Las pruebas de estrés son pruebas técnicas que también pueden incluir pruebas de rendimiento y de carga mientras el sistema se somete a un nivel estresante es decir bajos niveles de memoria, procesamiento, espacio insuficiente o fallas en el servidor. (Techtarget, 2017)

Pruebas de seguridad

Se hacen para verificar si la aplicación está protegida en la web ya que el robo de información y el acceso no autorizado son problemas más comunes y a continuación se presentan algunas de las técnicas para verificar el nivel de seguridad del sistema como Inyección, Autenticación rota y gestión de sesiones, Cross-Site Scripting (XSS), Insecure Direct Object References, falsa configuración de seguridad, sensible a la exposición de datos, control de acceso a nivel de función ausente, Cross-Site Request Forgery (CSRF), uso de componentes con vulnerabilidades conocidas, redirecciones y reenvíos no validados. (Allen, 2012)

4.2.3. Metodologías de pruebas existentes

Test Driven Development (TDD)

Esta práctica esta basa en el desarrollo orientado a pruebas y produce un código más simple, más cohesivo y menos acoplado que el creado por las formas tradicionales. Su uso está creciendo en muchos contextos de desarrollo y está asociado a una programación extrema.

Desarrollo impulsado por prueba de aceptación (ATDD)

Es un enfoque de descubrimiento de requisitos basado en actividades de colaboración en equipo. En este caso, las pruebas son creadas por el cliente, el desarrollador y el probador. Esta estrategia se llama tríada y se ejecuta antes de la aplicación de requisitos.

Desarrollo controlado por prueba de detección (STDD), conocido como Desarrollo impulsado por comportamiento (BDD)

Es una práctica que comunica todos los requisitos a todas las partes interesadas a través de pruebas. También se conoce como prueba de cliente.

Exploration tests

Es una prueba de software donde los probadores pueden interactuar con el sistema de cualquier manera y usar la información suministrada con el objetivo de explorar todas las características del sistema sin restricciones.

Pruebas de software de automatización

Significa automatizar las actividades de prueba de software, incluido el desarrollo y la ejecución de los scripts de prueba, verificación de requisitos de prueba y uso de herramientas de prueba. Una de las principales razones para automatizar las pruebas es disminuir el tiempo utilizado manualmente, aumentar la eficiencia al repetir la funcionalidad del sistema, especialmente las pruebas de regresión, donde los casos de prueba se ejecutan de forma iterativa e incremental después de los cambios realizados en el software.

4.3. Análisis comparativo de herramientas para la gestión y automatización de pruebas

En la actualidad con la velocidad con la que ahora se desarrolla y con las nuevas metodologías agiles, existen diversas herramientas que ayudan a diferentes procesos como lo son los siguientes:

- Aplicaciones y programas para la verificación de incidentes y gestionar la calidad del producto como: Jira, Mantis, Zoho Project, Basecamp, Assembla, Gemini, GitHub, Redmine, Bugzilla, VersionOne.
- Para el diseño del sistema donde se pueden realizar casos de prueba: Enterprise architect, CaseTracking, Testlink, Visual Studio Test Proffessional, Hewlett Packard Quality Center (HP QC).
- Para automatización de pruebas de software en aplicaciones web: Selenium, robot framework, Watir, Rational Functional Tester, Ranorex, Sahi, Selenide. HP QuickTest Professional.
- Para la automatización de pruebas de carga rendimiento y estrés existen herramientas como: JMeter, Wireshark, Beyond Compare.

Para las necesidades identificadas que se quieren mejorar temas como los son la agilidad en la ejecución de pruebas y el archivo de la documentación de las pruebas por esta razón se estudiaran diferentes herramientas nombradas anteriormente para mejorar el proceso.

4.3.1. Estudio herramientas de gestión de pruebas

Criterios de selección

Para la documentación de resultados de pruebas se requiere una herramienta que guarde los documentos de resultados de pruebas y que permitan sacar reportes de los mismos

Criterios de evaluación

Facilidad de instalación: No necesitar de muchos permisos, complementos o mucho conocimiento para realizar la instalación de la herramienta.

Facilidad de uso: Que sea fácil de utilizar, que no se necesiten conocimientos complejos para poder utilizarla.

Facilidad de informes y reportes: Que puedan utilizarse diferentes reportes con diferentes filtros.

Campos parametrizables: Poder parametrizar lo campos, cambiar o agrega campos.

Valoración de los criterios

De acuerdo con la información anteriormente descrita en los diferentes criterios se propone la siguiente escala para la medición:

Alto: La alternativa cumple con el criterio propuesto (80% - 100%)

Medio: La alternativa cumple con algo del criterio propuesto (40% - 79%)

Bajo: La alternativa no cumple con el criterio propuesto (0 – 39%)

A continuación, estudiaremos las diferentes herramientas para la gestión de pruebas

INFORMACIÓN	
Desarrollador	Sparx Systems
Origen	Australia
Nombre	Enterprise architect
Descripción general	es una herramienta de modelado y diseño visual basada en OMG UML. La plataforma admite: el diseño y la construcción de sistemas de software; modelar procesos comerciales; y modelado de dominios basados en la industria. Las empresas y organizaciones lo usan no solo para modelar la arquitectura de sus sistemas, sino también para procesar la implementación de estos modelos en todo el ciclo de vida de desarrollo de aplicaciones. (Sparx system, 2017)
Funcionalidades principales	• Gestión de requisitos • Modelado de negocios y análisis • Simulación modelos de diagramas de procesos y diagramas de secuencia, entre otros • Desarrollo del sistema • Gestión de pruebas
Precio	$1495 USD
EVALUACIÓN	
Bajo	Facilidad de instalación
Bajo	Facilidad de uso
Medio	Facilidad de informes y reportes
Medio	Campos parametrizables

INFORMACIÓN	
Desarrollador	Francisco Mancardi
Origen	Argentina
Nombre	Testlink
Descripción general	Es una herramienta de gestión de pruebas basada en la web. La aplicación proporciona especificaciones de prueba, planes de prueba y ejecución, informes, especificaciones de requisitos y colabora con conocidos rastreadores de fallas. (Development Team TestLink , 2017)
Funcionalidades principales	• Soporte de gráficos • Soporte de métricas • Creación y ejecución de prueba • Interfaz de usuario fácil de usar Grabación de defectos
Precio	Gratis
EVALUACIÓN	
Medio	Facilidad de instalación
Medio	Facilidad de uso
Bajo	Facilidad de informes y reportes
Bajo	Campos parametrizables

INFORMACIÓN	
Desarrollador	Microsoft
Origen	EE. UU.
Nombre	Visual Studio Test Proffessional
Descripción general	Herramienta para gestionar todas las actividades y equipo de pruebas, desde la planificación hasta la creación, la ejecución y el seguimiento desde una ubicación central, o desde paneles Kanban con características de calidad en línea. (Microsoft, 2017)
Funcionalidades principales	• Administración de Pruebas exploratorias • casos de pruebas

	• Pruebas basadas en explorador • Microsoft Test Manager
Precio	
EVALUACIÓN	
Alto	Facilidad de instalación
Medio	Facilidad de uso
Medio	Facilidad de informes y reportes
Bajo	Campos parametrizables

INFORMACIÓN	
Desarrollador	Hewlett-Packard
Origen	EE. UU.
Nombre	HP Quality Center
Descripción general	HP QC ha sido el software de administración de pruebas más utilizado; tiene todas las características necesarias de muchas maneras. Es una de las herramientas de alta gama que proporciona un seguimiento e informes sólidos. ALM también se puede conectar con un sistema de correo electrónico y enviar un correo electrónico para cualquier cambio a los miembros del equipo deseados.
Funcionalidades principales	• Planeación y gestión de testers • Integraciones predefinidas • Colaboración entre desarrolladores • Informes y visualización gráfica
Precio	$1400 USD
EVALUACIÓN	
Alto	Facilidad de instalación
Medio	Facilidad de uso
Medio	Facilidad de informes y reportes
Bajo	Campos parametrizables

Características	Enterprise architect	Testlink	Visual Studio Test Proffessional	HP Quality Center
Facilidad de instalación				
Facilidad de uso				
Facilidad de informes y reportes				
Campos parametrizables				
Costos	$1495 USD	Gratis	Gratis	$1400 USD

Alto	
Medio	
Bajo	

Conclusión

Ninguna de las herramientas anteriores cumple totalmente con las expectativas de para poder guardar los reportes de los casos de prueba, donde se puedan parametrizar diferentes campos importantes al momento de realizar las pruebas y que permita realizar reportes fáciles y exportables a Excel de forma estadística.

4.3.2. Estudio de herramientas automatización de pruebas

Criterios de selección

Para realizar pruebas funcionales se requiere buscar que herramientas existentes pueden ayudar a agilizar el proceso de pruebas de software en el desarrollo de aplicaciones web.

Criterios de evaluación

Para la evaluación de las herramientas a estudiar se tendrán en cuenta los siguientes criterios de evaluación:

Sistemas Operativos soportados: La herramienta permita ser utilizada en diferentes sistemas operativos.

Lenguajes: Soporte en diferentes lenguajes como Java, C#, Ruby, Groovy, Perl, PHP y Python.

Habilidades en programación para realización de pruebas robustas: Poder realizar pruebas robustas con diferentes componentes de datos y reglas para pruebas.

Curvas de aprendizaje: Se necesite poca experiencia para manejar la herramienta y contar con documentación sobre la misma para la grabación de pruebas.

Facilidad de instalación: No necesitar permisos, complementos o mucho conocimiento para realizar la instalación de la herramienta, solo una secuencia guiada de clics por la aplicación.

Grabar y reproducir: Tener facilidad en realizar el grabado de pruebas o la ejecución de las mismas para que cualquier miembro del equipo pueda reutilizarla.

Navegadores: Soporte diferentes navegadores para la ejecución d la prueba, que pueda ser utilizado en Firefox, Chrome, Safari etc.

Soporte de aplicaciones: Para el caso particular del sistema se necesita que soporte aplicaciones web.

Precio: Entre menos costo será más favorable para el desarrollo del proyecto.

Valoración de los criterios

De acuerdo con la información anteriormente descrita en los diferentes criterios se propone la siguiente escala para la medición:

Alto: La alternativa cumple con el criterio propuesto (80% - 100%)

Medio: La alternativa cumple con algo del criterio propuesto (40% - 79%)

Bajo: La alternativa no cumple con el criterio propuesto (0 – 39%)

A continuación, estudiaremos las diferentes herramientas para automatización de software

INFORMACIÓN	
Desarrollador	Jason Huggins
Origen	EE. UU.
Nombre	Selenium
Descripción general	Es una herramienta de código abierto, basada en aplicaciones web que permite grabar automáticamente pruebas y modificarlas de acuerdo con sus objetivos, esta herramienta además puede ser escrita en diversos lenguajes de programación como: Java, C#, Ruby, Groovy, Perl, PHP y Python y pueden ser ejecutadas en diferentes sistemas operativos como: Windows, Linux y OSX. Selenium automatiza los navegadores. Principalmente, es para automatizar aplicaciones web con fines de prueba, pero ciertamente no se limita a eso. Selenium cuenta con el apoyo de algunos de los proveedores de navegadores más grandes que han tomado (o están tomando) medidas para hacer que Selenium sea una parte nativa de su navegador. También es la tecnología principal en innumerables otras herramientas de automatización del navegador, API y marcos. (Selenium, 2017)
Funcionalidades principales	• Crear secuencias de comandos rápidas de reproducción de errores

	• Crear scripts para ayudar en pruebas exploratorias asistidas por automatización • Crear robustas suites y pruebas de automatización de regresión basadas en navegador • Escalar y distribuir scripts en muchos entornos
Precio	Gratis
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Alto	Lenguajes
Alto	Habilidades en programación para realización de pruebas robustas
Medio	Curvas de aprendizaje
Medio	Facilidad de instalación
Alto	Grabar y reproducir
Alto	Navegadores
Aplicaciones web	Soporte de aplicaciones

Tabla 2 Información de Selenium

INFORMACIÓN	
Desarrolladores	Nokia Networks, Pekka Klärck Janne Härkönen et al.
Origen	Finlandia
Nombre	Robot Framework
Descripción general	Es un framework para la automatización de pruebas de aceptación y desarrollo basado en pruebas de aceptación (ATDD). Tiene una sintaxis de datos de prueba fácil de usar y utiliza el enfoque de prueba basado en palabras clave. Sus capacidades de prueba pueden ampliarse mediante bibliotecas de prueba implementadas con Python o Java, y los usuarios pueden crear nuevas palabras clave de nivel superior a partir de las existentes utilizando la misma sintaxis que se utiliza para crear casos de prueba. El marco central se implementa mediante Python y también se ejecuta en Jython (JVM) y IronPython (.NET). (Introduction: Robotframework, 2017)

Funcionalidades principales	• Crear casos de prueba automatizadas.
Precio	$510 USD
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Medio	Lenguajes
Alto	Habilidades en programación para realización de pruebas robustas
Alto	Curvas de aprendizaje
Bajo	Facilidad de instalación
Bajo	Facilidad para grabar y reproducir
Alto	Navegadores
Aplicaciones web	Soporte de aplicaciones

Tabla 3 Información de Robot Framework

INFORMACIÓN	
Desarrolladores	Ranorex GmbH
Origen	Austria
Nombre	Ranorex
Descripción general	Es una aplicación para realizar pruebas a programas de escritorio, web o aplicaciones móviles, con módulos de código reutilizable, de fácil utilización. Los programadores profesionales pueden usar una API para C # y VB.NET para mejorar sus suites de prueba y grabaciones. (Ranorex GmbH, 2017)
Funcionalidades principales	• Pruebas integradas de aplicaciones de escritorio, web y móviles, y soporte para muchos controles y frameworks de terceros
Precio	$ 890 USD a $ 2,990 USD.
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Medio	Lenguajes

Bajo	Habilidades en programación para realización de pruebas robustas
Medio	Curvas de aprendizaje
Alto	Facilidad de instalación
Alto	Facilidad para grabar y reproducir
Alto	Navegadores
Aplicaciones web, de escritorio y móviles	Soporte de aplicaciones

Tabla 4 Información de Ranorex

INFORMACIÓN	
Desarrolladores	V. Narayan Raman
Origen	India
Nombre	Sahi
Descripción general	Es especialmente utilizado para pruebas entre navegadores / navegadores múltiples de aplicaciones web 2.0 complejas con gran cantidad de AJAX y contenido dinámico. Desarrollado para probar aplicaciones web, Sahi ofrece tanto una versión de código abierto como una profesional. Sahi funciona como un servidor proxy que le permite usarlo dentro de un navegador. Desde el panel Sahi, puede iniciar el navegador que desea probar. La grabación y captura de la interacción que desea realizar contra su aplicación. Sahi facilita el inicio de la automatización de pruebas de aplicaciones HTML simples. (Sahi, 2017)
Funcionalidades principales	• Realiza registro e informes de pruebas • Almacena los sets de datos • Reproducción de pruebas • Soporte en diferentes browsers • Envía correo electrónico de errores
Precio	$ 695 USD.
EVALUACIÓN	
Alto	Sistemas Operativos soportados

Bajo	Lenguajes
Bajo	Habilidades en programación para realización de pruebas robustas
Media	Curvas de aprendizaje
Media	Facilidad de instalación
Alto	Facilidad para grabar y reproducir
Alto	Navegadores
Aplicaciones web	Soporte de aplicaciones

Tabla 5 Información de Sahi

INFORMACIÓN	
Desarrolladores	Bret Pettichord, Paul Rogers y Jarmo Pertman
Origen	EE. UU.
Nombre	Watir
Descripción general	Una biblioteca de código abierto de Ruby para automatizar las pruebas. Watir interactúa con un navegador de la misma manera que las personas: haciendo clic en enlaces, completando formularios y validando texto. Alimentado por selenium. Watir permite automatizar Internet Explorer, Mozilla Firefox, Google Chrome y Safari (Watir, 2017)
Funcionalidades principales	• Escribir scripts de casos de prueba para la ejecución automatizada. • Interpretar todos los elementos de HTML de la página de manera que pueden ser externamente interpretados e incluso manipulados.
Precio	Gratis
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Medio	Lenguajes
Bajo	Habilidades en programación para realización de pruebas robustas
Medio	Curvas de aprendizaje

Medio	Facilidad de instalación
Bajo	Facilidad para grabar y reproducir
Alto	Navegadores
Aplicaciones web	Soporte de aplicaciones

Tabla 6 Información de Watir

INFORMACIÓN	
Desarrollador	IBM
Origen	EE. UU.
Nombre	Rational Functional Tester
Descripción general	Es una herramienta de pruebas de regresión y pruebas funcionales automatizadas. Este software proporciona capacidades de prueba automatizadas para pruebas funcionales, de regresión, GUI y basadas en datos. Es compatible con una amplia gama de aplicaciones, tales como basadas en web, .Net, Java, Siebel, SAP, aplicaciones basadas en emuladores de terminal, PowerBuilder, Ajax, Adobe Flex, Dojo Toolkit, GEF, documentos Adobe PDF, zSeries, iSeries y pSeries (IBM, 2017)
Funcionalidades principales	• Prueba de regresión y pruebas funcionales automatizadas.
Precio	$ 3,500 USD a $14,300 USD
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Medio	Lenguajes
Medio	Habilidades en programación para realización de pruebas robustas
Medio	Curvas de aprendizaje
Alto	Facilidad de instalación
Bajo	Facilidad para grabar y reproducir
Alto	Navegadores

Aplicaciones web	Soporte de aplicaciones

Tabla 7 Información de Rational Functional Tester

INFORMACIÓN	
Desarrollador	IBM
Origen	EE. UU.
Nombre	Selenide
Descripción general	Selenide es un framework para la automatización de pruebas con tecnología de Selenium WebDriver que ofrece un API conciso y fluido para pruebas Compatibilidad con Ajax para pruebas estables Selectores potentes, configuración simple. (Selenide, 2017)
Funcionalidades principales	• Prueba de regresión y pruebas funcionales automatizadas.
Precio	Gratis
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Medio	Lenguajes
Medio	Habilidades en programación para realización de pruebas robustas
Alto	Curvas de aprendizaje
Bajo	Facilidad de instalación
Bajo	Facilidad para grabar y reproducir
Alto	Navegadores
Aplicaciones web	Soporte de aplicaciones

Tabla 8 Información de Selenide

INFORMACIÓN	
Desarrollador	Robert C. Martin, Micah D. Martin, Patrick Wilson-Welsh & FitNesse contributors
Origen	EE. UU.

Nombre	FitNesse
Descripción general	Es un IDE que utiliza un servidor Wiki para interactuar con FIT (Framework for Integrated Test). FIT es una herramienta open-source diseñada para automatizar pruebas que requiere la colaboración de dos equipos: el equipo funcional (usuarios, perfiles de negocio) y el equipo de desarrollo. Acepta proyectos escritos en Java, Python o C #. (Fitnesse, 2017)
Funcionalidades principales	• Crear pruebas de aceptación
Precio	Gratis
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Medio	Lenguajes
Medio	Habilidades en programación
Medio	Curvas de aprendizaje
Media	Facilidad de instalación
Bajo	Facilidad para grabar y reproducir
Alto	Navegadores
Aplicaciones web	Soporte de aplicaciones

Tabla 9 Información de FitNesse

INFORMACIÓN	
Desarrollador	Tecnología KMS
Origen	EE. UU.
Nombre	Katalon Studio
Descripción general	Katalon Studio revoluciona el uso de frameworks de automatización de pruebas de código abierto. Proporciona plantillas de proyecto para organizar casos de prueba, repositorio de objetos y palabras clave. Totalmente compatible con pruebas Web, Android, iOS y API en todos los sistemas operativos

	Fácil de integrar con Jenkins, GIT, JIRA y qTest con complementos nativos. (Katalon Studio, 2017)
Funcionalidades principales	• Crear pruebas • Ejecutar pruebas • Ver los informes • Mantener pruebas
Precio	Gratis
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Medio	Lenguajes
Bajo	Habilidades en programación para realización de pruebas robustas
Alto	Curvas de aprendizaje
Alto	Facilidad de instalación
Bajo	Facilidad para grabar y reproducir
Alto	Navegadores
Aplicaciones web	Soporte de aplicaciones

Tabla 10 Información de Katalon Studio

INFORMACIÓN	
Desarrollador	Hewlett Packard Enterprise
Origen	EE. UU.
Nombre	HP QuickTest Professional(QTP)
Descripción general	Es una herramienta que consiente en realizar pruebas reales de dispositivos en las principales plataformas móviles (iOS, Android y Windows), principales navegadores (Chrome, Firefox, Internet Explorer y Safari), así como una amplia gama de tecnologías.
Funcionalidades principales	• Admite múltiples navegadores y plataformas • Pruebas distribuidas optimizadas • Solución para múltiples pruebas • Reconocimiento de objetos basado en imágenes

	Flujos de prueba visual (patrón)
Precio	$2400 USD
EVALUACIÓN	
Alto	Sistemas Operativos soportados
Medio	Lenguajes
Bajo	Habilidades en programación para realización de pruebas robustas
Alto	Curvas de aprendizaje
Alto	Facilidad de instalación
Bajo	Facilidad para grabar y reproducir
Alto	Navegadores
Aplicaciones web	Soporte de aplicaciones

Tabla 11 Información de HP QuickTest Professional(QTP)

Características	Selenium	Robot Framework	Ranorex	Sahi	Watir	Rational Functional Tester	Selenide	FitNesse	Katalon Studio	QTP
Sistemas Operativos soportados										
Lenguajes										
Habilidades en programación										
Curvas de aprendizaje										
Facilidad de instalación										
Grabar y reproducir										
Navegadores										
Soporte de aplicaciones	Web	Web, escritorio y móviles	Web	Web	Web	Web	Web	Web	Web	Web
Costos	Gratis	$510 USD	$ 2,990 USD	$695 USD	Gratis	$14,300 USD	Gratis	Gratis	Gratis	$2400 USD

Tabla 12 Cuadro comparativo de herramientas de automatización de pruebas

Alto	

Medio	
Bajo	

Conclusión

De acuerdo el estudio realizado y los resultados obtenidos en la Tabla 12 Se elige utilizar Selenium porque es una herramienta que se adapta a la necesidad del sistema ekogui, no tiene costo alguno y es una aplicación que además de ser de código abierto, basada en aplicaciones web, permite a los usuarios grabar automáticamente sus pruebas y modificarlas de acuerdo con sus objetivos esta herramienta puede ser escrita en diversos lenguajes de programación como: Java, C#, Ruby, Groovy, Perl, PHP y Python y pueden ser ejecutadas en diferentes sistemas operativos como: Windows, Linux y OSX.

4.4. Proceso actual

4.4.1. Tipos de pruebas

De acuerdo con los tipos de pruebas descritos en el marco teórico actualmente en la entidad se llevan a cabo los siguientes tipos de pruebas:

Pruebas funcionales: Son pruebas que se realizan a la aplicación de acuerdo con las funcionalidades previamente diseñadas para desarrollo, se centra en probar que funcione, se realizan con los escenarios descritos en los casos de prueba y se diligencian las evidencias en el mismo.

Pruebas de lineamientos gráficos: Son pruebas incluidas en las funcionales las cuales están enfocadas únicamente a la verificación de textos, logos,

mensajes, menús, tablas, formatos de fechas, tipos de números y letras, botones, etc.

Pruebas integrales: Son las pruebas que se hacen a todos los componentes a una parte del software verificando la comunicación entre los diferentes compontes, Se hacen probando los escenarios más destacados entre los casos de uso con un flujo rápido verificando que la interacción entre los componentes funcione, las evidencias se escriben o se pegan en un formato Word, donde se escribe que flujos y el caso de uso impactan.

Pruebas de regresión: Se realiza a partir de un cambio de código en la aplicación ya sea por un control de cambio o por un ajuste a un incidente, se toma como referencia las pruebas realizadas en las pruebas funcionales o de integración a discreción del tester.

Pruebas de carga y estrés: Pruebas para determinar y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios y/o transacciones que se espera en el ambiente de producción, se graban las pruebas con Jmeter y se corren sobre la Base de datos a través de JDBC.

4.4.2. Artefactos de gestión de pruebas

Para las pruebas funcionales se utilizan

- Casos de uso
- Casos de prueba
- Ejecución de pruebas integrales

4.4.3. Herramientas utilizadas para pruebas

- Jmeter

- Jira
- Enterprise Architect

4.4.4. Proceso actual del ciclo de vida del desarrollo

A continuación, se muestra como es el proceso actual en el ciclo de vida del software EKOGUI el cual tiene una metodología de desarrollo iterativo incremental, en un proceso de evolución constante el cual se describe a continuación en la Figura 2 Diagrama de procesos del proceso actual:

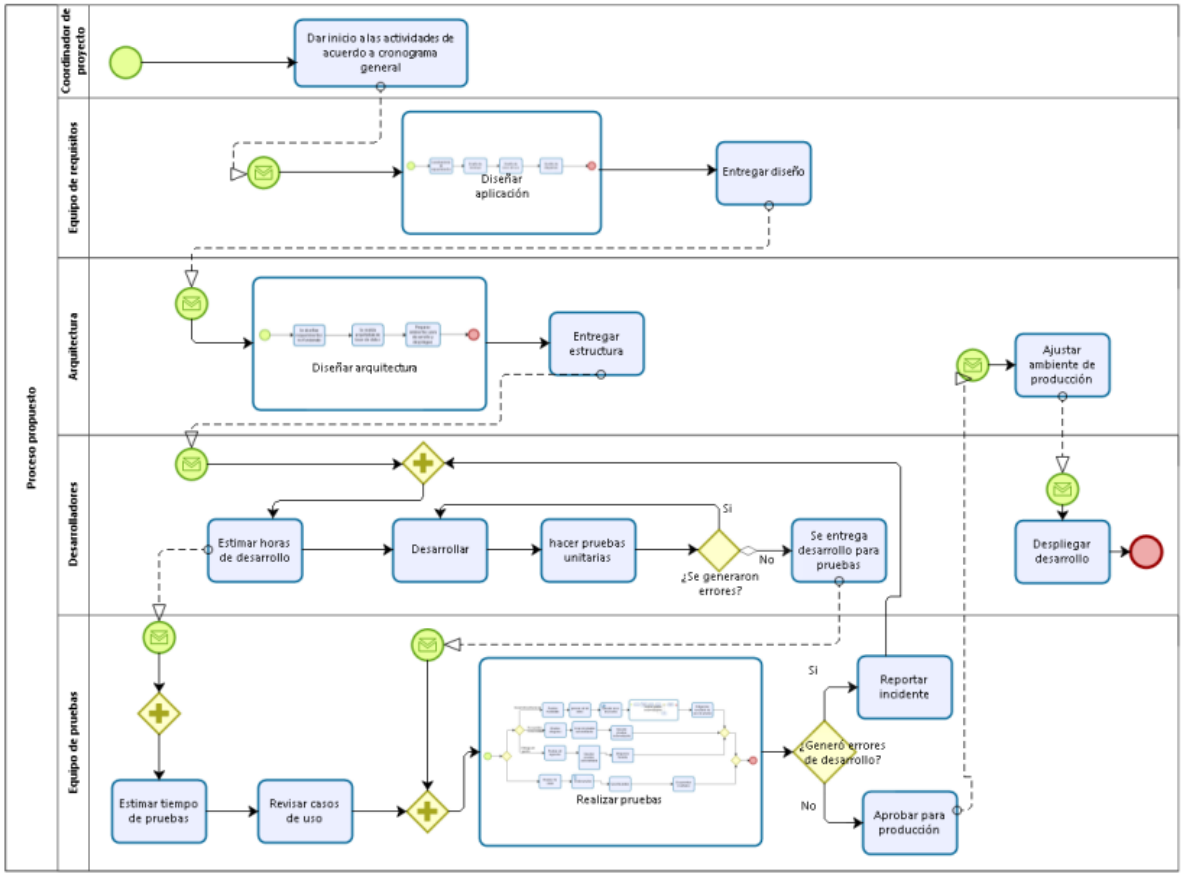


Figura 2 Diagrama de procesos del proceso actual

Subproceso diseñar aplicación

Este es el proceso realizado por el área de requisitos para el desarrollo del proyecto

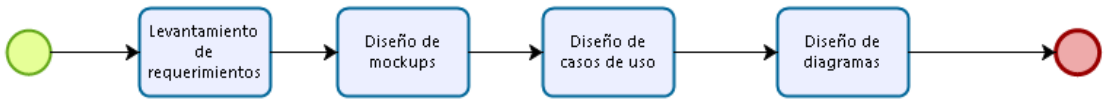


Figura 3 Subproceso de Figura 4 diseñar aplicación

Subproceso diseñar arquitectura

El siguiente es el detalle del proceso realizado por el área de arquitectura en el desarrollo del proyecto:

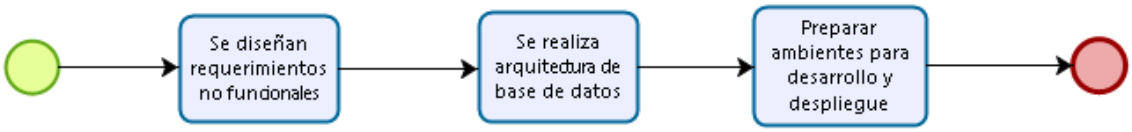


Figura 5 Subproceso de Figura 6 diseñar arquitectura

Subproceso realizar pruebas

A continuación, se presenta el proceso realizado en el área de pruebas donde se ve como se realizan las pruebas funcionales y de regresión y las pruebas de carga:

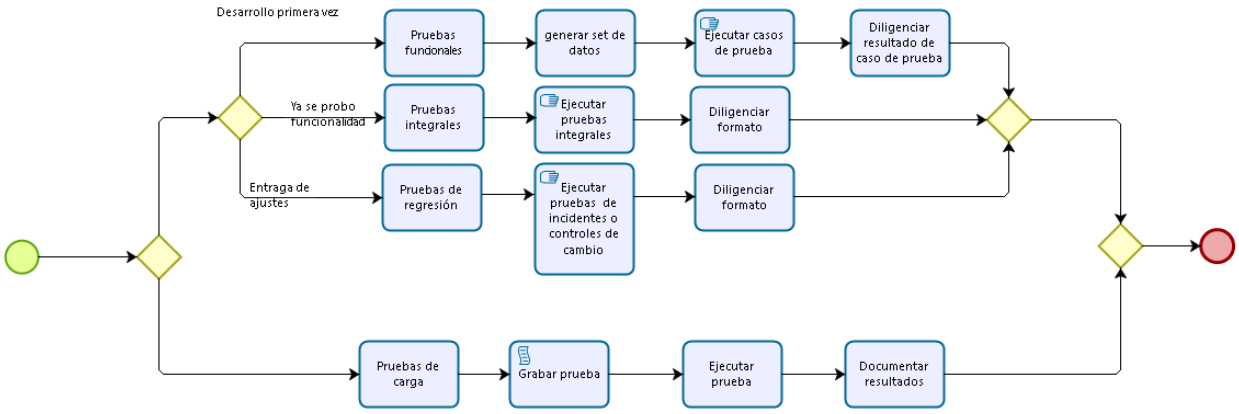


Figura 7 Subproceso de Figura 8 realizar pruebas

5. Propuesta

Después de analizar la información obtenida por la agencia se tomaron las siguientes medias:

5.1. Proceso propuesto del ciclo de vida de desarrollo (para pruebas)

En el proceso solo se modificó el subproceso de la realización de pruebas donde se describe el momento en el cual se realizarán las pruebas automatizadas:

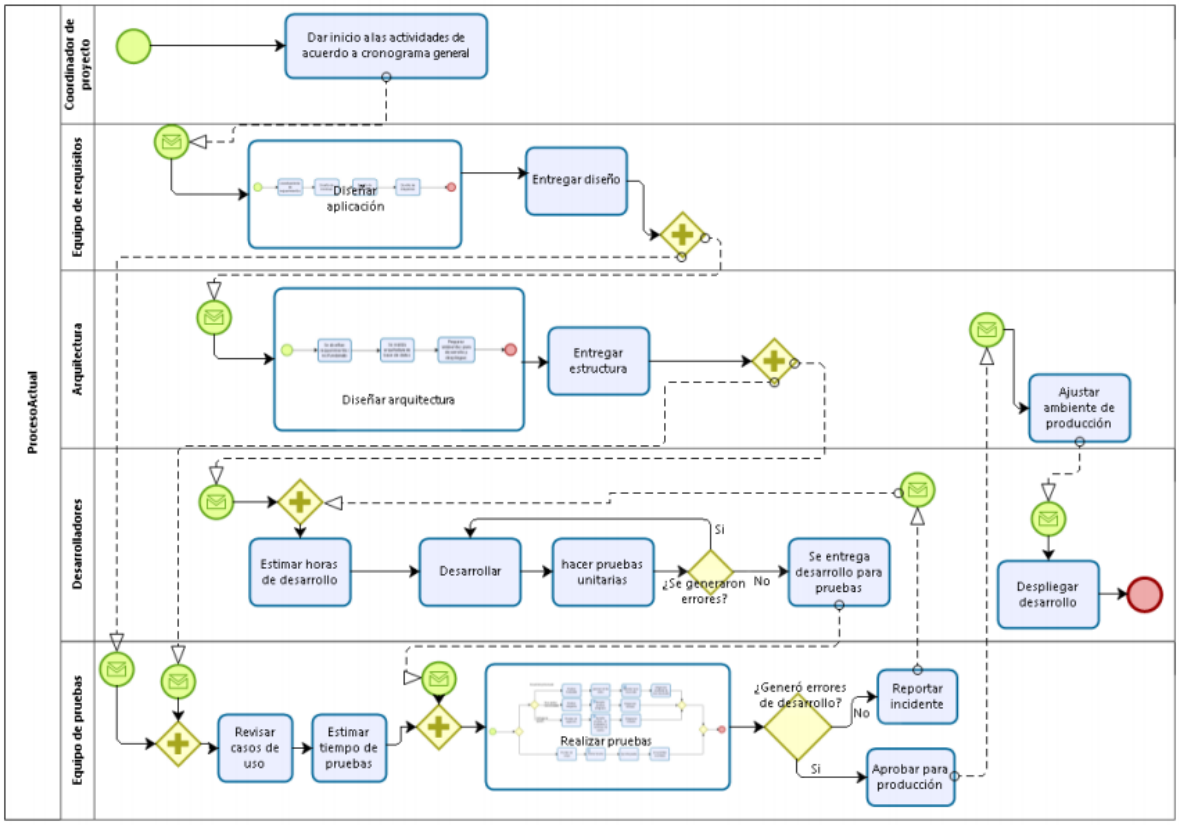


Figura 9 Diagrama de procesos de propuesta

5.1.1. Subproceso diseñar aplicación

Al subproceso de diseño no se le realizo cambios:

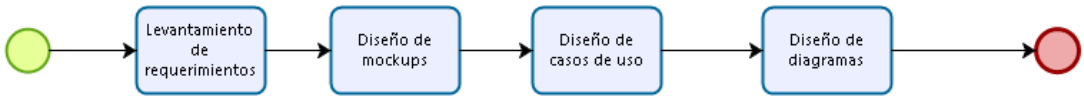


Figura 10 Subproceso de Figura 11 *diseñar aplicación*

5.1.2. Subproceso diseñar arquitectura

Al subproceso de arquitectura no se le realizo cambios:

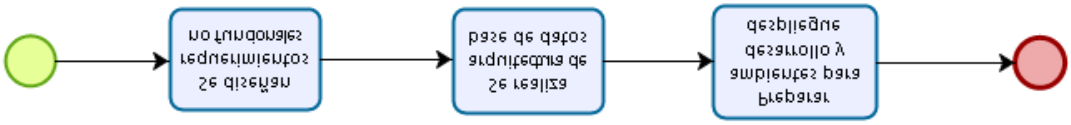


Figura 12 Subproceso de Figura 13 *diseñar arquitectura*

5.1.3. Subproceso realizar pruebas

Al subproceso de diseño realizar pruebas se le agrego el proceso de grabar pruebas al correr las pruebas funcionales y el proceso de ejecutarlas en el momento de las pruebas de regresión.

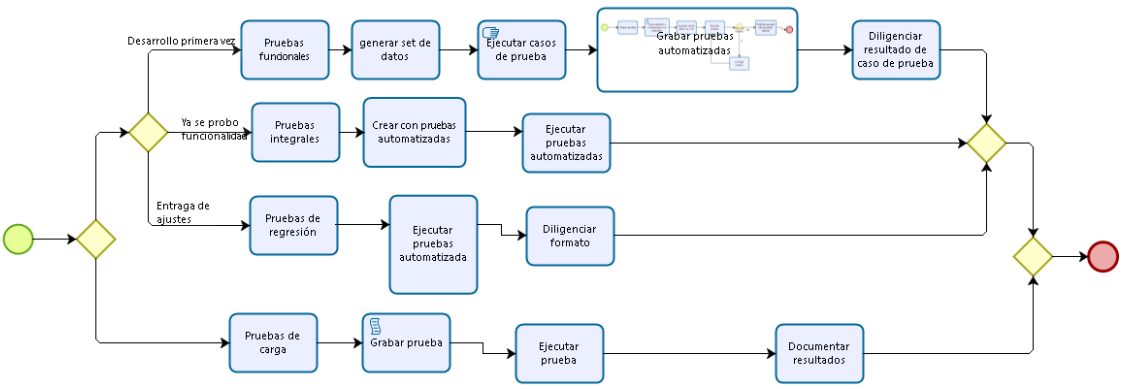


Figura 14 Subproceso de Figura 15 *realizar pruebas*

Subproceso grabar pruebas automatizadas

Para el proceso de automatización de pruebas se construyó un sub proceso para grabar las pruebas en la primera ejecución manual, se necesita grabar las pruebas, corregir errores de script d pruebas, crear un XML para selenium con el set de dato para pruebas, ejecutar la prueba para verificar errores y se propone publicar las pruebas para que cualquier tester en cualquier momento pueda realizarlas.

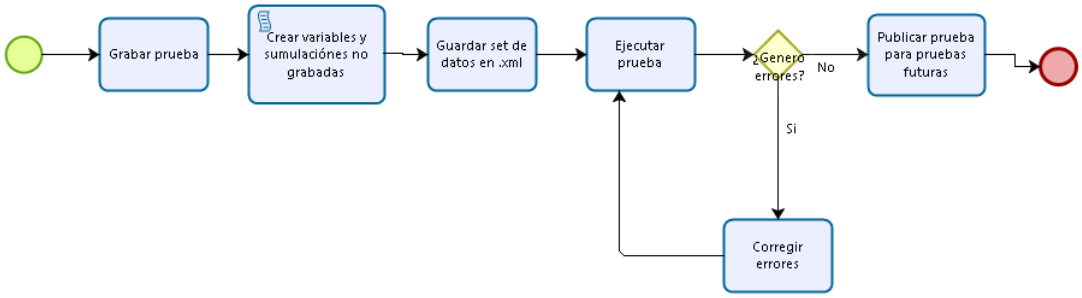


Figura 16 Subproceso de Figura 17 grabar pruebas automatizadas

5.2. Pruebas automatizadas de software

Se elige utilizar selenium como se concluye en el capítulo porque es una herramienta que se adapta a la necesidad del sistema ekogui que es una aplicación web además de ser una herramienta de código abierto, basada en aplicaciones web que permite a los usuarios grabar automáticamente sus pruebas y modificarlas de acuerdo con sus objetivos esta herramienta además puede ser escrita en diversos lenguajes de programación como: Java, C#, Ruby, Groovy, Perl, PHP y Python y pueden ser ejecutadas en diferentes sistemas operativos como: Windows, Linux y OSX. (Selenium, 2017)

Se diseño un prototipo de pruebas con selenium ID para realizar la creación de varios procesos arbitrales, realizando la integración de un archivo XML, con el fin de conocer la herramienta verificar tiempos. (Adjunto prueba)

5.3. Prototipo de gestor de archivo

Se realizo un formulario de reporte de incidentes en PHP con conexión con MySQL para la información a reportar en jira para tener un mayor control con la información que se reporta y poder sacar reportes de la información que se reporta ya que, aunque se maneja jira hay muchos datos que se estaban registrando en un Excel lo cual no permite realizar un reporte adecuado de la información deportada.

5.3.1. Diagramas UML

A continuación, se encuentran diferentes diagramas UML que fueron realizados para el gestor de archivos

Diagrama de secuencia

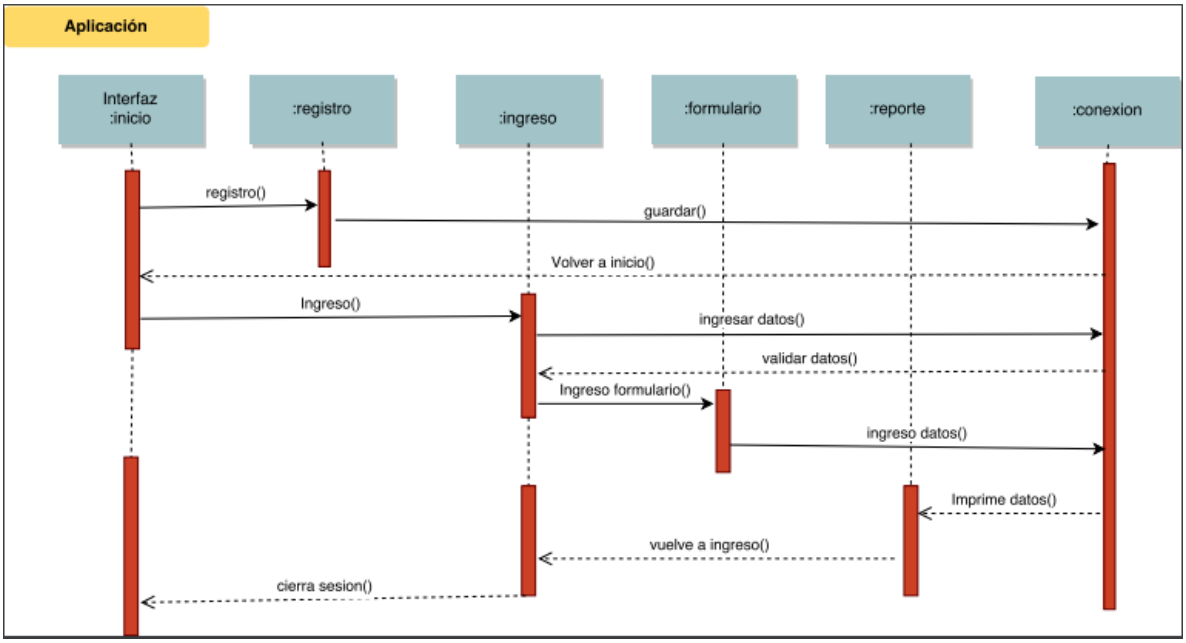


Figura 18 diagrama de secuencia

Diagrama de componentes

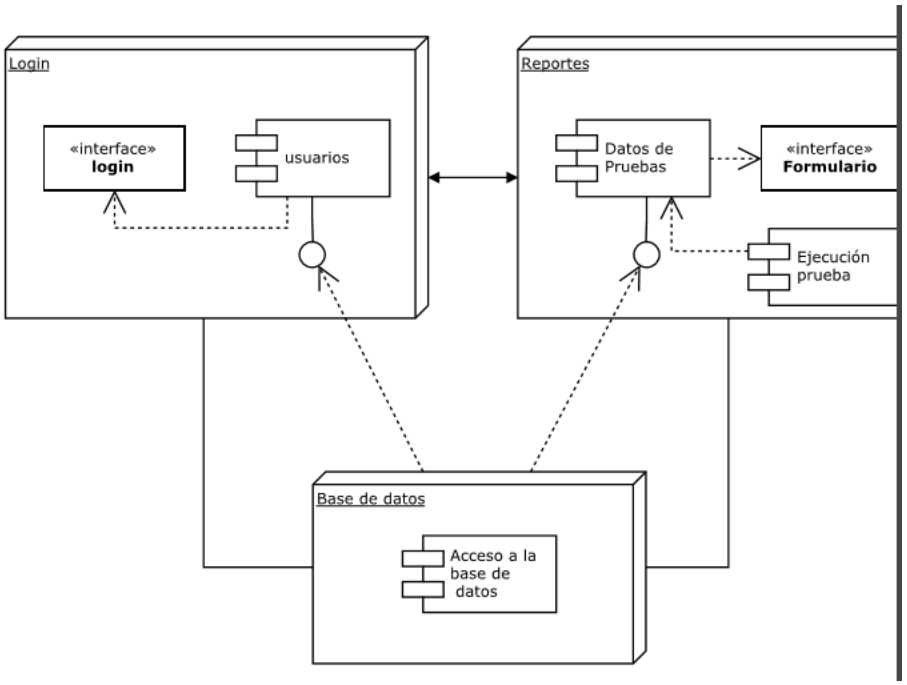


Figura 19 diagrama de componentes

Diagrama de clases

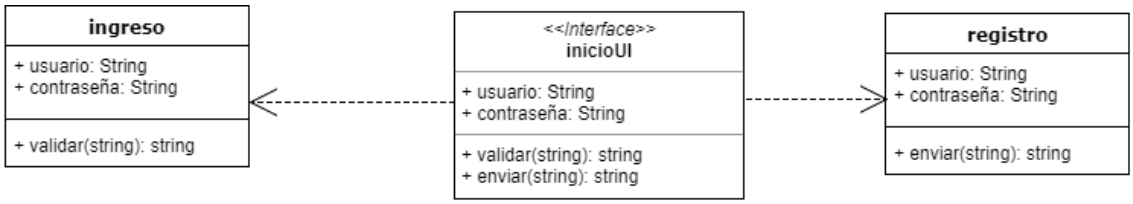


Figura 20 diagrama de clases inicio sesión

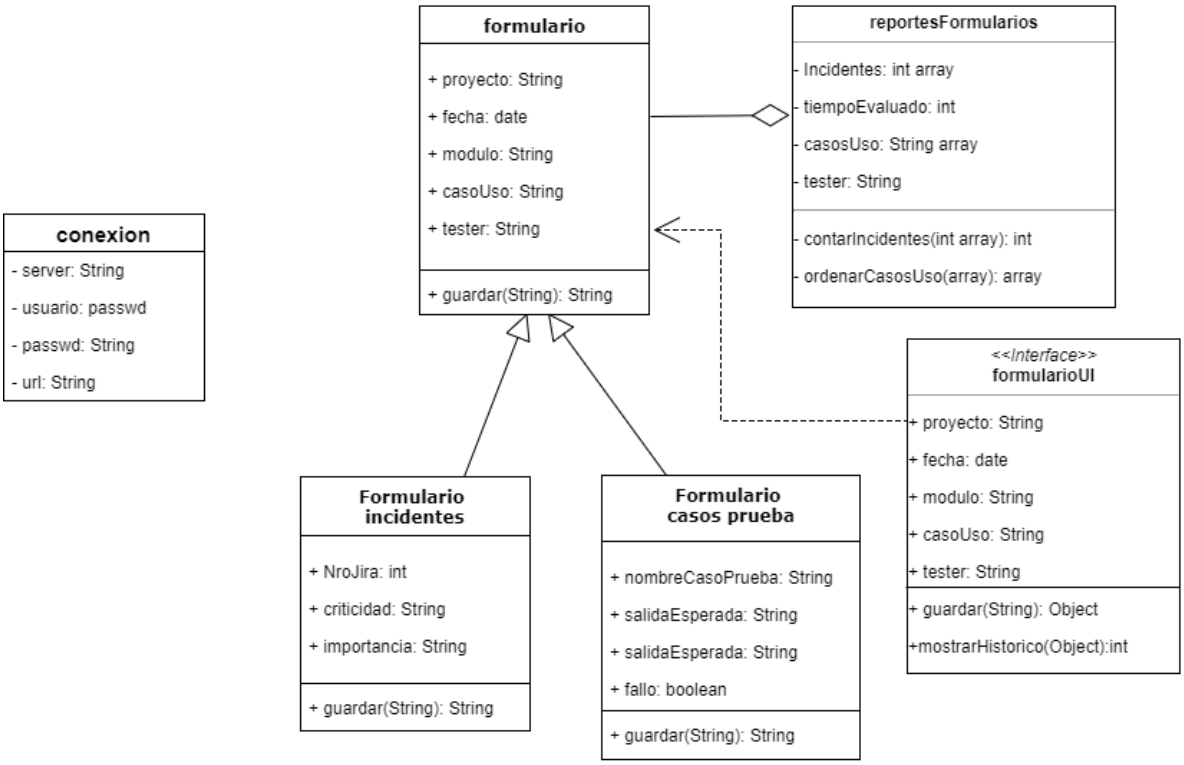


Figura 21 Diagrama de clases formulario y reportes

Tabla de Conversión de diagrama de clases

Clase	Classname + field: type + method(type): type
Interfaz	«interface» **Name**

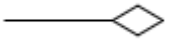
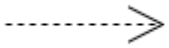
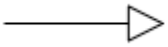
Agregación	
Dependencia	
Generalización	

Diagrama entidad relación

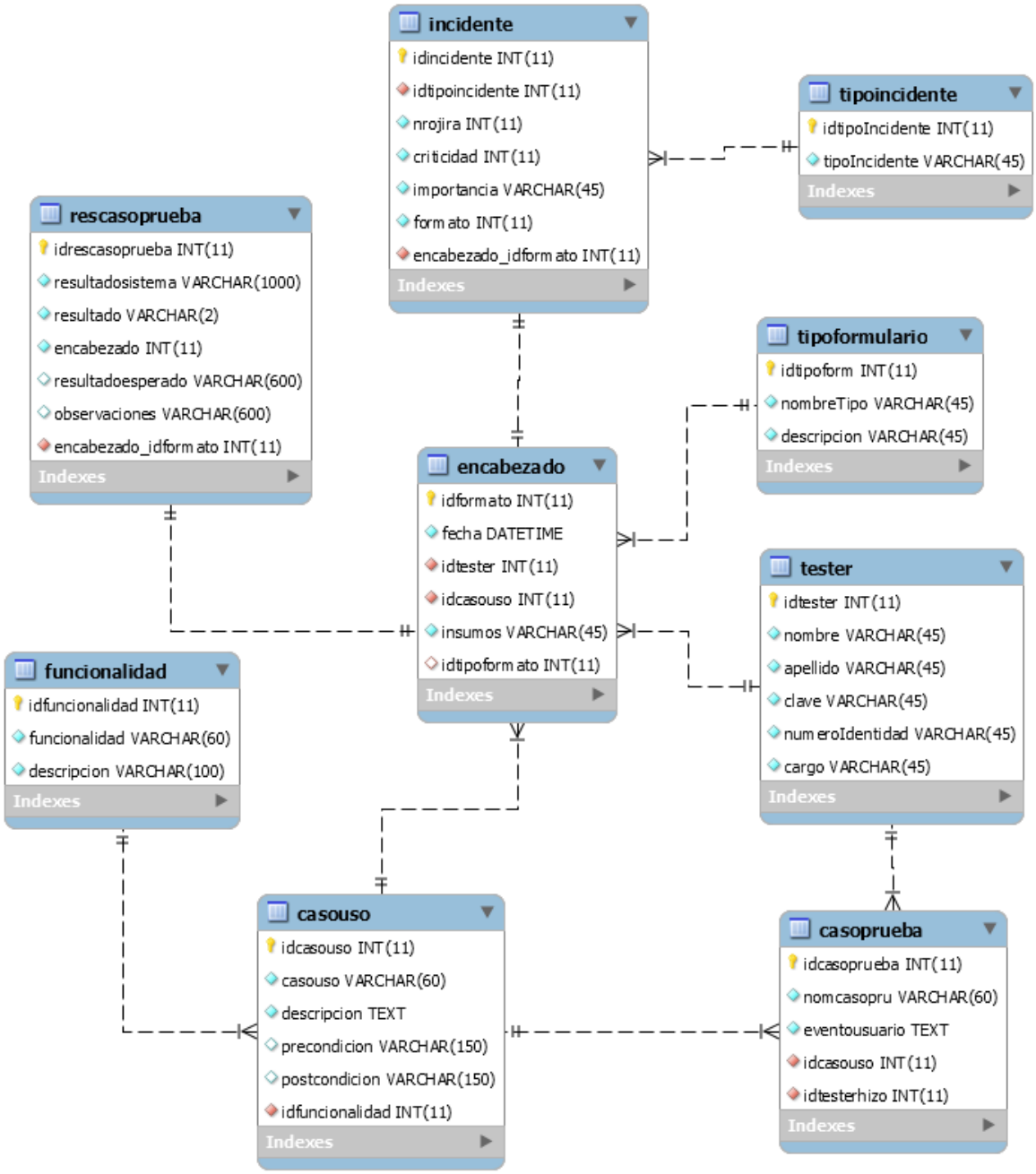


Figura 22 Diagrama de bases de datos

5.3.2. Pantallas

Inicio de sesión

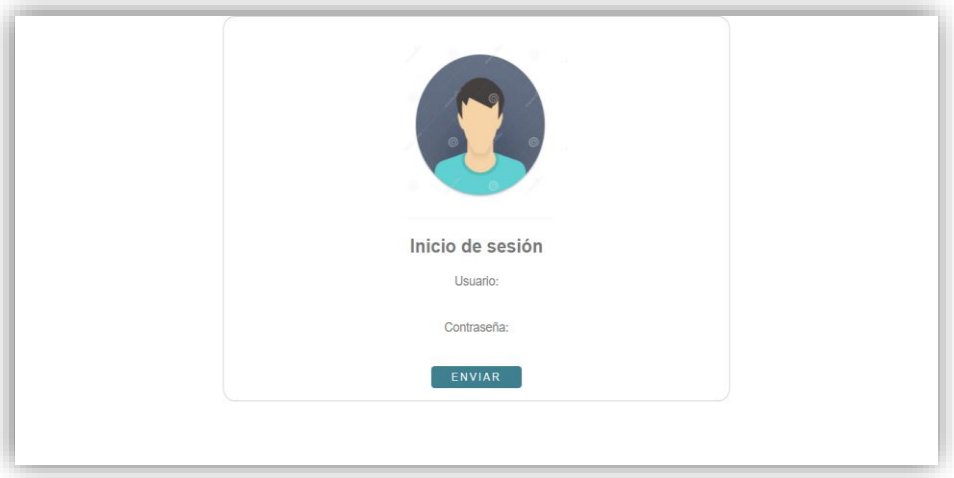


Figura 23 Inicio de sesión

Home

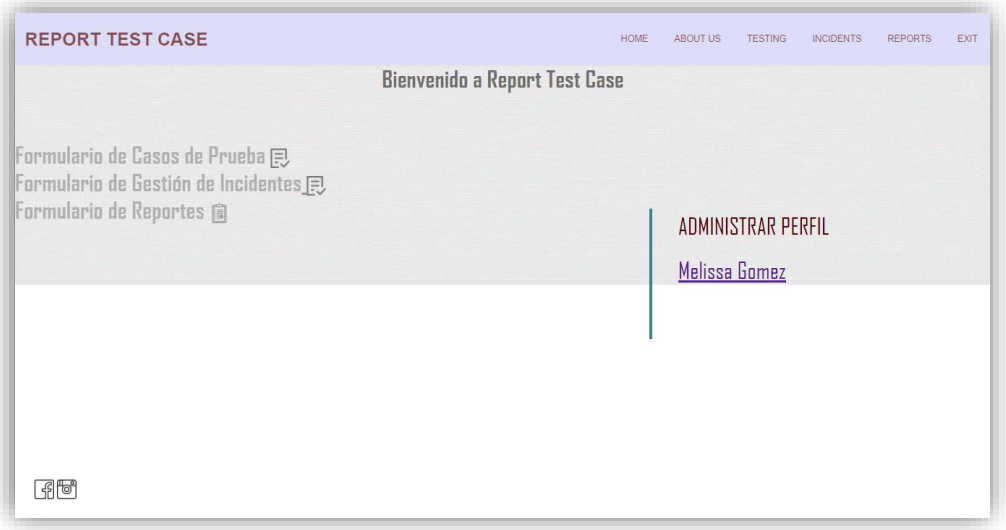


Figura 24 home de gestor de archivos

Acerca de nosotros

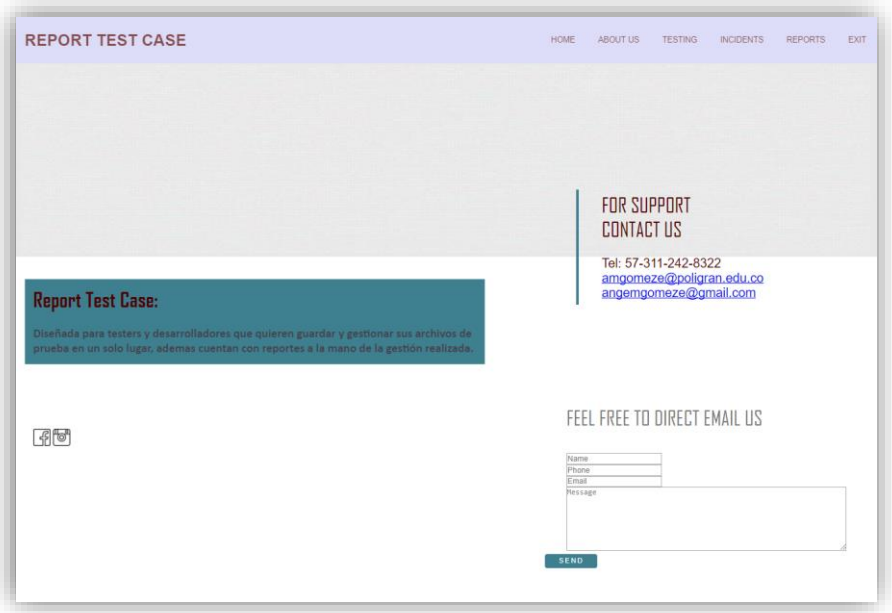


Figura 25 Acerca de nosotros

Formulario caso de prueba

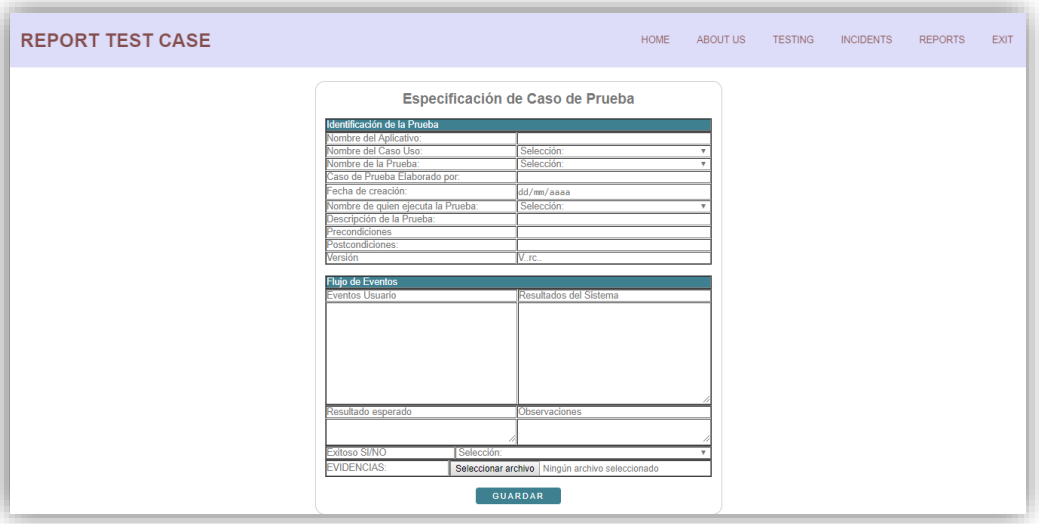


Figura 26 Formulario de casos de prueba

Formulario de incidentes

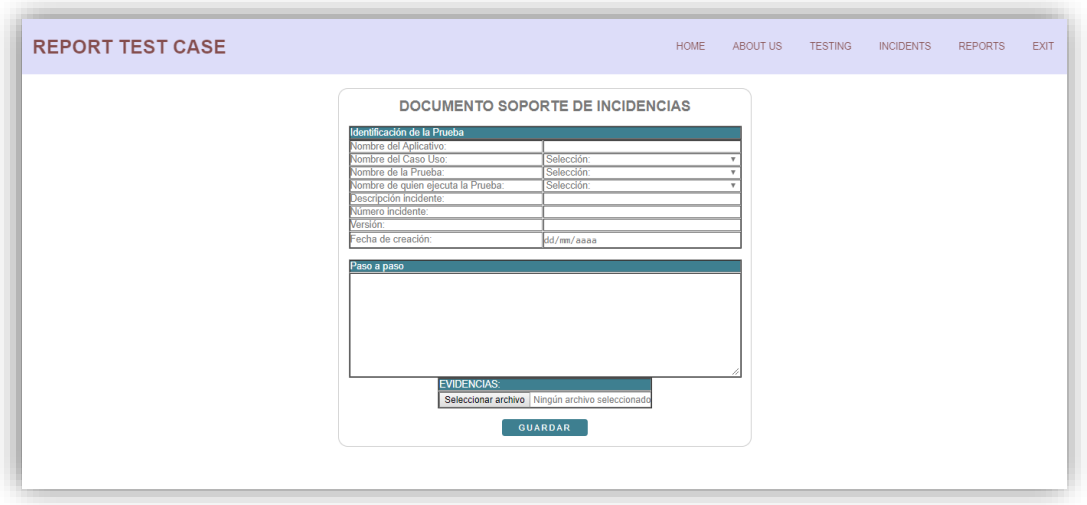


Figura 27 Formulario de incidentes

Reportes

Figura 28 Reportes de formularios

6. Proceso para la implementación de la metodología de automatización de pruebas

A continuación, se detalla el paso a paso realizado para la automatización de pruebas:

6.1. Caracterización del proceso actual

Se identifica proceso en la ANDJE actual, personas involucradas, herramientas utilizadas, procedimientos y se recolecto información de proceso actual donde se construyó una tabla de tiempos con los casos de prueba que se clasificaban de acuerdo con la complejidad de un caso de uso.

6.2. Identificar procesos susceptibles de automatización

Buscar metodologías, tipos de pruebas, información de herramientas, buscar casos de usos comunes o transversales.

6.3. Caracterizar el proceso futuro

Se realizó proceso de preparación del entorno con las herramientas establecidas y se realizaron posteriormente pruebas de conceptos de las herramientas, para empezar a utilizarlas, lo cual fue importante para entender las herramientas y comenzar a utilizarlas.

Se realizo diagrama de proceso con bizagi del proceso propuesto para realizar la automatización.

6.4. Planeación de la implementación

De acuerdo con el análisis de mercado de las diferentes herramientas se identifica que herramientas pueden ser implementadas y que herramientas son necesarias para su desarrollo.

Se comienza con la implementación de pruebas automatizadas, grabando las pruebas de los casos de uso identificados.

Se evalúan los tiempos y el impacto que esta produce para continuar con la metodología

6.5. Evaluar resultados

Se analizaron los resultados para mejorar en el proceso de ejecución de pruebas de software automatizadas y se ajustaron artefactos para una mejor obtención de estadísticas. Se presenta proyecto a directivas para implementación de pruebas automatizadas de todo el equipo involucrando los desarrolladores en el proceso de pruebas.

Para este fin se realizo un articulo el cual se encuentra en proceso de revisión.

7. Resultados

En este proyecto de grado se ha definido un proceso para la ejecución de las pruebas automatizadas.

Se implementó una herramienta destinada para la automatización de pruebas (selenium) la cual demostró ser una herramienta robusta y de fácil integración con diferentes herramientas una de ellas JMeter con la que se realiza la integración para realizar las pruebas de carga.

Se realizó el análisis de los resultados obtenidos en un proceso donde se trabajaba con 4 tester y se necesita realizar pruebas en promedio a 8 casos de uso de complejidad media donde inicialmente se recorrieron 15 casos de uso, para un proyecto que se encontraba en cambio constantes, para las pruebas realizadas manualmente se obtuvieron los siguientes resultados:

VERSIÓN	CICLO 1 Versión 5.0 rc2 Ambiente pruebas	CICLO 2 Versión 5.0 rc5 Ambiente pruebas	CICLO 2 Versión 5.0 rc6 Ambiente pruebas	CICLO 2 Versión 5.0 rc8 Ambiente pruebas	CICLO 3 Versión 5.0 rc10 Ambiente pruebas	CICLO 3 Versión 5.0 rc12 Ambiente Pruebas	Promedio
HORAS UTILIZADAS	130	12	6	4	126	80	59,7
N CASOS DE USO PROBADOS	15	6	3	2	12	13	8,5
HORAS EMPLEADAS X CU ENTRE 4 TESTERS	2,2	0,5	0,5	0,5	2,6	1,5	1,3

Tabla 13 Tiempo en la realización de pruebas manuales

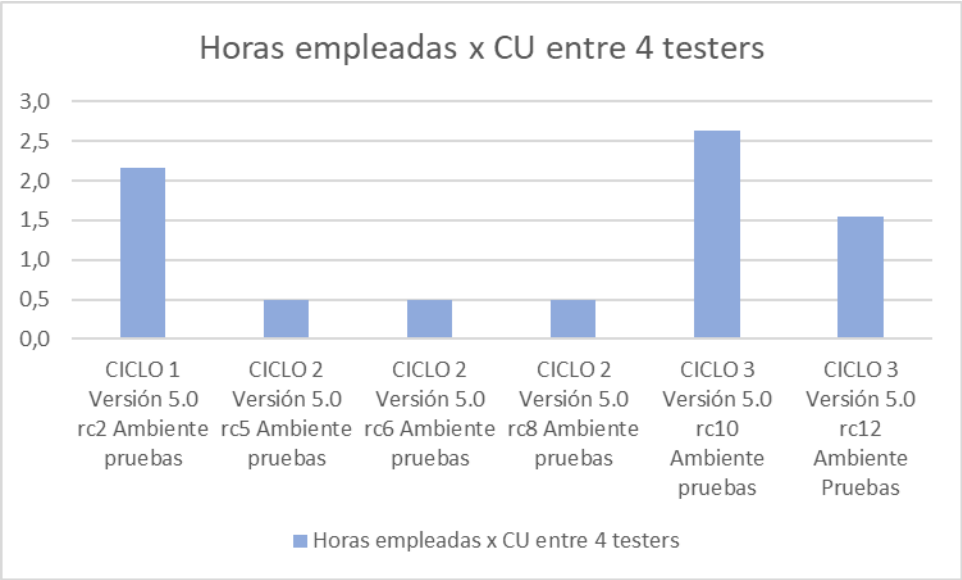


Figura 29 Horas empleadas en pruebas manuales

Luego de realizar la automatización de pruebas los siguientes fueron los resultados:

Versión	CICLO 1 Versión 6.0 rc4 y rc5 Ambiente pruebas	CICLO 1 Versión 7.0 rc1 Ambiente pruebas	CICLO 2 Versión 7.0 rc3 Ambiente pruebas	CICLO 2 Versión 7.0 rc4 Ambiente pruebas	CICLO 2 Versión 7.0 rc5, rc6 Ambiente pruebas	CICLO 2 Versión 7.0 rc7, 8 Ambiente pruebas	Promedio
Horas utilizadas	19	12	2	5	5	2	7,5
N casos de uso probados	9	8	2	6	6	3	5,7
Horas empleadas x CU entre 4 testers	0,5	0,4	0,3	0,2	0,2	0,2	0,3

Tabla 14 Tiempo de ejecución de prueba automatizadas

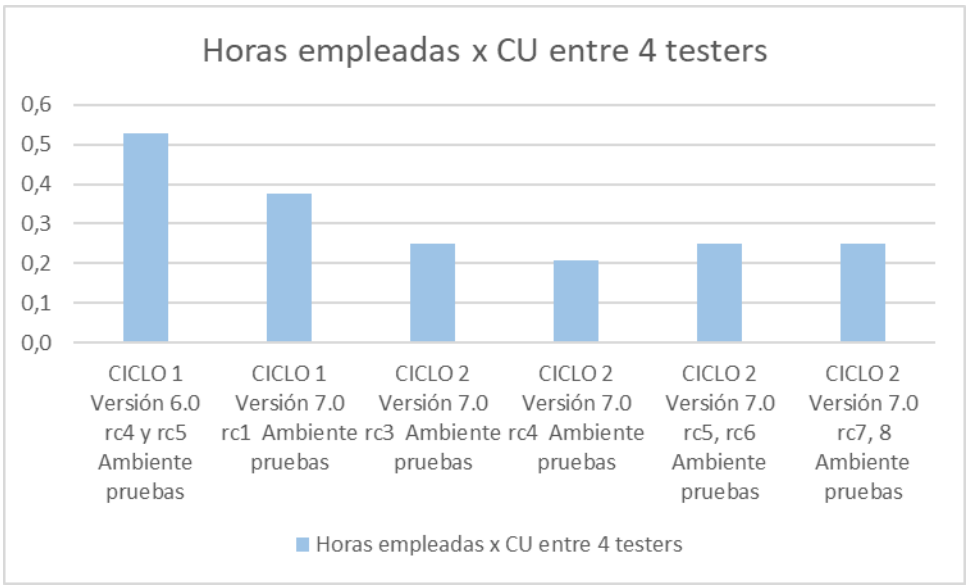


Figura 30 Tiempo empleado en pruebas automatizadas

Después de la implementación de la automatización de pruebas con Selenium el tiempo promedio de pruebas se redujo de 1,3 a 0,3 horas realizando las pruebas aproximadamente un 75% más rápido.

8. Conclusiones

El proyecto plantea una metodología de automatización de pruebas para proyectos futuros.

Se implementó una herramienta a la medida de las necesidades de la agencia nacional de defensa jurídica del estado, dando solución a una necesidad manifiesta en lo que tiene que ver con la automatización de pruebas enmarcado en el ciclo de desarrollo de software particular de la organización.

La implementación de la herramienta arrojó una mejora en los tiempos de ejecución de aproximadamente el 75%.

Al utilizar Selenium y estar basada en código abierto resulta una solución de bajo costo altamente conveniente teniendo en cuenta la naturaleza pública de la entidad, contribuye a optimizar costos de pruebas.

Adicionalmente se realizó un desarrollo especializado para la ANDJE para poder obtener estadísticas de pruebas más fácilmente.

9. Trabajo futuro

Se propone realizar una integración con las herramientas existentes para realizar informes y agilizar el proceso de reportes, documentación y codificación de pruebas con el fin de unificar el proceso y poder incluir a los desarrolladores en el proceso de automatización de pruebas.

Generalizar la metodología realizada para el ámbito de la agencia para otros proyectos tecnológicos.

Implementar pruebas de selenium en navegador Phantom para mejorar los tiempos de respuesta.

10. Bibliografía

Agencia nacional de defensa jurídica del estado. (20 de junio de 2013). *Ekogui*. Recuperado el 16 de septiembre de 2017, de ¿Qué es ekogui?: www.ekogui.gov.co

Agencia nacional de defensa jurídica del estado. (6 de junio de 2017). *Fuciones y deberes - ANDJE*. Obtenido de www.defensajuridica.gov.co: https://www.defensajuridica.gov.co/agencia/quienessomos/.../objetivos-funciones.aspx

Agencia Nacional de la Defensa jurídica de Estado. (18 de septiembre de 2017). *ANDJE*. Obtenido de https://www.defensajuridica.gov.co: https://www.defensajuridica.gov.co/agencia/dependencias/Paginas/direcciones.aspx

Allen, L. (2012). *Advanced Penetration Testing for Highly-Secured Environments: The Ultimate Securuty Guide.* Birminghan: Pack Publishing Ltd. Recuperado el 15 de octubre de 2017, de tutorialspoint: https://www.tutorialspoint.com/software_testing_dictionary/web_application_testing.htm

Collins, E., Dias-Neto, A., & de Lucena Jr., V. (2012). Strategies for Agile Software Testing Automation: An Industrial Experience. *Proceedings - International Computer Software and Applications Conference*, pp. 440-445. doi:10.1002/stvr.1639

Development Team TestLink . (13 de noviembre de 2017). *TestLink Open Source Test Management*. Obtenido de Testlink: http://testlink.org/

Escobar-Sánchez, M. E., & Fuertes-Díaz, W. (2015). Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2. *Fac. Ing, vol.24*(39), pp. 31-41.

Fitnesse. (11 de Noviembre de 2017). *Fitnesse*. Obtenido de Fitnesse: http://fitnesse.org/

Gil, C., Diaz, J., Orozco, M., de la Hoz, A., de la Hoz, E., & Morales, R. (2016). Agile Testing Practices in Software Quality: State of the Art Review. *Journal of Theoretical and Applied Information Technology, vol.92*(no. 1).

IBM. (30 de 10 de 2017). *Rational Functional Tester*. Obtenido de Rational Functional Tester: https://www.ibm.com/bs-en/marketplace/rational-functional-tester

International Software Testing Qualifications Board. (10 de octubre de 2017). *Certified Tester Foundation Level Syllabus, Version 2011.* Obtenido de Certified Tester Foundation Level Syllabus,Version 2011: http://www.bcs.org/upload/pdf/ct-foundation-syllabus.pdf

Introduction: Robotframework. (17 de octubre de 2017). Obtenido de Robotframework: robotframework.org

Janzen, D., & Saiedian, H. (2008). Does Test-Driven Development Really Improve Software Design Quality? *software metrics*, 77-84. Obtenido de Http://www.computer.org/software

Junta de andalucia. (01 de 10 de 2017). *Selenium y la automatización de las pruebas: Junta de andalucia*. Obtenido de Marco de desarrollo de la junta de andalucia: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/381

Jústiz-Núñez, D., Gómez-Suárez, D., Delgado, M. D., Dapena, Politécnico, S., & Antonio, J. (2014). *Testing process for software products at a quality laboratory*. Recuperado el 13 de octubre de 2017, de http://redalyc.org/pdf/3604/360433597003.pdf

Katalon Studio. (11 de Noviembre de 2017). *Katalon Studio: Best automated testing tool for web, mobile, API*. Obtenido de Katalon Studio: https://www.katalon.com/features/

Laurent, T., Ventresque, A., Papadakis, M., Christopher , H., & Le Traon, Y. (2015). Assessing and Improving the Mutation Testing Practiceof PIT. *School of Computer Science*.

Mera-Paz, J. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería Solidaria, vol. 12*(no. 20), pp. 163-176. doi:http://dx.doi.org/10.16925/in.v12i20.1482

Meyer, B. (2008). Seven Principles of Software Testing. *Software technologies IEEE*, 99-101.

Microsoft. (13 de noviembre de 2017). *Visual Studio Test Professional*. Obtenido de Visual Studio Test Professional: https://www.visualstudio.com/es/vs/test-professional/

Myers, G. J. (2004). *The Art of Software Testing, Second Edition.* Nueva Jersey: John Wiley & Sons, Inc.

Presidencia de la República. (1 de noviembre de 2011). Artículo 3. Por el cual se establecen los objetivos y la estructura de la Agencia Nacional de Defensa Jurídica del Estado. *(Decreto 4085 de 2011)*.

Pressman, R. S., & Maxim, B. R. (2015). Software Engienering A Pratitioner's Approach. En R. S. Pressman, & B. R. Maxim, *Software Engienering A Pratitioner's Approach* (págs. 466-495). Nueva York, Estados unidos de america: McGraw-Hill Education.

Ranorex GmbH. (30 de 10 de 2017). *Test Automation For Everyone: Ranorex*. Obtenido de Ranorex: https://www.ranorex.com/

Sahi. (21 de 10 de 2017). *Sahi Pro The tester's automation tool*. Obtenido de Sahi : http://sahipro.com/

Selenide. (21 de 10 de 2017). *SELENIDE*. Obtenido de SELENIDE: http://selenide.org/index.html

Selenium. (11 de 10 de 2017). *SeleniumHQ* . Obtenido de SeleniumHQ : http://www.seleniumhq.org

Sparx system. (12 de noviembre de 2017). *Enterprise Architect - Herramienta de diseño UML*. Obtenido de Sparx system: http://www.sparxsystems.com.ar/products/ea.html

Techtarget, S. S. (16 de 10 de 2017). *Search Software Quality Techtarget*. Obtenido de Understanding performance, load and stress testing: http://searchsoftwarequality.techtarget.com/answer/Understanding-performance-load-and-stress-testing

w3ii.com. (01 de 10 de 2017). *w3ii.com*. Obtenido de w3ii.com: http://www.w3ii.com/es/selenium/selenium_parameterizing_using_excel.html

Watir. (21 de 10 de 2017). *Watir Powered by Selenium*. Obtenido de Watir: http://watir.com/

WiklundK, E., & LundqvistK, S. (2017). Impediments for software test automation: A systematic literature review. *Softw Test Verif Reliab*, 20. Obtenido de https://doi.org/10.1002/stvr.1639

ANEXO I Requisitos de software

Descargar e instalar software Java (SDK)

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Descargar e instalar Mozilla Firefox 54.0.1 (Con opción de no actualización) https://ftp.mozilla.org/pub/firefox/releases/54.0.1/

Descargar e instalar Selenium IDE 2.9.1 en la página de complementos

https://addons.mozilla.org/es/firefox/addon/selenium-ide/

Descargar e instalar eclipse

http://www.eclipse.org/downloads/

Descargar e instalar Selenium WebDriver Java 3.8.1

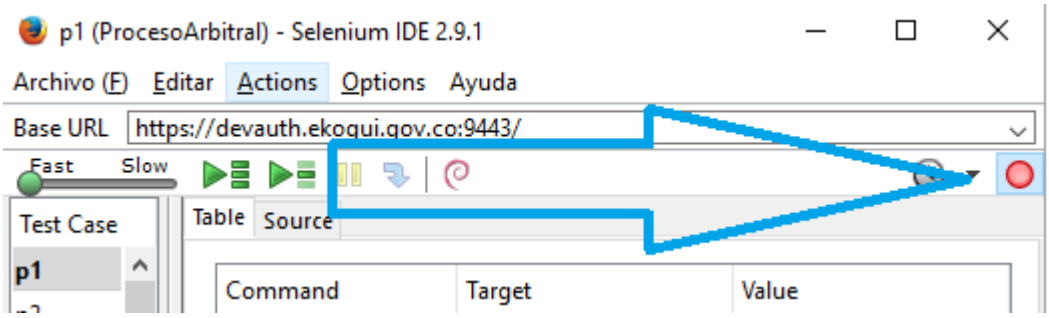
http://www.seleniumhq.org/download/

Luego de realizar la instalación puede empezar a utilizar Selenium IDE

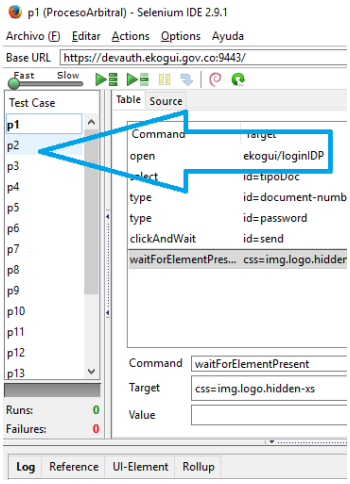
(Junta de andalucia, 2017)

Componentes de Selenium

Grabar prueba: Para comenzar a grabar podemos hacerlo con el botón rojo que se muestra a continuación:

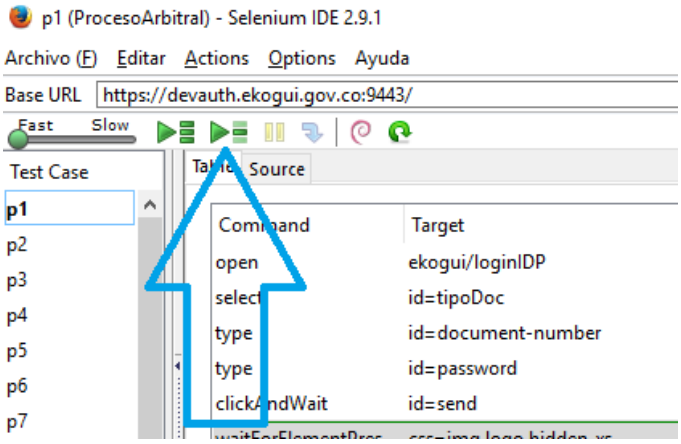


Casos de prueba: aparece en la parte izquierda**:**



(w3ii.com, 2017)

Reproducir pruebas automatizadas: Se encuentra en la parte de arriba:



ANEXO II Instalar complementos de Selenium 2.9.1 y uso.

Instalación

Instale los bloques Sel para el complemento IDE Selenium con el Administrador de complementos en Firefox desde:
https://addons.mozilla.org/en-US/firefox/addon/selenium-ide-sel-blocks/?src=search

Creación de pruebas conducidas por datos

Primero necesitamos identificar los datos de prueba para el script. El complemento Sel Bloques necesita los datos de prueba en formato XML.

Se necesita instalar complemento XML para selenium desde:

Para las pruebas realizadas por datos debe guardarse un archivo de datos Cree un archivo XML con el siguiente formato:

```
<? xml version="1.0" encoding="UTF-8" standalone="yes"?>
<testdata>
    <vars buscar1="Selenium IDE" buscar2="Selenium para pruebas"/>
    <vars buscar1="Pruebas automatizadas" buscar2=" otra prueba automatizadas"/>
</testdata>
```

Para más información

(https://unmesh.me/2012/12/04/data-driven-testing-with-selenium-ide/)

Comandos más utilizados para las pruebas

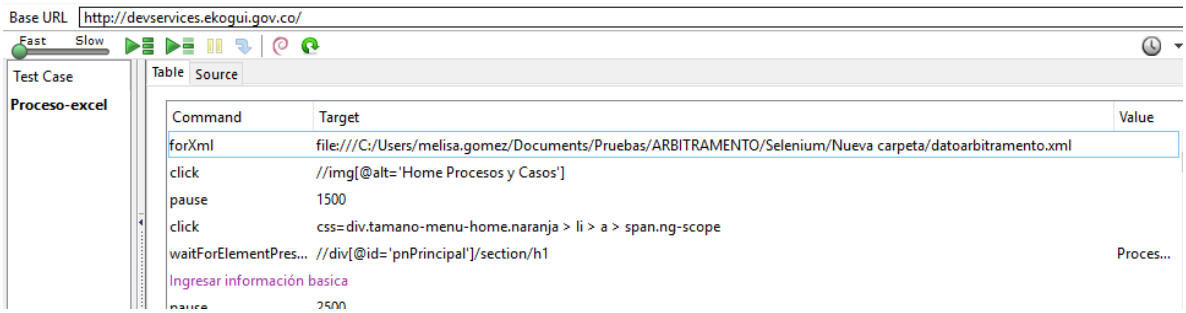
- Recorrer archivo Excel:

Command: forXML

Target: “Documento del cual se desea extraer los documentos”

- tope del forXML:

Command: endXML

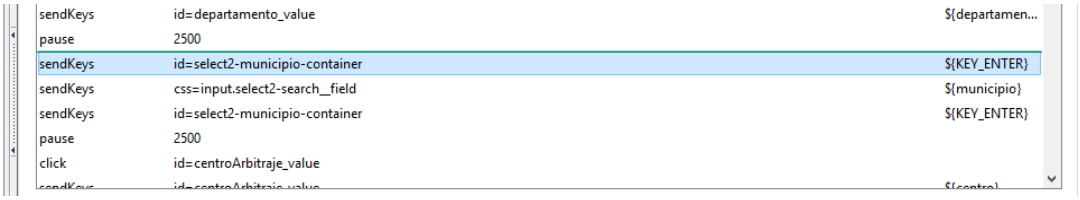


- Simular Enter

Command: sendKeys

Target: “campo”

Value: ${KEY_ENTER}

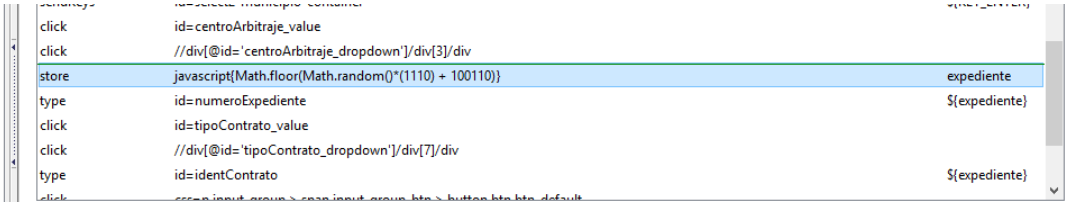


- Declarar variable

Command: store

Target: “campo”

Value: ${nombre de la variable}



- Digitar un campo

Command: type

Target: “campo”

Value: ${valor de variable} o texto a escribir

- Hacer click

Command: Click

Target: “campo”

- Pausar la prueba

Command: Pause

Target: “tiempo en milésimas de segundos”

- Esperar hasta que un elemento aparezca

Command: wairForElementPresent

Target: “campo que desea esperar que aparezca”

ANEXO III Convertir pruebas de Selenium IDE a Selenium WebDriver

Después de grabar pruebas en selenium IDE y comprobarlas para poder realizar pruebas en diferentes navegadores y versiones puede utilizar Selenium WebDriver

1. Navegue a Opciones> Formato y seleccione la opción " Java / Junit 4 / WebDriver ".
2. En Eclipse crear un nuevo proyecto.
3. Ingrese el nombre del proyecto.
4. Verificar carpeta src.
5. Crear nueva clase New> Class.
6. Ingresar en el nombre de clase.
7. En la clase pegar el código generado de Selenium IDE.

Para Firefox agregar la siguiente línea de código:

```
System.setProperty("webdriver.gecko.driver","\\geckodriver.exe");
    driver = new FirefoxDriver();
```

Ejemplo:

```
import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import org.junit.*;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class Arbitrales {

  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
      System.setProperty("webdriver.gecko.driver", "\\selenium\\geckodriver\\geckodriver.exe");
    driver = new FirefoxDriver();
    baseUrl = "http://devservices.ekogui.gov.co/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test
  public void testArbitrales() throws Exception {
      driver.get(baseUrl + "/ekogui/loginIDP");
    new Select(driver.findElement(By.id("tipoDoc"))).selectByVisibleText("Cédula de Ciudadania");
    driver.findElement(By.id("document-number")).clear();
    driver.findElement(By.id("document-number")).sendKeys("1019054714");
    driver.findElement(By.id("password")).clear();
    driver.findElement(By.id("password")).sendKeys("Agentesoporte_16");
    driver.findElement(By.id("send")).click();
  }

  @After
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }

  private boolean isElementPresent(By by) {
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }
```

```
  private boolean isAlertPresent() {
    try {
      driver.switchTo().alert();
      return true;
    } catch (NoAlertPresentException e) {
      return false;
    }
  }

  private String closeAlertAndGetItsText() {
    try {
      Alert alert = driver.switchTo().alert();
      String alertText = alert.getText();
      if (acceptNextAlert) {
        alert.accept();
      } else {
        alert.dismiss();
      }
      return alertText;
    } finally {
      acceptNextAlert = true;
    }
  }
}
```

Para Chrome agregar la siguiente línea de código:

```
System.setProperty("webdriver.chrome.driver", "\\chromedriver.exe");
      driver = new ChromeDriver();
```

Ejemplo:

```
import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import org.junit.*;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class ArbitralesChrome {

  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
      System.setProperty("webdriver.chrome.driver", "C:\\Users\\Ange\\Documents\\Proyectos Eclipse\\Selenium\\selenium\\chromedriver\\chromedriver.exe");
      driver = new ChromeDriver();
    baseUrl = "http://devservices.ekogui.gov.co/";
      driver.get(baseUrl + "/ekogui/loginIDP");
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test
  public void testArbitrales() throws Exception {

      Thread.sleep(5000);
    new Select(driver.findElement(By.id("tipoDoc"))).selectByVisibleText("Cédula de Ciudadania");
    driver.findElement(By.id("document-number")).clear();
    driver.findElement(By.id("document-number")).sendKeys("1019054714");
    driver.findElement(By.id("password")).clear();
    driver.findElement(By.id("password")).sendKeys("Agentesoporte_16");
    driver.findElement(By.id("send")).click();
  }

  @After
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }
```

```
  @After
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }

  private boolean isElementPresent(By by) {
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }

  private boolean isAlertPresent() {
    try {
      driver.switchTo().alert();
      return true;
    } catch (NoAlertPresentException e) {
      return false;
    }
  }

  private String closeAlertAndGetItsText() {
    try {
      Alert alert = driver.switchTo().alert();
      String alertText = alert.getText();
      if (acceptNextAlert) {
        alert.accept();
      } else {
        alert.dismiss();
      }
      return alertText;
    } finally {
      acceptNextAlert = true;
    }
  }
}
```

ANEXO IV Código de desarrollo especifico realizado para la documentación de pruebas de la (ANDJE)

A continuación, se hace una descripción del código del desarrollo realizado para la documentación de pruebas adicional el código se puede encontrar en la carpeta reportes con su respectiva base de datos:

Login.php

<!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>Ingreso</title> </head> <body> <link href='https://fonts.googleapis.com/css?family=Work+Sans:400,300,700' rel='stylesheet' type='css/css'> <link rel="stylesheet" href="css/style1.css"> <form action="checklogin.php" method="post"> <button class= "profile__avatar" id="profile" disabled="disabled"> </button> <h2 ="-50%">Inicio de sesión</h2>	Head de la pantalla con imagen de inicio y título de la pantalla
 <label class="label">Usuario:</label> <input type="text" id="username" name="username" class="input"/> </br> 	Campos de captura de información de usuario y contraseña de la aplicación.

<label class="label">Contraseña:</label> <input type="password" id="password" name="password" class="input" required pattern=.*\S.* /> 	
  <button class="btn" id="btnformulario">Enviar</button> </form> </body> </html>	Botón de envío con conexión a checklogin.php para comprobar información del usuario

Checklogin.php

<?php session_start(); ?>	Se crea una sesión en PHP después de comprobar la autenticación
<?php $username = $_POST['username']; $password = $_POST['password']; include("connexion.php"); $query = "SELECT * FROM tester where numeroIdentidad = '$username'"; $result = mysql_query($query) or die('Consulta fallida: ' . mysql_error());	Se traen los datos ingresados de usuario y contraseña capturados con método POST Comienza conexión a base de datos mysql a través de conexión.php Se hace consulta a base de datos para verificar si el usuario existe

if ($row = mysql_fetch_array($result)){
do {
if ($password== $row['clave']) {
 	
$_SESSION['loggedin'] = true;
$_SESSION['username'] = $user;
$_SESSION['start'] = time();
$_SESSION['expire'] = $_SESSION['start'] + (15 * 60);
	Si el usuario existe se comprueba que la contraseña ingresada sea la correcta Se hace inicio de sesión PHP con un time out de 15 minutos

echo " <script language='JavaScript'>
function B()
{
location.href ='home.php';
}
B();
</script>";
}	

else {
echo "Usuario y/o clave estan incorrectos.";
echo "
Volver a Intentarlo";
}
} while ($row = mysql_fetch_array($result));
 	
}
else {
echo "Usuario y/o clave estan incorrectos.";
echo "
Volver a Intentarlo";
}
?>
	En caso de no encontrar el usuario o contraseña muestra mensaje “Usuario y/o clave están incorrectos."

Logout.php

<?php
 	
session_start();
unset ($SESSION['username']);
session_destroy();
 	
echo " <script language='JavaScript'>
function B()
{
location.href ='login.php';
}
B();
</script>";
?>
	Renueva o crea sesión en caso de que no exista
	Cierra la sesión que se encuentra activa

verifySession.php

<?php session_start();	Verifica si existe una sesión autenticada activa Si no, aparece mensaje para que se autentique o cree una nueva cuenta

```
if (isset($_SESSION['loggedin'])
&& $_SESSION['loggedin'] == true)
{

} else {
   echo "Esta pagina es solo para
usuarios registrados.<br>";
   echo "<br><a
href='login.php'>Login</a>";
   echo "<br><br><a
href='index.php'>Registrarme</a>";

exit;
}

$now = time();

if($now > $_SESSION['expire']) {
session_destroy();

echo "Su sesión a terminado,
<a href='login.php'>Necesita Hacer
Login</a>";
exit;
}
?>
```

Menu.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en">

<head>
  <title>Reportes</title>

   <link rel="stylesheet"
href="css/style1.css">
</head>
<body>
    <div id="header" class="color">
          <center><h1>Report Test Case</h1>
        </div>
      </div>
      <div>
        <ul>
          <!-- put class="tab_selected" in the
li tag for the selected page - to highlight
which page you're on -->
          <li><a href="home.php">Home</a></li>
          <li><a href="about.php">Acerca de
nosostros</a></li>
          <li><a href="testCase.php">Casos de
prueba</a></li>
```

Se encuentra el menú en html

Incidentes Reportes Salir </html>	

Home.php

<?php include("verifySession.php"); ?> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>Crear formulario</title> <link rel="stylesheet" href="css/style1.css"> <?php include("menu.html"); ?> </head> <body>	Verifica que se encuentre en sesión y se agrega el menú y el título de pestaña
<form action="formulario.html" method="post"> Gestión de documentos <input	Título de la página.
type="button" class="btn2" id="test" name="test" value="Formulario Incidentes" onclick="testCase.php" /> </br> </br>	Aparece botón de enlace para iniciar diligenciar un formulario de casos de pruebas
<button class="btn2" id="btnformulario">Formulario Incidentes</button> </br> </br>	Aparece botón de enlace para iniciar diligenciar un formulario de solución de incidentes

<button class="btn2" id="btnformulario">Ver reportes</button> </body> </html>	Aparece botón de enlace para ver reportes

Testcase.php

<?php
include("verifySession.php");
?>
<?php
if(isset($_GET['opcion'])){
$opcion = $_GET['opcion'];
}
if(isset($_GET['opcion2'])){
$opcion2 = $_GET['opcion2'];
}
if(isset($_GET['projetcA'])){
$projetcA = $_GET['projetcA'];
}
?>
	Verifica que se encuentre en sesión y se agrega el menú y el título de pestaña

<!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>Crear formulario</title> <link rel="stylesheet" href="css/style1.css"> <?php include("menu.html"); ?> </head> <script language="javascript" type="text/javascript">	Título de la página.

function getRoute(){ var projetcA = document.getElementById('project').value;
var opcion = document.getElementById('useCase').value;
var opcion2 = document.getElementById('testCase').value;
location.href = "testCase.php?projetcA="+projetcA+ "&opcion="+opcion+"&opcion2="+opcion2;
}
	Función que renombra URL para cargar el formulario

</script> <body> <form action="save.php" method="post"> Especificación de Caso de Prueba 	Título de la pestaña

<TABLE BORDER=1>
<TD class="table">
Identificación de la Prueba
</TD>
</TABLE>
<TABLE BORDER=1>
<TR>
<TD WIDTH=300>
Nombre del Aplicativo:
</TD >
<TD WIDTH=300>
<input type="text" id="project" name="project" class="input" value="<?php echo "$projetcA" ?>" onchange="return getRoute();" required pattern=.*\S.* />
</TD>
</TR>
<TR>
<TD WIDTH=300>
Nombre del Caso Uso:
</TD>
<TD WIDTH=300>
<select name="useCase" id="useCase" onchange="return getRoute();" >
<?php include("connexion.php");
$query = "SELECT * FROM casouso where idcasouso='$opcion'";
$result = mysql_query($query) or die('Consulta fallida: ' . mysql_error());
while ($row = mysql_fetch_array($result))
{
?>
<option value=" <?php echo $row['idcasouso'];?> " >
<?php echo $row['casouso']; ?>
</option>
<?php
}
?>
<option value="0" >Selección:</option>
<?php
include("connexion.php");
$query = 'SELECT * FROM casouso';
$result = mysql_query($query) or die('Consulta fallida: ' . mysql_error());
	Tabla de ingreso, inputs y listas con contenido de mysql de la información básica

```
                        while ( $row =
mysql_fetch_array($result) )
                        {
                        ?>
                    <option value=" <?php
echo $row['idcasouso'];?> " >
                        <?php echo
$row['casouso']; ?>
                    </option>
                    <?php
                        }
                        ?>
                </select>
            </TD>
         </TR>
         <TR>
            <TD WIDTH=300>
               Nombre de la Prueba:
            </TD>
            <TD WIDTH=300>
               <select  name="testCase"
id="testCase" onchange="return
getRoute();" >
                    <?php

include("connexion.php");
                        $query2 = "SELECT *
FROM casoprueba where idcasoprueba
='$opcion2'";
                        $result2 =
mysql_query($query2) or die('Consulta
fallida: ' . mysql_error());
                        while ( $row =
mysql_fetch_array($result2) )
                        {
                          ?>
                    <option value=" <?php
echo $row['idcasoprueba'];?> " >
                        <?php echo
$row['nomcasopru']; ?>
                    </option>
                    <?php
                        }
                        ?>
                    <option value="0"
>Selección:</option>
                    <?php
include("connexion.php");
                        $query2 = "SELECT *
FROM casoprueba where idcasouso
='$opcion'";
                        $result2 =
mysql_query($query2) or die('Consulta
fallida: ' . mysql_error());
                        while ( $row =
mysql_fetch_array($result2) )
                        {
```

```
                    ?>
                  <option value=" <?php
echo $row['idcasoprueba'];?> " >
                    <?php echo
$row['nomcasopru']; ?>
                  </option>
                  <?php
                     }
                     ?>
               </select>
            </TD>
         </TR>
         <TR>
            <TD WIDTH=300>
               Caso de Prueba Elaborado
por:
            </TD>
            <TD WIDTH=300>
               <?php

include("connexion.php");
                  $query2 = "SELECT * FROM
tester where idtester = (select
idtesterhizo from casoprueba where
idcasoprueba='$opcion2')";
                  $result2 =
mysql_query($query2) or die('Consulta
fallida: ' . mysql_error());
                  while ( $row =
mysql_fetch_array($result2) )
                  {
                        ?>
               <?php echo $row['nombre']."
".$row['apellido']; ?>
               <?php   }       ?>
            </TD>
         </TR>
         <TR>
            <TD WIDTH=300>
               Fecha de creación:
            </TD>
            <TD WIDTH=300>
               <input type="date"
id="creationDate" name="creationDate"
class="input"
                  min="1990-01-01"
max="2018-12-31" required pattern=.*\S.*
/>
            </TD>
         </TR>
         <TR>
            <TD WIDTH=300>
               Nombre de quien ejecuta la
Prueba:
            </TD>
            <TD WIDTH=300>
```

```
                <select name="executester"
id="executester">
                  <option value="0"
>Selección:</option>
                  <?php

include("connexion.php");
                     $query4 = 'SELECT *
FROM tester';
                     $result4 =
mysql_query($query4) or die('Consulta
fallida: ' . mysql_error());
                     ?>
                  <?php
                     while ( $row1 =
mysql_fetch_array($result4) )
                     {
                       ?>
                  <option value=" <?php
echo $row1['idtester'];?> " >
                     <?php echo
$row1['nombre']." ".$row1['apellido']; ?>
                  </option>
                  <?php
                     }
                     ?>
               </select>
            </TD>
         </TR>
         <TR>
            <TD WIDTH=300>
               Descripción de la Prueba:
            </TD>
            <TD WIDTH=300
id="description" name="description">
               <?php

include("connexion.php");
                  $query = "SELECT
descripcion FROM casouso where
idcasouso='$opcion'";
                  $result =
mysql_query($query) or die('Consulta
fallida: ' . mysql_error());
                  ?>
               <?php
                  while ( $row1 =
mysql_fetch_array($result) )
                  {
                       ?>
               <?php echo
$row1['descripcion']; ?>
               </input>
               <?php
                  }
                  ?>
            </TD>
```

```
         </TR>
         <TR>
            <TD WIDTH=300>
               Precondiciones
            </TD>
            <TD WIDTH=300>
               <?php

include("connexion.php");
                  $query = "SELECT
precondicion FROM casouso where
idcasouso='$opcion'";
                  $result =
mysql_query($query) or die('Consulta
fallida: ' . mysql_error());
                  ?>
               <?php
                  while ( $row1 =
mysql_fetch_array($result) )
                  {
                        ?>
               <?php echo
$row1['precondicion']; ?>
               </input>
               <?php
                  }
                  ?>
            </TD>
         </TR>
         <TR>
            <TD WIDTH=300>
               Postcondiciones:
            </TD>
            <TD WIDTH=300>
               <?php

include("connexion.php");
                  $query = "SELECT
postcondicion FROM casouso where
idcasouso='$opcion'";
                  $result =
mysql_query($query) or die('Consulta
fallida: ' . mysql_error());
                  ?>
               <?php
                  while ( $row1 =
mysql_fetch_array($result) )
                  {
                        ?>
               <?php echo
$row1['postcondicion']; ?>
               </input>
               <?php
                  }
                  ?>
            </TD>
         </TR>
```

<TR> <TD WIDTH=300> Versión </TD > <TD WIDTH=300> <input type="text" id="version" name="version" class="input" value="V..rc.." required pattern=.*\S.* /> </TD> </TR> </TABLE> </br> <TABLE BORDER=1> <TD class="table"> Flujo de Eventos </TD> </TABLE>	

<TABLE BORDER=1>
<TR>
<TD WIDTH=300>
Eventos Usuario
</TD>
<TD WIDTH=300>
Resultados del Sistema
</TD>
</TR>
<TR>
<TD WIDTH=300 HEIGHT=150>
<?php
 	
include("connexion.php");
$query = "SELECT eventousuario FROM casoprueba where idcasoprueba='$opcion2'";
$result = mysql_query($query) or die('Consulta fallida: ' . mysql_error());
?>
<?php
while ($row1 = mysql_fetch_array($result))
{
?>
<?php echo $row1['eventousuario']; ?>
</input>
<?php
}
?>
</TD>
<TD WIDTH=300 WIDTH=300 HEIGHT=150 >
<textarea type="text" id="resultSystem" name="resultSystem" class="input" required
	Tabla de resultados del sistema carga la información de base de datos de acuerdo con el caso de prueba escogido.

```
                  pattern=.*\S.* />
</textarea>
            </TD>
         </TR>
      </TABLE>
      <div class="field">
         <TABLE BORDER=1>
            <TR>
               <TD WIDTH=300>
                  Resultado esperado
               </TD>
               <TD WIDTH=300>
                  Observaciones
               </TD>
            </TR>
            <TR>
               <TD WIDTH=300 >
                  <textarea type="text"
id="resultHope" name="resultHope"
class="input"  required
                        pattern=.*\S.* />
</textarea>
               </TD>
               <TD WIDTH=300 >
                  <textarea type="text"
id="observation" name="observation"
class="input"  required
                        pattern=.*\S.* />
</textarea>
               </TD>
            </TR>
         </TABLE>
         <TABLE BORDER=1>
            <TR>
               <TD WIDTH=200>
                  Exitoso SI/NO
               </TD>
               <TD WIDTH=400>
                  <select
name="successful" id="successful">
                     <option value="0"
>Selección:</option>
                     <option value="1"
>SI</option>
                     <option value="2"
>NO</option>
                  </select>
               </TD>
            </TR>
         </TABLE>
      </div>
      <div>
         <TABLE BORDER=1>
            <TR>
               <TD WIDTH=200>
                  <label for="fieldResult"
class="label">EVIDENCIAS: </label>
```

</TD> <TD WIDTH=400> <input name="image" id="image" type="file" size="20"> </TD> </TR> </TABLE> </BR> <button class="btn" id="btnformulario">Guardar</button> </form> 	

Report.php

<?php include("verifySession.php"); ?> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>Reportes</title> <?php include("menu.html"); ?> </head> <body> <form action="formulario.html" method="post"> Reportes de casos de prueba 	Verifica el estado de la sesión Carga el menú Muestra título de inicio de la pantalla
<?php include("connexion.php"); $query = "SELECT * FROM rescasoprueba"; $result = mysql_query($query) or die('Consulta fallida: ' . mysql_error()); ?> <TABLE BORDER=1> <tr class="table"> <td WIDTH=100><?php echo "Id resultado de prueba"; ?></td> <td	Muestra tabla con los formularios registrados por el usuario con opción de edición o eliminación.

```
WIDTH=100><?php echo "Resultado sistema";
?></td>

                        <td
WIDTH=100><?php echo "Fue exitoso?";
?></td>

                        <td
WIDTH=100><?php echo "Resultado
esperado"; ?></td>

                        <td
WIDTH=100><?php echo "Observaciones"
?></td>

                        <td
WIDTH=50><?php echo "Eliminar"?></td>

        </tr>

        </TABLE>

<?php

        while ( $row =
mysql_fetch_array($result) )
                                        {
?>
                    <TABLE BORDER=1>
                            <tr>

                        <td
WIDTH=100><?php echo
$row['idrescasoprueba']; ?></td>
                                <td
WIDTH=100><?php echo
$row['resultadosistema']; ?></td>

                        <td
WIDTH=100><?php echo $row['resultado'];
?></td>
<td WIDTH=100><?php echo
$row['resultadoesperado']; ?></td>

                        <td
WIDTH=100><?php echo
$row['observaciones']; ?></td>

                        <td WIDTH=50
id="<?php echo $row['idrescasoprueba'];
?>" name="<?php echo
$row['idrescasoprueba']; ?>"
                        <a
href="delete.php?<?php echo
$row['idrescasoprueba'];
?>">Eliminar</a></td>
        </tr>
```

</table> <?php } ?> </form> </body> </html>	

reportBug.php

<?php
include("verifySession.php");
?>
<?php
if(isset($_GET['opcion'])){
$opcion = $_GET['opcion'];
}
if(isset($_GET['opcion2'])){
$opcion2 = $_GET['opcion2'];
}
if(isset($_GET['projetcA'])){
$projetcA = $_GET['projetcA'];
}
?>
	Verifica que se encuentre en sesión y se agrega el menú y el título de pestaña

<?php
include("verifySession.php");
?>
<?php
if(isset($_GET['opcion'])){
$opcion = $_GET['opcion'];
}
if(isset($_GET['opcion2'])){
$opcion2 = $_GET['opcion2'];
}
if(isset($_GET['projetcA'])){
$projetcA = $_GET['projetcA'];
}
?>
	Verifica que se encuentre en sesión y se agrega el menú y el título de pestaña

<!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>Crear usuario</title> <link rel="stylesheet" href="css/style1.css"> <?php include("menu.html"); ?> </head> <body> <link href='https://fonts.googleapis.com/css? family=Work+Sans:400,300,700' rel='stylesheet' type='text/css'>	Se carga menú y nombre de la pestaña

<form action="save.php" method="post"> 	

DOCUMENTO SOPORTE DE INCIDENCIAS
 	
<TABLE BORDER=1>
<TD class="table">
Identificación de la Prueba
</TD>
 	
</TABLE>
<TABLE BORDER=1>
<TR>
<TD WIDTH=300>
Nombre del Aplicativo:
</TD >
<TD WIDTH=300>
<input type="text" id="project" name="project" class="input" value="<?php echo "$projetcA" ?>" onchange="return getRoute();" required pattern=.*\S.* />
</TD>
</TR>
<TR>
<TD WIDTH=300>
Nombre del Caso Uso:
</TD>
<TD WIDTH=300>
<select name="useCase"
 	
id="useCase" onchange="return getRoute();" >
<?php
include("connexion.php");
$query = "SELECT * FROM casouso where idcasouso='$opcion'";
$result = mysql_query($query) or die('Consulta fallida: ' . mysql_error());
while ($row = mysql_fetch_array($result))
{
?>
<option value=" <?php echo $row['idcasouso'];?> " >
<?php echo $row['casouso']; ?>
</option>
<?php
}
?>
<option value="0" >Selección:</option>
<?php
include("connexion.php");
	Se crea tabla de información de encabezado, información básica del incidente

$query = 'SELECT * FROM casouso';
$result = mysql_query($query) or
die('Consulta fallida: ' .
mysql_error());
while ($row =
mysql_fetch_array($result))
{
?>	

<option value=" <?php echo
$row['idcasouso'];?> " >
<?php echo $row['casouso']; ?>
</option>
<?php
}
?>
</select>
</TD>
</TR>
<TR>
<TD WIDTH=300>
 	
Nombre de la Prueba:
</TD>
<TD WIDTH=300>
<select
name="testCase" id="testCase"
onchange="return getRoute();" >
 	
<?php
 	
include("connexion.php");
 	
$query2 = "SELECT * FROM
casoprueba where idcasoprueba
='$opcion2'";
$result2 = mysql_query($query2)
or die('Consulta fallida: ' .
mysql_error());
while ($row =
mysql_fetch_array($result2))
{
?>
<option value=" <?php echo
$row['idcasoprueba'];?> " >
 	
<?php echo $row['nomcasopru'];
?>
</option>
<?php
}
?> <option value="0"
>Selección:</option>
<?php
include("connexion.php");
$query2 = "SELECT * FROM casoprueba
where idcasouso ='$opcion'";
	Campos input para guardar la información del paso a paso e información del incidente

$result2 = mysql_query($query2) or die('Consulta fallida: ' . mysql_error());
while ($row = mysql_fetch_array($result2))
{
?>
<option value=" <?php echo $row['idcasoprueba'];?> " >
<?php echo $row['nomcasopru']; ?>
</option>
<?php
}
?>
</select>
</TD>
</TR>
<TR>
<TD WIDTH=300>
Nombre de quien ejecuta la Prueba:
</TD>
<TD WIDTH=300>
<select >
<option value="0" >Selección:</option>
<?php
include("connexion.php");
$query4 = 'SELECT * FROM tester';
$result4 = mysql_query($query4) or die('Consulta fallida: ' . mysql_error());
?>	

Delete.php

<?php $idformato = $_POST["idformato"];	Se obtiene variable del ID del formulario a eliminar mediante el método POST

// Abrimos la conexión a la base de datos
include("connexion.php");
	Abrimos la conexión a la base de datos

$query = "delete from encabezado where idformato ='$idformato';"; $result = mysql_query($query) or die('Consulta fallida: ' . mysql_error()); $query2 = "delete from rescasoprueba where encabezado ='$idformato'"; $result2 = mysql_query($query2) or die('Consulta fallida: ' . mysql_error());	Querys de eliminación de datos en las tablas encabezado y rescasoprueba

`// Confirmamos que el registro ha sido eliminado con exito`
`echo "`
`<p>Los datos han sido borrados con exito.</p>`
`<p><a href='javascript:history.go(-1)'>VOLVER ATRÁS</a></p>`
`";`
`?>`
	Confirmamos que el registro ha sido eliminado con éxito

Save.php

`<?php`
`$project = $_POST["project"];`
`$useCase = $_POST["useCase"];`
`$testCase = $_POST["testCase"];`
`$creationDate = $_POST["creationDate"];`
`$executester = $_POST["executester"];`
`$version = $_POST["version"];`
`$resultSystem = $_POST["resultSystem"];`
`$resultHope = $_POST["resultHope"];`
`$observation = $_POST["observation"];`
`$successful = $_POST["successful"];`
`$image = $_POST["image"];`
 	
`if ($successful==1){`
`$successful="SI";`
`}else if($successful==2){`
`$successful="NO";`
`}`
	Se obtienen variable mediante el método POST

`// Abrimos la conexion a la base de datos` `include("connexion.php");`	Abrimos la conexión a la base de datos

`$nummaxc= mysql_query('select max(idformato) from encabezado');`
`$nummaxr= mysql_result($nummaxc,0);//`
`imprime el numero de registros existentes en encabezado`
	Se realiza consulta de id máximo de encabezado para la próxima inserción

$query = "INSERT INTO encabezado (`idformato`,`fecha`, `idtester`, `idcasouso`, `insumos`, `idtipoformato`) VALUES ($nummaxr+1,'$creationDate', '$executester', '$useCase', '$version', '1');"; $result = mysql_query($query) or die('Consulta fallida: ' . mysql_error()); $query2 = "INSERT INTO rescasoprueba (`resultadosistema`, `resultado`, `encabezado`, `resultadoesperado`, `observaciones`) VALUES ('$resultSystem', '$successful', $nummaxr+1, '$resultHope', '$observation');"; $result2 = mysql_query($query2) or die('Consulta fallida: ' . mysql_error());	
// Confirmamos el registro ha sido insertado echo " Los datos han sido guardados con exito. VOLVER ATRÁS "; ?>	Confirmamos que el registro ha sido insertado con éxito