Instituto Tecnológico Superior Zacatecas Sur

INFORME FINAL DE RESIDENCIA PROFESIONAL

APLICACIÓN DE TÉCNICAS DE EVALUACIÓN DE SOFTWARE EN LA EMPRESA E-QUALLITY

PRESENTA:

JOSÉ MANUEL ROMERO TAPIA

CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Tlaltenango de Sánchez Román, Zac., Diciembre de 2016

AUTORIZACIÓN DE ENCUADERNACIÓN.

Instituto Tecnológico Superior Zacatecas Sur

INFORME FINAL DE RESIDENCIA PROFESIONAL

APLICACIÓN DE TÉCNICAS DE EVALUACIÓN DE SOFTWARE EN LA EMPRESA E-QUALLITY

PRESENTA:

JOSÉ MANUEL ROMERO TAPIA

CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Tlaltenango de Sánchez Román, Zac., Diciembre de 2016

AGRADECIMIENTOS Y DEDICATORIAS.

Doy gracias a dios por permitirme llegar a éste momento de mi vida, porque siempre está a mi lado acompañándome en este camino lleno de obstáculos.

Agradezco a mis padres Tomas Romero Gálvez y María Elena Tapia Sedado y a mis hermanos Erik Armando Romero Tapia, Natalia Romero Tapia y María Belem Romero Tapia, por brindarme siempre su apoyo en toda mi carrera, porque siempre estuvieron conmigo en todo momento, brindándome su mano para cualquier circunstancia y salir adelante y no estancarme, sus palabras de aliento, su motivación, sus regaños, su ayuda, su presión, su cariño, me sirvieron para luchar por éste sueño que se veía muy lejano, pero gracias a dios y el apoyo de todos ustedes se está cumpliendo una meta importe en mi vida.

Agradezco a mi novia Sofía Berumen Ruvalcaba por apoyarme siempre y estar conmigo en esta etapa tan importante en mi vida, acompañándome en este camino lleno de obstáculos y nunca dejarme solo. Gracias por estar siempre conmigo.

Agradezco infinitamente a mis profesores ya que, sin su sabiduría, experiencias, conocimientos no fuera posible cumplir esta meta tan importante para mí, donde hubo regaños, consejos, reflexiones, los cuales me sirvieron para salir adelante, siempre estuvieron guiándome por este camino para llegar hasta el final. Gracias por resolver mis dudas, por ayudarme en problemas, por regañarme y hacerme ver lo era bueno y lo malo, sin ustedes no fuera posible cumplir este objetivo. Gracias siempre los llevaré presente en mi vida.

ÍNDICE

ÍNDICE DE GRÁFICAS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La calidad que contiene el software desarrollado alrededor del mundo constituye uno de los procesos más significativos en el ámbito de la Ingeniería de Software, debido a que para ser un sistema bien estructurado y desarrollado se requiere que cumplan normas de calidad significativas, bajo alguno de los modelos ya existentes como MOPROSOFT, ISO, CMMI, TMM y TPI.

La residencia profesional es el último paso para la carrera del estudiante y donde tiene la finalidad de que el alumno aplique el conocimiento adquirido de las aulas del Instituto Tecnológico Superior Zacatecas Sur en el ámbito laboral, así como adquirir experiencia profesional, contribuyendo al desarrollo tecnológico.

La intención es conocer el proceso de calidad que se le realiza al software para que éste cumpla las normas de calidad, la empresa e-Quallity S.A. de C.V. desarrolló una metodología la cual consta de evaluar el software, así mismo generar un diagnóstico de calidad; en el desarrollo de la residencia profesional" Aplicación de técnicas de evaluación de software en la empresa e-Quallity" se busca contribuir en la verificación de productos de software aplicando diferentes pruebas, técnicas y herramientas que ayuden a realizar un proceso más eficiente y certero, en busca de resultados; dichos resultados se presentan en gráficas y tablas para un mejor análisis y comprensión.

En el primer capítulo se presentan los aspectos generales, se hace la justificación y luego se detallan los problemas a resolver, para dar pie al objetivo

general y de igual manera los objetivos específicos, como también los alcances y limitaciones del proceso llevado a cabo en la residencia.

En el segundo capítulo se describe la información teórica vinculada a las pruebas de software, para ayudar a comprender mejor los términos y temas relacionados con las actividades que se realizaron en la empresa.

En el tercer capítulo de éste informe se describe de manera detallada las actividades que se realizaron en la empresa e-Quallity S.A. de C.V, así como también los resultados obtenidos en base a las actividades desarrolladas, recalcar que una parte de la información que se maneja en este capítulo sigue un protocolo de privacidad.

En el cuarto capítulo se plasman las conclusiones y recomendaciones de las actividades y resultados obtenidos durante la estancia de la residencia.

CAPÍTULO I

“ASPECTOS GENERALES”

1.1 JUSTIFICACIÓN

Actualmente, para poder competir en este mercado globalizado y altamente competido de las tecnologías de información y comunicación, la calidad no representa un valor agregado, sino más bien un prerrequisito indispensable; hablando específicamente del área de desarrollo de software, el proceso de pruebas ha pasado de ser sólo una necesidad a ser un proceso necesario para ahorrar tiempo y costo en la elaboración del software.

Existen diferentes técnicas y herramientas que son implementadas en los proyectos como apoyo administrativo facilitando el proceso de pruebas. La empresa e-Quallity S.A de C.V utiliza diferentes herramientas de apoyo entre ellas Testlink, Mantis, Microsoft Office Project cuyo objetivo es encontrar defectos en el software, donde se reconocen conceptos claves como lo son: escenario de prueba, caso de prueba, tipos de prueba, técnicas de prueba entre las cuales se derivan dos fundamentales técnicas de prueba estáticas y dinámicas, así como también el concepto de anomalías.

Estas razones, aunado a la exigencia de especialización para un exitoso ejercicio profesional, fueron los motivos que respaldan la decisión para desarrollar el proyecto “Aplicación de técnicas de evaluación de software en la empresa e-Quallity”, dicha empresa es dedicada y es altamente especializada en el área de Ingeniería de Pruebas de Software.

1.2 OBJETIVO GENERAL

Aplicar las diferentes herramientas y técnicas de prueba de software a los proyectos que se encuentran implementándose dentro de la empresa e-Quallity S.A. de C.V.

OBJETIVOS ESPECÍFICOS:

- Aprender a realizar actividades para la validación y verificación de software como son: revisión de documentos, diseño de casos de prueba y matrices de escenarios, ejecución de casos de prueba, pruebas exploratorias, reporte de anomalías, reportes de actividades.
- Aprender a evidenciar las áreas de mejora de los sistemas bajo prueba para colaborar oportunamente en su optimización.
- Aprender a documentar riesgos que se pueden presentar a lo largo de un proyecto.
- Conocer las técnicas y metodologías de prueba de software.
- Determinar de manera óptima las áreas de mejora respecto a pruebas de los sistemas desarrollados.
- Documentar los riesgos que se pueden presentar en el desarrollo de un sistema de software.
- Aplicar el conocimiento adquirido de los procesos de pruebas.
- Utilizar herramientas para la agilización de los procesos de prueba.
- Documentar resultados de las pruebas de calidad realizadas al software.

1.3 CARACTERIZACIÓN DEL ÁREA EN QUE SE PARTICIPÓ

El área en la que participo dentro de la empresa e-Quallity de S.A. de C.V. fue “Prestación de servicios profesionales”. La empresa se especializa en pruebas de software, la cual es pionera en la disciplina en el campo industrial, brindando servicios como diagnóstico de calidad de software, pruebas funcionales, outsourcing de pruebas de software, insourcing de pruebas de software, entre muchos más servicios.

MISIÓN DE LA EMPRESA

Ofrecer a organizaciones y personas capacitación y servicios especializados en prueba de software, que les ayuden a determinar y reducir riesgos de liberar o adquirir productos de baja calidad, costos de mantenimiento y re trabajo.

VISIÓN DE LA EMPRESA

En 5 años, ser el corporativo líder en prueba de software de América Latina, con los más altos estándares de calidad, que genera grandes beneficios en sus relaciones con clientes, colaboradores y socios.

VALORES:

Excelencia

Crecimiento como equipo

Innovación

Liderazgo

ORGANIGRAMA DE LA EMPRESA

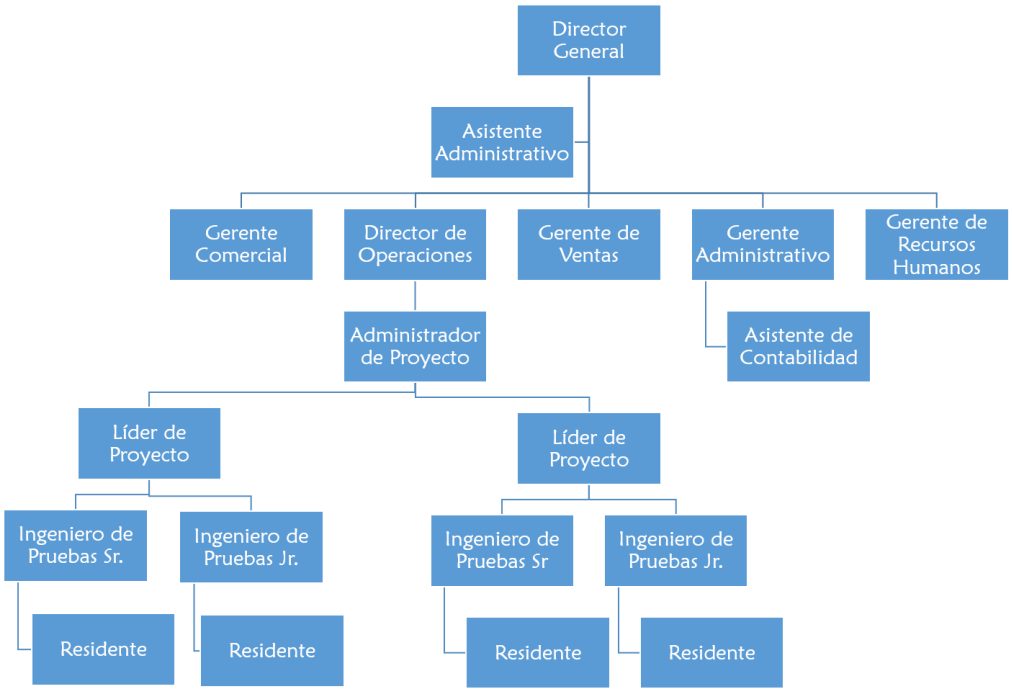


Figura 1. Organigrama de la empresa.

NOMBRE DE LA EMPRESA

e-Quallity.

RAZON SOCIAL

e-Quallity de S.A de C.V.

DOMICILIO DE LA EMPRESA

Centro del Software- Plaza del Ángel.

Av. López Mateos No. 2077, Oficina Z2, Código postal: 44510, Guadalajara, Jalisco. México

Teléfono: +55 (33) 3030-6909

1.4 PROBLEMAS A RESOLVER

La calidad de un software en un concepto el cual cuesta mucho adquirir por parte del desarrollador en su software, verificar si el software cuenta con las condiciones necesarias para alcanzar la calidad suficiente y éste saque su mayor provecho. Saber cuál es la mejor forma para diagnosticar la calidad del producto, así mismo como saber diagnosticar la madurez, operatividad y buscar la forma de reducir tiempos de entrega y costos de mantenimiento de un software.

Por lo tanto, se presenta el problema a resolver: “No es posible diagnosticar la calidad de un producto de software, para verificar si tiene las condiciones mínimas de operación”.

1.5. HIPÓTESIS

Al aprender y aplicar distintas técnicas de pruebas e implementarlas en proyectos de software, siguiendo los procesos, metodologías y conocimientos técnicos necesarios, permitirá comprender metodologías de pruebas y comprobar que con evaluar un proyecto de software es posible determinar su calidad, madurez y operatividad.

Por lo tanto, la hipótesis es: “La aplicación de distintas técnicas de pruebas en proyectos, de software, permite determinar la calidad, madurez y operatividad.”

1.6 ALCANCES

- Capacitación de procesos de prueba y de metodologías que apliquen a proyectos.
- Realizar la exploración de un sitio WEB, diseñar y ejecutar casos de prueba para un sistema de información.
- Conocimiento de los documentos que se utilizan dentro de cada una de las fases de la metodología.
- Poner en práctica lo adquirido sobre las capacitaciones de la metodología de pruebas en un proyecto interno de la empresa.
- Investigar temas para célula de investigación enfocado al ámbito de pruebas.

LIMITACIONES

- Los resultados obtenidos sólo se podrán externar de manera estadística.
- El período de tiempo que se tiene planeado para desarrollar el proyecto de residencias profesionales, concebido del 01 de agosto al 18 de noviembre del 2016.
- Dentro del desarrollo del proyecto se dedicarán 1 meses a capacitación, 1 mes para pruebas exploratorias y 2 meses al trabajo en un proyecto interno de la empresa e-Quallity S.A de C.V.

CAPÍTULO II

“FUNDAMENTO TEÓRICO”

2.1 PRUEBAS DE SOFTWARE

Las pruebas del software es una tarea crucial y a la vez muy desafiante dentro del proceso de desarrollo de software. Las pruebas permiten encontrar errores y problemas del software contra la especificación del mismo y cumple un rol fundamental en el aseguramiento de la calidad del producto. (Barrientos, 2014)

Entre los tipos de pruebas que se pueden realizar al software están las pruebas de unidad, de carga, integración y funcionales. Cada una de ellas tiene distintos objetivos y son realizadas en diferentes etapas del ciclo de desarrollo. (Barrientos, 2014)

En el primer tipo mencionado, se desarrollan pruebas a componentes individuales de un sistema de software. Los desarrolladores especifican y codifican pruebas para cubrir todos o al menos una parte significativa de los posibles estados/configuraciones del artefacto o unidad de software, para simular el entorno del componente y descubrir la presencia de errores o “bugs”. (Barrientos, 2014)

Dado que escribir todas esas pruebas de forma manual es costoso, las pruebas de unidad son generalmente realizadas de manera ineficiente o simplemente dejadas de lado. (Barrientos, 2014)

El panorama es aún peor, más allá del esfuerzo, porque las pruebas no pueden ser usadas para probar la usencia de errores en el software sino tan sólo la presencia. Por eso es necesario atacar el problema desde diferentes enfoques, cada uno teniendo sus fortalezas y ventajas. Actualmente existen muchas técnicas para hacer testing de software, y la mayoría de ellos se basan en la automatización de

pasos o caminos de ejecución, con valores fijos o componentes predefinidos (hard-coded) o estáticos, y condiciones específicas. (Barrientos, 2014)

2.1.1 TIPOS DE PRUEBAS

2.1.1.1 PRUEBAS UNITARIAS

Las pruebas unitarias tienen ese nombre debido a que se prueba la funcionalidad de cada método o función; únicamente contemplando la lógica que debe realizar y excluyendo la convivencia con otras clases o sistemas. Esto conlleva a que si este método utiliza funciones de otros métodos tenemos que ingeniárnosla para que esta lógica no afecte nuestro método que estamos probando. (Java & Arch, Blog, 2012)

El objetivo principal de las pruebas unitarias es asegurar que el código que hemos programado funciona realmente como lo esperamos; no precisamente refleja lo que el cliente nos pidió (aunque debería de ser), para eso existen otro tipo de pruebas como las de aceptación de usuario. Por lo tanto, las pruebas unitarias están destinadas para el aseguramiento del desarrollador y no para el usuario final. (Java & Arch, Blog, 2012)

Existen varios frameworks para poder realizar estas pruebas unitarias, tales como la familia XUnit: Junit para java CppUnit para C, PhpUnit y Nunit para .Net. Existe otra familia de frameworks igualmente famosa como TestNG. (Java & Arch, Blog, 2012)

2.1.1.2 PRUEBAS DE REGRESIÓN

Existen también las pruebas de regresión las cuales pueden ser pruebas de aceptación o pruebas funcionales, pero se realizan sobre una aplicación ya terminada para tener una base de su funcionalidad básica. (Java & Arch, Blog, 2012)

Este tipo de pruebas se realizan principalmente sobre una aplicación que no se tiene documentación o ningún tipo de pruebas, por lo que estas pruebas se deben realizar antes de hacerle una modificación a la aplicación. Por lo tanto, sólo hay que hacer pruebas del funcionamiento básico y no exhaustivo preferentemente. (Java & Arch, Blog, 2012)

2.1.1.3 PRUEBAS FUNCIONALES

Las pruebas funcionales son pruebas similares a la de aceptación con la diferencia que sí son técnicas y por lo tanto deberán incluir cada uno de los requerimientos funcionales. En general, las pruebas de aceptación incluyen pruebas funcionales y no funcionales. (Java & Arch, Blog, 2012)

Para este tipo de pruebas es necesario que se realicen también bajo ambientes controlados para poder manipular los datos de entrada y salida. Por ejemplo, en una base de datos que deberá alimentarse con datos ya conocidos antes de iniciar. Esto servirá para que las pruebas siempre funcionen y no marquen error cada vez que se ejecutan debido a que la base ya cambió de estado. (Java & Arch, Blog, 2012)

2.1.1.4 PRUEBAS DE INTEGRACIÓN

Las pruebas de integración son similares a las funcionales o las de aceptación, pero sobre repositorios reales, datos reales y con la interacción real con otros sistemas o componentes. (Java & Arch, Blog, 2012)

Principalmente sirven para asegurar que la implementación en algún ambiente en particular (UAT o Producción) ha sido exitosa. Aquí el término exitoso implica que la aplicación corre al 100% por lo tanto, todos los artefactos han sido instalados y configurados correctamente y que también su convivencia con otros componentes como base de datos, mensajería entre otros es correcto. Estas pruebas automatizadas ayudan mucho en tareas repetidas y aburridas cada vez que se hacen implementaciones. (Java & Arch, Blog, 2012)

2.1.1.5 PRUEBAS NO FUNCIONALES

Este tipo de pruebas (Java & Arch, Blog, 2012) están destinadas a probar que los requerimientos no funcionales han sido satisfechos completamente. Lo que en general se intenta probar son funcionalidades en el comportamiento bajo estrés, alta demanda, respuesta bajo los límites de tiempo establecidos por el cliente, ciertos criterios de look & feel.

2.1.1.6 PRUEBAS DE ESTRÉS

Las pruebas de estrés se proponen encontrar errores debido a recursos bajos o completitud de recursos. Poca memoria o espacio en disco puede revelar defectos en el sistema que no son aparentes bajo condiciones normales. Otros defectos pueden resultar de incluir recursos compartidos, como bloqueos de base de datos o

ancho de banda de la red. Las pruebas de estrés identifican la carga máxima que el sistema puede manejar. (Londoño, 2005)

El objetivo de esta prueba es investigar el comportamiento del sistema bajo condiciones que sobrecargan sus recursos. No debe confundirse con las pruebas de volumen: un esfuerzo grande es un pico de volumen de datos que se presenta en un corto período de tiempo. (Londoño, 2005)

Puesto que la prueba de esfuerzo involucra un elemento de tiempo, no resulta aplicable a muchos programas, por ejemplo, a un compilador o a una rutina de pagos. (Londoño, 2005)

Aunque muchas pruebas de esfuerzo representan condiciones que el programa encontrará realmente durante su utilización, muchas otras serán situaciones que nunca ocurrirán en la realidad. Esto no implica, sin embargo, estas pruebas no sean útiles. (Londoño, 2005)

Si se detectan errores durante estas condiciones “imposibles”, la prueba (Londoño, 2005) es valiosa porque es esperar que los mismos errores puedan presentarse en situaciones reales, algo menos exigentes.

2.1.1.7 PRUEBAS DE CALIDAD DE CÓDIGO

Este tipo de pruebas sirven para garantizar que la calidad del código es realmente óptima y que la probabilidad de tener errores o bugs en la codificación es mínima (nunca dejarán de existir los errores de software), pero al menos podemos hacer lo pertinente para disminuir la probabilidad). (Java & Arch, Blog, 2012)

2.1.1.8 PRUEBAS EXPLORATORIAS

Una estrategia básica para realizar las pruebas exploratorias es concebir un plan de ataque general, pero que permita desviarse de él por periodos cortos de tiempo e intereses diversos. Cem Kaner (Kaner, 1999) lo denomina el principio del “tour bus”, las personas bajan del bus y conocen los alrededores con visiones variadas. La clave es no perderse el tour entero. (Beatriz, s.f.)

El éxito o el fracaso de las pruebas exploratorias están íntimamente ligadas con lo que sucede en la mente del ingeniero de pruebas. Algunas habilidades deseables en los ingenieros es la capacidad de analizar el producto y evaluar los riesgos, utilizar herramientas y tener un pensamiento crítico acerca de lo que se sabe al momento de diseñar las pruebas. También es importante que los testers tengan la capacidad de distinguir lo observado de lo inferido; ya que las inferencias podrían inducirlos a no realizar pruebas que evidencien vulnerabilidades del producto. El uso de heurísticas desempeña un rol importante en la producción de ideas variadas. Una heurística es un mecanismo mental para generar pruebas variadas y acordes a las características del producto que se está probando, por lo que es deseable que los testers las tengan presentes al momento de realizar pruebas exploratorias. (Beatriz, s.f.)

En general, las pruebas meramente exploratorias requieren testers con mucha experiencia. Como ventaja se encuentra que es barato y rápido, como inconveniente, que, según algunos autores, produce escasa documentación y no facilita las medidas de cubrimiento. (Beatriz, s.f.)

Las pruebas exploratorias presentan una estructura externa fácil de describir (ver figura 2). Durante un período de tiempo un tester interactúa con un producto para cumplir una misión y reportar los resultados. Una misión describe qué se probará del producto, los tipos de incidentes que se buscan y los riesgos involucrados. La misión puede ser elegida por el tester o serle asignada por el líder de testing. (Beatriz, s.f.)

Los elementos externos básicos son: tiempo, tester, producto, misión y reportes. Esta aparente simplicidad permite desplegar una amplia gama de posibilidades para la aplicación de las pruebas exploratorias.(Beatriz, s.f.)

Figura 2. Módulos de pruebas exploratorias

2.1.1.9 PRUEBAS DE CAJA NEGRA

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la

estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. (Ecu Red, 2016)

La prueba de caja negra no es una alternativa a las técnicas de prueba de la caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la caja blanca. Muchos autores consideran que estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad (ver figura 3). Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien. (Ecu Red, 2016)

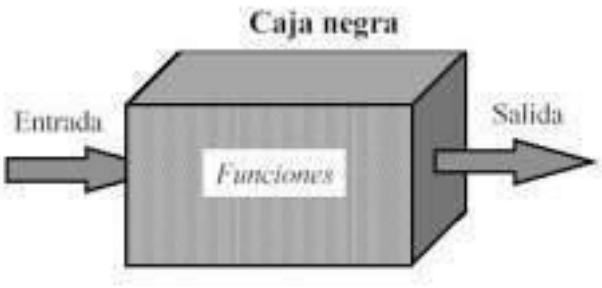


Figura 3. Tipo de prueba caja negra.

2.1.1.10 PRUEBAS DE CAJA BLANCA

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:

- Garantiza que se ejecute por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Funcionen todas las decisiones lógicas en las vertientes verdadera y falsa.
- Actúen todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican al software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad (ver figura 4). (Ecu Red, 2016)

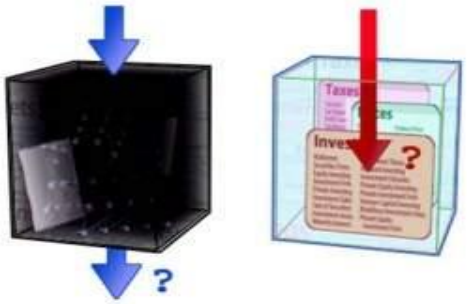


Figura 4. Tipo de prueba caja blanca.

2.1.2 TIPOS DE PRUEBAS PARA VIDEOJUEGOS

- Pruebas de funcionalidad: Esta prueba se enfoca a que el funcionamiento del videojuego cumpla con las expectativas de los requerimientos.
- Pruebas de compatibilidad: Como su nombre lo dice, es probar que el videojuego tenga compatibilidad con diferentes dispositivos.
- Pruebas de localización: Esta prueba busca que el videojuego cumpla con los estándares, por ejemplo, se instala en una región donde se habla español, hacer que el videojuego se cambié al lenguaje de dicha región.
- Pruebas beta: Esta prueba se realiza cuando se publica una parte del videojuego al público, para que dichos usuarios encuentren el número mayor de errores. (Juega Libre, 2015)
- Pruebas de carga: Estas pruebas se realizan para probar los límites del videojuego, poner a toda potencia donde se aloja y que se comporte de manera estable. (Juega Libre, 2015)
- Pruebas multijugador: Probar que el videojuego corresponda a las peticiones de otro usuario o viceversa, probar que los métodos de conexión trabajen y que se trabaje en paralelo con los diferentes usuarios. (Juega Libre, 2015)

2.2 MODELOS DE CALIDAD DE SOFTWARE

2.2.1 MODELO DE PRUEBA DE SOFTWARE TMM

TMM es la abreviación usada para Test Maturity Model. El TMM es un proceso de madurez del test que fue originalmente creado por el Illinois Institute of Technology como guía para la mejora de procesos de testing y como complemento al CMM(i). (SoftQaNetwork, 2007)

La estructura del TMM está basada en el CMM y en sus fases, ya que consiste también en 5 niveles que reflejan el grado de madurez del test. Para cada nivel de madurez, hay definidos un número de procesos. Los cinco niveles del TMM ayudarán a una organización a determinar la madurez de sus procesos de test y a identificar los pasos a seguir para introducir las mejoras necesarias para lograr niveles mayores de madurez. (SoftQaNetwork, 2007)

Los niveles de madurez son:

- Nivel 1 – Inicial: El test está sumido en un proceso caótico o simplemente no hay prueba alguna.
- Nivel 2 – Definición: Se caracteriza por la utilización de técnicas básicas de test que identificarán si el software está de acuerdo con sus especificaciones. Estructuración del proceso de test, planificaciones y estrategias.
- Nivel 3 – Integración: Se caracteriza por el test está completamente integrado en el ciclo de vida del software y es reconocido a todos los niveles del modelo en V. El plan de test está terminado, las estrategias han sido determinadas en función del previo análisis de riesgos basados en los requisitos.

- Nivel 4 – Gestión y Medición: El test es un proceso medido y cuantificado. La usabilidad es uno de los atributos de calidad utilizados en el test de software.
- Nivel 5 – Optimización, Prevención de defectos y control de calidad: Se caracteriza por mecanismos precisos para que el test pueda ser mejorado continuamente. En este nivel se utilizan procedimientos para seleccionar y evaluar herramientas de test. (SoftQaNetwork, 2007)

Cada día los sistemas son más complejos, haciendo necesario que se tomen mejoras en la calidad de los procesos para mejorar así la calidad final del producto. El modelo TMM proporcionará y proceso de desarrollo de software más eficiente y efectivo. El objetivo será a prevención y no en la detección. (SoftQaNetwork, 2007)

2.2.2 MODELO DE PRUEBA DE SOFTWARE TPI

Las pruebas de software a menudo consumen mucho tiempo y dinero. Muchas organizaciones se han dado cuenta que mejorando sus procesos de pruebas pueden solucionar estos problemas. Sin embargo, en la práctica resulta mucho más difícil saber qué medidas tomar para mejorar y controlar el proceso. (Javier, 2007)

El modelo TPI está basado en las mejores prácticas de la industria relativas a la mejora del proceso de pruebas. El modelo incluye guías prácticas para evaluar el nivel de madurez de prueba de una organización, así como los pasos para mejorar el proceso. El modelo se compone de varias áreas clave, que constituyen la base para mejorar y estructurar el proceso de test, algunos de los cuales son los siguientes:

-Estrategia de prueba.

-Modelo de ciclo de vida.

-Estimación y planeación.

-Especificación de técnicas de pruebas.

-Técnicas de pruebas estáticas.

2.3 CICLO DE VIDA DE LAS PRUEBAS

El ciclo de vida de las pruebas de software emplea el conocido modelo del ciclo de vida del software, cascada mejorado; con la variación de que la prueba es aplicada a cada una de las fases del ciclo de vida (ver figura 5). (e-Quallity, 2008)

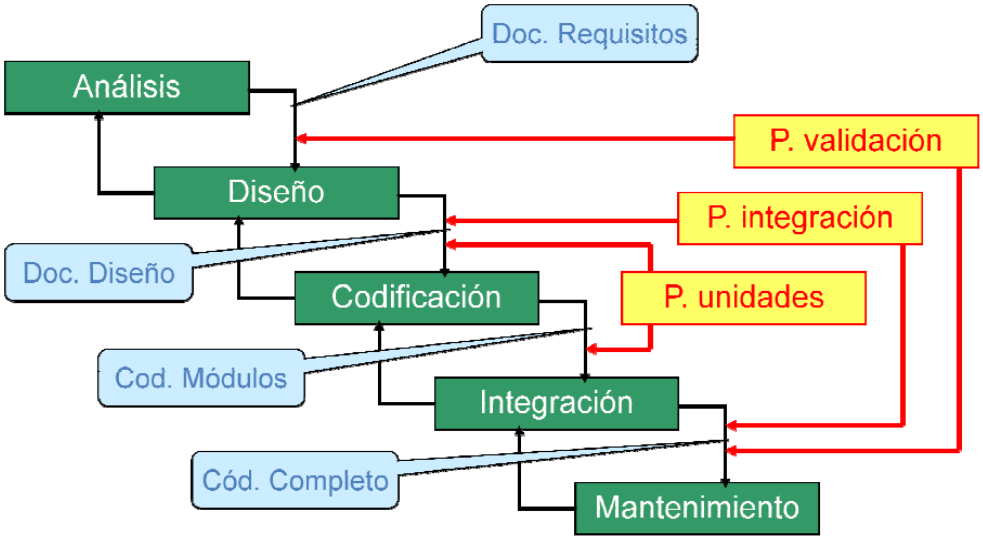


Figura 5. Ciclo de vida de las pruebas mejorado.

El ciclo comienza con la fase de análisis, de la cual se obtiene la documentación de los requerimientos del proyecto a probar, a dicha documentación se le aplican pruebas de validación. (e-Quallity, 2008)

En la fase de diseño, se obtiene la documentación respectivamente a la cual se le aplican pruebas de integración y pruebas de unidad. (e-Quallity, 2008)

En la fase de codificación, cuya salida son los módulos individuales que integraran el sistema, a cada uno de estos módulos se les aplican pruebas de unidad. (e-Quallity, 2008)

Para la fase de integración, se obtiene el código completo del sistema, al que se le aplican pruebas de integración y de validación. (e-Quallity, 2008)

Para la última fase de mantenimiento, no hay una definición estándar para determinar su salida, sin embargo, en caso de que el sistema sea sometido a un mantenimiento se le podrán aplicar pruebas de validación, integración y pruebas de unidad, según corresponda. (e-Quallity, 2008)

Además, es posible, gracias a la disposición del propio modelo, regresar a cualquier fase del ciclo de vida, y por supuesto aplicar las pruebas que sean necesarias. (e-Quallity, 2008)

Ahora bien, el ciclo de vida que sigue el proceso de pruebas se muestra en la figura 6. (e-Quallity, 2008)

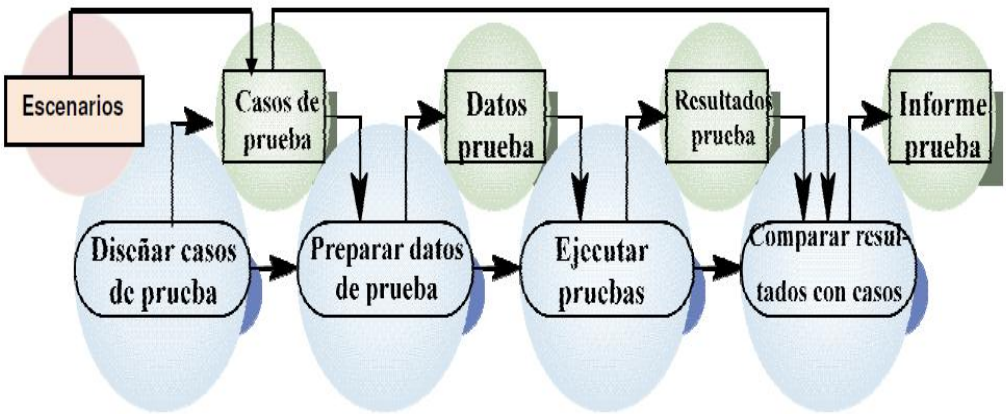


Figura 6. Proceso de Pruebas.

El proceso de Ingeniería en pruebas comienza con el diseño de escenarios, para a partir de estos, comenzar con el diseño de casos de prueba, un caso de prueba se diseña preparando los datos de prueba, para posteriormente aplicarlos y ejecutar las pruebas, se deben documentar los resultados de las pruebas para compararlos con los resultados que se esperaban de la ejecución de los casos de prueba y así poder emitir un informe de las pruebas aplicadas.(e-Quallity, 2008)

2.4 HERRAMIENTAS

El proceso de prueba de software es un proceso técnico especializado que requiere de profesionales altamente capacitados, el cual; debe poseer la capacidad de evaluar el sistema desde un punto de vista crítico siguiendo un procedimiento estructurado formalmente. No obstante, para facilitar las actividades desempeñadas a lo largo del proyecto; existen herramientas de apoyo administrativo que les permite a los ingenieros en pruebas, realizar las diferentes actividades de manera no sólo efectiva sino eficiente. (e-Quallity, 2008)

2.4.1 MANTIS

Es un software que constituye una solución completa para gestionar tareas en un equipo de trabajo. Es una aplicación OpenSource realizada con php y mysql que destaca por su facilidad y flexibilidad de instalar y configurar. Esta aplicación se utiliza para probar soluciones, hacer un registro de alteraciones y gestionar proyectos. La figura 7 muestra la herramienta Mantis. (Wikipedia, 2016)

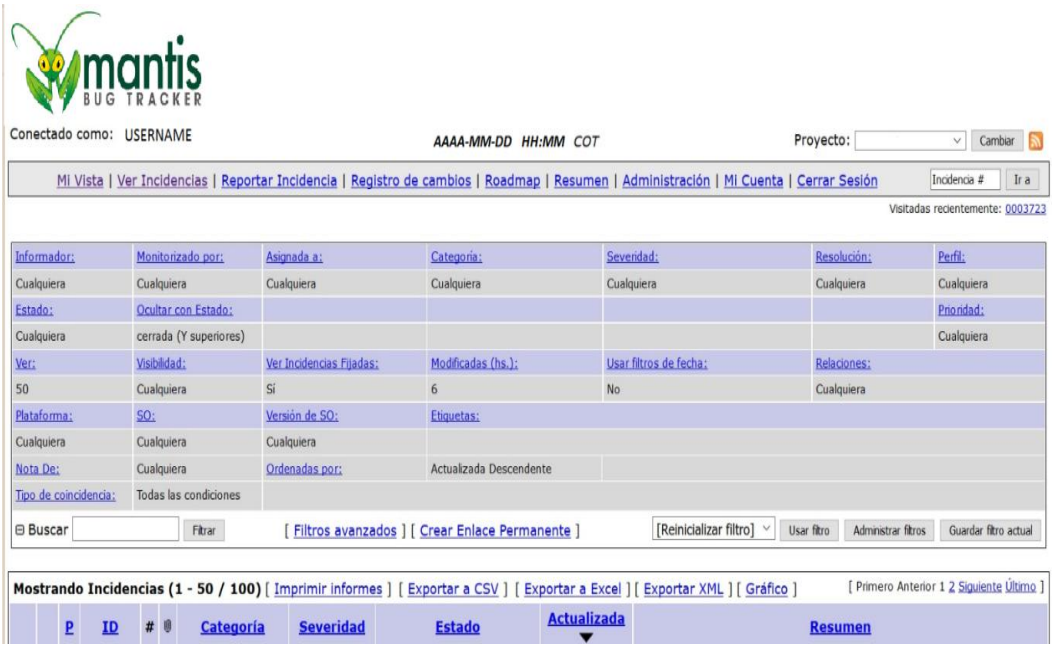


Figura 7. Ejemplo de registro de anomalías.

Esta aplicación permite la creación de diversas cuentas de usuario desde las cuales se puede informar de los bugs detectados. Con Mantis se puede dividir un proyecto en varias categorías, lo cual permite hacer un seguimiento más exacto de éste. El flujo de trabajo también se puede configurar desde la propia herramienta, de forma que muestra que causa el problema, quien puede analizarlos y quien puede atenderlos. (Wikipedia, 2016)

2.4.2 MICROSOFT OFFICE PROJECT

Es un software de la suite Microsoft Office usado para la administración de proyectos diseñado, desarrollado y comercializado por Microsoft para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo.

Como tal, Microsoft Project es utilizado para llevar un control de actividades, así mismo añadiéndole el tiempo que está planeado desarrollar la actividad (ver figura 8). (Rosa, 2013)

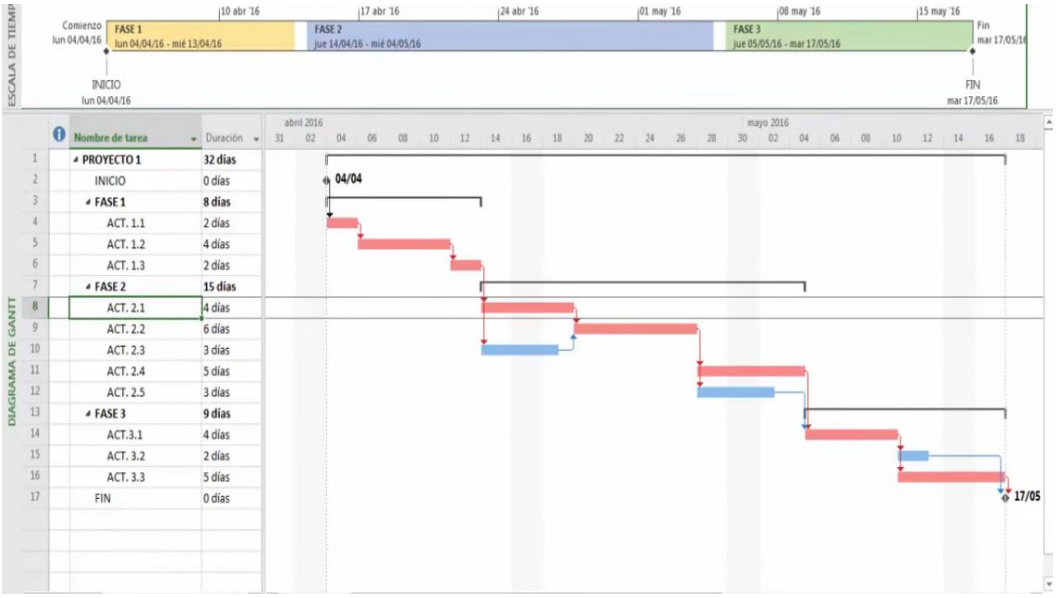


Figura 8. Ejemplo de plan de actividades.

2.5 TÉRMINOS USADOS COMUNMENTE EN EL ÁREA DE TESTING

2.5.1 ESCENARIOS DE PRUEBA

Un escenario de pruebas representa un arreglo o configuración detallada del software que aborda un proceso de negocio específico. Se pueden establecer escenarios por tipos de usuario. Ejemplo: escenario uno corresponde a un usuario "normal", el escenario dos corresponde a un usuario "supervisor" del software. También se pueden definir los escenarios de acuerdo a los tipos de pruebas que se realizarán. Por ejemplo: el escenario uno refiere a las pruebas funcionales, el escenario dos refiere a las pruebas de integración. Como se observa, un mismo

procedimiento de prueba puede ser aplicado en distintos escenarios de prueba. (Multidiciplinaria, Proyectos e Ingenieria, 2016)

La identificación de los escenarios de prueba permite establecer cruces por medio de matrices de casos versus escenarios que ayudan a aumentar la cobertura de las pruebas; es decir, identificar todos los casos de prueba, cuya ejecución certificarán la calidad del sistema. Estos escenarios de pruebas juntan en un documento. (Multidiciplinaria, Proyectos e Ingenieria, 2016)

- Matriz de escenarios: Se le llama así al documento que tiene la función de reunir cada uno de los escenarios de pruebas que aborda un proceso de negocio específico. (Multidiciplinaria, Proyectos e Ingenieria, 2016)

2.5.2 CASOS DE PRUEBA

Un caso de prueba o “test case” es, en ingeniería del software, un conjunto de condiciones o variables bajo las cuáles un analista determinará si una aplicación, un sistema software (software system), o una característica de éstos es parcial o completamente satisfactoria. (Wikipedia: La enciclopedia libre, 2016)

Se pueden realizar muchos casos de prueba para determinar que un requisito es completamente satisfactorio. Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito a menos que un requisito tenga requisitos secundarios. En ese caso, cada requisito secundario deberá tener por lo menos un caso de prueba. Algunas metodologías como RUP (Rational Unific Process) recomiendan el crear por lo menos dos casos de prueba para cada requisito, la prueba positiva de los

requisitos y la prueba negativa. Si la aplicación es creada sin requisitos formales, entonces los casos de prueba se escriben basados en la operación normal de programas de una clase similar. (Wikipedia: La enciclopedia libre, 2016)

Lo que caracteriza un escrito formal de caso de prueba es que hay una entrada conocida y una salida esperada, los cuales son formulados antes de que se ejecute la prueba. La entrada conocida debe probar una precondición y la salida esperada debe probar una postcondición. Los casos de prueba escritos, incluyen una descripción de la funcionalidad que se probará, la cual es tomada ya sea de los requisitos o de los casos de uso, y la preparación requerida para asegurarse de que la prueba pueda ser dirigida. Los casos de prueba escritos se recogen generalmente en una suite de pruebas mejor conocido como colección de casos de pruebas para la ejecución. (Wikipedia: La enciclopedia libre, 2016)

2.5.3 ANOMALÍAS

Anomalía es cualquier condición que se desvíe de las expectativas basadas en las especificaciones de requisitos, documentos de diseño, documentos de usuario, estándares o de la percepción o experiencia de alguien. Las anomalías pueden ser encontradas durante, revisiones, proceso de pruebas, análisis, compilación, o uso de productos de software o documentación aplicable. (Globe Testing, s.f.)

2.5.5 TÉCNICAS DE PRUEBA

Conjunto de estrategias que permiten encontrar de manera formal las entradas para el diseño de los casos de prueba, existen dos tipos de técnicas de prueba; las estáticas y las dinámicas. (Globe Testing, s.f.)

- Técnicas de Prueba Estáticas: Estrategias que permiten hacer revisiones a la aplicación sin someterla a ejecución, dentro de ellas se encuentra el Peer Review (Inspecciones entre pares), las Inspecciones y los Walkthroughs. (Globe Testing, s.f.)

- Técnicas de Prueba Dinámicas: Estrategias que permiten diseñar casos de prueba que prueben el sistema en ejecución, dentro de ellas se encuentran clases de equivalencia, valores al límite, tablas de decisión, los grafos causa-efecto y la exploración. (Globe Testing, s.f.)

    - Clases de Equivalencia: También llamadas partición de equivalencia es la porción del dominio de una entrada o una salida para la cual se asume que el comportamiento de un componente o sistema, basado en la especificación, es el mismo. (Globe Testing, s.f.)

    - Valores al Límite: Es el valor de entrada o de salida que se encuentra en la frontera de una partición de equivalencia o a la mínima distancia incremental a cualquier lado de la frontera, por ejemplo, el valor mínimo o máximo de un rango. (Globe Testing, s.f.)

    - Tablas de Decisión: La técnica de prueba es muy aplicable cuando la lógica a probar está basada en decisiones, principalmente si se puede expresar la lógica en forma de reglas tales como:

        - Si A es mayor que X entonces…

        - Si el cliente C tiene deuda entonces…

CAPÍTULO III

“DESARROLLO DE ACTIVIDADES”

3.1 PROCESO DE RESIDENCIA

3.1.1 CAPACITACIÓN Y METODOLOGÍA

En la residencia profesional, nos dieron diferentes capacitaciones para que aprendiéramos las metodologías en la ingeniería de pruebas (ver figura 9).

Figura 9. Capacitación de personal.

Primeramente, se llevó la capacitación en la cual nos dieron a conocer un escenario general de la metodología que maneja la empresa, como también, que otros procesos utilizan en el área de prueba de software.

Se realizaron cinco capacitaciones que abarcaron todas las fases de la metodología.

En la primera capacitación se dio a conocer la fase inicial de la metodología, la cual consta de analizar el software, basándose en los procesos que maneja dicha fase, como también documentos únicos de la empresa.

En el paso de los días, se realizó la segunda capacitación la cual se vio completamente la segunda fase de la metodología, ésta consta de ver que se necesita para realizar dichas pruebas, así mismo ésta fase maneja documentos que el ingeniero de pruebas o también llamado Tester se basará para realizar el proceso de pruebas.

En la tercera fase de la metodología es la más extensa de las 4 fases que se maneja dentro de la empresa, ésta fase se llevó a cabo en dos capacitaciones, dicha fase tiene la función de desarrollar todo lo que se ha realizado en las dos fases anteriores, cómo aplicar las pruebas al software.

En la última fase de la metodología, se impartió la última capacitación de las cinco que fueron impartidas, con el objetivo de verificar qué se hizo bien o mal en las fases anteriores, generando conclusiones y documentos que indican el fin de la metodología, así mismo ver que el sistema que se probó, cumpla con normas del proceso que son requeridas para que sea un software con calidad.

3.1.2 PROYECTO PILOTO

Se dio a conocer el software llamado “Sistema de Mantenimiento”, el cual se aplicó todo el conocimiento adquirido de las capacitaciones que fueron impartidas por la misma empresa, desde la primera fase hasta la última fase de la metodología, esto con la finalidad de llevar a práctica cada fase.

“Sistemas de Mantenimiento” consta de un proyecto interno de la misma empresa, el cual es entregado para realizar las prácticas en la organización después de haber llevado la capacitación necesaria para posteriormente aplicarlo a proyectos reales.

3.1.3 REVISIÓN DE DOCUMENTOS

Al finalizar las capacitaciones, el líder de proyectos, realizó la entrega del software “Sistema de Mantenimiento” el cual venía acompañado de varios documentos que hacían referencia al funcionamiento de dicho proyecto; cabe mencionar que sólo se aplicará todo el proceso de pruebas a una parte del software completo, es decir a un módulo asignado por parte del tutor.

Se tuvo que llevar una revisión completa de toda la documentación, aplicando la primera fase de la metodología la cual consiste analizar, observar cual es el alcance, saber cuáles son las metas, los objetivos de toda la documentación del proyecto al cual se le aplicarán las pruebas posteriormente.

3.1.4 DEFINICIÓN DE ACTIVIDADES Y ALCANCES

En la segunda fase del proceso de pruebas, se genera o se planea que actividades se realizarán a lo largo de las pruebas, con el fin de llevar un proceso bien estructurado y sobre todo aplicar la calidad desde cuando se planean las actividades hasta cuando se empiezan a ejecutar las actividades.

Lo primordial es generar un plan de actividades plasmando que fechas del periodo son las posibles para realizar la actividad. Las tareas resultantes fueron en base a la fase anterior donde se generó el análisis de los requerimientos del proyecto.

Para llevar a cabo un plan bien detallado fue necesario utilizar la herramienta Microsoft Project, esto con el fin de administrar bien las actividades que se realizaran en la siguiente fase donde se accionan los diseños de casos de pruebas.

La figura 10 muestra un ejemplo de cómo se administró las actividades planeadas que se realizarán a lo largo del proceso de pruebas, como también el periodo plasmado en un diagrama de Gantt en el cual muestra todo lo que se estará llevando a cabo en todo el proceso de pruebas hasta su terminación.

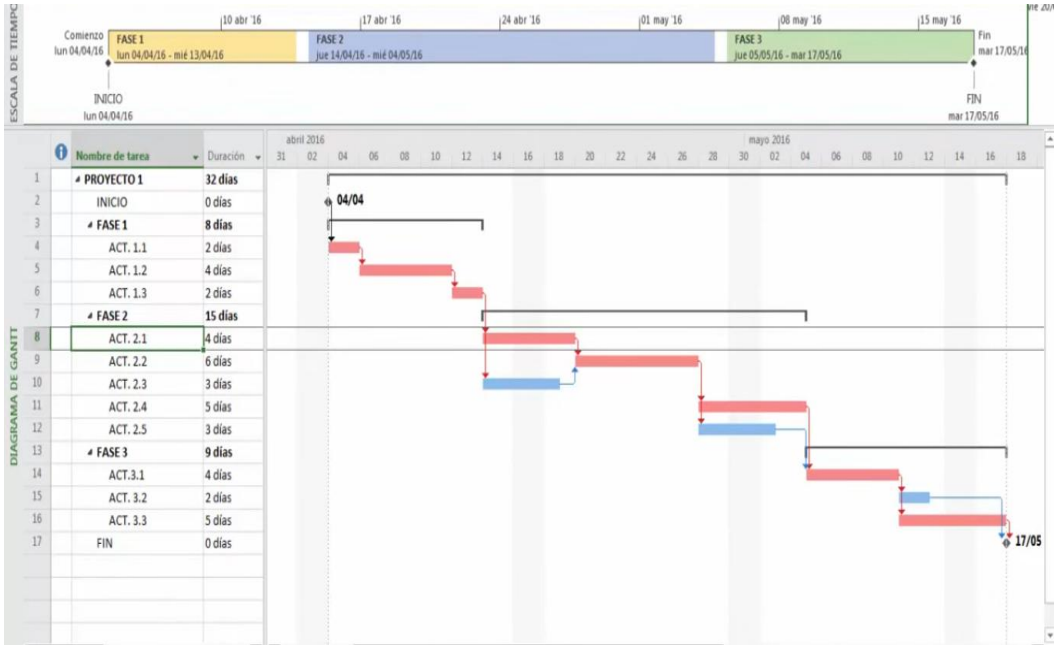


Figura 10. Ejemplo 2 de plan de actividades en Project.

3.1.5 DISEÑO DE CASOS DE PRUEBA Y MATRICES DE ESCENARIOS

En la tercera fase se pone en práctica el proceso de pruebas de acuerdo a la capacitación que se dio al inicio de la residencia, llevando a cabo el desarrollo de diseñar los casos de prueba y matrices de escenarios del módulo que se asignó al principio del proyecto.

Cada sección del sistema contiene varios submódulos, los cuales realizan funciones importantes del sistema. Al módulo que se le realizo todo el proceso de pruebas siguiendo la metodología de la empresa se plasma en la tabla 1.

Tabla 1. Módulo del proyecto “Sistema de Mantenimiento”.

1. Movimientos
1.1 Movimientos pendientes
1.2 Movimientos iniciados
1.3 Movimientos terminados

En el módulo “Movimientos” que fue asignado, se le realizó todo el procedimiento de pruebas desde la primera hasta la tercera fase.

3.1.5.1 TÉCNICAS DE PRUEBAS DISEÑADAS DEL MÓDULO ASIGNADO

Para el módulo “Movimientos” del proyecto “Sistema de Mantenimiento” se obtuvieron 3 submódulos, de los cuales se diseñaron 3 documentos que contienen las diferentes técnicas de pruebas a realizar correspondientes a cada submódulos, el submódulo “Movimientos pendientes” se realizó la técnica “Clases de equivalencia”, en el submódulo “Movimientos iniciados” también se realizó una técnica llamada “Valores al límite” y en el último submódulo “Movimientos terminados” se realizaron

tres, dos de ellas fueron las técnicas de los submódulos anteriores y una más llamada “Tablas de decisiones” (ver tabla 2).

Tabla 2. Número de técnica de prueba por submódulo.

Módulo “Movimientos”	No. de técnicas de pruebas	Técnicas de Pruebas
Movimientos pendientes	1	-Clases de equivalencia
Movimientos iniciados	1	-Valores al limite
Movimientos terminados	3	-Clases de equivalencia -Valores al limite -Tablas de decisión

3.1.5.2 MATRICES DE ESCENARIOS DE PRUEBAS DISEÑADOS

Para el módulo “Movimientos”, en la fase donde se diseñan el proceso de pruebas, se tuvo un total de 3 matrices de escenarios de pruebas, cada uno de estas matrices tiene un total de 7 escenarios haciendo un total de 21 escenarios por las tres matrices que se diseñaron en base a un análisis a las funcionalidades encontradas en las pantallas de cada uno de los submódulos (ver tabla 3).

Tabla 3. Matrices de escenarios diseñados.

Matrices de los escenarios	No. Escenarios diseñados
Matriz_pendientes	7
Matriz_iniciados	7
Matriz_terminados	7

3.1.5.3 CASOS DE PRUEBAS DISEÑADOS

Para el módulo “Movimientos”, la fase donde se diseñan el proceso de pruebas, se obtuvieron un total de 3 documentos de casos de pruebas, el total de los casos de pruebas que se diseñaron fue un total de 159 casos de pruebas por los tres documentos correspondientes a cada uno de los submódulos, estos casos de pruebas se diseñaron en base a los escenarios que se obtuvieron en cada submódulo (ver tabla 4).

Tabla 4. Casos de prueba por submódulo.

Módulo	Matrices de escenarios	Casos de Pruebas	No. de casos de pruebas
Movimientos pendientes	Matriz_pendientes	CasodePrueba_pendientes	55
Movimientos iniciados	Matriz_iniciados	CasodePrueba_iniciados	55
Movimientos terminados	Matriz_terminados	CasodePrueba_terminados	49

3.1.6 DISEÑO Y EJECUCIÓN DE CASOS DE PRUEBA

El proceso de prueba se realizaba con buen ritmo ya que se le dedicaba gran tiempo del día para probar el proyecto “Sistema de Mantenimiento”.

En la parte de diseño y ejecución se llevó a cabo una temática en la cual consistía en que los módulos del sistema fueran sorteados por parte del líder de proyectos, con la razón de ejecutar los casos de prueba que otro residente diseñó, en el sorteo, como resultado fue ejecutar los casos de prueba diseñados por Jesús Abel Hernández Hernández, con la finalidad de poner en práctica la redacción y

ortografía, más que nada interpretar los casos de prueba diseñados por terceras personar. El módulo a ejecutar se muestra en la tabla 5.

Tabla 5. Módulo a ejecutar.

2. Catálogos
2.1 Equipo & partes
2.1.1 Detalles
2.1.2 Mantenimiento
2.1.3 Orden de trabajo

Cabe mencionar que este módulo también pertenece al mismo proyecto “Sistema de Mantenimiento”, recalcar que fue divido el proyecto en módulos y por lo cual estos fueron divididos a los residentes.

El módulo anterior consistía en 3 submódulos a los cuales también se les aplicaron las primeras fases del proceso de pruebas. En el submódulo “Detalles” se diseñaron 74 casos de prueba, en el submódulo “Mantenimiento” con 49, en el submódulo “Orden de trabajo” con 24, generando un total de 147 casos de pruebas de las funciones principales de los submódulos que fueron ejecutados, posteriormente genero el resultado del analizar del módulo por parte del residente Jesús Abel Hernández

En los submódulos a ejecutar también se diseñó un total de 3 matrices de escenarios con 3 escenarios de prueba cada uno haciendo un total de 9 escenarios por todos (ver tabla 6).

Tabla 6. Casos de prueba a ejecutar.

Submódulo a ejecutar	No. matrices de escenarios	No. Escenarios de prueba	No. Casos de prueba
Detalles	1	3	74
Mantenimiento	1	3	49
Orden de trabajo	1	3	24

Las listas de sentencias son criterios específicos que contiene los módulos con el fin de validar y verificar componentes del sistema, por ejemplo, que se siga el mismo formato de letra, mismo estándar de diseño, las listas también son utilizadas para las pruebas de software.

Para las interfaces de los submódulo a ejecutar se generó una lista de 29 sentencias por cada submódulo a probar (ver tabla 7).

Tabla 7. Sentencias diseñadas para posible ejecución.

Módulo a ejecutar	No. sentencias de interfaz gráfica
Detalles	29
Mantenimiento	29
Orden de trabajo	29

3.1.6.1 EJECUCIÓN DE CASOS DE PRUEBA

Para llevar a cabo la ejecución de los casos de prueba fue necesario contar con los softwares instalados en la computadora haciendo esto posible el seguimiento de pasos necesarios para poner en práctica los diseños de los casos de prueba.

Se instaló el software del proyecto “Sistema de Mantenimiento” junto con el software manejador de base de datos “SQL Server” ocupando un rol importante para su completa funcionalidad. Para comenzar la ejecución de los casos, una vez terminado el diseño de todos los casos, la líder otorgó un usuario y un password a cada uno de los residentes, para iniciar la ejecución.

La finalidad de ejecutar los diseños de prueba es encontrar el número mayor posible de anomalías que se pudieran encontrar en el proyecto “Sistema de Mantenimiento”, así mismo generar el reporte de anomalías.

Al ejecutar, siguiendo los diseños de los casos de prueba, se van encontrando anomalías en el sistema, generando un registro de anomalías llevando una referencia en que caso de prueba se encontró y tomando evidencia para saber en qué parte se fue ubicada.

Se ejecutaron los 74 casos del primer submódulo “Detalles” encontrando un total de 10 anomalías, para el submódulo “Mantenimiento” se ejecutaron 49 encontrando un total de 7 anomalías, al submódulo “Orden de trabajo” se encontraron 12 anomalías (ver tabla 8).

Tabla 8. Anomalías encontradas en el proyecto “Sistema de Mantenimiento”.

Módulos/Submódulos	**Anomalías encontradas**
Detalles	10
Mantenimiento	7
Orden de trabajo	12

3.1.7 REPORTE DE ANOMALÍAS

Una vez terminado de ejecutar los casos de prueba de cada uno de los submódulos, se siguió trabajando con el reporte de anomalías, pero esta vez el registro se realizó con la ayuda de la herramienta de software “Mantis”, la cual necesita registrar las anomalías en la nube, teniendo una mejor administración de éstas (ver figura 11).

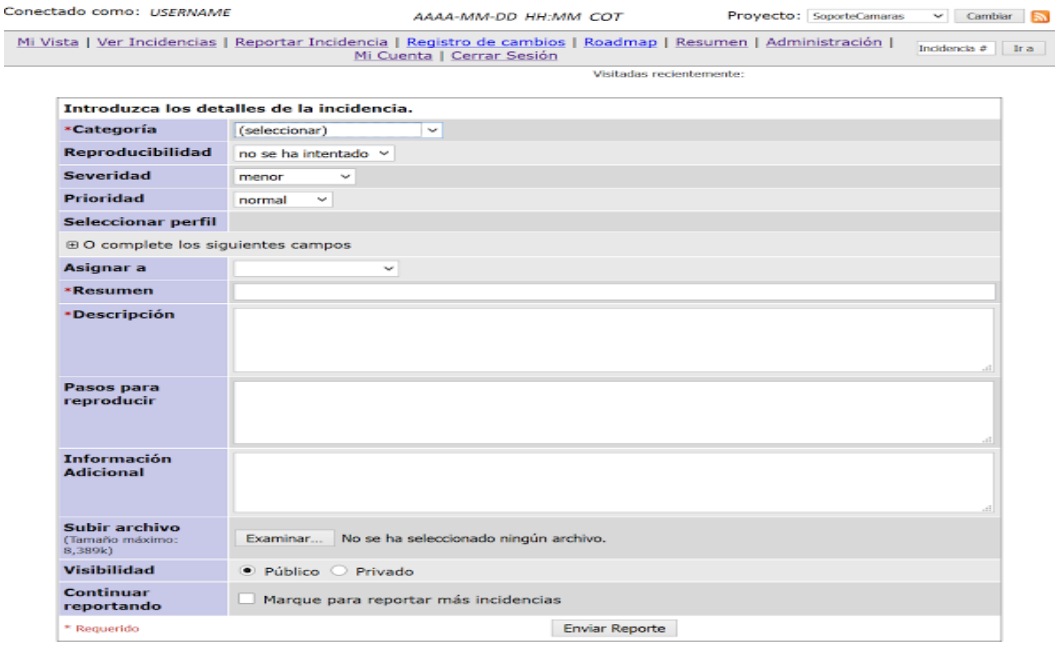


Figura 11. Ejemplo reporte de anomalías a Mantis.

Existen más herramientas que permiten llevar un registro de anomalías, considerando necesidades o simplemente gusto del usuario, se mostrarán algunas herramientas, que al igual que Mantis, ayudan a gestionar el registro de anomalías de un sistema, como se muestran en la figura 12, figura 13 y figura 14.

Figura 12. Bugzilla (herramienta para registro de errores).

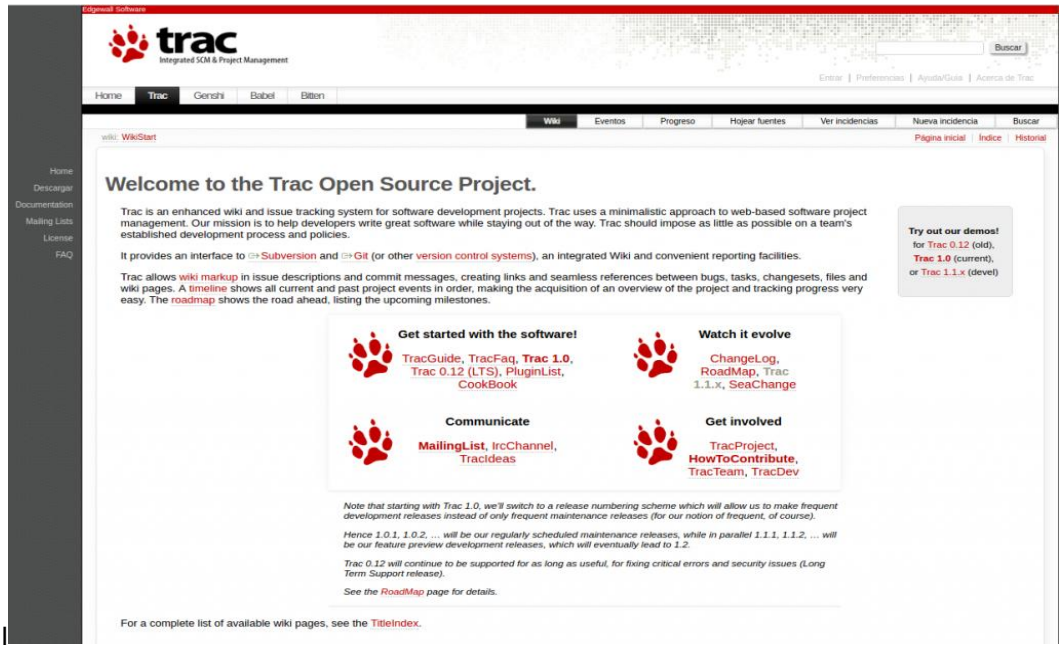


Figura 13. Trac (herramienta para seguimiento de errores).

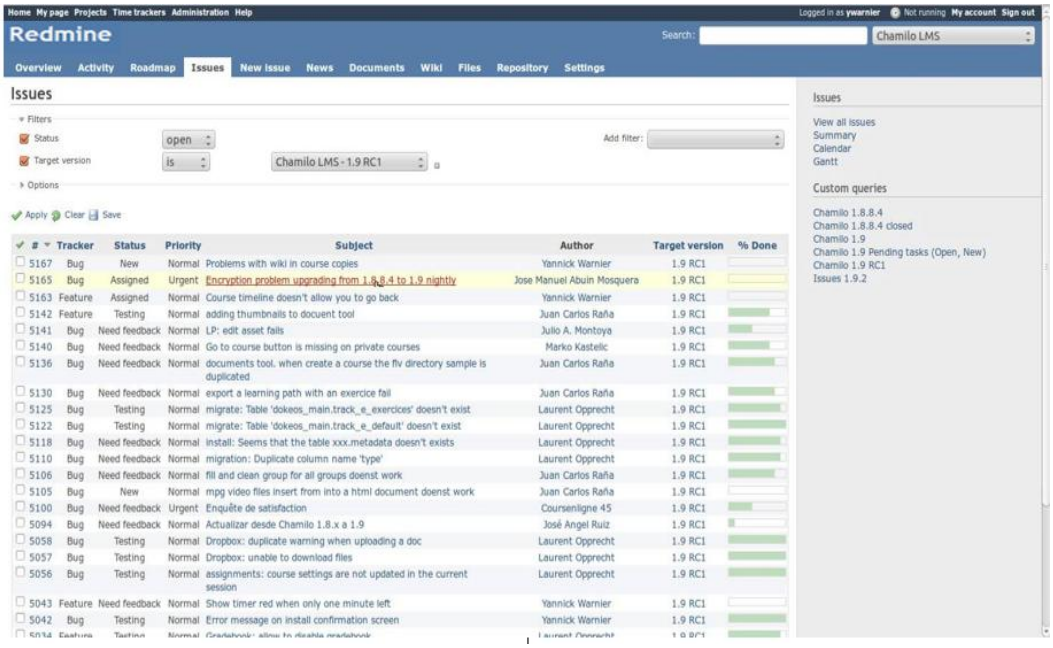


Figura 14. Redmine (herramienta para seguimiento de errores).

Cada una de estas herramientas, aparte de que ayudan al seguimiento de anomalías, también tienen la gran funcionalidad de gestionar proyectos, con características diferentes, pero todos con el mismo objetivo en común.

Al término de realizar el registro de anomalías, el siguiente proceso fue generar conclusiones y sacar estadísticas las cuales ayudan a entender mejor los resultados obtenidos de la ejecución.

3.1.8 ELABORACIÓN DE INFORME

En la fase final, el proyecto lleva su último proceso el cual consta de generar el resumen total de “Sistema de Mantenimiento”, mencionando que resultados se obtuvieron en tanto diseño como la ejecución de los casos de prueba.

Dicha fase se registró el total de anomalías mediante estadísticas las cuales ayudan a desarrollar un diagnóstico de la totalidad del proyecto haciendo hincapie en la calidad con la que cuenta el proyecto que se ejecutó.

Esta fase comunmente el ingeniero que prueba, plasma lo que adquirió mediante el proceso llevado a cabo, generando una experiencia la cual ayuda al ingeniero a desarrollar un punto importante sobre la calidad de un software.

3.1.9 RIESGOS EN EL PROCESO DE PRUEBAS

En cada proyecto que se desarrolla, se prueba o se documenta. se realizan en base a fases o etapas, el proceso del proyecto siempre esta propenso a riesgos que se puedan presentar independiente sea el motivo.

A lo largo del proceso de pruebas al que le fue aplicado al proyecto “Sistema de Mantenimiento”, se presentaron diferentes problemas, los cuales en un principio no se tomaron en cuenta los riesgos que se podrían presentar.

Esto llevo a que el proyecto presentará problemas los cuales atrasaron bastante el proceso, como tal se buscó la manera de mitigarlos en el momento que se activaron (ver tabla 9).

Tabla 9. Problemas durante la ejecución.

Descripción del problema	Solución del problema
Se tuvo problemas con el software de la base de datos ya que el proyecto “Sistema de Mantenimiento” tenía problemas	Se desinstaló el software de la base de datos y posteriormente se reinstaló para mitigar el problema de conexión.

de conexión entre ambos.	
Al redactar los diseños de casos de prueba se tenía problemas con la descripción del mismo.	Se practicó la redacción con ejercicios otorgados por el tutor, escuchar consejos que provenían de los mismos trabajadores.

3.1.10 PRUEBAS EXPLORATORIAS

Esta actividad se desarrolló en base al concepto pruebas exploratorias llevando a cabo una inspección detallada a una página web de publicidad otorgada por parte del asesor de la empresa.

Dicha actividad se realizó junto a otros dos compañeros residentes que también elaboraron las prácticas profesionales en la empresa.

Se realizó una inspección tipo exploratoria a la página web de publicidad, con un total de 3 módulos con funcionalidades principales que contiene dicha página. A estos tres módulos de la página web, se les aplicó todo el proceso de pruebas exploratorias, en la tabla 10 se muestran los módulos de la página:

Tabla 10. Módulos para pruebas exploratorias a la página web.

Inicio
Directorio
Usuarios

Tomando en cuenta los módulos a probar de la página web de publicidad, mediante la forma de exploración, se manipuló la página web de forma que el

objetivo principal trataba de buscar anomalías siguiendo el orden de los módulos correspondientes a la página.

3.1.10.1 REPORTE DE ANOMALÍAS DE PRUEBAS EXPLORATORIAS

Al terminar las pruebas exploratorias que se le realizó a la página web de publicidad, al igual que al proyecto “Sistema de Mantenimiento”, se hizo el registro de las anomalías encontradas en la página, agrupándolas es sus diferentes categorías. Las formas de los resultados son semejantes al del proyecto de mantenimiento, la única diferencia fue que estas anomalías no se registraron en la herramienta de Mantis, simplemente es su respectivo documento.

3.1.10.2 REPORTE DE ACTIVIDADES

El reporte de actividades, es una forma de llevar un registro diario de las actividades que se realizan dentro de la empresa, llevando un control total, así mismo indicando que actividad y cuánto tiempo se llevó en cada una. Este reporte se empezó a trabajar desde el primer día de trabajo hasta la fecha.

El reporte de actividades se trabaja diariamente (ver figura 15). Al término de la semana el reporte se envía al líder de proyectos en la última hora de trabajo con la finalidad de revisar las actividades realizadas junto con el tiempo de duración.

Nombre: José Manuel Romero Tapia
Líder(es) de Proyecto(s): Carlos Pérez

Actividades	Semana 1					Total
	01/09/2016	02/09/2016	03/09/2016	04/09/2016	05/09/2016	Semanas 4
Actividad 1	3.0					3.0
Actividad 1.1	0.5					0.5
Actividad 2	0.5					0.5
Actividad 2.1	2.0	1.0				3.0
Actividad 2.2	2.0	1.0				3.0
Actividad 3		0.5				0.5
Actividad 4		1.0				1.0
Actividad 5		2.0				2.0
Actividad 6		1.0	2.0	1.0		4.0
Actividad 6.1			2.0			2.0
Actividad 6.2		1.5	4.0	3.0	5.5	14.0
Actividad 7				2.0		2.0
Actividad 8				2.0	2.5	4.5
Minutos	480.0	480.0	480.0	480.0	480.0	**2400.0**
Horas	8.0	8.0	8.0	8.0	8.0	**40.0**

Figura 15. Reporte de actividades diarias.

3.1.11 CÉLULA DE INVESTIGACIÓN

La célula de investigación es una forma de fomentar la investigación dentro de la empresa, abrir el panorama más allá del enfoque que se tiene la empresa, buscar las nuevas tendencias de la tecnología de la información que están sobresaliendo en el mercado actual.

Las pruebas de software se pueden aplicar a todo aquello que cuenta con software, entonces el objetivo de la célula de investigación es buscar las tendencias actuales y ver de qué forma se le puede aplicar pruebas a esas tendencias del mercado.

En la empresa, fue otorgado un tema a investigar, enfocado a una tendencia del mercado actual, con la finalidad de crear un proceso, al cual se le pueda aplicar pruebas.

El tema a investigar fue “Metodologías de Videojuegos”, ya que el mercado del videojuego está creciendo a grandes pasos, se busca abrir un camino en la empresa para buscar la forma de aplicar pruebas a un videojuego (ver figura 15).

Figura 16. Metodología de videojuegos.

La célula se inició con una investigación buscando si existe una forma o una metodología que seguir para aplicar las pruebas a los videojuegos.

Las pruebas que se le pueden aplicar a un videojuego pueden variar dependiendo el género, que tipo de consolas, sobre la funcionalidad, pruebas de multijugador, pruebas beta, pruebas de cumplimiento, entre muchas más.

Se ha trabajado con la adaptación de varios documentos de la empresa, con el fin de que los documentos que se necesiten estén listos para cuando se llegue un proyecto de videojuegos.

Actualmente la célula de investigación se sigue trabajando y se sigue buscando la mejor forma de aplicar una metodología de pruebas a un videojuego.

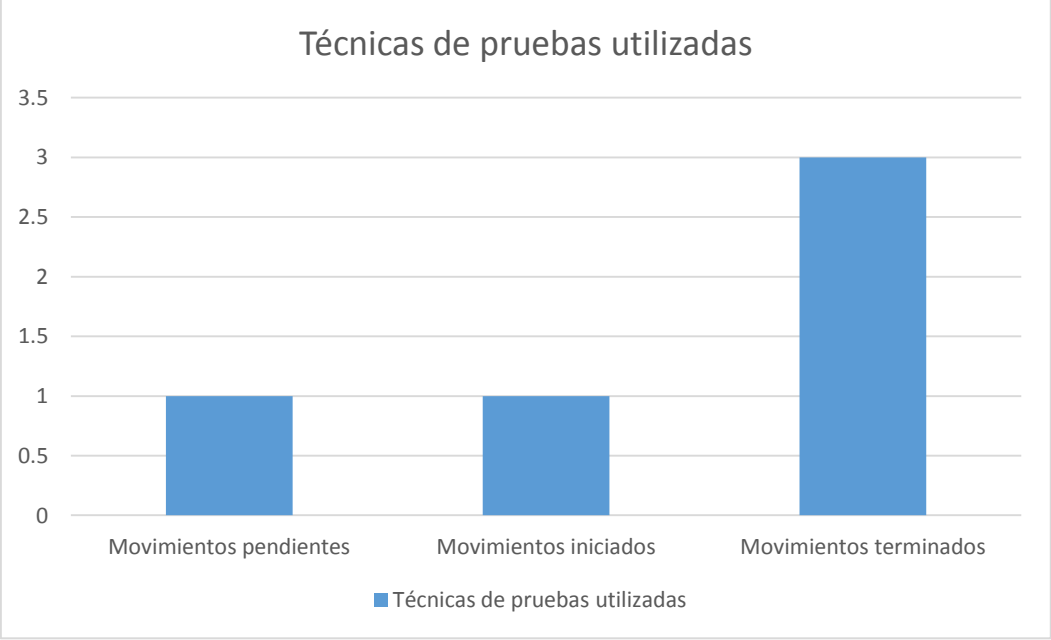
3.2 RESULTADOS, PLANOS, GRÁFICOS, PROTOTIPOS Y PROGRAMAS

3.2.1 RESULTADO DE DISEÑO DE CASOS DE PRUEBA Y MATRICES DE ESCENARIOS DEL PROYECTO PILOTO

Para entender mejor los diseños de los casos y matrices de escenarios fue necesario utilizar estadísticas representadas mediante gráficas las cuales ayudan entender mejor el diseño de los casos de prueba y matrices de escenarios de prueba, estas fueron realizadas mediante los resultados obtenidos durante el proceso de diseño de pruebas que fue llevado a cabo en la fase de diseño y ejecución.

3.2.1.1 TÉCNICAS DE PRUEBAS APLICADAS

Las técnicas de pruebas fueron utilizadas para identificar cuantas técnicas se utilizaron para el proceso de prueba para cada uno de los submódulos representadas en la gráfica 1 y con los datos correspondientes en la tabla 11.



Gráfica 1. Técnicas de pruebas utilizadas.

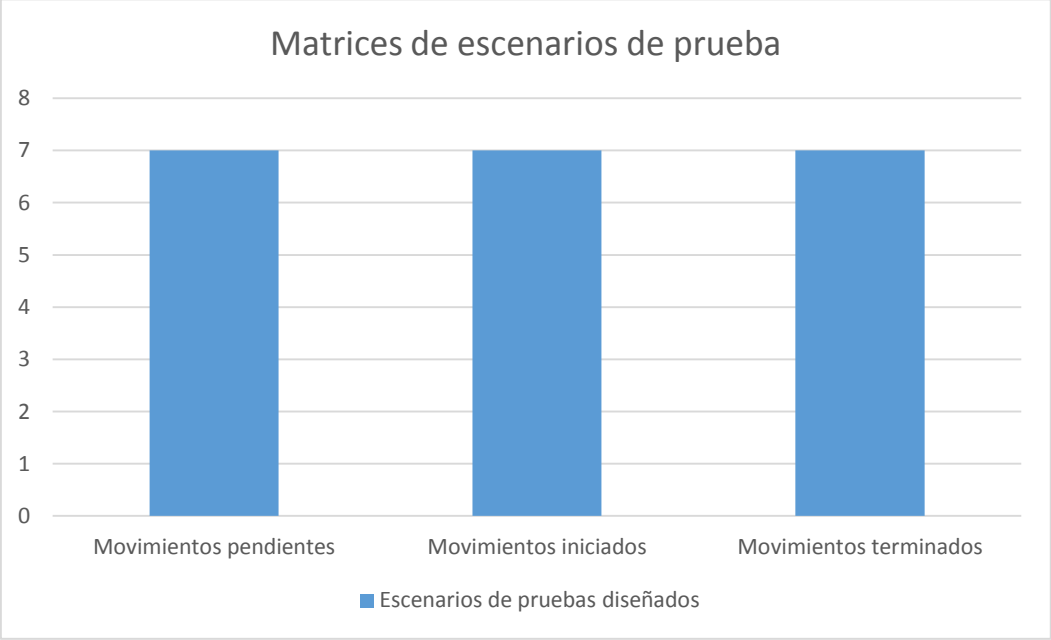
Tabla 11. Técnicas de pruebas utilizadas.

Módulo/Submódulo	Técnicas de prueba	Porcentaje
Movimientos pendientes	1	20%
Movimientos iniciados	1	20%
Movimientos terminados	3	60%
Total	5	100%

3.2.1.2 RESULTADOS DE MATRICES DE ESCENARIOS DISEÑADOS

Las matrices de escenarios es la parte en la cual el ingeniero de pruebas se basa para poder diseñar los casos prueba los cuales son utilizados para guiarse y así poder ejecutarlos a la par del manejo del sistema.

Como tal se diseñaron escenarios de prueba los cuales fueron un total de 21 escenarios, repartidos de 7 en cada submódulo (ver gráfica 2 y tabla 12).



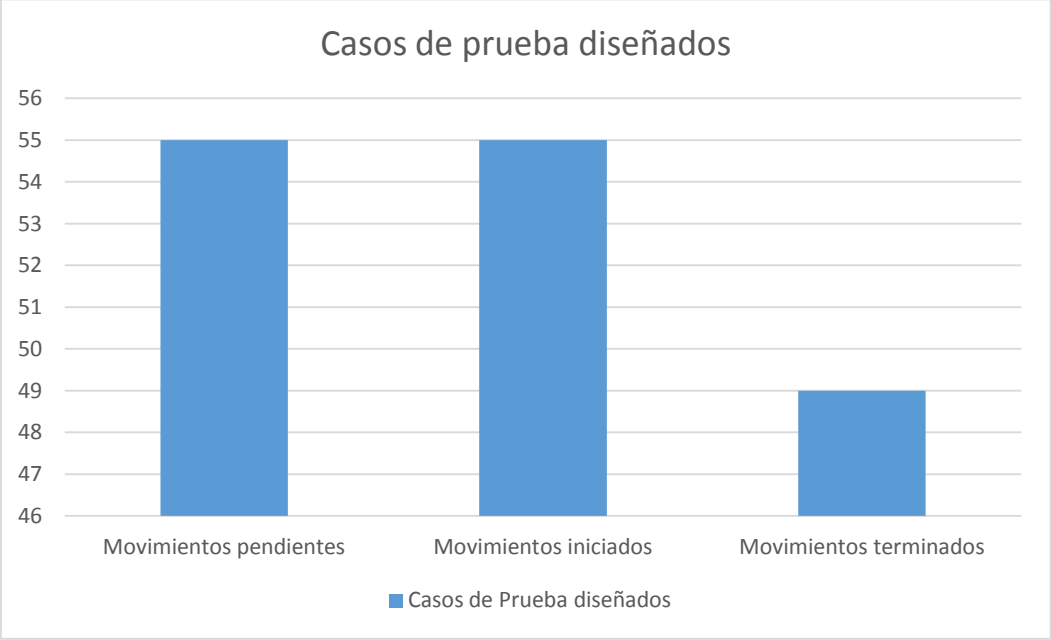
Gráfica 2. Escenarios de pruebas diseñados.

Tabla 12. Escenarios de pruebas y su porcentaje.

Módulo/Submódulo	Matrices de escenarios	Porcentaje
Movimientos pendientes	7	33.3%
Movimientos iniciados	7	33.3%
Movimientos terminados	7	33.3%
Total	21	100%

3.2.1.3 RESULTADOS DE CASOS DE PRUEBAS DISEÑADOS

Siguiendo el proceso de pruebas, al haber generado las matrices de escenarios, es fácil guiarse para crear o en otras palabras diseñar los casos de prueba, en cuanto al proyecto de mantenimiento se basó para diseñar un número de casos para cada submódulo, con una total de 159 casos de prueba diseñados por los tres submódulo (ver gráfica 3 y tabla 13).



Gráfica 3. Casos de pruebas diseñados.

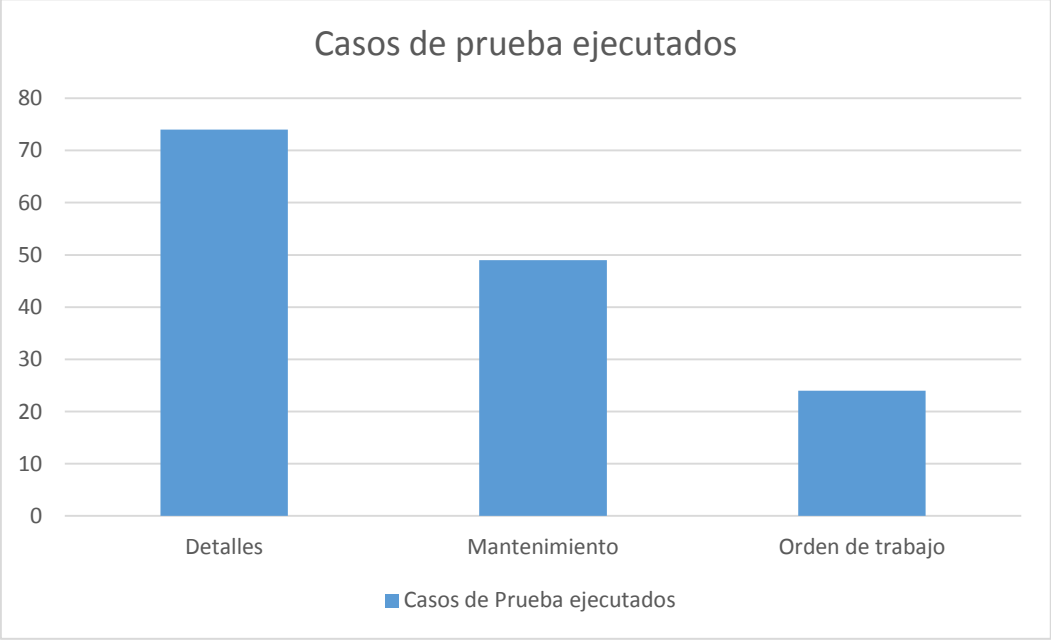
Tabla 13. Casos de prueba diseñados y su porcentaje.

Módulo/Submódulo	Casos de prueba diseñados	Porcentaje
Movimientos pendientes	55	34.6%
Movimientos iniciados	55	34.6%
Movimientos terminados	49	30.8%
Total	159	100%

3.2.2 RESULTADOS DE EJECUCIÓN DE CASOS DE PRUEBA

Como se había mencionado en el apartado del procedimiento, fueron ejecutados los casos de prueba, por lo cual la temática fue ejecutar diferentes casos que no pertenecieran a los diseñados, plasmados en los subtemas anteriores.

En los casos diseñados que fueron ejecutados, se tuvieron diferentes cantidades de casos como se visualiza en la gráfica 4 y la tabla 14.



Gráfica 4. Casos de pruebas ejecutados.

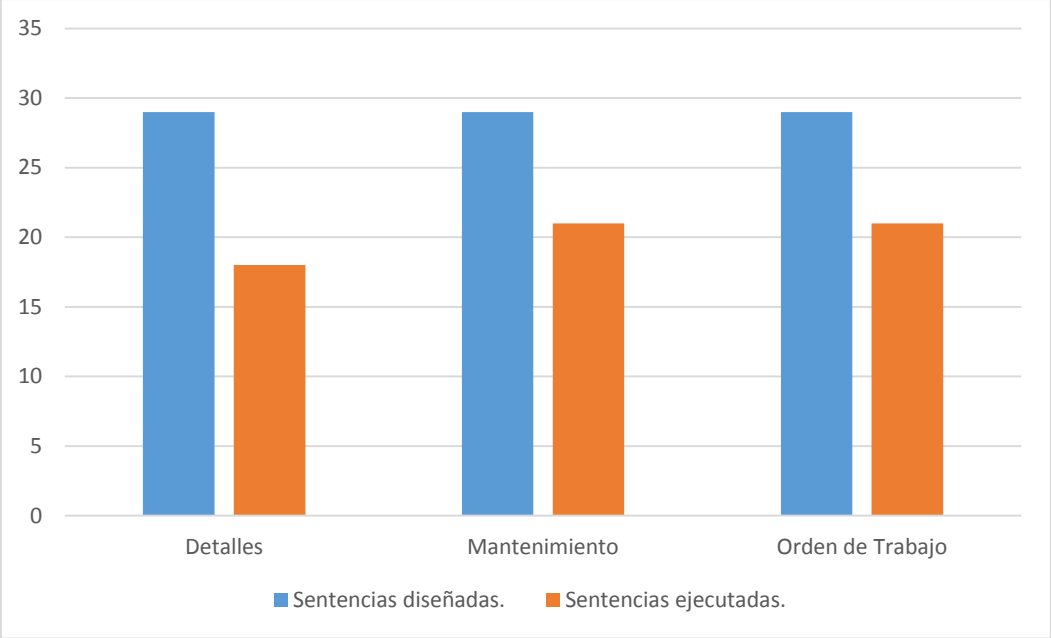
Tabla 14. Casos de prueba ejecutados y su porcentaje.

Módulo/Submódulo	Casos de prueba diseñados	Porcentaje
Detalles	74	50.4 %
Mantenimiento	49	33.3%
Orden de trabajo	24	16.3%
Total	147	100%

En el proceso de ejecución se encontraron un total de 29 anomalías, las cuales son como resultado de un mal desarrollo del proyecto “Sistema de Mantenimiento”, las anomalías son clasificadas por su gravedad.

3.2.2.1 LISTAS DE SENTENCIAS EJECUTADAS

Las listas de sentencias se ejecutaron al finalizar la ejecución de casos de prueba, con el fin de validar y verificar que sentencias aplicaron para cada submódulo (ver gráfica 5 y tabla 15).



Gráfica 5. Sentencias diseñadas vs. sentencias ejecutadas.

Tabla 15. Sentencias posibles vs. sentencias ejecutadas.

Submódulos	Lista de sentencias posibles	Sentencias ejecutadas
Detalles	29	18
Mantenimiento	29	21
Orden de trabajo	29	21

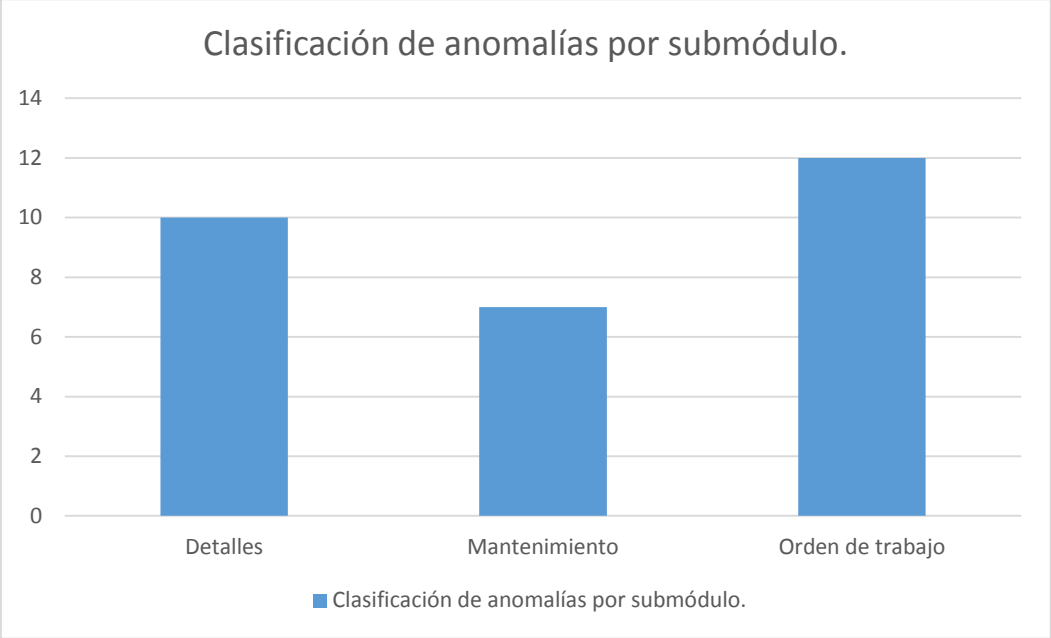
En los casos de prueba que fueron diseñados para ser ejecutados en base a los submódulos, se encontraron un total de 29 anomalías de los 3 submódulos con los que cuenta el módulo “Catálogos”. Las clasificaciones de las anomalías son las siguientes:

- Anomalías por módulo.
- Anomalías por severidad.
- Anomalías por prioridad.
- Anomalías por tipo.
- Anomalías por estado.
- Anomalías por navegador.

3.2.3 ANOMALÍAS DEL PROYECTO PILOTO

3.2.3.1 ANOMALÍAS POR MÓDULO

Esta clasificación de anomalías describe cuantas anomalías fueron obtenidas por cada submódulo del proyecto, siguiendo el proceso de pruebas (ver gráfica 6 y tabla 16).



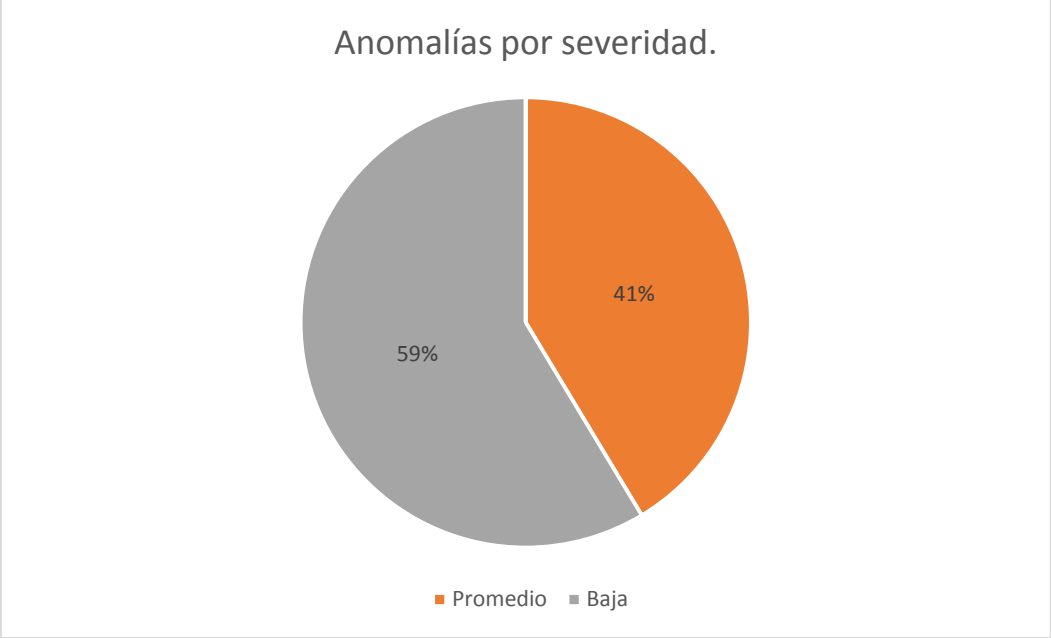
Gráfica 6. Anomalías por submódulo.

Tabla 16. Anomalías por módulo.

Submódulo	Cantidad de anomalías por submódulo	Porcentaje
Detalles	10	34.5%
Mantenimiento	7	24.1%
Orden de trabajo	12	41.4%
Total de anomalías	29	100%

3.2.3.2 ANOMALÍAS POR SEVERIDAD

Estas anomalías fueron encontraras de acuerdo por su severidad en el módulo “Catálogos” a ejecutar, verificar que tan importante es la gravedad de cada una de las anomalías de todos los submódulos (ver gráfica 7 y tabla 17).



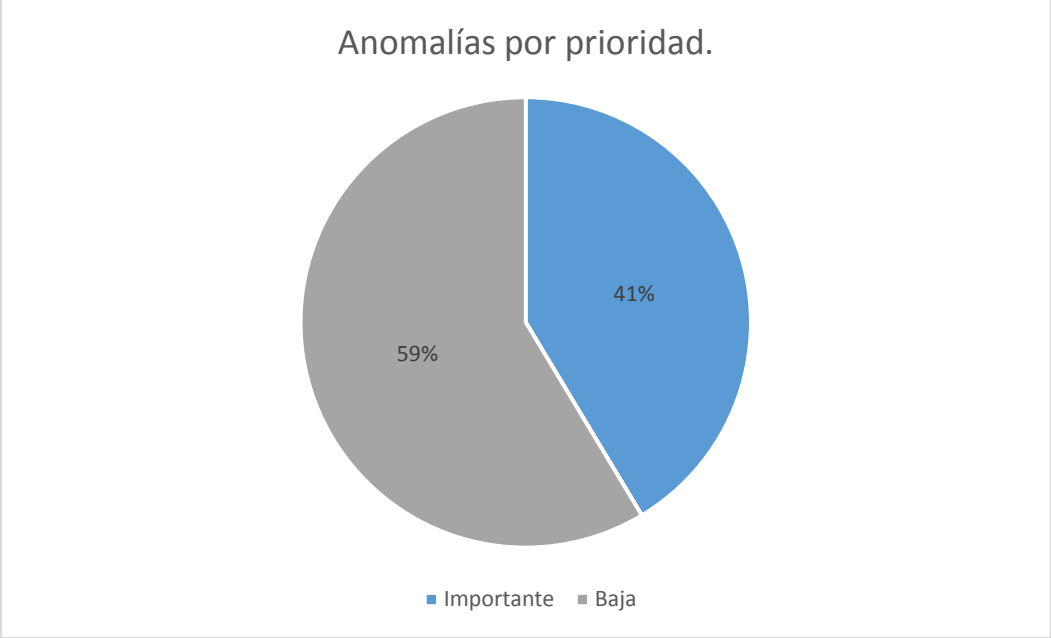
Gráfica 7. Anomalías por severidad.

Tabla 17. Anomalías por severidad.

Nivel de severidad	Cantidad de anomalías	Porcentaje
Promedio	12	41%
Baja	17	59%
Total de anomalías	29	100%

3.2.3.3 ANOMALÍAS POR PRIORIDAD

En este apartado las anomalías al igual que las de severidad, se clasifican por su prioridad es decir que tan importante es necesario resolver la anomalía (ver gráfica 8 y tabla 18).



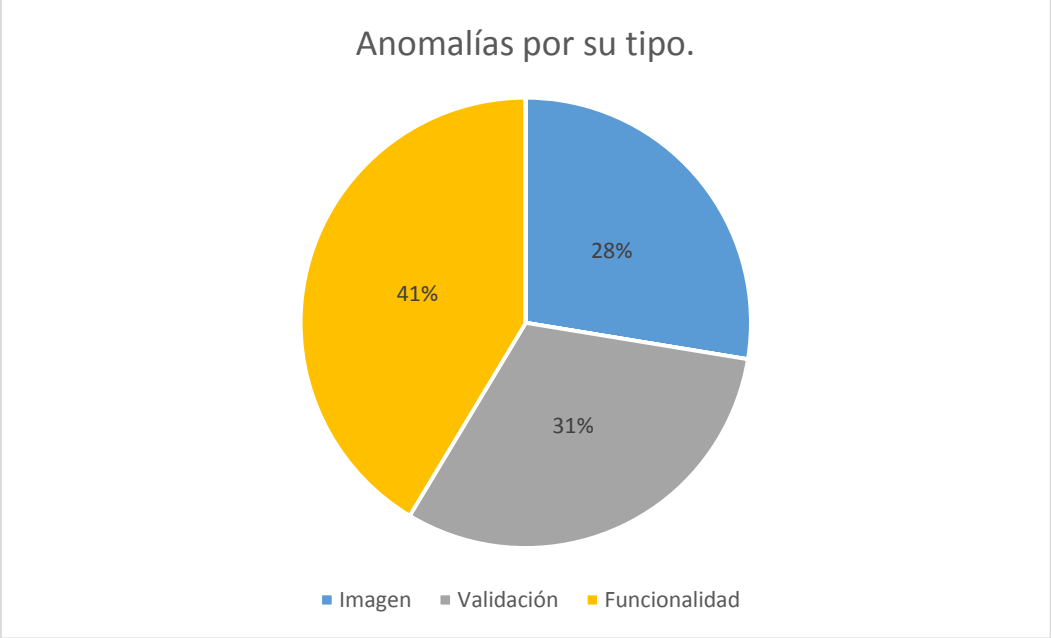
Gráfica 8. Anomalías por prioridad.

Tabla 18. Anomalías por prioridad.

Nivel de prioridad	Cantidad de anomalías	Porcentaje
Importante	12	41%
Baja	17	59%
Total de anomalías	29	100%

3.2.3.4 ANOMALÍAS POR SU TIPO

En este apartado son las anomalías que como tal se clasifican por su tipo, ayudan a visualizar a que parte pertenecen del sistema probado (ver gráfica 9 y tabla 19).



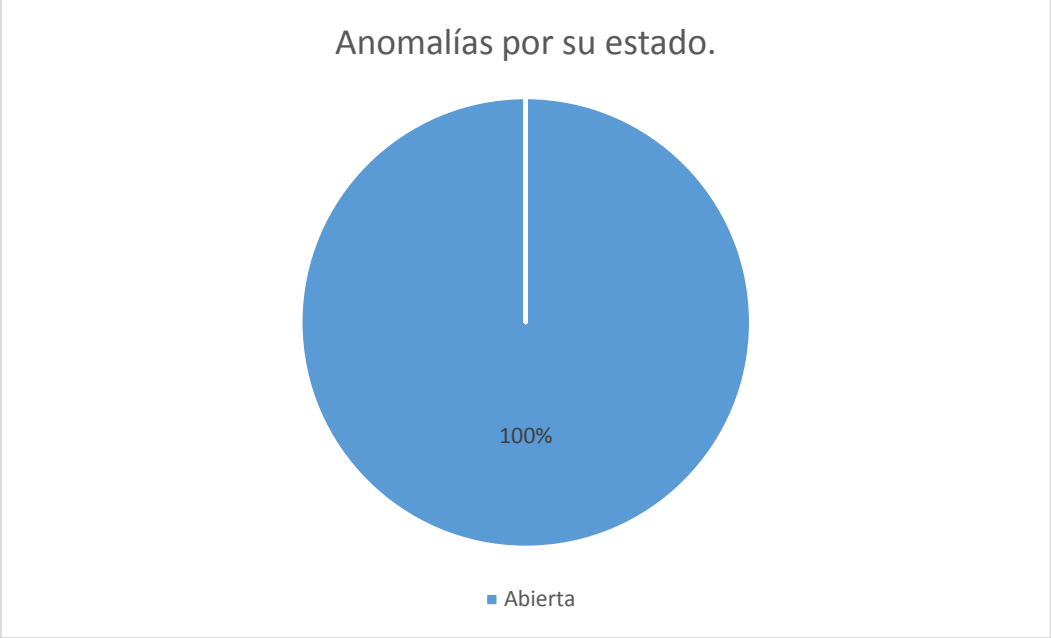
Gráfica 9. Anomalías por su tipo.

Tabla 19. Anomalías por su tipo.

Nivel de su tipo	Cantidad de anomalías	Porcentaje
Imagen	8	28%
Validación	9	31%
Funcionalidad	12	41%
Total de anomalías	29	100%

3.2.3.5 ANOMALÍAS POR ESTADO

Estas anomalías son de acuerdo a su estado en las que se encuentran en el momento de reportarlas, es decir, si se han trabajado en ellas o no (ver grafica 10 y tabla 20).



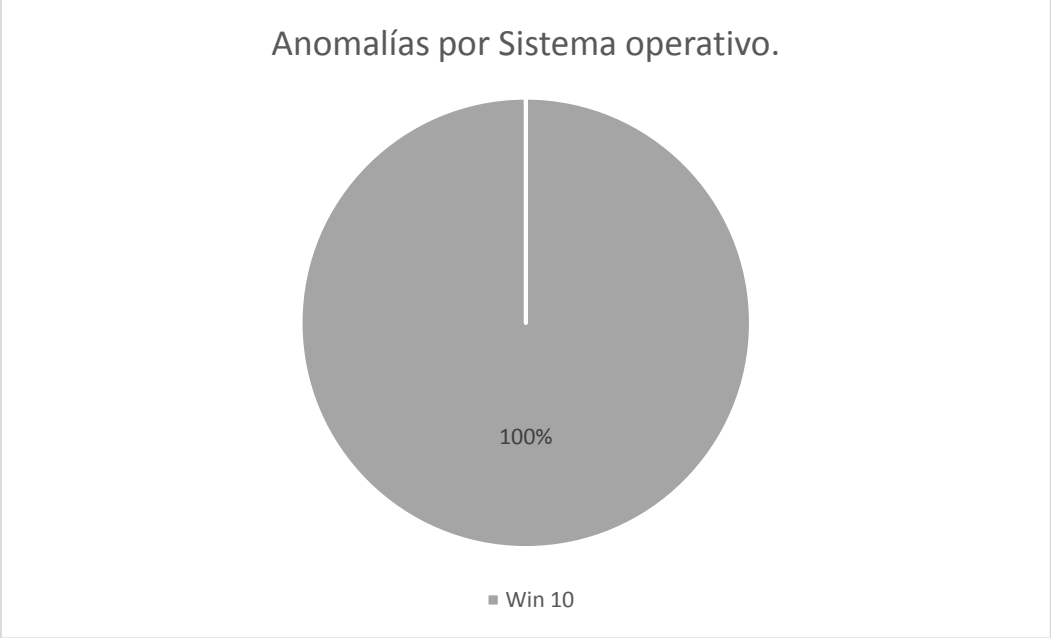
Gráfica 10. Anomalías por su estado.

Tabla 20. Anomalías por su estado.

Nivel de Estado	Cantidad de anomalías	Porcentaje
Abierta	29	100%
Total de anomalías	29	100%

3.2.3.6 ANOMALÍAS POR SISTEMA OPERATIVO

Estas anomalías son de acuerdo en que sistema operativo que se utilizó y por lo cual fueron detectadas al ejecutar los casos de prueba (ver gráfica 11 y tabla 21).



Gráfica 11. Anomalías por Sistema operativo.

Tabla 21. Anomalías por Sistema operativo.

Sistema Operativo	Cantidad de anomalías	Porcentaje
Win 10	29	100%
Total de anomalías	29	100%

3.2.3.7 ANOMALÍAS REPORTADAS POR IMPORTANCIA

En la siguiente tabla se mostrarán los submódulos los cuales fueron probados, así mismo observar la importancia de cada uno catalogado por el cliente, estos para ser ejecutados, también mostrando el total de anomalías encontradas por cada submódulo (ver tabla 22).

Tabla 22. Importancia y número de anomalías por submódulo.

Módulo/Submódulo a probar	Importancia 3- alta 2- promedio 1-baja	Cantidad de anomalías
2. Catálogos		
2.1 Equipos & Partes		
2.1.1 Detalles	2	10
2.1.2 Mantenimientos	3	7
2.1.3 Orden de trabajo	3	12

3.2.4 RESULTADO DE ANOMALÍAS POR FUNCIONALIDAD

Se observó que de las 29 anomalías 12 son de funcionalidad, las cuales fueron clasificadas como importantes, debido a que su principal función no se realizaba correctamente, en la tabla 23 se muestra una relación de las anomalías importantes con su escenario correspondiente.

Tabla 23. Anomalías por escenarios.

No. escenario.	Anomalías
1	3
3	2
6	5
8	2

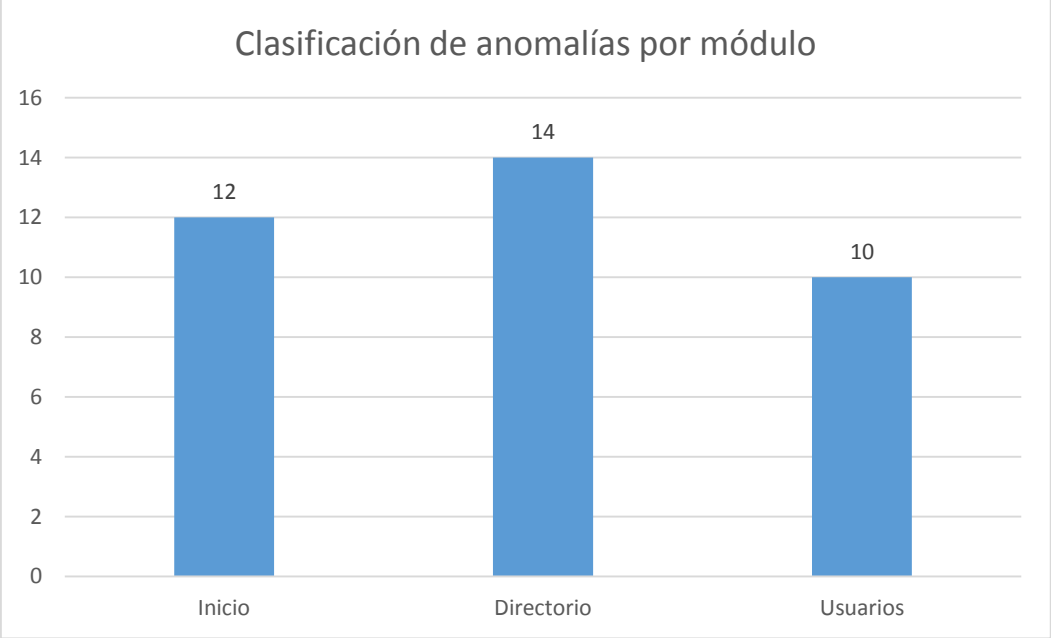
3.2.5 RESULTADO DE PRUEBAS EXPLORATORIAS

Las pruebas exploratorias que se realizaron a cada uno de los módulos de la página web de publicidad se encontraron un total de 36 anomalías de las cuales se caracterizan por dividirse en varios tipos de anomalías las cuales identifican un nivel de gravedad dependiendo de qué clasificación se utiliza:

- Anomalías por módulo.
- Anomalías por severidad.
- Anomalías por prioridad.
- Anomalías por tipo.
- Anomalías por estado.
- Anomalías por navegador.

3.2.5.1 ANOMALÍAS POR MÓDULO

En esta parte se describirá el total de anomalías que se encontraron por módulos en la página web de publicidad (ver gráfica 12 y tabla 24).



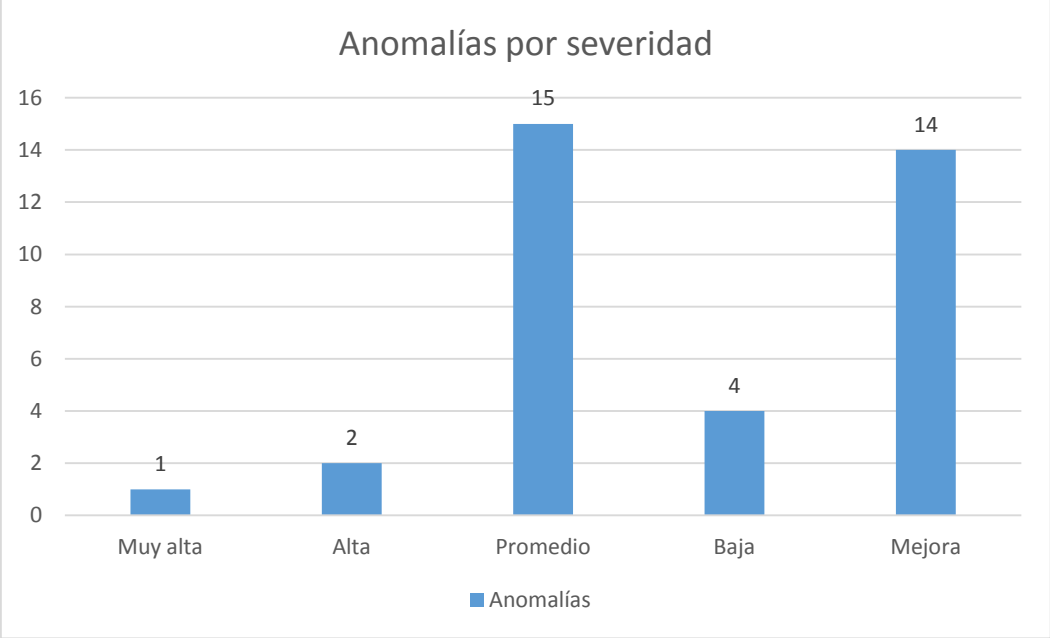
Gráfica 12. Anomalías por módulo en página web de publicidad.

Tabla 24. Anomalías por módulo en página web de publicidad.

Módulos	Cantidad de anomalías	Porcentajes
Inicio	12	33%
Directorio	14	39%
Usuarios	10	28%
Total de anomalías	36	100%

3.2.5.2 ANOMALÍAS POR SEVERIDAD

Se mostrarán en la gráfica 13 y tabla 25, el total de anomalías que se encontraron en la página web de publicidad de acuerdo al nivel clasificación de severidad.



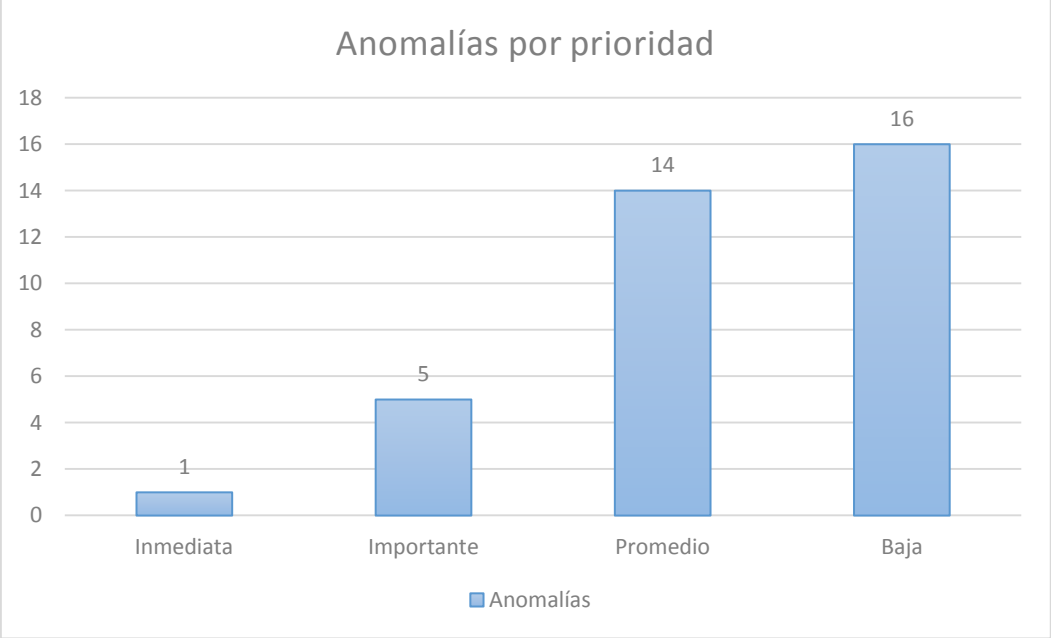
Gráfica 13. Anomalías por severidad en página web de publicidad.

Tabla 25. Anomalías por severidad en página web de publicidad.

Nivel de severidad	Cantidad de anomalías
Muy alta	1
Alta	2
Promedio	15
Baja	4
Mejora	14
Total de anomalías	36

3.2.5.3 ANOMALÍAS POR PRIORIDAD

A continuación, se mostrarán el total de las anomalías encontradas en la página web de publicidad clasificadas por prioridad (ver gráfica 14 y tabla 26).



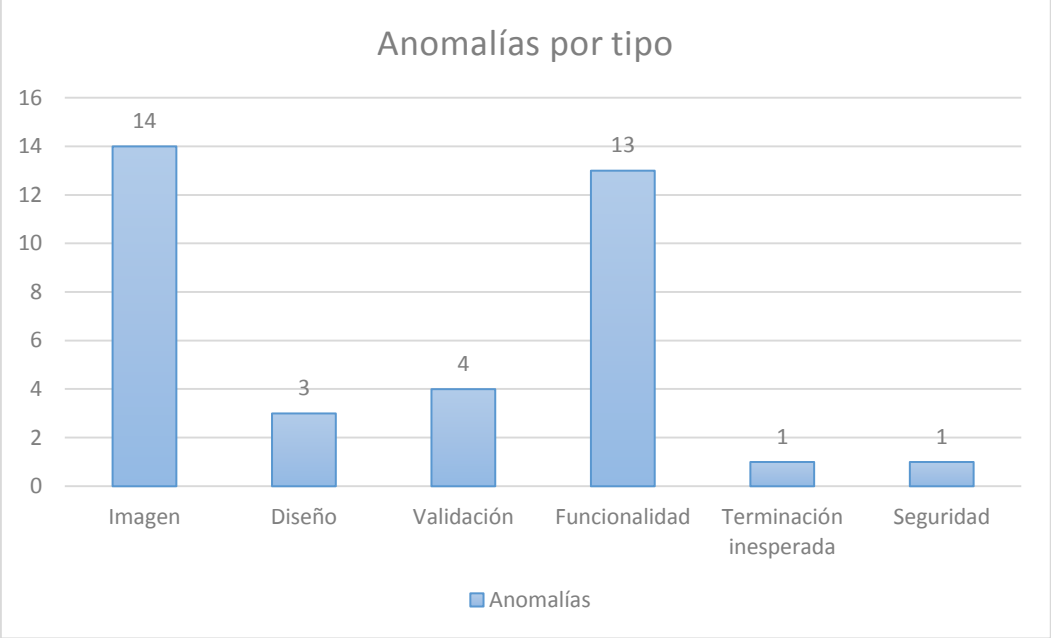
Gráfica 14. Anomalías por prioridad en página web de publicidad.

Tabla 26. Anomalías por prioridad en página web de publicidad.

Nivel de prioridad	Cantidad de anomalías
Inmediata	1
Importante	5
Promedio	14
Baja	16
Total de anomalías	36

3.2.5.4 ANOMALÍAS POR SU TIPO

En este apartado se mostrarán el total de las anomalías encontradas en la página web de publicidad clasificadas por su tipo. (ver gráfica 15 y tabla 27).



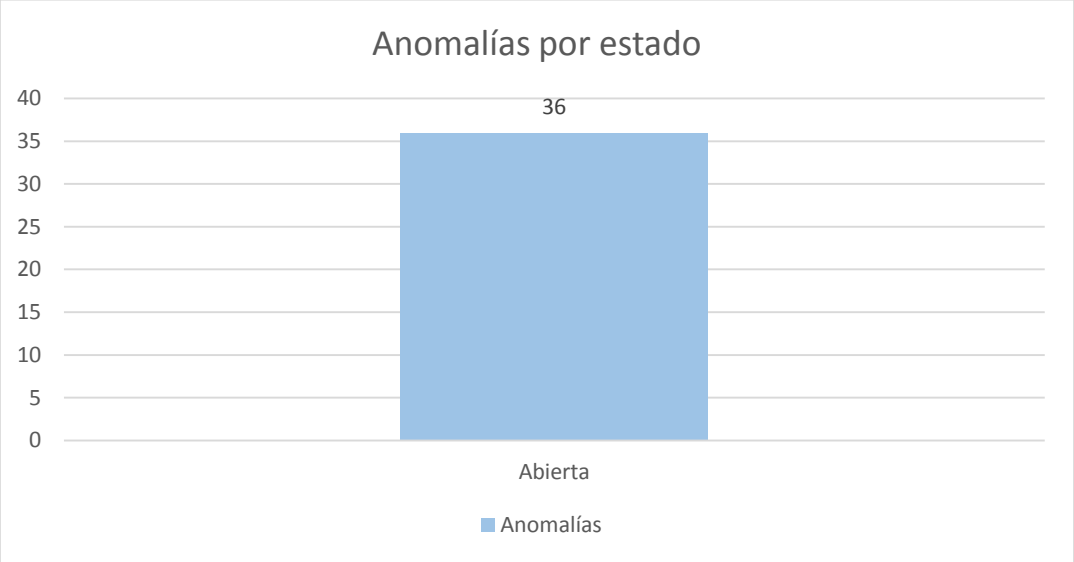
Gráfica 15. Anomalías por tipo en página web de publicidad.

Tabla 27. Anomalías por tipo en página web de publicidad.

Tipo de anomalía	Cantidad de anomalías
Imagen	14
Diseño	3
Validación	4
Funcionalidad	13
Terminación inesperada	1
Seguridad	1
Total de anomalías	36

3.2.5.5 ANOMALÍAS POR ESTADO

Se mostrarán el total de las anomalías encontradas en la página web de publicidad clasificadas por estado (ver gráfica 16 y tabla 28).



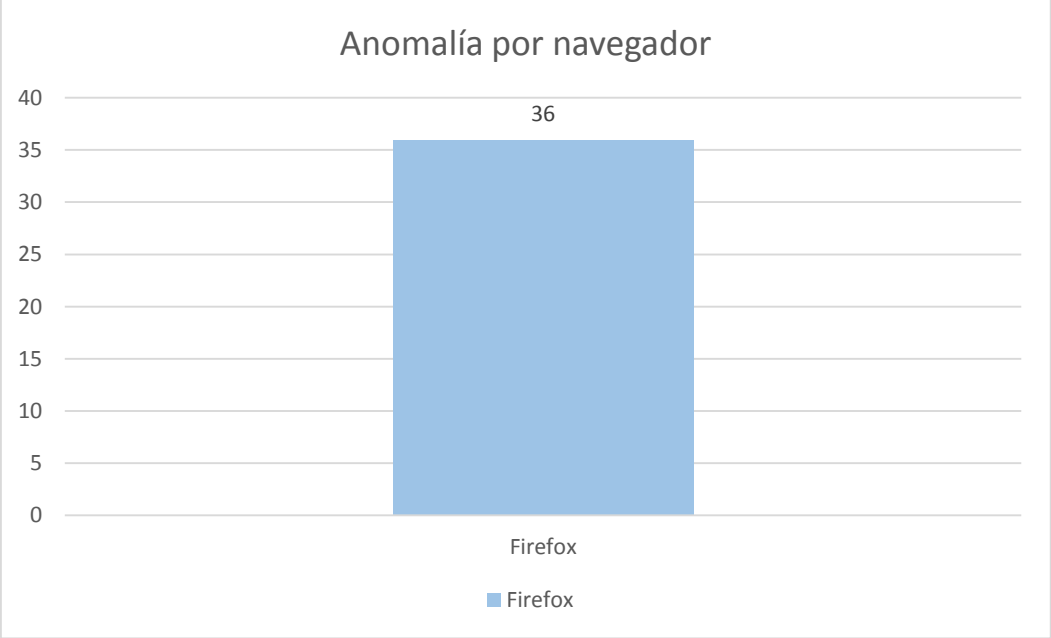
Gráfica 16. Anomalías por estado en página web de publicidad.

Tabla 28. Anomalías por estado en página web de publicidad.

Anomalía por estado	Cantidad de anomalías
Abierta	36
Total de anomalías	36

3.2.5.6 ANOMALÍAS POR NAVEGADOR

A continuación, se mostrarán el total de las anomalías encontradas en la página web de publicidad clasificadas por navegador. (ver gráfica 17 y tabla 29)



Gráfica 17. Anomalías por navegador en página web de publicidad.

Tabla 29. Anomalías por navegador en página web de publicidad.

Tipo de navegador	Anomalías
Firefox	36
Total de anomalías	36

3.2.6 SOPORTE A HERRAMIENTAS DE TESTLINK Y MANTIS

Se dio mantenimiento a las herramientas de Testlink y de Mantis, las cuales son herramientas de software libre, es decir, el código en el cual están desarrolladas es totalmente libre, lo cual se permite modificar el código como el usuario lo desee.

Se llevó una capacitación breve sobre cómo se utilizan estas dos herramientas, impartida por un empleado de la empresa, estas herramientas ya están instaladas en el servidor y modificadas a las necesidades de la empresa, con el fin de si existiera algún error en el funcionamiento o de modificar alguna parte de las herramientas, se pudiera realizar sin ningún problema.

Al término de la capacitación, se necesitó configurar los parámetros de conexión entre el servidor de la empresa y dispositivos, con el fin de que un correo sea enviado si se solicitará en la herramienta de Testlink.

La modificación de código era para cuando fuese necesario o se deseará restablecer la contraseña por parte de algún usuario, para hacer este proceso de restablecer contraseña es introducir el nombre del usuario y posteriormente dar clic en el botón de “Enviar” pero como resultado arrojaba un mensaje señalando el error “SMTP Error: Could not connect to SMTP host.” y era primordial corregir (ver figura 17).

Figura 17. Error de restablecimiento de contraseña en Testlink.

Para corregir este error, se investigó en Internet en que parte del código era necesario la modificación, con el objetivo de enviar la petición para restablecer la contraseña, mencionar que el servidor de la empresa recibe la petición y contesta enviando un correo para el usuario, considerando que el código que maneja testlink ya está estructurado y listo para instalarse en cualquier servidor, sólo es configurarlo a las necesidades de la empresa, en este caso sólo se requirió solucionar el detalle de correo.

Para mandar un correo fue necesario investigar que protocolo utiliza testlink, como resultado fue el protocolo SMTP, ya que este protocolo es básico para la transferencia simple de correo, el cual es utilizado para el intercambio de mensajes de correo electrónico entre dispositivos y servidores.

Se modificó una pequeña parte del código del archivo llamado “config.inc.php” (ver figura 18 y figura 19).

```
/* [SMTP] */

/**
 * @var string SMTP server name or IP address ("localhost" should work in the most cases)
 * Configure using custom_config.inc.php
 * @uses lib/functions/email_api.php
 */
$g_smtp_host        = 'SMTP.e-quallity.net';  # SMTP server MUST BE configured

# Configure using custom_config.inc.php
$g_tl_admin_email      = 'SoporteTestlink@e-quallity.net'; # for problem/error notification
$g_from_email          = 'SoporteTestlink@e-quallity.net';  # email sender
$g_return_path_email   = 'SoporteTestlink@e-quallity.net';

/**
 * Email notification priority (low by default)
 * Urgent = 1, Not Urgent = 5, Disable = 0
 **/
$g_mail_priority = 5;

/**
 * Taken from mantis for phpmailer config
 * select the method to mail by:
 * PHPMAILER_METHOD_MAIL - mail()
 * PHPMAILER_METHOD_SENDMAIL - sendmail
 * PHPMAILER_METHOD_SMTP - SMTP
 */
$g_phpMailer_method = PHPMAILER_METHOD_SMTP;

/** Configure only if SMTP server requires authentication */
$g_smtp_username    = 'SoporteTestlink';  # user
$g_smtp_password    = 'Mjs;,+127';  # password
```

Figura 18. Parte 1 de modificación de código para envío de correo.

```
/**
 * This control the connection mode to SMTP server.
 * Can be '', 'ssl','tls'
 * @global string $g_smtp_connection_mode
 */
$g_smtp_connection_mode = 'ssl';

/**
 * The smtp port to use.  The typical SMTP ports are 25 and 587.  The port to use
 * will depend on the SMTP server configuration and hence others may be used.
 * @global int $g_smtp_port
 */
$g_smtp_port = 25;

// ----------------------------------------------------------------------------
/* [User Authentication] */

/**
 * Login authentication method:
 *  'MD5' => use password stored on db
 *  'LDAP' => use password from LDAP Server
 */
$tlCfg->authentication['method'] = 'MD5';
```

Figura 19. Parte 2 de modificación de código para envío de correo.

Esta modificación fue la única que se realizó, ya que no resultaron más alteraciones de la herramienta en el momento. Se corrigió el problema de envío de correos para restablecer la contraseña de algún usuario que estuviera registrado en Testlink (ver figura 20 y figura 21).

Figura 20. Introducción de usuario para restablecer contraseña.

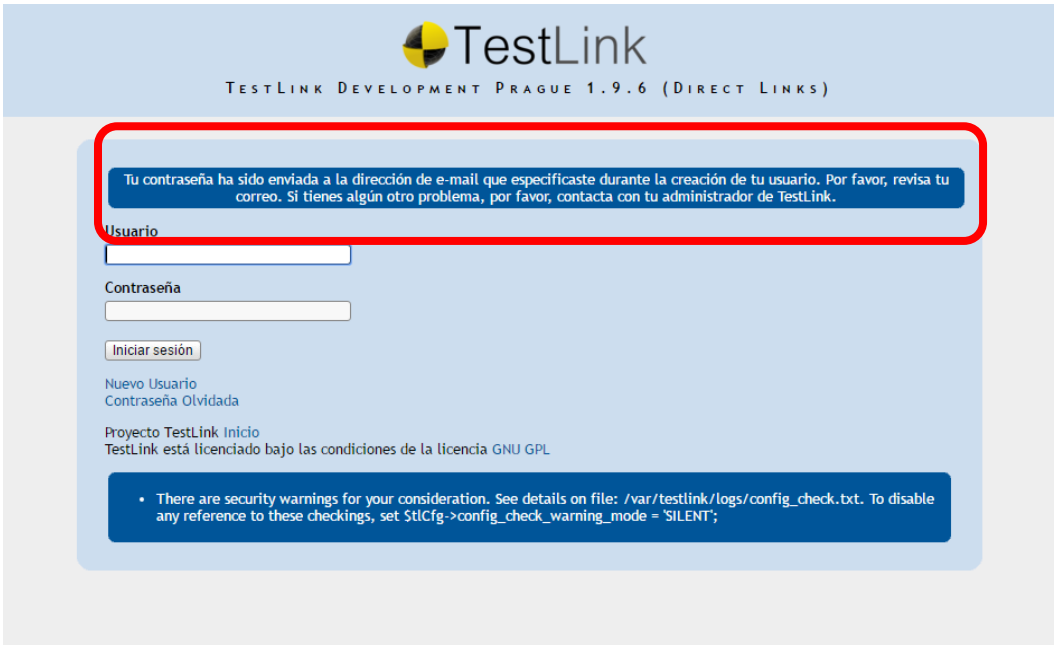


Figura 21. Envío de correo para restablecer contraseña.

3.2.7 RESULTADO DE CÉLULA DE INVESTIGACIÓN

En la célula de investigación se tuvieron varios resultados que hacen énfasis en varios puntos, mencionar que el mercado del videojuego es bastante amplio. Se

generó una tabla donde se observa cuáles son los géneros principales que existen en el mercado, con la finalidad de conocer el objetivo de cada uno de los géneros, consolas donde son compatibles y las pruebas principales que aplican al género de videojuego (ver tabla 30).

Tabla 30. Géneros de videojuegos.

Géneros de videojuegos	**Objetivo de videojuego**	**Consolas compatibles**	**Pruebas principales al género**
Espionaje táctico, infiltración, estrategia.	Género donde el personaje sea inadvertido y que no sea descubierto por los enemigos, buscar cual es la mejor estrategia para su alcanzar su objetivo.	-Xbox -Play Station -GameCube	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas de localización -Pruebas beta -Pruebas de carga -Pruebas multijugador
Cartas, pinball, puzzle, tablero.	Género donde se tiene en común la función de representan los juegos de mesa, estos tienen como objetivo hacer	-Xbox -Play Station -GameCube -Computadora -Súper Nintendo	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas beta

	puntos o realizar jugadas.	-Nintendo Wii -Sega Mega Drive -Nintendo 64	-Pruebas multijugador
Acción y aventura.	Tiene como objetivo, hacer que el personaje principal del videojuego realicé diferentes acciones controlado por el usuario, como también recorrer largos niveles y esquivar obstáculos para que el personaje cumpla con los puntos necesarios, este es muy parecido al género de plataformas la única diferencia del género acción y aventura es eliminar enemigos.	-Xbox -Play Station -GameCube -Computadora -Súper Nintendo -Nintendo Wii -Sega Mega Drive -Nintendo 64	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas de localización -Pruebas beta

Lucha (uno contra uno, lucha libre).	Género de videojuego que tiene como objetivo que el personaje luche contra uno o muchos personajes, teniendo como objetivo eliminar al contrincante.	-Xbox -Play Station -GameCube -Computadora -Súper Nintendo -Nintendo Wii	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas de localización -Pruebas beta -Pruebas de carga -Pruebas multijugador
Plataformas.	Este género busca que el personaje avance al siguiente nivel evadiendo obstáculos que pueden intervenir en el transcurso del personaje sobre la plataforma.	-Computadora -Súper Nintendo -Nintendo Wii -Sega Mega Drive -Nintendo 64	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas beta
Conducción.	Su principal función de este género es conducir un vehículo.	-Xbox -Play Station -GameCube -Súper Nintendo	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas de

			localización -Pruebas beta -Pruebas de carga -Pruebas multijugador
Deportivos.	El género consta de que el equipo de un usuario se enfrente a otros contrincantes ya sea dirigido por la computadora o por otros usuarios.	-Xbox -Play Station -GameCube -Computadora -Súper Nintendo -Nintendo Wii	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas de localización -Pruebas beta -Pruebas de carga -Pruebas multijugador
Construcción	El objetivo es construir una estructura, ya sea una ciudad, un hospital o un sistema de transportes, mantenerlo y gestionarlo, de forma que produzca beneficios para poder mejorar la	-Xbox -Play Station -GameCube -Computadora -Súper Nintendo -Nintendo Wii -Sega Mega Drive	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas de localización -Pruebas beta -Pruebas de carga

	estructura o construir nuevas.	-Nintendo 64	
Educativos	Género de carácter didáctico o pedagógico orientados principalmente a los jóvenes (niños en edad pre-escolar y posterior).	-Xbox -Play Station -GameCube -Computadora -Súper Nintendo -Nintendo Wii -Sega Mega Drive -Nintendo 64	-Prueba de funcionalidad -Pruebas de compatibilidad -Pruebas beta -Pruebas de carga

Para realizar cada una de estas pruebas existe un método, el cual se utiliza bastante y da mejores resultados es llamado “Reproducción y ejecución de trazas”, quiere decir que se va grabando por partes el transcurso del videojuego y va verificando el diseño y jugabilidad paso por paso y muy detalladamente, con el fin de encontrar los errores que posiblemente pudiera tener el videojuego.

3.3 COMPETENCIAS ALCANZADAS

En la residencia profesional se implementó el conocimiento adquirido de varias materias cursadas en el Instituto Tecnológico Superior Zacatecas Sur, tales como fundamentos de ingeniería de software, ingeniería de software, validación y verificación de software y gestión de proyectos de software. Se trataron temas importantes; se comprendió la importancia de la aplicación de estándares de calidad de software, saber gestionar un proyecto de software, se realizó prueba para diagnosticar calidad de software, se entendió lo significativo del uso de las herramientas y técnicas para gestionar un proyecto, como también el uso de modelos de ciclos de vida del desarrollo de software y detección de fallas.

Algunas de las competencias implicadas en la residencia fueron:

- Capacidad de organizar y planificar proyecto de software.
- Implementación de técnicas para la validación y verificación de software.
- Capacidad de analizar comportamiento y ambiente organizacional.
- Identificar y comprender tipos y características de requerimientos.
- Implementar técnicas de adquisición de datos (entrevistas, cuestionarios, lluvia de ideas, entre otros) para el proceso de un proyecto de software.
- Capacidad para implementar técnicas y herramientas en el ámbito de pruebas de software.
- Capacidad para comunicarse de manera oral y escrita dentro y fuera de la empresa.
- Capacidad para la solución de problemas, toma de decisiones.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

La calidad es muy importante en un software, ya que nos permite observar si dicho software cuenta con la calidad suficiente para colocarlo en el mercado y que cumpla con todos los estándares necesarias para que posteriormente tenga un buen resultado.

Dicha calidad representa un punto importante para cualquier sistema que trate de software, ya que el desarrollador busca crear productos con calidad y estos a su vez se introduzcan de forma rápida al mercado y por lo cual generen ingresos. Pero muchas de las veces el desarrollador por precipitarse hace entrega del software y éste no cumplió por un proceso que permita diagnosticar el nivel de calidad que posee el software y como resultado hace que el software falle y no genere los ingresos que se tenían planeados por motivos de que no funcionó correctamente o se encontraron anomalías en el software al trabajarlo.

La metodología empleada por la empresa es muy eficaz ya que el proceso que utiliza está muy bien empleado en proceso de pruebas. La metodología utilizada por e-Quallity S.A. de C.V. esta enfoca a probar software, mediante sus diferentes etapas con las que cuenta, con ayuda de diferentes técnicas de pruebas, herramientas de apoyo y documentos creados para tener un mejor control de información, teniendo un proceso de pruebas muy rígido, eficiente y eficaz, como resultado permite diagnosticar la calidad de un software tan eficaz de tal forma, que el software se mejore y se corrija y éste cumpla con una mayor calidad.

El proceso de pruebas que se le aplica a un software y verificar con cuanta calidad posee, ayudara al desarrollador a mejorar el software desde su creación y así desarrollar software que cumpla con la calidad necesaria.

Al trabajar en una empresa dedicada a las pruebas de software se abrió un panorama, añadiendo más al conocimiento sobre el desarrollo total de un software, por lo cual tener la oportunidad de aprender algo nuevo y ponerlo en práctica.

Al aplicar el conocimiento adquirido sobre los procesos de prueba, como desarrollador, extiende más el saber, de observar cómo se realizan las pruebas a un software, que al desarrollar tienes el conocimiento de que es lo que se va a probar y poder evitar que puedan salir anomalías posibles en tu software desarrollado.

Se inculca bastante la investigación sobre relación al tema de la célula de investigación, haciendo que se busque nuevas oportunidades en donde se pudieran aplicar las pruebas, haciendo esto que se buscara y se genere una metodología de pruebas especializadas a un tema.

El periodo de la residencia es el comienzo del camino la carrera en el ámbito laboral, donde se aplicará lo adquirido de la escuela, se aprenderá más y se preparará más para seguir creciendo como persona y como profesional.

4.2 RECOMENDACIONES

En el transcurso de la residencia profesional se observó diferentes factores en los cuales se puede hacer énfasis en varias situaciones que se presentaron.

El estudiante, el encargado de vinculación de la institución y el representante de la empresa, es primordial contar con una mayor comunicación, en base a que, en ocasiones la información que se otorga por parte de la institución hacia al residente y éste a su vez comunicarlo a la empresa puede ser mal entendida y por consecuencia se genera confusión y como tal se demora la entrega de documentos o información importante.

En la mayor parte se tiene que contar con un registro de actividades las cuales se tiene que estar reportando de manera constante de preferencia semanal al asesor de la institución educativa. En la estancia de la residencia, si la empresa no permite mencionar información importante en cuanto a actividades realizadas, por parte del asesor considerar la situación que maneja la empresa sobre la información, por lo cual la empresa debe realizar un comunicado sobre la situación y por medio del residente hacerla llegar al asesor para que esté considere las políticas de la empresa.

Se recomienda que el residente aplique el mayor esfuerzo posible y ofrezca todo el conocimiento adquirido de la institución educativa, para que la empresa donde se realizó la residencia lo tome en cuenta y en un futuro seguir trabajando ya no como residente si no como empleado.

REFERENCIAS BIBLIOGRÁFICAS Y VIRTUALES

Multidiciplinaria, Proyectos e Ingenieria. (2016). *mpi : Multidiciplinaria, Proyectos e Ingenieria*. Obtenido de http://www.mpi.cl/sd/es/00006/SS00075.html

B. P. (s.f.). *Centro de Ensayos de Software.* Obtenido de http://www.ces.com.uy/documentos/imasd/CES-Testing_Exploratorio.pdf

Barrientos, P. A. (25 de Abril de 2014). *SEDICI: Repositorio Institucional de la UNLP*. Obtenido de http://sedici.unlp.edu.ar/handle/10915/34969

Ecu Red. (2016). *Ecu Red: Conocimiento con todos y para todos*. Obtenido de https://www.ecured.cu/Pruebas_de_caja_blanca

Ecu Red. (2016). *Ecu Red: Conocimiento con todos y para todos.* Obtenido de https://www.ecured.cu/Pruebas_de_caja_negra

El otro lado. (9 de Febrero de 2014). *El otro lado .* Obtenido de http://www.elotrolado.net/wiki/G%C3%A9neros_de_los_videojuegos

e-Quallity. (12 de Enero de 2008). *Corporación e-Quallity S.A. de C.V*. Obtenido de www.e-quallity.net

Globe Testing. (s.f.). *Globe testing* . Obtenido de http://www.globetesting.com/glosario/anomalia/

J. P. (19 de Junii de 2007). *SoftQaNetwork: Your Quallity assurance network*. Obtenido de http://www.softqanetwork.com/tpi-testing-process-improvement

Java & Arch, Blog. (11 de Septiembre de 2012). *Java & Arch, Blog*. Obtenido de http://javablog.eliumontoya.com/home/tipodepruebasparadesarrollodesoftware

Juega Libre. (27 de Julio de 2015). *Juega libre.* Obtenido de http://juegalibre.virtual.uniandes.edu.co/index.php/2015/07/27/testing-metodologia/

Kaner, C. (1999). *Testing Computer Software, 2nd Edition.* Wiley.

Londoño, J. A. (6 de Abril de 2005). *Blogspot: Ingenieria de Software*. Obtenido de http://ing-sw.blogspot.mx/2005/04/tipos-de-pruebas-de-software.html

Patiño, G. G. (4 de Noviembre de 2015). *Prezi.* Obtenido de https://prezi.com/nqb1v18xrqy8/pruebas-de-caja-blanca/

Rosa, D. R. (15 de Abril de 2013). *Escuela de Organización Industrial: Master Executive en Administración y Direccion de empresas.* Obtenido de http://www.eoi.es/blogs/madeon/2013/04/15/microsoft-project-en-la-gestion-de-proyectos/

SoftQaNetwork. (22 de Marzo de 2007). *SoftQaNetwork: Your quallity assurance network*. Obtenido de http://www.softqanetwork.com/descubre-el-tmm

Test Link Tool. (16 de Septiembre de 2011). *Test Link Tool.* Obtenido de http://testinktool.blogspot.mx/2011/09/how-to-configure-smtp-in-testlink.html

Valero, L. C. (Junio de 2015). *Sprint.* Obtenido de http://eprints.ucm.es/32860/1/Luis%20Costero%20-%20Ejecuci%C3%B3n%20y%20adaptaci%C3%B3n%20de%20trazas%20para%20la%20automatizaci%C3%B3n%20de%20pruebas.pdf

Wikipedia. (27 de Abril de 2016). *Wikipedia: La Enciclopedia libre.* Obtenido de https://es.wikipedia.org/wiki/Mantis_Bug_Tracker

Wikipedia: La enciclopedia libre. (2016). *Wikipedia: La enciclopedia libre*. Obtenido de https://es.wikipedia.org/wiki/Caso_de_prueba

GLOSARIO

- **Bug:** Un error de software, comúnmente conocido como bug (Bicho), es un error o fallo en un programa de computador o sistema de software que desencadena un resultado indeseado.
- **CMMi:** Es un conjunto de modelos basados en las mejores prácticas en la gestión de los procesos, desarrollados a través de un proyecto conjunto en el que participaron el SEI (Software Engineering Institute).
- **CppUnit:** Es un marco de programación de pruebas unitarias para el lenguaje de programación C++.
- **CPU (Unidad Central de Procesamiento):** Parte importante de la computadora.
- **Framework:** Es un entorno o ambiente de trabajo para desarrollo; dependiendo del lenguaje normalmente integra componentes que facilitan el desarrollo de aplicaciones como el soporte de programa, bibliotecas, plantillas y más.
- **jUnit:** Es un framework java que permite la realización de la ejecución de clases de manera controlada, para poder comprobar que los métodos realizan su cometido de forma correcta.
- **Look & feel ("aspecto y tacto"):** Es una metáfora utilizada dentro del entorno de marketing para poder dar una imagen única a los productos, incluyendo áreas como el diseño exterior, la caja en que se entrega al cliente.
- **MySql:** Es un sistema de administración de bases de datos relacionales (Database Management System, DBMS).

- **NUnit:** Es un framework open source de pruebas de unidad para Microsoft .NET y Mono. Sirve al mismo propósito que JUnit trabaje en el mundo java.
- **OpenSource (Código abierto):** Es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones éticas y morales las cuales destacan en el llamado software libre.
- **PHP:** Es un lenguaje de código abierto especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.
- **RUP (Rational Unified Process o Proceso Unificado de Racional):** Es un proceso de ingeniería de software que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo.
- **Testers:** Es el que se encarga de realizar las pruebas respectivamente al proyecto de software.
- **TestNG:** Es un framework para pruebas y testing que trabaja con Java y está basado en JUnit (para java) y NUnit (para .NET), pero introduciendo nuevas funcionalidades que los hacen más poderosos y fáciles de usar.