EVALUACIÓN Y MANTENIMIENTO DEL SOFTWARE “SISTEMA DE INFORMACIÓN, EDUCACIÓN Y COMUNICACIÓN EN ENFERMEDADES INFECCIOSAS QUE CURSAN CON FIEBRES AGUDAS

CESAR ANDRES ALVARADO HERNANDEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICO-MECANICAS

ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA

BUCARAMANGA

2016

EVALUACIÓN Y MANTENIMIENTO DEL SOFTWARE SISTEMA DE INFORMACIÓN, EDUCACIÓN Y COMUNICACIÓN EN ENFERMEDADES INFECCIOSAS QUE CURSAN CON FIEBRES AGUDAS

CESAR ANDRES ALVARADO HERNANDEZ

Trabajo de grado para optar al título de Ingeniero de Sistemas

DIRECTOR
HUGO HERNANDO ANDRADE SOSA
MAGISTER EN INFORMÁTICA

CO-DIRECTOR
GERARDO MUÑOZ MANTILLA
PHD EPIDEMIOLOGY AND VECTOR BIOLOGY

CO-DIRECTOR
LUIS EDUARDO GUERRA GONZALEZ
MAGISTER EN INGENIERIA DE SISTEMAS E INFORMATICA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2016

DEDICATORIA

Dedicó este trabajo de grado:

A Dios Todopoderoso y a la Santísima Virgen por brindarme la sabiduría y la fortaleza que me permitió culminar satisfactoriamente esta etapa de mi vida

A mis padres y hermano razones de mi existencia y motores de mi vida por el apoyo constate e incondicional en los momentos difíciles durante el desarrollo de este proyecto. Que me dieron las fuerzas para conseguir esta meta, quienes me han brindado su amor, confianza y formación llena de valores que hacen de mí la persona que soy. A mi mamá, Mary luz Hernández por sus constantes oraciones, por el ánimo y el apoyo que hicieron posible culminar con este trabajo de grado. A mi padre Luis Emiro Alvarado por la confianza depositada en mí y por la ayuda que me permite hoy convertirme en una profesional.

A mis amigos con quienes compartí la carrera universitaria y estuvieron acompañándome en las buenas y en las malas. Siempre los recordare.

AGRADECIMIENTOS

Son muchas las personas especiales a quienes nos gustaría agradecer su amistad, apoyo, ánimo y compañía en las diferentes etapas de nuestra carrera. Sin importar en donde se encuentren o si alguna vez llegan a leer estos agradecimientos, queremos darles las gracias por formar parte de nuestro proceso de formación profesional.

Al Profesor Hugo Hernando Andrade Sosa, por su valiosa dirección, orientación y colaboración en el desarrollo de este proyecto.

A la Magister Luis Eduardo Guerra González, por su contribución, sus aportes y asesoría a través de todo el proceso de desarrollo del proyecto.

A nuestras familias por el apoyo incondicional, amor constante y paciencia a lo largo de nuestras vidas.

Y a nuestros amigos de carrera con quienes compartimos nuestra etapa de formación y fueron participes de nuestro crecimiento personal y profesional.

A los integrantes del grupo SIMON de Investigación por acogernos y colaborarnos con sus aportes.

A la Universidad Industrial de Santander y a la Escuela de Ingeniería de Sistemas e Informática, por la formación profesional recibida.

Tabla de contenido

Tabla de contenido……7
INTRODUCCIÓN……13
1.PRESENTACIÓN……15
1.1.DESCRIPCIÓN GENERAL DEL DOCUMENTO...……15
1.2.OBJETIVOS……16
1.2.1.Objetivo general……16
1.2.2.Objetivos específicos……17
1.3.JUSTIFICACIÓN……17
2.MARCO TEÓRICO……19
2.1. MANTENIMIENTO DE SOFTWARE……19
2.1.1. Tipos de mantenimiento software……20
2.1.2. Actividades de mantenimiento……21
2.2.CALIDAD DE SOFTWARE……22
2.2.1 Selección de métricas o atributos de calidad para la evaluación……23
2.2.2. Métricas……24
2.3.EVALUACIÓN DE SOFTWARE……25
2.3.1. Pruebas de software……26
2.4.ENTORNO DE DESARROLLO Y HERRAMIENTAS UTILIZADAS……27
2.4.1. Framework ria……27
2.4.2. Vaadin……27
2.4.3. Arquitectura de las aplicaciones de vaadin……28
2.4.4. Entorno de desarrollo eclipse……32
2.4.5. Servlets……34
2.4.6. Web service……34
2.5.LISTA DE REQUERIMIENTOS CON LOS QUE SE REALIZÓ LA EVALUACIÓN……34
3.MARCO METODOLÓGICO……37
3.1.INTRODUCCIÓN……37
3.2.MANTENIMIENTO SEGÚN LA NORMA……37
4.DESARROLLO DEL PROYECTO……41
4.1.RECEPCIÓN DEL PROTOTIPO……41
4.1.1. Instalación y exploración de la aplicación……41
4.1.2. Análisis del prototipo y resultados de pruebas anteriores……41
4.2.DISEÑO Y PLANEACIÓN DE PRUEBAS……43
4.2.1. Definición de la finalidad de la evaluación……43

4.2.2. Inspección de la información necesaria para la evaluación…….43
4.2.3. Evaluación del software saludmovil mediante nuevas pruebas……...44
4.2.4. Selección de herramientas software para la realización de pruebas y valoración del software…….55
4.3.IMPLEMENTACIÓN DE PRUEBAS……..57
4.4.EVALUACIÓN Y ANÁLISIS DE RESULTADOS…….62
4.4.1.Resultados de las pruebas……..62
4.5.CIERRE DEL PROCESO Y ENTREGA DEL PRODUCTO……...70
4.5.1. Identificación y clasificación de mejoras……..70
4.5.2. Pruebas de aceptación……73
4.5.3. Liberación……..74
5.CONCLUSIONES……....75
6.RECOMENDACIONES……...77
BIBLIOGRAFIA……....78
ANEXOS……...79

LISTA DE FIGURAS

Figura 1Tipos de mantenimiento 20
Figura 2 Modelo para la calidad Externa e Interna (ISO, 2001) 23
Figura 3 Modelo para la calidad de uso (ISO, 2001) 23
Figura 4 Proceso de evaluación de un producto software (ISO, 1999) 25
Figura 5 Arquitectura de aplicaciones Vaadin 29
Figura 6 Metodología propuesta por fases 38
Figura 7 Interfaz de SonarQube 57
Fígura 8 Ejemplo de ejecución de un caso de prueba con Selenium 58
Figura 9 Error de despliegue 62
Figura 10 Evaluación a la aplicación web con la herramienta SonarQube 64

LISTA DE ANEXOS

Anexo A Plan de pruebas ……...…79

Anexo B Evaluación de software……..……..95

RESUMEN

TÍTULO

Evaluación y mantenimiento del sistema de información educación y comunicación en enfermedades infecciosas que cursan con fiebre aguda

AUTOR

César Andrés Alvarado Hernández

PALABRAS CLAVE

Salud móvil, evaluación, mantenimiento, pruebas, desarrollo web, Tecnologías de Información y Comunicación, Vaadin.

DESCRIPCIÓN

El grupo SIMON de investigación adscrito a la escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander (UIS) en sus labores de investigación, en particular en la línea de informática promueve el uso de tecnologías de información y comunicación que sirvan como herramientas en la búsqueda del progreso y una mejor calidad de vida de la comunidad, como es el uso de páginas web dedicadas a la promoción de la salud y, prevención de enfermedades y epidemias. El presente trabajo de grado ha sido desarrollado como apoyo al proyecto de investigación "Determinación de zonas de riesgo de transmisión de *Trypanosoma Cruzi* vía oral e implementación de un sistema de alerta temprana para Chagas agudo en la ciudad de Bucaramanga" financiado por Colciencias y a cargo del profesor Gerardo Muñoz Mantilla, de la escuela de medicina de la UIS e integrante del grupo SIMON y dando seguimiento a la investigación mediante la evaluación y el mantenimiento del primer prototipo obtenido.

El sistema tiene como proposito principal dar prioridad a pacientes febriles de una entidad de salud de primer orden, el registro de los mismos, seguimiento y alerta temprana frente a casos que sean diagnosticados como dengue o chagas y, promoción y prevención de estas mismas enfermedades.

La aplicación web tiene como función registrar temperaturas y reportar alertas de epidemias de dengue y/o chagas, aunque también ofrece otras características como la gestión de usuarios, mapa y localización de pacientes, gráficas estadísticas de enfermedades por paciente, envío de correo electrónico, etc, y fue desarrollada utilizando la plataforma Java Enterprise Edition con implementación del framework Vaadin y el motor de bases de datos MySQL.

El proceso de mantenimiento pretende evidenciar las falencias que puedan presentarse en el software, las mejoras que se le puedan realizar y las adaptaciones que se requieran para una mejor cobertura y uso.

ABSTRACT

TÍTLE

Evaluation and maintenance of the education and communication information system on infectious diseases with acute fever *

AUTHOR

César Andrés Alvarado Hernández

KEYWORDS

Mobile health, evaluation, maintenance, testing, web development, Information and Communication Technologies, Vaadin.

DESCRIPTION

The SIMON research group, attached to the School of Systems Engineering and Computer Science of the Industrial University of Santander (UIS) in its research, particularly in the line of information technology promotes the use of information and communication technologies that serve as tools In the search for progress and a better quality of life for the community, such as the use of websites devoted to health promotion and disease prevention and epidemics. The present work of degree has been developed in support of the research project "Determination of risk zones of transmission of Trypanosoma cruzi by oral and implementation of an early warning system for acute Chagas in the city of Bucaramanga" financed by Colciencias and in charge of Professor Gerardo Muñoz Mantilla, from the medical school of the UIS and member of the SIMON group.

The main purpose of the system is to prioritize febrile patients from a first-rate health entity, register them, follow up and early warning against cases that are diagnosed as dengue or chagas disease, and promote and prevent these same diseases.

The web application has the function of registering temperatures and reporting alerts for dengue epidemics and / or wounds, but also offers other features such as user management, patient mapping and location, patient disease statistics, e-mail sending, etc. , And was developed using the Java Enterprise Edition platform with implementation of the Vaadin framework and the MySQL database engine.

The maintenance process is intended to highlight the shortcomings that may occur in the software, the improvements that can be made and the adaptations that are required for better coverage and use.

INTRODUCCIÓN

Durante los últimos años nuestra región santandereana ha venido atravesando por una serie de cambios climáticos los cuales conllevan a la aparición de enfermedades originadas por diferentes agentes etiológicos afectando a personas de ciertas zonas tropicales, de escasos recursos y/o que no tienen acceso a los servicios básicos de salud. En sectores urbanos y rurales de diferentes locaciones del departamento de Santander se ha incrementado la presencia de enfermedades, como lo es el Chagas o el dengue cuyo principal síntoma es fiebre; la fiebre puede indicar muchas enfermedades que por diversos motivos no están siendo bien clasificadas y por ende los pacientes no son bien diagnosticados y a su vez no se están atendiendo correctamente los pacientes con dichas enfermedades.

De acuerdo con esta problemática surge la idea de investigar sobre las vías de transmisión de enfermedades infecciosas en las personas y buscar una manera de contrarrestar la aparición y propagación de las dolencias que aquejan a nuestra región, a través de un proyecto de investigación financiado por Colciencias en el que están involucrados el grupo SIMON y la escuela de medicina pertenecientes a la Universidad Industrial de Santander, en cuyo marco se ha avanzado en la realización de un prototipo de sistema de información educación y comunicación, llamado SaludMovil, para priorizar acciones médicas y desplegar alertas tempranas en enfermedades que se manifiestan con fiebres mediante el uso de tecnologías de la información. El propósito del sistema es darles prioridad a pacientes que presenten fiebres intensas sin diagnosticar hacerle su respectivo seguimiento y alertar si se trata de un caso de Chagas o Dengue que genere riesgo de transmisión.

El prototipo se sometió a prueba por personas de tres barrios piloto de Bucaramanga dentro de las actividades del proceso de investigación del proyecto, realizando experiencias con potenciales pacientes y otros usuarios dejando como resultado de estas actividades recomendaciones para el mejoramiento pero que no son suficientes para la completa validación del software.

En la actualidad existe un conjunto de herramientas software que se han desarrollado en las que tienen en cuenta solo el desarrollo, las pruebas se realizan al final solo como requisito para la liberación del producto y para mostrar que se cumplió con los objetivos planteados, pero en las que se deja a un lado el mantenimiento, evitando que el producto

sea mejorado o adaptado a requisitos más completos y funcionales, desperdiciando de esta manera tiempo, esfuerzo y recursos entre otros.

Es por esto que el propósito de la evaluación mediante pruebas de campo y propias del desarrollo de software del prototipo de sistema de información SaludMovil adquiere importancia ya que se desea generar un software integro, que sirva como respaldo a la investigación y soporte al sistema de alerta temprana, que continúe funcionando y que sirva como ejemplo para posibles desarrollos posteriores semejantes.

1. PRESENTACIÓN

1.1. DESCRIPCIÓN GENERAL DEL DOCUMENTO

Este informe de trabajo de grado contiene información teórica acerca del mantenimiento de software, el cual fue base de desarrollo para dar continuidad al proyecto de grado SISTEMA DE INFORMACIÓN, EDUCACIÓN Y COMUNICACIÓN EN ENFERMEDADES INFECCIOSAS QUE CURSAN CON FIEBRES AGUDAS. Además, se presenta algunos referentes teóricos sobre sistemas de información como prototipos para la implementación de sistemas de alerta temprana.

En el capítulo 1, se realiza una presentación formal del proyecto, en la cual se describe el objetivo general y los objetivos específicos de este. Además se expone la justificación del proyecto.

En el capítulo 2, se presenta el fundamento teórico del proyecto el cual se divide en cinco ítems fundamentales. En el primer ítem se define el mantenimiento software, se describen los tipos de mantenimiento y las actividades que este involucra y se nombran las normas que definen el proceso. En el segundo ítem, se describe la calidad de software como objetivo del mantenimiento y las métricas usadas para una referencia de medida. En el tercer ítem, la evaluación del sistema, lo concerniente a pruebas de software, tipos y niveles de pruebas. En el cuarto ítem, se muestra las herramientas que se usaron para el desarrollo del proyecto se ofrece una descripción de la arquitectura de las aplicaciones desarrolladas bajo el framework vaadin, utilizado para implementar el sitio web y la tecnología de Servlets. se presenta una descripción del ambiente integrado de desarrollo eclipse, utilizado para implantar el framework y con él realizar el desarrollo de la aplicación web. Finalmente en el quinto ítem se presenta información general sobre los requerimientos que se desean satisfacer en la evaluación.

En el capítulo 3, se presenta la metodología de acuerdo con el estándar base de mantenimiento.

En el capítulo 4, se describe el desarrollo del proyecto explicando la metodología empleada y presentando las actividades realizadas en cada una de las fases del ciclo de validación y verificación del producto software.

En el capítulo 5, se presenta las conclusiones que surgieron a partir del desarrollo del ambiente software.

En el capítulo 6 se presentan recomendaciones a trabajos futuros, especialmente a trabajos que involucren continuar con esta iniciativa del grupo SIMON de investigaciones.

En el capítulo 7, se muestran referencias bibliográficas utilizadas en el transcurso del desarrollo del proyecto, en diversos aspectos como programación, ingeniería del software, TICs[1], etc.

1.2. OBJETIVOS

1.2.1. Objetivo general

- Realizar la evaluación y mantenimiento del "*Sistema de información, educación y comunicación en enfermedades infecciosas que cursan con fiebres agudas*" mediante técnicas de evaluación de software y de campo con algunos usuarios.

1.2.2. Objetivos específicos

- Evaluar el ambiente software, realizando pruebas de caja negra y caja blanca, a través de herramientas que faciliten esta labor, con el propósito de encontrar fallos y proponer mejoras.
- Realizar las experiencias con el software y con diferentes usuarios para evaluar la herramienta.

[1] Tecnologías de la información y comunicación

- Realizar la recolección de nuevos requerimientos del sistema con base en el análisis de los resultados de las pruebas con usuarios.
- Realizar la corrección de errores y sugerir recomendaciones para futuros desarrollos.

1.3. JUSTIFICACIÓN

Generalmente se cree que la realización de un software culmina con la instalación y puesta en funcionamiento, pero esto es solo la parte inicial del ciclo de vida del sistema, luego de la entrega y la aceptación provisional del software, es posible que sea necesario modificar algún modulo del sistema por la detección de errores o por nuevas funcionalidades exigidas por el usuario o cliente, a esta etapa es a lo que en ingeniería del software se le conoce como mantenimiento de software.

El desarrollo de software en la actualidad ha aumentado con el avanzar de la tecnología, lo que conlleva a que existan muchos programas informáticos para realizar una o varias tareas que suplan una necesidad particular, pero que no demuestran garantías para la validez de la información generada o la satisfacción de los usuarios.

La solución al desarrollar un software de calidad está en la evaluación y el mantenimiento que nos asegura un mejor resultado al final del ciclo de desarrollo del software y en la entrega final del producto.

Existen varias razones por las cuales realizar la evaluación y el mantenimiento de software, pero una muy importante es que es mejor realizar la evaluación y mantenimiento de software en su etapa de desarrollo, ya que después puede ser más difícil de modificar, por lo que el mantenimiento se volvería costoso e impreciso y esto se debe a que el mantenimiento realizado al final complica la posterior comprensión de la funcionalidad del sistema y probablemente la estructura original requerida en las necesidades de los usuarios.

Para muchos programadores y desarrolladores de software realizar el mantenimiento es una tarea poco atractiva por lo que no le dedican mucha atención y aunque no es tan

mencionada esta actividad si es muy necesaria ya que no solo fallan los sistemas sino también la estructura física donde se encuentra implantado.

En pocas palabras el mantenimiento de software es importante porque ayuda a prever fallas, adaptar funcionalidades y a corregir errores que se puedan presentar, ya que el interés por la calidad crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades.

Dentro del área de mantenimiento de software es claro que un sistema no evoluciona por si solo puesto que a medida que pasa el tiempo es significativo revisar para determinar si puede mejorarse. La evaluación y mantenimiento del prototipo de software Salud Móvil nos permite garantizar que se está aportando a la investigación[2] y al proyecto de maestría ya que el prototipo propuesto es un elemento importante para la realización de ambos. Esta actividad de mantenimiento promueve la solución de problemas informando sobre los orígenes de las fallas, partiendo desde cuáles son sus objetivos, planteando o escogiendo un método y proponer la solución con base en recursos o herramientas informáticos.

Este trabajo de evaluación y mantenimiento consta de actividades propias aprendidas durante el transcurso del plan de estudios del programa de Ingeniería de Sistemas como lo son el diseño y producción de software, procesos de reingeniería para el mejoramiento de herramientas informáticas, administración de bases de datos, manejo de herramientas informáticas, administración de información, personal y equipos entre otras y se hace apropiado para ser considerado como proyecto de pregrado y optar por el título de Ingeniero de Sistemas UIS.

[2] Determinación de zonas de riesgo de transmisión de *T. Cruzi* vía oral e implementación de un sistema de alerta temprana para Chagas agudo en la ciudad de Bucaramanga

2. MARCO TEÓRICO

2.1. MANTENIMIENTO DE SOFTWARE

El mantenimiento de productos software se centra en la realización de actividades asociadas a la corrección de errores, la búsqueda del mejoramiento de la calidad y la adaptación de funciones; de acuerdo con el pasar del tiempo y con el avance de la tecnología los requerimientos evolucionan, dichas actividades están encaminadas conforme a una serie de normas y parámetros que conllevan a un resultado satisfactorio.

De acuerdo a este contexto, existen estándares completos donde se establecen las estrategias y técnicas óptimas para el desarrollo de esta actividad, los estándares son:

Estándar IEEE 1219

El IEEE 1219[3] Standard for Software Maintenance, hasta 1998 es el único estándar que íntegramente se ocupa del proceso de mantenimiento del software. Describe un proceso iterativo para la gestión y ejecución de las actividades del proceso. Aunque sólo menciona las fases de desarrollo y de producción de un producto de software, éstas cubren todo su ciclo de vida, cualquiera que sea su tamaño o complejidad.

Estándar ISO 14764

Este es el estándar específico sobre mantenimiento del software publicado por la ISO en 1998. El estándar internacional ISO 14764[4] presenta los requerimientos para el proceso de mantenimiento del software, contiene las actividades y tareas del mantenedor, proporciona una guía que explica cómo llevar a cabo el proceso de mantenimiento y establece definiciones para los distintos tipos de mantenimiento. La guía es aplicable a la planificación, ejecución y control, mantenimiento, revisión y evaluación del proceso de mantenimiento.

La norma propone un plan que forma parte de la estrategia de mantenimiento, dicho plan es usado para guiar a los mantenedores de software, explica la necesidad de

[3] Referencia http://standards.ieee.org/cgi-bin/status?1219-1998
[4] Referencia de la fuente para la norma http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064

realizar mantenimiento, refiriéndose a quién efectúa ese trabajo y cómo se hace, contiene la documentación y responsabilidades de todos los involucrados. Además, debe incluir qué recursos hay disponibles para el mantenimiento, dónde se hace y cuándo comienza. Una vez definido dicho plan, el estándar propone establecer una guía para desarrollar el mantenimiento.

2.1.1. Tipos de mantenimiento software: Existen cuatro tipos de mantenimiento de acuerdo con la definición con los cuales abordar el mantenimiento:

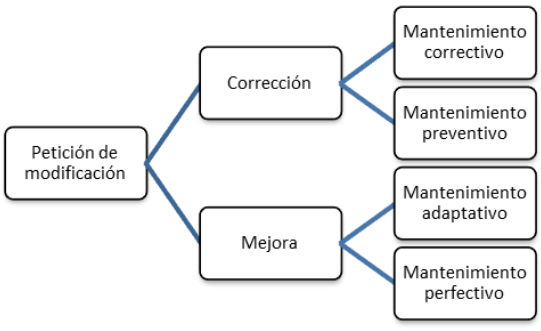
Mantenimiento correctivo: Consiste en realizar la búsqueda y localización de las fallas y después corregirlas o eliminarlas.

Mantenimiento preventivo: Consiste en detectar y corregir defectos antes de que generen fallas en el sistema. También se considera como la modificación del software para mejorar sus propiedades (por ejemplo, aumentando su calidad y/o su mantenibilidad) sin alterar sus especificaciones funcionales.

Mantenimiento perfectivo: Consiste en modificar el software para mejorar su rendimiento, pero sin alterar su estructura y funcionalidad.

Mantenimiento adaptativo: Radica en la modificación del software debido a cambios en el entorno (hardware o software) donde se ejecuta. El cambio requerido puede implicar desde un pequeño retoque en la estructura de un módulo hasta tener que reescribir prácticamente todo el programa.

Figura 1Tipos de mantenimiento



Fuente Norma ISO 14764

2.1.2. Actividades de mantenimiento: El desconocimiento de las actividades del software conlleva a que no sea valorado y que se asocie solamente a la corrección de errores sin embargo tiene un campo de acción más amplio dependiendo del tipo de mantenimiento que se realice algunos tipos de actividades son las siguientes:

- *Análisis de impacto y de costes/beneficios*: esta actividad se dedica a analizar diferentes alternativas de implementación y/o a comprobar su impacto en la planificación, coste y facilidad de operación.
- *Comprensión del cambio*: puede consistir en localizar el error y determinar su causa, o en comprender los requisitos de una mejora solicitada.
- *Diseño del cambio:* se refiere al diseño propuesto para el cambio, pudiéndose incluir un rediseño del sistema.
- *Codificación y pruebas unitarias:* se codifica y prueba el funcionamiento de cada componente modificado.
- *Inspección, certificación y consultoría*: esta actividad se dedica a inspeccionar el cambio, comprobar otros diseños, reuniones de inspección, etc.
- *Pruebas de integración:* se refiere a comprobar la integración de los componentes modificados con el resto del sistema.
- *Pruebas de aceptación:* en esta actividad, el usuario comprueba, junto al personal encargado del mantenimiento, la adecuación del cambio a sus necesidades.
- *Pruebas de regresión:* en esta actividad se somete el software modificado a casos de pruebas previamente almacenados y por los que ya pasó.
- *Documentación del sistema:* se revisa y reescribe, en caso necesario, la documentación del sistema para que se ajuste al producto software ya modificado.

2.2. CALIDAD DE SOFTWARE

Según Pressman (2005), la calidad de software es la concordancia del software producido con los requerimientos explícitamente establecidos y con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario.

Elementos que permiten evaluar la calidad de software

Para que la calidad pueda ser entendida y medida de alguna manera debe expresarse en términos que tengan sentido para el usuario, estos factores que afectan la calidad se dividen en dos grupos:

a) Defectos descubiertos en las pruebas
b) Facilidad de uso o de mantenimiento

Con base en lo anterior, estos conceptos dan una idea de cómo medir el sistema.

- Corrección: El grado en que el programa cumple con su especificación y satisfacer los objetivos que propuso el cliente.

- Confiabilidad: El grado en que se esperaría que un programa desempeña su función con la precisión requerida.

- Eficiencia: La cantidad de código y de recursos de cómputo necesarios para que un programa realice su función.

- Integridad: El grado de control sobre el acceso al software o los datos por parte de las personas no autorizadas.

- Facilidad de uso: El esfuerzo necesario para aprender, operar y preparar los datos de entrada de un programa interpretan la salida.

- Facilidad de mantenimiento: El esfuerzo necesario para localizar y corregir un error en un programa.

- Flexibilidad: El esfuerzo que demanda probar un programa con el fin de asegurar que realiza su función.

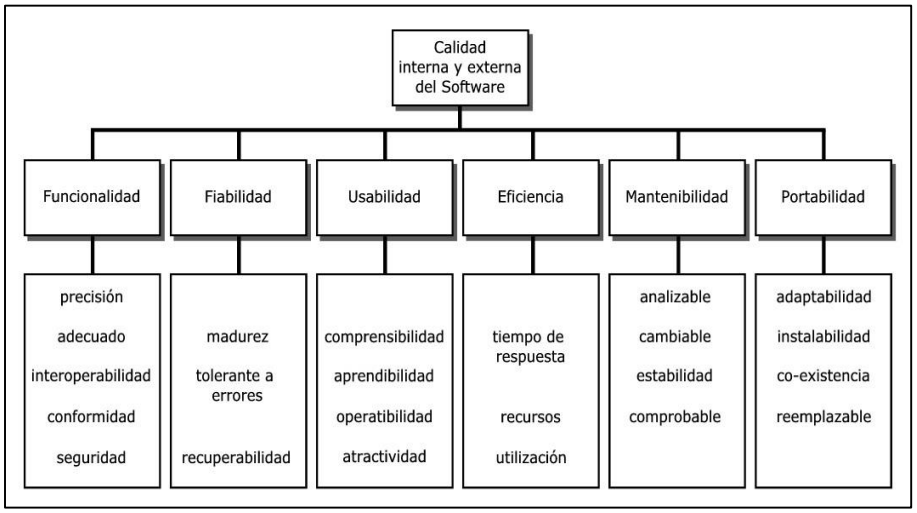
- Portabilidad: El esfuerzo necesario para transferir el programa de un entorno de hardware o software a otro.

2.2.1 Selección de métricas o atributos de calidad para la evaluación: Para poder medir la calidad y determinar un grado de esta, se establecen algunos conceptos y criterios que apoyan el proceso como lo son los modelos de calidad externa e interna y el modelo de calidad en uso. Partiendo de este contexto definimos a que hace referencia cada uno:

Calidad externa: Se mide en el comportamiento de un producto software como en una prueba.

Calidad interna: Se mide a partir de las características propias del sistema como el código fuente.

Figura 2 Modelo para la calidad Externa e Interna (ISO, 2001)



Fuente: Ingeniería del software un enfoque práctico. Roger Pressman (2005)

Calidad en uso: Se mide durante la manipulación efectiva de un usuario en un contexto determinado.

Figura 3 Modelo para la calidad de uso (ISO, 2001)

Fuente: Ingeniería del software un enfoque práctico. Roger Pressman (2005)

2.2.2. Métricas: Se define un conjunto de métricas o reglas que nos permitirán valorar el estado de la estructura del sistema y de la información de la que se dispone. Las Métricas de clase, métricas de interfaz y las métricas de paquete manejan un conjunto de conceptos sobre los cuales se derivan y organizan los atributos o propiedades.

- *Tamaño***:** consiste en medir los elementos que hacen parte del diseño, como por ejemplo número de operaciones de una clase etc.
- *Acoplamiento***:** consiste en la medida de la dependencia entre los elementos, una alta dependencia causara un impacto sobre la calidad del producto final en cuanto a su mantenibilidad
- *Complejidad:* consiste en medir el grado de conectividad entre los elementos de una unidad. Una alta complejidad influye en la comprensión del sistema y por ende se encontraría propensa a fallas.
- *Herencia:* esta clase se evalúa de acuerdo a la anchura y profundidad del árbol de herencia, esto implica que grandes o profundas tienden más a generar fallos.
- *Cohesión:* consiste en el grado en que se relacionan las responsabilidades de un componente. Una alta cohesión indica que un elemento de diseño presenta pocas responsabilidades estrechamente relacionadas, así mismo otorga una alta funcionalidad y es más comprensible.
- *Polimorfismo:* consiste en la posibilidad de que una operación tome muchas formas, permite referirse a objetos de clases diferentes mediante el mismo elemento de programa y realizar las mismas operación de varias formas
- *Encapsulación:* consiste en la ocultación de información, permite que un objeto pueda acceder a sus datos mediante sus propios métodos al mismo tiempo que los esconde de los demás objetos, esto asegura que los elementos no puedan cambiar el estado interno de otros objetos de forma inesperada, solo los propios métodos internos pueden acceder a su estado.

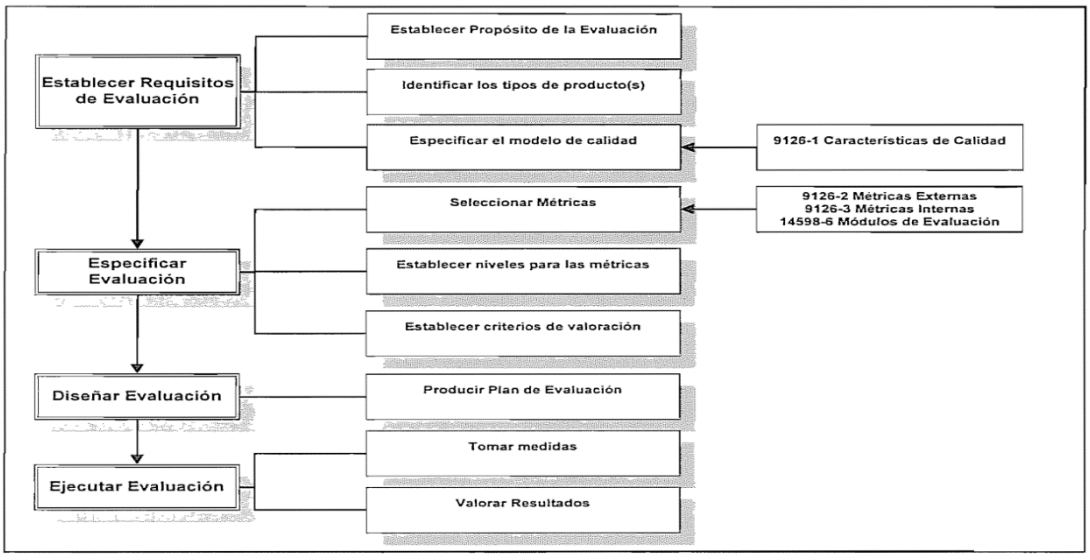
2.3. EVALUACIÓN DE SOFTWARE

La razón por las que se evalúa el software es que se necesita confiar en los datos recibidos por el sistema para que las acciones sean correctas en el caso de responder frente algún problema; es por eso que el software debe ser de calidad, dado que para respaldar la confianza que puede brindar un sistema no se puede recurrir solo a sistemas matemáticos se aplica una verificación empírica por medio de pruebas donde se tiene en cuenta los resultados esperados junto con los obtenidos, también se tienen en cuenta normas de calidad para que crezca la fiabilidad en los programas.

Existen modelos de calidad que nos ayudan a definir mejor el concepto y a especificar lo que se busca en un sistema para llegar a tener un software confiable, debido a que en el desarrollo de software es difícil fijar de manera concreta a que nos referimos con calidad.

De acuerdo con los estándares de mantenimiento, el modelo para realizar una evaluación adecuada a un producto software contempla varios elementos y tareas, en la siguiente imagen se presenta una guía:

Figura 4 Proceso de evaluación de un producto software (ISO, 1999)



Fuente: Ingeniería del software un enfoque práctico. Roger Pressman (2005)

2.3.1. Pruebas de software: Toda asociación que desarrolla software prueba sus productos ya que siempre quedan anomalías residuales de distinta gravedad que no

son evidentes al probar el sistema. Para entender el proceso de las pruebas de software es necesario partir de los conceptos básicos que involucra esta tarea.

Tipos de prueba

De manera general las pruebas se dividen en pruebas de verificación y pruebas de validación en donde las primeras se enfocan a que el software se esté construyendo correctamente mientras que las segundas nos ayudan a revisar que se esté construyendo el software correcto es decir que el resultado sea lo que el cliente realmente deseaba desde que dio sus requerimientos.

Niveles de pruebas

Cuando se habla de niveles de pruebas se refiere al conjunto de técnicas de verificación que se concentra en evaluar determinada parte del sistema como lo presenta el siguiente cuadro.

Tabla 1 Niveles de pruebas

Prueba	Objetivo
Unidad	Detectar errores en los datos, lógica y algoritmos
Integración	Detectar errores de interfaces y relación entre componentes
Funcional	Detectar errores en la implementación de requerimientos
Sistema	Detectar errores en el cubrimiento de los requerimientos
Aceptación	Detectar fallas en la implementación del sistema

2.4. ENTORNO DE DESARROLLO Y HERRAMIENTAS UTILIZADAS

2.4.1. Framework ria: Un framework RIA (Rich Internet Applications) es un framework diseñado para apoyar el desarrollo de sitios webs dinámicos, aplicaciones web y servicios web. Este tipo de frameworks intenta aliviar el exceso de carga asociado con actividades comunes usadas en desarrollos web.

2.4.2. Vaadin: Vaadin5 es un framework para el desarrollo de aplicaciones web Open Source modernas o también llamadas RIA (Rich Internet Applications) que posee una librería con numerosos componentes de usuario con los cuales construir aplicaciones web con un alto nivel de funcionalidad. Este framework se ejecuta del lado del servidor lo que significa que la mayoría de la lógica y por tanto la mayor carga del trabajo recae en el servidor.

Vaadin utiliza tecnología Ajax del lado del cliente cuya base es GWT (Google Web Toolkit) para renderizar la interfaz de usuario en el navegador. Las aplicaciones Google Web Toolkit se escriben en Java y son automáticamente “traducidas” a JavaScript y HTML. GWT es muy cómodo para el desarrollador porque le mantiene al margen de estas acciones de presentación y sin la necesidad de instalar ningún plugin para su presentación.

Debido a que HTML, JavaScript y otras tecnologías de navegador son esencialmente invisibles a la lógica de la aplicación, el navegador web actual como sola plataforma de cliente ligero. Un cliente ligero muestra la interfaz de usuario y comunica eventos del usuario con el servidor en un nivel bajo. El control de la lógica de la interfaz de usuario se ejecuta en un servidor web basado en Java, junto con su lógica de negocio. Por el contrario, una arquitectura normal de cliente-servidor incluye una gran cantidad de comunicaciones de aplicaciones específicas entre el cliente y el servidor.

Así como el motor del lado del cliente de JavaScript se ejecuta en el navegador, no se necesitan complementos del navegador para usar aplicaciones hechas con Vaadin. Esto le da una ventaja fuerte sobre frameworks basados en Flash, Applets de Java, u otros complementos. Detrás del modelo de desarrollo impulsado por el servidor, se hace el

[5] https://vaadin.com/docs/-/part/framework/introduction/intro-overview.html

mejor uso de las técnicas de AJAX (JavaScript y XML Asíncrono) que hacen posible la creación de Aplicaciones Dinámicas de Internet (RIA) que son tan sensibles e interactivas como aplicaciones de escritorio.

Además cuenta con el apoyo de Google Web Toolkit (GWT) para una amplia gama de navegadores, de modo que al desarrollador no tiene que preocuparse por la compatibilidad del navegador. Los programas de GWT están escritos en Java, pero compilados en JavaScript, liberando así al desarrollador de aprender JavaScript y otras tecnologías del navegador.

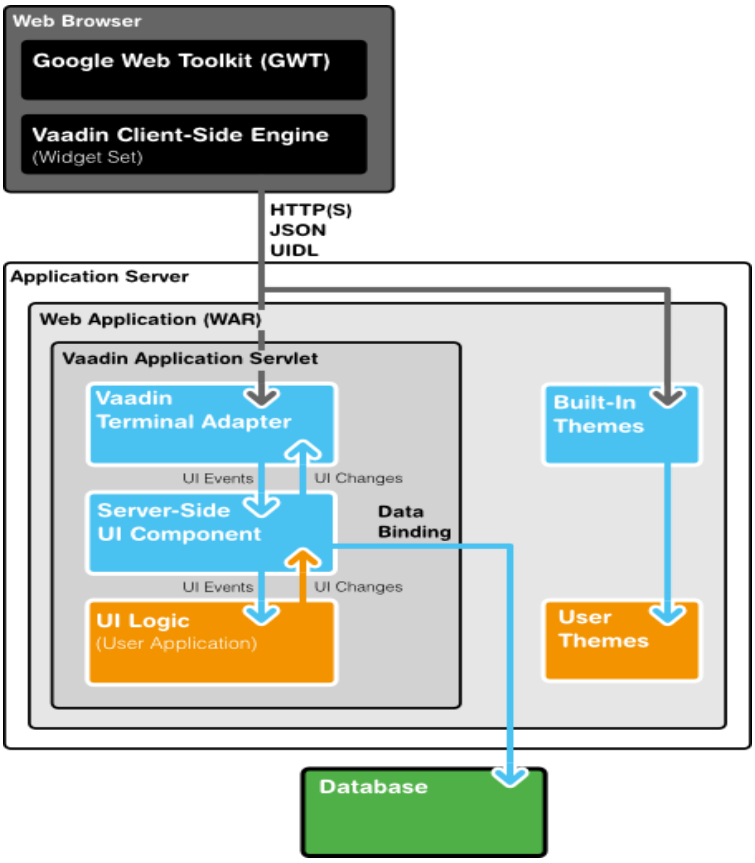
2.4.3. Arquitectura de las aplicaciones de vaadin: Las aplicaciones Vaadin consisten en una multitud de componentes de interfaz de usuario, temas para controlar la apariencia, y un modelo de datos que le permite vincular los componentes de interfaz de usuario directamente a los datos. De fondo se emplea un adaptador de terminal[6] para recibir las peticiones de los navegadores web y hacer las respuestas renderizando las páginas.

Una aplicación que utiliza Vaadin se ejecuta como un Servlet en un servidor web de Java, sirviendo peticiones HTT P. El adaptador de terminal recibe solicitudes de los clientes a través del servidor web de la API de Java Servlet, y los interpreta para los eventos de usuario para una determinada sesión en particular. Las sesiones son rastreadas utilizando cookies. Los eventos están asociados con los componentes de interfaz de usuario (UI) y entregados a la aplicación, el cual es controlado con *manejadores de eventos* (listeners). Si la lógica de la aplicación realiza cambios en los componentes de interfaz de usuario del lado del servidor, el adaptador de terminal los interpreta en el navegador web generando una respuesta. El motor del lado del cliente que se ejecuta en el navegador recibe las respuestas y los utiliza para hacer los cambios necesarios en la página en el navegador.

El nivel superior de una aplicación de usuario consiste en una clase *application* que hereda de *com.vaadin.Application.* Esto crea los componentes de interfaz de usuario que necesita, recibe los eventos relacionados con ellos, y hace los cambios necesarios en los componentes.

[6] Capa de abstracción permite a los usuarios utilizar aplicaciones Vaadin con prácticamente cualquier navegador web.

Figura 5 Arquitectura de aplicaciones Vaadin



Fuente http://vaadin.com/download/book-of-vaadin/current/translations/es/html/architecture.html

Las partes más importantes de la arquitectura y su función son las siguientes:

- **Componentes de Interfaz de Usuario**

  Contiene todos los componentes Vaadin con los que el usuario puede interactuar. Cada componente del lado del servidor tiene una contraparte del lado del cliente, con la cual el usuario interactúa. Los componentes del lado del servidor pueden serializar[7] la conexión del cliente con un adaptador de terminal. Los componentes del lado del cliente, a su vez, pueden serializar la interacción del usuario de nuevo a la aplicación, que es recibida en los componentes del lado del servidor como eventos. Los componentes transmiten estos eventos a la lógica de la aplicación.

[7] Consiste en un proceso de codificación de un objeto de una cantidad de datos en un medio de almacenamiento.

- **Motor de Lado del Cliente**

  El Motor del Lado del Cliente de Vaadin gestiona la renderización en el navegador web a través de GWT. Este comunica la interacción del usuario y los cambios de la interfaz de usuario con el Adaptador de Terminal del lado del servidor usando el Lenguaje de Definición de Interfaz de Usuario (UIDL), un lenguaje basado en JSON[8]. Las comunicaciones se realizan mediante HTTP asincrónico o peticiones HTTPS.

- **Adaptador de Terminal:** Los componentes de interfaz de usuario no se renderizan ellos mismos directamente como una página web, pero usan un Adaptador de Terminal. Esta capa de abstracción permite a los usuarios utilizar aplicaciones Vaadin con prácticamente cualquier navegador web. Para permitir este tipo de abstracción, los componentes de interfaz de usuario comunican sus cambios al Adaptador de Terminal, que los renderiza para el navegador del usuario. Cuando el usuario hace algo en la página web, los eventos son comunicados al adaptador de terminal (a través del servidor web) como peticiones asincrónica AJAX. El adaptador de terminal entrega los eventos del usuario a los componentes de interfaz de usuario, que se los entrega a la lógica de la aplicación de interfaz de usuario.

- **Temas (Estilos)**

  La interfaz de usuario se separa entre la presentación y la lógica. Mientras que la lógica de interfaz de usuario se maneja como código de Java, la presentación se define en temas como CSS. Vaadin proporciona unos temas por defecto. Los temas del usuario pueden, además de las hojas de estilo, usar plantillas HTML que definen diseños personalizados y otros recursos, tales como imágenes.

- **UIDL**

  El Adaptador de Terminal dibuja la interfaz de usuario a la página web y cualquier cambio en él usando un Lenguaje de Definición de Interfaz de Usuario (UIDL). Las comunicaciones UIDL se realizan mediante JSON (JavaScript Object Notation), que es un formato de intercambio de datos ligero que es

[8] **JSON:** *JavaScript Object Notation*, es un formato ligero para el intercambio de datos.

especialmente eficaz para interconectar con el código AJAX basadas en JavaScript en el navegador.

- **Eventos**

  La interacción del usuario con los componentes de interfaz de usuario crea eventos, que primero son procesados del lado del cliente con JavaScript y luego son pasados todo el camino a través del servidor HTTP, el adaptador de terminal, y las capas de los componentes de usuario a la aplicación.

- **Modelo de Datos**

  Además del modelo de interfaz de usuario, Vaadin proporciona un modelo de datos para interconectar los datos presentados en los componentes de interfaz de usuario. Usando el modelo de datos, los componentes de interfaz de usuario pueden actualizar los datos de la aplicación directamente, sin necesidad de ningún código de control. Todos los componentes de interfaz de usuario utilizan este modelo de datos internamente, pero pueden ser vinculados a una fuente de datos separada. Por ejemplo, puede vincular un componente *table* a una respuesta de las consultas SQL.

2.4.4. Entorno de desarrollo eclipse: Eclipse9 consiste en un Entorno de Desarrollo Integrado (IDE, Integrated Development Environment) abierto y extensible. Un IDE es un programa compuesto por un conjunto de herramientas útiles para un desarrollador de software, como los son el editor de código, compilador10/intérprete11 y un depurador12. Eclipse sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, PHP o Python. A la plataforma base de Eclipse se le pueden añadir extensiones (plugins) para extender la funcionalidad.

El entorno de desarrollo Eclipse se basa en la Plataforma de Cliente Rico (Rich Client Platform RCP). En una arquitectura de servidor cliente el término “cliente rico” se refiere a clientes que proveen la interfaz gráfica de usuario. A menudo los clientes ricos son

[9] http://web.archive.org/web/20130402233256/http://www.eclipse.org/whitepapers/eclipse-overview.pdf
[10] Analiza el programa fuente y lo traduce a otro equivalente escrito en otro lenguaje (por ejemplo, en el lenguaje de la máquina).

[11] Analiza el programa fuente y lo ejecuta directamente, sin generar ningún código equivalente.

[12] Programa usado para probar y depurar (eliminar los errores) de otros programas.

aplicaciones extensibles mediante conexiones y módulos, además tienen la ventaja de ser fáciles de distribuir y actualizar, como una función de actualización automática en línea o un mecanismo de inicio por internet (por ejemplo, mediante el inicio de una página Web Java).

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (plugins) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software.

Los siguientes componentes constituyen la plataforma de cliente rico para Eclipse:

- Plataforma principal - inicio de Eclipse, ejecución de plugins.
- OSGi - una plataforma para compilación estándar.
- El Standard Widget Toolkit (SWT) - widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto.
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes.

Los widgets de Eclipse están implementados por una herramienta de widget para Java llamada Standard Widget Toolkit (SWT), a diferencia de la mayoría de las aplicaciones Java, que usan las opciones estándar Abstract Window Toolkit (AWT) o Swing. La interfaz de usuario de Eclipse también tiene una capa GUI intermedia llamada JFace, la cual simplifica la construcción de aplicaciones basadas en Standard Widget Toolkit.

Adicionalmente Eclipse se extiende usando otros lenguajes de programación como son C/C++ y Python, y trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos. La arquitectura de complementos permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene por qué ser usado únicamente para soportar otros lenguajes de programación.

En cuanto a las aplicaciones clientes, Eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web etc.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. Mediante diversos complementos estas herramientas están también disponibles para otros lenguajes como C/C++ (Eclipse CDT) y en la medida de lo posible para lenguajes de script no tipados como PHP o Javascript. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadato en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se actualice el espacio de trabajo correspondiente.

2.4.5. Servlets: Los Servlets son programas escritos en Java que se ejecutan del lado de servidor. Un servlet es una clase Java, son usados para implantar aplicaciones del lado del servidor, reciben peticiones y devuelven resultados por el protocolo HTTP, normalmente una página HTML, pero también puede ser cualquier tipo MIME5, una imagen, etc.

2.4.6. Web service: Un **servicio web** (en inglés, *Web Service* o *Web services*) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. En otras palabras es una máquina que atiende las peticiones de los clientes web y les envía los recursos solicitados.

2.5. LISTA DE REQUERIMIENTOS CON LOS QUE SE REALIZÓ LA EVALUACIÓN

El sistema cuenta con unos requerimientos que lo definen y que especifican lo esencial de satisfacer luego de su implementación y nos enfoca en el proceso de verificación y validación del mantenimiento.

Requerimientos funcionales:

- El manejo de acceso por roles, así como consideraciones mínimas de seguridad.
- La visualización, registro y actualización de distintos tipos de usuario
- Un usuario de tipo administrador el cual va a tener acceso total a las funciones del sistema y debe ser el único que podrá administrar el contenido de la página web.
- Registrar temperaturas del paciente por medio de teléfono móvil, call-center o web.
- Asignar a los usuarios un código con fecha de uso para una cita médica prioritaria.
- Registrar, actualizar y eliminar citas médicas asignadas a un usuario
- Alertar sobre la no utilización de un código asignado para una cita prioritaria
- Alertar inmediatamente a los tipos de usuario correspondientes sobre un usuario con dengue o Chagas.
- Visualizar mapas con la localización de usuarios en sus distintos estados de salud registrados
- Visualizar gráficas estadísticas que permitan evaluar la receptividad y el funcionamiento del sistema
- La visualización de un historial de citas médicas por usuario
- La administración de grupos de usuarios (Grupos TSCV) que tienen como fin velar por el bienestar y la salud de las personas a nivel comunitario. Cada grupo asignado a una zona geográfica específica.
- La visualización, registro y modificación de acciones realizadas por Grupos TSCV a la comunidad.

Requerimientos no funcionales

- El sistema estará dividido en 2 partes, la parte WEB y la parte Móvil. La interfaz de usuario se implementará sobre un navegador Web. La aplicación móvil solo servirá para el envío y recepción de temperaturas y mensajes respectivamente.
- El software WEB debe estar diseñado y desarrollado sobre la plataforma multinivel de servicios, haciendo uso de la especificación estándar Java EE. Y debe poder ser utilizado bajo cualquier plataforma e independiente del navegador.

- El software móvil debe estar diseñado y desarrollado sobre la plataforma java, haciendo uso de la especificación estándar Java ME. Debe ser portable, capaz de ser utilizado en cualquier celular que tenga instalado la máquina virtual java para teléfonos móviles.
- El sistema debe mantenerse a bajo costo por esto se debe tener en cuenta en el desarrollo del mismo la utilización de software gratuito que permitan el desarrollo de programas para el funcionamiento del mismo

3. MARCO METODOLÓGICO

3.1. INTRODUCCIÓN

Para el desarrollo de este proyecto se acogió una metodología de evaluación de software basada en el estándar ISO/IEC 14764 cuyas principales actividades establecen la implementación de una estrategia de pruebas que se adapta a la estructura del proyecto y plantea una forma ordenada del proceso que contribuye a que las actividades sean más eficientes y consecuentes para llevar el control adecuado de las fallas y por consiguiente de sus soluciones. De acuerdo con el estándar los cambios realizados en el mantenimiento se abordan con base a una serie de fases.

Dentro de cada una de estas fases, el estándar define una serie de procedimientos que se han de llevar a cabo y con los que se identifica la documentación, las personas y productos de software que intervienen.
Esta norma plantea un proceso de mantenimiento con gran nivel de detalle y documentación para su desarrollo, haciéndolo muy útil y necesario en los lugares que se realiza mantenimiento del software.

3.2. MANTENIMIENTO SEGÚN LA NORMA

Para el mantenimiento del software existen dos estándares:

- ISO/IEC 14764
- IEEE 1219

ISO/IEC 14764: Este estándar describe la estructura del proceso de mantenimiento software. Esta norma identifica adecuadamente qué hacer en las actividades y tareas a desarrollar en el proceso de mantenimiento.

IEEE 1219: Esta norma describe un proceso iterativo para la gestión y ejecución de actividades para la planificación de mantenimiento de productos en desarrollo como para los existentes.

De acuerdo con las anteriores definiciones se toma como base el estándar ISO/IEC 14764 para el mantenimiento del software SaludMovil porque describe íntegramente el proceso y actualmente ha sustituido la norma IEEE 1219.

La norma propone un plan que forma parte de la estrategia de mantenimiento, dicho plan es usado para guiar a los mantenedores de software, explica la necesidad de realizar mantenimiento, refiriéndose cómo se hace y contiene la documentación. Además, debe incluir qué recursos hay disponibles para el mantenimiento, dónde se hace y cuándo comienza. Una vez definido dicho plan, el estándar propone establecer una guía para desarrollar el mantenimiento.

Requisitos de la guía o plan

Los requisitos que debe contener esta guía de acuerdo con el estándar son:

- La descripción del sistema al que se le brinda soporte, aquí se especifican todos los detalles del sistema.
- Identificación del estado inicial del software, para saber cuáles son los cambios nuevos realizados.
- Descripción del soporte para facilitar el comienzo del desarrollo del mantenimiento del software. (Involucrados en el proceso).

Fases de la evaluación y mantenimiento

Esta norma define cambios en un producto de software a través de un proceso de mantenimiento dividido en fases. Las fases propuestas se presentan de acuerdo al proceso del ciclo de vida del desarrollo, un desarrollo evolutivo y las fases del desarrollo de la evaluación y mantenimiento como se menciona a continuación:

- Análisis del prototipo, pruebas y resultados anteriores.
- Diseño y planeación de pruebas.
- Implementación de pruebas.

- Evaluación y análisis de resultados.
- Cierre del proceso y entrega del producto.

Figura 6 Metodología propuesta por fases

Análisis del prototipo, pruebas y resultados anteriores. En esta fase se recibe el prototipo del software para tener la primera interacción con el sistema y de esta manera tener una percepción del funcionamiento del software e idear las pruebas que se necesitan para entender su estructura y comportamiento. A demás se analizan las pruebas realizadas durante su desarrollo.

Diseño y planeación de pruebas. En esta fase se planea la forma de aplicar pruebas al sistema junto con el tipo de pruebas y lo que se pretende evaluar, el objetivo principal de la evaluación y su alcance, se diseñan los formatos, se escogen los escenarios y se especifican los recursos necesarios para la ejecución de la guía por último se exploran y elijen las herramientas software que ayudan a sistematizar el proceso. Esta guía como manifiesta la norma se realiza con el fin de tener un proceso ordenado y controlado, que facilite el seguimiento y entendimiento.

Implementación de pruebas. Una vez se tiene el plan de pruebas definido se pone en marcha la evaluación con las pruebas descritas en la guía, para conocer la estructura y funcionamiento del software.

Evaluación y análisis de resultados. Luego de aplicar los diferentes tipos de pruebas se organiza la información obtenida para verificar el estado del sistema, encontrar defectos, solucionar fallas y validar la congruencia de la funcionalidad que ofrece de acuerdo a los requerimientos propuestos en un inicio. A demás se tendrá en cuenta la extracción de nuevos requerimientos si surgen como sugerencia de mejora al software y ampliación de su funcionalidad.

Cierre del proceso y liberación del producto. Después de evaluar completamente el sistema y este haya sido aceptado por los usuarios se procede a la liberación del sistema, junto con la documentación necesaria.

Actividades del proceso de Mantenimiento

Estos son los aspectos fundamentales en cuanto a la estrategia de mantenimiento que propone el estándar. Las actividades que comprende el proceso son:

- Identificación del problema(s).
- Implementación del proceso.
- Análisis de modificaciones.
- Implementación de modificaciones.
- Revisión del mantenimiento.
- Retiro.

4. DESARROLLO DEL PROYECTO

4.1. RECEPCIÓN DEL PROTOTIPO

4.1.1. Instalación y exploración de la aplicación: Previo a la realización de las pruebas de caja negra y caja blanca se inicia probando todos los recursos presentes como la documentación e instaladores, se hizo el proceso en un computador portátil y un ordenador de escritorio con los requerimientos necesarios para su instalación localmente y usando dos sistemas operativos diferentes Windows en su distribución 7 home Basic y Linux en sus distribuciones Debian 7 Whezzy y/o Centos 7 red-hat.

Dado que uno de los requerimientos del proyecto es que fuera realizado a bajo costo y poder utilizar software libre para el funcionamiento y mantenimiento del mismo, se hizo muy conveniente contar con este requisito para la búsqueda de las herramientas necesarias, ya que todas se encuentran en la red y están disponibles las últimas versiones de los programas.

4.1.2. Análisis del prototipo y resultados de pruebas anteriores: Para tener un seguimiento ordenado y claro del mantenimiento de la aplicación la metodología especificada en el plan propone una serie de etapas, de acuerdo con esta, la primera etapa es un análisis de pruebas y resultados anteriores. Se realizaron cuatro pruebas de caja negra, de estrés, de casos de uso y de campo.

Tabla 2 Pruebas previas

Tipo de prueba	Errores obtenidos
Caja negra	➢ En el uso de mapas y geolocalización, la cual no puede ser corregida dado que este servicio es proporcionado gratuitamente por entidades independientes a la aplicación. ➢ El framework Vaadin identifica como error el evento del mouse de clic izquierdo en el navegador Internet Explorer v.10.

De campo	➢ Al momento de registrarse en la aplicación, dado que el navegador de los ordenadores utilizados era Internet Explorer en su versión 7, el cual no soporta la aplicación de geo-localización de Google Maps. ➢ Para evaluar la aplicación móvil se usaron teléfonos celulares de baja gama para en el registro de temperaturas de pacientes y visualización de alertas, de los cuales se obtuvieron los resultados esperados y permitieron observar la integración de los mismos. ➢ Se evaluó la receptividad del sistema por los distintos tipos de usuarios, observándose que la aplicación fue de fácil manejo y las actividades se realizaron exitosamente. Además se calificó el sistema como rápido y con poco tiempo de espera, al responder las peticiones creadas por el usuario.
De estrés(revisar)	➢ Tomando como base los resultados obtenidos, se permite afirmar que para menos de 160 usuarios conectados simultáneamente el sistema responde en un tiempo razonable (menos de 5 segundos).
Casos de uso	➢ Al realizar las pruebas de casos de uso se verificó que el sistema obedece completamente con la secuencia de los diagramas, en consecuencia, efectúa los casos de uso que se generaron para la realización de los requerimientos. Por tanto, la aplicación cumple con los requerimientos planteados.

Resultados

En síntesis, los resultados de las pruebas previas evidencian que no se muestra un gran número de errores al obtener los resultados de estas, o que los errores encontrados de

acuerdo al tipo de pruebas no generan interferencia con el funcionamiento del prototipo, por esto es necesario corregir las fallas encontradas y realizar diferentes tipos de pruebas para tratar de encontrar otros fallos o para mejorar la calidad del sistema.

La descripción se encuentra en el anexo_.

4.2. DISEÑO Y PLANEACIÓN DE PRUEBAS

4.2.1. Definición de la finalidad de la evaluación: Se establece el propósito de la evaluación para justificar el cumplimiento del objetivo que se desea satisfacer:

- Verificar la conformidad del estado del prototipo respecto a las necesidades primarias las cuales fueron la base para la realización del software.
- Evaluar la arquitectura implementada con el fin de encontrar fallos y complicaciones que impidan el funcionamiento del sistema.
- Documentar las evidencias encontradas y las sugerencias de solución a estas.

4.2.2. Inspección de la información necesaria para la evaluación: La información disponible para evaluar el software SaludMovil se obtuvo de la tesis de grado "*Sistema de información, educación y comunicación en enfermedades infecciosas que cursan con fiebres agudas"* la cual se compone de diagramas, código fuente y requerimientos. El código fuente se muestra en lenguaje Java.

Tabla 3 Información disponible para la evaluación.

Clase	Información disponible
Requerimientos	Funcionales
	No funcionales
Diagramas	De clase
	De secuencia
	De base de datos
Código	Numero de paquetes
	Numero de clases

4.2.3. Evaluación del software saludmovil mediante nuevas pruebas: Se realizó el proceso de evaluación del software mediante la complementación de pruebas a través de herramientas software que automaticen la actividad y de esta manera obtener información más detallada que la recopilada manualmente.

4.2.3.1. Plan de pruebas: Se diseñó un plan de pruebas que reúne toda la información necesaria para idear y controlar el esfuerzo de probar el software SaludMovil. Este plan especifica la descripción del software es decir su principal propósito, objetivos, lo que queremos lograr junto con las herramientas usadas para el desarrollo del trabajo. Este documento se encuentra en el Anexo a.

Criterio de Ejecución del Plan de Pruebas

- Conjunto de pruebas documentado incluyendo escenarios para el desarrollo de las mismas.
- Claridad en el procedimiento para la realización de las pruebas.
- El entorno de pruebas sea el adecuado para la realización de las mismas.
- Toda la documentación requerida debe estar disponible.

Criterio de Terminación del Plan de Pruebas

- Todas las pruebas se ejecutan.
- Las pruebas demostraran que existe un grado satisfactorio de capacidad.

4.2.3.2. Tipos de pruebas:

Pruebas de caja blanca

Este tipo de pruebas se centró en los detalles que conforman la aplicación y la manera en como cada parte o elemento del software interactúa con otro para crear en conjunto el aplicativo.

La realización de esta clase de pruebas es importante en los procesos de mantenimiento y evaluación de software porque nos permite conocer la arquitectura de la aplicación y por consiguiente el funcionamiento de esta, también es útil para ver posibles errores que afecten al código en cuanto a su calidad ya que esta es importante para medir los aciertos en cuanto al proceso de mantenimiento se trata.

Para esta labor se usaron las herramientas Sonar, SDmetrics y Ess-model, las dos primeras realizan evaluaciones similares pero se usaron ambas para complementar los resultados obtenidos y de esta manera tener un informe más completo y detallado.

El proceso consistió en conocer la estructura de la herramienta, entender su funcionamiento y determinar lo que sería factible modificar, de acuerdo con esto se usaron varias herramientas para automatizar el proceso y poder definir la cantidad de líneas de código, el número de clases y paquetes, el valor del conjunto de atributos para cada clase y en total y encontrar falencias en cuanto a la estética y buenas prácticas de programación.

Tabla 4 Descripción de prueba de caja blanca.

Objetivo de la Prueba:	• *Conocer la estructura del sistema*
Estrategia:	• Someter el código a evaluación para conocer su arquitectura mediante herramientas software
Recursos requeridos:	• *Eclipse IDE* • *Código fuente de la aplicación* • *Herramientas software: Sonar, SDMetrics, Ess-Model, Excel*
Criterios de evaluación:	• *Verificar el diseño y la organización del software para crear un plan de mantenimiento.*
Observaciones	• *El éxito de esta prueba depende de los archivos fuente y su buen estado, no debe contener modificaciones después de que fue entregado.*
Entregable	• *Informe de resultado de la prueba* • *Documentación y/o corrección de manuales o guías*

Pruebas de caja negra

El tipo de pruebas de caja negra comprueba las relaciones de entrada y salida entre las unidades del sistema. Dado que las unidades se comunican mediante interfaces definidas, su objetivo es confirmar que hace la unidad sin saber cómo, revisando que las entradas y salidas de las unidades relacionadas sean correctas.

Esta clase de pruebas nos permite verificar y validar el software en cuanto a su funcionalidad unitaria y completa permitiéndonos saber si se construye el software adecuado de la manera precisa.

La técnica establecida para el desarrollo y documentación de las pruebas de caja negra fue:

Se diseñaron casos de prueba de acuerdo con la funcionalidad especificada en la descripción del software, se usó la herramienta Selenium para grabar cada caso y de ahí poder exportarlo en formato java.

Cada caso consistió en tomar un módulo y ejecutarlo donde su resultado ofrecio información para realizar el seguimiento de los pacientes y observar:

- Lo que ocurría si los datos eran ilógicos o falsos.
- El tiempo que tardaba el sistema en dar respuesta a las peticiones.

Para de esta manera confirmar que el software es consistente en cualquier caso de mala manipulación por parte del usuario.

En cada caso se ejecutaron datos correctos e incorrectos para comprobar la firmeza del sistema y evidenciando algunos errores, se usaron diferentes datos para ejecutar el caso varias veces.

Los casos de prueba se tomaron basados en los requerimientos del prototipo para de esta manera ver que funcionalidades exigen corrección y cuales se mantienen vigentes, también se usaron actividades exploratorias y administrativas pertenecientes al software.

Tabla 5 Descripción de prueba de caja negra

Objetivo de la Prueba:	• *Verificar que los resultados obtenidos en los diferentes casos sea coherente con lo esperado.*
Estrategia:	• Probar la mayor cantidad posible de casos en los que haya intercambio y manipulación de información
Recursos requeridos:	• *Equipos con el software salud móvil* • *Software Selenium* • *Editor de texto*
Criterios de evaluación:	• *Los resultados esperados coincidan con lo especificado en cada caso.*
Observaciones	• *Esta prueba no incluye…*
Entregable	• *Resultado Pruebas de caja negra* • *Informe de resultado de la prueba* • *Documentación y/o corrección de manuales o guías*

Formato de casos de pruebas

Formato para la descripción e identificación de cada caso de prueba empleado para la evaluación del sistema.

Tabla 6 Formato de casos de prueba

Información General	
Identificador de caso de uso:	<Identificador del caso de uso a probar>

<table>
<tr><td colspan="2">Nombre de caso de uso:</td><td colspan="3"><Nombre del caso de uso a probar></td></tr>
<tr><td colspan="2">Descripción Prueba:</td><td colspan="3"><Descripción de la prueba que se va a realizar></td></tr>
<tr><td colspan="2">Responsable:</td><td colspan="3"><Nombre de persona que ejecuta la prueba></td></tr>
<tr><td colspan="5">Prerrequisitos</td></tr>
<tr><td colspan="5"><Prerrequisitos que se deben cumplir para iniciar la prueba></td></tr>
<tr><td colspan="5">Descripción de Casos de Prueba</td></tr>
<tr><td colspan="5">Caso: <Explicación breve del caso de prueba a ejecutar*.></td></tr>
<tr><td colspan="5">Instrucciones de Prueba
<Pasos que se deben seguir para poder realizar correctamente la prueba></td></tr>
<tr><td colspan="3">Escenarios de prueba</td><td>Respuesta esperada de la aplicación</td><td>Coincide (Si/No)</td></tr>
<tr><td>Campo</td><td>Valor</td><td>Tipo escenario</td><td></td><td></td></tr>
<tr><td><Nombre de campo a probar></td><td><Dato de prueba></td><td><Correcto/Incorrecto></td><td><Descripción completa de respuesta que debe dar la aplicación al usar el campo especificado con el dato de prueba></td><td><sí/no, según la respuesta obtenida></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="5">Criterios de Aceptación
<Lista de chequeo de las condiciones que se deben cumplir para aprobar el caso de uso que se esta probando.></td></tr>
</table>

Pruebas de sistema

Este tipo de pruebas buscan discrepancias entre el programa y el propósito del sistema enfocándose en los errores durante el proceso de diseñar la especificación de un requerimiento. Esta prueba no se trata de probar las funcionalidades del sistema ya que

sería redundante con las pruebas funcionales o de caja negra, su propósito particular es comparar el producto con sus objetivos originales o necesidades de los usuarios.

Según lo anterior se intenta demostrar:

a) Que el sistema no cumple con todas sus objetivos para forzar al reconocimiento de fallas y para tener una idea del nivel de calidad que tiene actualmente.

b) Que los módulos que componen el software interactúen unos con otros tal y como fue diseñado.

- **Pruebas de Instalación**

Con la experiencia nos damos cuenta que existen aplicaciones o sistemas software complicadas de instalar y que necesitan muchos pasos que ni sabemos para que se hacen, estos procesos provocan que los usuarios no tengan una experiencia acertada con el software y ya que la primer interacción del usuario con el sistema es la instalación hace que estas pruebas sean importantes para evitar que esta mala práctica influyan en la búsqueda de un producto diferente.

Se prueban los elementos que componen la aplicación para verificar que la documentación es correcta y fácil de seguir para una instalación sencilla.

Tabla 7 Prueba de Instalacion

Objetivo de la Prueba:	• *Verificar la instalación de la aplicación en diferentes entornos de hardware y software, siguiendo un procedimiento definido.*
Estrategia:	• *Basados en el Plan (recursos de hardware-software) para la aplicación, gestionar un entorno de implementación para realizar las pruebas.*
Recursos requeridos:	• *Equipos con distintas plataformas (Windows, Linux) para probar la versatilidad y mantenibilidad.* • *Script base de datos y procedimientos para el sistema* • *Archivo WAR para desplegar la aplicación en el servidor de aplicaciones web*
Criterios de evaluación:	• *Que el sistema funcione correctamente en el entorno de pruebas.* • *Que el proceso de instalación no presente fallas en el momento de su ejecución.*

	• *Que se haya tenido un manual claro de instalación.*
Observaciones	• *Esta prueba no incluye el correcto funcionamiento de la consola administrativa del servidor donde se realiza el despliegue, así como del software base.*
Entregable	• *Resultado Pruebas de Instalación* • *Informe de resultado de la prueba* • *Documentación y/o corrección de manuales o guías de instalación*

- **Pruebas de Integridad de datos**

Se consideró recomendable realizar las pruebas de bases de datos y procedimientos de manera independiente al software para poder confirmar que todas las formas de usar la información no generan pérdidas ni modificaciones sin autorización y obtener seguridad y confiabilidad en la información que se tiene al hacer consultas y demás operaciones con datos.

Tabla 8 Descripción de prueba de integridad de datos.

Objetivo de la Prueba:	• *Verificar que la información no sufra modificaciones por la manipulación de los diferentes usuarios o por procesos en la base de datos.*
Estrategia:	• Probar la base de datos independiente de la aplicación. • Invocar cada acceso a la base de datos usando datos validos e inválidos. • Verificar que cada proceso ocurra de manera correcta y que los datos de retorno sean los esperados en cada caso específico • Integrar la aplicación y la base de datos, luego realizar las solicitudes de información
Recursos requeridos:	• *Equipos con distintas plataformas (Windows, Linux) para probar la versatilidad y mantenibilidad.* • *Script base de datos y procedimientos para el sistema* • *Software para la creación y administración de bases de datos*

Criterios de evaluación:	• *Que el sistema funcione correctamente en el entorno de pruebas.* • *Que la manipulación de información no cambie de manera inesperada* • *.*
Observaciones	• *Esta prueba no incluye…*
Entregable	• *Resultado Pruebas de Integridad de datos* • *Informe de resultado de la prueba* • *Documentación y/o corrección de manuales o guías*

- **Pruebas de desempeño**

Esta prueba se diseñó para evidenciar el trabajo del software en tiempos de ejecución dentro del contexto de un sistema integrado. Se pretende ver que acontecimientos aparecían durante la exploración del software como tiempos de respuesta, información ofrecida, e interrupciones a pesar de los errores humanos en la manipulación del sistema. Así como el uso del prototipo bajo ciertas condiciones o bajo las prestaciones mínimas requeridas.

Tabla 9 Descripción de prueba de desempeño

Objetivo de la Prueba:	• Verificar y validar el desempeño en cuanto a los requerimientos sobre los cuales se diseñó y la usabilidad del prototipo
Estrategia:	• *Ejecutar casos de prueba y comparar los resultados de las mismas con un número determinado de peticiones al sistema* • *Identificar las variaciones en las respuestas*
Recursos requeridos:	• *Equipos con el software* • *Red de banda ancha*
Criterios de evaluación:	• *Que el sistema funcione correctamente en el entorno de pruebas.* • *Que la manipulación de información no cambie de manera inesperada* • *Los tiempos de espera sean mínimos y tolerables* • *El consumo de memoria es aceptable*

Observaciones	• Estas pruebas son sensibles a las condiciones sobre las cuales se ejecuta
Entregable	• *Resultado Pruebas de desempeño* • *Informe de resultado de la prueba* • *Documentación*

Prueba de campo

La realización de esta actividad se hará por fases

- La primera fase se realizara de la siguiente forma:

Cada usuario tendrá la oportunidad de explorar el software sin ninguna clase de orientación para probar lo intuitivo y usable que es el software, podrá registrarse y saltar de un menú a otro para familiarizarse con el ambiente como complemento la persona deberá registrar cualquier suceso o interrogante que encuentre durante la actividad en el siguiente formato.

Tabla 10 Formato de incidencias de la prueba con usuarios.

Nombre	Doc. Identidad	Fecha
Suceso	Tipo de usuario	Nombre de actividad

Como actividad principal cada uno registrara temperaturas para tener información en la base de datos que se usara posteriormente.

- La segunda fase se hará de la siguiente manera:

Como primera medida para esta etapa se realizara la descripción del software, explicando claramente su propósito.

Una vez los usuarios estén registrados el administrador concederá a cada uno un tipo de usuario donde se le asignaran una serie de actividades o casos de prueba acorde con su tipo y llenara un formato de preguntas donde plasmara los resultados de los procesos propuestos.

- Como tercera y última fase cada usuario llenara una encuesta donde responderá cuestiones acerca del software independientemente del tipo de usuario que haya manejado y donde también dejara constar sus inconformidades y sugerencias.

Los documentos completos de la prueba se encuentran en el anexo_.

Recurso humano

El personal empleado para la realización del proyecto y pruebas se compone de estudiantes, profesores, y gente del común. El propósito es que sea evaluado por gente con diferente tipo de conocimientos para ampliar la imagen o concepto que inspire el software y de igual manera sea un sistema más adaptado a los usuarios finales.

Tabla 11 Recurso humano para evaluación y mantenimiento

Recursos Humanos		
Rol	**Mínimo Recursos Recomendado**	**Responsabilidades Especificas**
Encargado de la evaluación y mantenimiento del software	1	• Identificar, Priorizar e implementar Casos de Pruebas. • Diseñar y Ejecutar Plan de Pruebas (Pruebas Unitarias e Integración). • Documentar Pruebas. • Criterios de Aceptación Pruebas.
Asesor del proceso de evaluación y mantenimiento	1	• Aprobación del diseño y plan de pruebas. • Identificar, Priorizar e implementar Casos de Pruebas. • Documentar Pruebas.

Analista de Desarrollo organizacional y procesos	1	• Criterios de Aceptación Pruebas Funcionales (basado en requerimientos funcionales) y Pruebas de documentación (Requerimientos).
Analista de Pruebas	1	• Validar y ejecutar Casos de Pruebas. • Diseñar y Ejecutar Plan de Pruebas (Pruebas de sistema). • Administración de Reportes. • Documentar Pruebas. • Realizar seguimiento al Plan de Pruebas y la ejecución de las diferentes pruebas. • Criterios de Aceptación Pruebas sistema (basado en requerimientos no funcionales). • Reportes de la ejecución de las pruebas • Acompañamiento en las pruebas de usuario
Usuarios	Mínimo 6*	• Identificar, Priorizar, diseñar e implementar Casos de Pruebas. • Ejecutar Pruebas de Usuarios. • Es necesario conformar un equipo de usuarios funcionales, determinado en un número mínimo de 7, para realizar los casos de pruebas con la metodología establecida, se les debe brindar capacitación en el tema y asignar esta tarea en el cronograma, para que sea terminada antes de la ejecución de pruebas funcionales internas.

4.2.4. Selección de herramientas software para la realización de pruebas y valoración del software: Durante el desarrollo del proceso de la evaluación, se exploró en diversas fuentes, en busca de herramientas que pudieran realizar parcial o completamente la evaluación de los componentes (diagramas, modelos, código fuente etc.) del software SaludMovil. A continuación se provee una breve descripción de las herramientas encontradas y seleccionadas:

SonarQube

Es una plataforma de código abierto que sirve para gestionar la calidad del código. Sonar es una aplicación web con las siguientes características: está basada en reglas, alertas, rangos, exclusiones y configuración; permite configuración online, dispone de una base de datos y permite combinar métricas en conjunto. Sonar permite extensiones a través de plugins para abarcar nuevos lenguajes de programación, para añadir nuevas reglas o para añadir nuevas métricas. Inicialmente, cubre el lenguaje de programación Java, sin embargo, ya existen plugins de código libre y comerciales que extienden la herramienta para cubrir lenguajes como Flex, PL/SQL o Visual Basic.

Ess-Model

Con ESS-MODEL, usted puede ver los diagramas de clases a partir del código. ESS-MODEL maneja Delphi / Kylix y los archivos fuente de Java, además produce una documentación completa en HTML con diagramas de clase.

Características

General

- Generación automática de diagramas de clases UML estándar de la fuente.

Entrada

- Java (.java) archivos de código fuente.
- Java. (.class) archivos binarios.

Navegación

- Explorador de árbol para facilitar la navegación del modelo de objetos.
- Visión en miniatura del diagrama.

Diagramas

- Diagramas de clases estáticas.
- Diseño automático inteligente.
- visibilidad de filtrado.
- Se indican las asociaciones y la herencia.

Instalación

- Archivo exe 700KB, no necesita instalación.

Otros

- Comando completo de control de línea.
- Delphi IDE integración.
- Interfaz de usuario fácil de arrastrar y soltar.
- Increíblemente rápido.

Salida

- Vista previa o generar documentación HTML con diagramas.
- Modelo de exportación de datos en formato XMI.
- Guardar diagrama como mapa de bits.
- Copia de todo o parte del diagrama en el portapapeles.

SDMetrics

Es una herramienta que analiza las propiedades estructurales de los modelos UML, para ello utiliza métricas orientadas a objetos para medir el diseño, la complejidad y la relación entre clases.

SDMetrics evalúa diagramas UML y para ello define un gran número de métricas que agrupa en: métricas de clases, métricas de interfaces, métricas de paquetes, métricas de casos de uso, métricas de máquina de estados, métricas de actividad, métricas de componentes y métricas generales de diagramas.

Selenium

Selenium se trata de un framework web compuesto por dos herramientas: Selenium IDE y SeleniumWebDriver. La primera, un plugin de Firefox que te genera un entorno de desarrollo y que permite crear casos de prueba para aplicaciones web. La segunda, Selenium WebDriver, ejecuta las pruebas. Este entorno de automatización de pruebas automáticas opera en los principales navegadores (IE, Mozilla, Chrome y Opera).

Además, permite pruebas para dispositivos móviles, para iPhone y Android. Utiliza los siguientes lenguajes: Python, Ruby, Java y C#. La licencia es “Apache 2.0 License”.

4.3. IMPLEMENTACIÓN DE PRUEBAS

Estrategia usada para la evaluación

Se complementó la evaluación realizando diferente tipos de pruebas para hacer un análisis más extenso y minucioso, y de esta manera conocer todos los elementos que componen el sistema incluyendo su documentación, para asegurar la calidad del producto en desarrollo y poder brindar a los usuarios un software fácil de ejecutar y entender, pero de gran utilidad.

Pruebas de caja blanca

Para la ejecución de este tipo de pruebas, se hizo uso de la herramienta Sonarqube la cual nos permitió la identificación y clasificación de problemas estructurales en la aplicación. El procedimiento para realizar la valoración fue el siguiente:

Se realiza la descarga de los instaladores y se guardan en el disco en este caso:

C: \sonarQube

C: \sonar-scanner

C: \sonar-examples

- server
- scanner
- repositorio de proyectos

Creamos una base de datos con el nombre sonar para guardar la información de la evaluación

Ubicamos la carpeta donde guardamos la descarga y se inicia el servidor ejecutando el archivo StarSonar.bat

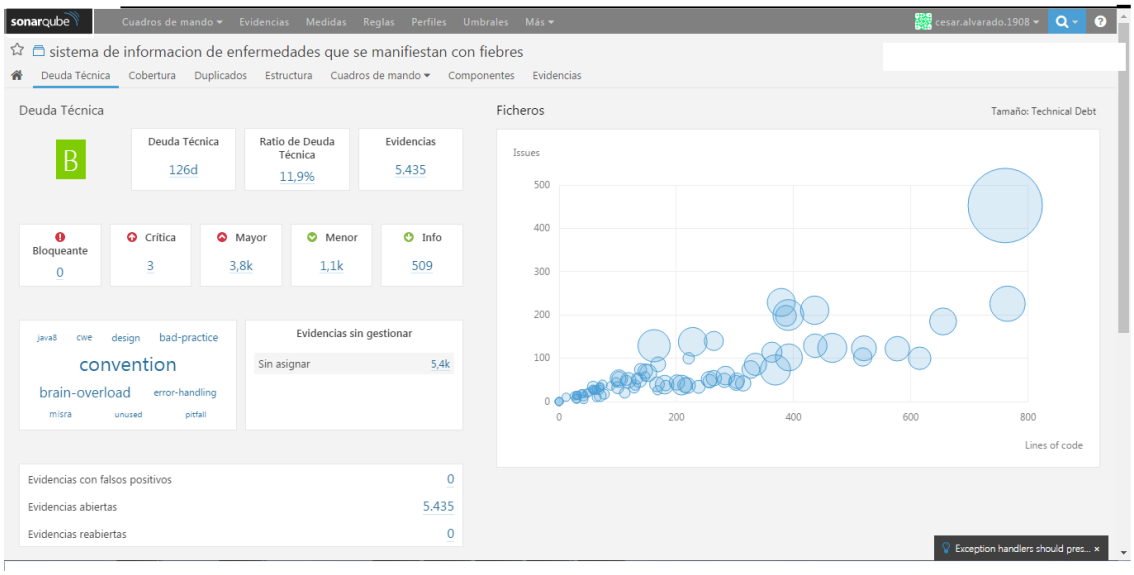
```
C:\sonarqube\bin\windows-x86-xx\StartSonar.bat
```

Para analizar los archivos abrimos una consola y ejecutamos las siguientes líneas de comandos

```
–cd C:\sonar-examples\projects\languages\java\sonar-runner\java-sonar-runner-simple
-C:\sonar-scanner\bin\sonar-scanner.bat
```

Para ver la plataforma y ubicar los resultados abrimos nuestro navegador y tecleamos localhost:9000.

Figura 7 Interfaz de SonarQube



Pruebas de caja negra

Se diseñaron los casos de prueba para el producto software, teniendo en cuenta las entradas posibles y los resultados esperados. Para la ejecución de este tipo de pruebas se usó la herramienta Selenium la cual nos genera un script de código en java y una grabación del procedimiento realizado para realizar la valoración. El proceso para el estudio fue el siguiente:

Instalamos el plugin en nuestro navegador y luego lo ejecutamos

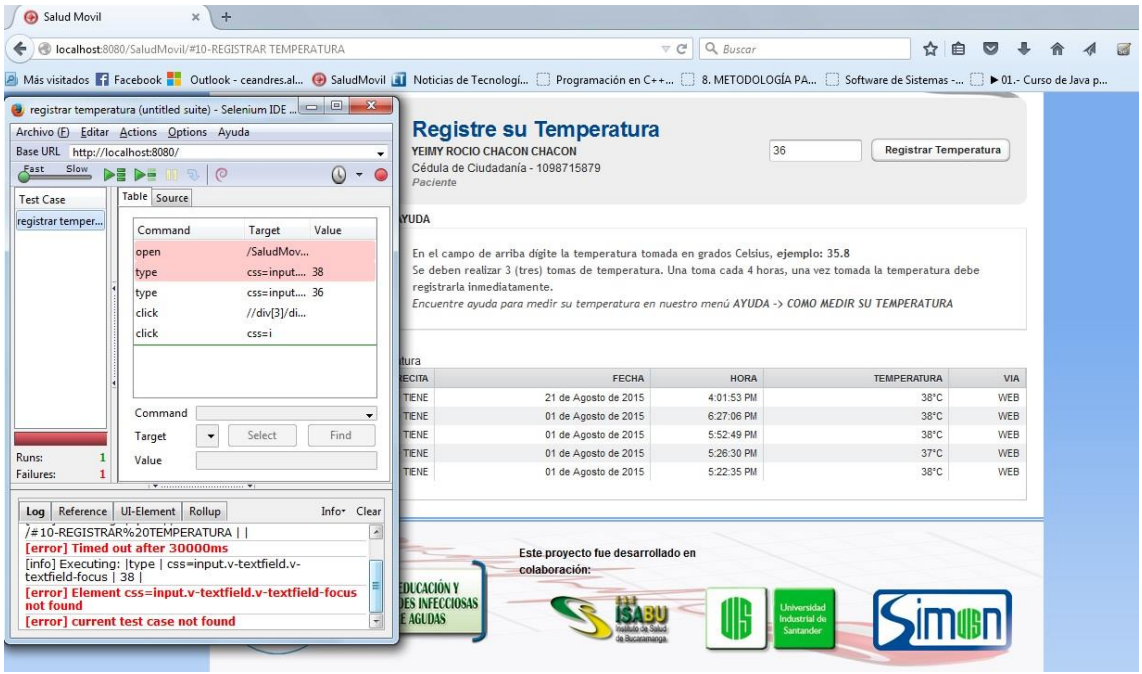
Abrimos en el navegador nuestra aplicación web

Iniciamos la grabación del caso de prueba con Selenium y simultáneamente ejecutamos el caso de prueba en la página

Una vez terminado el caso de prueba este es guardado, luego lo exportamos como un archivo java para vincularlo con la aplicación y probar con diferentes datos la funcionalidad del software. También es posible realizar la prueba con la herramienta directamente en el navegador cambiando los datos en los campos que ofrece la misma, pero en algunos casos no se ejecuta como se espera o genera errores que podrían confundir con la comprensión de la aplicación y de lo que realmente causa el defecto.

Repetimos el mismo proceso para todos y cada uno de los casos de prueba planteados.

Fígura 8 Ejemplo de ejecución de un caso de prueba con Selenium



Pruebas de instalación

Para la realización de esta actividad se usaron los siguientes recursos de hardware:

Computador portátil

- Windows 7 Home Basic- Service pack 1
- Procesador AMD E-450
- Memoria RAM de 4 GB
- Disco duro de 750 GB

Computador de escritorio

- Debían whezzy 7 ó Centos 7 Red-Hat
- Procesador Intel Pentium II
- Memoria RAM 2GB
- Disco duro de 500 GB

Recursos software.

- Servidor web Apache tomcat 7
- Motor de base de datos mysql 5.6 ,(mysqlworkbench para Windows) (phpmyadmin para Linux)
- JDK 7.
- Eclipse Java EE IDE for web developers. Versiones juno y mars

Instalación en plataforma Windows

Para la instalación de la aplicación web en un servidor local se necesitó de un conjunto de herramientas entre las cuales se encuentra, Entorno de desarrollo java Eclipse IDE junto con la instalación de plug-ins como el framework Vaadin en su versión 6 y 7, Mysql workbench para el diseño de las base de datos relacional, servidor web apache tomcat en su versión 5 o superior, herramienta de administración de bases de datos heidi y Java development kit 7 (JDK 7).

La instalación de las tecnologías de desarrollo no tuvo ningún problema y con base en la guía de instalación propuesta se completó la configuración de las herramientas, luego se usó el código fuente de la aplicación para desplegar el sistema localmente en donde se presentaron algunos errores.

- Luego de importar el archivo WAR[13] dentro del entorno eclipse presentaba errores al ejecutar el código.
- Los scripts de la base de datos presentaban algunos problemas por lo que no era posible lograr una conexión con la aplicación.
- La instalación de muchos plug-ins en eclipse presenta inconvenientes internos en el entorno, haciéndolo lento y en algunos casos provocando fallos en el

[13] Web Application Archive - Archivo de aplicación web es un archivo usado para distribuir una colección de elementos que constituyen una aplicación web

despliegue de la aplicación debido a los tiempos de respuesta sin importar el sistema operativo en el que actúe.

Luego de la corrección de los problemas anteriores se logró el despliegue de la aplicación web para dar paso al siguiente proceso de pruebas y de esta manera seguir con el mejoramiento del sistema.

Instalación en plataforma Linux

Para el despliegue de la aplicación en el servidor local se instalan y configuran los paquetes necesarios como Mysql Server 5, PhpMyAdmin, Apache Tomcat 5 o superior y el Java development kit (JDK7) mediante comandos o un gestor de descargas, después de tener el software necesario se ejecutan los scripts y demás archivos. Se obtuvieron los siguientes inconvenientes:

- Al importar el archivo WAR dentro del entorno eclipsen linux presentaba errores al ejecutar el código.
- Los scripts de la base de datos presentaban algunos problemas de sintaxis por lo que no era posible lograr una ejecución exitosa.
- Es posible desplegar la aplicación pero no permite conexión con la base de datos.

Luego de realizar las exhaustivas correcciones a los problemas encontrados en la instalación se amplía la guía de instalación junto con indicaciones de cómo solucionar inconvenientes en dado caso que se presenten, como por ejemplo la actualización de librerías, la ampliación del tamaño de archivos que implementa el gestor de aplicaciones o servidor web apache tomcat y la ejecución de scripts de la base de datos.

Pruebas de campo o de usuarios

Se contó con la participación de estudiantes de…..

Durante la primera fase de la prueba se recopilaron evidencias surgidas de la experiencia con la aplicación las cuales fueron registradas en la siguiente bitácora…

Durante la segunda y tercera fase se ejecutaron los casos de prueba propuestos para la valoración del software y se llenó un cuestionario con preguntas que permitirán medir la satisfacción de los usuarios y del éxito de la prueba…

4.4. EVALUACIÓN Y ANÁLISIS DE RESULTADOS

Con la evaluación realizada se pudo identificar fallos que afectan la calidad del software para su posterior corrección junto con nuevos requerimientos o sugerencias que ayudaran a que el software sea aún más funcional y se mantenga en crecimiento ofreciendo de esta manera más soporte a las necesidades de los usuarios.

4.4.1. RESULTADOS DE LAS PRUEBAS

Pruebas de instalación

El proceso se realizó tanto en la plataforma Windows como en Linux para ambas acciones se siguió la documentación establecida.

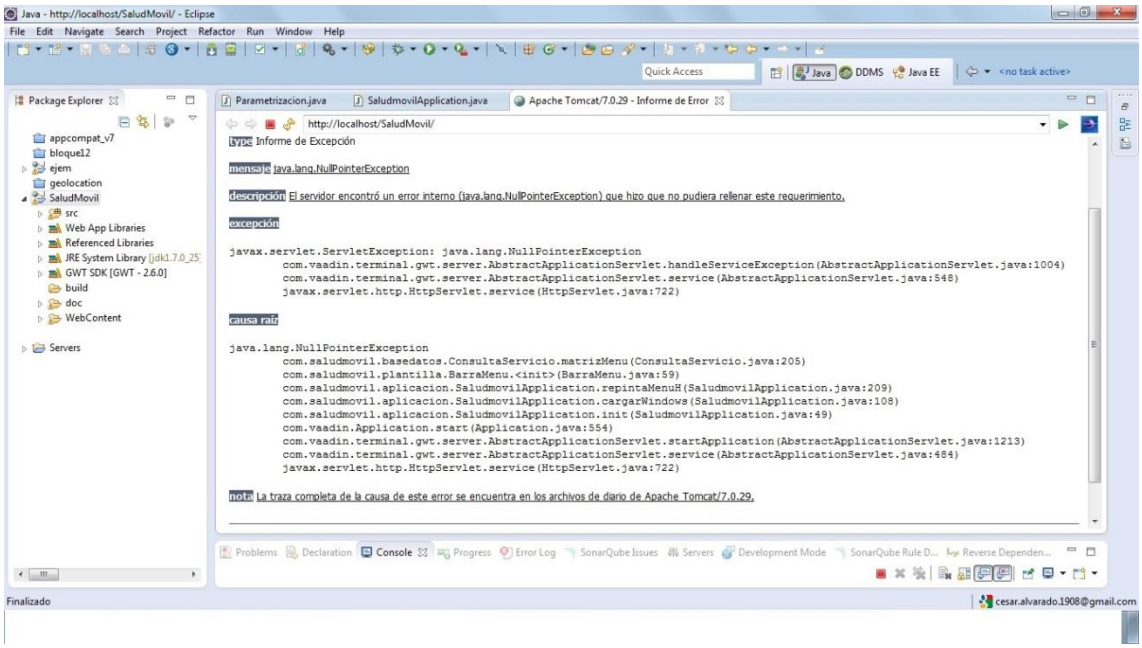
Para Linux

Se instalaron los paquetes para la distribución utilizada y posteriormente se configuraron sin dificultades, durante el despliegue se presentaron inconvenientes con la implementación del código fuente debido a incompatibilidad en la codificación de los archivos. Se logra el despliegue pero no hay conexión con la base de datos debido al conector (JDBC) de base de datos usado, la codificación entre la aplicación y la base de datos no coincide, también se presentan problemas de sintaxis en el código de la aplicación y de la base de datos.

Para Windows:

Se completó satisfactoriamente la instalación y configuración de cada uno de las herramientas necesarias para el despliegue de la aplicación en un servidor local.

Figura 9 Error de despliegue



Pruebas de integridad de datos

Para el uso del esquema de la base de datos, en los archivos se deben configurar los caracteres para los delimiter de no hacerlo se genera un error en la ejecución de los scripts de procedimientos de la base de datos.

Las consultas realizadas con el código generan los resultados esperados incluso con datos erróneos, para estos casos se generan notificaciones o advertencias que promueven la buena manipulación del sistema y evitan que el usuario continúe ejecutando un proceso al cometer errores inconscientes.

El estudio se completó de manera independiente al software y luego integrándolos para comprobar el comportamiento del sistema y que los datos no sean vulnerables en la modificación o durante el ingreso de información

Pruebas de desempeño

Durante la ejecución de los casos con los que se midió el tiempo de respuesta del software con respecto a las instrucciones solicitadas como promedio demora entre uno

y dos segundos en el intercambio de menús y demás peticiones independientemente del ancho de banda ofrecido por el servidor.

La información recibida durante las peticiones coincide con lo solicitado y sin generar demoras.

Pruebas de caja blanca

Las pruebas de caja blanca permitieron revisar la estructura del software, evidenciando defectos en cuanto a buenas prácticas de desarrollo y atributos de calidad. Se obtuvo un sumario de problemas donde la corrección de cada violación a las reglas de desarrollo permitirá que aumente la calidad del software en esta etapa, a continuación los errores:

Completitud:

- Clases que aparentemente no se usan. Este defecto se debe a que las clases son usadas como tipos de parámetros o variables en procedimientos de otras en total 51 de las 90 clases presentan esta violación a las reglas.

Correctitud:

- El 35% de las clases tienen operaciones duplicadas
- El 15% de las clases encontradas en el código fuente presentan un nombre incorrecto en relación con los diagramas obtenidos por las herramientas.
- Existen clases o atributos con nombres idénticos
- Se presentan bloques de código duplicados

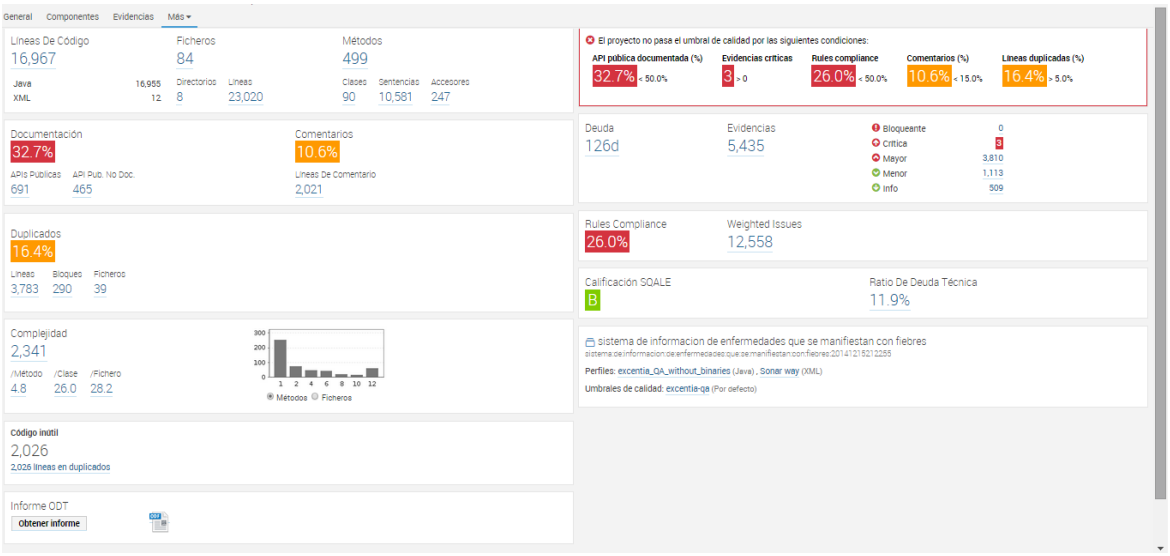
Estilo:

- El 5% de las clases tienen más de 60 atributos y operaciones es decir son clases enormes.

La corrección de los anteriores errores permitirá una facilidad de comprensión para la realización de futuros mantenimientos.

Algunas modificaciones son complicadas debido a que es posible incurrir en nuevos errores por lo que no conocemos plenamente la lógica usada por los desarrolladores es aconsejable realizar los cambios en presencia de los programadores.

Figura 10 Evaluación a la aplicación web con la herramienta SonarQube



Con ayuda de la herramienta sonar se pudo incursionar dentro del prototipo permitiéndonos conocer lo que podría afectar el código y como se podría mejorar:

Saludmovil web es un producto software desarrollado con el entorno de desarrollo eclipse y el framework vaadin en lenguaje java, posee 90 clases, 499 métodos y 247 atributos. En esta aplicación se lograron identificar problemas de complejidad, completitud, duplicidad cuyas principales evidencias son:

Tabla 12 Tamaño del código

	TOTAL
Número de líneas	23,020
Número de líneas de código	16,967
Número de archivos	84
Número de clases	90
Número de directorios o paquetes	8
Numero de métodos	499
Número de accesos	247

Número de sentencias	10,581

Tabla 13 Documentación del código

	TOTAL
Número de comentarios	2,021
Densidad de líneas de comentario	10.6%
Número de APIs sin documentación	465
Densidad de APIs documentadas	32.7%

Tabla 14 Duplicidad de código

	TOTAL
Bloques de código duplicado	290
Líneas de código duplicado	3,783
Archivos con código duplicado	39
Densidad de líneas duplicadas	16.4%

Tabla 15 Complejidad

	TOTAL
Clases complejas	26
Archivos complejos	28.2
Métodos complejos	4.8

Tabla 16 Normas de cumplimiento y buenas prácticas

	TOTAL
Número total de evidencias	5,435
Bloqueantes	0
Críticas	3
Mayor	3,810
Menor	1,113

Tipo Info	509
Reglas de compilación	26%

Pruebas de caja negra

Se diseñaron los casos de prueba de acuerdo a los requerimientos del sistema sobre los cuales íbamos a evaluar la funcionalidad del sistema.

Tabla 17 Lista de casos de prueba.

Tipo de usuario	Casos de prueba	Resultados esperados
Paciente	• Validar usuario	Registrar e ingresar
	• Registrar temperaturas	Tomas de temperaturas y seguimiento
	• Ver historial de citas	Lista de citas asignadas a el mismo
	• Visualizar mapas	Mapa con las zonas de riesgo de las enfermedades
Medico	• Asignar cita	Agregar cita a un paciente diagnosticado con fiebre sin diferenciar
	• Ver historial de citas del paciente	Cantidad de citas que ha tenido un paciente
	• Buscar pacientes	Información de los pacientes a su cargo
Secretaria	• Registrar nuevos usuarios	Agregar nuevos usuarios con determinado tipo de usuario
	• Revisar notificaciones	Lista notificaciones que crea el sistema para su tipo de usuario en especifico
	• Buscar código de cita	Información de las citas asignadas
TSCV	• Administrar alertas	• Lista de alertas generadas por citas prioritarias de pacientes, por incremento de casos de fiebres • Posibilidad de generar acciones

	• Ver pacientes	Datos de pacientes incluyendo citas prioritarias
	• Generar acciones	Crear una actividad preventiva
Call-center	• Registrar paciente	Agregar uno o más pacientes
	• Ver pacientes	Datos de pacientes incluyendo citas prioritarias
	• Ver notificaciones	Lista de actividades o citas programadas para pacientes.

Primera ejecución de los casos de pruebas

Tabla 18 Resultados prueba caja negra

Actividad	Resultado
Total casos de prueba	15
Casos de prueba satisfactorios	11
Casos de prueba fallidos	4
No ejecutados	0
Bugs reportados	11
Críticos	3
Medios	0
Bajos	2
Info	6

Segunda ejecución de casos de pruebas.

Tabla 19 Resultados Prueba de caja negra (segunda Iteración)

Actividad	Resultado
Total casos de prueba	15
Casos de prueba satisfactorios	13
Casos de prueba fallidos	2
No ejecutados	0
Bugs reportados	8
Críticos	1
Medios	0
Bajos	2
Info	5

- Los errores críticos que surgieron se refiere a clases duplicadas que a criterio de la aplicación podrían generar errores más graves en la ejecución o funcionalidad del sistema.
- (click-Locator) El caso de prueba en el que se involucre una búsqueda muestra un error al registrar un usuario en el ítem para seleccionar el tipo de documento, la acción de dar click para escoger causa la recarga hacia una nueva página que no existe o no se muestra, aunque no presenta inconveniente para el desarrollo del proceso solo sugiere una modificación del código.
- los mapas de la ubicación de usuarios no se presenta en algunas ocasiones debido al servidor externo que lo soporta y debido a que dichas funcionalidades provistas por el framework se encuentran obsoletas o en modificación, solo aparecen las coordenadas en el momento en que se hace el registro
- la aplicación esta prevista de mensajes de error o advertencias en el caso de que se inserten datos erróneos, previniendo de esta manera que los resultados a las peticiones contengan información inválida.
- Se corrigieron las fallas en el software que eran viables y que no causaran daño en la funcionalidad del software, y se ejecutaron nuevamente los casos de prueba con nuevos datos para evitar obtener los mismos resultados.
- Luego del segundo estudio persiste la falla en la geolocalización de pacientes para cualquier tipo de usuario, los demás anomalías que se presentan son sugerencias del framework para mejorar la calidad de la programación o evitar dualidades en la interpretación que hace el compilador.

Las pruebas de caja negra evidenciaron que se cumple en su gran mayoría con los requerimientos especificados inicialmente pero igualmente demuestran que los errores en el código de no ser corregidos afectarían la agilidad del software y los tiempos de respuesta.

Algunas funcionalidades no se pueden corregir definitivamente porque provienen de desarrolladores externos y a medida que pasa el tiempo son mejoradas y actualizadas dejando las que tenemos en uso como obsoletas, solo se puede hacer uso de ellas manteniendo las versiones o al importar un archivo JAR que contenga el código del cual se sirve.

Pruebas de campo

De acuerdo a los resultados de las pruebas anteriores se extraerán modificaciones y sugerencias a realizar.

Con la ejecución de la prueba de campo es probable que surjan nuevas necesidades que sean de carácter vital para mejorar la funcionalidad y calidad de la aplicación.

4.5. CIERRE DEL PROCESO Y ENTREGA DEL PRODUCTO

En esta etapa se da fin al primer proceso de mantenimiento en donde se realizaron las mejoras a los fallos encontrados, la extracción de nuevos requerimientos y la postulación de sugerencias que contribuyan a mejorar el prototipo y/o ampliar su funcionalidad.

4.5.1. Identificación y clasificación de mejoras: Para mantener una correcta organización y control del mantenimiento es aconsejable hacer una revisión periódica de los elementos de modificación, de esta manera se puede tener un seguimiento y dar continuidad al proceso en el futuro.

Para el caso del sistema de información SaludMovil se condenso en una tabla las modificaciones las cuales cuentan con las siguientes características:

- Asignar un Número de Identificación.
- Clasificación del tipo de mantenimiento.
- Análisis de la modificación.
- Realizar una estimación de la magnitud de la modificación.
- Determinar la prioridad de la modificación.
- Asignar tareas planificadas para su implementación.

Análisis

Para este análisis se tuvo en cuenta los resultados obtenidos en la verificación del software mediante la implementación de las pruebas técnicas y de campo realizada al código fuente del sistema los cuales permitieron conocer la estructura interna, el estado

de los atributos, los posibles problemas de diseño y las indicaciones más apropiadas para la corrección y mejoramiento.

Estudio de viabilidad

Luego de considerar las soluciones o modificaciones se concluyó que todas las fallas en código no son corregidas debido a que algunas generarían complicaciones para la ejecución del software, estas falencias son librerías que sirven de complemento al framework y el API de este. Los defectos que no se modificaran son mínimos y corresponden a estilo sin embargo funcionan y permanecerán en la versión que tienen ya que actualizar a una versión consecuente implicaría la modificación de todo el código, puesto que cambiar una dejaría como obsoleta otra.

MODIFICACIONES

Una vez se identificó la Modificación, comienza el Mantenimiento del Software. En esta fase los procedimientos a seguir son:

• Asignación de un Número de Identificación:
• Clasificación del tipo de mantenimiento.
  - Perfectivo
  - Correctivo
  - Adaptativo
- Clase en la que se presentó el defecto.
- Métrica que se infringió.

• Descripción de la modificación para determinar si se acepta, se deniega o se evalúa.
  - Descripción de la modificación.

• Realizar una estimación preliminar de la magnitud de la modificación:
  - Grande (Requiere de bastante trabajo y tiempo).
  - Mediana (Requiere de trabajo y tiempo considerable).
  - Pequeña (Requiere de poco trabajo y tiempo).

• Priorizar la modificación en términos de:

- Alta (Urgente).
- Medio (Necesario).
- Bajo (Opcional).

• Asignar Solicitudes de Modificación a bloques de tareas planificadas para su implementación.

  ▪ Tareas que se deberían ejecutar para su realización.

- Resolución
  - Sin resolver
  - Eliminada
  - corregida

A continuación se muestran las modificaciones realizadas después de la evaluación del software.

Tabla 20 Modificaciones

Id	Tipo de mantenimiento	Estimación	Prioridad	Descripción de modificación	Actividades
01	Correctivo	Grande	Alta	Modificar el código de la aplicación web para lograr el despliegue y conexión con la base de datos.	• Revisar el código. • Ubicar la clase y línea. • Modificar las sentencias
02	Correctivo	Grande	Alta	Corregir la sintaxis del código de la base de datos para evitar fallos en conexiones y despliegue.	• Editar las sentencias de tablas y procedimientos • Probar esquema
03	perfectivo	Mediana	Media	Modificar la presentación de la interfaz web, para que este más acorde a la temática que	•

				maneja el software.	
04	correctivo	Grande	Media	Modificar el código para mejorar la calidad de acuerdo a las métricas planteadas y que está incumpliendo	• Localizar las fallas y editar las clases para mejorar la calidad: • Probar que no se altere la funcionalidad debido a los cambios.
05	Perfectivo	Grande	Alta	Reactivar la funcionalidad de geolocalización	• Cambiar versión obsoleta. • Exportar librerías. • Cambiar sintaxis de la funcionalidad. • Probar
06					
07	Adaptativo	Media	Baja	Ampliar la documentación de instalación de la aplicación en sistemas operativos como Linux y dejando soluciones a fallas que se puedan presentar por la configuracon que trae por defecto la plataforma.	• Realizar la descripción de las herramientas necesarias para la instalación junto con su configuración. •

4.5.2. Pruebas de aceptación: Esta prueba se realizó con estudiantes de la facultad de salud en ___ se compuso de las siguientes fases:

Pruebas de Usuarios

Tabla 21 Prueba de Aceptación

Objetivo de la Prueba:	• *Validar los procesos.* • *Validar que se cumplan los requerimientos funcionales establecidos.* • *Validar los requerimientos mínimos para el funcionamiento de la aplicación.* • *Realizar la certificación del correcto funcionamiento de los requerimientos probados.*
Estrategia:	*Elaboración y ejecución de un conjunto de casos de Pruebas(Ver formato anexo), usando datos validos e inválidos para verificar lo siguiente:* • *Los resultados esperados ocurren cuando se usan datos válidos e inválidos.* • *Se despliegan mensajes de error cuando se usan datos inválidos.* • *Cada regla de negocio es propiamente aplicada.* • *Realizar set de pruebas de los requerimientos mínimos para el adecuado funcionamiento de la aplicación*
Herramientas Requeridas:	*Sugerencia:* • *Microsoft Excel 2013* •
Responsables	• *Usuarios Estudiantes de la escuela de salud de la universidad industrial de Santander.* • *Analista de Pruebas: Acompañamiento a las pruebas de usuario*
Criterios de evaluación:	• *El resultado de cada caso de prueba debe ser igual al resultado de salida esperado.* • *Incluir tanto datos de entrada válidos y esperados como no válidos e inesperados.* • *Encontrar fallas al ejecutar los diferentes casos de pruebas.* • *La aplicación cumple con los requerimientos funcionales especificados en la fase de análisis.* • *La aplicación cumple con los requerimientos mínimos para el funcionamiento*

4.5.3. Liberación: Luego de realizadas las pruebas y mantenimiento se procede a la liberación mediante los siguientes pasos:

- Software instalado.
- Documentación completa.

5. CONCLUSIONES

- A pesar del uso de herramientas software en la evaluación del sistema, el proceso requirió de un trabajo manual adicional para la interpretación de los resultados y poder mostrar claramente la información de la estructura del prototipo, de igual manera otro aspecto que supone un esfuerzo y tiempo adicional es que cada desarrollador utiliza una lógica diferente en cada proyecto que realiza, y la comprensión no resulta sencilla.

- La geolocalización es una funcionalidad importante para los sistemas de control y seguimiento de una población, el cumplimiento de las acciones y el servicio a los pacientes por lo que es de vital importancia reactivar esta funcionalidad con la creación de una nueva versión del software bajo el mismo ambiente de desarrollo en una versión actualizada u otro diferente.

- La funcionalidad de geolocalización no pudo ponerse en marcha debido a que el desarrollo se hizo con tecnológica que ya fue declarada como obsoleta y aunque se cambió a una versión nueva del framework de desarrollo, persiste la falencia debido a que la versión influye en las demás funcionalidades

- La estrategia de mantenimiento planteada fue apta ya que se pudo mostrar que es importante someter el software a un constante estudio para evitar errores de cualquier tipo incluyendo depreciación de funcionalidades ofrecidas por los marcos de desarrollo sobre los cuales se diseñó el prototipo.

- La realización de los procesos de pruebas resulta satisfactoria puesto que se vio claramente la necesidad de una evaluación del prototipo, al mostrarse errores provocados por diferentes factores y con la corrección de estos mejorar la calidad del desarrollo.

- La plataforma Java Micro Edition se torna limitada para la clase de dispositivos móviles que se usan actualmente ya que los sistemas operativos con que cuentan los aparatos no incluye la maquina sobre la cual se ejecutan las aplicaciones, por lo que se ve la necesidad de un cambio de tecnología y un nuevo desarrollo para las tecnologías móviles actuales

- La evaluación exhaustiva y la corrección rigurosa de errores de sintaxis, ejecución y lógica resulta en un trabajo delicado de realizar. Por esto se debe ser cuidadoso y ceñirse a la metodología planteada para la corrección y así evitar que surjan nuevos errores y dañar la funcionalidad del software.

- El desarrollo de este proyecto fue útil a sus autores en la formación como ingenieros de sistemas, porque permitió poner en práctica las herramientas provistas durante el transcurso de la carrera como matemáticas, modelado con Dinámica de Sistemas, Mantenimiento de Software, bases de datos y programación, en el desarrollo de herramientas informáticas y habilidades en el planteamiento de soluciones a problemas.
- Las pruebas con usuarios son de gran utilidad al desarrollar y mantener esta clase de software, porque proveen de información sobre el diseño y funcionalidades que tiene o podría tener el sistema.

6. RECOMENDACIONES

- Migrar el programa a otro lenguaje de programación para evitar inconvenientes de funcionalidades inestables por pertenecer a terceros.

- Realizar el proceso de mantenimiento regularmente para estar actualizados con las versiones y funcionalidades que ofrezcan los entornos y frameworks de desarrollo.

- Promover actividades para la difusión y utilización del software.

- Concentrar mayores esfuerzos en el interior del grupo de investigación SIMON por estudiar la posibilidad de creación aplicaciones similares que ayuden a la revisión y control de otras enfermedades regularmente incidentes y riesgosas.

BIBLIOGRAFIA

B. P. Lientz, E. B. (s.f.). *Characteristics of application software maintenance.* Obtenido de ACM Digital Library: http://dl.acm.org/citation.cfm?id=359522

Bennet P. Lientz, E. B. (2011). *Problems in application software maintenance.* Obtenido de ACM Digital Library: http://dl.acm.org/citation.cfm?id=358796

CAGLEY, T. (15 de Mayo de 2014). *Software process and measurement*. Recuperado el 7 de Agosto de 2014, de http://tcagley.wordpress.com/2014/05/15/testing-principles-part-1-this-is-not-pokemon/

GALLESDIC, I. G. (2013). *Revista Antioqueña de las Ciencias Computacionales y la Ingenieria de Sistemas.* Obtenido de La prueba de software como ciencia: http://fundacioniai.org/raccis/v3n1/n4a5.pdf

GUERRA, L. E. (2013). *Diseño de un sistema de información para la alerta temprana en salud, caso: fiebres no diferenciadas en Bucaramanga.* Bucaramanga.

IMBET Augusto, C. J. (2013). *Sistema de informacion,educacion y comunicacion de enfermedades infecciosas que cursan con fiebres agudas.* Bucaramanga.

LEHMAN Meir, R. J. (Diciembre de 2002). *Software Evolution and Software Evolution Processes.* Obtenido de SpringerLink: http://link.springer.com/article/10.1023%2FA%3A1020557525901

MUÑOZ, G. (2012). *Determinación de zonas de riesgo de transmisión de T. cruzi via oral e implementación de un sistema de alerta temprano para chagas agudo en Bucaramanga.* Bucaramanga.

MYERS, G. (2004). *The art of software testing.* John Wiley & Sons.

PRESSMAN, R. (2005). *Ingenieria del software un enfoque practico.* Madrid: Mcgraw-hill.

SOMERVILLE, I. (2005). *Ingenieria del software.* Madrid: pearson.

WEITZENFELD, A. (2005). *Ingenieria del software orientado a objetos con UML,Java e Internet.* Mexico: Thompson.

Wikipedia. (s.f.). *Eclipse_(software)*. Recuperado el 15 de 05 de 2012, de http://es.wikipedia.org/wiki/Eclipse_(software)

ANEXOS

Anexo A

PLAN DE PRUEBAS

INTRODUCCIÓN

El siguiente documento describe los pasos que se realizaran para el proceso de evaluación del sistema de información, educación y comunicación en enfermedades infecciosas que cursan con fiebres agudas (SaludMovil), en el cual quedara plasmada la metodología, estrategias y demás elementos concernientes a la evaluación.

DESCRIPCIÓN DEL SISTEMA

El sistema tiene como función principal dar prioridad a pacientes febriles de una entidad de salud de primer orden, el registro de los mismos, seguimiento y alerta temprana frente a los casos que sean diagnosticados como dengue o Chagas. Se desea que el sistema sea accesible a través de Internet.

Se presentan distintos tipos de usuario:

- ✓ Paciente.
- ✓ Médico del centro de salud.
- ✓ Trabajador en salud comunitario y voluntario (TSCV).
- ✓ Entidades de Salud correspondientes (EDS).
- ✓ Secretaria del centro de salud.
- ✓ Call-Center.
- ✓ Administrador del sistema.

El sistema debe contemplar características de gestión de usuarios tales como:

- ✓ Registro de usuario.
- ✓ Anulación de usuario.
- ✓ Cambio de contraseña de acceso.
- ✓ Recordatorio de contraseña de acceso.
- ✓ Vinculación de usuario a un tipo de usuario.

Todo usuario al ser registrado se le asigna un tipo de usuario, que puede ser cambiado posteriormente por el administrador del sistema.

Solo los usuarios registrados y de tipo paciente pueden registrar temperaturas en el sistema. Este registro se puede hacer por medio de un teléfono móvil con el software del sistema previamente instalado, por medio del usuario Call-Center o ingresando a la página web del sistema.

El servicio de registro de temperatura permite que un paciente ingrese tres (3) veces su temperatura (la temperatura debe ser tomada por el paciente justo antes del envío de la misma), cada toma de temperatura separada con un lapso de tiempo de más o menos cuatro (4) horas. Al finalizar el envío de temperaturas el sistema valida si el paciente

necesita o no una cita médica. Si necesita una cita médica se le asigna un código al paciente para que este asista al centro de salud y sea atendido con prioridad. El sistema le permite visualizar a los usuarios del centro de salud (médico y secretaria) una lista de registros de los códigos que fueron asignados a los pacientes con su respectiva fecha de uso y vinculación del paciente.

En el sistema se registra la información obtenida en la cita médica, (Paciente, Médico y el diagnóstico del médico) y se ofrece la opción de crear un nuevo código de cita prioritaria para una posterior cita médica.

Los usuarios de tipo TSCV son trabajadores de salud comunitarios y voluntarios que conforman grupos (un líder TSCV y varios integrantes TSCV). Cada grupo de TSCV está a cargo de los pacientes de una zona. En consecuencia, el sistema divide el mapa de acción en sectores, sectores que son adicionados o modificados por el administrador, cada sector asignado a un grupo de TSCV y cada paciente relacionado a un grupo o sector.

Al líder TSCV de un grupo se le informa por medio de un correo electrónico si un paciente de su zona es diagnosticado con dengue o Chagas, o si se le asignó un código para una cita prioritaria y el paciente no hizo uso de dicho código para que, en colaboración con el centro de salud, estén atentos de dicho paciente y de la evolución de la enfermedad. De la misma manera se le informa a las entidades de salud correspondientes cuando el paciente es diagnosticado con dengue o Chagas.

Los usuarios de tipo TSCV tienen un espacio en el sistema para registrar actividades que estos realicen en la comunidad, actividades que deben tener como propósito ayudar a la prevención de dengue o Chagas.

Dado que se debe permitir el seguimiento de las enfermedades febriles de los pacientes, el sistema visualiza el historial de las citas médicas de cada paciente, una lista de pacientes de acuerdo a su estado de salud en un periodo de tiempo, un mapa donde se situé la ubicación de los pacientes dependiendo su estado de salud, y gráficas que muestren:

- ✓ El número de citas realizadas semanalmente en un periodo de tiempo, en general y por grupo de TSCV.
- ✓ El número de estados de pacientes (dengue, Chagas, fiebre no diferenciada, trasladado, y demás estados registrados en la cita médica), semanalmente en un periodo de tiempo, en general y por grupo de TSCV.
- ✓ Porcentaje de pacientes que han registrado temperatura.
- ✓ Porcentaje de pacientes que han recibido un código para una cita prioritaria (de todos los que registraron temperatura, cuantos hicieron la tarea completa de enviar las tres temperaturas).
- ✓ Porcentaje de pacientes por grupo que asistieron a la cita médica.
- ✓ El número de temperaturas enviadas por los distintos medios (web, call-center, móvil) en general y por grupo.

Esta información es vista solo por los médicos y secretarias del centro de salud, y por las entidades de salud correspondientes. El TSCV puede obtener solo información de los pacientes que estén vinculados a su grupo de trabajo.

Y por último se maneja un área de notificaciones que le indica a la secretaria del centro de salud y al administrador del sistema:

- Un paciente sin grupo.
- Un grupo sin líder.
- Un grupo sin integrantes.
- Un correo enviado al líder el grupo TSCV de un paciente que no utilizó el código que se le asignó.
- Un paciente tiene un código asignado para el día siguiente (Solo para usuarios call-center).
- Cuando se ocasiona un error al envío de un correo que indica que un paciente no utilizó el código asignado, ya sea porque el paciente no tiene un grupo asignado, o el grupo del paciente no tiene líder o por que el líder no tiene un correo asignado.

PLAN DE PRUEBAS

PROPÓSITO Y OBJETIVOS

Este documento describe el plan para probar las funcionalidades y características del sistema SM[14], con el fin de encontrar la mayor cantidad de problemas, errores y fallas. De esta manera se promoverá el mantenimiento de aplicaciones y el aseguramiento de la calidad de procesos y software.

Este documento está basado sobre los siguientes objetivos:

- ✓ Realizar pruebas técnicas con el fin de encontrar posibles errores o fallos
- ✓ Comprobar que la información existente del proyecto y los componentes de software sean válidos para las pruebas.
- ✓ Describir las estrategias a ser empleadas.
- ✓ Identificar los recursos requeridos y estimar los esfuerzos de las pruebas.
- ✓ Listar los elementos a entregar, producto de las actividades de pruebas.

REQUERIMIENTOS DE PRUEBAS

La lista identifica aquellos elementos (requerimientos funcionales, no funcionales) que han sido identificados como objetivos de las pruebas. Esta lista representa el qué será probado. Los detalles de cada prueba serán determinados posteriormente mientras los casos de prueba sean identificados.

Pruebas de integridad de datos y Bases de Datos

- ✓ Verificar el acceso a la Base de Datos.
- ✓ Verificar el acceso simultáneo en la lectura de registro de las distintas tablas.
- ✓ Verificar el bloqueo realizado durante actualizaciones de registros de las tablas transaccionales
- ✓ Verificar la correcta obtención de data actualizada.

[14] Para el desarrollo del documento se utilizara el acrónimo SM para referenciar el sistema en cuestión Salud Móvil,

Pruebas de instalacion

- ✓ verificar que los archivos de instalación no presenten fallos
- ✓ verificar que la instalación coincida con los pasos propuestos en la documentación

Pruebas del sistema

- ✓ Verificar el CU[15] Login/Logout
- ✓ Verificar el CU Registrar alertas
- ✓ Verificar el CU Registrar temperaturas
- ✓ Verificar el CU Consultar usuarios
- ✓ Verificar el CU Generar cita prioritaria
- ✓ Verificar el CU Buscar paciente
- ✓ Verificar el CU asignar código de cita
- ✓ Verificar el CU ver graficas

Pruebas de desempeño

- ✓ Verificar el tiempo de respuesta para registrar un usuario
- ✓ Verificar el tiempo de respuesta para registrar temperaturas
- ✓ Verificar el tiempo de respuesta para hacer consultas
- ✓ Verificar el tiempo de respuesta para generar citas.

Pruebas de carga

Verificar la respuesta del sistema cuando tiene un determinado número de usuarios accediendo al software.

RECURSOS

- ✓ Código fuente.
- ✓ Documentación
- ✓ Aplicación montada en el servidor

ESTRATEGIA DE PRUEBAS

La estrategia de pruebas presenta el alcance recomendado para la prueba de aplicaciones de software. La sección previa describe qué será probado; ésta describirá cómo será probado.

Las consideraciones principales para la estrategia de pruebas son las técnicas a usarse y los criterios para determinar si la prueba fue completada.

TIPOS DE PRUEBAS

PRUEBAS DE INTEGRIDAD DE DATOS Y BASES DE DATOS

[15] Caso de uso

La base de datos y los procesos de la base de datos se debería probar de manera independiente a la aplicación, para esta prueba solo se requiere el ingreso de datos (datos de usuario, importar, exportar contenidos) por parte de los usuarios y registrar cualquier anomalía en el proceso, y luego se realiza una posterior revisión de la base de datos para comprobar la integridad de los datos.

Objetivo

Asegurar que los métodos de acceso y los procesos funcionen apropiadamente y sin corrupción de datos

Técnicas

Invocar cada método de acceso a la BD, intentando con datos válidos e inválidos. Inspeccionar la base de datos para asegurar que la data ha sido poblada como se esperaba, que todos los eventos ocurran apropiadamente, o revisar la información retornada para asegurar que la información obtenida fue correcta (por las razones correctas).

Criterio de cumplimiento

Todos los métodos de acceso a la base de datos y procesos funcionan como fueron diseñados y sin corrupción de datos.

PRUEBAS DE INSTALACIÓN

Las pruebas de instalación tienen dos propósitos. El primero es asegurar que el sistema puede ser instalado en todas las configuraciones posibles, tales como nuevas instalaciones, actualizaciones, instalaciones completas o personalizadas, y bajo condiciones normales o anormales; estas últimas incluyen insuficiente espacio en disco, falta de privilegios para algunas tareas, etc.

El segundo propósito es verificar que, una vez instalado, el sistema opera correctamente. Esto usualmente implica correr un número significativo de pruebas de Funcionalidad.

Objetivo

Verificar y validar que el sistema se instala apropiadamente en cada cliente, bajo las siguientes condiciones:

- Instalaciones nuevas, nuevas máquinas a las que nunca se les ha instalado el sistema.
- Actualizar máquinas previamente instaladas con el sistema.
- Instalar versiones viejas en máquinas previamente instaladas con el sistema.

Técnicas

Realizar la instalación paso a paso como aparece en la documentación.

Criterios de cumplimiento

El aplicativo se presenta correctamente instalado y no presenta fallas en su ejecución.

PRUEBAS DEL SISTEMA

Las pruebas sobre la aplicación deberían enfocarse en requerimientos que puedan ser asociados directamente a casos de uso (o funciones de negocio), y reglas del negocio. Las metas de estas pruebas son verificar la aceptación, el procesamiento y obtención de data apropiada, así como la apropiada implementación de reglas del negocio. Este tipo de pruebas está basado en las técnicas de caja negra, utilizando para ello la GUI y analizando los resultados.

Objetivo

Asegurar la navegación apropiada en la aplicación; el correcto ingreso de datos, procesamiento y obtención.

Técnicas

Ejecutar cada CU, cada flujo de CU o función, usando data válida e inválida, para verificar:

a) Que los resultados ocurran cuando la data sea válida.

b) Que se muestren apropiados mensajes de error o alerta cuando data inválida sea empleada.

c) Cada regla de negocio es apropiadamente aplicada.

Criterio de cumplimiento

Todas las pruebas planificadas fueron ejecutadas. Todos los defectos de pruebas han sido manejados.

Cada ventana fue verificada exitosamente para comparar si se sigue el estándar o no.

PRUEBAS DE DESEMPEÑO

Realizar las pruebas que miden los tiempos de respuesta, las tasas de transacción y otros requerimientos sensibles al tiempo. La meta de las pruebas de desempeño es verificar y validar que los requerimientos de desempeño han sido alcanzados. Este tipo de pruebas es ejecutado muchas veces, y cada ejecución emplea una carga subrepticia (background load) en el sistema.

Objetivo

Validar el tiempo de respuesta para transacciones diseñadas o funciones de negocio bajo las siguientes condiciones:

a) volumen normal anticipado

b) volumen de caso mal anticipado.

Técnicas

Usar scripts de prueba desarrollados por pruebas de modelo de negocio (pruebas de sistema). Modificar archivos de datos (para incrementar el número de transacciones) o modificar los scripts para incrementar el número de iteraciones en que cada transacción

ocurre. Lo scripts deben correr en una sola máquina (en el mejor de los casos simular un usuario único, una única transacción) y ser repetido en múltiples clientes (virtuales o actuales).

Criterio de cumplimiento

Una transacción / un único usuario. El cumplimiento exitoso de estas pruebas, es cuando no se encuentran fallas en los tiempos esperados o requerido (en cada transacción). Múltiples transacciones / múltiples usuarios. El cumplimiento exitoso de estas pruebas, es cuando no se encuentran fallas en los tiempos aceptables.

PRUEBAS DE CARGA

Las pruebas de carga miden las situaciones en las que el sistema se somete a variaciones en su carga de trabajo para evaluar la habilidad del sistema para continuar funcionando adecuadamente, más allá de la carga de trabajo esperada. Adicionalmente, las pruebas evalúan las características de desempeño (tiempos de respuestas, tasas de transacción y otros problemas sensibles a tiempos).

Objetivo

Verificar el tiempo de respuesta del sistema para transacciones diseñada o casos de negocio bajo condiciones de carga de trabajo variada.

Técnicas

Pruebas de uso desarrolladas para ciclos de prueba de negocio. Modificar archivos de datos incrementando el número de transacciones) o las pruebas para incrementar el número de veces en que una transacción ocurre.

Criterio de cumplimiento

Múltiples transacciones / múltiples usuarios. El cumplimiento exitoso de estas pruebas, es cuando no se encuentran fallas en los tiempos aceptables.

PRUEBA CON USUARIOS

Objetivo

Verificar el comportamiento y la respuesta de los usuarios frente al diseño y funcionalidad de aplicación, y obtener información de la aceptación, de sugerencias o de posibles errores o inconvenientes que los usuarios hayan podido experimentar.

Técnica

La realización de esta actividad se hará por fases

- La primera fase se realizara de la siguiente forma:

Cada usuario tendrá la oportunidad de explorar el software sin ninguna clase de orientación para probar lo intuitivo y usable que es el software, podrá registrarse y saltar de un menú a otro para familiarizarse con el ambiente como complemento la persona

deberá registrar cualquier suceso o interrogante que encuentre durante la actividad en el siguiente formato.

Tabla 22 Formato de incidencias de la prueba con usuarios.

Nombre	Doc. Identidad	Fecha
Suceso	Tipo de usuario	Nombre de actividad

Como actividad principal cada uno registrara temperaturas para tener información en la base de datos que se usara posteriormente.

- La segunda fase se hará de la siguiente manera:

Como primera medida para esta etapa se realizara la descripción del software, explicando claramente su propósito.

Una vez los usuarios estén registrados el administrador concederá a cada uno un tipo de usuario donde se le asignaran una serie de actividades o casos de prueba acorde con su tipo y llenara un formato de preguntas donde plasmara los resultados de los procesos propuestos.

- Como tercera y última fase cada usuario llenara una encuesta donde responderá cuestiones acerca del software independientemente del tipo de usuario que haya manejado y donde también dejara constar sus inconformidades y sugerencias.

Criterios de cumplimiento

Todos los casos de prueba fueron ejecutados y evidenciaron cualquier suceso, correcto o incorrecto y sugerencias e inconformidades.

CRITERIO DE CUMPLIMIENTO PARA EL PLAN

Todas las pruebas han sido ejecutadas y los límites del sistema son alcanzados/excedidos sin que el software falle.

SELECCIÓN DE LAS HERRAMIENTAS DE PRUEBA DE SOFTWARE QUE SE USARAN PARA AUTOMATIZAR EL PROCESO DE EVALUACIÓN.

Sonar: Es una plataforma de código abierto que sirve para gestionar la calidad del código. Sonar es una aplicación web con las siguientes características: está basada en reglas, alertas, rangos, exclusiones y configuración; permite configuración online, dispone de una base de datos y permite combinar métricas en conjunto. Sonar permite extensiones a través de plugins para abarcar nuevos lenguajes de programación, para añadir nuevas reglas o para añadir nuevas métricas.

Selenium: Compuesto por dos herramientas: Selenium IDE y SeleniumWebDriver. La primera, un plugin de Firefox que te genera un entorno de desarrollo y que permite crear casos de prueba para aplicaciones web. La segunda, Selenium WebDriver, ejecuta las pruebas. Este entorno de automatización de pruebas automáticas opera en los principales navegadores (IE, Mozilla, Chrome y Opera). Además, permite pruebas para dispositivos móviles, para iPhone y Android. Utiliza los siguientes lenguajes: Python, Ruby, Java y C#. La licencia es “Apache 2.0 License”.

SDmetrics: Es una herramienta que analiza las propiedades estructurales de los modelos UML, para ello utiliza métricas orientadas a objetos para medir el diseño, la complejidad y la relación entre clases.
SDmetrics evalúa diagramas UML y para ello define un gran número de métricas que agrupa en, métricas de clases, métricas de interfaces, métricas de paquetes, métricas de casos de uso, métricas de máquina de estados, métricas de actividad, métricas de componentes y métricas generales de diagramas.

Ess-Model: Es una herramienta que permite la generación de diagramas UML a partir del código fuente, no es necesario desplegar los proyectos simplemente arrastrar y soltar el archivo fuente e inmediatamente se genera el diagrama. Ess-Model maneja código Delphi y java y puede producir documentación HTML completa con diagramas de clases.

MySQL Workbench 5.2: MySQL Workbench proporciona herramientas visuales para un desarrollador o arquitecto de datos que permite modelar, generar y gestionar bases de datos. Incluye todo lo que un modelador de datos necesita para la creación de modelos ER complejos, ingeniería directa e inversa, y también ofrece características clave para la realización de las tareas difíciles de gestión del cambio y de la documentación que normalmente requieren mucho tiempo y esfuerzo.

ENTREGABLES

Los elementos a entregar después de realizar las actividades correspondientes a la evaluación serán documentos en donde se registraran los resultados de cada prueba con datos obtenidos de las herramientas software y procesos estáticos.

Clasificación de la Severidad de los Defectos

Se definen 3 tipos de problemas según la gravedad de los defectos:

- ✓ **Problemas tipo A - Crítica:** Se define como falla tipo A, a toda falla bloquearte a nivel funcional, tal como inconsistencia en transacciones, caída del sistema, falta de implementación o falla en la funcionalidad principal, degradación grave de performance, incompatibilidad de componentes integrados.
- ✓ **Problemas tipo B - Media:** Se define como falla tipo B a toda falla que afecte puntualmente a una función secundaria pero que no impide que se lleven a cabo normalmente las funciones centrales del sistema. Ejemplo de fallas tipo B son: falla en una operación de búsqueda accesoria.
- ✓ **Problemas tipo C - Baja:** Se define como falla tipo C a toda falla menor que no afecta el funcionamiento general del sistema. Se trata de problemas estéticos, errores de ortografía, formato de impresión, etc.

REPORTE DE ERRORES

Para el reporte de errores se elabora un formulario, el formulario contiene campos para identificar los errores y las acciones que estaba realizando el usuario al momento de presentarse el error, el reporte de errores está asociado a los casos de pruebas, por lo tanto se debe registrar el caso de prueba y el paso donde se presentó.

CASOS DE USO PARA PRUEBAS

Registrar usuarios

Información General	
Identificador de caso de uso:	CU1
Nombre de caso de uso:	Registrar usuario
Descripción Prueba:	Registrar en el sistema un nuevo usuario
Responsable:	
Prerrequisitos	
Descripción del Caso de Prueba	
Caso: La prueba comienza ingresando a la página web y pinchando en la opción de registrarse, enseguida aparecerá un formulario con la información requerida por el sistema se llenan todos los campos y se envía la información, si falta algún campo por llenar el sistema enviara un mensaje que informa que no se ha llenado	

completamente de lo contrario surgirá un mensaje confirmando la satisfactoria creación del usuario.
Instrucciones de Prueba 1. Abrir la página de salud móvil 2. Buscar la pestaña y registrar un usuario nuevo 3. Llenar el formulario y enviar información 4. Esperar la confirmación exitosa del proceso

Escenarios de prueba			**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
Campo	**Valor**	**Tipo escenario**		
Tipo de documento	nulo	real	Imposible registro porque falta completar información del formulario	si
Fecha de nacimiento	18-07-22	real	La fecha no coincide	si

Criterios de Aceptación
El nuevo usuario es creado satisfactoriamente.

Registrar temperaturas

Información General	
Identificador de caso de uso:	CU3
Nombre de caso de uso:	Registrar temperatura
Descripción Prueba:	Ingresar los registros de temperaturas medidos por los usuarios
Responsable:	
Prerrequisitos	

Estar registrado como usuario del sistema				
Descripción de Casos de Prueba				
Caso: para poder ingresar datos se debe estar registrado, luego en el menú usuario se busca la opción y se procede a la siguiente donde se llenaran los campos con la temperatura en grados centígrados, de acuerdo con el valor ingresado el sistema avisara al usuario si se encuentra en un rango de fiebre o si su temperatura es normal es decir dentro de los valores regulares para una persona sana. Solo se registrara las temperaturas que se consideran elevadas es decir por encima de 36 grados Celsius.				
Instrucciones de Prueba • Ingresar como usuario • Registrar temperatura • Verificar la respuesta obtenida				
Escenarios de prueba			**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
Campo	**Valor**	**Tipo escenario**		
Temperatura	48	Correcto	El sistema enviara un mensaje anunciando que la temperatura no está en un rango normal y le pedirá que registre su temperatura luego de un tiempo para ver si después de este tiempo persiste la calidez	si
Temperatura	36		El sistema avisara que no presenta temperatura alta	si
Criterios de Aceptación • Correcto envío de temperatura • Mensajes de alertas				

Búsqueda de paciente

Información General	
Identificador de caso de uso:	CU4

Nombre de caso de uso:	Búsqueda de pacientes
Descripción Prueba:	Ver la información de los pacientes registrados y que presentan o presentaron casos de fiebre
Responsable:	Usuario tipo medico

Prerrequisitos

Estar registrado en el sistema y haber ingresado

Descripción de Casos de Prueba

Caso: Estando registrado como médico buscamos en la barra el menú paciente y escogemos la opción ver pacientes, nos envía a una pantalla donde nos muestra dos campos con los cuales se puede hacer la búsqueda uno es por el nombre y otro por el número del documento si el proceso es correcto debe mostrar toda la información del paciente junto con el historial de citas médicas y la ubicación del paciente en un mapa.

Instrucciones de Prueba

- Ingresar al sistema
- Ir al menú pacientes y seleccionar ver pacientes
- Digitar el dato para la búsqueda
- Revisar información obtenida

Escenarios de prueba			**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
Campo	**Valor**	**Tipo escenario**		
nombre			Nombre completo, documento de identidad categoría	si

Criterios de Aceptación

Se obtiene información detallada de cada paciente

Asignar cita médica prioritaria

Información General		
Identificador de caso de uso:	CU5	
Nombre de caso de uso:	Asignar citas	
Descripción Prueba:	Registrar citas para pacientes con posible caso de fiebre sin diferenciar	
Responsable:	Usuario tipo medico	
Prerrequisitos		
Estar registrado en el sistema y haber ingresado		
Descripción de Casos de Prueba		
Caso: estando registrado como usuario tipo medico se escoge el paciente al cual se le va sugerir la cita prioritaria debido a que presenta un posible caso de enfermedad manifestada por fiebre, el usuario persiste con síntomas y es necesario realizar una consulta presencial en un centro de salud. En el menú paciente aparece en la pantalla de registrar cita médica un formulario con 2 campos obligatorios: médico y estado del paciente, se debe poner una descripción del motivo de la cita aparte de seleccionar una opción de enfermedad u otro caso diferente, además hay una opción para asignar un nuevo código de cita médica para una posterior cita prioritaria donde se debe especificar la fecha de la próxima cita.		
Instrucciones de Prueba • Ir al menú pacientes • Asignar cita • Especificar fecha y confirmar		
Escenarios de prueba	**Respuesta esperada de la aplicación**	**Coincide (Si/No)**

Campo	Valor	Tipo escenario		
Criterios de Aceptación Cita prioritaria asignada correctamente				

Generar alertas

Información General	
Identificador de caso de uso:	CU11
Nombre de caso de uso:	Generar alerta
Descripción Prueba:	Generar alertas para informar sobre posibles casos de enfermedad en la comunidad para tomar medidas, rastrear las causas y controlar una posible propagación.
Responsable:	TSCV, secretaria
Prerrequisitos	
Estar registrado en el sistema y haber ingresado	
Descripción de Casos de Prueba	
Caso: después de conocer los resultados de las citas médicas y de la generación de nuevas citas se puede determinar si existe un caso de enfermedad confirmado de acuerdo al diagnóstico emitido por el medico	
Instrucciones de Prueba <Pasos que se deben seguir para poder realizar correctamente la prueba>	

Escenarios de prueba			**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
Campo	**Valor**	**Tipo escenario**		

Criterios de Aceptación Todos los tipos de usuarios involucrados en la transacción de información quedan avisados				

Visualizar de gráficas

Información General				
Identificador de caso de uso:	CU6			
Nombre de caso de uso:	Visualización de graficas estadísticas			
Descripción Prueba:	Consultar las gráficas que ofrece el sistema para ver el uso del sistema por parte del usuario			
Responsable:				
Prerrequisitos				
Estar registrado en el sistema y haber ingresado				
Descripción de Casos de Prueba				
Caso: En el menú acciones se encuentra la opción de gráficas, nos dirigimos a esta función y nos aparece una página donde nos permite seleccionar la clase de información que queremos revisar y que está representada por una gráfica estadística, para generar la gráfica una vez que hayamos escogido la clase de información que queremos conocer simplemente definimos una fecha de comienzo y otra final.				
Instrucciones de Prueba • Ingresar como usuario • Ir al menú acciones • Seleccionar la opción graficas				
Escenarios de prueba			**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
Campo	**Valor**	**Tipo escenario**		

Fecha		correcto	Genera la gráfica, sin ningún error y con la información correcta	si
fecha		incorrecta	Muestra mensajes del porque no se generó la grafica	si
Criterios de Aceptación Información correcta del uso del sistema				

Anexo B:

EVALUACION DE SOFTWARE

Realizar pruebas de caja blanca es importante para conocer la arquitectura de la aplicación y por consiguiente el funcionamiento de esta, también es útil para ver posibles errores que afecten al código en cuanto a su calidad ya que esta es importante para medir los aciertos en cuanto al proceso de mantenimiento se trata.

Para esta labor se usaron las herramientas sonar, SDmetrics y Ess-model, las dos primeras realizan evaluaciones similares pero se usaron todas para complementar los resultados obtenidos y de esta manera tener un informe más completo y detallado.

Herramientas usadas para el proceso de pruebas de caja blanca

Qué son las herramientas case ?

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.

Herramientas software utilizadas:

Sonar

SonarQube es una plataforma para evaluar código fuente. Es software libre y usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD o FindBugs para obtener métricas que pueden ayudar a mejorar la calidad del código de nuestros programas. Además, tiene soporte para más diferentes lenguajes de programación entre los que se encuentran Java, C#, C / C++ y Python entre otros.

SonarQube ofrece un seguimiento a lo largo del desarrollo y/o mantenimiento de un programa informático para apoyar a la mejora continua. Sin embargo, también puede ser utilizado para realizar análisis aislados y obtener informes acerca de nuestros proyectos.

Ess-Model

Con Ess-Model, usted puede ver los diagramas de clases a partir del código. Ess-Model maneja Delphi / Kylix y los archivos fuente de Java, además produce una documentación completa en HTML con diagramas de clase.

Características

Elemento	Descripción
General	- Generación automática de diagramas de clases UML estándar de la fuente
Entrada	- Java (.java) archivos de código fuente. - Java. (.class) archivos binarios.
Navegación	- Explorador de árbol para facilitar la navegación del modelo de objetos. - Visión en miniatura del diagrama.
Diagramas	- Diagramas de clases estáticas. - Diseño automático inteligente. - Visibilidad de filtrado. - Se indican las asociaciones y la herencia.
Instalación	- Archivo exe 700KB, no necesita instalación.
Otros	- Comando completo de control de línea. - Delphi IDE integración. - Interfaz de usuario fácil de arrastrar y soltar. - Increíblemente rápido.
Salida	- Vista previa o generar documentación HTML con diagramas. - Modelo de exportación de datos en formato XMI. - Guardar diagrama como mapa de bits. - Copia de todo o parte del diagrama en el portapapeles.

Procedimiento:

- El proceso para obtener los diagramas de clases es el siguiente:

Abrimos el programa ESS-MODEL, Vamos a File → Open Folder, buscamos la ruta donde se encuentran los archivos de código fuente del programa, al que se le desea generar la documentación UML.
Inmediatamente el software identifica los paquetes y las clases, reconociendo los atributos, métodos y asociaciones que los componen.

- Para generar las imágenes:

Vamos a Diagram → Save as picture, seleccionamos la ruta destino para guardar el diagrama que estemos visualizando en formato .png.

Para generar la documentación:

Vamos a File→ generate documentation, escogemos la ruta destino donde se guardará la correspondiente documentacion del software comprendida en archivos .html e imagenes .png

- Para exportar .XMI-file:

Vamos a File→ Export model to XMI-File, escogemos el directorio destino y con esto tendremos nuestro modelo .xmi, listo para evaluarlo con un software de métricas.

SD-METRICS

Es una herramienta de métricas de diseño de software para UML. SDMetrics analiza las propiedades estructurales de los modelos UML. Utiliza medidas orientadas a objetos de diseño, acoplamiento, tamaño y complejidad para:

- Establecer puntos de referencia de calidad para identificar los posibles problemas de diseño desde el principio.
- Predecir las cualidades relevantes del sistema, como la propensión a fallos o mantenimiento, para enfocar mejor sus esfuerzos de revisión y las pruebas.
- Aumentar la calidad y efectividad del sistema de aseguramiento de calidad, encontrar más fallos con anterioridad y ahorrar costes de desarrollo.
- Perfeccionar LOC o el esfuerzo en estimaciones para la implementación y las pruebas.

Características

SDMetrics incluye un conjunto de medidas orientadas a objetos de diseño (OO), que abarca las propiedades estructurales de los elementos de diseño de todos los tipos de diagramas UML1.x y UML2.x

Mide todos los atributos de diseño de importación el tamaño, el acoplamiento, la complejidad y más, en todos los niveles de detalle, desde el modelo del subsistema, a nivel de paquete hasta las clases y las operaciones.

Verificación de la regla de diseño

Las normas de diseño y la heurística inspeccionan automáticamente el diseño UML en términos de integridad, coherencia, exactitud, problemas en el estilo de diseño como los ciclos de dependencia, y mucho más.

Retroalimentación temprana de calidad

Cuanto más tarde se detecta un fallo en el proceso de desarrollo, más costoso es de arreglar. SDMetrics encuentra problemas en la etapa de diseño, antes de que estén comprometidos con el código fuente.

Conjunto de medidas y normas de diseño extensibles

No se limita al conjunto de medidas y normas incorporadas. SDMetrics tiene un mecanismo flexible y potente para definir y calcular las nuevas reglas y medidas de su propio diseño, adaptado a las prácticas de su desarrollo.

Comparación de diseños

Calcula deltas de métricas de tamaño para cuantificar el crecimiento en tamaño entre las dos versiones de un proyecto, identificar las partes del diseño del sistema que han experimentado grandes cambios, o evaluar alternativas de solución a un problema de diseño.

Interoperabilidad con herramientas UML

SDMetrics está diseñado para trabajar con todas las herramientas de modelado UML con XMI exportación.

- SDMetrics soporta todas las versiones de XMI 1.0, 1.1, 1.2, 2.0 y 2.1 actualmente en uso.
- Importación XMI flexible, configurable para hacer frente a las extensiones propietarias del metamodelo UML, y herramientas que se apartan de las normas XMI.
- Use SDMetrics con herramientas de ingeniería inversa que producen archivos XMI de C + +, Java, Delphi, o el código fuente de Smalltalk,. NET, etc, para realizar mediciones en el diseño de esas fuentes.

Datos de exportación

Los datos de las mediciones de diseño resultan más eficaces de utilizar cuando se someten a procedimientos de análisis estadísticos de gran alcance. SDMetrics exporta los datos de las mediciones y estadísticas descriptivas en diversos formatos (tablas de texto separadas por tabuladores, HTML, OpenOffice.org Calc, y XML para Microsoft Excel XP) para la fácil importación en las aplicaciones de oficina, programa de hojas de cálculo y paquetes estadísticos.

GUI interactiva

Con la facilidad de uso de la interfaz gráfica de usuario de SDMetrics, se pueden explorar interactivamente los datos de medición, identificar los valores anómalos, explorar histogramas y gráficos Kiviat.

Interfaz de línea de comandos

La medición y las características de exportación de datos también son accesibles a través de una interfaz de línea de comandos. La ejecución del análisis automatizado permite integrar SDMetrics en su entorno de desarrollo.

Velocidad

SDMetrics es rápido. Un archivo XMI de 120MB con cientos de miles de elementos de diseño se procesa en cuestión de segundos.

Plataformas compatibles

SDMetrics se ejecuta en todas las plataformas que soportan la versión 6 de Java Runtime Environment (Windows XP/Vista/7, Unix y Linux).

Procedimiento

Abrir uno a uno los archivos XMI obtenidos a partir de la herramienta ESS-Models y ejecutar el comando "*calcule metrics*" en la herramienta SDMetrics

Copiar manualmente cada unos de los valores de métricas que se presentan de acuerdo a el atributo que haya tenido una incidencia.

Algunas métricas no se presentan por la herramienta por lo que es necesario realizar el proceso manualmente, para esta tarea se despliega el árbol de diagramas de la aplicación con la herramienta Ess-model y se recorre manualmente para buscar la información de la métrica presente.

Métricas empleadas para la evaluación de software

Métricas proporcionadas por SDmetrics

A continuación se mencionan solo las métricas que formaron parte de la evaluación software, para la selección de dichas métricas se le dio relevancia a aquellas que nos permitieran identificar problemas en la estructura de nuestros productos software.

Indicadores de clase		
Métrica	**Categoría**	**Descripción**
NumAttr	Tamaño	El número de atributos de la clase.
NumOps	Tamaño	El número de operaciones en una clase.
NumPubOps	Tamaño	El número de operaciones en una clase pública.
Setters	Tamaño	El número de operaciones con un nombre que comienza con 'set'.
Getters	Tamaño	El número de operaciones con un nombre que comienza con 'get', 'es', o 'tiene'.
IFImpl	Herencia	El número de interfaces de la clase implementa.
NOC	Herencia	El número de niños de la clase (la generalización de UML).
NumDesc	Herencia	El número de descendientes de la clase (la generalización de UML).
NumAnc	Herencia	El número de los antepasados de la clase.
DIT	Herencia	La profundidad de la clase en la jerarquía de herencia.
OpsInh	Herencia	El número de operaciones heredadas.
AttrInh	Herencia	El número de atributos heredados.
Dep_Out	De acoplamiento (de importación)	El número de elementos sobre los que depende esta clase.
Dep_In	De acoplamiento (de exportación)	El número de elementos que dependen de esta clase.
NumAssEl_ssc	Enganche	El número de elementos asociados en el mismo ámbito (namespace) como la clase.
NumAssEl_sb	Enganche	El número de elementos asociados en el mismo ámbito que la rama de la clase.

NumAssEl_nsb	Enganche	El número de elementos asociados no en el poder mismo alcance que el de clase.
EC_Attr	De acoplamiento (de exportación)	El número de veces que la clase se utiliza como externamente tipo de atributo.
IC_Attr	De acoplamiento (de importación)	El número de atributos en la clase con otra clase o interfaz como su tipo.
EC_Par	De acoplamiento (de exportación)	El número de veces que la clase se utiliza como externamente tipo de parámetro.
IC_Par	De acoplamiento (de importación)	El número de parámetros en la clase con otra clase o interfaz como su tipo.
MsgSent	De acoplamiento (de importación)	El número de mensajes enviados.
MsgRecv	De acoplamiento (de exportación)	El número de mensajes recibidos.
MsgSelf	Complejidad	El número de mensajes enviados a las instancias de la misma clase.
Diags	Diagrama	El número de veces que aparece la clase en un diagrama.

Métricas de paquetes		
Métrica	**Categoría**	**Descripción**
NumCls	Tamaño	El número de clases en el paquete.
NumOpsCls	Tamaño	El número de operaciones en las clases del paquete.
NumInterf	Tamaño	El número de interfaces en el paquete.
R	Complejidad	El número de relaciones entre clases e interfaces en el paquete.
Dep_Out	De acoplamiento (de importación)	El número de dependencias donde UML el paquete es el cliente.
Dep_In	De acoplamiento (de exportación)	El número de dependencias donde UML el paquete es el proveedor.
DepPack	De acoplamiento (de importación)	El número de paquetes en los que las clases e interfaces de este paquete depende.
Diags	Diagrama	El número de veces que el paquete aparece en el diagrama.

MÉTRICAS CALCULADAS MANUALMENTE

Métricas de requisitos		
Métrica	**Categoría**	**Descripción**
NOAS	Especificación	Consiste en el número de pasos que los actores ejecutan dentro de la especificación del caso de uso.
NOSS	Especificación	Consiste en el número de pasos que el sistema ejecuta dentro de la especificación de un caso de uso.
NOUS	Especificación	Consiste en el número de pasos que cubre la especificación de un caso de uso, donde se inicia la ejecución de otros casos de uso (incluidos ó extendidos).
NOS	Especificación	Consiste en el número de pasos que cubre la especificación de un caso de uso. Su fórmula es: NOAS+ NOSS+ NOUS. Su valor normal debe estar en el rango de: 3-9

NOCS	Especificación	Consiste en el número de pasos condicionales que cubre la especificación de un caso de uso.
NOE	Especificación	Consiste en el número de excepciones que cubre la especificación de un caso de uso.
RAAS	Especificación	Consiste en la tasa de acciones del actor en un caso de uso. Su fórmula es: NOAS/NOS. Su valor normal debe estar en el rango de: 30%-70%
RUCS	Especificación	Consiste en la tasa de acciones de otros casos de uso en el caso de uso. Su fórmula es: NOUS/NOS. Su valor normal debe estar en el rango de: 0%-25%
CC	Especificación	Consiste en la complejidad ciclomática de un caso de uso. Su fórmula es: NOCS + NOE+1. Su valor normal debe estar en el rango de: 1-4.
NAU	Diagrama caso de uso y secuencia	Número de actores asociados a un caso de uso.
NMU	Diagrama caso de uso y secuencia	Número de mensajes que intervienen para satisfacer la implementación del caso de uso.
NSCU	Diagrama caso de uso y secuencia	Valora el número de clases cuyos objetos intervienen en algún escenario del caso de uso.
NIE	Diagrama caso de uso y secuencia	Consiste en el número de veces que un caso de uso es incluido o extiende a otros casos de uso.

REGLAS DE DISEÑO PROPORCIONADAS POR SDMETRICS

SDMetrics comprueba el cumplimiento de las reglas de diseño UML siguientes. Puede modificar estas reglas de diseño existentes y añadir nuevas reglas propias, adaptadas a sus prácticas de diseño UML y las normas de desarrollo.

A continuación se mencionan las reglas de diseño que mostraron problemas durante la evaluación del software.

Las reglas de clase		
Regla	**Categoría**	**Descripción**
Clase Dios	Estilo	La clase tiene más de 60 atributos y operaciones.
Palabra clave	Nombramiento	Nombre de la clase es una aplicación Java o C + + palabra clave.
Sin nombre	Completitud	Clase no tiene nombre.
Sin usar	Completitud	La clase no se utiliza en cualquier lugar.

Tipo de datos de normas		
Regla	**Categoría**	**Descripción**

Palabra clave	Nombramiento	Nombre del tipo de datos es una aplicación Java o C + + palabra clave, buscar otro nombre para él.
Sin nombre	Completitud	El tipo de datos no tiene nombre.

Las normas de propiedad		
Regla	**Categoría**	**Descripción**
Palabra clave	Nombramiento	Nombre del atributo es una aplicación Java o C + + palabra clave.
Atributo Público	Estilo	No constante atributo es público.
Sin nombre	Completitud	El atributo no tiene nombre.

Reglas de operación		
Regla	**Categoría**	**Descripción**
Palabra clave	Nombramiento	Nombre de la operación es una aplicación Java o C + + palabra clave.
Sin nombre	Completitud	Operación no tiene nombre.

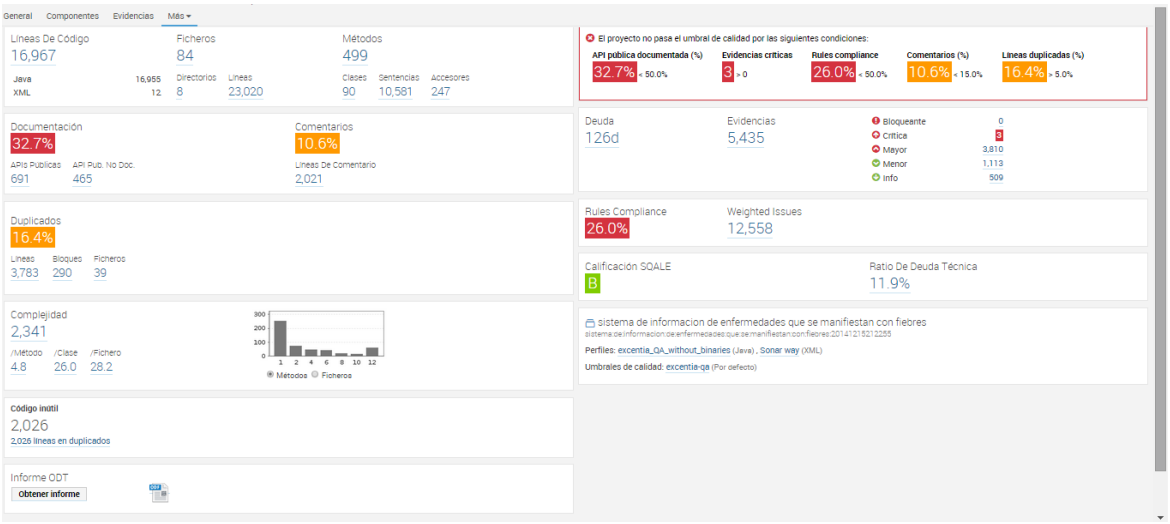
Normas de parámetros		
Regla	**Categoría**	**Descripción**
Palabra clave	Nombramiento	Nombre del parámetro es una aplicación Java o C + + palabra clave.
Sin nombre	Completitud	Parámetro no tiene nombre.

RESULTADOS DE LA EVALUACIÓN SOFTWARE

Los siguientes resultados corresponden a la evaluación de 2 productos software (Salud móvil-SIDEF y SaludMovilME), para la realización de esta se emplearon las herramientas Sonar, EssModel, SDMETRICS y Excel.

Evaluación a la aplicación web

Para esta evaluación se usó el software SonarQube. A continuación se muestra la interfaz y los resultados.



Evaluación SaludMovil

Saludmovil web es un prototipo de software desarrollado con el entorno de desarrollo Eclipse y el framework vaadin en lenguaje java, posee 90 clases de la herramienta y 80 propias del lenguaje, 499 métodos y 247 atributos. En esta aplicación se lograron identificar problemas de complejidad, completitud, duplicidad cuyas principales evidencias son:

- Cantidad de comentarios
- Bloques de código duplicado
- Líneas de código duplicadas
- Clases enormes
- Clases no usadas
- Errores de sintaxis que generan advertencias, mensajes de tipo INFO y fallas críticas de acuerdo a normas o buenas prácticas de programación, pero que no afectan la ejecución del sistema.

Las evidencias encontradas en la evaluación afectan el sistema en diferentes campos, como el uso inadecuado de memoria por parte de sentencias inútiles o mal declaradas perjudicando la eficiencia del aplicativo.

Otra evidencia encontrada incurre en la legibilidad del código afectando la mantenibilidad y el entendimiento del mismo, la aplicación de sangría en las líneas de código influye en la manera como el compilador revisa los bloques y como lo entiende para evitar resultados triviales.