Universidad de las Ciencias Informáticas

Facultad 5

Título: Herramienta para la gestión de pruebas no funcionales para proyectos del CEDIN.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autora: Yensy Alonso Romero

Tutora: Ing. Marelis Virgen Pérez García

Co-tutores: Ing. Dayanis Castellanos Rodríguez

Ing. Dagmar Mir Leyva

La Habana, Junio 2014

"Año 56 de la Revolución"

"La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia."

John Ruskin

Declaración de autoría

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Yensy Alonso Romero

Autora

_____

Ing. Marelis Virgen Pérez García

Tutora

_____ _____

Ing. Dayanis Castellanos Rodríguez Ing. Dagmar Mir Leyva

Co-tutora **Co-tutor**

Datos de contactos

Tutora: Ing. Marelis Virgen Pérez García.

Síntesis del Tutor:

- Especialidad de graduación: Ingeniero en Ciencias Informáticas en la UCI.
  - Categoría docente: Profesor Instructor.
  - Años de experiencia en el tema: 4 años.
  - Años de graduado: 5 años.

- E-mail: mvperez@uci.cu

Co-Tutora: Ing. Dayanis Castellanos Rodríguez.

Síntesis del Co-Tutora:

- Especialidad de graduación: Ingeniero en Ciencias Informáticas en la UCI.
  - Categoría docente: Recién graduado en Adiestramiento.
  - Años de experiencia en el tema: 3 años.

- E-mail: dcastellanosr@uci.cu

Co-Tutor: Ing. Dagmar Mir Leyva.

Síntesis del Co-Tutor:

- Especialidad de graduación: Ingeniero en Ciencias Informáticas en la UCI.
  - Categoría docente: Recién graduado en Adiestramiento.

- E-mail: dmir@uci.cu

Agradecimientos

A dios por permitirme estar hoy aquí frente a todos ustedes.

A mi madre que a pesar de no estar aquí en este momento tan importante en mi vida está más presente que nunca porque gracias a ella, a su apoyo incondicional, a su amor y cariño, a su entrega, a su fuerza interior, tuve las agallas para empezar una vida sin su presencia, a sobreponerme a circunstancias que hacen derrumbar a uno, a no solo pensar en hoy sino también en el mañana, a ser una madre ejemplar para mi hijo, en fin le agradezco con toda mi alma y mi corazón por ser la madre que todo hijo desea tener, por siempre estar a mi lado, por saber corregirme cuando las cosas no estaban bien, por todo eso y mucho más la amaré con todas mis fuerzas y estaré siempre agradecida.

A mi bebe que es el regalo más lindo que me ha dado la vida y que gracias a él tuve la fuerza y la valentía de salir adelante en el momento más triste de mi existencia.

A mi papá por apoyarme cuando más lo necesité a pesar de las diferencias que hubo entre nosotros, al igual que a su esposa y familia por tratarme con mucho cariño y dedicación.

A mi hermano y mi sobrina por formar una parte de mi vida.

A mis tías y primas por apoyarme y darme cariño cuando más lo necesitaba en especial a Dulce y Yuya que han sido mis segundas madres.

A mis amistades por permitirme vivir con ellos momentos tan lindos e inolvidables a lo largo de la carrera en especial a Aylin y Dagmar por ser más que mis amigos ser como hermanos para mí.

A mis tutores, a los miembros del tribunal y oponente por tener paciencia conmigo y apoyarme.

A Yosbel por ser un buen padre.

A la familia de mi hijo, Martha, Guillermina, Mari y Marilú las cuales como ven dejo de últimas porque gracias a ellas puede lograr este sueño, a pesar de muchas dificultades siempre me ayudaron para poder estar hoy aquí y por querer y cuidar tanto a mi titi, por lo que siempre estaré agradecida.

A todos muchas gracias.

Yensy.

Dedicatoria

A mi madre con todo el amor y cariño que solo ella supo darme.

A mi niño bello que es hoy mi razón de ser y por quien vivo orgullosa y deseosa de luchar.

A mi sobrina Giselle que le sirva de ejemplo para que en un futuro sea una profesional.

Resumen

El Centro de Informática Industrial (CEDIN) es uno de los centros de desarrollo de la Universidad de las Ciencias Informáticas (UCI), dicho centro cuenta con el grupo de trabajo de Calidad y Pruebas, que tiene como principios básicos el soporte y puesta a punto de los productos que se desarrollan en el centro. Una de las funciones de este grupo es la ejecución de pruebas de software, donde es muy común encontrar durante su desarrollo, carencias en la validación de los requerimientos no funcionales de rendimiento, seguridad, fiabilidad y usabilidad por solo mencionar algunos, debido a la inexistencia de un procedimiento de trabajo para ejecutarlos. Es por ello que el objetivo principal de la presente investigación es desarrollar una herramienta que facilite el proceso de pruebas no funcionales en los proyectos del CEDIN. Como resultado se obtuvo un procedimiento que define detalladamente cómo realizar las pruebas no funcionales de los productos del centro y una herramienta para la gestión de las mismas, proporcionándole así a los productos una mejor calidad en cuanto a requisitos no funcionales. Una vez confeccionada la propuesta se realizaron pruebas unitarias y pruebas de aceptación, logrando un 100% de pruebas correctas.

Palabras clave: procedimiento automatizado, pruebas no funcionales, requisitos no funcionales.

Índice de contenidos

Índice de Figuras

Índice de Tablas

Introducción

Los nuevos avances científico-técnicos en las ramas de la comunicación, la electrónica y las computadoras han revolucionado el mundo en todos los aspectos. El poderoso auge de las tecnologías de la información y las comunicaciones (TIC) ha cambiado los paradigmas y estrategias reconocidas y establecidas por muchos años como válidas. Dentro de las TIC, la industria del software alcanza una posición relevante, por su característica de controlar o hacer accesible, en la mayoría de los casos, los adelantos electrónicos.

Actualmente el desarrollo de software, por sus múltiples aplicaciones en los negocios, se ha convertido en uno de los elementos fundamentales de la informática y las comunicaciones, trayendo consigo que las industrias, en busca de mejorar la productividad y la calidad de sus servicios, automaticen sus procesos. Debido a este crecimiento continuo del desarrollo de software, se hace vital que el producto se desarrolle con la mayor calidad posible.

Los clientes necesitan productos con características que satisfagan sus necesidades y expectativas expresadas en la especificación del producto, estas son denominadas requisitos del cliente. Para validar que el producto desarrollado cumple con estos es necesario someterlo a un proceso de pruebas de software que se encarga de su verificación, donde se confirma mediante el aporte de evidencia objetiva que se han cumplido los requerimientos especificados en el diseño del producto.

El Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas (UCI) es uno de los centros de desarrollo, que se centra fundamentalmente en el desarrollo de productos y brindar servicios informáticos, dentro de las áreas de automatización industrial. Una de las metas trazadas por el CEDIN consiste en elevar la calidad del software que se produce, avalando y certificando los productos mediante la implantación de mecanismos que permitan mejorar la calidad de los procesos de desarrollo de software. Es por ello que dicho centro cuenta con el grupo de trabajo de Calidad y Pruebas, que tiene como principios básicos el soporte y puesta a punto de los productos que se desarrollan en el centro, además se encarga de velar porque las soluciones realizadas en el centro, cumplan con los requerimientos definidos por el cliente. Una de las funciones de este grupo es la ejecución de pruebas de software, realizadas por probadores que tienen como resultado las no conformidades identificadas al producto. A pesar de realizar este proceso de pruebas, es muy común

encontrar durante el desarrollo carencias en la validación de los requerimientos no funcionales en el producto final, debido a la inexistencia de un procedimiento de trabajo para ejecutarlas. Durante la ejecución de las pruebas se detectan grandes problemas de rendimiento, seguridad, fiabilidad y usabilidad por solo mencionar algunos.

Por lo anterior se define el siguiente **problema científico:** ¿Cómo mejorar el proceso de pruebas no funcionales para los proyectos del CEDIN?

El **objeto de estudio** se enmarca en el desarrollo de pruebas de software, definiéndose como **campo de acción** el proceso para realizar pruebas no funcionales a todos los proyectos del CEDIN.

Para solucionar el problema planteado se define como **objetivo general:** Desarrollar una herramienta que facilite el proceso de pruebas no funcionales en los proyectos del CEDIN.

Para dar cumplimiento al objetivo propuesto, se definen las siguientes **tareas de investigación:**

1. Elaboración del marco teórico de la investigación.
2. Adaptación y ampliación del procedimiento de pruebas no funcionales de los laboratorios virtuales a todos los proyectos del centro.
3. Descripción de un procedimiento para la gestión de pruebas no funcionales a todos los proyectos del CEDIN.
4. Análisis y diseño de la herramienta para la gestión de las pruebas no funcionales.
5. Implementación de la herramienta para la gestión de pruebas no funcionales del software.
6. Validación de la herramienta a través de la realización de las pruebas unitarias y pruebas de aceptación.

Métodos teóricos:

Análisis-síntesis: este método permitió analizar bibliografía sobre los conceptos empleados en la realización del procedimiento para pruebas a requisitos no funcionales, analizando todos los documentos para la extracción de los elementos más importantes sobre el tema en cuestión.

Histórico–lógico: se utilizó para realizar un análisis con mayor profundidad de los antecedentes y las tendencias actuales referidas a los procedimientos de pruebas no

funcionales de software y la evolución de herramientas para la realización de este tipo de pruebas.

Métodos Empíricos:

Revisión de la documentación: Este método permitió seleccionar la información necesaria en la investigación a partir del estudio de documentos y diferentes bibliografías.

Capítulo 1. Fundamentación Teórica

1.1. Introducción

En el presente capítulo se muestra un estudio general de lo que es calidad de software, así como los estándares o normas para evaluar las características de calidad que tributan al cumplimiento de los requisitos no funcionales, se analizan los conceptos y términos relacionados con las pruebas de software, los requisitos no funcionales y sus clasificaciones, haciendo énfasis en las herramientas y tecnologías existentes para la construcción de la aplicación, además se hace un análisis y selección de las metodologías de desarrollo del software.

1.2. Calidad de Software

Dada que la competencia cada día es más fuerte, es necesario que las empresas se preocupen en dar un mejor producto. La complejidad de los problemas que hoy buscan una solución en el software ha aumentado de manera considerable. Este crecimiento ha sobrepasado el aumento en la habilidad de desarrollar y mantener el software por parte de las organizaciones encargadas.

Dentro del contexto de Ingeniería de Software, se tomará la definición de calidad en el software propuesta por la organización internacional de estándares (ISO/IEC DEC 9126): “La totalidad de características de un producto de software que tienen como habilidad, satisfacer necesidades explícitas o implícitas”. Se puede decir también que el software tiene calidad si cumple o excede las expectativas del usuario en cuanto a:

1. Funcionalidad (que tenga un propósito).
2. Ejecución (que sea práctico).
3. Confiabilidad (que haga lo que debe).
4. Disponibilidad (que funcione bajo cualquier circunstancia).
5. Apoyo, a un costo menor o igual al que el usuario está dispuesto a pagar.

Resumiendo se puede decir, que la calidad de software se refiere a: “Los factores de un producto de software que contribuyen a la satisfacción completa y total de las necesidades de un usuario u organización”. (1)

1.3. Normas y estándares existentes para evaluar la calidad de software

Internacionalmente desde hace algún tiempo se han creado y acordado estándares que proporcionan métricas que permiten validar si un producto de software cumple con dichos enfoques de calidad. Entre estos estándares se destacan el ISO 9126 y el ISO

14598, los cuales se completaron alrededor del año 2004 y en la actualidad están siendo supervisados por el proyecto SQuaRE (*Security Quality Requirements Engineering*) ISO 25000:2005, en el cual se planea reemplazarlos pero conservar sus mismos conceptos. A continuación se describen sus principales características:

Norma ISO/IEC 9126: Es un estándar internacional para la evaluación de la calidad del software como producto final. Este provee un *framework* para la evaluación de la calidad y se divide en cuatro partes que tratan los temas siguientes: modelo de la calidad (9126-1), métricas externas (9126-2), métricas internas (9126-3) y métricas de calidad en uso (9126-4). Estas partes de la norma incluyen características (con sub-características), con métricas que permiten establecer la calidad del producto final desde sus diferentes perspectivas. (2)

Norma ISO/IEC 14598: Este estándar permite tener un conjunto de elementos de evaluación y métricas que sirven para saber a través de un proceso específico de desarrollo de software, como se ha elaborado. Además en sus diferentes etapas, establece un marco de trabajo para evaluar la calidad de los productos de software proporcionando, además características con métricas y requisitos para los procesos de evaluación de los mismos. (2)

Norma ISO/IEC 25000: En lo que se refiere a calidad del producto la norma ISO/IEC 25000 proporciona una guía para el uso de las nuevas series de estándares internacionales, llamados Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE). Constituyen una serie de normas basadas en la ISO 9126 y en la ISO 14598 (Evaluación del Software), y su objetivo principal es guiar el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad. Establece criterios para la especificación de requisitos de calidad de productos software, sus métricas y su evaluación. (3)

Norma ISO/IEC 17799:2007: En Cuba ha sido elaborada una norma cubana NC ISO/IEC 17799:2007 (Publicada por la ISO en 2005), elaborada por el Comité Técnico de Normalización NC/CTN 18 de Tecnología de la Información.

Esta norma es una adopción idéntica por el método de traducción de la Norma Internacional ISO/IEC17799:2005. ISO/IEC 17799 es un estándar para la seguridad de la información que proporciona recomendaciones de las mejores prácticas en la gestión de la seguridad de la misma a todos los interesados y responsables en iniciar, implantar o mantener sistemas de gestión de la seguridad de la información (4).

1.4. Pruebas de software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que hacen a la excelencia del desempeño de un programa, así como también la mejor publicidad que una empresa dedicada a la producción de software pueda tener. “Las técnicas para encontrar problemas en un programa son extensamente variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad”. (5)

Probar es una práctica habitual de todo proceso productivo, que básicamente consiste en verificar que un producto posee las características deseadas por los clientes. Prácticamente a todos los productos a lo largo del proceso productivo se les realizan comprobaciones que hacen que este sea el adecuado.

En el caso del software ocurre lo mismo. Al igual que sucede en un proceso productivo normal, cada una de estas partes deben ser probadas. La naturaleza y el tipo de prueba a realizar con el software, varía a medida que el desarrollo avanza. Se puede decir que el desarrollo de software es un proceso productivo como otro cualquiera, con sus características y particularidades. Dentro de cada una de las etapas de desarrollo de un software, las pruebas son fundamentales, ya que a partir de ellas es posible controlar que los productos cumplan requisitos mínimos operativos, además de garantizar la calidad de los mismos.

“Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados, y se realiza una evaluación de algún aspecto del sistema o componente.”(6)

La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Además, esta etapa implica:

- Verificar la interacción de componentes.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. Es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta ese momento. (6)

1.5. Requisitos no funcionales

Los requisitos no funcionales (RNF) son primordiales para el éxito de los sistemas. Si bien los requisitos no funcionales suelen ser difíciles de definir y cuantificar con objetividad, es importante identificarlos, al menos en términos generales, para que puedan estudiarse. Es muy difícil establecer una separación entre requisitos funcionales y no funcionales, ya que la decisión de si es uno u otro puede venir por el nivel de detalle del documento de requisitos. Además, los requisitos no funcionales son difíciles de expresar, y mucho más de ser recogidos en un documento de requisitos utilizando las mismas técnicas que para los requisitos funcionales. Hay que tener en cuenta que normalmente los errores debido a requisitos no funcionales son los más difíciles y caros de resolver. Los RNF deben establecer restricciones en el producto que está siendo desarrollado en el proceso de desarrollo y en restricciones específicas que el producto pueda tener. Los requisitos no funcionales son difíciles de verificar/testear, y por ello son evaluados subjetivamente. (7)

1.5.1. Generalidades

- Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.
- Se conocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el software.
- No son parte de la razón fundamental del producto pero sí son necesarios para hacer funcionar el producto de la manera deseada.
- No modifican la funcionalidad del producto y sí añaden funcionalidad al producto.
- Describen la experiencia del usuario cuando trabaja con el producto y fundamentalmente son las características que se representan por casos de usos. (7)

1.5.2. Clasificaciones de los requisitos no funcionales:

Existen diferentes clasificaciones de los requisitos no funcionales entre ellas:

- **Requisito de Fiabilidad**

Por parte del usuario la fiabilidad es uno de los requisitos más importantes debido a que debe tener en cuenta la recuperación frente a fallos de conexión: asegurar que no se pierdan los datos del perfil definido por el usuario. Esto incluye no perderlos en el envío al servidor o a otras máquinas, como no perderlos si hay un fallo de conexión entre el receptor del usuario y el servidor.

La recuperación frente a fallos del sistema: posibilidad de reiniciar el sistema. Verificar la fiabilidad en la autenticación de los usuarios y la posibilidad de dar marcha atrás en la definición del perfil de cada usuario. (7)

- **Requisito de Seguridad**

Este requisito tiene en cuenta la seguridad y el ambiente en el que se usará el sistema. Por lo que se debe contemplar la seguridad física del lugar donde se usa la aplicación, los controles administrativos que se establecen de acceso al sistema y las regulaciones legales que afecta o determina el uso del sistema y que serán tenidas en cuenta si se incumple. La seguridad puede ser tratada en tres aspectos diferentes:

Confidencialidad: La información manejada por el sistema está protegida de acceso no autorizado y divulgación.

Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos. Pueden incluir también mecanismos de chequeo de integridad y realización de auditorías.

Disponibilidad: Significa que los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado. (7)

- **Requisito de Eficiencia**

Se debe tener en cuenta el requisito de eficiencia porque convendría que los usuarios examinaran con detenimiento hasta qué punto el proyecto se ajusta a sus expectativas en cuanto a los tiempos de respuesta y es capaz de prestar servicio adecuadamente de acuerdo al tipo y tamaño para el que ha sido concebido. (7)

- **Requisito de Disponibilidad**

Hoy las instituciones o centros de trabajos utilizan los sistemas informáticos, conectados a la red. Un cambio fundamental lo constituirá el aumento espectacular de la dependencia de los usuarios con respecto a la red, de modo que si el software dejara de estar disponible, a los usuarios les será imposible continuar su trabajo. Por consiguiente, convendría que los usuarios que se disponen a contratar un sistema identifiquen las necesidades de sus usuarios en cuanto a disponibilidad y las detallen en el momento de la contratación. (7)

- **Requisito de Usabilidad**

La definición de este requisito es una cuestión de especial importancia, pues un proyecto puede fracasar porque sus usuarios no encuentren sencillo su uso, porque no exista una interfaz sencilla y atractiva, o un manual que describa el funcionamiento y el uso del sistema al usuario final. (7)

- **Requisito de Soporte**

Le brinda al usuario la facilidad de instalación, facilidad de mantenimiento, lo que requiere código y diseño documentado y facilidad de actualización hacia versiones más modernas. (7)

- **Requisitos de Software**

Debe mencionarse el software del que se debe disponer, después de implementado el sistema. (7)

- **Requisitos de Hardware**

Se deben enunciar los elementos de hardware que se necesitan para que el software cumpla sus funcionalidades. (7)

- **Requisitos de apariencia o interfaz externa**

Este tipo de requisito describe la apariencia del producto. Es importante destacar que no se trata del diseño de la interfaz en detalle sino que especifica cómo se pretende que sea la interfaz externa del producto. También pueden ser necesidades de cumplir con normas, o con los estándares de la empresa para la cual se esté desarrollando el software. (7)

1.6. Pruebas no funcionales

Las pruebas no funcionales se realizan para verificar que el software desarrollado cumple con los requerimientos no funcionales establecidos por el cliente, son aquellas evaluaciones que se realizan sobre los elementos que están presentes en los

resultados de la ejecución de funcionalidades del sistema. Existen varios tipos de pruebas no funcionales, entre las más comunes están las pruebas de seguridad, de rendimiento, de usabilidad, de portabilidad, entre otras. (8)

Se debe conocer que las pruebas funcionales se encargan de testear una funcionalidad en específico, es decir un conjunto de entradas son aplicadas a un módulo y según las salidas que este proporcione se verifica si funciona correctamente o no. Si bien este tipo de pruebas son necesarias, no son las únicas y no muestran el comportamiento real de la aplicación bajo condiciones de producción. Por tal motivo se hace necesario realizar las pruebas no funcionales, estas se encargan de ver el comportamiento de la aplicación bajo situaciones similares a las de producción. (9)

Existen pruebas no funcionales tales como:

- Pruebas de Rendimiento (Estrés)

Este tipo de pruebas no se realiza para encontrar defectos en una aplicación, sino para eliminar cuellos de botella y establecer una línea base que pueda ser utilizada en un futuro para comparar el incremento en el rendimiento de la aplicación bajo pruebas. Este servicio comúnmente incluye las pruebas de carga, pruebas de estrés y pruebas de resistencia, entre otras. (9)

- Pruebas de Disponibilidad

Las pruebas de disponibilidad verifican que sus procesos de negocio están funcionando de forma correcta, al margen del estado de sus recursos de hardware. Incluyen la implantación de herramientas para verificar de manera continua que su entorno está funcionando como espera y su disponibilidad está dentro de los límites que han sido previamente establecidos. (9)

- Pruebas de Seguridad

Las pruebas de seguridad cubren el proceso de evaluar la seguridad de sus sistemas desde un punto de vista externo y sin conocimiento previo de los mismos. Los sistemas y políticas de seguridad son analizados exhaustivamente con el fin de encontrar fallos de seguridad, tanto en el diseño, como en la implementación de la aplicación. (9)

- Pruebas de Aceptación Operacional (OAT por sus siglas en inglés)

Las OAT se realizan como paso inmediatamente anterior a la puesta en producción de un sistema. El objetivo general de este tipo de pruebas es comprobar que la aplicación

dispone de la fiabilidad y el soporte necesarios para su puesta en marcha en producción. (9)

1.7. Procedimiento de pruebas no funcionales

Un procedimiento de prueba especifica cómo realizar uno o varios casos de pruebas. Este puede ser una instrucción para un individuo sobre cómo realizar un caso de prueba manualmente o puede ser una especificación de cómo interactuar manualmente con una herramienta automatizada para crear componentes ejecutables de pruebas. (10)

Es además un método de ejecución o pasos a seguir, en forma secuenciada y sistemática, en la consecución de un fin.

Se llama procedimiento de trabajo a la disposición de medios materiales y humanos para llegar en base a un plan a desarrollar actividades coordinadas para lograr un fin laboral, por ejemplo un aumento en las ventas. (10)

En los diferentes centros de producción se entrevistaron una o dos personas vinculadas con la realización de las pruebas de software, pudiendo observarse que escasamente se definen procedimientos para la realización de pruebas no funcionales y que en la mayoría de los centros se realizan acciones aisladas para comprobar el rendimiento o algún otro requisito no funcional, pero no tienen formalizada alguna propuesta que mida cuan satisfactorio ha sido el cumplimiento de estos requisitos. Se encontró en el proyecto Laboratorios Virtuales del CEDIN un procedimiento que evalúa algunos requisitos no funcionales y luego de un estudio de dicho procedimiento se toman aspectos coincidentes que pueden servir de guía para la realización de pruebas no funcionales de los productos desarrollados en el CEDIN, ejemplo de ello es el trabajo de diploma realizado el curso anterior titulado Procedimiento de Pruebas No Funcionales a Laboratorios Virtuales. (11)

1.8. Herramientas y tecnologías

Herramienta CASE

Las herramientas CASE (del inglés *Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software.

Se elige Visual Paradigm para Lenguaje Unificado de Modelado (LUM o *UML* por sus siglas en inglés, *Unified Modeling Language*), herramienta UML profesional que soporta

el ciclo de vida completo del desarrollo de software. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso y generar documentación. Cuenta con la ventaja de ser una herramienta multiplataforma y permite la integración con Eclipse IDE. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar, y compatible entre ediciones. Genera código para Java y exportación como HTML. (12)

Lenguaje de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras (13). Se elige como lenguaje de programación Java.

HTML

Es el lenguaje con el que se describen las páginas web. Permite escribir texto en forma estructurada, y compuesto por etiquetas que marca el inicio y fin de cada elemento en el documento. Puede contener imagen, sonido, video y son representados por los navegadores (14).

CSS

Las hojas de estilo en cascada (CSS por sus siglas en inglés y acrónimo de *Cascading Style Sheets*), es un lenguaje para definir la presentación de un documento estructurado escrito en HTML o XML. Su uso permite el control centralizado de la presentación de un sitio web completo, agilizando considerablemente su actualización (15).

Lenguaje para el modelado

El Lenguaje de Modelado Unificado (UML por sus siglas en inglés de *Unified Modeling Languaje*), es el lenguaje que se utilizará para modelar el componente. UML es el lenguaje más conocido y utilizado en la actualidad para el modelado de sistemas de software, que permite además construir y documentar los elementos que forman un sistema de software orientado a objetos. Es un lenguaje en el que está descrito el modelo y es aplicable en gran variedad de formas para dar soporte a la metodología de desarrollo de software definida para la solución (16).

Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de

herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. (17)

Aunque existen hoy en día un sin número de entornos de desarrollo integrado como por ejemplo: *Code::Blocks, Eclipse, Lazarus, KDevelop ,Netbeans, etc.,* se elige como IDE *Eclipse*, basado en la licencia *Common Public License Version 1.0* ("CPL"), porque es totalmente extensible al permitir la creación de nuevos componentes, así como utilizar los que ya están implementados.

Entre sus principales características como IDE de Java se encuentran (17):

- Editor visual con sintaxis coloreada.
- Compilación incremental de código.
- Modifica e inspecciona valores de variables.
- Avisa de los errores cometidos mediante una ventana secundaria.
- Depura código que resida en una máquina remota.

Frameworks RIA para el desarrollo de interfaces web

En la actualidad existen disimiles *frameworks* RIA[1] que ofrecen una experiencia de uso rica y atractiva que mejora la satisfacción del usuario e incrementa la productividad del desarrollador, ofrecen ventajas para desarrollar una interfaz web, hacen que el desarrollo sea más rápido y se obtenga software de calidad. Son librerías con componentes reutilizables y funciones que facilitan el trabajo. Ofrecen patrones que mejoran el desarrollo del software.

Para el análisis se han escogido cuatro de estas potentes librerías que existen hoy en día en el mundo, estas son: Dojo Toolkit, JQuery, Ext JS y Vaadin. Las tres primeras son basadas en JavaScript y se ejecutan en el cliente, a diferencia de Vaadin que es basada en Java y se ejecuta del lado del servidor (18).

Dojo Toolkit

Es una librería JavaScript que proporciona una API para el control y manipulación de historial, permite la manipulación de URL y marcadores/favoritos, el trabajo con widgets[2]

[1] *Rich Internet Application:* Aplicaciones web que ofrecen una experiencia enriquecida al usuario, con mejoras en cuanto a la conexión y la interactividad que la convierte en una aplicación más natural, más viva.
[2] Abreviación de las palabras *window* y gadget, es una mini-aplicación de ordenador usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor. Dentro de sus

e incluye el ordenamiento de tablas, validación de formularios, menús y barra de menús. La habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente (19).

Su idea es abstraer al desarrollador de las complejidades de DHTML [3] y de las discrepancias existentes entre navegadores, que hacen el código JavaScript a utilizar sea diferente. Dojo Toolkit está formado por un grupo de bibliotecas de capas. También puede usar el API de bajo nivel y capas compatibles para escribir scripts portables y simplificar aquellos que son complejos que le permite personalizar la distribución de Dojo Toolkit para la aplicación. Proporciona un potente entorno de programación incorporando un sistema de control de eventos, un API E/S y un lenguaje estándar mejorado que simplifican la vida de JavaScript para diseñadores. Se pueden usar las herramientas de Dojo Toolkit para construir unidades de prueba para el código JavaScript y optimizar el desarrollo.

Algunas de las características que presenta esta librería son la independencia en cuanto al intérprete, ya que asegura un soporte para el mayor número de plataformas, y es manejable en cuanto a la incompatibilidad entre navegadores web.

Permite el desarrollo rápido de aplicaciones ricas de Internet con una alta calidad visual y funcional, además de la compatibilidad con los navegadores modernos. Es el tipo de herramientas a la cual los desarrolladores experimentados recurren para la construcción de aplicaciones web y aplicaciones para móviles con características de escritorio. Dojo se adapta a las necesidades del desarrollador (20).

JQuery

Es una librería JavaScript rápida y concisa, que maneja eventos, realiza animaciones y añade interacciones AJAX en la web, facilita la forma de trabajar con documentos HTML. JQuery ha sido diseñado para cambiar la forma de escribir JavaScript.

Las tareas más comunes que se realizan a la hora de desarrollar con JavaScript para agregar funcionalidad dinámica a las páginas que siguen el patrón de seleccionar un elemento o grupo de elementos y operar sobre estos elementos de diversas maneras: como mostrar u ocultar el elemento, agregar alguna clase CSS, animar o modificar sus atributos, normalmente utilizando JavaScript nativo u otra librería resultaría en escribir

objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

[3] Es la combinación de HTML, JavaScript, DOM y CSS

una docena de líneas de código por cada una de estas tareas. Sin embargo gracias a los creadores de JQuery que han creado un sistema muy potente de selección y de expresiones para la identificación de elementos de una página, estas tareas comunes se pueden desarrollar de forma trivial (21).

Year	Make	Model
1965	Ford	Mustang
1970	Toyota	Corolla
1979	AMC	Jeep CJ-5
1983	Ford	EXP
1985	Dodge	Daytona
1990	Chrysler	Jeep Wrangler Sahara
1995	Ford	Ranger
1997	Chrysler	Jeep Wrangler Sahara
2000	Chrysler	Jeep Wrangler Sahara
2005	Chrysler	Jeep Wrangler Unlimited
2007	Dodge	Caliber R/T

Figura 1 Ejemplo de formato con JQuery

Con la utilización de JQuery solamente se hace necesario utilizar una línea de código para darle formato a la tabla Zebra (ver **¡Error! No se encuentra el origen de la referencia.**), tarea la cual constaría de varias líneas de códigos en cualquier otro marco de trabajo. Algunos de sus elementos y características se mencionan a continuación:

- Atributos: Permiten añadir, eliminar, seleccionar, cambiar y leer los atributos de los elementos DOM de forma sencilla.
- CSS: Funciones para el cambio de estilo.
- Desplazamiento: Funciones de filtrado y búsqueda.
- Eventos: Control de eventos para elementos DOM.
- Manipulación: Funciones de inserción y modificación.
- Utilidades: Otras funciones genéricas adicionales, permiten referenciar objetos sin problemas de espacios de nombres.
- Efectos: Sencillos efectos visuales. Si no fuera por los plugins que hay desarrollados para JQuery quedaría muy por debajo de otros marcos de trabajo en este punto.

Ext JS

La librería Ext JS, fue concebida en sus inicios para extender las funcionalidades de la librería YUI e integrar AJAX, Prototype y Scriptaculous. Con el tiempo se convirtió en un framework independiente y a principio del 2007 se creó una compañía para comercializar y dar soporte al Ext JS.

Ext JS es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado. Existen docenas de widgets a escoger en Ext JS, incluyendo composiciones automáticas de páginas, pestañas, menús, barras de herramientas, diálogos, vistas en árbol (5). Proporciona un selector de nodos DOM extremadamente poderoso llamado DomQuery (puede usarse como una librería independiente, pero en el contexto de Ext JS se usará para seleccionar elementos para poder interactuar con ellos a través de la interfaz Element, contiene mucho de los métodos y propiedades de DOM que se necesitará, proporcionando una interfaz conveniente, unificada y multi-navegador).

Para trabajar con las librerías de Ext JS anterior a la versión 2.0 era necesario que existiera un adaptador. A partir de la versión 2.0, Ext JS proporciona su propio adaptador, ya no es necesario incluir el de YUI, Scriptaculous (incluir en el caso de usar alguna función de estas librerías que no esté en Ext).

Para desarrollar con Ext JS se emplean IDEs de desarrollo como: el Aptana Studio[4] que es una potente herramienta disponible para el apoyo directo en el desarrollo de aplicaciones (22).

Vaadin

Esta librería fue creada en sus inicios (año 2001) por un grupo de desarrolladores que tenían la idea de desarrollar un nuevo paradigma de programación, que permitiera la creación de interfaces de usuarios reales para aplicaciones reales, usando un lenguaje de programación robusto. En sus inicios era llamada Mill Toolkit, no fue hasta 2006 que fue renombrada por Vaadin.

Es un framework de desarrollo de aplicaciones web del lado del servidor que da la posibilidad al desarrollador de construir interfaces de usuario de alta calidad con el lenguaje Java. Provee una librería de componentes de interfaces y la posibilidad de crear componentes personalizados. Sus mayores potencialidades están en la facilidad

[4]Es uno de los IDE libres más potentes para el desarrollo web, es rápida, personalizable y le da al desarrollador las herramientas necesarias para ser más productivo.

de uso, extensibilidad y reusabilidad. Vaadin fue diseñado para crear aplicaciones web de negocio reales, no para diseñar sitios web (23).

La idea clave en el modelo de programación que propone Vaadin es que es independiente de la web, permitiendo desarrollar interfaces de usuario como si se estuviera desarrollando aplicaciones Java de escritorio con herramientas convencionales, tales como AWT, Swing[5], o SWT. Inclusive de forma mucho más intuitiva. Es un framework ideal para desarrolladores que no tengan mucho conocimiento de las tecnologías web como HTML, JavaScript y deseen ser productivo y concentrarse en la lógica de la aplicación.

Para el desarrollo con la librería se puede emplear cualquier IDE, pero es recomendado que se utilicen los *plugins* para Eclipse IDE, el cual se ha convertido en el ambiente estándar de desarrollo Java.

Figura 2 Arquitectura General de Vaadin

La figura 2 muestra la arquitectura base de cualquier aplicación web desarrollada con Vaadin. Vaadin no solo es un framework del lado del servidor, sino que tiene un motor del lado del cliente que corre como un programa JavaScript, el cual permite renderizar la interfaz de usuario y delegar la interacción con el usuario al servidor (23).

Selección del framework.

En esta sección se han abordado algunas de las principales herramientas más utilizadas en el mundo y en Cuba, específicamente en la UCI para la implementación de interfaces web, tanto de herramientas Java del lado del servidor como librerías JavaScript del lado del cliente. Se ha realizado un estudio de las principales características de cada una de ellas con el objetivo de seleccionar la que más se adecua a las necesidades del sistema que se desea desarrollar.

Tabla 1. Comparativa de frameworks.

Indicadores	DojoToolkit	JQuery	ExtJS	Vaadin

[5] Librería Java para crear Interfaces Gráficas de Usuario

Conocimiento de la librería	No	No	No	Conocimiento Java
Documentación	Si	Si	Si	Si
Tecnologías que se deben dominar para su utilización.	HTML, JavaScript, CSS	HTML JavaScript, CSS	HTML JavaScript CSS	Java
Widgets	Si	plugins	Si	Si

De acuerdo a las características de estas herramientas presentadas en la tabla anterior, la opción ideal para implementar las interfaces del sistema a desarrollar es Vaadin, por posibilitar el desarrollo de interfaces sin necesidad de conocer las tecnologías de la web como HTML, JavaScript y CSS. El equipo de desarrollo solo se concentraría en desarrollar la lógica de la aplicación y no invertiría tiempo en aprender estas tecnologías, ya que el desarrollo con Vaadin se basa en conocer la sintaxis de Java. Además posee las características necesarias para desarrollar aplicaciones con niveles de usabilidad aceptables y se puede lograr el desarrollo con el menor esfuerzo posible y simplificando el tiempo de desarrollo, por lo cual se considera el más idóneo, atendiendo a las necesidades del sistema a implementar.

Apache Tomcat

Apache Tomcat (también llamado *Jakarta Tomcat* o simplemente *Tomcat*) funciona como un contenedor de *servlets* desarrollado bajo el proyecto *Jakarta* en la *Apache Software Foundation*. *Tomcat* implementa las especificaciones de los *servlets* y de *Java Server Pages* (JSP) de *Sun Microsystems*. *Tomcat* es un servidor web con soporte de *servlets* y JSPs. *Tomcat* no es un servidor de aplicaciones, como *JBoss* o *JOnAS*. Incluye el compilador *Jasper,* que compila JSPs convirtiéndolas en *servlets*. El motor de *servlets* de *Tomcat* a menudo se presenta en combinación con el servidor web *Apache*. Dado que *Tomcat* fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java. (24)

Sistema Gestor de Bases de Datos

A pesar de que hoy en día existen diversos gestores de base de datos como por ejemplo: Microsoft SQL Server, MySQL, PostgreSQL, etc., se elige PostgreSQL debido a que es un Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos, de código abierto y brinda un control de concurrencia multi-versión (MVCC por sus siglas

en inglés) que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación, entre el que se encuentra Java.

Posee características significativas del motor de datos, entre las que se pueden incluir las subconsultas, los valores por defecto, las restricciones a valores en los campos (constraints) y los disparadores (triggers). El código fuente se encuentra disponible para todos sin costo alguno. Posee una integridad referencial e interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, ECPG, PYTHON y RUBY. Funciona en todos los sistemas operativos Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Por todas estas características se decide su utilización en la solución planteada. (25)

JDBC

Java Database Connectivity, más conocida por sus siglas JDBC, es una API[6] que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos url que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos y accede a ella estableciendo una conexión, para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí se puede realizar con cualquier tipo de tareas, con la base de datos a las que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, entre otras tareas.

Tabla 2 Principales clases para el uso de JDBC.

Clase	Descripción
DriverManager	Para cargar un driver.
Connection	Para establecer conexiones con las bases de datos.

[6] Interfaz de programación de aplicaciones.

Statement	Para ejecutar sentencias SQL y enviarlas a las BD.
ResultSet	Para almacenar el resultado de la consulta.

JUnit

Es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. Es un conjunto de clases (*framework*) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces *JUnit* devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, *JUnit* devolverá un fallo en el método correspondiente.

Es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

El propio *framework* incluye formas de ver los resultados (*runners*) que pueden ser en modo texto, gráfico (*AWT* o *Swing*) o como tarea en *Ant*. (27)

1.9. Metodologías de desarrollo del software

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tiene aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. (28)

Entre las metodologías de desarrollo del software más utilizadas se encuentran *Rational Unified Process* (RUP) y *Extreme Programing* (XP).

Rational Unified Process (RUP):

La metodología RUP, llamada así por sus siglas en inglés *Rational Unified Process*, divide en 4 fases el desarrollo del software:

- **Inicio**: El objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**: En esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción**: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- **Transición**: El objetivo es llegar a obtener el *release* del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

- Disciplina de Desarrollo
- Disciplina de Soporte

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (29)

Extreme Programing (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo, pequeño equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Características de XP, la metodología se basa en:

- **Pruebas Unitarias:** se basa en las pruebas realizadas a los principales procesos, se hacen pruebas de las fallas que pudieran ocurrir.
- **Re-fabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores

participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa. (29)

Metodología seleccionada para el desarrollo del sistema

Partiendo del análisis anterior, XP es la metodología seleccionada ya que sus características favorecen el desarrollo de la solución. Entre las cuales están: la retroalimentación continua entre el cliente y el equipo de desarrollo, la producción de versiones del sistema de manera rápida, el diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto y además, soporta el cambio de requerimientos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

El ciclo de vida ideal al emplear la metodología XP consta de seis fases, de las cuales se abordarán en el presente trabajo las primeras cuatro fases: exploración, planificación, iteración y producción.

Conclusiones parciales del capítulo

Luego del análisis de la bibliografía consultada se obtuvo una visión más clara acerca de los conceptos de calidad y pruebas de software. Además se elige como norma la ISO 25000 y la ISO 17999:2007 para apoyar el procedimiento de pruebas no funcionales. Como metodología de desarrollo se escogió XP y para la implementación de la herramienta el framework Vaadin y el entorno de desarrollo Eclipse.

Capítulo 2. Propuesta de solución

2.1. Introducción

En el presente capítulo se describe el procedimiento para la realización de pruebas no funcionales a productos del CEDIN. Dicho procedimiento fue creado para dar cumplimiento a los objetivos planteados en la investigación, se centra en definir los roles y sus responsabilidades, las actividades correspondientes, los artefactos de entrada y salida que intervienen en la evaluación de los requisitos no funcionales, específicamente de portabilidad, mantenimiento, fiabilidad, eficiencia, usabilidad y seguridad. Se describe además una herramienta para gestionar las pruebas no funcionales que se desarrollarán con el procedimiento.

2.2. Procedimiento para realizar pruebas a requisitos no funcionales de productos del CEDIN

Para el desarrollo de este procedimiento se parte de los procedimientos descritos en la tesis de grado titulada: Procedimiento de pruebas no funcionales para laboratorios virtuales con fines educativos (11). En el caso de los requisitos no funcionales de portabilidad, mantenimiento, fiabilidad, usabilidad y eficiencia las actividades del procedimiento van dirigidas a orientar al probador para la obtención de los parámetros necesarios para el cálculo de las métricas que se proponen en la ISO 25000 y para el requisito de seguridad las que se proponen en la ISO/IEC 17799: 2007, para la evaluación de cada uno de estos requisitos, una vez obtenidos estos parámetros, se ejecutan las pruebas correspondientes utilizando la herramienta para la gestión de pruebas no funcionales implementada en el presente trabajo, donde se registran los parámetros necesarios para las métricas. En esta herramienta se van a mostrar los resultados de la evaluación de cada uno de los requisitos y se procede a la actividad 5 “Análisis de Resultados” para analizar el grado de cumplimiento de los requisitos no funcionales del producto en pruebas y se asignan responsabilidades para la resolución de las no conformidades, en caso de ser detectadas.

2.2.1. Descripción del Procedimiento para evaluar el cumplimiento de la usabilidad de los productos del CEDIN

Para evaluar la capacidad del producto de software de ser comprendido, aprendido, utilizado y de ser atractivo para el usuario, cuando se utilice bajo las condiciones especificadas se debe seguir el procedimiento general para la realización de pruebas no

funcionales a los productos del CEDIN. Para realizar la primera actividad de dicho procedimiento general el artefacto de entrada es la descripción del producto.

A continuación se presenta la Tabla 3 que recoge cada una de las sub-características de usabilidad con su definición y el procedimiento con que debe ser medida la misma.

Tabla 3 Métricas para la medición de la característica usabilidad

Nombre	Definición	Metas	Procedimiento
Integridad de la descripción de producto (Subcaracterística: Comprensibilidad).	X = A/ B A = número de funciones comprendidas. B = número total de funciones.	¿Qué proporción de funciones (o tipos de funciones) son comprendidas después de leer la descripción del producto?	Conducir las pruebas de usuario. Realizar entrevistas a usuarios. Observar el comportamiento del usuario. Contar el número de funciones que fueron adecuadamente comprendidas en comparación con el número total de funciones en el producto. 0 <= X <= 1 A mayor cercanía al 1 resultará mejor.
Accesibilidad a demos. (Subcaracterística: Comprensibilidad).	X = A/ B A = número de demos/tutoriales a los que pueden acceder los usuarios exitosamente. B = número total de demos/tutoriales a los que se puede acceder.	¿A qué proporción de demos/tutoriales pueden acceder los usuarios?	Conducir las pruebas de usuario. Observar el comportamiento del usuario. 0 < X <= 1 Mientras más cercano se esté al 1, mejor.
Comprensibilidad de entradas y salidas. (Subcaracterística:	X = A/B A = número de elementos de entrada y que suministra el sistema de software como salida comprendidas correctamente.	¿Pueden los usuarios comprender lo que se requiere como entrada y lo que suministra el sistema de software	Conducir las pruebas de usuario. Realizar entrevistas a usuarios. Observar el comportamiento del usuario. Contar el

Comprensibilidad).	B = número total de elementos de entrada y que suministra el sistema de software como salida proporcionados por la interfaz.	como salida?	número de elementos de entrada. 0 <= X <= 1 Mientras más cercano al 1, mejor.
Adaptabilidad de la apariencia de la interfaz (Subcaracterística: Atracción)	X = A/ B A = número de elementos de la interfaz del sistema cuya apariencia puede ser adaptada por el usuario. B = número de elementos de la interfaz del sistema cuya apariencia querría adaptar el usuario.	¿Qué proporción de los elementos de la interfaz puede ser, por su apariencia, adaptado por el usuario para la satisfacción del mismo?	Conducir las pruebas de usabilidad. Observar el comportamiento del usuario. 0 <= X <= 1 Mientras más cercano al 1, mejor.

A continuación se muestra paso a paso el procedimiento textual para evaluar la usabilidad en los productos del CEDIN.

Tabla 4 Procedimiento textual para evaluar la usabilidad en los productos del CEDIN

Procedimiento textual para evaluar la usabilidad en los productos del CEDIN.		
Criterio de entrada.	Plan de pruebas.	
Criterio de salida.	Acta de liberación del producto.	
No.	**Descripción**	**Salida**
1.	1.1. Asigna la ejecución de la prueba para evaluar la usabilidad. (Administrador de la calidad) 1.2. Buscar los artefactos para realizar las pruebas pertinentes. (Probador).	Petición del GESPRO para preparar el entorno de la prueba para evaluar la usabilidad (resuelto).
2.	2.1. Realizar entrevistas a usuarios para verificar que proporción de funciones son comprendidas por el usuario después de leer la descripción del producto. 2.2. Registrar en la Aplicación para la gestión de las pruebas no funcionales, la cantidad de funciones que fueron adecuadamente comprendidas y la cantidad	Petición del GESPRO (resuelto). Reportes de resultados de

	total de funciones en el producto. Indicadores que corresponden a la métrica Integridad de la descripción de producto (Probador). 2.3. Observar el comportamiento de varios usuarios al usar demos y tutoriales del producto. 2.4. Registrar en la Aplicación para la gestión de las pruebas no funcionales, el número de demos/tutoriales a los que pueden acceder los usuarios exitosamente y el número total de demos/tutoriales a los que se puede acceder. Indicadores que corresponden a la métrica Accesibilidad a demos (Probador). 2.5. Realizar entrevistas a usuarios y verificar si pueden los usuarios comprender lo que se requiere como entrada y lo que suministra el sistema de software como salida. 2.6. Registrar en la Aplicación para la gestión de las pruebas no funcionales, el número de elementos de entrada, que suministra el sistema de software como salida comprendidas correctamente y el número total de elementos de entrada, que suministra el sistema de software como salida proporcionados por la interfaz. Indicadores que corresponden a la métrica Comprensibilidad de entradas y salidas (Probador). 2.7. Observar el comportamiento de los usuarios para verificar la proporción de los elementos de la interfaz que pueden ser, por su apariencia, adaptado por el usuario para la satisfacción del mismo. 2.8. Registrar en la Aplicación para la gestión de las pruebas no funcionales, el número de elementos de la interfaz del sistema cuya apariencia puede ser adaptada por el usuario y el número de elementos de la interfaz del sistema cuya apariencia querría adaptar el usuario. Indicadores que corresponden a la métrica Adaptabilidad de la apariencia de la interfaz (Probador).	la aplicación para la gestión de las pruebas no funcionales para la usabilidad.
3.	3.1. En caso que existan No Conformidades ir a la actividad 4. 3.2. En caso que no existan No conformidades ir a la actividad 5.	
4.	4.1. El probador debe registrar las No conformidades en el registro de no conformidades.	Petición del GESPRO (No conformidades asignadas) y Registro de No Conformidades.
5.	5.1. Ejecutar el cálculo de las métricas registradas en la Aplicación para la	Reportes de

	gestión de las pruebas no funcionales. (Probador). 5.2. Evaluar el nivel de implementación de la usabilidad (Probador).	resultados de la aplicación para la gestión de las pruebas no funcionales para la usabilidad.

2.2.2. Descripción del Procedimiento para evaluar el cumplimiento de la seguridad para los productos del CEDIN

En la presente investigación se desarrolla un procedimiento para evaluar la capacidad del producto de garantizar la confidencialidad, integridad y disponibilidad de la información.

El procedimiento fue creado para dar cumplimiento a los objetivos planteados en la investigación, se centra en definir los roles y sus responsabilidades, las actividades correspondientes, los artefactos de entrada y salida que intervienen en la evaluación del requisito no funcional de seguridad de los productos de software. Dicho procedimiento calcula las métricas definidas en la ISO/IEC 17799 para la medición de la característica de calidad seguridad. (4)

A continuación se presenta la Tabla 5 que recoge cada una de las sub-características de seguridad con su definición y el procedimiento con que debe ser medida la misma.

Tabla 5 Métricas para la medición de la característica seguridad.

Nombre	Definición	Metas	Procedimiento
Autenticación de un Sistema	X = A/ B X: Grado de seguridad en la autenticación de un sistema. A: Fallos ocurridos en la autenticación de un sistema. Es la cantidad de fallos a los que el sistema es vulnerable. B: Posibles fallos que pueden ocurrir cuando se autentica un usuario en un sistema.	¿Qué tan segura es la autenticación del sistema?	Conducir las pruebas a la autenticación de usuarios bajo condiciones erróneas y observar el comportamiento del sistema ante estas fallas. Contar los fallos ocurridos en la autenticación del

			sistema, en comparación con el número total de fallos que pueden ocurrir. 0 <= X <= 1 A mayor cercanía al 1 resultará mejor.
Inyección de Código SQL y Cross-Site Scripting (XSS)	X = (A + B/2)*100 X = Grado de Seguridad en la Inyección de Código SQL y Cross-Site Scripting (XSS). A: Funcionalidades que utilizan la eliminación de código malicioso SQL, toma valores de 1 si existen y 0 si no existen. B: Funcionalidades que utilizan la eliminación de código malicioso Cross-Site Scripting (XSS), toma valores de 1 si existen y 0 si no existen.	¿Cómo se comporta el grado de seguridad que tienen las aplicaciones ante una Inyección de Código SQL y Cross-Site Scripting (XSS)?	Conducir las pruebas a verificar la existencia de funciones que utilizan la eliminación de código malicioso SQL y Cross-Site Scripting (XSS). Mientras mayor cercanía tenga X al 100, mejor.
Acceso a la Aplicación	X = (A + B)/C * 100 X: Grado de Seguridad para evaluar el Acceso a la Aplicación. A: Cantidad de páginas o módulos que redireccionan a la página de autenticación cuando se acceden a ellas. B: Indica si el sistema utiliza el mecanismo de denegación IP, tomando valor 0 ó 1. C: Número de páginas o módulos que posee la aplicación	¿Cómo se comporta el grado de seguridad que tienen las aplicaciones ante el Acceso a la Aplicación?	Conducir las pruebas hacia la verificación del Grado de Seguridad para evaluar el Acceso a la Aplicación. Mientras mayor cercanía tenga X al 100, mejor.
Concurre	X = (A/ B)*100	¿Cómo se	Conducir las

ncia de usuarios utilizando el mismo nombre de usuario y contraseña	X: Grado de seguridad al evaluar la concurrencia de usuarios utilizando el mismo nombre de usuario y contraseña. A: Cantidad de usuarios y contraseñas diferentes en una aplicación en un mismo tiempo. B: Cantidad de usuarios y contraseñas totales en una aplicación en un mismo tiempo.	comporta el grado de seguridad al evaluar la concurrencia de usuarios utilizando el mismo nombre de usuario y contraseña?	pruebas hacia la verificación del Grado de Seguridad para evaluar la concurrencia de usuarios utilizando el mismo nombre de usuario y contraseña. Mientras mayor cercanía tenga X al 100, mejor.
Administración de Sesiones	X = (A + B + C +D) / 4*100 X: Grado de seguridad al evaluar la administración de sesiones. A: Grado de seguridad a la hora de tomar la dirección IP desde donde se autentica. Esta variable puede tomar valor de 0 ó 1. B: Grado de seguridad a la hora de tomar el nombre de usuario desde donde se autenticó. Esta variable puede tomar valor de 0 ó 1. C: Grado de seguridad a la hora de tomar el tiempo de la última petición del usuario en una sesión. Esta variable puede tomar valor de 0 ó 1. D: Grado de seguridad en el estado de expiración de la sesión. Esta variable puede tomar valor de 0 ó 1.	¿Cómo se comporta el grado de seguridad al evaluar la administración de sesiones?	Conducir las pruebas hacia la verificación del Grado de Seguridad para evaluar la administración de sesiones. Mientras mayor cercanía tenga X al 100, mejor.
Gestión de Usuarios de un Sistema	X = A/B X: Grado de Seguridad para la gestión de usuarios de un sistema. A: Cantidad de fallos del sistema ocurridos en la gestión de usuarios de un sistema.	¿Cómo se comporta el grado de seguridad al evaluar la gestión de	Conducir las pruebas hacia la verificación del grado de seguridad para evaluar la

	B: Número de posibles fallos que se han definido a los cuales la aplicación puede ser vulnerable.	usuarios de un sistema?	gestión de usuarios de un sistema. 0 <= X <= 1 A mayor cercanía al 1 resultará mejor.
Seguridad	(∑X/6)*100	¿Cómo se comporta el grado de seguridad del sistema?	Mientras mayor cercanía tenga X al 100, mejor.

A continuación en la Tabla 6 se muestra paso a paso el procedimiento textual para evaluar la seguridad de un producto de software.

Tabla 6 Descripción textual del procedimiento para evaluar la seguridad de los productos desarrollados en el CEDIN.

Procedimiento textual para evaluar la seguridad de los productos desarrollados en el CEDIN.		
Criterio de entrada.	Plan de pruebas.	
Criterio de salida.	Acta de entrega interna del producto.	
No.	**Descripción**	**Salida**
1.	1.1. Asigna la ejecución de la prueba para evaluar la seguridad. (Administrador de la calidad) 1.2. Buscar los artefactos para realizar las pruebas pertinentes. (Probador).	Petición para preparar el entorno de la prueba para evaluar la seguridad (Resuelta).
2.	2.1. Autenticarse al sistema bajo condiciones erróneas como usuario o contraseñas incorrectas o inyección de código. 2.2. Registrar en la Plantilla de métricas, la cantidad de Fallos Ocurridos en la Autenticación de un Sistema y Posibles Fallos que pueden ocurrir cuando se autentica un usuario en un sistema. Indicadores que corresponden a la Métrica de Seguridad para evaluar la Autenticación de un Sistema (Probador).	Petición (Resuelta). Plantilla de métricas (actualizada) .

	2.3. Verificar en las funcionalidades del sistema si existen algunas funcionalidades que utilizan la eliminación de código malicioso SQL y Cross-Site Scripting (XSS). 2.4. Registrar en la Plantilla de métricas, 0 si no utiliza funcionalidades para la eliminación de código malicioso SQL y si no utiliza funcionalidades para la eliminación de código malicioso Cross-Site Scripting (XSS) o 1 si las utiliza. Indicadores que corresponden a la Métrica de Seguridad para evaluar la Inyección de Código SQL y Cross-Site Scripting (XSS) (Probador). 2.5. Verificar si el sistema utiliza el mecanismo de denegación IP y verificar la cantidad de páginas o módulos del sistema que re-direccionan a la página de autenticación cuando se acceden a ella. 2.6. Registrar en la Plantilla de métricas, 0 si no utiliza mecanismos de denegación de IP o 1 si utiliza mecanismos de denegación IP y registrar también el número de páginas o módulos que posee la aplicación. Indicadores que corresponde a la Métrica de Seguridad para evaluar el Acceso a la Aplicación y el número de páginas o módulos que re-direccionan a la página de autenticación cuando se acceden a ellas. Indicadores que corresponden a la Métrica de Seguridad para evaluar el Acceso a la Aplicación (Probador). 2.7. Contar en la base de datos de la aplicación la cantidad de usuarios y contraseñas diferentes en una aplicación en un mismo tiempo y la cantidad de usuarios y contraseñas totales en una aplicación en un mismo tiempo. 2.8. Registrar en la Plantilla de métricas, la cantidad de usuarios y contraseñas diferentes en una aplicación en un mismo tiempo y la cantidad de usuarios y contraseñas totales en una aplicación en un mismo tiempo. Indicadores que corresponden a la Métrica de Seguridad para evaluar la Concurrencia de Usuarios (Probador). 2.9. Verificar si el sistema utiliza la funcionalidad de verificar la dirección IP desde donde se autentica, verificar también el grado de seguridad a la hora de tomar el nombre de usuario desde donde se autenticó, a la hora de tomar el tiempo de la última petición del usuario en una sesión, en el estado de expiración de la sesión. 2.10. Registrar en la Plantilla de métricas, 0 si no se utilizan las funcionalidades de verificar la dirección IP desde donde se autentica, de	

	conocer el nombre del usuario donde se autenticó, de controlar el tiempo de la última petición del usuario, de verificar si una sección ya expiró o 1 si se utilizan estas funcionalidades. Indicadores que corresponden a la Métrica de Seguridad para evaluar la Administración de Sesiones (Probador). 2.11. Verificar la cantidad de fallos del sistema ocurridos en la gestión de usuarios de un sistema y la cantidad de posibles fallos a usuarios del sistema. 2.12. Registrar en la Plantilla de métricas, el número de fallos a los que el sistema es vulnerable y el número de posibles fallos que se han definido a los cuales la aplicación puede ser vulnerable. Indicadores que corresponden a la Métrica de Seguridad para evaluar la Gestión de Usuarios en un Sistema (Probador).	
3.	3.1. En caso que existan No Conformidades ir a la actividad 4. 3.2. En caso que no existan No conformidades ir a la actividad 5.	
4.	4.1. El probador debe registrar las No conformidades en el registro de no conformidades.	Petición (No conformidades asignadas) y Registro de No Conformidades.
5.	5.1. Ejecutar el cálculo de las métricas en la Plantilla de métricas. (Probador). 5.2. Evaluar el nivel de implementación de la seguridad (Probador).	Reportes de resultados a partir del cálculo de las métricas.

2.3. Herramienta para la gestión de pruebas no funcionales

La herramienta para la gestión de pruebas no funcionales surge a partir de la necesidad de automatizar la gestión de las pruebas para evaluar los requerimientos no funcionales en los proyectos del CEDIN. Es una aplicación web, multiplataforma, desarrollada en Eclipse haciendo uso del framework Vaadin, que permite la entrada de los datos necesarios para el cálculo de cada una de las métricas correspondientes a las características para la obtención de la calidad del software en cuanto a requerimientos

no funcionales. Los requisitos que se evalúan en la herramienta son: portabilidad, mantenimiento, fiabilidad, eficiencia, usabilidad y seguridad.

2.4. Exploración

La metodología de desarrollo XP comienza con la fase de exploración. Durante esta etapa se realiza el proceso de identificación de las historias de usuario, estas constituyen uno de los artefactos más importantes que se generan en la metodología, pues son la forma en que se especifican los requisitos del sistema (28). El ciclo de vida de XP enfatiza en el carácter iterativo e incremental del desarrollo. Una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas, que en el caso de XP corresponden a un conjunto de historias de usuarios.

2.4.1. Historias de Usuario

La historias de usuario (HU), se escribe desde la perspectiva del cliente, aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido de estas debe ser concreto y sencillo. Se realiza una por cada característica principal del sistema, es utilizada para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos. Las historias de usuario quedan estructuradas de la siguiente manera:

- **Nombre**: nombre descriptivo de la HU.
- **Prioridad**: grado de prioridad que le asigna el cliente a la HU en dependencia de su importancia. Los valores que puede tomar son: alta, media o baja.
- **Complejidad**: grado de complejidad que le asigna el desarrollador a la HU luego de analizarla. Los valores que puede tomar son: alta, media o baja.
- **Estimación**: unidades de tiempo estimadas por el desarrollador para darle cumplimiento a la HU.
- **Iteración**: número de la iteración en la cual será implementada la HU.
- **Descripción**: descripción simple que brinda el cliente sobre lo que debe hacer la funcionalidad en cuestión. (29)

Adicionalmente a cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.

Durante esta fase se identificaron once historias de usuario, cada una de ellas respondiendo a las diferentes funcionalidades solicitadas por el cliente y dando una idea

de cómo debe ser su posterior implementación. A continuación se describen algunas de las historias de usuario identificadas, seguidas de un prototipo de interfaz, para consultar las restantes historias de usuario ver Anexo #1.

Tabla 7 Historia de Usuario Autenticación.

Historia de Usuario	
Número: 1	**Nombre:** Autenticación
Prioridad en Negocio: Alta	**Complejidad en Desarrollo:** Alta
Puntos Estimados: 3	**Iteración Asignada:** 1
Descripción: El sistema permitirá la autenticación de usuario mediante el acceso con las credenciales del dominio UCI.CU.	

Figura 3 Prototipo de interfaz para la Historia de Usuario Autenticación

Tabla 8 Historia de Usuario Mostrar listado de proyectos.

Historia de Usuario	
Número: 2	**Nombre:** Mostrar listado de proyectos
Prioridad en Negocio: Baja	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 1
Descripción: El sistema mostrará en una tabla todos los proyectos adicionados por el usuario, incluyendo las opciones de adicionar, modificar y eliminar un proyecto, además permite adicionar, modificar y eliminar los productos pertenecientes a un proyecto seleccionado.	

Figura 4 Prototipo de interfaz para la Historia de Usuario Mostrar listado de proyectos

Tabla 9 Historia de Usuario Adicionar un proyecto.

Historia de Usuario	
Número: 3	**Nombre:** Adicionar un proyecto
Prioridad en Negocio: Alta	**Complejidad en Desarrollo:** Media
Puntos Estimados: 2	**Iteración Asignada:** 1
Descripción: El sistema le permitirá al usuario agregar un nuevo proyecto, admitiéndole especificar el nombre del mismo.	

Figura 5 Prototipo de interfaz para la Historia de Usuario Adicionar un proyecto

Tabla 10 Historia de Usuario Mostrar listado de productos.

Historia de Usuario	
Número: 6	**Nombre:** Mostrar listado de productos.
Prioridad en Negocio: Baja	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 1
Descripción: El sistema mostrará en una tabla todos los productos adicionados por el usuario, incluyendo las opciones de modificar y eliminar un producto.	

Figura 6 Prototipo de interfaz para la Historia de Usuario Mostrar listado de productos

Tabla 11 Historia de Usuario Adicionar un producto.

Historia de Usuario	
Número: 7	**Nombre:** Adicionar un producto
Prioridad en Negocio: Alta	**Complejidad en Desarrollo:** Media
Puntos Estimados: 2	**Iteración Asignada:** 1
Descripción: El sistema le permitirá al usuario agregar un nuevo producto, admitiéndole configurar el nombre y la versión del mismo.	

Figura 7 Prototipo de interfaz para la Historia de Usuario Adicionar un producto.

Tabla 12 Historia de Usuario Realizar Pruebas.

Historia de Usuario	
Número: 10	**Nombre:** Realizar Pruebas
Prioridad en Negocio: Alta	**Complejidad en Desarrollo:** Alta
Puntos Estimados: 3	**Iteración Asignada:** 2
Descripción: El sistema permitirá la realización de las pruebas a partir de los requisitos no funcionales que se decida evaluar. Se debe especificar el producto al cual se le realizará la prueba, así como los requisitos que se evaluarán. Luego se debe seleccionar la opción “Finalizar” para terminar la prueba.	

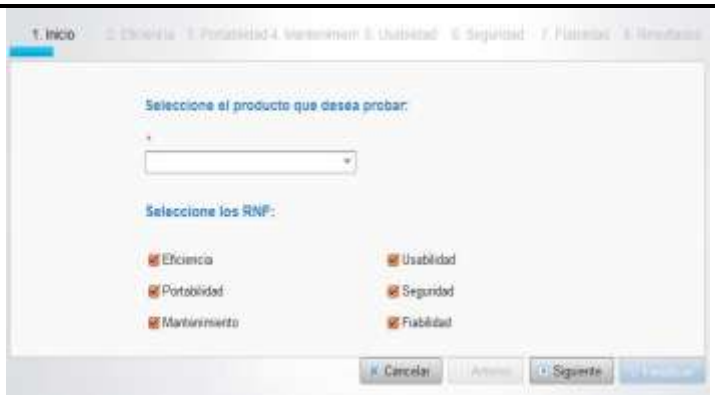

Figura 8 Prototipo de interfaz para la Historia de Usuario Realizar Pruebas.

Tabla 13 Historia de Usuario Mostar Resultados.

Historia de Usuario	
Número: 11	**Nombre:** Mostrar Resultados
Prioridad en Negocio: Alta	**Complejidad en Desarrollo:** Alta
Puntos Estimados: 3	**Iteración Asignada:** 2
Descripción: El sistema mostrará los resultados correspondientes a las pruebas realizadas permitiendo que estos sean exportados a Excel.	

Figura 9 Prototipo de interfaz para la Historia de Usuario Mostrar Resultados.

2.5. Planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Esta se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción.

Es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación.

La planificación no debe ser estricta puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir. Una buena

estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para unos pocos meses (30).

2.5.1. Estimación del tiempo por las historias de usuario

Para el desarrollo del sistema propuesto se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, llegándose a las conclusiones que se muestran en la siguiente tabla.

Tabla 14 Estimación de tiempo por Historia de Usuario.

No	Historia de Usuario	Puntos de estimación
1	Autenticación	3
2	Mostrar listado de Proyectos	2
3	Adicionar un Proyecto	1
4	Modificar un Proyecto	1
5	Eliminar un Proyecto	1
6	Mostrar listado de Productos	2
7	Adicionar un Producto	1
8	Modificar un Producto	1
9	Eliminar un Producto	1
10	Realizar Pruebas	3
11	Mostar Resultados	3
12	Modificar Prueba	3
13	Eliminar Prueba	1
14	Mostrar listado de pruebas	1
15	Exportar las pruebas	1
16	Importar resultados	1
Total		**26**

El tiempo total estimado para el desarrollo del sistema es de 26 puntos de estimación, que equivalen a seis meses y dos semanas ideales de trabajo sin ningún tipo de interrupción.

2.5.2. Plan de Iteraciones

Una vez identificadas las historias de usuario del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas se procede a la planificación de la etapa

de implementación del proyecto. En base a lo antes mencionado se decide realizar esta etapa en dos iteraciones, las cuales se detallan a continuación.

Iteración 1

Esta iteración tiene como objetivo la implementación de las historias de usuario referentes a la autenticación del sistema y a la gestión de los proyectos y los productos. Durante el transcurso de la misma se crearán las funcionalidades encargadas de autenticar usuarios, adicionar, modificar y eliminar tanto proyectos como productos. Al final se contará con una primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación para el grupo de trabajo.

Iteración 2

El objetivo de esta iteración es la implementación de las historias de usuario referentes a la realización de las pruebas y mostrar los resultados de las mismas. Al finalizar se contará con una versión de prueba que contará con los procedimientos encargados de analizar los datos especificados por el usuario y emitir un resultado de las pruebas realizadas. Como resultado de ésta iteración, el sistema será puesto en funcionamiento durante un período de tiempo para evaluar su desempeño.

2.5.3. Plan de duración de las iteraciones

Como parte del ciclo de vida de un proyecto utilizando XP se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con que se cuenta. Este plan se encarga de mostrar las historias de usuario que serán abordadas en cada una de las iteraciones, así como la duración estimada de estas últimas y el orden en que se implementarán las historias de usuario.

Tabla 15 Plan de duración de las iteraciones.

Iteración	Orden de las historias de usuario a implementar	Duración de la iteración
Iteración 1	1. Autenticación	13 semanas
	2. Mostrar listado de proyectos	
	3. Adicionar un Proyecto	
	4. Modificar un Proyecto	
	5. Eliminar un Proyecto	
	6. Mostrar listado de productos	
	7. Adicionar un Producto	

	8. Modificar un Producto	
	9. Eliminar un Producto	
Iteración 2	10. Realizar Pruebas	13 semanas
	11. Mostrar Resultados	
	12. Modificar Prueba	
	13. Eliminar Prueba	
	14. Mostrar listado de pruebas	
	15. Exportar las pruebas	
	16. Importar resultados	

2.5.4. Plan de Entregas

A continuación se presenta el plan de entregas ideado para la fase de implementación. Como resultado del mismo se obtendrán versiones del sistema al finalizar cada iteración en la fecha aproximada que se indica en la siguiente tabla.

Tabla 16 Plan de entrega.

Módulo	Fin 1ra Iteración 4ta semana de Marzo	Fin 2da Iteración 2da semana de Junio
Herramienta para la gestión de pruebas no funcionales para proyectos del CEDIN	0.1	1.0

2.6. Tarjeta CRC

Las tarjetas CRC (Clase, Responsabilidad, Colaborador) son una herramienta de reflexión en el diseño de software orientado a objetos. Fueron propuestas por Ward Cunningham y Kent Beck. Se utilizan normalmente cuando primero se determinan las clases que se necesitan y cómo van a interactuar y después se implementa la solución. (31)

La técnica de las tarjetas CRC se utiliza para guiar al sistema a través de análisis basados en la responsabilidad. Las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema, y las clases con las que necesitan colaborar para completar sus responsabilidades. A continuación se muestran algunas de las tarjetas CRC que fueron confeccionadas, para consultar las restantes ver Anexo #2.

Tabla 17 Tarjeta CRC de la clase GestorPNF.

Tarjeta CRC	
Clase: GestorPNF	
Responsabilidades	**Colaboraciones**
Intermediario entre la capa de presentación y la capa de acceso a datos.	ProductoDAO ProyectoDAO PruebaDAO RNFDAO

Tabla 18 Tarjeta CRC de la clase Proyecto.

Tarjeta CRC	
Clase: ProyectoDAO	
Responsabilidades	**Colaboraciones**
Adicionar Proyecto. Modificar Proyecto. Eliminar Proyecto. Actualizar Proyecto. Obtener todos los proyectos. Obtener un proyecto dado su identificador.	JDBCUtil Proyecto

Tabla 19 Tarjeta CRC de la clase Producto.

Tarjeta CRC	
Clase: ProductoDAO	
Responsabilidades	**Colaboraciones**
Adicionar Producto. Modificar Producto. Eliminar Producto. Actualizar Producto. Eliminar los productos pertenecientes a un proyecto determinado. Obtener los productos de un proyecto determinado. Obtener todos los productos. Obtener un producto dado su identificador.	JDBCUtil Producto

Conclusiones parciales del capítulo

Quedó descrito un procedimiento para evaluar el cumplimiento de los requisitos no funcionales y se levantaron las funcionalidades que debe poseer la herramienta. Se generaron los artefactos que propone la metodología XP en la fase de exploración y planificación para el desarrollo de la herramienta, definiéndose dos iteraciones, agrupando en cada una las historias de usuario de acuerdo a su importancia, logrando así mejorar el procedimiento existente.

Capítulo 3. Implementación y prueba del sistema

En el presente capítulo se aborda la fase de Producción, donde se detallan las iteraciones llevadas a cabo durante la etapa de implementación del sistema y se describen las pruebas realizadas para comprobar su funcionamiento.

3.1. Producción

En esta fase se genera todo el código fuente necesario para satisfacer las historias de usuario definidas para la solución y se describen todas las tareas realizadas en cada iteración. Al inicio de cada historia de usuario, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable.

3.1.1. Iteraciones en el desarrollo del sistema

En esta fase se genera todo el código fuente necesario para satisfacer las historias de usuario definidas para la solución y se describen todas las tareas realizadas en cada iteración.

Una vez realizada la planificación, se determinaron dos iteraciones de desarrollo sobre el sistema, obteniendo como resultado un producto con todas las restricciones y características deseadas por el cliente para posteriormente ser utilizado. A continuación se detallan cada una de las iteraciones, mostrando las tareas más importantes para cada iteración. Para consultar el resto de las tareas, ver Anexo #3.

Iteración 1

Tabla 20 Historias de usuario abordadas en la primera iteración.

No HU	Historias de Usuarios	Tiempo de Implementación	
		Estimación	Real
1	Autenticación.	3	3
2	Mostrar listado de proyectos.	2	2
3	Adicionar un Proyecto.	1	1
4	Modificar un Proyecto.	1	1
5	Eliminar un Proyecto.	1	1
6	Mostrar listado de productos.	2	2
7	Adicionar un Producto.	1	1

8	Modificar un Producto.	1	1
9	Eliminar un Producto.	1	1

Tareas de las historias de usuario pertenecientes a la primera iteración

Tabla 21 Primera tarea asociada a la historia de usuario Autenticación.

Tarea por Historia de Usuario	
Número de la tarea: 1	**Número de historia de usuario:** 1
Nombre de la tarea: Realizar autenticación mediante el LDAP UCI	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 3/1/2014	**Fecha fin:** 10/1/2014
Descripción: El sistema realizará una búsqueda en el LDAP UCI, del usuario que intenta autenticarse, en caso de encontrarlo le permite el acceso.	

Tabla 22 Primera tarea asociada a la historia de usuario Adicionar un Proyecto.

Tarea por Historia de Usuario	
Número de la tarea: 2	**Número de historia de usuario:** 3
Nombre de la tarea: Adicionar un proyecto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10/1/2014	**Fecha fin:** 17/1/2014
Descripción: El sistema obtiene los datos del proyecto especificados por el usuario y adiciona un proyecto a la BD.	

Tabla 23 Primera tarea asociada a la historia de usuario Adicionar un Producto.

Tarea por Historia de Usuario	
Número de la tarea: 3	**Número de historia de usuario:** 7
Nombre de la tarea: Adicionar un producto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17/1/2014	**Fecha fin:** 24/1/2014
Descripción: El sistema obtiene los datos del producto especificados por el usuario y adiciona un producto a la BD.	

Iteración 2

Tabla 24 Historias de usuario abordadas en la segunda iteración.

NO HU	Historias de Usuario	Tiempo de Implementación	
		Estimación	Real

10	Realizar Pruebas.	3	3
11	Mostrar Resultados.	3	3
12	Modificar Prueba	3	3
13	Eliminar Prueba	1	1
14	Mostrar listado de pruebas	1	1
15	Exportar las pruebas	1	1
16	Importar resultados	1	1

Tareas de las historias de usuario pertenecientes a la segunda iteración

Tabla 25 Primera tarea asociada a la historia de usuario Realizar Pruebas.

Tarea por Historia de Usuario	
Número de la tarea: 4	**Número de historia de usuario:** 10
Nombre de la tarea: Obtener los datos entrados por el usuario y emitir una evaluación como resultado de la prueba.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 24/1/2014	**Fecha fin:** 14/2/2014
Descripción: El usuario contará con las opciones que le permitirán especificar para cada RNF los datos requeridos para realizar las pruebas no funcionales del producto.	

Tabla 26 Primera tarea asociada a la historia de usuario Mostrar Resultados.

Tarea por Historia de Usuario	
Número de la tarea: 5	**Número de historia de usuario:** 11
Nombre de la tarea: Obtener los resultados emitidos como parte del resultado de las pruebas y visualizarlos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 14/2/2014	**Fecha fin:** 7/3/2014
Descripción: El sistema mostrará mediante una interfaz visual los resultados de las pruebas realizadas.	

3.2. Pruebas

En la fase de producción se requieren de pruebas adicionales y revisiones de rendimiento antes de que el sistema al haber concluido las iteraciones definidas sea trasladado al entorno de producción.

Uno de los pilares fundamentales de XP es el proceso de pruebas, el cual anima a los desarrolladores a probar constantemente tanto como sea posible. Mediante esta

filosofía se reduce el número de errores no detectados así como el tiempo entre la introducción de este en el sistema y su detección. Todo esto contribuye a elevar la calidad del producto y la seguridad de los desarrolladores a la hora de introducir modificaciones.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática, y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo el resultado esperado por el cliente.

3.2.1. Pruebas unitarias

Las pruebas unitarias son efectuadas a nivel de código y la resolución de todas las incidencias asociadas a ellas da lugar a que se pueda plantear que el sistema está listo para ser revisado con el cliente.

En el transcurso de la implementación de cada historia de usuario perteneciente a una determinada iteración, se realizaron al código generado una serie de pruebas para determinar su correcto funcionamiento. Para la realización de dichas pruebas se utilizó el *framework* JUnit, implementado basándose en el patrón xUnit[7] usando lenguajes de programación orientados a objetos.

Para la utilización de JUnit se implementó una clase de prueba nombrada **`PrincipalTest`**, en la cual se definieron los casos de pruebas que validaron las funcionalidades principales de la solución desarrollada. A continuación se muestran los detalles de esta clase y los resultados finales que arrojaron las pruebas unitarias.

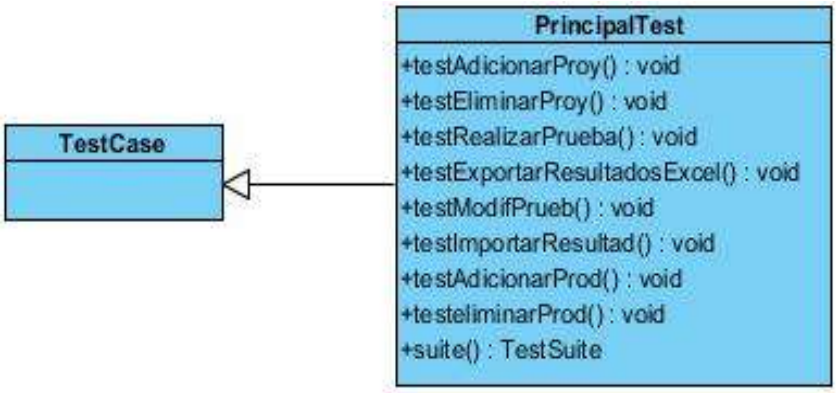

Figura 10 Diagrama de clases implementadas para JUnit

Resultados de las pruebas unitarias

[7] Se refiere a cualquier miembro de la familia de *frameworks* de pruebas automatizadas

Para un total de ocho casos de pruebas definidos, luego de ejecutar varias veces las pruebas y corregir los errores identificados, se obtienen finalmente un total de cero errores y cero fallas, como se evidencia en la Figura 9.

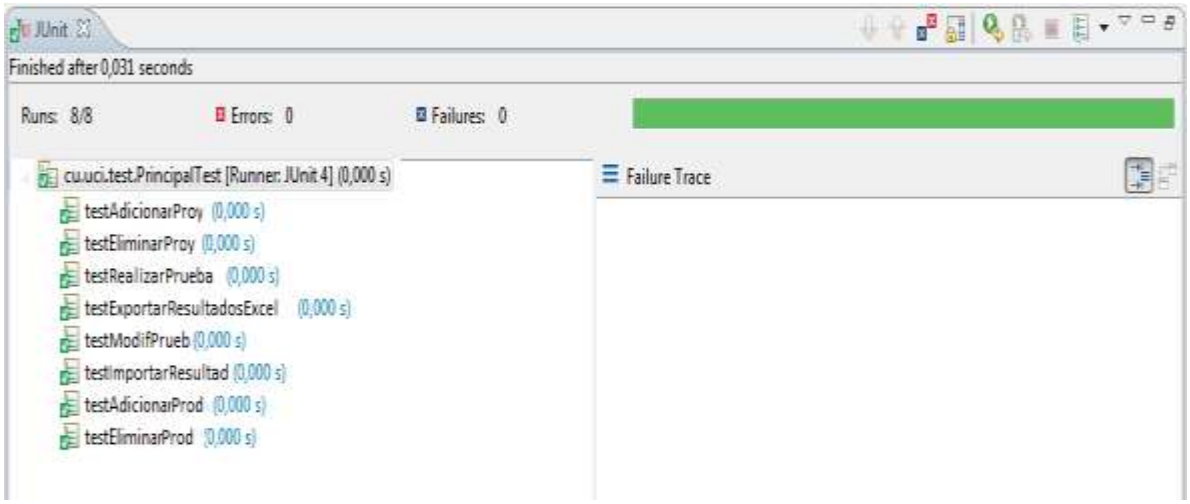

Figura 11 Resultados de los casos de prueba utilizando JUnit

La realización de las pruebas unitarias permitió que los errores fueran acotados y más fáciles de localizar, el código se documentó para posteriores análisis, y fomentaron el cambio, ya que facilitaron que el desarrollador modificara el código con el objetivo de mejorar su estructura.

3.2.2. Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las historias de usuario. Durante las iteraciones las historias de usuario seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una HU ha sido implementada correctamente. Una HU puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de estas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable.

Las pruebas de aceptación tienen más peso que las unitarias ya que constituyen un indicador de la satisfacción del cliente con la solución.

Como resultado de las pruebas de aceptación se obtendrán artefactos descritos en tablas, estas contarán con los siguientes campos:

- **Código:** servirá como identificador de la prueba realizada, a su vez será sugerente al nombre de la prueba a la que hace referencia.

- **UH**: tendrá el nombre de la historia de usuario a la que hace referencia la prueba a realizar.
- **Nombre**: nombre que se le da a la prueba a realizar.
- **Descripción**: se describe la funcionalidad que se desea probar.
- **Condiciones de Ejecución**: mostrará las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.
- **Entradas / Pasos de Ejecución**: se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Resultado esperado**: se hará una breve descripción del resultado que se espera obtener con la prueba realizada.
- **Evaluación de la prueba**: acorde al resultado de la prueba realizada se emitirá una evaluación sobre la misma. Esta evaluación tendrá uno de los tres resultados que a continuación se describen:
  - Bien: cuando el resultado de la prueba es exactamente el esperado por el usuario.
  - Parcialmente bien: cuando el resultado no es completamente el esperado por el cliente o usuario de la aplicación y muestra resultados erróneos o fuera de contexto.
  - Mal: cuando el resultado de la prueba realizada genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contexto, trayendo como consecuencia que la funcionalidad requerida por el cliente no tenga resultado, lo que invalida también la UH.

Resultados de las pruebas de aceptación

Las pruebas de aceptación se llevaron a cabo durante las dos iteraciones establecidas para la implementación del sistema. En cada iteración, entre el equipo de desarrollo y el cliente, se realizaron varias pruebas de este tipo, al principio se detectaron varias no conformidades en cuanto a la redacción de los mensajes de información al usuario, en

la validación de los datos entrados por este último, y algunas funcionalidades que no arribaron al resultado que se esperaba.

A continuación se muestran algunos de los casos de pruebas de aceptación realizados para las historias de usuario que fueron establecidas, en las cuales se evidencia el correcto funcionamiento del sistema, se decide exponer solo estas debido a que se generaron muchas pruebas en el transcurso de cada iteración, considerándose estas las más relevantes. Para consultar los casos de pruebas restantes, ver Anexo #4.

Tabla 27 Prueba de aceptación 1 sobre la historia de usuario 1

Caso de Prueba de Aceptación	
Código: P1_ HU1	**Historia de Usuario:** 1
Nombre: Autenticación de usuario.	
Descripción: Prueba para la funcionalidad autenticar usuario.	
Condiciones de Ejecución: El usuario deber ser miembro del dominio UCI.	
Entradas/Pasos de Ejecución: El usuario deberá introducir el nombre de usuario y la contraseña. En caso de no ser así mostrará un mensaje de error indicando que no debe dejar campos vacíos.	
Resultado Esperado: Si el usuario forma parte del dominio UCI podrá acceder al sistema. En caso del usuario cometer algún error en el proceso de autenticación, el sistema mostrará un mensaje con las posibles causas por las cuales falló la autenticación.	
Evaluación de la Prueba: Bien.	

Tabla 28 Prueba de aceptación 1 sobre la historia de usuario 3

Caso de Prueba de Aceptación	
Código: P1_ HU3	**Historia de Usuario:** 3
Nombre: Adicionar un proyecto.	
Descripción: Prueba para la funcionalidad de adicionar un proyecto.	
Condiciones de Ejecución: Se debe haber especificado el nombre del proyecto. El nombre del proyecto no puede estar presente en uno de los proyectos ya adicionados.	
Entradas/Pasos de Ejecución: 1- El usuario seleccionará la opción de “adicionar” y luego especificará el nombre del proyecto. 2- El usuario no especifica el nombre del proyecto.	
Resultado Esperado: 1- El sistema adiciona el proyecto y muestra un mensaje informando que este ha sido adicionado correctamente.	

2- El sistema mostrará un mensaje de error indicando que no se debe dejar campos vacíos.
Evaluación de la Prueba: Bien.

Tabla 29 Prueba de aceptación 1 sobre la historia de usuario 7

Caso de Prueba de Aceptación	
Código: P1_ HU7	**Historia de Usuario:** 7
Nombre: Adicionar un producto.	
Descripción: Prueba para la funcionalidad de adicionar un producto.	
Condiciones de Ejecución: Se debe haber especificado el nombre y la versión del producto. La versión del producto no puede estar presente en el mismo producto.	
Entradas/Pasos de Ejecución: 1- El usuario seleccionará la opción de “adicionar producto” y luego especificará el nombre y la versión del producto. 2- El usuario no especifica el nombre y la versión del producto.	
Resultado Esperado: 1- El sistema adiciona el producto y muestra un mensaje informando que este ha sido adicionado correctamente. 2- El sistema mostrará un mensaje de error indicando que no debe dejar campos vacíos.	
Evaluación de la Prueba: Bien.	

Tabla 30 Prueba de aceptación 1 sobre la historia de usuario 10

Caso de Prueba de Aceptación	
Código: P1_ HU10	**Historia de Usuario:** 10
Nombre: Realizar prueba.	
Descripción: Prueba para la funcionalidad de realizar una prueba.	
Condiciones de Ejecución: El usuario deberá seleccionar la opción ¨Nueva Prueba¨.	
Entradas/Pasos de Ejecución: 1- El usuario seleccionará un producto y los requisitos no funcionales que desea evaluar, luego deberá especificar los datos correspondientes a cada requisito seleccionado. 2- Una vez especificados todos los datos correspondientes, se mostrarán los resultados de la prueba realizada. Se seleccionará la opción ¨Finalizar¨ para guardar la prueba.	
Resultado Esperado: 1- El sistema valida que se hayan especificado correctamente los datos en caso contrario muestra un mensaje indicando que no se admiten campos vacíos. El sistema se encarga de validar que los datos se hayan introducido correctamente. Una vez finalizado salva la prueba.	

2- El sistema muestra correctamente los resultados de la prueba.
Evaluación de la Prueba: Bien.

Tabla 31 Prueba de aceptación 1 sobre la historia de usuario 11

Caso de Prueba de Aceptación	
Código: P1_ HU11	**Historia de Usuario:** 11
Nombre: Mostrar resultados.	
Descripción: Prueba para la funcionalidad de mostrar los resultados obtenidos.	
Condiciones de Ejecución: Deben haberse realizado pruebas previamente.	
Entradas/Pasos de Ejecución: El usuario deberá seleccionar la opción ¨Mostrar resultados¨.	
Resultado Esperado: El sistema muestra todas las pruebas realizadas con el resultado correspondiente a cada requisito no funcional que se evaluó al realizar la misma.	
Evaluación de la Prueba: Bien.	

Conclusiones parciales del capítulo

Se realizó la implementación y prueba del sistema. El plan de entrega fue cumplido acorde al tiempo planificado para el desarrollo de las tareas por cada historia de usuario. Se validó por parte del cliente el cumplimiento de los requisitos y se comprobó la aceptación del sistema.

Conclusiones Generales

Como resultado del presente trabajo, se puede concluir que:

- La elaboración del marco teórico de la investigación perteneciente a los principales conceptos y herramientas relacionados a las pruebas de requisitos no funcionales y su gestión, permitió la selección adecuada de las herramientas, tecnologías y metodología utilizadas en el desarrollo de la presente investigación.
- El procedimiento se organizó adecuadamente, identificando roles, requerimientos a probar, actividades a seguir para dar cumplimiento al proceso de prueba no funcionales a los productos del CEDIN con ayuda de herramientas automatizadas.
- La creación de la aplicación para la gestión de pruebas no funcionales y su posterior uso, permitió la evaluación de los requerimientos no funcionales de portabilidad, mantenimiento, fiabilidad, eficiencia, usabilidad y seguridad, de esta forma se apoyó la puesta en práctica del procedimiento propuesto en el presente trabajo.
- Con la realización de las pruebas de aceptación, se validó el cumplimiento de los requisitos y se comprobó la aceptación del sistema por parte de los clientes.

Recomendaciones

- Incluir nuevos requisitos no funcionales tales como, interfaz y apariencia, disponibilidad y soporte en el proceso de realización de las pruebas, en aras de garantizar una mayor calidad en los productos del CEDIN.

Referencias Bibliográficas

1. Carlos, V. L. MEJORES PRÁCTICAS PARA EL ESTABLECIMIENTO Y ASEGURAMIENTO DE LA CALIDAD DE SOFTWARE. 2008.

2. SimonCed. .Resumen de las Normas de Calidad: ISO/IEC 9126 y ISO/IEC 14598. [En línea] 5 de 11 de 2012. [Citado el: 27 de 11 de 2013.] http://www.Resumen de las Normas de Calidad ISO_IEC 9126 y ISO_IEC 14598_AmericanBPM.com.

3. genaro007. Iso 25000. [En línea] Mayo de 2001. [Citado el: 30 de 11 de 2013.] http://www.25000/2295307.

4. Tecnología, Comité Técnico de Normalización NC/CTN 18 de la. Norma cubana ISO/IEC17799:2005. Habana: Oficina nacional de normalización, 2007.

5. Burnstein.llene. Practical Software Testing. 2002.

6. Raymond McLeood, Jr. Software Testing, Testing Across the Entire, Software Developmen Life Cyclet. 2007.

7. Giraldo, O.P. Ingeniería de Requisitos. 2007. Volumen, 13.

8. Tipos de pruebas de software. 2011. No1.

9. S.A, Testhouse Consultores. Pruebas no Funcionales. [En línea] 2011-2012. [Citado el: 2 de 12 de 2013.] http://www.Pruebas NO Funcionales - Testhouse.htm..

10. Hilda. Procedimiento. [En línea] 26 de 11 de 2008. [Citado el: 2 de 12 de 2013.] http://deconceptos.com/general/procedimiento. [En línea] 2010. [Citado el: 5 de 12 de 2013.] http://es.wikipedia.org/wiki/Procedimiento.

11. Castellanos, Dayanis. *Procedimento de pruebas no funcionales para laboratorios virtuales.* UCI, La Habana: s.n., 2013.

12. Visual Paradigm. [En línea] [Citado el: 25 de Mayo de 2013.] http://www.visual-paradigm.com/.

13. kioskea.net. [En línea] [Citado el: 7 de Junio de 2013.] http://es.kioskea.net/contents/304-lenguajes-de-programacion.

14. Aulaclic. [En línea] [Citado el: 16 de Junio de 2013.] http://www.aulaclic.es/html/index.htm..

15. W3 Consortium. [En línea] [Citado el: 17 de Junio de 2013.] http://www.w3.org/Style/CSS/.

16. Object Management Group - UML. [En línea] [Citado el: 17 de Junio de 2013.] http://www.uml.org/.

17. Slideshare. [En línea] [Citado el: 25 de Mayo de 2013.] http://www.slideshare.net/GhaBiithahh/entornos-de-desarrollo-integrados.

18. Neil, Theresa. DEVELOP RIA. [En línea] 2011. http://www.developria.com/2011/01/the-gartner-ria-report-in.html.

19. EcuRed. [En línea] http://www.ecured.cu/index.php/Dojo_Framework.

20. Dojo. [En línea] http://dojotoolkit.org/.

21. Bibeault, Bear y Y.K. JQuery in Action. 2008.

22. Fronckowiak, John. IBM.DeveloperWorks. [En línea] 2008. http://www.ibm.com/developerworks/web/library/wa-aj-extjs/.

23. Gronroos, Marko. Book of Vaadin: 4th Edition. 2011.

24. The Apache Software Foundation. Apache Tomcat. [En línea] 1999-2014. [Citado el: 2014 de Febrero de 20.] http://tomcat.apache.org/.

25. The PostgreSQL Global Development Group. PostgreSQL. [En línea] 1996-2014. [Citado el: 2014 de Febrero de 24.] http://www.postgresql.org/.

26. The PostgreSQL Global Development Group. PostgreSQL JDBC Driver. [En línea] 1996-2013. [Citado el: 2014 de Febrero de 24.] http://jdbc.postgresql.org/.

27. GitHub, Inc. JUnit. [En línea] [Citado el: 2014 de Febrero de 24.] http://junit.org/.

28. José H. Canós, Patricio Letelier y Mª Carmen Penadés. *Métodologías Ágiles en el Desarrollo de Software. .* Valencia: s.n.

29. Sánchez, María A. Mendoza. Metodologías De Desarrollo De Software. 2004.

30. Kent Beck, F. Planeando en Programación Extrema. 2000.

31. LSI Departamento de Lenguajes y Sistemas Informáticos. [En línea] [Citado el: 25 de Mayo de 2013.] http://lsi.ugr.es/~ig1/docis/crc.pdf.

Anexos

Anexo #1: Historias de usuario

Anexo 1 Historia de Usuario Modificar un Proyecto

Historia de Usuario	
Número: 4	**Nombre:** Modificar un Proyecto
Prioridad en Negocio: Baja	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 1
Descripción: El sistema le permitirá al usuario modificar un proyecto seleccionado, admitiéndole modificar el nombre y los atributos del mismo.	

Figura 12 Prototipo de interfaz para la Historia de Usuario Modificar un Proyecto.

Anexo 2 Historia de Usuario Eliminar un Proyecto

Historia de Usuario	
Número: 5	**Nombre:** Eliminar un Proyecto
Prioridad en Negocio: Baja	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 1
Descripción: El sistema le permitirá al usuario eliminar un proyecto seleccionado.	

Figura 13 Prototipo de interfaz para la Historia de Usuario Eliminar un Proyecto.

Anexo 3 Historia de Usuario Modificar un Producto

Historia de Usuario

Número: 8	**Nombre:** Modificar un producto
Prioridad en Negocio: Baja	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 1
Descripción: El sistema le permitirá al usuario modificar un producto seleccionado, admitiéndole modificar el nombre y los atributos del mismo.	

Figura 14 Prototipo de interfaz para la Historia de Usuario Modificar un Producto.

Anexo 4 Historia de Usuario Eliminar un Producto

Historia de Usuario	
Número: 9	**Nombre:** Eliminar un Producto
Prioridad en Negocio: Baja	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 1
Descripción: El sistema le permitirá al usuario eliminar un producto seleccionado.	

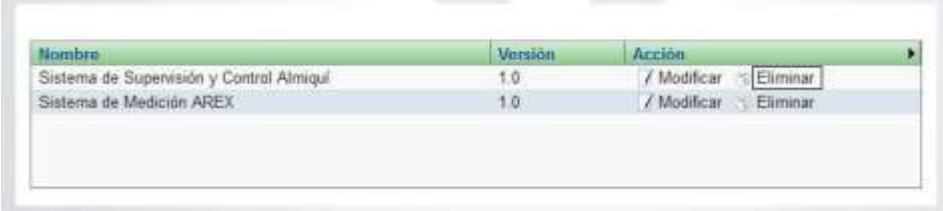

Figura 15 Prototipo de interfaz para la Historia de Usuario Eliminar un Producto.

Anexo 5 Historia de Usuario Modificar Prueba

Historia de Usuario	
Número: 12	**Nombre:** Modificar Prueba
Prioridad en Negocio: Alta	**Complejidad en Desarrollo:** Alta
Puntos Estimados: 3	**Iteración Asignada:** 2
Descripción: El sistema le permitirá al usuario modificar una prueba seleccionada.	

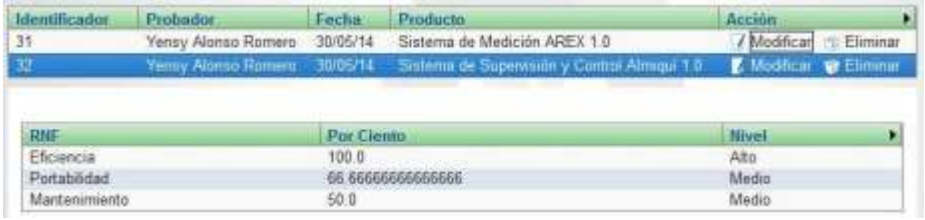

Figura 16 Prototipo de interfaz para la Historia de Usuario Modificar Prueba.

Anexo 6 Historia de Usuario Eliminar Prueba

Historia de Usuario	
Número: 13	**Nombre:** Eliminar Prueba
Prioridad en Negocio: Baja	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 2
Descripción: El sistema le permitirá al usuario eliminar una prueba seleccionada.	

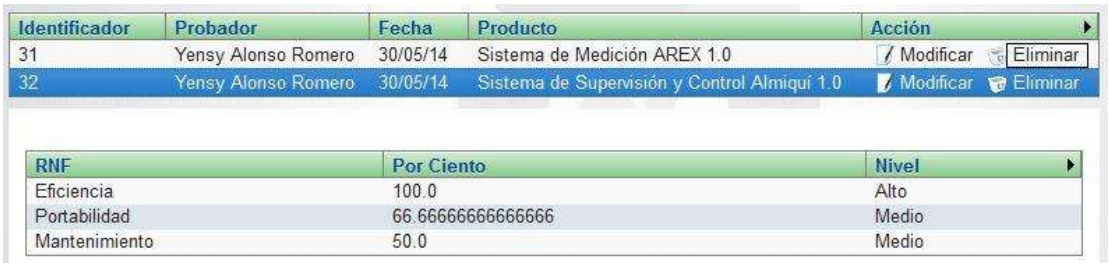

Figura 17 Prototipo de interfaz para la Historia de Usuario Eliminar Prueba.

Anexo 7 Historia de Usuario Mostrar listado de pruebas

Historia de Usuario	
Número: 14	**Nombre:** Mostrar listado de pruebas
Prioridad en Negocio: Media	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 2
Descripción: El sistema mostrará en una tabla todas las pruebas realizadas por el usuario, incluyendo las opciones de modificar y eliminar una prueba.	

Figura 18 Prototipo de interfaz para la Historia de Mostrar listado de pruebas.

Anexo 8 Historia de Usuario Exportar las pruebas

Historia de Usuario	
Número: 15	**Nombre:** Exportar las pruebas
Prioridad en Negocio: Media	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 2
Descripción: El sistema permitirá exportar a Excel las pruebas realizadas.	

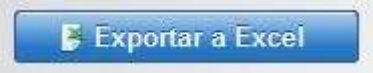

Figura 19 Prototipo de interfaz para la Historia de Exportar las pruebas.

Anexo 9 Historia de Usuario Importar resultados

Historia de Usuario	
Número: 16	**Nombre:** Importar resultados
Prioridad en Negocio: Media	**Complejidad en Desarrollo:** Baja
Puntos Estimados: 1	**Iteración Asignada:** 2
Descripción: El sistema permitirá importar desde Excel los resultados obtenidos de las pruebas.	

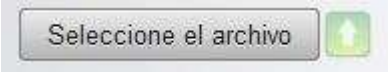

Figura 20 Prototipo de interfaz para la historia de Importar resultados.

Anexo #2: Tarjetas CRC

Anexo 10 Tarjeta CRC de la clase Ldap

Tarjeta CRC	
Clase: Ldap	
Responsabilidades	**Colaboraciones**
Autenticación del Sistema.	Usuario

Verificar la existencia de un usuario en el dominio. Obtener el usuario autenticado.	

Anexo 11 Tarjeta CRC de la clase Usuario

Tarjeta CRC	
Clase: Usuario	
Responsabilidades	**Colaboraciones**
Brindar los atributos del usuario para que puedan ser utilizados posteriormente por otras clases.	

Anexo 12 Tarjeta CRC de la clase RNF

Tarjeta CRC	
Clase: RNF **Subclases:** Eficiencia, Fiabilidad, Mantenimiento, Portabilidad, Seguridad y Usabilidad.	
Responsabilidades	**Colaboraciones**
Obtener el porciento calculado luego de haber aplicado la métrica correspondiente al tipo de requisito. Obtener el nivel deducido a partir del por ciento.	

Anexo 13 Tarjeta CRC de la clase Eficiencia

Tarjeta CRC	
Clase: Eficiencia **Súper clase**: RNF	
Responsabilidades	**Colaboraciones**
Realizar el procedimiento para evaluar el cumplimiento de la eficiencia haciendo uso de las métricas establecidas para medir este tipo de requisito.	

Anexo 14 Tarjeta CRC de la clase Fiabilidad

Tarjeta CRC
Clase: Fiabilidad **Súper clase**: RNF

Responsabilidades	Colaboraciones
Realizar el procedimiento para evaluar el cumplimiento de la fiabilidad haciendo uso de las métricas establecidas para medir este tipo de requisito.	

Anexo 15 Tarjeta CRC de la clase Mantenimiento

Tarjeta CRC	
Clase: Mantenimiento **Súper clase**: RNF	
Responsabilidades	**Colaboraciones**
Realizar el procedimiento para evaluar el cumplimiento del mantenimiento haciendo uso de las métricas establecidas para medir este tipo de requisito.	

Anexo 16 Tarjeta CRC de la clase Portabilidad

Tarjeta CRC	
Clase: Portabilidad **Súper clase**: RNF	
Responsabilidades	**Colaboraciones**
Realizar el procedimiento para evaluar el cumplimiento de la portabilidad haciendo uso de las métricas establecidas para medir este tipo de requisito.	

Anexo 17 Tarjeta CRC de la clase Usabilidad

Tarjeta CRC	
Clase: Usabilidad **Súper clase**: RNF	
Responsabilidades	**Colaboraciones**
Realizar el procedimiento para evaluar el cumplimiento de la usabilidad haciendo uso de las métricas establecidas para medir este tipo de requisito.	

Anexo 18 Tarjeta CRC de la clase Seguridad

Tarjeta CRC	
Clase: Seguridad **Súper clase**: RNF	
Responsabilidades	**Colaboraciones**
Realizar el procedimiento para evaluar el cumplimiento de la seguridad haciendo uso de las métricas establecidas para medir este tipo de requisito.	

Anexo 19 Tarjeta CRC de la clase Producto

Tarjeta CRC	
Clase: Producto	
Responsabilidades	**Colaboraciones**
Brindar los atributos pertenecientes a un proyecto para que puedan ser utilizados posteriormente por otras clases.	

Anexo 20 Tarjeta CRC de la clase Proyecto

Tarjeta CRC	
Clase: Proyecto	
Responsabilidades	**Colaboraciones**
Brindar los atributos pertenecientes a un producto para que puedan ser utilizados posteriormente por otras clases.	

Anexo 21 Tarjeta CRC de la clase JDBCUtil

Tarjeta CRC	
Clase: JDBCUtil	
Responsabilidades	**Colaboraciones**
Establece la conexión con la base de datos.	

Anexo 22 Tarjeta CRC de la clase Prueba

Tarjeta CRC	
Clase: Prueba	
Responsabilidades	**Colaboraciones**
Brindar los atributos pertenecientes a una	

prueba para que puedan ser utilizados posteriormente por otras clases.	

Anexo 23 Tarjeta CRC de la clase Prueba DAO

Tarjeta CRC	
Clase: PruebaDAO	
Responsabilidades	**Colaboraciones**
Adicionar Prueba. Modificar Prueba. Eliminar Prueba. Actualizar Prueba. Obtener todas las pruebas. Obtener una prueba dado su identificador. Obtener todas las pruebas de un probador determinado.	JDBCUtil Prueba RNFDAO

Anexo 24 Tarjeta CRC de la clase RNFDAO

Tarjeta CRC	
Clase: RNFDAO	
Responsabilidades	**Colaboraciones**
Adicionar RNF. Modificar RNF. Eliminar RNF. Actualizar RNF. Obtener todos los RNF. Obtener todos los RNF de una prueba determinada. Obtener un RNF dado su identificador.	JDBCUtil RNF

Anexo #3: Tareas de las historias de usuario

Anexo 25 Primera tarea asociada a la historia de usuario Mostrar el listado de proyectos

Tarea por Historia de Usuario	
Número de la tarea: 6	**Número de historia de usuario:** 2
Nombre de la tarea: Mostrar los proyectos añadidos por el usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2

Fecha inicio: 7/3/2014	**Fecha fin:** 14/3/2014
Descripción: El sistema mostrará mediante una interfaz visual los proyectos añadidos.	

Anexo 26 Primera tarea asociada a la historia de usuario Mostrar el listado de productos

Tarea por Historia de Usuario	
Número de la tarea: 7	**Número de historia de usuario:** 6
Nombre de la tarea: Mostrar los productos añadidos por el usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 14/3/2014	**Fecha fin:** 21/3/2014
Descripción: El sistema mostrará mediante una interfaz visual los proyectos añadidos.	

Anexo 27 Primera tarea asociada a la historia de usuario Modificar un Proyecto

Tarea por Historia de Usuario	
Número de la tarea: 8	**Número de historia de usuario:** 4
Nombre de la tarea: Modificar un proyecto que decida el usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 21/3/2014	**Fecha fin:** 28/3/2014
Descripción: El usuario tendrá la opción que le permita editar un proyecto seleccionado, especificando el nuevo nombre.	

Anexo 28 Primera tarea asociada a la historia de usuario Modificar un Producto

Tarea por Historia de Usuario	
Número de la tarea: 9	**Número de historia de usuario:** 8
Nombre de la tarea: Modificar un producto que decida el usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 28/3/2014	**Fecha fin:** 4/4/2014
Descripción: El usuario tendrá la opción que le permita editar un producto seleccionado, especificando el nuevo nombre y su versión.	

Anexo 29 Primera tarea asociada a la historia de usuario Eliminar un Proyecto

Tarea por Historia de Usuario	
Número de la tarea: 10	**Número de historia de usuario:** 5
Nombre de la tarea: Eliminar un proyecto que decida el usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 4/4/2014	**Fecha fin:** 11/4/2014

Descripción: El usuario tendrá la opción que le permita seleccionar el proyecto que desea eliminar de la BD.

Anexo 30 Primera tarea asociada a la historia de usuario Eliminar un Producto

Tarea por Historia de Usuario	
Número de la tarea: 11	**Número de historia de usuario:** 9
Nombre de la tarea: Eliminar un producto que decida el usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11/4/2014	**Fecha fin:** 18/4/2014
Descripción: El usuario tendrá la opción que le permita seleccionar el producto que desea eliminar de la BD.	

Anexo 31 Primera tarea asociada a la historia de usuario Modificar Prueba

Tarea por Historia de Usuario	
Número de la tarea: 12	**Número de historia de usuario:** 12
Nombre de la tarea: Modificar una prueba.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 18/4/2014	**Fecha fin:** 9/5/2014
Descripción: El usuario tendrá la opción que le permita modificar una prueba.	

Anexo 32 Primera tarea asociada a la historia de usuario Eliminar Prueba

Tarea por Historia de Usuario	
Número de la tarea: 13	**Número de historia de usuario:** 13
Nombre de la tarea: Eliminar una prueba.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 9/5/2014	**Fecha fin:** 16/5/2014
Descripción: El usuario tendrá la opción que le permita eliminar una prueba.	

Anexo 33 Primera tarea asociada a la historia de usuario Mostrar listado de pruebas

Tarea por Historia de Usuario	
Número de la tarea: 14	**Número de historia de usuario:** 14
Nombre de la tarea: Mostrar el listado de pruebas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16/5/2014	**Fecha fin:** 23/5/2014

Descripción: El usuario tendrá la opción que le permita visualizar todas las pruebas realizadas.

Anexo 34 Primera tarea asociada a la historia de usuario Exportar las pruebas

Tarea por Historia de Usuario	
Número de la tarea: 15	**Número de historia de usuario:** 15
Nombre de la tarea: Exportar las pruebas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 23/5/2014	**Fecha fin:** 30/5/2014
Descripción: El usuario tendrá la opción que le permita exportar todas las pruebas realizadas en un documento Excel.	

Anexo 35 Primera tarea asociada a la historia de usuario Importar las pruebas

Tarea por Historia de Usuario	
Número de la tarea: 16	**Número de historia de usuario:** 16
Nombre de la tarea: Importar las pruebas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 30/5/2014	**Fecha fin:** 6/6/2014
Descripción: El usuario tendrá la opción que le permita importar todas las pruebas realizadas desde un documento Excel.	

Anexo #4: Casos de Prueba de Aceptación

Anexo 36 Prueba de aceptación 1 sobre la historia de usuario 8

Caso de Prueba de Aceptación	
Código: P1_ HU8	**Historia de Usuario:** 8
Nombre: Modificar un producto.	
Descripción: Prueba para la funcionalidad de modificar un producto.	
Condiciones de Ejecución: Se debe haber especificado los nuevos atributos del producto.	
Entradas/Pasos de Ejecución: 1- El usuario seleccionará la opción de “modificar” y luego especificará el nombre y la versión del producto, o solo aquel que desee modificar. 2- El usuario no especifica el nombre y la versión del producto.	
Resultado Esperado: 1- El sistema modifica el producto y muestra un mensaje informando que este ha sido modificado correctamente. 2- El sistema mostrará un mensaje de error indicando que no se debe dejar campos vacíos.	

Evaluación de la Prueba: Bien.

Anexo 37 Prueba de aceptación 1 sobre la historia de usuario 9

Caso de Prueba de Aceptación	
Código: P1_ HU9	**Historia de Usuario:** 9
Nombre: Eliminar un producto.	
Descripción: Prueba para la funcionalidad de eliminar un producto.	
Condiciones de Ejecución: El sistema debe mostrar un listado con todos los productos que han sido adicionados, incluyendo la opción de eliminar el producto.	
Entradas/Pasos de Ejecución: El usuario selecciona la opción de “eliminar” el producto.	
Resultado Esperado: El sistema elimina el producto y muestra un mensaje informando que este ha sido eliminado correctamente.	
Evaluación de la Prueba: Bien.	

Anexo 38 Prueba de aceptación 1 sobre la historia de usuario 4

Caso de Prueba de Aceptación	
Código: P1_ HU4	**Historia de Usuario:** 4
Nombre: Modificar un proyecto.	
Descripción: Prueba para la funcionalidad de modificar un proyecto.	
Condiciones de Ejecución: Se debe haber especificado el nuevo nombre del proyecto. El mismo no puede estar presente en uno de los proyectos ya adicionados.	
Entradas/Pasos de Ejecución: 1- El usuario seleccionará la opción de “modificar” y luego especificará el nuevo nombre del proyecto. En caso de no ser así mostrará un mensaje de error indicando que debe llenar los campos vacíos. 2- El sistema mostrará un mensaje de error indicando que no se debe dejar campos vacíos.	
Resultado Esperado: 1- El sistema modifica el proyecto y muestra un mensaje informando que este ha sido modificado correctamente. 2- El sistema mostrará un mensaje de error indicando que no se debe dejar campos vacíos.	
Evaluación de la Prueba: Bien.	

Anexo 39 Prueba de aceptación 1 sobre la historia de usuario 5

Caso de Prueba de Aceptación	
Código: P1_ HU5	**Historia de Usuario:** 5
Nombre: Eliminar un proyecto.	

Descripción: Prueba para la funcionalidad de eliminar un proyecto.
Condiciones de Ejecución: El sistema debe mostrar un listado con todos los proyectos que han sido adicionados, incluyendo la opción de eliminar proyecto.
Entradas/Pasos de Ejecución: El usuario seleccionará la opción de “eliminar”.
Resultado Esperado: El sistema elimina el proyecto y muestra un mensaje informando que este ha sido eliminado correctamente.
Evaluación de la Prueba: Bien.

Anexo 40 Prueba de aceptación 1 sobre la historia de usuario 6

Caso de Prueba de Aceptación	
Código: P1_ HU6	**Historia de Usuario:** 6
Nombre: Mostrar los productos que han sido adicionados.	
Descripción: Prueba para la funcionalidad de mostrar el listado de los productos.	
Condiciones de Ejecución: Deben existir productos adicionados.	
Entradas/Pasos de Ejecución: El usuario deberá seleccionar la opción ¨Listar productos ¨.	
Resultado Esperado: El sistema muestra los productos adicionados por el usuario.	
Evaluación de la Prueba: Bien.	

Anexo 41 Prueba de aceptación 1 sobre la historia de usuario 2

Caso de Prueba de Aceptación	
Código: P1_ HU2	**Historia de Usuario:** 2
Nombre: Mostrar los proyectos que han sido adicionados.	
Descripción: Prueba para la funcionalidad de mostrar el listado de los proyectos.	
Condiciones de Ejecución: Deben existir proyectos adicionados.	
Entradas/Pasos de Ejecución: El usuario deberá seleccionar la opción ¨Listar proyectos ¨.	
Resultado Esperado: El sistema muestra los proyectos adicionados por el usuario.	
Evaluación de la Prueba: Bien.	