Universidad de las Ciencias Informáticas

Propuesta de herramientas para la automatización de las pruebas de software en el CISED

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Rachel Hernández Pérez

Yuniel Martínez González

Tutores:

Ing. Dolennis Concepción Hidalgo

Ing. Noichel Juan Hernández

La Habana, Cuba

Junio de 2011

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo titulado: “Propuesta de herramientas para la automatización de las pruebas de software en el CISED”, y autorizamos al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____ días del mes de junio del año 2011.

Rachel Hernández Pérez

_____
Firma del Autor

Yuniel Martínez González

_____
Firma del Autor

Ing. Dolennis Concepción Hidalgo

_____
Firma del Tutor

Ing. Noichel Juan Hernández

_____
Firma del Tutor

Dedicatoria:

De Rachel:
A mi mami, papi y tatico por darme tanto amor y guiarme siempre por el buen camino.
A mi familia, por darle tanto drama, acción y comedia a mi vida.

De Yuniel:
A mami, Nino y Yaikel, los que siempre creyeron en mí.
A mí mismo, la persona más fuerte que he conocido.

Gracias a:

Dolennis, porque toda la gloria del mundo cabe en un grano de maíz.
Noichel y Yeneidys, por hacerlo parecer todo tan simple.
La High Life, los amigos que nos han acompañado en el caos, en la gloria y en el guizazo.
Irina, Jenny y Osmín, porque la perfección es una prueba de software de n iteraciones.

De Yuniel:
A toda mi familia, por crear una estrella.
A Rachel, por ser la hermana que nunca tuve.
A Ivan, por tomarlo tan en serio.
A los que vieron algo más que un nerd.

.De Rachel:
A mi mamá, mi papá y mi tatico, por apoyarme y guiarme. Ustedes han sido mi sostén, los autores de la persona que soy hoy y los que me han llevado hasta la orilla.

A mi tía Angelita mi segunda madre, por entregar su amor y tiempo sin esperar nada a cambio.
A Ivan, mi puchito, porque tu amor es consuelo en la tristeza, serenidad en el tumulto, reposo en la fatiga y esperanza en la desesperación.
A Yuniel, por ser mi amigo más gruñón y sincero.
A la familia Pérez Arencibia, por quererme como a una hija.

RESUMEN

En el Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) existen numerosos proyectos en los cuales se aplican pruebas de software para garantizar la calidad de los productos. El objetivo de este trabajo es definir una propuesta de herramientas que permita automatizar las pruebas de software en el CISED. Para ello se realizó un estudio sobre el Proceso de Desarrollo de Software (PDS) en la entidad, dentro del cual se hizo énfasis en las plataformas, lenguajes de programación, sistemas operativos, pruebas y otros elementos necesarios para eliminar numerosas dificultades existentes en el proceso de aseguramiento de la calidad.

Conjuntamente con la propuesta, y a partir de un estudio realizado, en este documento se describe el proceso de desarrollo de una aplicación web que permite gestionar la información concerniente a herramientas para pruebas de software. Se especifican los artefactos construidos según plantea la metodología Programación Extrema (XP), usada para guiar el desarrollo del sitio web; también se detallan las funcionalidades que debe brindar el sistema a los desarrolladores y personal de aseguramiento de la calidad.

Palabras claves: automatizar, calidad, herramientas, pruebas, software

ÍNDICE

INTRODUCCIÓN

El vertiginoso desarrollo de la informática y las comunicaciones en las últimas cuatro décadas, ha propiciado que la industria mundial del software sea increíblemente competitiva. Hoy, un producto no es bueno solo porque solucione un problema, sino porque es único, necesario y se distingue por su calidad.

El Estado cubano, consciente de la importancia de la informatización de la sociedad, ha creado en los últimos años una serie de empresas e instituciones educacionales que han convertido a la Isla en un reconocido mercado de software dentro de América Latina. Entre las universidades de Cuba, hay una que se destaca por ser la mayor fuente de ingenieros en Ciencias Informáticas de la nación, donde estudiantes y profesores juegan un papel fundamental en el desarrollo de aplicaciones para empresas tanto nacionales como foráneas. La Universidad de las Ciencias Informáticas (UCI), creada en 2002 por idea del Comandante en Jefe Fidel Castro Ruz, se erige como una futurista y necesaria institución científica.

Dentro de la UCI, los múltiples proyectos que se desarrollan han sido organizados en centros; uno de ellos es el Centro de Identificación y Seguridad Digital (CISED), donde actualmente se desarrollan soluciones con diferentes metodologías de desarrollo de software, los cuales cumplen diversas funciones relacionadas con los sistemas de identificación y seguridad de varias naciones.

Dentro del CISED existe el Grupo de Calidad de Software, encargado de las revisiones de los proyectos del Centro, así como de la realización de auditorías, pruebas y prestación de asesoramiento en materia de calidad.

Situación problémica:

Actualmente todo el proceso de aseguramiento de la calidad en el Centro se realiza manualmente a través de documentos de texto y hojas de cálculo. Las pruebas al código no se efectúan generalmente y se utilizan pocas herramientas para facilitar las pruebas y revisiones, lo que provoca la aparición de dificultades, entre las que se encuentran: el empleo de un mayor tiempo y esfuerzo por el personal de los diferentes proyectos, presencia de errores en el desarrollo del software y omisión de revisiones de la implementación.

El personal de aseguramiento de la calidad tiene poco o ningún conocimiento sobre la mayoría de las herramientas que permiten la automatización de las actividades de pruebas. Tanto en el CISED como en el resto de los centros de desarrollo de la UCI se dispone de poca documentación sobre el tema. Además, no existe una aplicación que permita la gestión de las herramientas para pruebas de software y la información concerniente a estas.

Al no utilizar herramientas automatizadas para la realización de las pruebas, se emplea un mayor tiempo en la realización de actividades asociadas a estas, se realizan menos iteraciones antes de ser liberado el software, por lo cual a veces se incumplen los plazos de entrega, no son consultadas las buenas prácticas asociadas a cada una de estas herramientas para pruebas y existen dificultades en la generación de registros con los errores encontrados en el producto.

Para la concepción de este trabajo, se hace necesario definir como **problema científico de la investigación:** ¿cómo lograr una mayor agilidad en la realización de las pruebas de software?, teniendo como **objeto de estudio:** los procesos de Gestión de la Calidad del Software (GCS).

Para dar solución a la situación problémica descrita anteriormente, se plantea como **objetivo de la investigación:** definir herramientas que permitan automatizar las pruebas de software en el CISED.

El **campo de acción** queda definido como: el proceso de pruebas automatizadas de software en el CISED.

Para dar cumplimiento al objetivo de la investigación se derivan los **objetivos específicos:**

- Describir el estado del arte de la utilización de herramientas para la realización de las pruebas de software.
- Caracterizar el Proceso de Desarrollo de Software en el CISED.
- Definir la propuesta de herramientas a utilizar para la realización de pruebas de software adaptada a las características de la GCS en el CISED.
- Desarrollar una aplicación que permita la gestión de herramientas para pruebas de software y la información concerniente a estas.

La **idea a defender** queda trazada de la siguiente manera: la utilización de herramientas para pruebas de software facilitará la realización de las mismas en el Centro de Identificación y Seguridad Digital.

Para dar cumplimiento a los objetivos planteados, se establecen las **tareas de la investigación** que se relacionan a continuación:

- Estudiar los procesos de GCS.
- Estudiar las características del Proceso de Desarrollo del Software (PDS) en el CISED.
- Determinar herramientas utilizadas en el PDS para la gestión de calidad.
- Estudiar las herramientas para pruebas usadas en la UCI.
- Proponer herramientas para automatizar las pruebas de software.
- Ofrecer una guía para utilizar las herramientas propuestas.
- Realizar la planificación de una aplicación para gestionar las herramientas.
- Construir los artefactos que dicta la metodología de desarrollo de software que se seleccione.
- Implementar la aplicación.
- Probar la aplicación.

Estrategia de la investigación

Seguidamente se relacionan los **métodos de la investigación** que serán utilizados para el desarrollo del trabajo de diploma.

Métodos teóricos:

Analítico-sintético: se consultará la bibliografía necesaria para dar cumplimiento a las tareas de la investigación y se realizará un resumen de los principales aspectos de cada una de ellas.

Análisis histórico-lógico: se analizará el proceso de GCS en el período de existencia del CISED, para conocer las diferentes actividades que se realizan y cómo se lleva a cabo su desarrollo.

Modelación: este método permite reproducir situaciones de la realidad de una manera más simple; se utilizará para la realización de los modelos o diagramas durante el desarrollo de la aplicación de gestión de herramientas para pruebas.

Métodos empíricos:

Observación: permitirá analizar todas las actividades de GCS y la utilización de herramientas que existen para su automatización.

Encuesta: durante la investigación se realizarán encuestas para determinar las tecnologías, metodologías de desarrollo de software, arquitecturas, herramientas para pruebas de software utilizadas y otras características del PDS en el CISED.

Entrevista: se realizarán a especialistas que estén relacionados con la GCS en el Centro y que utilicen o hayan utilizado herramientas relacionadas con la calidad.

Medición: el empleo de métricas para productos informáticos jugará un papel fundamental en la obtención de resultados cuantitativos que demostrarán las ventajas de usar herramientas para pruebas. Se medirán aspectos tales como: los tiempos de respuesta de las aplicaciones, la cantidad de líneas de código revisadas, el tiempo que se puede ahorrar usando dichas herramientas, entre otros.

Resultados esperados:

Con la elaboración de este trabajo de diploma se esperan los siguientes resultados: serán automatizadas las pruebas de software en el CISED, se detectarán errores en las aplicaciones de una manera más rápida desde las primeras fases de desarrollo del software, se podrá disminuir la probabilidad de ocurrencia de errores en los productos del Centro, se dispondrá de una aplicación para gestionar las distintas herramientas y se elevará la calidad de los productos de software en el CISED.

Estructuración del contenido y una breve explicación de sus partes:

Capítulo I: Fundamentación teórica.

Se expondrán una serie de conceptos y clasificaciones asociadas a la calidad y específicamente a las evaluaciones de software. Se describirán distintas aplicaciones para pruebas de software existentes en el mundo y su aplicación en la universidad. Por último, se mencionarán las tecnologías y herramientas seleccionadas para desarrollar una aplicación web que permita la gestión de herramientas para pruebas de software.

Capítulo II: Propuesta de herramientas para pruebas de software.

Se analizarán las diferentes características del Proceso de Desarrollo de Software en el CISED, para definir, de acuerdo con las pruebas que se realicen y las tecnologías que sean utilizadas, un conjunto de herramientas que permitan automatizar las pruebas de software del Centro.

Capítulo III: Sistema de gestión de herramientas para pruebas de software.

Se realizará el diseño e implementación de una aplicación web que permita la gestión de herramientas para pruebas de software. Se probará la aplicación y se registrarán los resultados.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

La industria de software ha ocupado un lugar prominente dentro del mercado internacional, donde los productos de software ofertados deben poseer la calidad como cualidad inherente, factor determinante en el éxito de un producto.

Si alguien dice que la calidad no le da ventaja a una organización, entonces se equivoca. Cualquier institución que deja de lado este importante concepto en el desarrollo del ciclo de vida del software, puede destruir su propia imagen y el valor de la empresa. Por esta razón, definir y aplicar un plan de prueba de software al producto que se desarrolla, es una actividad de gran importancia.

En el presente capítulo serán conceptualizados los elementos claves de la gestión de la calidad, así como las principales herramientas de pruebas existentes con el objetivo de realizar, posteriormente, una propuesta factible para la organización y desarrollar una aplicación que permita la gestión de toda la información asociada a las mismas.

1.1 Calidad del software

La rápida evolución de los productos y la aparición de sistemas han favorecido la existencia de un escenario de alto grado de intensidad competitiva entre las empresas productoras de software. Actualmente, los usuarios reclaman productos y servicios informáticos con mayor calidad, sustentados en requerimientos desafiantes y mayores necesidades.

La calidad del software ha seguido una trayectoria de constante mejora, debido a la importancia que tiene su gestión y la utilización de técnicas para garantizar la misma en el desarrollo de software y colocarse entre los primeros en un mercado asiduamente competitivo. Calidad es “el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”. (The Institute of Electrical and Electronics Engineers, Inc., 1990)

La Organización Internacional de Estándares (ISO) define la calidad del software como: “la totalidad de características de un producto de software que le confiere la capacidad de satisfacer necesidades explícitas e implícitas”. (Cueva Lovelle, 2009)

Existen múltiples definiciones sobre este término, dados a conocer por otras importantes organizaciones y personalidades de la Ingeniería de Software; en resumen, cada uno de ellos establece que un producto dispondrá de la calidad del software necesaria, si son satisfechos el conjunto de requisitos establecidos por el cliente, implicando una mejora en productividad y competitividad.

La calidad del software puede medirse después de elaborado el producto, pero puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

Gestión de la Calidad del Software

Debe entenderse por gestión de la calidad al “modo en que la dirección planifica el futuro, implanta los programas y controla los resultados de la función calidad con vistas a su mejora permanente”. (Udaondo Durán, 2008)

Según la norma ISO 9001:2008 de la Organización Internacional para la Estandarización, la gestión de la calidad del software es el conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implanta por medios tales como la planificación de la calidad, el control de la calidad, el aseguramiento (garantía) de la calidad y la mejora de la calidad, en el marco del sistema de calidad. (Iberoamerican Journal of Project Management, 2011)

Lograr la implementación de la gestión de la calidad no es un objetivo sencillo, es todo un desafío profesional, pero vale la pena intentar alcanzarlo si se desea desarrollar productos con una notable calidad, además de dotar a las instituciones de ventajas tales como:

- Mejorar continuamente la productividad y la competitividad.
- Dar al cliente lo que desea.
- No hacer más que lo necesario.

- Involucrar todos los niveles de la empresa.

Procesos de Gestión de la Calidad del Software

Los procesos de gestión de la calidad del software incluyen todas las actividades que determinan las políticas, los objetivos y las responsabilidades relativas a la calidad, de modo que el proyecto satisfaga las necesidades por las cuales se emprendió. Implementan el sistema de gestión de la calidad a través de la política, los procedimientos y los procesos de planificación de calidad, aseguramiento de calidad y control de calidad, con actividades de mejora continua de los procesos que se realizan durante todo el proyecto, según corresponda.

Los procesos de gestión de la calidad del software son:

Planificación de calidad: son identificadas las normas de calidad relevantes para el proyecto y se determina cómo satisfacerlas.

Realizar aseguramiento de calidad: se aplican las actividades planificadas y sistemáticas relativas a la calidad, para asegurar que el proyecto utilice todos los procesos necesarios para cumplir con los requisitos.

Realizar control de calidad: son supervisados los resultados específicos del proyecto, para determinar si cumplen con las normas de calidad relevantes e identificar modos de eliminar las causas de un rendimiento insatisfactorio.

Estos procesos interaccionan entre sí y cada proceso puede implicar el esfuerzo de una o más personas o grupos de personas, dependiendo de las necesidades del proyecto. Cada proceso tiene lugar por lo menos una vez en cada proyecto y se realiza en una o más fases del mismo (en caso de que se encuentre dividido por fases). (Project Management Institute, 2004)

1.2 Evaluación del software

La construcción de sistemas de software es una actividad de gran dificultad, por lo que la probabilidad de presencia de errores en estas aplicaciones, aún después de la entrega de un producto informático al cliente, no es remota. Lamentablemente, las capacidades para realizar una medición de la fiabilidad de un

sistema informático son pobres frente a las capacidades en otros campos de la ingeniería, donde con expresiones matemáticas, los ingenieros pueden predecir los comportamientos de sus creaciones.

La evaluación de software, es un proceso que tiene como finalidad determinar el grado de eficacia y eficiencia del producto de software con que han sido empleados los recursos destinados a alcanzar los objetivos previstos, lo que posibilita la determinación de las desviaciones y la adopción de medidas correctivas que garanticen el cumplimiento adecuado de las metas presupuestadas.

Tipos de evaluaciones del software

En términos generales, se pueden distinguir dos tipos de evaluaciones durante el proceso de desarrollo: verificaciones y validaciones. El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) las definen como:

- **Verificación:** proceso para determinar si los productos de una cierta fase del desarrollo de software cumplen o no los requisitos establecidos durante la fase anterior.
- **Validación:** proceso de evaluación del software al final del desarrollo para asegurar el cumplimiento de las necesidades del cliente. (The Institute of Electrical and Electronics Engineers, Inc., 1990)

El Modelo de Madurez y Capacidad Integrado (CMMI por sus siglas en inglés) define los tipos de evaluación, como:

- **Verificación:** proceso para asegurar que los productos de trabajo seleccionados responden a los requerimientos especificados.
- **Validación:** proceso para demostrar que un producto o componente satisface su uso pretendido, en el ambiente operativo planeado. (Bonilla Sánchez, 2006)

En otras palabras, la verificación es la comprobación de que el producto se ha hecho de la manera correcta con el uso de estándares, procedimientos, etc., mientras que la validación es la comprobación de que se cumpla con lo pactado con el cliente.

Técnicas de evaluación del software

Tanto para la realización de verificaciones como de validaciones se pueden utilizar distintos tipos de técnicas, las cuales se agrupan en dos categorías:

Técnicas de evaluación estáticas: buscan faltas sobre el sistema en reposo. Estudian los distintos modelos que componen el software buscando posibles errores en los mismos. Estas técnicas se pueden aplicar, tanto a requisitos como a modelos de análisis, diseño y código.

Técnicas de evaluación dinámicas: generan entradas al sistema con el objetivo de detectar fallos, cuando el sistema ejecuta dichas entradas. Los fallos se observan cuando se detectan incongruencias entre la salida esperada y la salida real. La aplicación de técnicas dinámicas es también conocida como pruebas de software y se aplican generalmente sobre código, puesto que es, hoy por hoy, el único producto ejecutable del desarrollo. (Juristo, y otros, 2006)

- **Técnicas de evaluación estáticas**

La evaluación estática de los distintos artefactos o productos que se generan en el desarrollo de software tiene como objetivo comprobar la calidad de dichos artefactos.

Las técnicas de evaluación estáticas se aplican en el mismo orden en que se van generando los distintos productos del desarrollo siguiendo una filosofía *top-down.* Esta técnica de evaluación es el único modo disponible de evaluación de artefactos para las primeras fases del proceso de desarrollo (análisis y diseño), cuando no existe código. (Juristo, y otros, 2006)

La evaluación estática se realiza conjuntamente con el proceso de construcción del sistema, que consta de una actividad de evaluación correspondiente con cada actividad de desarrollo; es decir, la actividad de definición de requisitos de usuario va acompañada de una actividad de revisión de requisitos de usuario, y así, sucesivamente.

A las técnicas de evaluación estáticas de artefactos del desarrollo, se les conoce de modo genérico por **revisiones,** las que pretenden detectar manualmente defectos en cualquier producto del desarrollo, es decir: modelos, documentos, clases y otros artefactos, siempre sin ejecutar la aplicación.

Una vez realizadas las revisiones pertinentes a la evaluación estática, se procede a ejecutar la evaluación dinámica sobre el código.

- **Técnicas de evaluación dinámicas**

La evaluación dinámica comienza únicamente cuando finaliza la actividad de codificación, y le sigue así una estrategia *bottom-up*. A la aplicación de las técnicas de evaluación dinámicas se le denomina también pruebas de software.

Concretamente, las pruebas de software se pueden definir como una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas (configuración de la prueba), registrándose los resultados obtenidos. Seguidamente se realiza un proceso de evaluación en el que los resultados obtenidos se comparan con los resultados esperados para localizar fallos en el software. Estos fallos conducen a un proceso de depuración en el que es necesario identificar la falta asociada con cada fallo y corregirla, lo que da lugar a una nueva prueba. Como resultado final se puede obtener una determinada predicción de fiabilidad, o un cierto nivel de confianza en el software probado. (Juristo, y otros, 2006)

Las técnicas de evaluación dinámicas o pruebas proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. Estas técnicas se agrupan en:

Técnicas de caja blanca o técnicas estructurales: se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.

Técnicas de caja negra o técnicas funcionales: realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. (Juristo, y otros, 2006)

Durante la evaluación dinámica del sistema es importante aclarar que se detecta un fallo en un programa, no la falta que provoca la aparición del mismo.

1.3 Pruebas

Los procesos asociados a la realización de pruebas de software son de vital importancia cuando se requiere comprobar la calidad de los artefactos construidos, es por ello que se realizan en los proyectos comprobaciones necesarias para lograr la obtención de un producto de software idóneo. A continuación serán analizados una serie de elementos correspondientes a esta esencial tarea.

Definición

La prueba del software es un proceso que puede planificarse y especificarse sistemáticamente. Se puede llevar a cabo el diseño de casos de prueba, definir una estrategia y evaluar los resultados en comparación con las expectativas prescritas. (Sommerville, 2005)

Concretamente la prueba de software se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas (configuración de la prueba), registrándose los resultados obtenidos. (Juristo, y otros, 2006)

Las pruebas de software son actividades para verificar la calidad de un programa informático, donde son identificados errores de implementación, usabilidad, seguridad, entre otros.

¿Quién las hace?

Durante las primeras etapas de la prueba, es el ingeniero del software quien realiza todas las pruebas. Sin embargo, conforme al proceso de prueba, los especialistas en pruebas se van incorporando.

Objetivos

El proceso de pruebas de software tiene dos objetivos:

1- Demostrar al desarrollador y al cliente que el software satisface los requerimientos. Para el software a medida, esto significa que debería haber al menos una prueba para cada requisito de los documentos de requerimientos del sistema y del usuario.

2- Descubrir defectos en el software donde el comportamiento de éste es incorrecto, no es deseable o no cumple su especificación. La prueba de defectos está relacionada con la eliminación de todos los tipos de comportamientos del sistema no deseables, tales como: caídas del sistema, interacciones no permitidas con otros sistemas, cálculos incorrectos y corrupción de datos. (Sommerville, 2005)

Los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software y, de alguna manera, indican la calidad del software como un todo. (Pressman, 2002)

Principios

Existen varios principios que deben cumplir las pruebas de software:

- A todas las pruebas se les debería poder hacer una trazabilidad con los requisitos.
- Las pruebas deberían planificarse mucho antes de que empiecen. La planificación puede empezar tan pronto como esté completo el modelo de requisitos.
- El principio de Pareto[1] es aplicable a la prueba del software.
- Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”. Las primeras pruebas planeadas y ejecutadas se centran generalmente en módulos individuales, a medida que estas avanzan, desplazan su punto de mira en un intento de encontrar errores en grupos integrados de módulos, y finalmente en el sistema entero.
- No son posibles las pruebas exhaustivas. Es imposible ejecutar todas las combinaciones de caminos durante las pruebas.

Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente. Los responsables de la creación del sistema, no son los más adecuados para llevar a cabo las pruebas al software.

Técnicas de prueba

Las pruebas se pueden clasificar de acuerdo a la parte visible o no visible de la aplicación, es decir, si se prueba el código o la funcionalidad a través de la interfaz.

- **Pruebas de caja blanca**

Las pruebas de caja blanca, también llamadas pruebas estructurales, técnicas de caja transparente o de cristal, analizan de manera minuciosa el funcionamiento y estructura interna del programa. Su objetivo es elaborar casos de pruebas que permitan la ejecución de todas las sentencias del programa, incluidas las condiciones, al menos una vez cada una.

[1]El principio de Pareto implica que al 80% de todos los errores descubiertos durante las pruebas se les puede hacer un seguimiento de hasta un 20% de todos los módulos del programa.

Debido a que en un programa sería muy complejo probar todos los caminos posibles, existen criterios que permiten decidir cuáles de ellos se deben examinar. Ver Anexo 1. Criterios para diseñar casos de prueba de caja blanca y caja negra.

- **Pruebas de caja negra**

Las pruebas de caja negra, también conocidas como pruebas funcionales o pruebas de comportamiento, se enfocan en la interfaz de un programa, no en la manera en que ha sido implementado. La analogía de la "caja negra" se debe a que el código ("lo que está en el interior de la caja") no se puede ver debido al "color negro".

Normalmente, un programa tiene un conjunto numeroso de posibles entradas y salidas. Por eso, para diseñar una prueba de caja negra se debe escoger solo un grupo de posibles entradas, entre las cuales debe haber un conjunto de entradas que representen una alta probabilidad de generar salidas erróneas, con el objetivo de probar si el programa está preparado para tratar los errores que se puedan generar en algún momento.

Para diseñar un caso de prueba de caja negra existen varios criterios, los cuales se encuentran detallados en el Anexo 1. Criterios para diseñar casos de prueba de caja blanca y caja negra.

Niveles de prueba

Cuando se va a examinar un programa, debe hacerse comenzando por los componentes más simples hasta llegar al nivel más alto, que sería el sistema. Estos niveles, también llamados estrategias, son los siguientes:

- **Pruebas unitarias:** son las inspecciones que se le realizan a cada módulo de manera independiente. La responsabilidad de este nivel recae generalmente en quien lo creó. Su objetivo es comprobar que se ha implementado correctamente. Los casos de pruebas de caja blanca tienen que diseñarse luego de ser implementado el módulo, no así los de caja negra, que pueden confeccionarse de forma anticipada.

Se entiende como **módulo** a un componente (generalmente una clase) que debe implementar una función independiente simple; puede ser probado por separado y no deberá tener más de 500 líneas de código.

- **Pruebas de integración:** integran la reinspección de los módulos probados de manera combinada. Esto se hace con el objetivo de chequear que la combinación de las salidas de cada módulo se ejecuta correctamente. Entiéndase “combinación” como el uso de los resultados por un componente.

La integración se realiza de dos formas: incremental y no incremental. Esta última consiste en probar el funcionamiento de todos los módulos a la vez, lo cual puede traer consigo una alta ocurrencia de fallos y gran dificultad para detectarlos.

La integración incremental es la selección de un conjunto de módulos cuyo funcionamiento conjunto representan sub-funciones específicas; puede ser de dos tipos: **ascendente y descendente.**

Integración descendente: consiste en integrar los módulos comenzando por el programa principal y luego se van añadiendo a la prueba los módulos que se le subordinan, utilizando el recorrido primero en profundidad o primero a lo ancho. Para realizar esto, debe tenerse en cuenta la estructura de la aplicación de una manera jerárquica en forma de árbol.

Cuando se selecciona el recorrido primero en profundidad, se escoge un camino principal de manera arbitraria, para lo cual debe tenerse en cuenta, que algunos módulos que no formen parte de dicho camino, deberían agregarse para un adecuado funcionamiento de los módulos superiores.

En el caso del recorrido primero a lo ancho, se integran los módulos inmediatamente subordinados, para luego añadirse a la prueba los del siguiente nivel, hasta llegar a los módulos más elementales.

Integración ascendente: se comienzan a integrar los módulos que están más abajo en la estructura del programa, es decir, los que no dependen de ningún otro módulo para poder ejecutarse. Para llevarlo a cabo, es recomendable, seleccionar un grupo de módulos que realicen una sub-función específica y se añade temporalmente un componente controlador para supervisar la entrada y salida de los datos del caso de prueba. Luego de ser probado, se elimina el controlador y se continúa integrando grupos, moviéndose hacia arriba en la estructura del programa.

- **Prueba del sistema:** todo el software, integrado con el *hardware*, se prueba para comprobar que se cumplen con los requisitos funcionales y no funcionales establecidos.

Esta prueba suele hacerse inicialmente en el entorno del desarrollador, denominadas **Pruebas Alfa**, y seguidamente en el entorno del cliente, denominadas **Pruebas Beta**. (Juristo, y otros, 2006)

Las pruebas del sistema aplicadas para comprobar características de un producto informático, generalmente especificadas en los requisitos no funcionales, son llamadas **tipos de prueba**; se clasifican en:

Pruebas de seguridad: buscan fallos de seguridad en el diseño e implementación del sistema verificando los mecanismos de control de acceso para evitar alteraciones indebidas en los datos.

Pruebas de rendimiento: verifican que los tiempos de respuesta están dentro de los intervalos establecidos en los requisitos no funcionales del sistema.

Pruebas de carga o volumen: comprueban que el sistema soporte cargas masivas de datos. Se busca establecer cuándo el sistema empieza a operar por debajo de los requisitos establecidos.

Pruebas de resistencia o estrés: consiste en poner cierta cantidad de usuarios virtuales accediendo concurrentemente durante varias horas y verificar que la aplicación responde a las peticiones en el tiempo establecido.

Pruebas de usabilidad: consisten en comprobar la adaptabilidad del sistema a las necesidades de los usuarios para asegurar que se acomoda a su modo habitual de trabajo, así como para identificar las facilidades que aporta introducir datos en el sistema y obtener los resultados.

Pruebas de fiabilidad: comprueba que un programa realice su objetivo satisfactoriamente (sin fallos) en un determinado período de tiempo y en un entorno concreto.

Pruebas de compatibilidad: ayudan a determinar si el producto funcionará correctamente con otro *hardware* y software en el entorno pretendido.

Pruebas de recuperación: consisten en demostrar que el sistema puede recuperarse ante fallos, tanto en *hardware* como en software, sin comprometer la integridad de los datos.

Pruebas de estructura: se realizan a aplicaciones web, evaluando el estado de sus enlaces y contenido. Esta prueba se basa en evaluar aspectos tales como: existencia de enlaces rotos, existencia de contenido huérfano y mostrar el contenido deseado.

Continuando con los niveles de prueba:

- **Pruebas de aceptación:** es una prueba que realiza el cliente para determinar su conformidad con el resultado final del sistema, o lo que es lo mismo, los requisitos pactados. Los casos de prueba deben ser ejecutados por el usuario, incluso es aconsejable que sea él mismo quien los proponga. Estas pruebas deben realizarse en el ambiente donde se va a explotar el sistema.
- **Pruebas de regresión:** son las pruebas que se le realizan a un módulo o conjunto de estos luego de ser modificados. Su objetivo es comprobar que los cambios no han originado efectos adversos. Incluyen la repetición de casos de pruebas realizados anteriormente relacionados con la parte del sistema modificada, la revisión de los procedimientos manuales preparados antes del cambio para asegurar que permanecen correctamente y la obtención impresa del diccionario de datos de forma que se compruebe que los elementos de datos que han sufrido algún cambio son correctos. (Rojas, y otros, 2007)

Para realizar las pruebas unitarias deben aplicarse tanto las técnicas de prueba de caja blanca como las de caja negra; no así las pruebas de integración, de sistema y aceptación, para las cuales se utilizan solo las pruebas de caja negra en la construcción de los casos de prueba.

Las técnicas de pruebas que más se realizan son las de caja negra, debido al valor que tiene el cumplimiento de los requisitos funcionales para el usuario final, así como la complejidad que tienen las inspecciones al código.

En el caso de los niveles de pruebas, existe una mayor similitud en cuanto a su popularidad. Son muy practicadas las pruebas unitarias y las pruebas de sistema. Las pruebas de integración se emplean repetidas veces en las grandes aplicaciones, con numerosos módulos. Las pruebas de regresión, aunque algunos autores, no las reconocen como un nivel, pudieran ser consideradas como las más populares entre los desarrolladores, por lo que significan para encontrar los defectos. Por último, las pruebas de aceptación son ejecutadas, al menos, una vez en cada proyecto.

1.4 Automatización de las pruebas de software

La automatización de pruebas es la parte del ciclo de calidad, en la que el software de automatización es utilizado para controlar la ejecución de pruebas, comparación de resultados, preparación de precondiciones y realización de informes. (Rubiano, y otros, 2009)

La automatización de pruebas comprende la utilización de software de automatización para ejecutar pruebas de forma desatendida (sin interacción alguna por parte del ingeniero de pruebas). Las pruebas automatizadas son mayormente efectivas en entornos donde los cambios son frecuentes o en aplicaciones en las que se esperan *builds*[2] y *releases*[3] críticos. Los procesos de negocio que no cumplen estas características deben ser ejecutados de forma manual. (Testhouse, 2010)

Existe una gran variedad de herramientas de automatización de pruebas y marcos de trabajo disponibles, los cuales han creado un reto a la hora de aplicarlas. Esto genera la necesidad de destinar un equipo con experiencia en el sector a la automatización de pruebas. (Testhouse, 2010)

En el mundo existen varias compañías y organizaciones encargadas de producir y actualizar herramientas que permiten la automatización de pruebas de software. Ver Anexo 2. Empresas y organizaciones que desarrollan herramientas para pruebas de software.

- **Herramientas que permiten la automatización de las pruebas de software**

Para ver más información sobre las herramientas remitirse al Anexo 3. Herramientas que permiten la automatización de las pruebas de software.

Herramientas comerciales

Microsoft Visual Studio 2010: es capaz de realizar diferentes pruebas a aplicaciones desarrolladas en .NET.

[2] *Build* (construcción): en Informática, se le llama así al proceso de convertir los archivos de código fuente de un programa y en uno o varios artefactos que puedan ejecutarse en una computadora, a menudo un archivo ejecutable.
[3] *Release* (lanzamiento): en Informática, se dice de la versión de un producto de software que será publicada o puesta a disposición del cliente.

HP Quality Center: permite disponer de una aplicación web para la realización de pruebas y gestionar cualquier información relacionada con estos fines.

HP QuickTest Professional: soporta la automatización de pruebas de regresión y pruebas funcionales.

HP Performance Center: ofrece una plataforma de pruebas de rendimiento de las aplicaciones de tipo corporativo que se puede gestionar mediante la web.

HP LoadRunner: ayuda a prevenir problemas de rendimiento.

HP WebInspect: ofrece una rápida exploración y una amplia evaluación de la seguridad de las aplicaciones web.

Benchmark Factory: simula el acceso de miles de usuarios a servidores de bases de datos, archivos, Internet y correo.

PerformaSure: permite detectar servidores o procesos que degradan el funcionamiento de la aplicación distribuida.

Rational Robot: permite automatizar las pruebas de regresión de aplicaciones .NET, Java, web y otras.

Spotlight: realiza diagnósticos de alta precisión sobre problemas de rendimiento en tiempo real y muestra gráficamente toda la actividad de la aplicación.

Data Factory: genera bases de datos de desarrollo, pruebas y control de calidad.

JProbe: permite detectar los “puntos calientes” de los componentes de una aplicación Java, tales como el uso de la memoria.

JTest: realiza pruebas de funcionalidad del software de manera automática para aplicaciones empresariales.

Herramientas libres

Open Load Tester: puede ser utilizado para probar casi cualquier aplicación basada en web.

JUnit: marco de trabajo para realizar pruebas unitarias a aplicaciones Java.

TestLink: permite gestionar casos de prueba y organizarlos en planes.

Mantis: es una herramienta para la gestión de defectos.

Selenium IDE: es un complemento para el navegador Mozilla Firefox que permite realizar pruebas funcionales.

JMeter: herramienta para realizar pruebas de rendimiento, una de las más conocidas.

Backtrack 4: es una distribución de Linux basada en Ubuntu que incluye numerosas aplicaciones para realizar pruebas de seguridad.

Sonar: permite realizar pruebas unitarias en Java, determinar la complejidad del código, identificar errores potenciales y duplicaciones.

Cobertura: está escrita en Java y permite saber la cantidad de código probado.

Checkstyle: ayuda a los programadores a escribir código Java que se adhiera a los estándares.

PMD: complemento de código abierto que se integra a varios IDE (Eclipse, NetBeans, JBuilder, etc.) y busca problemas potenciales.

Las herramientas hasta ahora mencionadas solo son algunas de las más populares entre desarrolladores y probadores de aplicaciones construidas en plataformas libres y propietarias.

Haciendo un resumen de la utilización de software para pruebas, se puede plantear que para las técnicas de prueba de caja blanca en la plataforma .NET, evidentemente **Visual Studio**, es el programa con mayor popularidad, al contener funcionalidades que permiten probar el código. Este IDE tiene actualmente más de 6 millones de desarrolladores en su espectro de usuarios, lo que lo convierte en el número uno de su tipo para los sistemas operativos de **Microsoft**.

HP, destaca como la compañía que domina el mercado del software para pruebas, desarrollando y actualizando múltiples herramientas para garantizar el cumplimiento de casi todo requisito. **Quest Software,** por su parte, es la compañía que más ha avanzado en el último quinquenio; tiene cien mil clientes y ha logrado vender diversas aplicaciones especializadas en verificar software, las cuales se han

relacionado aquí. Tanto **HP** como **Quest**, han dedicado sus esfuerzos fundamentalmente a la plataforma **Windows**.

Para las pruebas de software en aplicaciones desarrolladas en **Java**, se destaca **JMeter**, la cual es gratuita y sumamente versátil, permitiendo pruebas funcionales, de rendimiento, etc. Otras herramientas gratuitas muy populares por su utilidad son **JUnit** y **Open Load Tester**.

En la UCI se utilizan varias herramientas; ellas son: **JMeter**, **JUnit**, **Selenium IDE**, **Visual Studio 2008 Team System y 2010, Nessus, Nmap, Nikto, Wapiti, W3af, Zenmap, Ksser, APadora, TestMaker, PHPUnit y Shadow Security Scanner**. El centro Calisoft realiza pruebas funcionales con **Selenium IDE**, mientras que en los proyectos se acostumbra a usar **JUnit** para pruebas unitarias en aplicaciones desarrolladas en Java. **JMeter** es muy usada para realizar pruebas de estrés. Los proyectos que desarrollan sobre la plataforma Microsoft .NET, usan **Visual Studio** para la realización de las pruebas al código, excepto los que programan en **Visual Studio .NET 2003**, que hacen las pruebas unitarias con **NUnit**. De manera general, el empleo de este tipo de software no es muy avanzado en todos los centros de la Universidad de las Ciencias Informáticas.

Actualmente, numerosas instituciones de Cuba están sufriendo un proceso de migración a software libre, el cual tiene varios objetivos, entre los que se encuentran: eliminar la dependencia tecnológica, ampliar las alternativas de desarrollo, abaratar los productos, mejorar la seguridad de los sistemas y optimizar el rendimiento de las aplicaciones. Por tanto, la propuesta de herramientas para pruebas de software automatizadas priorizará las tecnologías libres.

1.5 Metodología, tecnologías y herramientas para el desarrollo de la aplicación

A continuación se describirán las tecnologías, herramientas y metodología a utilizar en el desarrollo de la aplicación que permitirá la gestión de las herramientas para pruebas de software. Para ver más información sobre el análisis realizado para esta selección ver el Anexo 5. Comparación de la metodología, tecnologías y herramientas utilizadas con otras similares.

Arquitectura cliente-servidor

La arquitectura cliente-servidor puede definirse como una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible y permita simplificar las actualizaciones y mantenimiento del sistema. (Universidad Carlos III, 2007)

Esta arquitectura puede considerarse como una variante de la arquitectura en capas, que define responsabilidades para cada capa, permitiendo una mejor gestión de los cambios en las aplicaciones, así como la transparencia e independencia entre ellas.

El modelo cliente-servidor es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que, definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información. Se establece así un enlace de comunicación transparente entre los elementos que conforman la estructura.

El sistema, al estar basado en la web, responde a este tipo de arquitectura y tiene como **características**:

- Orientado a servicios: el servidor los ofrece y el cliente los consume.
- Compartición de recursos: servicios ofrecidos a muchos clientes. Un servidor puede atender muchos clientes que solicitan esos servicios.
- Transparencia de ubicación: el servidor es un proceso que puede residir en el mismo ordenador que el cliente o en uno distinto a lo largo de una red. Un programa puede ser un servidor en un momento y convertirse en un cliente posteriormente.
- Interacción a través de mensajes, para envío y respuesta de servicios.
- Servicios encapsulados, exponiendo los servicios a través de interfaces, lo que facilita la sustitución de servidores sin afectar a los clientes y permite a la vez una fácil escalabilidad.
- Mezcla e igualdad: una de las más importantes ventajas de este paradigma. Una aplicación cliente-servidor, idealmente es independiente del *hardware* y de sistemas operativos; mezclando e igualando estas plataformas.

Sus **principales ventajas** son: la centralización del control mediante un servidor; la escalabilidad, permitiendo aumentar la capacidad de clientes y servidores por separado; un mejor aprovechamiento de la

potencia de cómputo debido a que reparte el trabajo; la reducción del tráfico en la red y el uso de interfaces gráficas variadas.

Aplicaciones web

Una aplicación web es aquella en la que los usuarios, usando un navegador, acceden a un servidor web a través de Internet o de una Intranet. Las aplicaciones web son populares debido a la facilidad de uso del navegador web como cliente ligero. La habilidad, para actualizar y mantener aplicaciones web sin distribuir e instalar software en miles de potenciales clientes, es otra razón de su popularidad. Las aplicaciones web generan dinámicamente una serie de páginas en un formato estándar como HTML (Lenguaje de Marcado de Hipertexto) o XHTML (Lenguaje Extendido de Marcado de Hipertexto), soportado por navegadores web comunes.

Las aplicaciones web son una especialización y concreción de las aplicaciones cliente-servidor, o sea, su arquitectura general es la de un sistema cliente-servidor, donde tanto el cliente (el navegador) como el servidor (el servidor web), y el protocolo, mediante el que se comunican (el HTTP: Protocolo de Transferencia de Hipertexto*)* son estándar, y no han de ser creados por el desarrollador. La parte del cliente de las aplicaciones web está formada por el código HTML que forma la página web, con opción a código ejecutable, mediante los lenguajes *script* de los navegadores (JavaScript, VBScript, PerlScript) o mediante pequeños programas (*applets*) en Java. La parte del servidor está formada por un programa o *script* que es ejecutado por el servidor web, y cuya salida se envía al navegador del cliente. (SPL Sistemas de Información, 2008)

Para el desarrollo del sistema se tiene presente las **ventajas** de las aplicaciones web:

- **Multiplataforma:** pueden ser utilizadas a través de múltiples plataformas, tanto de *hardware* como de software.
- **Actualización instantánea:** debido a que todos los usuarios de la aplicación hacen uso de un solo programa que radica en el servidor, los usuarios siempre utilizarán la versión más actualizada del sistema.

- **Suave curva de aprendizaje:** los usuarios, como utilizan la aplicación a través de un navegador, hacen uso del sistema tal como si estuvieran navegando por Internet, por lo cual su acceso es más intuitivo.
- **Fácil de integrar con otros sistemas:** debido a que se basa en protocolos estándares, la información manejada por el sistema puede ser accedida con mayor facilidad por otros sistemas.
- **Acceso móvil:** el usuario puede acceder a la aplicación con la única restricción de que cuente con un acceso a la red privada de la organización o a Internet, dependiendo de las políticas de dicha organización.

Lenguajes de programación web

En la actualidad existe una gran variedad de lenguajes de programación para aplicaciones web. Se define su uso de acuerdo a la plataforma y gestor de base de datos que se desee utilizar. Dichos lenguajes se clasifican en dos partes fundamentales: los lenguajes del lado del servidor y los lenguajes del lado del cliente.

Entre los lenguajes del lado del servidor más sobresalientes están: Perl, ASP (*Active Server Pages*), PHP, Java, entre otros. Estos se caracterizan por desarrollar la lógica de negocio dentro del servidor, además de ser los encargados del acceso a bases de datos, tratamiento de la información, etc. Del lado del cliente se encuentra principalmente el JavaScript, encargado de aportar dinamismo a la aplicación en los navegadores. Seguidamente se analiza el lenguaje utilizado para el desarrollo de este trabajo.

Procesador de Hipertexto (PHP)

PHP es un lenguaje de código abierto utilizado para fines generales; está especialmente pensado para desarrollos web y puede ser embebido en páginas HTML. (The PHP Group, 2001)

Este lenguaje de programación se ejecuta en el servidor web, es rápido e independiente de la plataforma. Dispone de gran documentación y posee una extensa librería con múltiples funcionalidades. Admite la programación orientada a objetos, posee compatibilidad con las bases de datos más comunes, tales como: MySQL, Oracle y PostgreSQL; y proporciona soporte para la mayoría de los protocolos de comunicación conocidos, como: HTTP, IMAP (Protocolo de Acceso a Mensajes de Internet), FTP, LDAP, entre otros.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales. Puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, Microsoft Windows y Mac OS. Además, soporta la mayoría de los servidores web de hoy en día.

CodeIgniter

CodeIgniter es un conjunto de herramientas para personas que construyen su aplicación web usando PHP. Su objetivo es permitir desarrollar proyectos mucho más rápidos de lo que se podría, si se escribiese desde cero, proporcionándole un rico juego de librerías para tareas comúnmente necesarias, así como una interfaz simple y estructura lógica para acceder a esas librerías. Permite enfocarse en el proyecto minimizando la cantidad de código necesario para una tarea dada. (Lozano, 2006)

Ventajas del marco de trabajo CodeIgniter:

- Amplia documentación disponible.
- Compatibilidad con una amplia variedad de servidores y configuraciones.
- Ayuda a disminuir los tiempos de entrega gracias a la reutilización de código.
- Tiene una activa comunidad de usuarios.

Metodología de desarrollo de software

El desarrollo del 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 están aquellas propuestas más tradicionales, que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Las metodologías tradicionales han demostrado ser muy efectivas en muchos proyectos, pero no en su totalidad.

Por otro lado, existe la filosofía de las metodologías ágiles, centrada en otras dimensiones, como por ejemplo, el factor humano o el producto de software, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

No existe una metodología absoluta, por lo cual se debe tener previo conocimiento de las características del proyecto antes de seleccionar la metodología que será utilizada, debido a que la realización de esta actividad constituye un factor determinante en el éxito del proyecto.

Para el desarrollo del software se decidió utilizar metodologías ágiles, específicamente la Programación Extrema (XP).

Entre los elementos determinantes para su elección se encuentran:

- El tamaño del grupo de desarrollo, en este caso, de dos personas.
- Necesidad de resultados tangibles a corto plazo.
- Imposibilidad, para un grupo de desarrollo pequeño, de asumir una metodología robusta, debido a la cantidad excesiva de roles y documentación generada en el ciclo de vida del proyecto.

La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código. Los objetivos de XP son muy simples: la satisfacción del cliente y potenciar al máximo el trabajo en grupo. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. (Wells, 2009)

XP supone que:

- Las personas son claves en los procesos de desarrollo.
- Los programadores son profesionales, no necesitan supervisión.
- Los procesos se aceptan y se acuerdan, no se imponen.
- Desarrolladores y gerentes comparten el liderazgo del proyecto.
- El trabajo de los desarrolladores con las personas que conocen el negocio es regular, no puntual. (Calero Solís, 2003)

Ventajas de XP:

- Se consiguen productos usables con mayor rapidez.
- El equipo de desarrollo está más contento y motivado. Las razones son, por un lado el que la XP no permite excesos de trabajo (se debe trabajar 40 horas a la semana), y por otro la comunicación entre los miembros del equipo que consigue una mayor integración entre ellos.
- Permite una mejor adaptabilidad a los cambios, que se traduce en una reducción de costos.

- Apropiado para entornos volátiles, equipos de desarrollo pequeños (de 2 a 10 desarrolladores) y proyectos de alto riesgo.
- Permite tener retroalimentación continua de los usuarios a través de las entregas frecuentes.
- Gracias a la filosofía de programación en parejas, se consigue que los desarrolladores apliquen las buenas prácticas que se les ofrecen con la XP.
- El proceso de integración es continuo, por lo que el esfuerzo final para la integración es nulo.
- Se consiguen productos más fiables y robustos contra los fallos gracias al diseño de las pruebas de forma previa a la codificación. (Cortizo Pérez, y otros)

Lenguaje de modelado

En la modelación de la aplicación, se utilizará el Lenguaje Unificado de Modelado (UML por sus siglas en inglés). Este es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

El Lenguaje Unificado de Modelado (UML) ayuda al usuario a entender la realidad de la tecnología y la posibilidad de que reflexione antes de invertir y gastar grandes cantidades en proyectos que no estén seguros en su desarrollo, reduciendo el costo y el tiempo empleado en la construcción de las piezas que constituirán el modelo. (Booch, y otros, 2000)

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y proporciona un estándar que permite al analista de sistemas, generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que estén involucrados en el proceso de desarrollo. Un modelo UML indica qué es lo que supuestamente hará el sistema pero no cómo lo hará. (Larman, 1999)

A continuación se mencionarán las **herramientas** a utilizar en el desarrollo de la aplicación:

Servidor web Apache

Apache es un servidor de código abierto HTTP para sistemas operativos modernos. De renombre, por ser seguro, eficiente y un servidor web expandible; los servicios Apache están de acuerdo (o en

sincronización) con los estándares HTTP actuales. Para realizar la instalación del servidor Apache de forma más sencilla se usará el servidor independiente de plataforma XAMPP (**X**: cualquier plataforma, **A**pache, **M**ySQL, **P**HP, **P**erl). Apache es el servidor web más popular en el Internet desde Abril de 1996 y fue desarrollado por la Fundación de Software Apache.

Entorno integrado de desarrollo

Un entorno de desarrollo integrado (IDE, siglas inglesas de *Integrated Development Environment*) es un programa informático compuesto por un conjunto de herramientas de programación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Puede dedicarse en exclusiva a un solo lenguaje de programación, o bien, poder utilizarse para varios.

Entre los entornos integrados de desarrollo más utilizados se encuentran: Visual Studio, Eclipse, NetBeans, entre otros. Se decidió trabajar con NetBeans debido a que permite la programación en PHP; es un IDE poseedor de una interfaz amigable y de gran potencia.

NetBeans

Este IDE es una herramienta para programadores, pensada para escribir, compilar, depurar y ejecutar programas. Su misión consiste en evitar tareas repetitivas, facilitar la escritura de código correcto, disminuir el tiempo de depuración e incrementar la productividad del desarrollador. Es un producto libre y gratuito sin restricciones de uso y está escrito completamente en Java.

Sistema de Gestión de Base de Datos (SGBD)

Un SGBD es un conjunto coordinado de programas, procedimientos, lenguajes, etc. que suministra a los distintos tipos de usuarios los medios necesarios para describir y manipular los datos almacenados en la base de datos, garantizando su seguridad. (Chávez Moreno, 2011)

MySQL

MySQL es el sistema de gestión de base de datos relacional más popular de código abierto del mundo. Se desarrolla y se distribuye con el apoyo de Oracle Corporation. Tiene licencia GPL. Su arquitectura lo hace extremadamente rápido, fiable, fácil de usar y adaptar. (Oracle Corporation, 2010)

Entre sus ventajas se pueden encontrar:

- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Soporta gran variedad de sistemas operativos.
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.

Herramienta CASE

Las herramientas CASE (Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas con el objetivo de aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero; ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Visual Paradigm

El Visual Paradigm es una poderosa herramienta CASE de modelación visual. Utiliza UML para el modelado, permitiendo crear tipos diferentes de diagramas en un ambiente totalmente visual. Es muy sencillo de usar, fácil de instalar y actualizar. Genera código para varios lenguajes. Tiene disponible distintas versiones: *Enterprise, Professional, Standard, Modeler, Personal* y *Community*. Facilita licencias especiales para fines académicos.

Se propone Visual Paradigm como herramienta a utilizar específicamente la versión *Enterprise* de Visual Paradigm, debido a que posibilita la representación gráfica de los diagramas que permiten ver el sistema desde diferentes perspectivas, como el de componentes, despliegue, secuencia, casos de uso, clase, actividad, estado, entre otros. Además, identifica requisitos y comunica información, se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos, además, permite ver las relaciones entre los componentes del diseño y mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico. (Company Headquarters, 2010)

1.6 Conclusiones

A partir del análisis realizado de los temas relacionados con la gestión de la calidad y las herramientas para pruebas de software se concluye, que las mismas constituyen una atractiva alternativa al engorroso trabajo realizado como parte del plan de pruebas dentro del Centro de Identificación y Seguridad Digital (CISED), reforzando la idea a defender de la investigación.

Se hizo un estudio de las principales herramientas para pruebas de software con el objetivo de hacer, posteriormente, una propuesta con aquellas que sean consideradas más adecuadas para los proyectos que son desarrollados dentro del CISED.

CAPÍTULO 2

PROPUESTA DE HERRAMIENTAS PARA PRUEBAS DE SOFTWARE

En el presente capítulo se expondrá una caracterización de los Procesos de Desarrollo de Software (PDS) en el CISED, donde a través de entrevistas y encuestas realizadas a los jefes de departamentos del Centro y miembros de los proyectos, se registrarán las tecnologías, metodologías, patrones arquitectónicos y pruebas que se utilizan; posteriormente se propondrán un conjunto de herramientas para pruebas de software que permitirán facilitar el trabajo para la verificación y validación de las aplicaciones.

2.1 Caracterización de los PGCS en el CISED

Realizar la gestión de la calidad de un software en un proyecto es una labor difícil; las dificultades crecen considerablemente cuando deben ser gestionados simultáneamente múltiples desarrollos. El CISED está compuesto por los siguientes departamentos: **Identificación, Biometría, Tarjetas Inteligentes, Seguridad Digital y Soluciones Integrales**, los cuales tienen el objetivo de planificar y controlar la elaboración de los múltiples proyectos que en su interior son desarrollados. Para ver los proyectos de cada uno de los departamentos y su clasificación remitirse al Anexo 6. Organización de los proyectos del CISED y su clasificación.

Al realizarse un levantamiento de los proyectos existentes, se pudo detectar la diversidad de metodologías y pruebas utilizadas y realizadas respectivamente por cada uno de ellos. Esta caracterización se realizó mediante un conjunto de encuestas y entrevistas al personal del Centro. Ver Anexo 7. Encuesta realizada al personal del CISED sobre las características de los proyectos.

En el CISED es implementado el SGC a través de los procesos: **planificación, aseguramiento y control de la calidad**.

- **Planificación**

La planificación es un proceso clave donde se identifican las normas y métricas de calidad relevantes para el proyecto. Son definidas y registradas numerosas actividades asociadas al desarrollo del proyecto en cronogramas que se establecen para planificar el tiempo y los recursos para cada una de ellas, tales como: las fases del ciclo de vida del proyecto, la realización de evaluaciones y auditorías, reuniones de equipo, entre otras.

En este proceso, son desarrollados cursos de capacitación al personal y son definidos los responsables de la calidad en el proyecto, teniendo en cuenta las tareas y los conocimientos que debe cumplir y tener cada rol. Los encargados deben poseer un dominio de la descripción preliminar del proyecto, conocer la forma en que se irá desarrollando el software, los entregables previstos, la metodología que será utilizada, las pruebas a realizar, así como aspectos internos y organizativos.

Durante la planificación se deben definir:

- Puntos de chequeo que permitan verificar el estado del proceso.
- Objetivos medibles y claros para las fases y las iteraciones.
- Tareas atómicas y bien delimitadas por su tamaño para poder precisar su avance.
- Los artefactos que deben producirse en cada etapa para que una vez obtenidos sean evaluados.
- Cuáles son los tipos de elementos de configuración que se obtendrán en el sistema.
- Listas de chequeo asociadas a cada uno de los artefactos.
- Planes de prueba.
- Las plantillas que se utilizarán para cada uno de los documentos.
- Un glosario de términos del proyecto.
- Una herramienta que permita realizar control de versiones sobre los elementos de configuración.

Metodologías de desarrollo de software utilizadas

Una actividad determinante en la GCS es la selección de una metodología de desarrollo de software, la cual define artefactos, fases, flujos de trabajo, roles y otras cuestiones influyentes sobre las actividades que garantizan la calidad.

En el CISED se utilizan las siguientes metodologías: *Extreme Programming* (XP), *Rational Unified Process* (RUP), *Microsoft Solutions Framework* (MSF) para CMMI y para *Agile Software Development*, *Feature*

Driven Development (FDD) y *Scrum* XP (SXP). Cada una de ellas plantea actividades particulares para el aseguramiento de la calidad, y por consiguiente, para la realización de las pruebas.

XP otorga gran importancia a la realización de pruebas de software, incluso, aconseja diseñarlas antes de implementar la aplicación. En esta metodología se crean, a partir de las historias de usuario, pruebas de aceptación, las cuales son revisadas por el cliente junto a los resultados obtenidos. También se realizan pruebas unitarias.

Las historias de usuarios se consideraran listas cuando cumplen todas las pruebas. No debe existir ninguna propiedad o característica del programa que no sea verificada en algún momento. Esta metodología sugiere que cuanto más "fresco" se tenga el código, más fácil será interpretar el error, descubrir sus causas, sus dependencias y sus consecuencias. El momento ideal para probar un código es justo después de haber terminado de programarlo, por eso, es tan importante la realización de pruebas automatizadas que agilicen el proceso de detección de errores y fallas.

RUP está dividido en cuatro fases, las cuales se fragmentan en flujos de trabajo, uno de las cuales es el de Pruebas. Esta metodología plantea que las pruebas deben planificarse para cada iteración, diseñarse e implementarse a través de los llamados casos de pruebas, creando componentes para automatizarlas, así como realizar las pruebas registrando los resultados y verificando la corrección de las no conformidades.

SXP es un híbrido cubano de metodologías ágiles que tiene como base las metodologías *Scrum* y XP que permiten actualizar los procesos de desarrollo de software para el mejoramiento de su producción. Consta de 4 fases: Planificación-Definición, Desarrollo, Entrega y Mantenimiento, cada una desglosada en flujos de trabajo y actividades que generan artefactos. (Iberoamerican Journal of Project Management, 2011)

MSF tiene varios modelos para darle solución a distintos problemas. En el Modelo de Equipos de Trabajo los principales objetivos que se siguen para garantizar la calidad son: entregar el sistema o solución dentro de las restricciones del proyecto (tiempo, recursos, costos), identificar los problemas o riesgos de importancia para el usuario, manejándolos en forma oportuna y asegurar que el usuario final sepa cómo usar el sistema.

FDD es una metodología que cubre las fases de diseño y construcción del software a través de iteraciones. El proceso iterativo incluye inspección de diseño, codificación, pruebas unitarias, integración e

inspección de código. Las listas de funcionalidades generadas son revisadas por los usuarios y los responsables para su validación y aprobación. Uno de los roles adicionales de la metodología es el Probador.

Para ver las metodologías de desarrollo de software usadas en los proyectos del CISED remitirse al Anexo 8. Metodologías de desarrollo de software usadas en los proyectos del CISED.

Patrones arquitectónicos usados

El uso de una determinada arquitectura está relacionado con las tecnologías que permiten su desarrollo. En el CISED se emplean varias arquitecturas de desarrollo de software combinándose de acuerdo a las necesidades de los desarrolladores, las cuales son: arquitectura de repositorio, arquitectura en capas, arquitectura orientada a objetos, arquitectura basada en componentes, arquitectura de máquinas virtuales, arquitectura basada en eventos, arquitectura orientada a servicios y arquitectura basada en recursos. Para ver una descripción detallada de cada uno de estos patrones remitirse al Anexo 9. Patrones arquitectónicos usados en el CISED.

Tecnologías de desarrollo en los proyectos del CISED

Son múltiples las tecnologías que se utilizan en el Centro y que dan soporte a esas arquitecturas; desde herramientas libres a propietarias, desde sistemas operativos basados en Linux hasta la última versión de Windows; los clientes e intereses son diversos.

Dentro de las tecnologías web se utilizan: los lenguajes PHP, CSS (*Cascading Style Sheets*), JavaScript, XML y XHTML; la tecnología ASP.NET y el Sistema de Gestión de Contenidos (CMS) Drupal. Los navegadores usados son: Internet Explorer, Opera, Mozilla Firefox y Google Chrome; las versiones usadas son variables, aunque son recientes. También se utiliza la técnica de desarrollo web AJAX y el marco de trabajo Symfony.

Tabla 2.1.1. Entornos de desarrollo integrados utilizados en el CISED asociados a los lenguajes que soportan (Fuente: Elaboración propia)

IDE	Lenguaje

NetBeans 6.9, Eclipse 3.5 y Gemalto Developer Suite 3.4	Java
MonoDevelop, Visual Studio(VS)2003, 2005, 2008 Team System y 2010	Visual C Sharp
Visual Studio (VS) 2003, 2005, 2008 Team System y 2010	Visual C++
PL/SQL Developer	PL/SQL
NetBeans 6.9	PHP

Team Foundation Server 2010 es utilizado para gestionar diferentes procesos de los proyectos; está integrado con VS 2010. Para estos fines también se utiliza el Paquete de Gestión de Proyectos (GESPRO 1.0), más conocido como Redmine.

Las herramientas de modelado empleadas son: Embarcadero Erwin Studio, para el diseño de las bases de datos; Visual Paradigm y Altova para la mayoría de los diagramas.

Los SGBD que se usan son: Oracle 10g y PostgreSQL 9.0, mientras que los lenguajes de consultas que usan estos gestores son PL/SQL y PL/pgSQL respectivamente.

Para el mapeo de objetos a bases de datos se usan las herramientas TierDeveloper, Hibernate, Doctrine (integrado con Symfony) y Entity Framework 4.0 (integrado con VS 2010).

El resto de los marcos de trabajo usados son: Microsoft .Net Framework 1.1, 2.0, 3.5 y 4.0; Microsoft Ajax; SmartCard Framework (desarrollado por el departamento de Tarjetas Inteligentes); Windows Presentation Foundation (WPF); Mono .NET Framework; JavaCard Development Kit (JCDK); Axis; Spring y Bison (desarrollado por el proyecto Identidad Cuba para el desarrollo de aplicaciones web basadas en flujo de actividades en .NET).

Las distribuciones de Linux empleadas son: Ubuntu, CentOS, Suse Linux Enterprise Server y RedHat Enterprise Linux. También se utilizan: Windows XP Service Pack 3, Windows Server 2003 y Windows 7 Ultimate.

- **Aseguramiento de la calidad**

Durante el **aseguramiento de la calidad**, se comienza a dar cumplimiento a las actividades planificadas para asegurar que el proyecto cumpla con los requisitos definidos inicialmente.

Son desarrolladas por parte del personal interno o externo, pruebas y revisiones estructuradas e independientes como consecuencia de la aparición de entregables, para verificar y validar los requisitos una vez capturados, así como determinar si las actividades realizadas cumplen con las políticas y los procedimientos del proyecto y de la organización, registrándose de esta forma, las no conformidades obtenidas.

El aseguramiento de la calidad del software comprende principalmente: las métricas; la verificación y validación a lo largo del ciclo de vida, incluyendo pruebas y procesos de revisión y auditorías; gestión de configuración; el control de la documentación y un procedimiento que asegure los ajustes a los estándares en el proceso de desarrollo siempre que esto sea posible.

En el CISED se le presta mayor atención a la verificación y validación, gestión de configuración y el control de la documentación.

Revisiones

Los tipos de revisiones que se realizan en el CISED son:

Reuniones informales: son reuniones donde se discuten problemas técnicos.

Presentaciones formales: consisten en presentar requisitos, diseños, aplicaciones o cualquier producto ante clientes, directivos o personal técnico.

Revisiones técnicas formales: son revisiones planificadas para cada iteración de un proyecto. Tienen como objetivos: detectar errores en la implementación, verificar que el software revisado cumple con los requisitos y comprobar el uso de estándares predefinidos. También, hacer más manejables a los proyectos y garantizar la uniformidad del sistema a través de: chequear la correspondencia de la documentación con el Expediente de Proyecto, revisar la ortografía de los artefactos, evaluar la modelación de cualquiera que sea el flujo de trabajo en desarrollo, identificar incongruencias,

ambigüedades, errores gramaticales y otros de tipo lingüístico, realizar un registro de las no conformidades detectadas para su posterior corrección, etc.

Auditorías

Para ver una descripción de las etapas que conforman una auditoría en el CISED remitirse al Anexo 17. Etapas de una auditoría dentro del CISED.

Pruebas

Las pruebas, independientemente de cada metodología, se realizan con cierta periodicidad de acuerdo a los planes de aseguramiento de la calidad de cada proyecto; se pueden realizar para probar una parte del producto o el producto terminado, conforme a las iteraciones definidas.

En el Centro se aplican las técnicas de pruebas de caja blanca y caja negra, así como los niveles: pruebas unitarias, pruebas de integración, pruebas de regresión, pruebas de aceptación y las pruebas de sistema, que comprenden las pruebas de seguridad, usabilidad, estrés y rendimiento, por solo citar las más utilizadas. Para ver las pruebas que se aplican o se prevé que se apliquen en los proyectos del CISED remitirse al Anexo 10. Pruebas que se aplican o se prevé que se apliquen en los proyectos del CISED.

- **Control de la calidad**

El **control de la calidad** en el CISED implica supervisar los resultados específicos del proyecto para identificar los modos de eliminar las causas de resultados insatisfactorios. Involucra de igual modo: la realización de un control sistemático marcado por los diferentes hitos definidos en el cronograma del avance del trabajo que actualiza cada miembro del equipo, la documentación de todos los cambios realizados y la existencia de una persona que apruebe dicho cambio, la comunicación de cada uno de los cambios efectuados a los miembros del equipo y los elementos afectados por este. Además, se lleva un registro versionado de todas las no conformidades encontradas en pruebas, revisiones y auditorías.

2.2 Propuesta de herramientas para pruebas de software

La propuesta de herramientas para automatizar pruebas de software en el CISED está realizada de acuerdo con un estudio realizado a las tecnologías, metodologías y arquitecturas utilizadas en dicho Centro. Debido a que ninguna herramienta responde específicamente a una metodología o arquitectura, la

propuesta está organizada de acuerdo a los niveles y técnicas de prueba existentes, y por cada uno de ellos, las tecnologías a las cuales se desea dar soporte para realizar pruebas automatizadas. Para ver una comparación de las herramientas propuestas con otras similares remitirse a Anexo 18. Comparación entre las herramientas propuestas con otras similares.

Durante la investigación fueron estudiadas una serie de aplicaciones que finalmente no se incluyeron dentro de la propuesta por alguna de las siguientes razones: sus funcionalidades ya están incluidas en otras herramientas, ausencia de interfaz gráfica, gran complejidad en el trabajo por consola, una alta curva de aprendizaje, y por último, muchas son propietarias.

A pesar de lo expresado anteriormente, muchas de esas herramientas deberían ser tenidas en cuenta por algunos proyectos en específico para su uso futuro, sobre todo las propietarias, por tratarse de aplicaciones muy completas, fiables y con un buen soporte técnico. Las dos listas siguientes están compuestas por aplicaciones recomendadas que no están en la propuesta:

Libres: Checkstyle, Cobertura, Open Load Tester, Sonar, TestLink y Web Inject.

Propietarias: Benchmark Factory, HP LoadRunner, HP Performance Center, HP QuickTest Professional, HP WebInspect, JTest, PerformaSure, Rational Robot, Spotlight y Testing Master.

Las pruebas de recuperación, usabilidad y compatibilidad no han sido cubiertas por la propuesta por tratarse de evaluaciones ambiguas que dependen en gran medida de la opinión de una persona y no de un software.

El esquema siguiente resume las herramientas propuestas de acuerdo a las técnicas de pruebas:

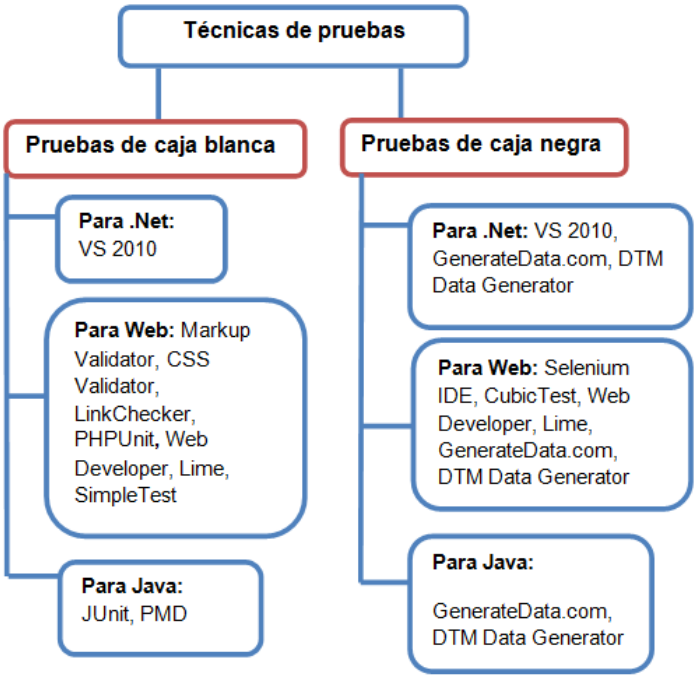

Figura 2.2.1. Herramientas para pruebas de caja blanca y caja negra (Fuente: Elaboración propia)

La tabla siguiente resume las herramientas propuestas de acuerdo a los niveles para pruebas:

Tabla 2.2.2. Herramientas propuestas para cada nivel de prueba (Fuente: Elaboración propia)

Niveles	Para .NET	Para Web	Para Java
Pruebas unitarias	VS 2010, VS 2008, NUnit	Selenium IDE, CubicTest, Markup Validator, CSS Validator, LinkChecker, Web Developer, PHPUnit, Lime, SimpleTest	JUnit, PMD
Pruebas de integración	VS 2010, VS 2008	Selenium IDE	
Pruebas de sistema	VS 2010, VS 2008	Selenium IDE,	

		CubicTest, Web Developer	
• Pruebas de seguridad	OpenVAS, Nmap	SQL Inject Me, Web Application Attack and Audit Framework, OpenVAS, Nmap	OpenVAS, Nmap
• Pruebas de rendimiento	VS 2010, VS 2008	JMeter	JProbe
• Pruebas de resistencia o estrés	VS 2010, VS 2008	JMeter	JProbe
• Pruebas de fiabilidad	VS 2010, VS 2008	JMeter, Selenium IDE, CubicTest	JProbe
• Pruebas de estructura	LinkChecker, Web Developer	LinkChecker, Web Developer	LinkChecker, Web Developer
Pruebas de aceptación	VS 2010, VS 2008	Markup Validator, CSS Validator, LinkChecker, Selenium IDE, JMeter, Web Developer	
Pruebas de regresión	Todas las herramientas para .NET propuestas	Todas las herramientas para Web propuestas	Todas las herramientas para Java propuestas

A continuación, la propuesta de herramientas para pruebas de software en el CISED:

- **JUnit**

JUnit es un marco de trabajo de código abierto que permite realizar pruebas unitarias de aplicaciones Java de forma 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.

JUnit es también un medio para 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.

- **Selenium IDE**

Selenium IDE es un complemento para el navegador Mozilla Firefox que permite realizar pruebas funcionales a aplicaciones web. Para hacer una prueba con esta herramienta es necesario efectuar un primer intento manualmente, mientras el programa es capaz de ir grabando las acciones sobre los formularios: introducir un texto, seleccionar una opción, dar clic en un botón aceptar, entre otras. Luego el probador o desarrollador tiene la oportunidad de repetir la prueba cuantas veces quiera, ya no de manera manual, sino de forma automática, pudiendo regular la velocidad y cambiando, si lo desea, los valores de los campos de entrada.

- **Web Developer**

Web Developer es una extensión para los navegadores Mozilla Firefox y Google Chrome destinada fundamentalmente a los diseñadores de aplicaciones web. Permite realizar pruebas no funcionales sobre cualquier tipo de página web. Cubre fundamentalmente las pruebas de compatibilidad y usabilidad al verificar elementos como CSS, formularios, imágenes, JavaScript, colores, redirecciones, enlaces, marcos, resolución y muchos otros.

Consiste en una barra de herramientas muy bien organizada en varios grupos de opciones que el desarrollador o probador puede usar para habilitar y deshabilitar objetos, comprobar la concordancia del código con los estándares, mostrar información relevante sobre las etiquetas, etc. Es distribuida bajo licencia GPL.

- **MarkUp Validator**

Es una aplicación web perteneciente al W3C que permite validar codificaciones HTML, XHTML y otros lenguajes de marcado. Está publicada bajo la licencia *W3C Software Notice and License,* la cual es compatible con GPL. Tiene tres formas de validar un código: accediendo directamente al sitio a probar, subiendo un archivo o escribiendo en un panel lo que se desea validar. El análisis que realiza está basado

en los estándares internacionales establecidos; ofrece información útil sobre las violaciones cometidas ayudando a los desarrolladores y probadores a detectar las no conformidades de una página web de acuerdo a las buenas prácticas de programación. Ofrece una interfaz amigable y fácil de usar, disponible desde cualquier ordenador con conexión a Internet.

- **CSS Validator**

Es una aplicación web perteneciente al W3C que permite validar hojas de estilo en cascada. Tiene licencia *W3C Software Notice and License*, que es compatible con GPL. Al igual que el MarkUp Validator, analiza el código de acuerdo a estándares internacionales; ofrece un reporte sobre las dificultades encontradas ayudando a detectar las no conformidades de una página web de acuerdo a las buenas prácticas de programación. Ofrece una interfaz amigable y fácil de usar, disponible desde cualquier ordenador con conexión a Internet.

- **LinkChecker**

LinkChecker es un software libre y de código abierto desarrollado por Bastian Kleineidan cuya única finalidad es comprobar los enlaces de los documentos HTML. Esta es una herramienta útil para realizar pruebas no funcionales en sitios de pequeño, mediano y gran tamaño debido a que se puede configurar la profundidad de la búsqueda de los enlaces, así como definir un número de hilos para su ejecución, aumentando la rapidez de la prueba.

LinkChecker soporta enlaces HTTP, HTTPS, FTP, al correo, archivos locales, etc. Es capaz de ignorar el protocolo de exclusión robots.txt contra los programas que simulan el acceso de usuarios a la web.

Durante la investigación fueron identificadas varias versiones de LinkChecker: una perteneciente al W3C con dos variantes: web y escritorio, y otra de escritorio que corre sobre Linux solamente. Esta última fue la seleccionada para integrar la propuesta, debido a su sencillez y bajo consumo de recursos de *hardware*.

- **PMD**

PMD[4] es una herramienta de análisis estático de código (analiza el código y no el programa en ejecución) en busca de estructuras potencialmente peligrosas tales como: posibles errores, código "muerto" (variables no accedidas, bloques de ejecución inalcanzables, etc.), código sub-óptimo y bloques con una estructura poco legible o más complicada de lo necesario.

El complemento PMD posee un conjunto de reglas estáticas, que escanea código Java e identifica problemas. Se integra con diversos IDE tales como: Eclipse, JBuilder, NetBeans, etc. Está bajo la licencia *Berkeley Software Distribution* (BSD) y fue desarrollado por SourceForge[5].

- **CubicTest**

CubicTest es un complemento de código abierto para Eclipse que permite el diseño de pruebas funcionales para aplicaciones web. La herramienta CubicTest proporciona controles y elementos de páginas web que permiten realizar comprobaciones más robustas y reutilizables.

Dentro de las funcionalidades de CubicTest se destacan:

- Grabación de pruebas: las acciones ejecutadas en Firefox (u Opera que también lo soporta).
- Creación de pruebas manualmente (para el caso de que se cree la prueba antes que la aplicación).
- Ejecución de pruebas automáticas.

- **JMeter**

Apache JMeter es una herramienta para pruebas, libre y escrita en Java, que permite realizar pruebas funcionales y de rendimiento sobre aplicaciones web.

JMeter puede simular una carga pesada en un servidor, red o un objeto para poner a prueba su resistencia o para analizar el rendimiento general en diferentes tipos de carga. Además, puede ayudar a probar la aplicación de regresión por lo que le permite crear *scripts* de prueba, facilita una rápida detección de cuellos de botella existentes debido al tiempo excesivo de respuesta, tiene una estructura en árbol que le da potencia, permitiendo que sea la imaginación de quien la use la que ponga los límites a la hora de

[4] PMD: no son siglas. Es el nombre tal cual.
[5] SourceForge: es una central de desarrollos de software que controla y gestiona varios proyectos de software libre.

diseñar el plan de prueba y muestra los resultados de las pruebas en una amplia variedad de informes y gráficas.

Algunos objetivos específicos de JMeter:

- Asegurar que ante una carga de trabajo determinada, las páginas a las que se accede responden en el intervalo de tiempo especificado por el grupo de diseñadores.
- Determinar el tiempo medio de respuesta que obtendrá el usuario.
- Determinar el número máximo de usuarios concurrentes que pueden acceder a una página específica, o transacciones por segundo que la aplicación es capaz de soportar.
- Identificar las páginas con una mayor desviación típica en sus tiempos de respuesta.
- **PHPUnit**

PHPUnit es un marco de trabajo escrito en PHP para automatizar pruebas unitarias de aplicaciones escritas en dicho lenguaje. Es libre, de código abierto y está disponible para su uso y descarga desde cualquier sistema operativo.

En esta propuesta se recomendará el uso del *framework* conjuntamente con el IDE NetBeans en sus últimas versiones, las cuales se integran perfectamente con PHPUnit, permitiendo programar y probar sistemas escritos en PHP de una manera sencilla.

También se recomienda que el entorno de pruebas usando PHPUnit esté configurado sobre Ubuntu. Esto es porque la instalación del marco de trabajo en Windows, junto a su correcta integración con el IDE se puede complicar cuando la conexión a Internet que tengan los probadores y/o desarrolladores sea a través de un proxy. En Linux, con solo instalar NetBeans, la Máquina Virtual de Java (MVJ) y PHPUnit, no habrá necesidad de configurar nada, a la vez que la generación de casos de pruebas será muy fácil.

- **SQL Inject Me**

SQL Inject Me es una herramienta de código abierto, publicada bajo la licencia GPL y utilizada para detectar vulnerabilidades a inyecciones SQL en aplicaciones web. Es una herramienta dirigida a desarrolladores, probadores y al personal de control de calidad, ligera, fácil de usar y en lugar de utilizar

un *proxy* como muchas herramientas web para realizar pruebas en aplicaciones, se integra directamente con Firefox.

SQL Inject Me trabaja mediante la presentación de los formularios HTML, sustituyendo el valor de los campos con cadenas que son representativos de un ataque de inyección SQL. Esta herramienta no intenta poner en peligro la seguridad del sistema a examinar sino que busca los puntos de entrada posibles para un ataque contra el sistema sin existir escaneo de puertos, detección de paquetes, *hacking* de contraseñas o ataques al cortafuego.

- **Web Application Attack and Audit Framework (W3AF)**

W3AF permite encontrar y explotar las vulnerabilidades de aplicaciones web. Es considerada un marco de trabajo para pruebas de intrusión web, liberada bajo la licencia GPL versión 2, está disponible en todos los sistemas operativos populares. Está desarrollado en Python.

Web Application Attack and Audit Framework está diseñado para ser utilizado en auditorías a entornos web. Puede utilizarse por expertos en seguridad que no sean necesariamente programadores y por investigadores de vulnerabilidades o de productos de seguridad. La herramienta consta de varios perfiles y complementos que permiten realizar numerosas pruebas de manera sencilla y rápida.

La arquitectura de W3AFse divide en dos partes principales:

1- El centro, que coordina el proceso y proporciona características que son utilizadas por los complementos, que encuentran las vulnerabilidades y las explotan.

2- Los complementos, los cuales están conectados y comparten información entre ellos utilizando una base de conocimientos.

Cada conjunto de *plugins* está asociado a perfiles que vienen incluidos por defecto en la herramienta, tales como: *OWASP Top 10*, *Fast Scan*, *Full Audit* y *Full Audit Manual Disc*.

- **Microsoft Visual Studio**

Microsoft Visual Studio (VS) es empleado por varios proyectos del Centro en sus versiones **.NET 2003 Enterprise Architect**, **2005 Professional**, **2008 Team System** y **2010 Ultimate**.

Microsoft Visual Studio .NET 2003 no tiene soporte para pruebas en ninguna de sus versiones. **Microsoft Visual Studio 2005** permite realizar pruebas, pero es en la versión **Team System,** específicamente: pruebas unitarias, pruebas web, pruebas de carga, pruebas manuales, pruebas genéricas y pruebas ordenadas. Debido a esa inconveniencia, es que la herramienta **NUnit** está incluida en esta propuesta.

Microsoft Visual Studio 2008 Team System soporta pruebas unitarias, pruebas manuales, pruebas web, pruebas de carga (que se divide en pruebas de estrés, pruebas de rendimiento y pruebas de capacidad de planeación), pruebas genéricas y pruebas ordenadas. Adicionalmente, tiene varias herramientas que permiten la gestión de las pruebas, entre las que se encuentran **Test View**, **Test List Editor** y **Test Results**.

Microsoft Visual Studio 2010 en todas sus versiones permite realizar distintas pruebas a través de las siguientes herramientas que están integradas a la *suite*: **Test View**, **Test List Editor**, **Test Results**, **CodeCoverageResults**, **Test Runs** y **Test Impact View**. Las pruebas que se pueden llevar a cabo con ellas son: pruebas unitarias, pruebas unitarias de base de datos, pruebas de interfaz de usuario (IU) codificadas, pruebas de carga (dentro de la cual están las pruebas de rendimiento web) y pruebas genéricas. La versión de VS en la que están presentes todas las funcionalidades de pruebas es la **Ultimate**.

Las pruebas de integración no están explícitamente soportadas, pero a través de un conjunto de pruebas unitarias, pruebas ordenadas, pruebas web y pruebas genéricas, se puede probar la integración de varios módulos.

Como se ha visto, en todas las versiones el soporte para pruebas es diferente. La versión que se propone para el CISED es **Visual Studio 2010 Ultimate**.

- **OpenVAS**

OpenVAS es el acrónimo de *Open Vulnerability Assessment System*, un completo escáner de vulnerabilidades que permite evaluar los riesgos de seguridad en los equipos de una red y cerrar sus debilidades proactivamente utilizando otras herramientas libres. Es un software libre licenciado con GPL, inspirado en Nessus, programa que luego de varias versiones fue convertido en propietario, dejando el

camino abierto para que OpenVAS fuera la nueva alternativa a las pruebas de seguridad mediante herramientas que usan el patrón arquitectónico cliente-servidor.

OpenVAS posee una interfaz gráfica a modo de cliente y un servidor con un conjunto de pruebas para detectar problemas de seguridad en sistema remotos y aplicaciones; esto tiene la ventaja de que varias estaciones de trabajo pueden conectarse al servidor OpenVAS con los clientes y ejecutar las pruebas desde un lugar centralizado, ahorrándose el trabajo de tener que configurar las pruebas localmente, contando con una interfaz sencilla en idioma español. Hoy esta herramienta es un referente obligado a la hora de realizar pruebas se seguridad utilizando software libre.

- **NUnit**

NUnit es un marco de trabajo que realiza pruebas unitarias, desarrollado para escribir y ejecutar pruebas en .NET. Está desarrollado en C# y pertenece a la familia de *frameworks* xUnit. NUnit, es una herramienta de código abierto, acogida entre los desarrolladores .NET por las siguientes características:

- Es multiplataforma, pudiendo ejecutarse en Windows y Linux.
- Provee dos formas de ejecutar y administrar las pruebas: una interfaz gráfica y una interfaz de consola.
- Ofrece una interfaz simple que informa si una prueba o un conjunto de pruebas fallaron o pasaron.
- Presenta dos tipos de funcionamiento diferentes:
    - Atributos personalizados: atributos que le indican a NUnit cómo interpretar y ejecutar las pruebas implementadas en el método o clase.
    - Aserciones: son métodos del marco de trabajo de NUnit utilizados para comprobar y comparar valores.

NUnit está licenciado con *BSD-style*. Tiene varios complementos para ampliar su aplicación, entre los que se encuentran: NUnit.Forms, para que sea capaz de manejar pruebas de elementos de interfaz de usuario en *Windows Forms* y NUnit.ASP, para manejar pruebas de elementos de interfaz de usuario en ASP.NET.

- **Nmap**

Nmap es un programa de código abierto con licencia GPL que sirve para efectuar rastreo de puertos. Se usa para evaluar la seguridad de sistemas informáticos, así como para descubrir servicios o servidores en

una red informática. Es una herramienta imprescindible para todo administrador de sistemas; es usado para pruebas de penetración y tareas de seguridad informática en general. También se emplea para identificar computadoras de una red, determinar qué sistema operativo y versión utilizan las máquinas descubiertas, encontrar características del *hardware* de red del ordenador objeto de prueba, etc. Es muy simple de usar, necesita bajas prestaciones de *hardware*, es multiplataforma y tiene varias interfaces de usuario; la oficial es Zenmap.

- **JProbe**

JProbe es una solución completa de optimización del rendimiento para entornos Java que permite, en un tiempo mínimo, descubrir, diagnosticar y resolver incidencias de bajo rendimiento en aplicaciones JEE (*Java Enterprise Edition*) y JSE (*Java Standard Edition*) descendiendo su nivel de análisis hasta la línea de código individual. Es una herramienta de análisis de desempeño Java que proporciona una solución integral para ayudar a los desarrolladores a descubrir las causas y corregir los problemas de ejecución y memoria en las aplicaciones Java. JProbe permite probar sus aplicaciones sin ningún cambio en el código y se integra fácilmente con servidores de aplicaciones, IDE, JDK (*Java Development Kit*) y sistemas operativos.

JProbe ofrece tres tipos de análisis:

Análisis de memoria: permite a los desarrolladores identificar y resolver problemas de uso de memoria, para garantizar la eficiencia y la estabilidad de los programas.

Análisis de rendimiento: permite a los desarrolladores identificar y resolver cuellos de botella y puntos muertos, para garantizar la óptima ejecución y escalabilidad de los programas.

Cobertura de análisis: permite a los desarrolladores identificar líneas de no ejecución de código durante las pruebas de unidad, para garantizar la cobertura de la prueba y corrección del programa.

- **SimpleTest**

SimpleTest es un *framework* para realizar pruebas unitarias a proyectos desarrollados en PHP. Drupal tiene integrado un módulo con SimpleTest y un conjunto de pruebas unitarias predefinidas, las cuales se pueden ejecutar desde el propio CMS. Este marco de trabajo añade varias herramientas para la

realización de las pruebas; se pueden cambiar variables de configuración, establecer permisos a usuarios, etc. Existen casos de prueba creados para casi todos los elementos de un sitio desarrollado en Drupal; algunos de ellos son: *blogs*, comentarios, conexiones con la base de datos, gestión de usuarios, perfiles y otros aspectos.

- **Lime**

Symfony incluye su propio *framework* para pruebas unitarias llamado Lime, el que proporciona soporte para las pruebas unitarias; es más eficiente que otros marcos de trabajo de pruebas para PHP y tiene las siguientes ventajas:

1- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas. No todos los *frameworks* de pruebas garantizan un entorno de ejecución "limpio" para cada prueba.
2- Las pruebas de Lime son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados de Lime utilizan diferentes colores para mostrar de forma clara la información más importante.
3- Symfony utiliza Lime para sus propias pruebas y su "*regression testing*", por lo que el código fuente de Symfony incluye muchos ejemplos reales de pruebas unitarias y funcionales.
4- El núcleo de Lime se valida mediante pruebas unitarias.
5- Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta únicamente de un archivo, llamado "*lime.php*", y no tiene ninguna dependencia.
6- Las pruebas que se muestran en las secciones siguientes utilizan la sintaxis de Lime, por lo que funcionan directamente en cualquier instalación de Symfony.

Nota: Las pruebas unitarias y funcionales no están pensadas para lanzarlas en un servidor de producción. Se trata de herramientas para el programador, por lo que solamente deberían ejecutarse en la máquina de desarrollo del programador y no en un servidor de producción. (Potencier, y otros, 2008)

2.3 Herramientas de apoyo

Existen herramientas que no son para realizar pruebas específicamente, pero tienen gran utilidad para llevarlas a cabo.

- **GenerateData.com**

GenerateData.com es una aplicación web libre escrita en PHP, JavaScript, AJAX y HTML que sirve para generar datos automáticamente para la base de datos MySQL. Es muy fácil de instalar y configurar. Su uso es muy intuitivo. Su utilidad radica en que con ella se puede llenar una base de datos con cualquier cantidad de tablas y filas, especificando el tipo de dato de cada columna (nombre, número de teléfono, correo electrónico, ciudad, estado, provincia, país, fecha, dirección y otros). Los datos se pueden exportar en XML, Excel, HTML, CSV o SQL. Contiene una pequeña ayuda para rellenar los campos; permite guardar y cargar los formularios generados.

- **DTM Data Generator**

DTM Data Generator es una utilidad simple, poderosa y totalmente personalizable que genera datos de base de datos con fines de prueba. La gran ventaja de este software es que es capaz de crear una amplia variedad de tablas de prueba y que admite plantillas definidas por el usuario; estos datos pueden insertarse fácilmente en una base de datos. La configuración para cada tabla está controlada por separado. Es capaz de crear instrucciones SQL para cualquier operación. Soporta PostgreSQL, Oracle, MySQL, Microsoft SQL Server y otras bases de datos poco relevantes. (Softpedia.com, 2011)

DTM Data Generator es propietaria, su versión de pruebas genera sólo 10 registros por tabla. Una alternativa a ella es **Data Generator for PostgreSQL,** la cual es utilizada en la UCI y es también propietaria; como su nombre lo indica es solo para PostgreSQL y genera 100 registros por tabla en su versión de prueba.

2.4 Conclusiones

Durante la elaboración del capítulo fue caracterizado el PDS en el CISED, aportando las especificidades necesarias para la elaboración consecuente de una propuesta que incluyera herramientas para la automatización de las pruebas de software y proporcionara una mejora a la actividad de pruebas.

La propuesta de herramientas para pruebas automatizadas de software permitirá darle solución a diversas problemáticas existentes; constituirá una vía para agilizar el trabajo tanto de desarrolladores como de probadores y aseguradores de la calidad. Se podrán realizar múltiples cálculos que medirán la calidad del código implementado y será posible seguir buenas prácticas y estándares de una manera más sencilla.

Los casos de prueba de caja negra para web podrán ser almacenados para reutilizarse en versiones posteriores de las aplicaciones.

Fueron propuestos programas para pruebas de software que se integran a varias tecnologías usadas en el CISED.

CAPÍTULO 3

SISTEMA DE GESTIÓN DE HERRAMIENTAS PARA PRUEBAS DE SOFTWARE

Para dar cumplimiento al objetivo “desarrollar una aplicación que permita la gestión de herramientas para pruebas de software y la información concerniente a ellas”, se utilizará la metodología de desarrollo de software *Extreme Programming*, mejor conocida como XP.

XP pertenece a un nuevo grupo de metodologías ágiles que han tenido auge en los últimos años, debido a la presentación de nuevas formas para desarrollar aplicaciones en un corto lapso de tiempo y con poco personal. Por la sencillez de la aplicación que se desea desarrollar, XP es la alternativa ideal.

El objetivo de este capítulo es mostrar el progreso del desarrollo de la aplicación cuyo nombre será **Hepsoft**.

XP define seis fases: Exploración, Planificación, Iteraciones a primera liberación, Producción, Mantenimiento y Muerte (Benk, 1999); por tanto, se expondrá lo realizado durante las cuatro primeras fases, necesarias para desarrollar la aplicación. Se explicará el diseño de la solución, sus principales componentes, las relaciones entre ellos y los artefactos más importantes generados.

3.1 Metáfora

La aplicación **Hepsoft** permitirá gestionar las herramientas para pruebas de software del CISED. Es un sitio web al cual puede acceder cualquier usuario y consultar las aplicaciones que para estos fines se utilizan en cada uno de los proyectos del Centro, además de visualizar los tipos de pruebas y tecnologías asociadas a estas aplicaciones.

Un conjunto de datos importantes se mostrarán por cada herramienta, permitiendo acceder a una ayuda sobre su uso. Cualquier usuario autenticado podrá proponer aplicaciones para estos fines, teniendo la posibilidad de introducir los mismos datos que normalmente tienen las entradas del catálogo.

El sistema tendrá un administrador, que contará con permisos para gestionar cualquier información, incluyendo la aceptación de las propuestas de nuevas herramientas que hagan los usuarios.

3.2 Conceptos asociados al dominio del problema

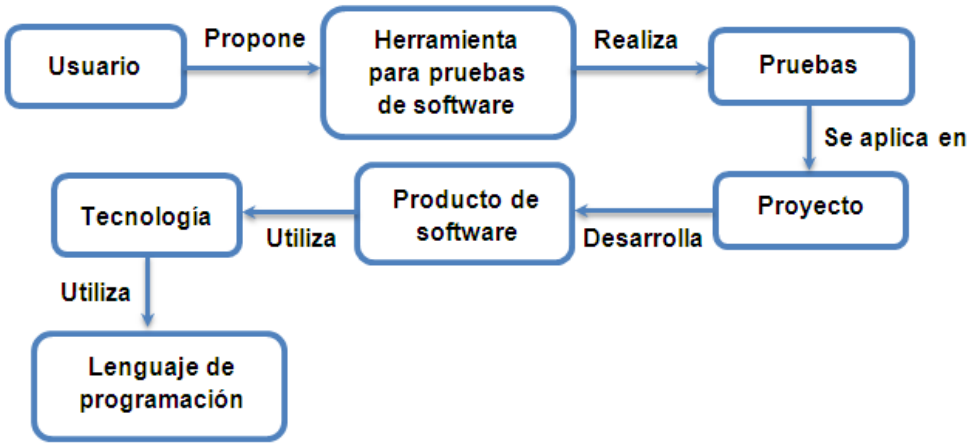

Figura 3.2.2. Mapa conceptual del problema (Fuente: Elaboración propia)

A continuación se relacionan los conceptos más relevantes para entender el funcionamiento de la aplicación:

Departamento: estructura organizativa del CISED dentro de la cual se ubican proyectos relacionados.

Herramienta para pruebas de software: aplicaciones que permiten la automatización de las pruebas de software.

Lenguaje de programación: es una manera estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Tiene un conjunto de reglas sintácticas y semánticas que definen a un programa informático.

Producto de software: es una aplicación informática, programa informático o cualquier componente que pueda ser usado en un dispositivo de cómputo, con o sin fines de lucro.

Proyecto: conjunto de actividades que buscan cumplir con un objetivo específico que debe ser alcanzado en un plazo de tiempo previamente definido respetando un presupuesto.

Tipo de prueba: pruebas que verifican las características de calidad de un programa informático, como el rendimiento, el volumen de información, entre otras.

Tecnología: es el conjunto de conocimientos que permiten fabricar objetos. En este contexto se ha utilizado el concepto para referirse a librerías, marcos de trabajo, lenguajes de programación y otros elementos relevantes en una prueba de software.

Usuario: persona que accede a la aplicación web a través de una computadora.

3.3 Requisitos no funcionales

Para ver los requerimientos no funcionales de la aplicación ver Anexo 4. Requerimientos no funcionales.

3.4 Historias de usuario y responsabilidades

Uno de los artefactos más importantes que genera la metodología XP son las historias de usuario (HU). Son escritas por el propio cliente, tal y como ven ellos las necesidades del sistema, por tanto, son descripciones cortas y escritas en el lenguaje del usuario sin terminología técnica. Las HU conducen a las pruebas de aceptación, las cuales validarán que las funcionalidades se hayan implementado correctamente.

Para el desarrollo de la aplicación, se considerarán las técnicas y niveles de prueba como datos no modificables ni eliminables, debido a que siempre se aplican de alguna forma en los proyectos. Los tipos de prueba son gestionables, pues su elección es muy variable.

A continuación la HU Administrar herramienta. Para ver el resto de las historias de usuario remitirse al Anexo 11. Historias de usuario.

Tabla 3.4.3. HU_3 Administrar herramienta (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_3	**Usuario:** Administrador
Nombre historia: Administrar herramienta	

Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Yuniel Martínez González	

Descripción: Un administrador podrá elegir, introducir una herramienta, modificarla o eliminarla, siempre brindando los datos requeridos por la aplicación, en este caso:

- Nombre*
- Desarrollador*
- Última versión estable*
- Año*
- Idioma* (Inglés/Español/Otro/Multilenguaje)
- Tipo de software* (Libre/Propietario)
- Licencia*
- Lenguaje de desarrollo*
- Sitio web oficial*
- Tipo de prueba*
- Lenguajes soportados*
- Complementos opcionales
- Tamaño del archivo*
- Sistema operativo* (Linux/Windows/Multiplataforma)
- Requerimientos de software*
- Requerimientos de *hardware**
- Ayuda (tutorial)
- Palabras claves*
- Autor de la propuesta*
- Aceptada (Sí/No)*

* Campos obligatorios

Al modificar o eliminar se muestra una lista con los siguientes datos: Nombre, Tipo de software, Sistema Operativo, Aceptada (Sí/No).

Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error.

Responsabilidades

La metodología XP no define expresamente actores para desarrollar responsabilidades dentro de un sistema, pero es válido aclarar que el sistema tendrá tres tipos de usuarios que accederán a él:

Invitado: solo podrá ver la información del sitio, pero no podrá gestionar nada.

Usuario: podrá ver la información y proponer herramientas.

Administrador: podrá consultar la información, gestionar las herramientas y su información, así como administrar los usuarios y sus responsabilidades.

En la siguiente tabla se pueden ver las responsabilidades que podrá desempeñar cada usuario que acceda al sistema.

Tabla 3.4.4. Trazabilidad Responsable/HU (Fuente: Elaboración propia)

Responsable	Invitado	Usuario	Administrador
Historia(s) de Usuario(s)	Buscar herramienta	Buscar herramienta Proponer herramienta Autenticar usuario Administrar mensajes de contacto (enviar mensaje solamente)	Buscar herramienta Administrar departamento Administrar tipo de prueba Administrar herramienta Administrar proyecto Administrar lenguaje Autenticar usuario Administrar usuario Administrar mensajes de contacto Administrar tecnología

Estimación del esfuerzo por historia de usuario

En XP, durante la fase de planificación, se realiza una estimación del esfuerzo que costará implementar cada historia de usuario; este valor se expresará en días. Con el transcurso de las iteraciones, se irá acercando a la realidad. Los resultados obtenidos en esta estimación se exponen en la siguiente tabla:

Tabla 3.4.5. Estimación del esfuerzo por HU (Fuente: Elaboración propia)

Historia de Usuario	Estimación (en días)
Autenticar usuario	2
Administrar usuario	7
Administrar herramienta	7
Administrar proyecto	7
Administrar departamento	7
Administrar tipo de prueba	7
Administrar lenguaje	7
Buscar herramienta	7
Proponer herramienta	2
Administrar mensajes de contacto	2
Administrar tecnología	1

3.5 Plan de iteraciones

Como parte del ciclo de vida de un proyecto se crea el Plan de iteraciones, con el objetivo de mostrar la duración y el orden en que serán implementadas las HU dentro de cada iteración. Para la solución se han definido once HU divididas en 2 iteraciones, para una duración total del proyecto de 8 semanas. Ver Anexo 12. Plan de iteraciones.

3.6 Plan de entregas

Luego de concluir con la elaboración de las historias de usuario, se crea el Plan de entregas para estimar el tiempo de desarrollo de las mismas. Este artefacto se construye para determinar qué período de tiempo puede tardar la implementación de cada una de las historias, definiéndose las fechas en que serán liberadas las versiones funcionales del producto.

Tabla 3.6.6. Plan de entregas (Fuente: Elaboración propia)

Entregable	Fin iteración 1	Fin iteración 2
Hepsoft	12 de mayo del 2011	30 de mayo del 2011

3.7 Diseño

Arquitectura

Hepsoft basa su implementación en el patrón Arquitectura en Capas, específicamente en el Modelo-Vista-Controlador (MVC), el cual es muy común en las aplicaciones web. Esta herramienta está desarrollada con el marco de trabajo CodeIgniter, que da soporte al MVC.

El modelo representa la estructura de datos. Las clases modelo contienen funciones para obtener, insertar, eliminar y actualizar información en la base de datos. La vista es la información presentada al usuario; normalmente será una página web, pero en CodeIgniter, una vista también puede ser un fragmento de una página como un encabezado, pie de página o RSS (*Really Simple Syndication*). El controlador sirve como un intermediario entre el modelo, la vista y cualquier otro recurso necesario para procesar la petición HTTP y generar una página web. (Lozano, 2006)

Estructura de CodeIgniter

El siguiente gráfico ilustra el flujo de los datos a través del sistema:

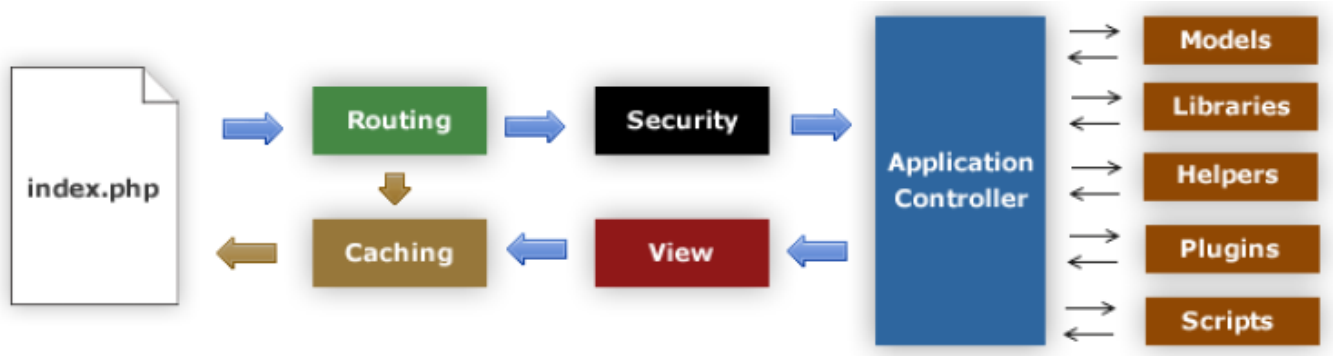

Figura 3.7.3. Flujo de datos a través de CodeIgniter (Fuente: Lozano, 2006)

El archivo **index.php** sirve como controlador frontal, inicializando los recursos básicos necesarios para ejecutar CodeIgniter.

En ***Routing*** se examina la petición HTTP para determinar qué se debe hacer.

Si un archivo de caché existe (***Caching***), es enviado directamente al explorador, sobrepasando el sistema de ejecución normal.

Seguridad (***Security***). Antes que el controlador sea cargado, la petición HTTP y cualquier dato suministrado por el usuario son filtrados por seguridad.

El controlador (***Application Controller***) carga los modelos (***Models***), librerías (***Libraries***), complementos (***Plugins***) y cualquier otro recurso (***Helpers***, ***Scripts***, etc.) necesario para procesar la petición específica.

La Vista (***View***) es presentada y enviada al explorador web para ser visualizada. Si el cacheo está habilitado, la vista es cacheada primero para que las peticiones subsecuentes puedan ser servidas.

Tarjetas CRC

Otro de los artefactos que genera XP son las tarjetas CRC (Clase, Responsabilidad y Colaboración), las cuales permiten al programador centrarse y apreciar el desarrollo orientado a objetos, olvidándose de los malos hábitos de la programación clásica. Para el desarrollo de la aplicación se decidió crear una tarjeta CRC por cada modelo implementado (clase de la capa Modelo del patrón Modelo-Vista-Controlador), debido a que estas representan las funcionalidades principales del sitio. Ver Anexo 14. Tarjetas CRC.

Modelo Entidad-Relación

A continuación el Modelo Entidad-Relación (MER) de la base de datos Hepsoft:

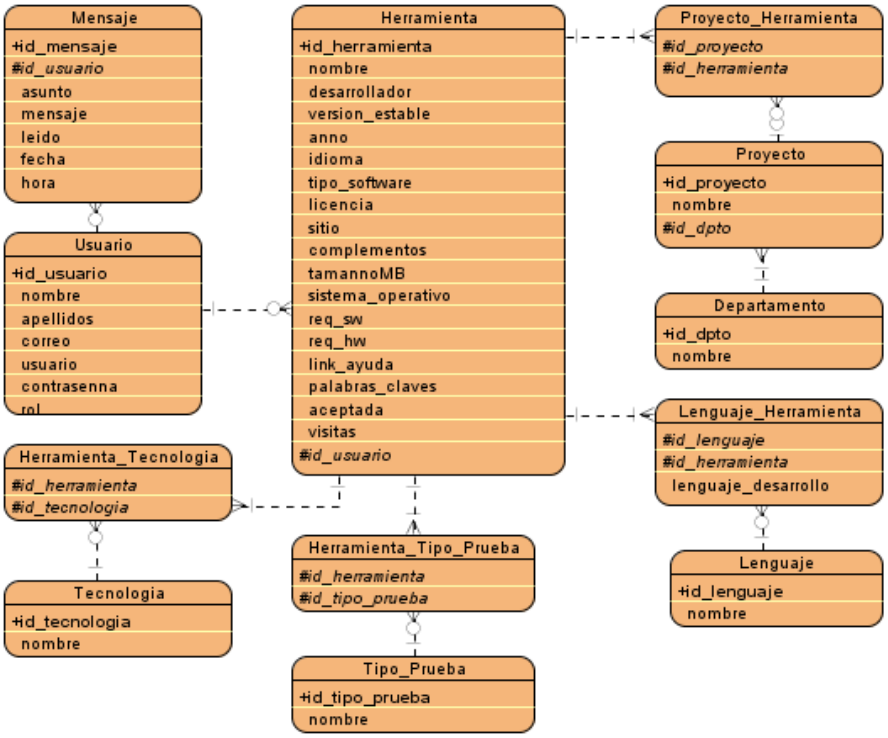

Figura 3.7.4. Modelo Entidad-Relación (Fuente: Elaboración propia)

Diagrama de despliegue

La aplicación estará desplegada en un solo servidor junto con la base de datos. Las estaciones de trabajo se conectarán a la aplicación mediante el protocolo HTTP.

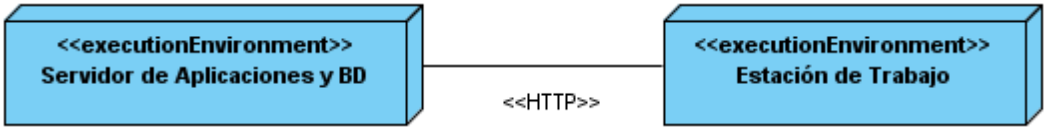

Figura 3.7.5. Diagrama de despliegue (Fuente: Elaboración propia)

3.8 Implementación del sistema

Iteración 1

En la primera iteración se desarrollan las historias de usuarios de mayor prioridad; esto tiene como objetivo obtener una primera versión del producto con las principales características o funcionalidades

para ser mostrado al cliente. En XP el trabajo de cada iteración se divide en **tareas de ingeniería**, las que especifican las acciones llevadas a cabo por los programadores en cada historia de usuario, puesto que estas no son lo suficientemente detalladas para saber qué implementar. De acuerdo al Plan de iteraciones, las historias de usuario se agruparon en dos iteraciones. A continuación se muestran las tareas de ingeniería derivadas de cada historia de usuario a desarrollar en la primera iteración:

Tabla 3.8.7. Tareas de ingeniería por cada HU de la primera iteración (Fuente: Elaboración propia)

Historia de usuario	Tarea
Administrar departamento	- Crear departamento - Mostrar listado de departamentos - Modificar los datos del departamento - Eliminar departamento
Administrar tipo de prueba	- Crear prueba - Mostrar listado de pruebas - Modificar los datos de la prueba - Eliminar prueba
Administrar herramienta	- Registrar herramienta - Mostrar listado de herramientas - Modificar herramienta - Eliminar herramienta
Administrar proyecto	- Registrar proyecto - Mostrar listado de proyectos - Modificar los datos del proyecto - Eliminar proyecto
Administrar lenguaje	- Registrar lenguaje - Mostrar listado de lenguajes - Modificar los datos del lenguaje - Eliminar lenguaje

Iteración 2

En la segunda iteración se implementarán las historias de usuarios menos importantes para el cumplimiento de los objetivos de la aplicación.

Tabla 3.8.8. Tareas de ingeniería por cada HU de la segunda iteración (Fuente: Elaboración propia)

Historia de usuario	Tarea
Autenticar usuario	- Desarrollar la autenticación del sistema
Administrar usuario	- Registrar usuario - Mostrar listado de usuarios - Modificar los datos del usuario - Eliminar usuario
Buscar herramienta	- Desarrollar búsqueda simple - Desarrollar búsqueda avanzada
Proponer herramienta	- Proponer herramienta
Administrar mensajes de contacto	- Enviar mensaje - Ver mensaje - Eliminar mensaje
Administrar tecnología	- Registrar tecnología - Mostrar listado de tecnologías - Modificar tecnología - Eliminar tecnología

Para ver los detalles de cada tarea de ingeniería remitirse al Anexo 13. Tareas de ingeniería.

3.9 Interfaces de la aplicación

A continuación se muestran algunas interfaces de la aplicación:

Figura 3.9.6. Página de inicio de la aplicación (Fuente: Elaboración propia)

En la figura anterior se puede apreciar la página de inicio de la aplicación. El sitio tiene un menú superior a través del cual se puede acceder rápidamente a varios listados de acuerdo a las opciones seleccionadas. Existe un menú para las técnicas de prueba y otro para los niveles de prueba, dentro de las cuales se puede acceder a las tecnologías registradas. Si se da clic sobre alguno de esos submenús se mostrarán las herramientas para pruebas correspondientes.

También en el menú superior se puede acceder a “Proponer herramienta”, una vía para que los usuarios registrados puedan proponer el uso de una herramienta que consideren importante. La página “Contáctenos” permite enviar mensajes con dudas, sugerencias, etc.; los administradores pueden ver los

nuevos mensajes y eliminarlos. Por último, la página de “Ayuda” brinda información útil sobre el uso del sitio, preguntas frecuentes y temas relacionados con la aplicación.

En la parte izquierda de la página se observa un bloque llamado “Mi cuenta”, el cual sustituye al clásico formulario de autenticación que tienen los sitios donde los usuarios se pueden registrar y luego autenticarse, algo disponible también en esta aplicación. Debajo se encuentra un bloque con las diez aplicaciones más vistas ordenadas descendentemente. Por último, se encuentran un conjunto de enlaces relacionados con el uso de la aplicación.

En la parte derecha se puede apreciar la existencia de un campo para realizar una búsqueda general. Si se accede al botón “Avanzada”, se muestran un conjunto de atributos pertenecientes a las herramientas, mediante los cuales se puede refinar la búsqueda.

Figura 3.9.7. Resultados de una búsqueda (Fuente: Elaboración propia)

Debajo del bloque “Buscar”, se encuentran las opciones administrativas del sitio, las cuales son accesibles solo por los administradores. Esas funcionalidades son: Registrar, Modificar y Eliminar Tecnologías, Usuarios, Tipos de Pruebas, Departamentos, Proyectos, Lenguajes y Herramientas.

3.10 Pruebas

Durante el desarrollo se realizan pruebas para verificar que se está cumpliendo con lo documentado en las historias de usuario. Para ello es necesario repetir las pruebas con el objetivo de chequear la manera en que los cambios afectan las funcionalidades.

La metodología XP propone realizar pruebas unitarias y pruebas de aceptación.

Pruebas unitarias

XP propone realizar pruebas unitarias frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Una buena práctica es escribir el código de las pruebas previo a la implementación de las historias de usuario, pero esto se puede complicar cuando estas han sido descritas de manera ambigua.

Planificación: fueron planificadas dos iteraciones de pruebas por cada iteración de desarrollo. Se planificó la realización de cuatro pruebas por cada funcionalidad (tarea de ingeniería), aplicando individualmente cuatro herramientas para pruebas.

Se utilizó Selenium IDE, un complemento para el navegador Mozilla Firefox que ayuda a realizar, de una manera sencilla, pruebas funcionales a aplicaciones web. También se realizaron pruebas usando Web Developer y LinkChecker para validar el uso de los estándares web, los enlaces, atributos de las imágenes, entre otros elementos. SQL Inject Me, que se empleó para realizar inyecciones SQL a los formularios. Luego de codificar cada funcionalidad, fueron ejecutadas las pruebas planificadas y los errores detectados fueron corregidos para, posteriormente, repetir las mismas pruebas.

Resultados de las pruebas unitarias

Una cuantificación de las no conformidades detectadas en cada iteración de pruebas y una imagen que muestra la prueba realizada por una herramienta, se pueden ver en el Anexo 16. Resultados de las pruebas.

La grabación de diferentes operaciones sobre los formularios con Selenium IDE, permitió repetir esas mismas pruebas luego de realizar varias modificaciones a las mismas funcionalidades probadas; esto fue útil para detectar campos que no se habían validado, ahorrando tiempo en la realización de la prueba.

Además, se detectó la incompatibilidad de una propiedad HTML con el navegador Mozilla Firefox; para resolver este problema se buscó mostrar los datos de otra manera.

El uso de LinkChecker permitió comprobar los enlaces del sitio, detectando rápidamente errores en las direcciones especificadas en la implementación. Esta herramienta sirvió también para comprobar la seguridad del marco de trabajo contra mecanismos automáticos, evitando el acceso a directorios no permitidos dentro de la carpeta raíz de la aplicación.

Con SQL Inject Me se efectuó un ataque masivo con todas las pruebas de la herramienta a cada uno de los formularios de la aplicación, detectándose dos fallas en dos consultas que fueron corregidas utilizando de manera correcta las funciones del marco de trabajo para obtener elementos de la base de datos. Varias funcionalidades de Web Developer fueron usadas para comprobar los elementos visuales del sitio.

Pruebas de aceptación

Las pruebas de aceptación comprueban que las historias de usuario han sido implementadas correctamente al final de cada iteración. Por cada HU se pueden definir todas las pruebas de aceptación necesarias para asegurar su correcto funcionamiento. El objetivo final de éstas es garantizar que los objetivos han sido cumplidos y que el sistema es aceptable. (Veigas Chkout, y otros, 2010)

Planificación: para el desarrollo de esta fase, se definió una nueva forma de representación de los casos de prueba, que incluye una síntesis de los datos requeridos por la plantilla “Diseño de Casos de Prueba HU” del Expediente de Proyecto vigente en la UCI. Dicho documento cubre el conjunto de pruebas funcionales relacionadas con una HU. Se planificó la realización de dos iteraciones de pruebas al final de la implementación de todas las historias de usuario.

Para ver los casos de pruebas de aceptación ir al Anexo 15. Casos de prueba de aceptación.

Resultados de las pruebas de aceptación

Una cuantificación de las no conformidades detectadas en cada iteración de pruebas se puede ver en el Anexo 16. Resultados de las pruebas.

Fueron realizadas todas las pruebas planificadas, detectándose varios errores; los más relevantes fueron: mensajes de error no adecuados de acuerdo a la funcionalidad probada, campos no validados y funcionalidades administrativas que no comprobaban el rol del usuario autenticado. Las dificultades fueron corregidas, ejecutándose pruebas de regresión para asegurar la conformidad con las historias de usuario.

3.11 Conclusiones

Luego de terminadas las fases de exploración, planificación, iteraciones a primera liberación y producción se concluye que:

- El desglose de las historias de usuario en tareas de la ingeniería permitió facilitar la implementación de las funcionalidades.
- El desarrollo guiado por pruebas aseguró el cumplimiento de los objetivos trazados en las historias de usuario.
- La utilización del marco de trabajo CodeIgniter facilitó la correcta implementación del patrón arquitectónico Modelo-Vista-Controlador, asegurando la aplicación contra ataques comunes.
- El sistema, a partir de los resultados obtenidos durante la fase de prueba, se encuentra listo para desplegarse.

CONCLUSIONES GENERALES

Este trabajo finaliza con la obtención de una propuesta de herramientas para pruebas de software automatizadas en el Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI), y una aplicación web que permite gestionar la información concerniente a dichas herramientas. Ambos resultados han sido realizados de manera flexible ante posibles cambios en el Proceso de Desarrollo de Software (PDS) de la organización donde se aplica, proponiendo mayormente tecnologías libres acordes con las políticas que existen en el país y en la UCI para lograr una independencia tecnológica.

La investigación sobre el software de pruebas existente en el mercado mundial y la caracterización del PDS en el CISED, fueron vitales para definir una propuesta de herramientas que permita automatizar las pruebas de software en la entidad. Esta propuesta deberá facilitar la realización de los procesos de verificación y validación dentro del CISED, agilizando el desarrollo de los productos y permitiendo obtener resultados de mayor calidad.

Hepsoft, la aplicación web creada para gestionar el software de pruebas, podrá ayudar a desarrolladores y personal de aseguramiento de la calidad del Centro, a realizar la automatización de las actividades de prueba, contando con un medio confiable para documentarse sobre los programas que necesitan de acuerdo a los lenguajes, entornos de desarrollo, marcos de trabajo y otras tecnologías utilizadas en los proyectos donde trabajan.

RECOMENDACIONES

Luego de finalizado el presente trabajo, aunque se han cumplido los objetivos trazados inicialmente, se debe tener en cuenta, que tanto la propuesta de herramientas para pruebas de software como la aplicación web para gestionar este tipo de programas, pueden ser perfeccionadas y actualizadas en un futuro para adecuarse a los cambios que ocurran en la organización para la cual se han creado.

Para lograr una optimización de los resultados, se recomienda:

- Evaluar la posibilidad de incluir en la propuesta de herramientas la Plataforma de prueba para algoritmos biométricos desarrollada en el CISED.
- Teniendo en cuenta las opiniones de los usuarios, agregar nuevas funcionalidades a la aplicación.
- Integrar la aplicación con el portal del CISED.

REFERENCIAS BIBLIOGRÁFICAS

Benk, Kent. 1999. *Extreme Programming Explained*. 1999.

Bonilla Sánchez, Beatriz. 2006. Adictos al Trabajo. *Adictos al Trabajo.* [En línea] 31 de 03 de 2006. [Citado el: 23 de Enero de 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=cmmi2y3.

Booch, Grady, Rumbaugh, James y Jacobson, Ivar. 2000. *El Lenguaje Unificado de Modelado. Manual de referencia.* Madrid : Pearson Educación, 2000. 84-7829-036-2.

Calero Solís, Manuel. 2003. *Una explicación de la programación extrema (XP).* Madrid : s.n., 2003.

Chávez Moreno, Oswaldo Daniel. 2011. Slideshare. *Slideshare.* [En línea] 2011. [Citado el: 16 de Enero de 2011.] http://www.slideshare.net/oswchavez/clase-1-sistema-de-gestion-de-base-de-datos.

Company Headquarters. 2010. Visual Paradigm. *Visual Paradigm.* [En línea] 2010. [Citado el: 16 de Enero de 2011.] http://www.visual-paradigm.com/shop/vpuml.jsp.

Cortizo Pérez, José Carlos, Expósito Gil, Diego y Ruiz Leyva, Miguel. *eXtreme Programming.*

Cueva Lovelle, Juan Manuel. 2009. *Calidad del Software.* España : s.n., 2009.

EJIE S.A Sociedad Informática del Gobierno Vasco. 2005. *JUnit: Manual de Usuario.* 2005.

Estrada Rodríguez, Liudmila de Las Mercedes. 2010. *Propuesta de una arquitectura en el dominio de media para los productos del departamento de señales digitales.* La Habana : s.n., 2010. Tesis.

Grupo Soluciones Innova. 2010. Rational Robot. [En línea] Grupo Soluciones Innova, 2010. [Citado el: 17 de 01 de 2011.] http://www.rational.com.ar/herramientas/robot.html.

Hewlett-Packard Development Company, L.P. 2011. HP Performance Center. [En línea] Hewlett-Packard Development Company, L.P., 2011. [Citado el: 15 de 01 de 2011.]

https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17_4000_10__.

Huanca, Daniel. 2009. Pruebas de Caja Negra. [En línea] 2009. [Citado el: 17 de 01 de 2011.] http://herrorsoft.zxq.net/pruebacajanegra.html.

Iberoamerican Journal of Project Management. 2011. Red Iberoamericana de Ingeniería de Proyectos. *Red Iberoamericana de Ingeniería de Proyectos.* [En línea] Iberoamerican Journal of Project Management, 2011. [Citado el: 4 de Abril de 2011.] http://usbvirtual.usbcali.edu.co/ijpm/index.php?option=com_content&view=article&id=31:sxp-metodologia-agil-para-el-desarrollo-de-software&catid=2:volumen2&Itemid=2. 2027-7040.

Juristo, Natalia, Moreno, Ana M. y Vegas, Sira. 2006. *Técnicas de evaluación de* software*.* 2006.

Larman, Craig. 1999. *UML y Patrones. Introducci6n a1 an6lisis y diseiio orientado a objetos.* México : Prentice Hall, 1999. 970-17-0261-1.

Lozano, Víktor. 2006. *Manual de CodeIgniter en Español.* Medellín : Conocimiento Virtual Academia ltda., 2006.

—. 2010. Visual Studio Test Professional 2010. [En línea] 2010. [Citado el: 15 de 01 de 2011.] http://www.microsoft.com/spain/visualstudio/products/2010-editions/test-professional.

Minguet Melián, Jesús M. y Hernández Ballesteros, Juan Francisco. 2003. *La calidad del* software *y su medida.* España : Centro de estudios Ramón Areces, s.a., 2003.

Open Demand Systems. 2000. OpenDemand System. *OpenDemand System.* [En línea] 2000. [Citado el: 18 de Enero de 2011.] http://www.opendemand.com/openload/faq.shtml.

Oracle Corporation. 2010. MySQL. *MySQL.* [En línea] 2010. [Citado el: 16 de Enero de 2011.] http://dev.mysql.com/doc/refman/5.0/en/what-is-mysql.html.

Potencier, Fabien y Zaninotto, François. 2008. *Symfony la guía definitiva.* 2008.

Pressman, Roger S. 2002. *Ingeniería de Software Un enfoque práctico.* s.l. : McGraw-Hill, 2002.

Project Management Institute. 2004. *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK®).* Estados Unidos : Project Management Institute, Inc., 2004. 1-930699-73-5.

Quest Software. 2005. [En línea] 10 de 10 de 2005. [Citado el: 23 de 11 de 2010.] http://www.alta-tecnologia.com/brochures/quest/Quest_Brochure_Corporativo.pdf.

Rojas, Johanna y Barrios, Emilio. 2007. Pruebas de Regresión. [En línea] ARQUISOFT, 2007. [Citado el: 13 de 01 de 2011.] http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node57.html.

Rubiano, y otros. 2009. SlideShare. *Automatizacion De Pruebas De Software.* [En línea] Universidad Alejandro de Humbolt, 2009. [Citado el: 13 de 01 de 2011.] http://www.slideshare.net/Rubiano/automatizacion-de-pruebas-de-software-1245969.

Scalone, Fernanda. 2006. Software Quality Management. *Software Quality Management.* [En línea] 1 de Noviembre de 2006. [Citado el: 14 de Diciembre de 2010.] http://softqm.blogspot.com/2006/11/gestin-de-la-calidad-del-software.html.

Softpedia.com. 2011. Softpedia. [En línea] SoftNews NET SRL, 03 de 05 de 2011. [Citado el: 18 de 05 de 2011.] http://www.softpedia.com/es/programa-DTM-Data-Generator-Professional-167243.html.

Sommerville, Ian. 2005. *Ingeniería del* software*.* España : Pearson Educación, s.a., 2005. ISBN:84-7829-074-5.

SPL Sistemas de Información. 2008. SPL Sistemas de Información. *SPL Sistemas de Información.* [En línea] 2008. [Citado el: 14 de Enero de 2011.] http://www.spl-ssi.com/?sec=articulos&subsec=descripcion&v=aplicaciones_web.

Testhouse. 2010. Automatizacion. [En línea] Testhouse Consultores S.A., 2010. [Citado el: 13 de 01 de 2011.] http://www.es.testhouse.net/automatizacion/.

—. 2010. HP. [En línea] Testhouse Consultores SA, 2010. [Citado el: 15 de 01 de 2011.] http://www.es.testhouse.net/hp/.

—. 2010. Software Libre. [En línea] Testhouse Consultores SA, 2010. [Citado el: 16 de 01 de 2011.] http://www.es.testhouse.net/software-libre/.

The PHP Group. 2001. PHP. *PHP.* [En línea] 2001. [Citado el: 15 de Enero de 2011.] http://www.php.net/.

Tortorella, Pablo. 2006. Introducción a PerformaSure . [En línea] 25 de 10 de 2006. [Citado el: 19 de 01 de 2011.] http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=58.

Udaondo Durán, Miguel. 2008. *Gestión de Calidad.* España : Diaz de Santos, s.a., 2008. ISBN:84-7978-013-4.

Universidad Carlos III. 2007. Universidad Carlos III de Madrid. *Universidad Carlos III de Madrid.* [En línea] 2007. [Citado el: 14 de Enero de 2011.] http://www.it.uc3m.es/mcfp/docencia/si/material/1_cli-ser_mcfp.pdf.

Veigas Chkout, Zenia y Ramos Arias, Osmany. 2010. *Portal para la gestión de la información del Centro de Identificación y Seguridad Digital.* La Habana : s.n., 2010.

Wells, Don. 2009. Extreme Programming: A gentle introduction . *Extreme Programming: A gentle introduction .* [En línea] 28 de Septiembre de 2009. [Citado el: 13 de Enero de 2011.] http://www.extremeprogramming.org/.

The Institute of Electrical and Electronics Engineers, Inc.1990. *IEEE Standard Glossary of*. New York, Estados Unidos:s.n.,1990.0-7381-0391-8

BIBLIOGRAFÍA

Juristo, Natalia, Moreno, Ana M. y Vegas, Sira. 2006. *Técnicas de evaluación de* software*.* 2006.

Lozano, Víktor. 2006. *Manual de CodeIgniter en Español.* Medellín : Conocimiento Virtual Academia ltda., 2006.

Microsoft. 2010. Cómo: Crear una prueba de rendimiento web codificada. [En línea] MSND, 2010. [Citado el: 23 de 03 de 2010.] http://msdn.microsoft.com/es-es/library/ms182550.aspx.

—. 2010. Cómo: Crear y ejecutar una prueba unitaria. [En línea] MSDN, 2010. [Citado el: 02 de 03 de 2011.] http://msdn.microsoft.com/es-es/library/dd286656.aspx.

—. 2010. Cómo: Crear y grabar una nueva prueba de rendimiento web. [En línea] MSND, 2010. [Citado el: 22 de 03 de 2010.] http://msdn.microsoft.com/es-es/library/ms182539.aspx.

—. 2010. Cómo: Guardar y abrir resultados de pruebas en Visual Studio. [En línea] MSND, 2010. [Citado el: 02 de 03 de 2011.] http://msdn.microsoft.com/es-es/library/ms404662.aspx.

Microsoft Corporation. 2010. Visual Studio 2010 Ultimate. [En línea] Microsoft Corporation, 2010. [Citado el: 02 de 03 de 2011.] http://www.microsoft.com/spain/visualstudio/products/2010-editions/ultimate/system-requirements.

Microsoft. 2011. Crear y editar pruebas de carga. [En línea] MSDN, 2011. [Citado el: 07 de 03 de 2011.] http://msdn.microsoft.com/es-es/library/dd728098.aspx.

—. 2010. Modificar los modelos de combinación de pruebas para especificar la probabilidad de que un usuario virtual ejecute una prueba. [En línea] MSDN, 2010. [Citado el: 01 de 04 de 2011.] http://msdn.microsoft.com/es-es/library/dd997826.aspx.

—. 2010. Probar la interfaz de usuario con pruebas de IU automatizadas. [En línea] MSDN, 2010. [Citado el: 11 de 03 de 2011.] http://msdn.microsoft.com/es-es/library/dd286726.aspx.

Nmap. 2011. Nmap. *Nmap.* [En línea] Nmap.org, 2011. [Citado el: 6 de Abril de 2011.] http://nmap.org.

Pereda Viñolo, Katerina y Fernández Santana, Vismar. 2010. *Plataforma para el desarrollo de servicios en línea utilizando tarjetas inteligentes.* UCI. La Habana : s.n., 2010. Tesis.

Proenza, Y. 2005. *Introducción al modelo conceptual.* Ciudad de La Habana, Universidad de las Ciencias Informáticas : s.n., 2005.

Scalone, Fernanda. 2006. Software Quality Management. *Software Quality Management.* [En línea] 1 de Noviembre de 2006. [Citado el: 14 de Diciembre de 2010.] http://softqm.blogspot.com/2006/11/gestin-de-la-calidad-del-software.html.

—. 2010. Software Libre. [En línea] Testhouse Consultores SA, 2010. [Citado el: 16 de 01 de 2011.] http://www.es.testhouse.net/software-libre/.

Tortorella, Pablo. 2006. Introducción a PerformaSure . [En línea] 25 de 10 de 2006. [Citado el: 19 de 01 de 2011.] http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=58.

GLOSARIO DE TÉRMINOS

AJAX: acrónimo de *Asynchronous JavaScript And XML* (JavaScript y XML asíncronos). Es una técnica de desarrollo web para crear aplicaciones interactivas.

ASP.NET: es un marco de trabajo para aplicaciones web desarrollado y comercializado por Microsoft.

Axis: marco de trabajo del lenguaje de programación Java para el trabajo con los servicios web.

***Bottom-up*:** estrategia de procesamiento de información, características de las ciencias de la información, especialmente en lo relativo al software. En el modelo *bottom-up* las partes individuales se diseñan con detalle y luego se enlazan para formar componentes más grandes, que a su vez se enlazan hasta que se forma el sistema completo.

Ciclo de calidad: es el conjunto de actividades de aseguramiento de la calidad del software, que se inicia con el proyecto y se va repitiendo a lo largo del desarrollo de una aplicación, extendiéndose incluso durante las actualizaciones del producto.

Complejidad ciclomática: es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa.

CSS: las hojas de estilo en cascada (en inglés: *Cascading Style Sheets*) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML.

Cuello de botella: también llamado “embudo” es el hecho de que un sistema no pueda atender numerosas solicitudes afectándose su funcionamiento.

Encapsulamiento: es una característica de la programación orientada a objetos en la que cada tipo objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de manera inesperada.

Herencia: en la programación orientada a objetos, es el mecanismo fundamental para implementar la reutilización y extensibilidad del software. A través de ella los diseñadores pueden construir nuevas clases

partiendo de una jerarquía de clases ya existente evitando con ello el rediseño, la modificación y verificación de la parte ya implementada.

IDE: acrónimo inglés de *Integrated Development Environment* (Entorno de Desarrollo Integrado). Es un programa informático compuesto por un conjunto de herramientas de programación y otras.

IMAP: acrónimo inglés de *Internet Message Access Protocol*. Es un protocolo de acceso a mensajes de Internet. Sirve para lograr la comunicación entre los servidores de correo electrónico y sus clientes.

Incidencia: en Informática, es un evento inesperado que afecta negativamente a la calidad de un servicio o el funcionamiento de un sistema o aplicación.

J2EE: Java Platform, Enterprise Edition o Java EE, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Java con arquitectura de n capas distribuidas y que se apoya en componentes de software ejecutándose sobre un servidor.

J2SE: Java Platform, Standard Edition o Java SE, es una colección de interfaces del lenguaje de programación Java útiles para muchos programas de la plataforma Java.

***Java Database Connectivity* (JDBC):** es una API (*Application Programming Interface*) 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 lenguaje para bases de datos SQL (*Structured Query Language*) del modelo que se utilice.

JavaCard Development Kit: marco de trabajo para tecnologías de tarjetas inteligentes.

JMS: acrónimo inglés de *Java Message Service* (Servicio de Mensajes Java), es una API que permite a los componentes de aplicaciones basados en la plataforma Java crear, enviar, recibir y leer mensajes.

Lenguajes *script*: son códigos que se insertan dentro de un documento HTML para añadir funcionalidades a una página web, mejorar la interacción con el usuario y modificar el diseño.

Microsoft Ajax Library: es una colección de clases en JavaScript estandarizadas incluidas con ASP.NET AJAX (conjunto de extensiones para ASP.NET). Es admitida por la mayoría de los navegadores más populares y puede ser usada para construir aplicaciones web centradas en el cliente.

Mono .Net Framework: marco de trabajo para el IDE MonoDevelop.

Polimorfismo: en programación orientada a objetos, el polimorfismo se refiere a la capacidad para que varias clases derivadas de una antecesora utilicen un mismo método de forma diferente.

RAM: acrónimo inglés de *Random-Access Memory* (Memoria de Acceso Aleatorio). Es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados.

RSS: acrónimo inglés de *Really Simple Syndication*, un formato XML para compartir contenido en la web. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos.

***Scripts* de prueba:** es un código escrito en un lenguaje determinado, que puede ser usado por diferentes herramientas para probar un componente o un programa informático.

***Simple Object Access Protocol* (SOAP):** es un protocolo que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML (*eXtensible Markup Language*).

Spring: marco de trabajo de código abierto de desarrollo de aplicaciones para la plataforma Java.

***Top-down*:** estrategia de procesamiento de información, especialmente en lo relativo al software. En el modelo *top-down* se formula un resumen del sistema, sin especificar detalles. Cada parte del sistema se refina diseñando con mayor detalle. Cada parte nueva es entonces redefinida, cada vez con mayor detalle, hasta que la especificación completa es lo suficientemente detallada para validar el modelo.

***Transmission Control Protocol* (TCP):** protocolo usado en Internet para la comunicación entre ordenadores.

W3C: acrónimo de *World Wide Web Consortium*. Es una organización internacional encargada de estandarizar tecnologías que se usan en Internet.

***Windows Presentation Foundation* (WPF):** es una tecnología de Microsoft que permite el desarrollo de interfaces de interacción en Windows tomando las mejores características de las aplicaciones Windows y de las aplicaciones web.

ANEXOS

Anexo 1. Criterios para diseñar casos de prueba de caja blanca y caja negra

Para pruebas de caja blanca:

Cobertura de sentencias: se escriben casos de prueba suficientes para que cada sentencia en el programa se ejecute, al menos, una vez.

Cobertura de decisión: se escriben casos de prueba suficientes para que cada decisión en el programa se ejecute una vez con resultado verdadero y otra con resultado falso.

Cobertura de condiciones: se escriben casos de prueba suficientes para que cada condición en una decisión tenga una vez resultado verdadero y otra resultado falso.

Cobertura decisión/condición: se escriben casos de prueba suficientes para que cada condición en una decisión tome todas las posibles salidas, al menos una vez, y cada decisión tome todas las posibles salidas, al menos una vez.

Cobertura de condición múltiple: se escriben casos de prueba suficientes para que todas las combinaciones posibles de resultados de cada condición se invoquen, al menos una vez.

Cobertura de caminos: se escriben casos de prueba suficientes para que se ejecuten todos los caminos de un programa, entendiendo camino como una secuencia de sentencias encadenadas desde la entrada del programa hasta su salida. (Juristo, y otros, 2006)

Para pruebas de caja negra:

Particiones de equivalencia: este método busca crear clases de datos de cada campo de entrada de un programa. Su objetivo es descubrir tipos de errores que permitan reducir la cantidad de casos de pruebas a ejecutar. Esto es importante, porque con un caso de prueba que se corresponde con una clase de dato, sería suficiente para demostrar que con otro valor similar, también se obtendría el mismo resultado, tanto errores como respuestas válidas.

Análisis de valores límites: esta técnica se centra en diseñar los casos de pruebas con los valores límites que puedan ser admitidos por los campos, debido a que proporcionan una idea más precisa de la efectividad de la evaluación. El análisis también se aplica a los datos de salida.

Métodos basados en grafos: este método plantea que se debe crear un grafo donde los nodos son objetos, tales como datos, módulos o grupo de sentencias, en dependencia de cómo se considere diseñar los casos de prueba. Las conexiones entre los nodos serían las relaciones entre esos objetos.

Luego de haber creado el grafo, los casos de pruebas que se diseñen quedarán de forma tal que se visiten todos los nodos y sus relaciones. El ingeniero americano Boris Bézier definió un conjunto de modelos a tener en cuenta a la hora de aplicar este método:

- **Modelado del flujo de transacción:** los nodos representan los pasos de alguna transacción y los enlaces representan las conexiones lógicas entre los pasos.
- **Modelado de estado finito**: los nodos representan diferentes estados del software observables por el usuario (por ejemplo, cada una de las pantallas que aparecen cuando un telefonista coge una petición por teléfono), y los enlaces representan las transiciones que ocurren para moverse de estado a estado (por ejemplo, petición-información).
- **Modelado de flujo de datos**: los nodos objetos de datos y los enlaces son las transformaciones que ocurren para convertir un objeto de datos en otro.
- **Modelado de planificación**: los nodos son objetos de programa y los enlaces son las conexiones secuenciales entre esos objetos. Los pesos de enlace se usan para especificar los tiempos de ejecución requeridos al ejecutarse el programa. (Huanca, 2009)

Pruebas de comparación: también son llamadas pruebas de mano a mano; se le realizan a distintas versiones de una misma aplicación; si las versiones se desarrollan a la vez, lo lógico es que después de terminadas arrojen los mismos resultados. También pueden servir para determinar semejanzas y diferencias entre una versión superior y otra inferior.

Prueba de la tabla ortogonal: se realiza cuando los campos de entrada son pocos y los posibles valores también lo son. El objetivo es probar exhaustivamente con cada posible valor del dominio, debido a que aplicar otros métodos podría ser más costoso.

Prueba de conjetura de errores: para llevar a cabo esta prueba se realiza una lista de errores o situaciones comunes que suelen provocar fallos o resultados inesperados en las aplicaciones. No existe un procedimiento estándar a seguir, solo que la lista debe ser confeccionada por una persona con experiencia realizando evaluaciones de software.

Análisis causa-efecto: este modelo pretende que se cree una gráfica donde se planteen la causa y el efecto de determinados problemas con el objetivo de tener una visión clara de los errores presentes. Algunos autores incluyen este método como un modelo basado en grafos. Es importante aclarar que está muy relacionado con el anterior.

Anexo 2. Empresas y organizaciones que desarrollan herramientas para pruebas de software

Microsoft Corporation: corporación estadounidense dedicada al desarrollo de software y *hardware* para computadoras, videojuegos, entre otros productos. Desde hace varios años, ha implementado como parte de la *suite*[6] de Visual Studio, herramientas que permiten la gestión de la calidad de las aplicaciones desarrolladas con el Entorno de Desarrollo Integrado (IDE).

Hewlett-Packard Company (HP): es una compañía estadounidense dedicada a producir computadoras, programas informáticos, periféricos y a ofrecer servicios relacionados con tecnologías de la información. Es la empresa número uno del mercado de las pruebas de software. Presume que ha desarrollado numerosas aplicaciones que cubren todas las necesidades de los clientes.

International Business Machines (**IBM**): es una firma multinacional estadounidense encargada de producir *hardware*, software y ofrecer distintos servicios. Actualmente es propietaria de Rational Software, y por ende, del programa Rational Robot, famoso por las pruebas que lleva a cabo.

[6] *Suite*: en Informática es un conjunto de programas.

Quest Software, Inc: es una compañía multinacional estadounidense, encargada del desarrollo de numerosas herramientas para la gestión de sistemas. Algunas de esas aplicaciones permiten automatizar pruebas de software usando diversas tecnologías como Java, Microsoft .NET, MySQL, Oracle, entre otras.

También existen varias organizaciones, compañías y desarrolladores de herramientas libres para pruebas de software. Algunos de los más importantes son los siguientes:

Proyecto Jakarta: es una organización que crea y mantiene aplicaciones de código abierto para la plataforma Java. Todos sus productos son liberados bajo la Licencia de Apache. Dentro de las tantas herramientas que implementa este proyecto hay varias para pruebas, entre ellas están: Cactus, un marco de trabajo para pruebas unitarias, y JMeter, un programa para pruebas funcionales y de rendimiento.

Proyecto MantisBT: es un grupo de desarrolladores distribuidos en diferentes países y encargados de mantener la herramienta MantisBT. Están asociados con varias compañías importantes en diversos negocios, las cuales han utilizado la aplicación.

Comunidad TestLink: es una comunidad abierta de probadores de varias partes del mundo, sin ánimos de lucro y no asociados a ninguna compañía. Ellos han desarrollado la herramienta TestLink.

Anexo 3. Herramientas que permiten la automatización de las pruebas de software

Herramientas comerciales

Microsoft Visual Studio 2010: la *suite* de Visual Studio tiene un conjunto de herramientas llamadas ALM (*Application Lifecycle Management,* en español: Aplicación de Gestión del Ciclo de Vida), dentro de las cuales, hay varias para equipos especializados en el control de la calidad, con el objetivo de hacer más simple el proceso de la planificación y ejecución de pruebas. Visual Studio Test Professional 2010 proporciona una interfaz para pruebas de varios tipos. Con este programa se pueden crear planes de pruebas, conjuntos de pruebas y casos de pruebas con capacidades de anidamiento. (Microsoft, 2010)

HP Quality Center: este software permite disponer de una aplicación web para la realización de las pruebas en la cual se puede gestionar cualquier información relacionada con estos fines. Admite la gestión

de requisitos, planificación y programación de pruebas, análisis de los resultados obtenidos, así como la administración de no conformidades. Está disponible en tres versiones que determinan el nivel de especialización de las actividades.

HP QuickTest Professional: esta herramienta soporta la automatización de pruebas de regresión y pruebas funcionales; utiliza el concepto de pruebas basadas en palabras claves para simplificar la creación y el mantenimiento de las pruebas. Permite a los probadores crear casos de prueba mediante la detección de flujos directamente desde las pantallas de la aplicación, mediante una tecnología de captura especializada. (Testhouse, 2010)

HP Performance Center: ofrece una plataforma de pruebas de rendimiento de las aplicaciones de tipo corporativo que se puede gestionar mediante la web, proporcionando acceso durante 24 horas, los 7 días de la semana, a una infraestructura de pruebas compartida. También se puede reducir los costos de software y *hardware* mediante la predicción con precisión de la capacidad del sistema, permite descubrir la causa de los problemas de rendimiento de las aplicaciones de forma rápida y precisa, así como aplicar buenas prácticas acerca de cómo lograr un buen rendimiento. (Hewlett-Packard Development Company, L.P., 2011)

HP LoadRunner: este software ayuda a prevenir problemas de rendimiento y detectar cuellos de botella antes de actualizar o implantar una aplicación.

HP WebInspect: realiza pruebas de seguridad en aplicaciones web construidas o no sobre las nuevas tecnologías Web 2.0[7]. Ofrece una rápida exploración y una amplia evaluación de la seguridad de las aplicaciones web. Identifica vulnerabilidades de seguridad que no son detectables por otros programas. (Testhouse, 2010)

Benchmark Factory: es una herramienta para simular el acceso de miles de usuarios a servidores de bases de datos, archivos, Internet y correo. Esto permite darle solución a los problemas de rendimiento de las aplicaciones, incluso antes de ser entregadas a los clientes. Es propiedad de Quest Software.

[7]Web 2.0: es un término que surgió en 2004 para referirse al conjunto de nuevos avances de la Internet, entre los cuales están: la colaboración entre los usuarios, el alojamiento de videos y las redes sociales.

PerformaSure: fue desarrollada por Quest Software y permite: detectar servidores o procesos que degradan el funcionamiento de la aplicación distribuida, detectar problemas en las consultas o las conexiones con bases de datos (mediante un analizador SQL), encontrar cuellos de botella y generar estadísticas sobre el trabajo de la aplicación. (Tortorella, 2006)

Rational Robot: desarrollado dentro de la familia de programas de Rational Software, actualmente propiedad de International Business Machines (IBM), es una herramienta encargada de automatizar las pruebas de regresión de aplicaciones .NET, Java, web y otras aplicaciones con interfaz gráfica. Este software proporciona casos de prueba para objetos comunes, como menús, listas y casos de prueba especializados para los objetos específicos del entorno de desarrollo. Además, se integra con las herramientas de Rational Team Unifying Platform para realizar el seguimiento de defectos, gestión de cambios y rastreo de requerimientos. (Grupo Soluciones Innova, 2010).

Spotlight: desarrollada por Quest Software, realiza diagnósticos de alta precisión sobre problemas de rendimiento en tiempo real y muestra gráficamente toda la actividad de la aplicación en una interfaz de usuario intuitiva, lo que facilita la rápida resolución de problemas sin necesidad de crear *scripts* a medida o utilizar complejas herramientas de ayuda. Spotlight está disponible para Oracle, SQL Server, servidores web y otras tecnologías. (Quest Software, 2005)

Data Factory: está desarrollada por Quest Software. Genera bases de datos de desarrollo, pruebas y control de calidad con millones de filas de datos de prueba sintácticamente correctos y con pleno significado, lo que permite disponer de una visión realista del rendimiento de las aplicaciones antes de ponerlas en explotación. (Quest Software, 2005)

JProbe: permite detectar los “puntos calientes” de los componentes de una aplicación Java, tales como el uso de la memoria, el uso de la Unidad Central de Procesamiento (CPU), los hilos de ejecución, y a partir de ellos, bajar al nivel del código fuente que los provoca ofreciendo una serie de consejos o buenas prácticas de codificación para la resolución del problema. Optimiza código escrito en Java y ayuda a los usuarios a diagnosticar y resolver con rapidez los problemas de rendimiento en entornos de aplicación J2EE y J2SE mediante análisis a nivel de código fuente. (Quest Software, 2005)

JTest: es un software multiplataforma que realiza pruebas funcionales de manera automática. Soporta diversos marcos de trabajo de desarrollo Java y utiliza JUnit para hacer los casos de prueba. Es una

solución integrada para la automatización de una amplia gama de prácticas probadas para mejorar la calidad del software. Se centra en las prácticas de validación de código Java. Es utilizado por compañías como Cisco Systems, TransCore y AIG United Guaranty.

Herramientas libres

Open Load Tester: es un programa desarrollado por IBM. Puede ser utilizado para probar casi cualquier aplicación basada en web, así como casi cualquier cliente web, incluyendo la tecnología AJAX, JavaScript, *applets* Java, Flash y mucho más. Cuenta con capacidades para pruebas de regresión y pruebas funcionales, las cuales le permiten comprobar de forma sencilla los resultados esperados. Es una aplicación basada en navegador; se puede utilizar desde cualquier plataforma que soporte Internet Explorer o Firefox y es compatible con las versiones de Windows posteriores a 1998. (Open Demand Systems, 2000)

JUnit: es un conjunto de clases creadas por Erich Gamma[8] y Kent Beck[9], 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. JUnit es un paquete Java utilizado para automatizar los procesos de prueba, al código que indique el usuario. (EJIE S.A Sociedad Informática del Gobierno Vasco, 2005). Otras versiones del *framework* han sido creadas para soportar nuevos lenguajes de programación como PHP, C#, C++, Delphi y muchos otros.

TestLink: es una herramienta para la gestión de pruebas. Está desarrollada y actualizada por una comunidad abierta de probadores y tiene licencia GPL[10]. Permite gestionar casos de prueba y organizarlos en planes, los cuales permiten realizar un seguimiento de la ejecución de las pruebas, tener en cuenta la

[8]Erich Gamma: Informático suizo, es actualmente el director del centro tecnológico OTI en Zúrich y lidera el desarrollo de la plataforma Eclipse. Formó parte del grupo de los cuatro (Gang of Four o, simplemente, GoF) que popularizaron los patrones de diseño. También fue el creador de JUnit, junto a Kent Beck.
[9]Kent Beck: Uno de los creadores de la metodología ágil para el desarrollo de software conocida como programación extrema. También creó, junto con Erich Gamma el marco de trabajo de pruebas unitarias para Java, JUnit.
[10]Licencia GPL: es una licencia creada por la Free Software Foundation orientada principalmente a proteger la libre distribución, modificación y uso de software.

trazabilidad de los requisitos y priorizar las tareas. TestLink es una aplicación web, escrita en PHP, construida para poder trabajar con las bases de datos Microsoft SQL Server, MySQL y PostgreSQL.

Mantis: es una herramienta para la gestión de defectos y tiene licencia GPL. Está diseñada para la creación de proyectos, asignación de incidencias y para dar seguimiento a actividades afines. Fue desarrollada en PHP para usarla con las bases de datos Microsoft SQL Server, MySQL o PostgreSQL. Su nombre oficial es MantisBT (Mantis Bug Tracker).

Es posible integrar **Mantis** y **TestLink** para obtener una trazabilidad requisito-caso de prueba-defecto. Esto permite eliminar la duplicidad de errores y casos de pruebas que no responden a ninguna funcionalidad, también denominados casos de prueba "huérfanos".

Selenium IDE: es un complemento para el navegador Mozilla Firefox, disponible para las versiones iguales o posteriores a la 1.5. Forma parte del conjunto de herramientas SeleniumHQ y fue diseñado para realizar pruebas funcionales, es decir, pruebas de caja negra, en aplicaciones web. Permite repetir la misma prueba, cambiando valores en los formularios, reproduciendo las acciones de manera visual, rápida o lentamente para una mejor apreciación de los pasos. Además, está apto para guardar una prueba o un conjunto de ellas, con el objetivo de repetirlas cuando el sistema haya sido modificado; es ideal para pruebas de regresión.

JMeter: fue desarrollada por Jakarta Project (Proyecto Jakarta) para realizar pruebas funcionales y de rendimiento. Está escrita en Java y es representada por la Licencia Apache[11]. Puede ser usada para llevar a cabo pruebas unitarias en conexiones de bases de datos con *Java Database Connectivity* (JDBC), así como también aplicaciones web, *File Transfer Protocol* (FTP), servicios web, SOAP (*Simple Object Access Protocol*), JMS (Servicio de Mensajes Java) y LDAP (Protocolo Ligero de Acceso a Directorios). Sirve además para generar carga en los sistemas.

Backtrack 4: es una distribución de Linux basada en Ubuntu que incluye numerosas aplicaciones para realizar pruebas de seguridad y análisis informático forense. Gracias a las aplicaciones incluidas, Backtrack se ha convertido en una distribución imprescindible para los administradores de sistemas y

[11]Licencia Apache (Apache License o Apache Software License) es una licencia de software libre creada por la Apache *Software* Foundation (ASF). Está vigente para las versiones 1.0, 1.1 y 2.0 no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

profesionales de la auditoría informática. La distribución incluye utilidades para la auditoría de redes inalámbricas, *scanner* de puertos y vulnerabilidades, *sniffers*[12], archivos de *exploits*[13], etc. (Testhouse, 2010)

Sonar: es una herramienta para realizar pruebas unitarias, determinar la complejidad del código, identificar errores potenciales y duplicaciones, aportar reglas de codificación, realizar comentarios, etc. A través de otras herramientas, Sonar muestra en una interfaz los resultados del análisis realizado en forma de métricas. Las aplicaciones con las que trabaja son varias, entre ellas están: Checkstyle, PMD, Cobertura, etc.

Cobertura: está escrita en Java; permite saber la cantidad de código probado, lo cual la hace ideal para usarla como "métrica" en los proyectos. También calcula la complejidad ciclomática de un método, la cantidad de clases en un fichero, el número de posibles caminos a probar, entre otros datos.

Checkstyle: es una herramienta para ayudar a los programadores a escribir código Java que se adhiera a los estándares. Automatiza el proceso de chequeo de código para facilitarles a los programadores esta importante tarea. Es ideal para los proyectos que quieren esforzarse en codificar de forma estándar. Es altamente configurable y puede dar soporte a varios estándares de código. Para todo ello, se han escrito varios complementos para entornos de desarrollo, entre los cuales se encuentran Checklipse y ChaeckstyleBeans, para los IDE Eclipse y NetBeans respectivamente.

PMD: es un complemento de código abierto que se integra a varios IDE (Eclipse, NetBeans, JBuilder, etc.) y busca problemas potenciales, entre los que se encuentran: posibles errores (entradas vacías, omisión de tratamiento de errores, etc.), código no usado, sentencias muy complejas o innecesarias y código duplicado.

Anexo 4. Requerimientos no funcionales

Requerimientos de software

Estaciones de trabajo:

[12] *Sniffer*: es un programa para monitorear redes, usado generalmente con fines maliciosos.
[13] *Exploit* (explotar): es un componente de software o una secuencia de comandos para aprovechar un error, fallo o vulnerabilidad, a fin de causar un comportamiento no deseado en los programas informáticos o *hardware*.

Para un correcto funcionamiento de la aplicación se requiere un:

- Sistema operativo: multiplataforma
- Navegador web: Internet Explorer 8 o superior, Mozilla Firefox, Opera, Google Chrome (o cualquier otro que cumpla con los estándares actuales de codificación)

Servidor de aplicaciones y base de datos:

- Sistema operativo: multiplataforma
- Servidor web Apache 5.3.0
- Gestor de Base de Datos MySQL 5.1.37

Requerimientos de *hardware*

Estaciones de trabajo

- Periféricos: mouse y teclado
- Tarjeta de red
- 256 MB de RAM
- Procesador Intel Pentium III o superior
- 50 MB de espacio en disco

Servidor de aplicaciones y base de datos:

- Tarjeta de red
- 1 GB de RAM (*Random-Access Memory*)
- Procesador Intel Pentium IV o superior
- 500 MB de espacio en disco

Requisitos de apariencia o interfaz externa

La aplicación debe poseer:

- Una interfaz sencilla y amigable. Su formato deberá ser mantenido en todas las páginas que la componen.

- Colores claros con contraste entre el texto y el fondo para no dificultar la lectura.
- Un menú principal que le brinde al usuario rapidez y facilidad para obtener la información que desea.
- Optimizado para una resolución de 1024x768 píxeles.

Requisitos de usabilidad

La aplicación deberá:

- Estar implementada para que usuarios con poca experiencia puedan utilizarla fácilmente.
- Poseer todos los textos y mensajes en idioma español.

Requisitos de rendimiento

- La aplicación deberá ser capaz de atender pedidos y enviar respuestas a 50 usuarios al mismo tiempo.

Requisitos de seguridad

Confiabilidad:

- La información estará protegida contra accesos no autorizados utilizando mecanismos de validación que lo puedan garantizar: usuario, contraseña y nivel de acceso, de manera que cada uno pueda tener disponible solamente las opciones relacionadas con su actividad y tenga datos de acceso propios, garantizando así la confidencialidad.

Confidencialidad:

- Para almacenar la información de los usuarios en la base de datos se utilizará el mecanismo de encriptación del marco de trabajo seleccionado.

Integridad:

- La aplicación brindará garantía de un tratamiento adecuado de las entradas del usuario.
- El sistema muestra alertas para que el usuario confirme si desea proseguir ante acciones irreversibles (por ejemplo: las eliminaciones).

Disponibilidad:

- El sistema deberá tener un 100% de disponibilidad, por lo que podrá ser usado las 24 horas del día por todos sus usuarios.

Requisitos de soporte

La aplicación poseerá:

- Una ayuda para que el usuario entienda dónde buscar lo que necesita.
- Documentación adicional.
- Una guía de usuario para cada herramienta de prueba propuesta.

Anexo 5. Comparación de la metodología, tecnologías y herramientas utilizadas con otras similares

A continuación el análisis realizado para elegir la metodología de desarrollo de software que será utilizada:

Las metodologías ágiles están especialmente preparadas para cambios durante el proyecto, sus procesos son menos controlados, poseen pocos principios, el contrato es bastante flexible, son utilizadas por pequeños grupos (menos de diez integrantes) que radican en el mismo sitio, generan pocos artefactos y no poseen gran cantidad de roles.

Algunos ejemplos de metodologías ágiles son:

- ***Extreme Programming*** **(XP)**, uno de los ejemplos más exitosos de metodología ágil
- *Scrum*
- *Crystal*
- *Evolutionary Project Management* (Evo)
- *Feature Driven Development* (FDD)
- *Adaptive Software Developmen* (ASD)
- *Lean Development (*LD) y *Lean Software Development* (LSD)

Por su parte, las metodologías tradicionales manifiestan cierta resistencia al cambio, los procesos son mucho más controlados con numerosas políticas, existe un contrato prefijado, se caracterizan por ser utilizadas por grupos grandes y posiblemente distribuidos, generan gran cantidad de artefactos y poseen numerosos roles.

Ejemplo de metodología tradicional:

- ***Rational Unified Process* (RUP)**

Fueron destacadas las metodologías XP y RUP, producto a que figuran como las más usadas cada una en su respectivo grupo.

RUP es un proceso formal, provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Fue desarrollado por *Rational Software*, y está integrado con toda la *suite* Rational. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de modelado.

Las cuatro fases del ciclo de vida son: Inicio, Elaboración, Construcción y Transición.

Presenta las ventajas siguientes:

- Evaluación en cada fase que permite cambios de objetivos.
- Funciona bien en proyectos de innovación.
- Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- Seguimiento detallado en cada una de las fases.

Posee las siguientes desventajas:

- La evaluación de riesgos es compleja.
- Excesiva flexibilidad para algunos proyectos.
- Se pone al cliente en una situación que puede ser muy incómoda para él.

- El cliente deberá ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él.

***Extreme Programming*:**

La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Presenta las siguientes ventajas:

- Se consiguen productos usables con mayor rapidez.
- Se consigue tener un equipo de desarrollo más contento y motivado. Las razones son, por un lado, el que XP no permite excesos de trabajo (se debe trabajar 40 horas a la semana), y por otro, la comunicación entre los miembros del equipo consigue una mayor integración entre ellos.
- Permite una mejor adaptabilidad a los cambios, que se traduce en una reducción de costos.
- Apropiado para entornos volátiles, equipos de desarrollo pequeños (de 2 a 10 desarrolladores) y proyectos de alto riesgo.
- Permite tener retroalimentación continua de los usuarios a través de las entregas frecuentes.
- Gracias a la filosofía de programación en parejas (*pair programming*), se consigue que los desarrolladores apliquen las buenas prácticas que se les ofrecen con la XP.
- El proceso de integración es continuo, por lo que el esfuerzo final para la integración es nulo.
- Se consiguen productos más fiables y robustos contra los fallos gracias al diseño de las pruebas de forma previa a la codificación.

Posee dificultades al delimitar el alcance del proyecto con el cliente.

Una vez realizado este análisis, fue sencillo determinar que la metodología ágil XP, por sus particularidades ya definidas con anterioridad, era la más acorde con las características del sistema que se desea desarrollar.

- **Herramientas**

A continuación el análisis realizado para determinar las herramientas a utilizar en la aplicación:

Ejemplos de herramientas CASE:

- **Rational RoseEnterprise Edition**
- **Visual Paradigm**
- JBuilder
- PowerDesigner

Fueron destacadas las principales herramientas CASE usadas y estudiadas en la universidad; se realizó un análisis teniendo en cuenta las características presentes en cada una de ellas con el objetivo de seleccionar la idónea para el modelado del software a desarrollar.

Rational Rose Enterprise Edition: propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Rational Rose Enterprise Edition permite el desarrollo iterativo, el trabajo en grupo, generar código y realizar ingeniería inversa.

Visual Paradigm: soporta las últimas versiones de UML. Provee el modelado de procesos de negocio, además de un generador de mapeo de objetos-relacionales para los lenguajes Java, .NET y PHP. Es un estándar ampliamente utilizado que proporciona valiosa ayuda a los profesionales visualizando, comunicando y aplicando sus diseños.

Visual Paradigm posee una interfaz amigable, es la principal herramienta de modelado utilizada en la universidad y además, ayuda a los equipos de desarrollo a sobresalir de la acumulación de trabajo y desplegar el proceso de desarrollo de software, lo que permite maximizar y acelerar tanto las contribuciones individuales como las de equipo.

Se decide utilizar como herramienta CASE, Visual Paradigm, debido a las características presentes detalladas con anterioridad y por ser considerada superior al Rational Rose Enterprise Edition.

Igualmente, se hizo un análisis para determinar el entorno de desarrollo que se utilizará para la implementación del sistema. Existen varios IDE para PHP, el cual es el lenguaje de programación seleccionado para desarrollar la aplicación. Algunos de los más populares son:

Zend Studio: este IDE comercial multiplataforma está entre los tres más populares de su tipo. Su desarrollo se debe al trabajo de la compañía Zend Technologies Ltd, la cual lanza actualizaciones regularmente. Zend Studio tiene una amplia ayuda para su trabajo, fue escrito en Java y tiene una interfaz muy amigable. Soporta tanto PHP 4 como PHP 5 y no requiere de la instalación de PHP en la estación de trabajo. Es capaz de autocompletar código, resaltar errores, mostrar parámetros dentro de las funciones y métodos, resaltar el emparejamiento de paréntesis y corchetes, entre otras opciones. Tiene varias funciones de depuración, permite la escritura de otros lenguajes como HTML y JavaScript, soporta el control de versiones a través de herramientas para estos fines como lo es Subversion y no consume muchos recursos de *hardware*.

Komodo Edit: es un programa libre y multiplataforma de la compañía *ActiveState Software Inc*. Fue escrito en varios lenguajes de programación, entre los que se encuentran C++, Perl, Python y JavaScript, por solo citar a los más conocidos. Está distribuido bajo la Licencia Pública de Mozilla. Soporta otros lenguajes aparte de PHP, entre los que se encuentran Perl, Python, Ruby, CSS, HTML, XML, JavaScript y SQL. Tiene integrado un cliente FTP. Su interfaz y funcionalidad están limitadas comparadas con otros entornos.

PHP Eclipse: es un software de código abierto disponible para diversos sistemas operativos, desarrollado originalmente por IBM y luego por la Fundación Eclipse. Tiene una gran cantidad de usuarios que contribuyen a su desarrollo. Está escrito en Java y registrado con la Licencia Pública de Eclipse. Asociado a Eclipse, se ha desarrollado un proyecto llamado PDT (PHP Development Tools, en español: Herramientas de Desarrollo PHP). El software resultante tiene varias características propias de un IDE, pero con ciertas limitaciones en cuanto a funcionalidad, comparado con otras herramientas similares.

PHP Designer: sencillo, elegante, ligero y funcional, así es este IDE pensado especialmente para PHP, pero que también soporta otros lenguajes como XHTML, CSS, Java, Perl, JavaScript, C# y SQL. Tiene librerías de código habitual, cliente FTP, opciones de autocompletado y otras. Es un software propietario perteneciente a la compañía MP Software; fue escrito para sistemas operativos Windows.

PHPEdit: fue desarrollado por la compañía WaterProof SARL para plataformas Windows y es un software comercial. Soporta otros lenguajes como CSS y HTML. Una licencia libre está disponible para uso particular. Tiene un cliente FTP, manuales, se integra con controladores de versiones, chequea la sintaxis del código, entre otras opciones.

A pesar de la existencia de estos entornos, se eligió a **NetBeans** como el IDE para desarrollar la aplicación porque es libre, tiene todas las características de cualquier programa avanzado de este tipo y tiene un uso amplio en el CISED y en la Universidad, lo cual es demostrado por las múltiples opiniones favorables recibidas por parte de desarrolladores del Centro.

Anexo 6. Organización de los proyectos del CISED y su clasificación

Tabla 6.9. Organización de los proyectos del CISED y su clasificación (Fuente: Elaboración propia)

Proyecto	Departamento	Clasificación
Integración Bancos	Identificación	Desarrollo
Migración: • Integración del Sistema de Gestión del SAIME con los Equipos de Autochequeo Migratorio (sub-proyecto)	Identificación	Desarrollo
Sistema de Gestión de Entidades Externas	Identificación	Desarrollo
Pasaporte Diplomático y de Servicio	Identificación	Desarrollo
Dactilab	Biometría	Investigación + Desarrollo
Facelab	Biometría	Investigación + Desarrollo
Signlab	Biometría	Investigación + Desarrollo
Sistema de Identificación, Inmigración y Extranjería de la República de Cuba	Soluciones Integrales	Desarrollo
Plataforma para el Desarrollo de Servicios en Línea Utilizando Tarjetas Inteligentes	Tarjetas Inteligentes	Desarrollo
Solución para la Gestión de la Información en Tarjetas Inteligentes:	Tarjetas Inteligentes	Desarrollo

• Secure Data Manager Applet (sub-proyecto) • Secure Data Manager Middleware (sub-proyecto)		
Solución para la Gestión de Información en Credenciales Universitarias	Tarjetas Inteligentes	Desarrollo
Applet para la Gestión de Información Personal y Servicios	Tarjetas Inteligentes	Investigación + Desarrollo
Middleware para la Gestión de Información Personal y Servicios	Tarjetas Inteligentes	Desarrollo
Solución para la Lectura de Documentos de Viaje Electrónicos: • Applet LDS (sub-proyecto) • Middleware LDS (sub-proyecto)	Tarjetas Inteligentes	Desarrollo
Solución para la Gestión de Información Biométrica en Tarjetas Inteligentes: • Applet para la Verificación Biométrica en Tarjetas Inteligentes (sub-proyecto) • Middleware para la Verificación Biométrica en Tarjetas Inteligentes (sub-proyecto)	Tarjetas Inteligentes	Desarrollo
Sistema de Administración de Identidades	Tarjetas Inteligentes	Desarrollo
Sistema de Gestión de Servicios para la CIE	Seguridad Digital	Desarrollo
Sistema para la Verificación del SOD	Seguridad Digital	Desarrollo
Soluciones PKI	Seguridad Digital	Desarrollo

Anexo 7. Encuesta realizada al personal del CISED sobre las características de los proyectos

La encuesta realizada consistió en un documento con la siguiente orden: “Complete la siguiente tabla con los datos correspondientes a los proyectos en los cuales usted trabaja”.

Esta fue la tabla:

Tabla 7.10. Formato de la encuesta realizada al personal del CISED (Fuente: Elaboración propia)

Proyecto	Metodología	Pruebas (aplicadas o planificadas)	Fase	Lenguajes de programación y base de datos.	IDE	*Framework*	SGBD	Sistema operativo (incluyendo servidor de base de datos)

Anexo 8. Metodologías de desarrollo de software usadas en los proyectos del CISED

Tabla 8.11. Metodologías de desarrollo de software usadas en los proyectos del CISED (Fuente: Elaboración propia)

Proyecto	Metodología
Integración Bancos	RUP
Migración: • Integración del Sistema de Gestión del SAIME con los Equipos de Autochequeo Migratorio (sub-proyecto)	RUP
Sistema de Gestión de Entidades Externas	SXP
Pasaporte Diplomático y de Servicio	FDD
Dactilab	RUP
Facelab	RUP
Signlab	RUP
Sistema de Identificación, Inmigración y Extranjería de la República de Cuba	MSF for CMMI
Plataforma para el Desarrollo de Servicios en Línea Utilizando Tarjetas Inteligentes	XP
Solución para la Gestión de la Información en Tarjetas Inteligentes: • Secure Data Manager Applet(sub-proyecto) • Secure Data Manager Middleware(sub-proyecto)	XP
Solución para la Gestión de Información en Credenciales Universitarias	XP
Applet para la Gestión de Información Personal y Servicios	XP
Middleware para la Gestión de Información Personal y Servicios	XP
Solución para la Lectura de Documentos de Viaje Electrónicos:	XP

• Applet LDS (sub-proyecto) • Middleware LDS (sub-proyecto)	
Solución para la Gestión de Información Biométrica en Tarjetas Inteligentes: • Applet para la Verificación Biométrica en Tarjetas Inteligentes (sub-proyecto) • Middleware para la Verificación Biométrica en Tarjetas Inteligentes (sub-proyecto)	XP
Sistema de Administración de Identidades	MSF for ASD
Sistema de Gestión de Servicios para la CIE	XP
Sistema para la Verificación del SOD	XP
Soluciones PKI	XP

Anexo 9. Patrones arquitectónicos usados en el CISED

Arquitectura de repositorio: está compuesta por una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él. Estos sistemas se han usado en aplicaciones que requieren complejas interpretaciones de proceso de señales (reconocimiento de patrones, reconocimiento de voz, etc.).

Arquitectura en capas: cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior. Al dividir un sistema en capas, cada una de ellas puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. Este sistema en capas facilita el diseño modular, en la que cada capa encapsula un aspecto concreto y permite además la construcción de sistemas débilmente acoplados. En la práctica, las capas suelen ser entidades complejas y están compuestas por varios paquetes o subsistemas.

Arquitectura orientada a objetos: los componentes de esta arquitectura se basan en principios orientados a objetos: encapsulamiento, herencia y polimorfismo. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos.

Arquitectura basada en componentes: los componentes son las unidades de modelado, diseño e implementación. Las interfaces están separadas de las implementaciones y conjuntamente con sus interacciones son el centro del diseño arquitectónico.

Arquitectura de máquinas virtuales: consiste en sistemas que se representan mediante un programa a interpretar y una máquina de interpretación. Estas variedades incluyen un extenso espectro que está comprendido desde los llamados lenguajes de alto nivel hasta los paradigmas declarativos no secuenciales de programación (entiéndase: lenguajes con pocas restricciones como los de bajo nivel).

Arquitectura basada en eventos: está basada en la promoción de la producción, la detección y el consumo. Puede ser usada en el diseño e implementación de aplicaciones y sistemas que transmiten eventos entre los componentes de software ligeramente acoplados.

Arquitectura orientada a servicios: construye toda la topología de la aplicación mediante implementaciones, interfaces y llamados a estas. Es una relación entre servicios y consumidores de servicios, ambos lo suficientemente amplios como para representar una función de negocio completa.

Arquitectura basada en recursos: define recursos identificables y métodos para acceder y manipular el estado de esos recursos. El caso de referencia es la *World Wide Web*, donde las direcciones identifican los recursos y HTTP es el protocolo de acceso. (Estrada Rodríguez, 2010)

Anexo 10. Pruebas que se aplican o se prevé que se apliquen en los proyectos del CISED

Tabla 10.12. Pruebas que se aplican o se prevé que se apliquen en los proyectos del CISED (Fuente: Elaboración propia)

Proyecto	Pruebas
Integración Bancos	Prueba de caja blanca, prueba de caja negra e integración
Migración: • Integración del Sistema de Gestión del SAIME con los Equipos de Autochequeo Migratorio (sub-proyecto)	Pruebas de caja negra, pruebas de caja blanca y pruebas unitarias
Sistema de Gestión de Entidades Externas	Caja negra
Pasaporte Diplomático y de Servicio	Prueba de caja negra, pruebas unitarias y pruebas de integración
Dactilab	No definidas
Facelab	No definidas

Signlab	No definidas
Sistema de Identificación, Inmigración y Extranjería de la República de Cuba	Pruebas unitarias, de carga, pruebas web, funcionales, de fiabilidad, usabilidad y aceptación
Plataforma para el Desarrollo de Servicios en Línea Utilizando Tarjetas Inteligentes	No definidas
Solución para la Gestión de la Información en Tarjetas Inteligentes: • Secure Data Manager Applet (sub-proyecto) • Secure Data Manager Middleware (sub-proyecto)	No definidas
Solución para la Gestión de Información en Credenciales Universitarias	No definidas
Applet para la Gestión de Información Personal y Servicios	No definidas
Middleware para la Gestión de Información Personal y Servicios	No definidas
Solución para la Lectura de Documentos de Viaje Electrónicos: • Applet LDS (sub-proyecto) • Middleware LDS (sub-proyecto)	No definidas
Solución para la Gestión de Información Biométrica en Tarjetas Inteligentes: • Applet para la Verificación Biométrica en Tarjetas Inteligentes (sub-proyecto) • Middleware para la Verificación Biométrica en Tarjetas Inteligentes (sub-proyecto)	No definidas
Sistema de Administración de Identidades	No definidas
Sistema de Gestión de Servicios para la CIE	Pruebas de caja negra
Sistema para la Verificación del SOD	Pruebas de caja negra
Soluciones PKI	Pruebas de caja negra

Anexo 11. Historias de usuario

Tabla 11.13. HU_1 Autenticar usuario (Fuente: Elaboración propia)

Historia de Usuario

Número: HU_1	**Usuario:** Usuario
Nombre historia: Autenticar usuario	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Rachel Hernández Pérez	
Descripción: El usuario introduce su nombre y su contraseña para acceder a la aplicación.	
Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error.	

Tabla 11.14. HU_2 Administrar usuario (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_2	**Usuario:** Administrador, Invitado al sitio
Nombre historia: Administrar usuario	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Rachel Hernández Pérez	
Descripción: Un administrador podrá elegir, crear un usuario, modificarlo o eliminarlo, siempre brindando los datos requeridos por la aplicación, en este caso: • Nombre • Apellidos • Usuario • Contraseña • Repetir contraseña • Correo • Seleccionar el rol del usuario (Usuario/Administrador)	

Al modificar o eliminar el usuario se muestra una lista con los siguientes datos: Nombre y Apellidos, Correo, Usuario y Rol. El invitado solo podrá registrarse sin seleccionar su rol, que es Usuario por defecto.
Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error.

Tabla 11.15. HU_4 Administrar proyecto (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_4	**Usuario:** Administrador
Nombre historia: Administrar proyecto	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Yuniel Martínez González	
Descripción: Un administrador podrá elegir, crear un proyecto, modificarlo o eliminarlo, siempre brindando los datos requeridos por la aplicación, en este caso: • Nombre del proyecto • Departamento al cual pertenece Al modificar o eliminar se muestra una lista con los siguientes datos: Nombre y Departamento.	
Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error.	

Tabla 11.16. HU_5 Administrar departamento (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_5	**Usuario:** Administrador
Nombre historia: Administrar departamento	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo

Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Yuniel Martínez González	
Descripción: Un administrador podrá elegir, crear un departamento, modificarlo o eliminarlo, siempre brindando los datos requeridos por la aplicación, en este caso: • Nombre del departamento Al modificar o eliminar se muestra una lista con los siguientes datos: Nombre.	
Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error.	

Tabla 11.17. HU_6 Administrar tipo de prueba (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_6	**Usuario:** Administrador
Nombre historia: Administrar tipo de prueba	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Rachel Hernández Pérez	
Descripción: Un administrador podrá elegir, crear un tipo de prueba, modificarla o eliminarla, siempre brindando los datos requeridos por la aplicación, en este caso: • Nombre del tipo de prueba Al modificar o eliminar se muestra una lista con los siguientes datos: Nombre.	
Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error.	

Tabla 11.18. HU_7 Administrar lenguaje (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_7	**Usuario:** Administrador

Nombre historia: Administrar lenguaje	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Rachel Hernández Pérez	
Descripción: Un administrador podrá elegir, crear un lenguaje, modificarlo o eliminarlo brindando para esta funcionalidad: • Nombre del lenguaje Al modificar o eliminar se muestra una lista con los siguientes datos: Nombre.	
Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error.	

Tabla 11.19. HU_8 Buscar herramienta (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_8	**Usuario:** Usuario, Administrador e invitados al sitio
Nombre historia: Buscar herramienta	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Yuniel Martínez González.	
Descripción: Cualquier usuario podrá realizar búsquedas generales de acuerdo a un término introducido o búsquedas avanzadas de acuerdo a los siguientes criterios: • Nombre • Desarrollador • Última versión estable • Año • Idioma (Inglés/Español/Otro/Multilenguaje) • Tipo de software (Libre/Propietario)	

• Licencia • Lenguaje de desarrollo • Sitio web oficial • Tipo de prueba • Lenguajes soportados • Complementos opcionales • Tamaño del archivo • Sistema operativo (Linux/Windows/Multiplataforma) • Requerimientos de software • Requerimientos de *hardware* • Palabras claves • Autor de la propuesta • Aceptada (Sí/No) Los resultados de la búsqueda se mostrarán en una lista que tendrá los siguientes datos: Nombre, Tipo de software, Sistema Operativo, Aceptada (Sí/No). Se podrá elegir una herramienta de la lista para visualizar sus datos.
Observaciones: Se mostrará un mensaje de error en caso de no introducir ningún criterio de búsqueda. Si no se encuentra ninguna herramienta que coincida con el criterio insertado se lanzará una alerta.

Tabla 11.20. HU_9 Proponer herramienta (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_9	**Usuario:** Usuario
Nombre historia: Proponer herramienta	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Yuniel Martínez González	
Descripción: Un usuario podrá proponer una herramienta siempre brindando los datos requeridos por la aplicación, en este caso:	

• Nombre* • Desarrollador* • Última versión estable* • Año* • Idioma* (Inglés/Español/Otro/Multilenguaje) • Tipo de software* (Libre/Propietario) • Licencia* • Lenguaje de desarrollo* • Sitio web oficial* • Tipo de prueba* • Lenguajes soportados* • Complementos opcionales • Tamaño del archivo* • Sistema operativo* (Linux/Windows/Multiplataforma) • Requerimientos de software* • Requerimientos de hardware* • Ayuda (tutorial) • Palabras claves* * Campos obligatorios
Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error. La herramienta será registrada por defecto como **no aceptada**.

Tabla 11.21. HU_10 Administrar mensajes de contacto (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_10	**Usuario:** Usuario, Administrador
Nombre historia: Administrar mensajes de contacto	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 3	**Iteración asignada:** 2

Programador responsable: Yuniel Martínez González
Descripción: Un administrador podrá enviar mensajes y ver los enviados por cualquier otro usuario, así como eliminarlos si lo desea. Un usuario solamente podrá enviar mensajes. Para enviar los mensajes se mostrará un formulario de contacto con los campos: • Asunto* • Mensaje* * Campos obligatorios También se registrarán: Remitente, Fecha y Hora del mensaje. En la página de contacto, se le mostrará una opción a los administradores, mediante la cual tendrán acceso a los mensajes recibidos. Podrán verlos o eliminarlos. Se mostrarán los datos registrados de los mensajes.
Observaciones: Si los campos se dejan en blanco se muestra un mensaje de error. Se validará la selección de un solo mensaje para visualizarlo.

Tabla 11.22. HU_11 Administrar tecnología (Fuente: Elaboración propia)

Historia de Usuario	
Número: HU_11	**Usuario:** Administrador
Nombre historia: Administrar tecnología	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Yuniel Martínez González	
Descripción: Un administrador podrá elegir, registrar una tecnología, modificarla o eliminarla, siempre brindando los datos requeridos por la aplicación, en este caso: • Nombre de la tecnología Al modificar o eliminar se muestra una lista con los siguientes datos: Nombre.	
Observaciones: Si los datos entrados son incorrectos, están en blanco o no coinciden con lo esperado, la aplicación muestra el mensaje correspondiente a cada error.	

Anexo 12. Plan de iteraciones

Tabla 12.23. Plan de iteraciones (Fuente: Elaboración propia)

Iteración	Número	Historia de usuario	Duración estimada (en días)
1	HU_5	Administrar departamento	41
	HU_6	Administrar tipo de prueba	
	HU_3	Administrar herramienta	
	HU_4	Administrar proyecto	
	HU_7	Administrar lenguaje	
2	HU_1	Autenticar usuario	16
	HU_2	Administrar usuario	
	HU_8	Buscar herramienta	
	HU_9	Proponer herramienta	
	HU_10	Administrar mensajes de contacto	
	HU_11	Administrar tecnología	

Anexo 13. Tareas de ingeniería

Tareas de ingeniería de la primera iteración

Tabla 13.24. HU_5 Tarea 1 Crear departamento (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de historia:** 5
Nombre de la tarea: Crear departamento	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 1 de abril del 2011	**Fecha fin:** 3 de abril del 2011

Programador responsable: Yuniel Martínez González
Descripción: Se añadirá una vista que será cargada mostrando los campos necesarios para crear un departamento. Se validarán los datos y se registrarán.

Tabla 13.25. HU_5 Tarea 2 Mostrar listado de departamentos (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de historia:** 5
Nombre de la tarea: Mostrar listado de departamentos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 3 de abril del 2011	**Fecha fin:** 5 de abril del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Crear una vista que cargue un listado de departamentos mostrando sus nombres.	

Tabla 13.26. HU_5 Tarea 3 Mostrar listado de departamentos (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de historia:** 5
Nombre de la tarea: Modificar los datos del departamento	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 6 de abril del 2011	**Fecha fin:** 8 de abril del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Permitir seleccionar un elemento de una lista de departamentos, mostrar sus datos dentro de un formulario pudiendo ser modificados; validarlos y actualizarlos en la base de datos.	

Tabla 13.27. HU_5 Tarea 4 Eliminar departamento (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de historia:** 5
Nombre de la tarea: Eliminar departamento	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 8 de abril del 2011	**Fecha fin:** 10 de abril del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Permitir seleccionar uno o varios departamentos dentro de una lista. Mostrar un mensaje de alerta, si se presiona el botón Eliminar. Eliminar la selección, si se acepta la alerta.	

Tabla 13.28. HU_6 Tarea 1 Crear prueba (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de historia:** 6
Nombre de la tarea: Crear prueba	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11 de abril del 2011	**Fecha fin:** 13 de abril del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Se añadirá una vista que será cargada mostrando los campos necesarios para crear una prueba. Se validarán los datos y se registrarán.	

Tabla 13.29. HU_6 Tarea 2 Mostrar listado de pruebas (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de historia:** 6
Nombre de la tarea: Mostrar listado de pruebas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 13 de abril del 2011	**Fecha fin:** 15 de abril del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Crear una vista que cargue un listado de pruebas mostrando sus nombres.	

Tabla 13.30. HU_6 Tarea 3 Modificar los datos de la prueba (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de historia:** 6
- **Nombre de la tarea:** Modificar los datos de la prueba	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17 de abril del 2011	**Fecha fin:** 19 de abril del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Permitir seleccionar un elemento de una lista de pruebas, mostrar sus datos dentro de un formulario pudiendo ser modificados; validarlos y actualizarlos en la base de datos.	

Tabla 13.31. HU_6 Tarea 4 Eliminar prueba (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de historia:** 6
Nombre de la tarea: Eliminar prueba	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1

Fecha inicio: 19 de abril del 2011	**Fecha fin:** 21 de abril del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Permitir seleccionar una o varias pruebas dentro de una lista. Mostrar un mensaje de alerta, si se presiona el botón Eliminar. Eliminar la selección, si se acepta la alerta.	

Tabla 13.32. HU_3 Tarea 1 Registrar herramienta (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de historia:** 3
Nombre de la tarea: Registrar herramienta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 22 de abril del 2011	**Fecha fin:** 24 de abril del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Se añadirá una vista que será cargada mostrando los campos necesarios para crear una herramienta. Se validarán los datos y se registrarán.	

Tabla 13.33. HU_3 Tarea 2 Mostrar listado de herramientas (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de historia:** 3
Nombre de la tarea: Mostrar listado de herramientas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 24 de abril del 2011	**Fecha fin:** 26 de abril del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Crear una vista que cargue el listado de herramientas mostrando sus nombres.	

Tabla 13.34. HU_3 Tarea 3 Modificar herramienta (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de historia:** 3
Nombre de la tarea: Modificar herramienta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 27 de abril del 2011	**Fecha fin:** 29 de abril del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Permitir seleccionar un elemento de una lista de herramientas, mostrar sus datos dentro de un formulario pudiendo ser modificados; validarlos y actualizarlos en la base de datos.	

Tabla 13.35. HU_3 Tarea 4 Eliminar herramienta (Fuente: Elaboración propia)

Tarea	
Número de tarea:4	**Número de historia:** 3
Nombre de la tarea: Eliminar herramienta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 29 de abril del 2011	**Fecha fin:** 2 de mayo del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Permitir seleccionar una o varias herramientas dentro de una lista. Mostrar un mensaje de alerta, si se presiona el botón Eliminar. Eliminar la selección, si se acepta la alerta.	

Tabla 13.36. HU_4 Tarea 1 Registrar proyecto (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de historia:** 4
Nombre de la tarea: Registrar proyecto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 3 de mayo del 2011	**Fecha fin:** 5 de mayo del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Se añadirá una vista que será cargada mostrando los campos necesarios para crear un proyecto. Se validarán los datos y se registrarán.	

Tabla 13.37. HU_4 Tarea 2 Mostrar listado de proyectos (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de historia:** 4
Nombre de la tarea: Mostrar listado de proyectos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 5 de mayo del 2011	**Fecha fin:** 7 de mayo del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Crear una vista que cargue el listado de proyectos mostrando sus nombres.	

Tabla 13.38. HU_4 Tarea 3 Modificar los datos del proyecto (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de historia:** 4
Nombre de la tarea: Modificar los datos del proyecto	

Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 8 de mayo del 2011	**Fecha fin:** 10 de mayo del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Permitir seleccionar un elemento de una lista de proyectos, mostrar sus datos dentro de un formulario pudiendo ser modificados; validarlos y actualizarlos en la base de datos.	

Tabla 13.39. HU_4 Tarea 4 Eliminar proyecto (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de historia:** 4
Nombre de la tarea: Eliminar proyecto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10 de mayo del 2011	**Fecha fin:** 12 de mayo del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Permitir seleccionar uno o varios proyectos dentro de una lista. Mostrar un mensaje de alerta, si se presiona el botón Eliminar. Eliminar la selección, si se acepta la alerta.	

Tareas de ingeniería de la segunda iteración

Tabla 13.40. HU_1 Tarea 1 Desarrollar la autenticación del sistema (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de historia:** 1
Nombre de la tarea: Desarrollar la autenticación del sistema	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 13 de mayo del 2011	**Fecha fin:** 14 de mayo del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Cargar una vista que solicite usuario y contraseña. Validar los datos introducidos y asignar un rol en caso de ser válidos, permitiendo que el usuario tenga acceso a las funcionalidades que le corresponden. Mostrar un mensaje de error en caso de no ser válidos los datos.	

Tabla 13.41. HU_2 Tarea 1 Registrar usuario (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de historia:** 2
Nombre de la tarea: Registrar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1

Fecha inicio: 15 de mayo del 2011	**Fecha fin:** 17 de mayo del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Se añadirá una vista que será cargada mostrando los campos necesarios para crear un usuario. Se validarán los datos y se registrarán.	

Tabla 13.42. HU_2 Tarea 2 Mostrar listado de usuarios (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de historia:** 2
Nombre de la tarea: Mostrar listado de usuarios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17 de mayo del 2011	**Fecha fin:** 19 de mayo del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Crear una vista que cargue el listado de usuarios mostrando sus nombres.	

Tabla 13.43. HU_2 Tarea 3 Modificar los datos del usuario (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de historia:** 2
Nombre de la tarea: Modificar los datos del usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 20 de mayo del 2011	**Fecha fin:** 22 de mayo del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Permitir seleccionar un elemento de una lista de usuarios, mostrar sus datos dentro de un formulario pudiendo ser modificados; validarlos y actualizarlos en la base de datos.	

Tabla 13.44. HU_2 Tarea 4 Eliminar usuario (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de historia:** 2
Nombre de la tarea: Eliminar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 22 de mayo del 2011	**Fecha fin:** 24 de mayo del 2011
Programador responsable: Rachel Hernández Pérez	
Descripción: Permitir seleccionar uno o varios usuarios dentro de una lista. Mostrar un mensaje de alerta, si se presiona el botón Eliminar. Eliminar la selección, si se acepta la alerta.	

Tabla 13.45. HU_9 Tarea 1 Proponer herramienta (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de historia:** 9
Nombre de la tarea: Proponer herramienta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 25 de mayo del 2011	**Fecha fin:** 26 de mayo del 2011
Programador responsable: Yuniel MartínezGonzález	
Descripción: Se añadirá una vista que será cargada mostrando los campos necesarios para registrar una herramienta. Se validarán los datos y se registrarán.	

Tabla 13.46. HU_10 Tarea 1 Enviar mensaje (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de historia:** 10
Nombre de la tarea: Enviar mensaje	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 27 de mayo del 2011	**Fecha fin:** 27 de mayo del 2011
Programador responsable: Yuniel MartínezGonzález	
Descripción: Se mostrará un formulario para introducir los datos del mensaje, los cuales serán validados y registrados.	

Tabla 13.47. HU_10 Tarea 2 Eliminar mensaje (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de historia:** 10
Nombre de la tarea: Eliminar mensaje	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 27 de mayo del 2011	**Fecha fin:** 28 de mayo del 2011
Programador responsable: Yuniel MartínezGonzález	
Descripción: Se listarán los mensajes enviados, permitiendo la selección de uno o varios mensajes para su eliminación. Se validará la selección.	

Tabla 13.48. HU_10 Tarea 3 Mostrar mensaje (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de historia:** 10

Nombre de la tarea: Mostrar mensaje	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 28 de mayo del 2011	**Fecha fin:** 28 de mayo del 2011
Programador responsable: Yuniel MartínezGonzález	
Descripción: Se listarán los mensajes enviados, permitiendo la selección de un mensaje para su lectura. Se validará la selección.	

Tabla 13.49. HU_11 Tarea 1 Registrar tecnología (Fuente: Elaboración propia)

Tarea	
Número de tarea: 1	**Número de historia:** 11
Nombre de la tarea: Registrar tecnología	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 29 de mayo del 2011	**Fecha fin:** 30 de mayo del 2011
Programador responsable: Yuniel MartínezGonzález	
Descripción: Se añadirá una vista que será cargada mostrando el campo Nombre (de la tecnología). Se validará y se registrará.	

Tabla 13.50. HU_11 Tarea 2 Mostrar listado de tecnologías (Fuente: Elaboración propia)

Tarea	
Número de tarea: 2	**Número de historia:** 11
Nombre de la tarea: Mostrar listado de tecnologías	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 29 de mayo del 2011	**Fecha fin:** 30 de mayo del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Crear una vista que cargue el listado de tecnologías mostrando sus nombres.	

Tabla 13.51. HU_11 Tarea 3 Modificar los datos de la tecnología (Fuente: Elaboración propia)

Tarea	
Número de tarea: 3	**Número de historia:** 11
Nombre de la tarea: Modificar los datos de la tecnología	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 29 de mayo del 2011	**Fecha fin:** 30 de mayo del 2011
Programador responsable: Yuniel Martínez González	

Descripción: Permitir seleccionar un elemento de una lista de tecnologías, mostrar sus datos dentro de un formulario pudiendo ser modificados; validarlos y actualizarlos en la base de datos.

Tabla 13.52. HU_11 Tarea 4 Eliminar tecnología (Fuente: Elaboración propia)

Tarea	
Número de tarea: 4	**Número de historia:** 11
Nombre de la tarea: Eliminar tecnología	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 29 de mayo del 2011	**Fecha fin:** 30 de mayo del 2011
Programador responsable: Yuniel Martínez González	
Descripción: Permitir seleccionar uno o varias tecnologías dentro de una lista. Mostrar un mensaje de alerta, si se presiona el botón Eliminar. Eliminar la selección, si se acepta la alerta.	

Anexo 14. Tarjetas CRC

Tabla 14.53. Tarjeta CRC Usuario (Fuente: Elaboración propia)

Usuario	
Responsabilidades	**Colaboradores**
- Permitir introducir los datos del usuario - Comprobar que los datos del usuario sean correctos - Otorgar permisos al usuario de acuerdo a un rol - Registrar usuario - Mostrar listado de usuarios - Permitir seleccionar el usuario a modificar - Mostrar los datos de un usuario seleccionado - Permitir introducir los datos a modificar - Modificar los datos del usuario - Permitir seleccionar el/los usuario(s) a eliminar - Permitir eliminar el usuario	- Mensaje - Herramienta

Tabla 14.54. Tarjeta CRC Herramienta (Fuente: Elaboración propia)

Herramienta	
Responsabilidades	**Colaboradores**
- Registrar herramienta	- Proyecto

- Mostrar listado de herramientas - Permitir seleccionar la herramienta a modificar - Mostrar los datos de la herramienta - Permitir introducir los datos a modificar si se desea - Modificar los datos de la herramienta - Permitir seleccionar la(s) herramienta(s) a eliminar - Mostrar una alerta de eliminación - Permitir eliminar la(s) herramienta(s) - Proponer herramienta - Buscar herramienta	- Tipo_Prueba - Lenguaje

Tabla 14.55. Tarjeta CRC Proyecto (Fuente: Elaboración propia)

Proyecto	
Responsabilidades	**Colaboradores**
- Registrar proyecto - Mostrar listado de proyectos - Permitir seleccionar el proyecto a modificar - Mostrar los datos del proyecto - Permitir introducir los datos a modificar - Modificar los datos del proyecto - Permitir seleccionar el/los proyecto(s) a eliminar - Mostrar una alerta de eliminación - Permitir eliminar el/los proyecto(s)	- Herramienta - Departamento

Tabla 14.56. Tarjeta CRC Tipo_Prueba (Fuente: Elaboración propia)

Tipo_Prueba	
Responsabilidades	**Colaboradores**
- Crear prueba - Mostrar listado de pruebas - Permitir seleccionar la prueba a modificar - Mostrar los datos de la prueba - Permitir introducir los datos a modificar si se desea - Modificar los datos de la prueba	- Herramienta

- Permitir seleccionar la(s) prueba(s) a eliminar - Mostrar una alerta de eliminación - Permitir eliminar la(s) prueba(s)	

Tabla 14.57. Tarjeta CRC Departamento (Fuente: Elaboración propia)

Departamento	
Responsabilidades	**Colaboradores**
- Crear departamento - Mostrar listado de departamentos - Permitir seleccionar el departamento a modificar - Mostrar los datos del departamento - Permitir introducir los datos a modificar si se desea - Modificar los datos del departamento - Permitir seleccionar el/los departamento(s) a eliminar - Mostrar una alerta de eliminación - Eliminar el/los departamento(s)	- Proyecto

Tabla 14.58. Tarjeta CRC Lenguaje (Fuente: Elaboración propia)

Lenguaje	
Responsabilidades	**Colaboradores**
- Crear lenguaje - Mostrar listado de lenguajes - Permitir seleccionar el lenguaje a modificar - Mostrar los datos del lenguaje - Permitir introducir los datos a modificar si se desea - Modificar los datos del lenguaje - Permitir seleccionar el/los lenguaje(s) a eliminar - Mostrar una alerta de eliminación - Eliminar el/los lenguaje(s)	- Herramienta

Tabla 14.59. Tarjeta CRC Mensaje (Fuente: Elaboración propia)

Mensaje	
Responsabilidades	**Colaboradores**

- Enviar mensaje - Mostrar listado de mensaje - Eliminar mensaje - Permitir seleccionar el/los mensaje(s) a eliminar - Mostrar una alerta de eliminación	- Usuario

Tabla 14.60. Tarjeta CRC Tecnología (Fuente: Elaboración propia)

Tecnología	
Responsabilidades	**Colaboradores**
- Crear tecnología - Mostrar listado de tecnologías - Permitir seleccionar la tecnología a modificar - Mostrar los datos de la tecnología - Permitir introducir los datos a modificar si se desea - Modificar el nombre de la tecnología - Permitir seleccionar la(s) tecnología(s) a eliminar - Mostrar una alerta de eliminación - Eliminar la(s) tecnología (s)	- Herramienta

Anexo 15. Casos de prueba de aceptación

Nótese en las tablas de los casos de prueba un código (Ejemplo: HU1_P1) que indica el número de la HU, y dentro de esta, el número de la prueba (P). A continuación se representan las pruebas de aceptación para las HU:

Tabla 15.61. Caso de prueba de aceptación HU1_P1 "Autenticar usuario" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Autenticar usuario"		
Código: HU1_P1		**Historia de Usuario:** Autenticar usuario
Descripción de la funcionalidad: Prueba de funcionalidad para la autenticación del usuario.		
Condiciones de ejecución: -		
Flujo central: Ir al bloque "Conectarse". Ingresar: Usuario y Contraseña. Ejecutar la opción "Entrar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra el bloque "Mi cuenta" a la izquierda con el nombre del usuario y las opciones "Modificar" y "Salir".

		En caso de ser administrador, se muestran a la derecha las opciones de administración.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.62. Caso de prueba de aceptación HU1_P2 "Salir del sistema" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Salir del sistema"		
Código: HU1_P2		**Historia de Usuario:** Autenticar usuario
Descripción de la funcionalidad: Prueba de funcionalidad para eliminar la autenticación del sistema.		
Condiciones de ejecución: El usuario tiene que estar autenticado.		
Flujo central: Ir al panel "Mi cuenta", opción "Salir".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
		La interfaz muestra las opciones por defecto del sistema.

Tabla 15.63. Caso de prueba de aceptación HU2_P1 "Registrar usuario" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Registrar usuario"		
Código: HU2_P1		**Historia de Usuario:** Administrar usuario
Descripción de la funcionalidad: Prueba de funcionalidad para el registro de un usuario.		
Condiciones de ejecución: Para registrar varios usuarios con el rol Administrador, un administrador tiene que estar autenticado.		
Flujo central: Por un administrador: Ir al panel "Administración" > "Usuarios" > "Registrar". Se ingresan: Nombre, Apellidos, Correo, Usuario, Contraseña y se repite, Rol, y se ejecuta la acción "Registrar". Por un invitado: Ir al bloque "Conectarse" > "Registrar"; el dato "Rol" no se visualizará.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		En caso de ser administrador quien realice la operación: se muestra un mensaje indicando que se ha insertado correctamente el usuario, permitiendo el registro de otro usuario. En caso de ser otra persona quien realice la operación: se muestra un mensaje indicando que se ha insertado correctamente el usuario.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.

	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.64. Caso de prueba de aceptación HU2_P2 "Modificar usuario" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Modificar usuario"		
Código: HU2_P2		**Historia de Usuario:** Administrar usuario
Descripción de la funcionalidad: Prueba de funcionalidad para la modificación de los datos del usuario.		
Condiciones de ejecución: Un administrador tiene que estar autenticado para poder probar la modificación de otros usuarios mediante el panel "Administración". Un usuario tiene que estar autenticado para modificar sus datos.		
Flujo central: Si es administrador quien realiza la operación: Ir al panel "Administración" > "Usuarios" > "Modificar". Seleccionar un usuario de la lista mostrada y ejecutar "Modificar". Se modifican cualquiera de los datos del usuario: Nombre, Apellidos, Correo, Usuario, Contraseña y se repite, Rol, y ejecutar la acción "Modificar". En otro caso: el usuario autenticado puede modificar sus datos en el bloque "Mi cuenta", opción "Modificar" y el campo "Rol" no se visualizará.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de un usuario a modificar.		Se muestra un formulario con los datos del usuario permitiendo su modificación.
	No se selecciona un usuario a modificar.	Se muestra un mensaje indicando que se debe seleccionar un usuario.
Introducción correcta de los datos del usuario.		En caso de ser administrador quien realice la operación: se muestra un mensaje indicando que se ha modificado correctamente el usuario, permitiendo la selección de otro usuario. En caso de ser otra persona quien realice la operación: se muestra un mensaje indicando que se ha modificado correctamente el usuario.
	Introducción incorrecta de los datos del usuario.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.65. Caso de prueba de aceptación HU2_P3 "Eliminar usuario" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Eliminar usuario"	
Código: HU2_P3	**Historia de Usuario:** Administrar usuario

Descripción de la funcionalidad: Prueba de funcionalidad para eliminar uno o varios usuarios.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel “Administración” > “Usuarios” > “Eliminar”. Seleccionar un usuario de la lista mostrada o varios de ellos. Ejecutar la acción “Eliminar”. Aceptar la alerta de eliminación.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de uno o varios usuarios a eliminar.		Se muestra una alerta de eliminación. Si se acepta, se elimina la selección y se muestra la lista actualizada de usuarios. Si no se acepta, la lista se muestra como estaba.
	No se selecciona un usuario a eliminar.	Se muestra un mensaje indicando que se debe seleccionar un usuario y se vuelve a mostrar la lista de usuarios.

Tabla 15.66. Caso de prueba de aceptación HU3_P1 “Registrar herramienta” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Registrar herramienta”		
Código: HU3_P1		**Historia de Usuario:** Administrar herramienta
Descripción de la funcionalidad: Prueba de funcionalidad para el registro de una herramienta.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel “Administración” > “Herramientas” > “Registrar”. Introducir los datos: Nombre*, Desarrollador*, Última versión estable*, Año*, Idioma* (Inglés/Español/Otro/Multilenguaje), Tipo de software* (Libre/Propietario), Licencia*, Lenguaje de desarrollo*, Sitio web oficial*, Tipo de prueba*, Lenguajes soportados*, Complementos opcionales, Tamaño del archivo*, Sistema operativo*, Linux/Windows/Multiplataforma), Requerimientos de software*, Requerimientos de *hardware**, Ayuda (tutorial), Palabras claves*. Ejecutar la acción “Registrar”. *Campos obligatorios		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha insertado correctamente la herramienta, permitiendo el registro de otra.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.67. Caso de prueba de aceptación HU3_P2 "Modificar herramienta" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Modificar herramienta"		
Código: HU3_P2	**Historia de Usuario:** Administrar herramienta	
Descripción de la funcionalidad: Prueba de funcionalidad para modificar una herramienta.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Herramientas" > "Modificar". Seleccionar una herramienta de la lista mostrada y ejecutar "Modificar". Modificar alguno de los datos: Nombre*, Desarrollador*, Última versión estable*, Año*, Idioma* (Inglés/Español/Otro/Multilenguaje), Tipo de software* (Libre/Propietario), Licencia*, Lenguaje de desarrollo*, Sitio web oficial*, Tipo de prueba*, Lenguajes soportados*, Complementos opcionales, Tamaño del archivo*, Sistema operativo* (Linux/Windows/Multiplataforma), Requerimientos de software*, Requerimientos de *hardware**, Ayuda (tutorial), Palabras claves*, Autor de la propuesta* y Aceptada* (Sí/No). Ejecutar la acción "Modificar". *Campos obligatorios		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha insertado correctamente la herramienta, permitiendo el registro de otra.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.68. Caso de prueba de aceptación HU3_P3 "Eliminar herramienta" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Eliminar herramienta"		
Código: HU3_P3	**Historia de Usuario:** Administrar herramienta	
Descripción de la funcionalidad: Prueba de funcionalidad para eliminar una o varias herramientas.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Herramientas" > "Eliminar". Seleccionar una o varias herramientas de la lista que se muestra. Ejecutar la acción "Eliminar". Aceptar la alerta de eliminación.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de una o varias herramientas.		Se muestra una alerta de eliminación. Si se acepta, se elimina la selección y se muestra la lista actualizada de herramientas. Si no se acepta, la lista se muestra como estaba.

	No se seleccionan herramientas.	Se muestra un mensaje indicando que se debe seleccionar una herramienta y se vuelve a mostrar la lista de herramientas.

Tabla 15.69. Caso de prueba de aceptación HU4_P1 "Registrar proyecto" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Registrar proyecto"		
Código: HU4_P1		**Historia de Usuario:** Administrar proyecto
Descripción de la funcionalidad: Prueba de funcionalidad para registrar un proyecto.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Proyectos" > "Registrar". Introducir: Nombre y Departamento. Ejecutar la acción "Registrar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha insertado correctamente el proyecto, permitiendo el registro de otro.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.70. Caso de prueba de aceptación HU4_P2 "Modificar proyecto" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Modificar proyecto"		
Código: HU4_P2		**Historia de Usuario:** Administrar proyecto
Descripción de la funcionalidad: Prueba de funcionalidad para modificar un proyecto.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Proyectos" > "Modificar". Seleccionar un proyecto de la lista que se muestra. Modificar cualquiera de los datos: Nombre y Departamento. Ejecutar la acción "Modificar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha modificado correctamente el proyecto, permitiendo la selección de otro.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los

		campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.71. Caso de prueba de aceptación HU4_P3 "Eliminar proyecto" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Eliminar proyecto"		
Código: HU4_P3		**Historia de Usuario:** Administrar proyecto
Descripción de la funcionalidad: Prueba de funcionalidad para eliminar uno o varios proyectos.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Proyectos" > "Eliminar". Seleccionar uno o varios proyectos de la lista que se muestra. Ejecutar la acción "Eliminar". Aceptar la alerta de eliminación.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de uno varios proyectos.		Se muestra una alerta de eliminación. Si se acepta, se elimina la selección y se muestra la lista de proyectos actualizada. Si no se acepta, la lista se muestra como estaba.
	No se seleccionan proyectos.	Se muestra un mensaje indicando que se debe seleccionar un proyecto y se vuelve a mostrar la lista.

Tabla 15.72. Caso de prueba de aceptación HU5_P1 "Registrar departamento" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Registrar departamento"		
Código: HU5_P1		**Historia de Usuario:** Administrar departamento
Descripción de la funcionalidad: Prueba de funcionalidad para registrar un departamento.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Departamentos" > "Registrar". Introducir el nombre del departamento. Ejecutar la acción "Registrar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha insertado correctamente el departamento, permitiendo el registro de otro.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo nuevamente.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo nuevamente.

Tabla 15.73. Caso de prueba de aceptación HU5_P2 "Modificar departamento" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Modificar departamento"		
Código: HU5_P2		**Historia de Usuario:** Administrar departamento
Descripción de la funcionalidad: Prueba de funcionalidad para modificar un departamento.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Departamentos" > "Modificar". Seleccionar un departamento de la lista que se muestra. Modificar el nombre del departamento. Ejecutar la acción "Modificar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha modificado correctamente el departamento, permitiendo la selección de otro.
	Introducción incorrecta de algunos de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.74. Caso de prueba de aceptación HU5_P3 "Eliminar departamento" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Eliminar departamento"		
Código: HU5_P3		**Historia de Usuario:** Administrar departamento
Descripción de la funcionalidad: Prueba de funcionalidad para eliminar uno o varios departamentos.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Departamentos" > "Eliminar". Seleccionar uno o varios departamentos de la lista que se muestra. Ejecutar la acción "Eliminar". Aceptar la alerta de eliminación.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de uno o varios departamentos.		Se muestra una alerta de eliminación. Si se acepta, se elimina la selección y se muestra la lista de departamentos actualizada. Si no se acepta, la lista se muestra como estaba.
	No se seleccionan departamentos.	Se muestra un mensaje indicando que se debe seleccionar un departamento y se vuelve a mostrar la lista.

Tabla 15.75. Caso de prueba de aceptación HU6_P1 "Registrar tipo de prueba" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Registrar tipo de prueba"

Código: HU6_P1		**Historia de Usuario:** Administrar tipo de prueba
Descripción de la funcionalidad: Prueba de funcionalidad para registrar un tipo de prueba.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel “Administración” > “Tipos de Prueba” > “Registrar”. Introducir el nombre del tipo de prueba. Ejecutar la acción “Registrar”.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha insertado correctamente el tipo de prueba, permitiendo el registro de otro.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo nuevamente.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo nuevamente.

Tabla 15.76. Caso de prueba de aceptación HU6_P2 “Modificar tipo de prueba” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Modificar tipo de prueba”		
Código: HU6_P2		**Historia de Usuario:** Administrar tipo de prueba
Descripción de la funcionalidad: Prueba de funcionalidad para modificar un tipo de prueba.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel “Administración” > “Tipos de Prueba” > “Modificar”. Seleccionar un tipo de prueba de la lista que se muestra. Modificar el nombre del tipo de prueba. Ejecutar la acción “Modificar”.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha modificado correctamente el tipo de prueba, permitiendo la selección de otro.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.77. Caso de prueba de aceptación HU6_P3 “Eliminar tipo de prueba” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Eliminar tipo de prueba”

Código: HU6_P3		**Historia de Usuario:** Administrar tipo de prueba
Descripción de la funcionalidad: Prueba de funcionalidad para eliminar uno o varios tipos de prueba.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel “Administración” > “Tipos de Prueba” > “Eliminar”. Seleccionar uno o varios tipos de prueba de la lista que se muestra. Ejecutar la acción “Eliminar”. Aceptar la alerta de eliminación.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de uno o varios tipos de prueba.		Se muestra una alerta de eliminación. Si se acepta, se elimina la selección y se muestra la lista actualizada de tipos de prueba. Si no se acepta, la lista se muestra como estaba.
	No se seleccionan tipos de prueba.	Se muestra un mensaje indicando que se debe seleccionar un tipo de prueba y se vuelve a mostrar la lista.

Tabla 15.78. Caso de prueba de aceptación HU7_P1 “Registrar lenguaje” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Registrar lenguaje”		
Código: HU7_P1		**Historia de Usuario:** Administrar lenguaje
Descripción de la funcionalidad: Prueba de funcionalidad para registrar un lenguaje.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel “Administración” > “Lenguajes” > “Registrar”. Introducir el nombre del lenguaje. Ejecutar la acción “Registrar”.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha insertado correctamente el lenguaje, permitiendo el registro de otro.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo nuevamente.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo nuevamente.

Tabla 15.79. Caso de prueba de aceptación HU7_P2 “Modificar lenguaje” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Modificar lenguaje”

Código: HU7_P2		**Historia de Usuario:** Administrar lenguaje
Descripción de la funcionalidad: Prueba de funcionalidad para modificar un lenguaje.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel “Administración” > “Lenguajes” > “Modificar”. Seleccionar un lenguaje de la lista que se muestra. Modificar el nombre del lenguaje. Ejecutar la acción “Modificar”.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha modificado correctamente el lenguaje, permitiendo la selección de otro.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.80. Caso de prueba de aceptación HU7_P3 “Eliminar lenguaje” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Eliminar lenguaje”		
Código: HU7_P3		**Historia de Usuario:** Administrar lenguaje
Descripción de la funcionalidad: Prueba de funcionalidad para eliminar uno o varios lenguajes.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel “Administración” > “Lenguajes” > “Eliminar”. Seleccionar uno o varios lenguajes de la lista que se muestra. Ejecutar la acción “Eliminar”. Aceptar la alerta de eliminación.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de uno o varios lenguajes.		Se muestra una alerta de eliminación. Si se acepta, se elimina la selección y se muestra la lista actualizada de lenguajes. Si no se acepta, la lista se muestra como estaba.
	No se seleccionan lenguajes.	Se muestra un mensaje indicando que se debe seleccionar un lenguaje y se vuelve a mostrar la lista.

Tabla 15.81. Caso de prueba de aceptación HU8_P1 “Búsqueda general” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Búsqueda general”	
Código: HU8_P1	**Historia de Usuario:** Buscar herramienta
Descripción de la funcionalidad: Prueba de funcionalidad para buscar herramientas de manera general.	
Condiciones de ejecución: -	

Flujo central: En el bloque “Buscar” escribir cualquier palabra y ejecutar la acción “Buscar”.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción de cualquier palabra.		Se muestra una lista de herramientas de acuerdo a la palabra introducida.
	Campo en blanco.	Se muestra un mensaje indicando que se debe introducir una palabra.

Tabla 15.82. Caso de prueba de aceptación HU8_P2 “Búsqueda avanzada” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Búsqueda avanzada”		
Código: HU8_P2		**Historia de Usuario:** Buscar herramienta
Descripción de la funcionalidad: Prueba de funcionalidad para buscar herramientas de manera avanzada.		
Condiciones de ejecución: -		
Flujo central: En el bloque “Buscar” seleccionar “Avanzada”. En la página seleccionar uno o varios de los siguientes criterios: Nombre, Desarrollador, Año, Idioma (Inglés/Español/Otro/Multilenguaje), Tipo de software (Libre/Propietario), Licencia, Lenguaje de desarrollo, Sistema operativo, (Linux/Windows/Multiplataforma), Tipo de prueba, Lenguajes soportados, Autor de la propuesta. Ejecutar la acción “Buscar”.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Seleccionar uno o varios criterios.		Se muestra una lista de herramientas de acuerdo a los criterios seleccionados.
	Campo en blanco.	Se muestra un mensaje indicando que se debe seleccionar algún criterio.

Tabla 15.83. Caso de prueba de aceptación HU9_P1 “Proponer herramienta” (Fuente: Elaboración propia)

Caso de prueba de aceptación “Proponer herramienta”	
Código: HU9_P2	**Historia de Usuario:** Proponer herramienta
Descripción de la funcionalidad: Prueba de funcionalidad para proponer una herramienta.	
Condiciones de ejecución: El usuario tiene que estar autenticado.	
Flujo central: Seleccionar “Proponer Herramienta” en el menú superior. Introducir los datos: Nombre*, Desarrollador*, Última versión estable*, Año*, Idioma* (Inglés/Español/Otro/Multilenguaje), Tipo de software* (Libre/Propietario), Licencia*, Lenguaje de desarrollo*, Sitio web oficial*, Tipo de prueba*, Lenguajes soportados*, Complementos opcionales, Tamaño del archivo*, Sistema operativo *(Linux/Windows/Multiplataforma), Requerimientos de software*, Requerimientos de *hardware*, Ayuda (tutorial) y Palabras claves*. Ejecutar la acción “Enviar”.	

*Campos obligatorios		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción de todos los campos de manera correcta.		Se muestra un mensaje indicando que la herramienta ha sido propuesta correctamente.
	Introducción de algún campo de manera incorrecta.	Se muestra un mensaje indicando los campos introducidos incorrectamente.
	Campos requeridos en blanco.	Se muestra un mensaje de error indicando los campos requeridos que se han dejado en blanco.

Tabla 15.84. Caso de prueba de aceptación HU10_P1 "Enviar mensaje" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Enviar mensaje"		
Código: HU10_P1		**Historia de Usuario:** Administrar mensajes de contacto
Descripción de la funcionalidad: Prueba de funcionalidad para enviar un mensaje.		
Condiciones de ejecución: El usuario tiene que estar autenticado.		
Flujo central: En el menú "Contáctenos" introducir el asunto y el cuerpo del mensaje. Luego seleccionar la opción "Enviar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducir todos los datos correctamente.		Se muestra un cartel indicando que el mensaje fue enviado correctamente.
	Campos en blanco.	Se muestra un mensaje indicando que se debe introducir asunto y cuerpo del mensaje.

Tabla 15.85. Caso de prueba de aceptación HU10_P2 "Mostrar mensaje" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Mostrar mensaje"	
Código: HU10_P2	**Historia de Usuario:** Administrar mensajes de contacto
Descripción de la funcionalidad: Prueba de funcionalidad para mostrar los mensajes enviados por los usuarios.	
Condiciones de ejecución: Un administrador tiene que estar autenticado y debe haber al menos 1 mensaje en la bandeja de entrada.	
Flujo central: En la página "Contáctenos" seleccionar la opción "Mostrar mensajes". Luego seleccionar un mensaje y dar clic en "Leer mensaje".	

Clases válidas	Clases inválidas	Resultado esperado
Seleccionar un mensaje.		Se muestra el mensaje y sus datos: Asunto, Remitente, Fecha, Hora y Mensaje.
	No se selecciona ningún mensaje.	Se muestra un mensaje indicando que se debe seleccionar un mensaje.
	Se selecciona más de un mensaje para leer.	Se muestra un mensaje indicando que se debe seleccionar solo un mensaje.

Tabla 15.86. Caso de prueba de aceptación HU10_P3 "Eliminar mensaje" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Eliminar mensaje"		
Código: HU10_P3		**Historia de Usuario:** Administrar mensajes de contacto
Descripción de la funcionalidad: Prueba de funcionalidad para eliminar.		
Condiciones de ejecución: Un administrador tiene que estar autenticado y debe haber al menos 1 mensaje en la bandeja de entrada.		
Flujo central: En la página "Contáctenos" seleccionar la opción "Mostrar mensajes". Luego seleccionar uno o varios mensajes y dar clic en "Eliminar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Seleccionar uno o varios mensajes.		Son eliminados correctamente los mensajes seleccionados. Se muestra un cartel indicando la cantidad de mensajes que han sido eliminados.
	No se selecciona ningún mensaje.	Se muestra un mensaje indicando que se debe seleccionar un mensaje.

Tabla 15.87. Caso de prueba de aceptación HU11_P1 "Registrar tecnología" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Registrar tecnología"		
Código: HU11_P1		**Historia de Usuario:** Administrar tecnología
Descripción de la funcionalidad: Prueba de funcionalidad para registrar una tecnología.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Tecnologías" > "Registrar". Introducir el nombre de la tecnología. Ejecutar la acción "Registrar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha insertado correctamente la tecnología, permitiendo el registro de otra.

	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo nuevamente.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo nuevamente.

Tabla 15.88. Caso de prueba de aceptación HU11_P2 "Modificar tecnología" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Modificar tecnología"		
Código: HU11_P2		**Historia de Usuario:** Administrar tecnología
Descripción de la funcionalidad: Prueba de funcionalidad para modificar una tecnología.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Tecnologías" > "Modificar". Seleccionar una tecnología de la lista que se muestra. Modificar el nombre de la tecnología. Ejecutar la acción "Modificar".		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Introducción correcta de los datos.		Se muestra un mensaje indicando que se ha modificado correctamente la tecnología, permitiendo la selección de otro.
	Introducción incorrecta de alguno de los datos.	Se muestra un mensaje indicando que alguno de los datos es incorrecto, permitiendo introducirlo de nuevo.
	Campos en blanco.	Se muestra un mensaje indicando que alguno de los campos es requerido, permitiendo introducirlo de nuevo.

Tabla 15.89. Caso de prueba de aceptación HU11_P3 "Eliminar tecnología" (Fuente: Elaboración propia)

Caso de prueba de aceptación "Eliminar tecnología"		
Código: HU11_P3		**Historia de Usuario:** Administrar tecnología
Descripción de la funcionalidad: Prueba de funcionalidad para eliminar uno o varias tecnologías.		
Condiciones de ejecución: Un administrador tiene que estar autenticado.		
Flujo central: Ir al panel "Administración" > "Tecnologías" > "Eliminar". Seleccionar uno o varias tecnologías de la lista que se muestra. Ejecutar la acción "Eliminar". Aceptar la alerta de eliminación.		
Clases válidas	**Clases inválidas**	**Resultado esperado**
Selección de uno o varias tecnologías.		Se muestra una alerta de eliminación. Si se acepta, se elimina la selección y se muestra la lista actualizada de tecnologías. Si no se acepta, la lista se muestra como

		estaba.
	No se seleccionan tecnologías.	Se muestra un mensaje indicando que se debe seleccionar una tecnología y se vuelve a mostrar la lista.

Anexo 16. Resultados de las pruebas

Resultados de pruebas unitarias

En la siguiente tabla se cuantifican las no conformidades detectadas en cada una de las iteraciones de pruebas (Ip_n), donde n es el número de la iteración.

Tabla 16.90. Cantidad de no conformidades detectadas por cada iteración de pruebas unitarias (Fuente: Elaboración propia)

Historias de Usuario		Herramientas para pruebas de software							
		Selenium IDE		Web Developer		LinkChecker		SQL Inject Me	
		Ip_1	Ip_2	Ip_1	Ip_2	Ip_1	Ip_2	Ip_1	Ip_2
Iteración de desarrollo 1	Administrar departamento	3	1	2	0	0	0	0	0
	Administrar tipo de prueba	3	1	1	0	0	0	0	0
	Administrar herramienta	5	2	4	1	1	0	1	0
	Administrar proyecto	1	0	2	0	0	0	0	0
	Administrar lenguaje	2	1	2	0	0	0	0	0
Iteración de desarrollo 2	Autenticar usuario	3	1	0	0	0	0	0	0
	Administrar usuario	3	1	2	0	0	0	1	0
	Buscar herramienta	4	1	2	0	0	0	0	0

	Proponer herramienta	2	1	2	1	1	0	0	0
	Administrar mensajes de contacto	4	2	0	0	0	0	0	0
	Administrar tecnología	2	0	0	0	0	0	0	0

En la siguiente imagen se puede apreciar un ejemplo de una prueba realizada para verificar la alineación de encabezados, campos de formularios, bloques y otros elementos HTML:

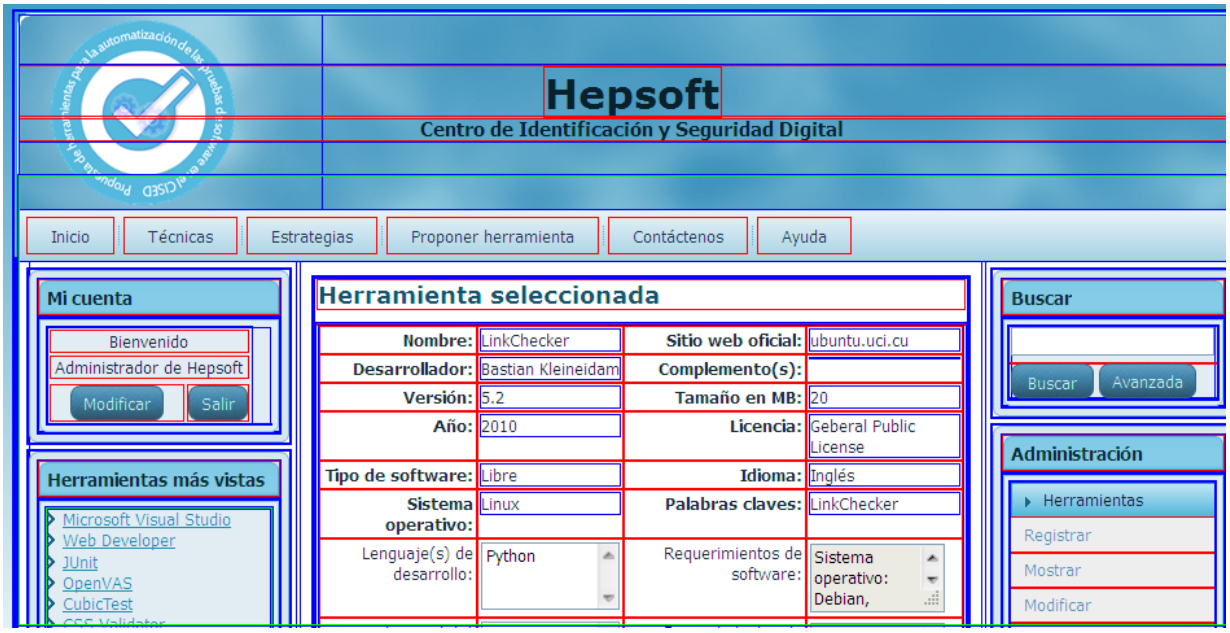

Figura 16.8. Prueba realizada con Web Developer (Fuente: Elaboración propia)

Resultados de pruebas de aceptación

En la siguiente tabla se cuantifican las no conformidades detectadas en cada una de las iteraciones de pruebas (Ip_n), donde n es el número de la iteración.

Tabla 16.91. Cantidad de no conformidades detectadas por cada caso de prueba de aceptación (Fuente: Elaboración propia)

Casos de prueba de aceptación	Ip_1	Ip_2	Casos de prueba de aceptación	Ip_1	Ip_2
Autenticar usuario	3	1	Modificar tipo de prueba	1	0
Salir del sistema	1	0	Eliminar tipo de prueba	1	0
Registrar usuario	1	0	Registrar lenguaje	1	0
Modificar usuario	2	0	Modificar lenguaje	0	0
Eliminar usuario	1	0	Eliminar lenguaje	1	0
Registrar herramienta	4	1	Búsqueda general	3	1
Modificar herramienta	4	1	Búsqueda avanzada	2	0
Eliminar herramienta	1	0	Proponer herramienta	3	0
Registrar proyecto	1	0	Enviar mensaje	1	0
Modificar proyecto	0	0	Mostrar mensaje	2	0
Eliminar proyecto	1	0	Eliminar mensaje	1	0
Registrar departamento	1	0	Registrar tecnología	1	0
Modificar departamento	0	0	Modificar tecnología	0	0

Eliminar departamento	1	0	Eliminar tecnología	1	0
Registrar tipo de prueba	1	0			

Anexo 17. Etapas de una auditoría dentro del CISED

A continuación son detalladas las etapas por las que una auditoría es ejecutada dentro del entorno del Centro:

Etapa de coordinación

En esta etapa se solicita la auditoría o revisión por parte de Director del CISED al Administrador de Calidad del proyecto. En el caso de que fuese solicitada por el Director del Centro o un jefe de departamento, la solicitud será dirigida al Asesor de Calidad, y en la otra variante, el asegurador sería la persona que decidirá realizar la auditoría o revisión.

Una vez aprobada la solicitud, se procede a designar un grupo de auditores que ocuparán los roles de: Auditor Líder, Auditor, Guía y Observador. Posteriormente se procede a enviar a todos los involucrados, tanto auditores como auditados, una notificación de realización de la misma con una semana previa al comienzo.

Etapa de determinación de viabilidad de la auditoría o revisión

Durante esta etapa se revisa si existen evidencias suficientes para realizar la auditoría o revisión.

Etapa de caracterización de la auditoría o revisión

Son determinados los objetivos y el alcance, donde son descritos la extensión y los límites de la auditoría, tales como ubicación, unidades de la organización, actividades y procesos que van a ser auditados, así como el período de tiempo cubierto por la auditoría. Además, son definidos los criterios de la auditoría o revisión como una referencia frente a la cual se determina la conformidad y pueden incluir políticas, procedimientos, lineamientos, reglamentos y requisitos del sistema de gestión.

Etapa de planificación de la auditoría o revisión

En esta etapa son definidos los métodos y las técnicas a utilizar para recopilar información. Se asignan las tareas al equipo auditor y se actualiza el Plan de Aseguramiento de la Calidad que recoge el Plan de Revisiones y Auditorías. Esto último se efectúa por el equipo interno del proyecto si es quien está a cargo de actualizar estos planes, de lo contrario es realizado por el Administrador de la Calidad.

Etapa de desarrollo de la auditoría o revisión

Se realiza una reunión de apertura, se recopila y verifica información. De acuerdo con la revisión realizada, se identifican conformidades y no conformidades, siendo estas últimas registradas en el documento de No Conformidades que comprende el registro de defectos, elementos probados y no probados.

Etapa de evaluación de la auditoría o revisión

Se evalúa el cumplimiento del Plan de Revisiones y Auditorías. Se concluye esta etapa con la evaluación del desempeño de los auditores.

Etapa de seguimiento de la auditoría o revisión

Durante esta etapa final se verifica la implementación de las acciones correctivas o de mejora a los defectos encontrados. Esta verificación puede ser realizada como parte de una auditoría o revisión posterior.

Anexo 18. Comparación entre las herramientas propuestas con otras similares

A continuación se relacionan distintas herramientas analizadas con otras similares, evaluando su conformidad con varias características importantes.

Leyenda:

C_1: Rendimiento, **C_2:** Tipo de software, **C_3:** Sistema operativo, **C_4:** Curva de aprendizaje, **C_5:** Documentación disponible, **C_6:** Dependencia de otros programas.

Herramienta: incluida en la propuesta de herramientas para pruebas de software.

Herramienta: no incluida en la propuesta de herramientas para pruebas de software.

Tabla 18.92. Comparación entre las herramientas propuestas con otras similares (Fuente: Elaboración propia)

Criterio de selección	Herramientas	Características					
		C_1	C_2	C_3	C_4	C_5	C_6
Pruebas funcionales para web	**CubicTest**	Alto	Libre	Multiplataforma	Baja	Mucha	No
Pruebas funcionales para web	**Selenium IDE**	Alto	Libre	Multiplataforma	Baja	Mucha	Sí
Pruebas funcionales para web	HP QuickTest Professional	Bajo	Propietario	Windows	Baja	Mucha	No
Diferentes pruebas para web	Open Load Tester	Bajo	Libre	Windows	Alta	Mucha	No
Pruebas de caja blanca para web	**Web Developer**	Alto	Libre	Multiplataforma	Baja	Mucha	Sí
Pruebas de caja blanca para web	**CSS Validator**	Alto	Libre	Multiplataforma	Baja	Mucha	Sí
Pruebas de caja blanca para web	**Markup Validator**	Alto	Libre	Multiplataforma	Baja	Mucha	Sí
Pruebas de estructura	**LinkChecker**	Alto	Libre	Multiplataforma	Baja	Poca	No
Pruebas de estructura	LinkChecker (complemento)	Bajo	Libre	Multiplataforma	Baja	Poca	Sí
Generación de datos	**DTM Data Generator**	Alto	Propietario	Windows	Media	Mucha	Sí

Generación de datos	**GenerateData.com**	Alto	Libre	Multiplataforma	Baja	Poca	Sí
Pruebas unitarias y de caja blanca en Java	**JUnit**	Alto	Libre	Multiplataforma	Media	Mucha	Sí
Pruebas unitarias y de caja blanca en Java	**PMD**	Alto	Libre	Multiplataforma	Media	Mucha	Sí
Pruebas unitarias y de caja blanca en Java	Checkstyle	Alto	Libre	Multiplataforma	Media	Poca	Sí
Pruebas unitarias y de caja blanca en Java	Cobertura	Alto	Libre	Multiplataforma	Media	Poca	Sí
Pruebas unitarias en Java	Sonar	Alto	Libre	Multiplataforma	Alta	Poca	Sí
Pruebas unitarias en Java	JTest	Alto	Propietario	Windows	Alta	Poca	No
Diferentes pruebas para .NET	**Microsoft Visual Studio 2010**	Medio	Propietario	Windows	Alta	Mucha	No
Diferentes pruebas para .NET	**Microsoft Visual Studio 2008**	Medio	Propietario	Windows	Alta	Mucha	No
Diferentes pruebas para .NET	**Microsoft Visual Studio 2005**	Medio	Propietario	Windows	Alta	Mucha	No
Pruebas unitarias en .NET	**NUnit**	Alto	Libre	Windows	Alta	Poca	Sí
Pruebas de	**Nmap**	Medio	Libre	Multiplataforma	Baja	Poca	No

seguridad							
Pruebas de seguridad para web	**Web Application Attack and Audit Framework**	Bajo	Libre	Multiplataforma	Baja	Poca	No
Pruebas de seguridad	**OpenVAS**	Medio	Libre	Multiplataforma	Media	Poca	No
Pruebas de seguridad para web	**SQL Inject Me**	Alto	Libre	Multiplataforma	Baja	Mucha	Sí
Pruebas de seguridad para web	SQL Injection	Alto	Libre	Multiplataforma	Alta	Poca	No
Pruebas de seguridad para web	Powerfuzzer	Alto	Libre	Multiplataforma	Media	Poca	No
Pruebas de seguridad para web	Nessus	Alto	Propietario	Multiplataforma	Media	Mucha	Sí
Pruebas de seguridad para web	Web Inject	Medio	Libre	Multiplataforma	Media	Poca	No
Pruebas de seguridad para web	HP WebInspect	Medio	Propietario	Windows	Alta	Mucha	No
Pruebas unitarias para PHP	**SimpleTest**	Alto	Libre	Multiplataforma	Media	Poca	Sí
Pruebas unitarias para PHP	**PHPUnit**	Alto	Libre	Multiplataforma	Media	Mucha	Sí
Pruebas unitarias para PHP	**Lime**	Alto	Libre	Multiplataforma	Baja	Mucha	Sí
Pruebas unitarias en	Rational Robot	Medio	Propietario	Windows	Alta	Mucha	Sí

varios lenguajes							
Pruebas de rendimiento y estrés	**JMeter**	Medio	Libre	Multiplataforma	Alta	Mucha	Sí
Pruebas de rendimiento en Java	**JProbe**	Alto	Libre	Windows	Alta	Mucha	No
Pruebas de rendimiento y estrés	JCrawler	Bajo	Libre	Multiplataforma	Baja	Poca	Sí
Pruebas de rendimiento	Spotlight	Medio	Propietario	Windows	Alta	Mucha	No
Pruebas de rendimiento	HP LoadRunner	Medio	Propietario	Windows	Alta	Mucha	No
Pruebas de rendimiento	HP Performance Center	Medio	Propietario	Windows	Alta	Mucha	No
Pruebas de carga y estrés	Testing Master	Alto	Propietario	Windows	Baja	Poca	No
Pruebas de carga y estrés	The Grinder	Alto	Libre	Multiplataforma	Alta	Poca	No
Pruebas de carga y estrés	Benchmark Factory	Alto	Propietario	Windows	Alta	Mucha	No