Universidad de las Ciencias Informáticas

Facultad 2

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

Herramienta para la generación automática de diseños de casos de pruebas para las aplicaciones desarrolladas con XILEMA-Base-Web

Autor(es): Osmel Roján Herrera
Yenisleidi Serrano Gutiérrez
Tutor(es): Ing.Yaislenis Landabe Barbarú
Ing. Katia Ramírez Bruzón

La Habana, abril de 2015
"Año 57 de la Revolución"

Declaración de Autoría

Declaramos que Osmel Roján Herrera y Yenisleidi Serrano Gutiérrez somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad 2 para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de junio del 2015.

_____	_____
Firma del Autor	Firma del Autor
Osmel Roján Herrera	Yenisleidi Serrano Gutiérrez
_____	_____
Firma del Tutor	Firma del Tutor
Ing.Katia Ramírez Bruzón	Yaislenis Lambade Barbarú


"En la tierra hace falta
personas que trabajen
más y critiquen menos,
que construyan más y
destruyan menos, que
prometan menos y
resuelvan más, que
esperen recibir menos y
dar más, que digan mejor
ahora que mañana."
Che

Agradecimientos

De Yenisleidi

A mi mamá y mi papá por todo el sacrificio, amor, entrega, dedicación y confianza que siempre me han brindado y por esperar siempre lo mejor de mí y aguantarme durante tanto tiempo.

A toda mi familia en general en especial a mí prima Yanet y Lilianne por su apoyo incondicional en todo momento, por corregirme y enseñarme a ser una mejor persona cada día.

A mi hermana y sobrino por compartir momentos de alegría y de tristeza.

A mis abuelos por preocuparse por mí y ayudarme en todo momento. A mis compañeros de aula, en especial a Ismaray, a Yanet, a Arian, Dania, Zoya, Yanidza, Yiray a Leobanis y a Tony, por compartir junto a ustedes muchas horas de estudio y muchos momentos felices.

A mis tutoras porque sin ella esta tesis no hubiese sido posible.

A mi compañero de tesis Osmel, sin el cual no hubiera podido llegar a la meta y por soportarme durante todo este tiempo.

De Osmel

A mi familia donde me forme, crecí y son todo para mí:

A mi mamá por darme la vida y después dedicarme la de ella, pues siempre has estado pendiente de mí, apoyándome y comprendiendo todas mis decisiones, y por ser la persona más importante para mí en este mundo.

A mi papá por tanto sacrificio y dedicación para que yo pudiera estar aquí.

A mi hermana por tanto apoyo, ayuda y sacrificio incondicional, y por ser mí segunda mamá.

A mi hermano por tanta confianza, apoyo, consejos y ayuda durante todo este tiempo.

Y a mis sobrinitos pues siempre los tengo presentes.

A todas y cada una de las personas que de una forma u otra me brindaron su ayuda durante mi estancia en esta universidad, mis más sinceros agradecimientos.

Especialmente a todas aquellas personas que desde la Facultad de Artemisa vinieron conmigo y me sirvieron de ayuda y apoyo en cierto momento y durante todo este tiempo, especialmente a Gisselle.

A los amigos y amigas que encontré aquí en la universidad, especialmente a Lisetvis.

A mis tutoras por su dedicación, consejos, ayuda y confianza, sin ustedes hubiera sido imposible este trabajo.

A mi compañera de tesis por haber compartido conmigo la realización de este trabajo y todo este tiempo que hemos estado en la Universidad.

Y a Dios por que sin él no hubiera sido posible este momento de tanta emoción.

Dedicatoria

De Yenisleidi

A mi mamá y papá Lidia y Reinaldo que me inspiraron y me dieron fuerzas para seguir adelante y por ser las personas más especiales de mi vida y por esperar siempre lo mejor de mí. A mis abuelos, mi hermana Misleidis y mi sobrino Robertico por su apoyo incondicional y por darme ánimo en todo momento.

De Osmel

A mi familia, especialmente a mis mamá por ser mi inspiración y por quien hoy me he convertido finalmente en un profesional, por haberme educado de la manera en que lo hizo, por todo el amor que me da y por toda la confianza que siempre depositó en mí.

Resumen

El presente Trabajo de Diploma tiene como objetivo desarrollar una Herramienta para la Generación Automática de Diseños de Casos de Pruebas para las aplicaciones desarrolladas con Xilema-Base-Web.

Para ello se realizó un estudio de los sistemas similares existentes, se seleccionó como metodología de desarrollo XP, como lenguaje de programación Python 3.4.3 y framework de desarrollo Django 1.7.7; el IDE de desarrollo es PyCharm en su versión 3.0.1 y como gestor de base de datos PostgreSQL 9.4. Se generaron los artefactos correspondientes a la metodología seleccionada como: historias de usuarios, plan de iteraciones, plan de entrega, tareas de ingeniería y las tarjetas CRC.

Se obtuvo como resultado una herramienta que además de permitir la generación automática de diseños de casos de prueba, facilita el trabajo a los analistas de los proyectos que desarrollan aplicaciones con Xilema-Base-Web. Mediante las pruebas se validó el correcto funcionamiento del sistema.

Palabras Claves: condiciones de ejecución, diseños de casos de prueba, escenarios, proyecto, repuesta del sistema.

Índice de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

En el mundo actual se percibe un incremento acelerado del uso de las tecnologías, que se refleja en el incremento de la automatización de cada uno de los procesos que se ejecutan en las principales industrias de la sociedad. Esto trae consigo la existencia de empresas y compañías, queriendo incursionar en la producción de software. Todas estas innovaciones tecnológicas conllevan a una exigente evaluación de la calidad de los productos (1).

La calidad de un software se refiere a un producto que cumpla con una serie de atributos tales como: seguridad, manejabilidad, fácil comprensión, modularidad, adaptabilidad, portabilidad, robustez, entre otros.

No siempre se logra que el resultado final del producto cumpla los requisitos establecidos. Por tal motivo se le realizan pruebas al software durante el ciclo de desarrollo del mismo, lo que constituye un proceso aplicado al producto para corregir los errores que presenta.

El instrumento adecuado para determinar el estatus de la calidad de un producto software es el proceso de pruebas (2). En dicho proceso antes mencionado se distinguen las pruebas técnicas y las pruebas funcionales. Las pruebas técnicas son la responsabilidad de los ingenieros de software que han desarrollado el producto, y el responsable para las pruebas funcionales es el técnico de pruebas que dispone de los conocimientos y aptitudes necesarios para esta tarea tan importante y específica.

En proyectos a gran escala las pruebas funcionales son la responsabilidad de un equipo de pruebas, que consiste de uno o varios técnicos, un coordinador de pruebas y un gestor de pruebas o de calidad. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado de cumplimiento según los requerimientos. Para ello el diseño de las pruebas se basa en la creación de casos de prueba cuya ejecución permitirá observar posibles problemas existentes en la aplicación.

Como parte de los esfuerzos de Cuba por ascender a planos superiores en la rama de la informática, han surgido diferentes entidades dedicadas al desarrollo de software. Las mismas deben garantizar la entrega de sus productos con la mayor calidad posible y para ello es sumamente importante no olvidar detalles ni etapas en el desarrollo de una aplicación. Una de estas etapas es la de pruebas, fundamental para aumentar la calidad de los productos y con ellos las posibilidades de insertarse entre las naciones reconocidas como exportadoras de software.

En Cuba se encuentra la Universidad de las Ciencias Informáticas (UCI), un centro de altos estudios que tiene como objetivos informatizar el país y desarrollar la industria de software para contribuir al desarrollo económico del mismo. La UCI se ha dedicado a la producción y comercialización de software, insertándose poco a poco en esta compleja industria mediante convenios de cooperación con varias naciones, proyectándose a otros mercados foráneos (3).

La Universidad de las Ciencias Informáticas posee un modelo flexible de centros docente-productor que le permite formar profesionales altamente calificados, así como producir aplicaciones y servicios informáticos. En la misma se encuentra el centro de Telemática (TLM) de la Facultad 2 destinado al desarrollo de sistemas y servicios informáticos en la rama de las telecomunicaciones y seguridad informática. Dicho centro posee un grupo de Calidad el cual se encarga de garantizar la calidad del producto final que es entregado a los clientes, además se llevan a cabo pruebas funcionales a las diversas aplicaciones que se desarrollan en la entidad, con el objetivo de validar la calidad y funcionalidad de las mismas. Para realizar las pruebas se realiza previamente en un documento con formato xls, un diseño de casos de pruebas que sirve como guía al personal que está probando la aplicación, lo que hace que se dedique un tiempo considerable para realizar esta actividad. Generalmente cuando este artefacto es entregado al grupo de calidad para su revisión presenta varias incongruencias como: ausencia e incorrecta descripción de las variables y ausencia de algunos escenarios. La definición de este artefacto es una tarea del analista del proyecto, el cual es una de las personas que más conoce el negocio del mismo. En muchas ocasiones el sistema es tan grande que demanda de mucho tiempo para que el analista diseñe todos los casos de prueba de la aplicación, conllevando además a la inserción de valores incorrectos que al final se transforman en no conformidades para la aplicación.

Por todo lo antes expuesto se plantea como **problema a resolver**: ¿Cómo generar diseños de casos de prueba de forma automática para las aplicaciones web desarrolladas con Xilema-Base-Web?

A partir del problema científico se define como **objeto de estudio** para esta investigación: el proceso de diseño de casos de prueba, y como **campo de acción:** proceso de diseño de casos de prueba para las aplicaciones desarrolladas con Xilema-Base-Web.

Se propone como **objetivo general**: Desarrollar una herramienta que permita generar diseños de casos de prueba de manera automática a las aplicaciones web desarrolladas con Xilema-Base-Web.

Para cumplir con lo antes descrito en esta investigación se han trazado las siguientes **tareas investigativas:**

- Realización de un estudio de las tendencias actuales sobre sistemas similares existentes.
- Análisis de las metodologías de desarrollo, herramientas y tecnologías posibles a utilizar en el desarrollo de la solución.
- Levantamiento de requisitos.
- Diseño e implementación del sistema.
- Realización de pruebas de validación de la aplicación.

Como **posible resultado** se espera obtener una herramienta para la generación automática de diseños de casos de prueba a las aplicaciones desarrolladas con Xilema-Base-Web.

Una vez concluida la investigación y resuelto dicho problema se espera mejorar la rapidez, seguridad y confiabilidad en la generación de diseños de casos de prueba a las aplicaciones desarrolladas con Xilema-Base-Web.

Para realizar las tareas de la investigación se emplearon los siguientes métodos científicos:

Teóricos:

- **Analítico - Sintético:** La utilización de este método ha servido para analizar y comprender la teoría y documentación relacionada con el tema de investigación, permitiendo así extraer las ideas fundamentales y al mismo tiempo detallar la información necesaria para el modelado correcto del negocio.
- **Modelación:** Se utilizó para el desarrollo del modelo de la base de datos del sistema, mediante el cual se pudo crear abstracciones con el propósito de explicar la realidad.

Empíricos:

- **Medición:** Este método se tuvo en cuenta para las pruebas que se le realizan al software.

El presente trabajo de diploma, está estructurado en 3 capítulos, a continuación se muestra una breve descripción de cada uno de ellos:

Capítulo 1: "Fundamentación teórica", se incluyen los aspectos teóricos que dan sustento a la investigación. Se realiza además un análisis detallado de la metodología de desarrollo, herramientas y tecnologías a utilizar en el proceso de desarrollo del sistema, para llegar a una solución factible que cumpla con el objetivo propuesto en este trabajo de diploma.

Capítulo 2: "Propuesta de solución", se incluyen las principales características y funcionalidades que debe cumplir el sistema a desarrollar, así como el diseño y la estructuración del mismo. Se generan además los artefactos que propone la metodología escogida.

Capítulo 3: "Implementación y Prueba", se describen los patrones que se utilizarán, también se implementan las funcionalidades del sistema a través de las tareas de la ingeniería describiéndose cada una de ellas. Además se detallan las pruebas realizadas al software con el objetivo de asegurar la calidad y eficiencia de la aplicación realizada.

Capítulo 1: Fundamentación Teórica

Introducción

En el presente capítulo se abordan conceptos y definiciones relacionadas con la calidad de software, realizando un estudio de contenidos teóricos que sustentan la investigación, la problemática y las herramientas y tecnologías más adecuadas para brindar un enfoque sobre la solución. Estas herramientas y tecnologías utilizadas han sido seleccionadas siguiendo la estrategia de desarrollo usada en el centro, la cual va dirigida a la investigación; proporcionando información al lector sobre los principales resultados del estudio realizado sobre las tendencias actuales, y la justificación del conjunto de herramientas y metodología de desarrollo de software, que permiten darle solución al problema en cuestión.

1.1 Conceptos Fundamentales

1.1.1 Calidad

La Norma Standard ISO E 9000:2000 de la Organización Internacional para la Estandarización define a la calidad como: "La totalidad de rasgos y características de un producto o servicio, que conllevan la aptitud de satisfacer necesidades preestablecidas o implícitas" (4).

1.1.2 Calidad de Software

Pressman define la calidad del software como: “la concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente” (5).

En la definición de la calidad del software pueden estar involucrados aspectos como la ausencia de defectos, aptitud para el uso, seguridad y confiabilidad. Sin embargo, hay algo importante que se debe tener presente: la calidad del software debe ser construida desde el comienzo, no es algo que puede ser añadido después. Para que el producto final sea de calidad, el proceso por el cual éste es elaborado debe tener también calidad (5).

1.1.3 Aseguramiento de la Calidad

Sridharan indica que mientras el software que se está desarrollando reúne los requerimientos, y su desempeño es el esperado, es preciso que se supervisen las actividades de desarrollo del software y su rendimiento en distintas oportunidades durante cada fase del ciclo de vida. Este es el papel del aseguramiento de la calidad del software (6).

- Hay tres aspectos muy importantes con relación aseguramiento de la calidad del software:
  - La calidad no se puede probar, se construye.
  - El aseguramiento de la calidad del software no es una tarea que se realiza en una fase particular del ciclo de vida de desarrollo.
  - Las actividades asociadas con el aseguramiento de la calidad del software deben ser realizadas por personas que no estén directamente involucradas en el esfuerzo de desarrollo.
- Pressman considera que el aseguramiento de la calidad del software comprende una gran variedad de tareas asociadas (7):
  - Preparar un plan de aseguramiento de la calidad del software para un proyecto.
  - Participar en el desarrollo del proceso de descripción del proyecto de software.
  - Revisar las actividades de ingeniería del software para verificar su consistencia con el proceso de software definido.
  - Auditar el producto de software para verificar el cumplimiento del proceso de software definido.
  - Asegurar que las divergencias en el trabajo de software sean documentadas de acuerdo a los estándares definidos.
  - Almacenar cualquier inconformidad y reportarla a la gerencia media.

1.1.4 Pruebas de Software

Las pruebas son un elemento crítico para la calidad del software. La importancia de los costos asociados a los errores, promueve la definición y aplicación de un proceso de pruebas minuciosas y bien planificadas. Estas permiten validar y verificar el software, entendiendo como validación del software el proceso externo al equipo de desarrollo, que determina si el software satisface los requisitos; y verificación

como el proceso interno que determina si los productos de una fase satisfacen las condiciones de la misma (8).

Este sistema, como cualquier otro en ingeniería, puede probarse de dos formas:

- Conociendo la función específica para la que fue diseñado.
- Conociendo el funcionamiento del producto.

El primer enfoque se centra en las llamadas pruebas de caja negra y el segundo en las pruebas de caja blanca (8).

1.1.5 Pruebas de Funcionalidad

Se denominan pruebas funcionales, a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados. Es común que este tipo de pruebas sean desarrolladas por analistas de proyectos con apoyo de algunos usuarios finales. Esta etapa suele ser la última de las pruebas, y al dar conformidad sobre esta, el paso siguiente es el pase a producción (2).

1.1.6 Método de Caja Negra

Para Pressman las pruebas de caja negra se centran en los requisitos funcionales del software. Es decir, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Se trata de un enfoque que intenta descubrir diferentes tipos de errores que no se encuentran con los métodos de caja blanca (9).

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos (9).

1.1.7 Técnicas de diseño de casos de prueba

El diseño de casos de prueba se centra en un conjunto de técnicas que satisfacen los objetivos globales de la prueba. Las técnicas más comunes son:

Análisis de los valores límites

La experiencia muestra que los casos de prueba que exploran las condiciones límites producen mejor resultado que aquellos que no lo hacen. Las condiciones límites son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el

análisis de valores límites como técnica de prueba. Esta técnica lleva a elegir los casos de prueba que ejerciten los valores límites (10).

Particiones de equivalencia

Pressman presenta la partición equivalente dentro del Método de Prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (10).

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (sí o no) (10).

1.1.8 Diseño de Casos de Prueba

El diseño de casos de prueba es una parte de las pruebas de componentes y sistemas en la que se diseñan los casos de pruebas (entradas y salidas esperadas) para probar el sistema. El objetivo del proceso de diseño de casos de prueba es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos. Para diseñar un caso de prueba se selecciona una característica del sistema o del componente que se está probando (11).

1.1.8.1 Diseño de Casos de Prueba basados en casos de usos

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales. Sin embargo, no son solo una herramienta para especificar los requisitos de un sistema, también guían su diseño, implementación, y prueba; guían el proceso de desarrollo (11). Los casos de prueba basados en casos de uso reflejan la funcionalidad de los mismos, ya que muestran una secuencia ordenada de eventos, al describir flujos básicos, flujos alternos, precondiciones y post-condiciones, estos se expresan a través de los diferentes escenarios que posee este artefacto (12).

Se diseña un caso de prueba para cada caso de uso. Los escenarios de un caso de uso se identifican describiendo los distintos caminos del flujo básico y flujo alternativo del caso de uso.

1.1.8.2 Diseño de Casos de Prueba basados en requisitos

Para demostrar que un sistema está construido según las especificaciones de requisitos, se utilizan pruebas basadas en requisitos. Por cada uno, se debe diseñar uno o más casos de prueba. Al final de la ejecución de la prueba, se deben generar informes sobre las pruebas que se realizaron y los requisitos que se cubrieron. Basándose en esta información el cliente y las diversas partes interesadas pueden decidir si un sistema se encuentra listo para ser transferido a la fase de prueba siguiente o no (13).

1.2 Metodologías de desarrollo de software

Las Metodologías de Desarrollo de Software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo sistema, pero los requisitos de un software son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación de aplicaciones.

Se podrían clasificar en dos grandes grupos:

- Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas Metodologías Pesadas, dentro de este grupo se encuentra RUP (Rational Unified Process) que es una de las metodologías pesadas más conocidas y utilizadas.
- Las metodologías orientadas a la interactuación con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas Metodologías ligeras/ágiles, dentro de este grupo se encuentra la metodología XP (Extreme Programming) (14).

1.2.1 Programación Extrema (XP)

La Programación Extrema o Extreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck y De Jean. Es la más destacada de los procesos ágiles de desarrollo de software.

Las características fundamentales de esta metodología son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.

- Pruebas unitarias continuas, frecuentemente, repetidas y automatizadas, incluyendo pruebas de regresión.
- Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- Frecuente integración del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo (15).

Por las características antes expuestas y teniendo en cuenta además que sirve para el modelado completo del sistema, la metodología de desarrollo de software seleccionada, para guiar el proceso de desarrollo del presente trabajo es XP.

1.3 Herramienta CASE

La realización de un nuevo software requiere que los procesos sean organizados y completados en forma correcta y eficiente. Las herramientas CASE (Computer Aided Software Engineering / Ingeniería de Software Asistida por Ordenador) fueron desarrolladas para automatizar esos procesos y aumentar la productividad en el desarrollo de software, reduciendo el coste de los mismos en términos de tiempo y de dinero.

Visual Paradigm para UML es una herramienta Case profesional que soporta el ciclo de vida completo del desarrollo de software, está dotada de una buena cantidad de módulos para facilitar el trabajo durante la confección de un software. Posee, entre otras, las siguientes características que garantizan la calidad del producto final:

- Es profesional: permite crear un conjunto bastante amplio de artefactos utilizados con mucha frecuencia durante la confección de un software. Todos estos, cumpliendo con el Standard UML 2.0.
- Brinda un número considerable de estereotipos a utilizar, lo que permite un mayor entendimiento de los diagramas.
- Generación de documentación: permite documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Se decidió utilizar como herramienta CASE el Visual Paradigm por las características antes mencionadas. Otra gran ventaja que posee Visual Paradigm es que tiene soporte para generar código en el lenguaje de programación Python y base de datos en postgres y mysql (16).

1.4 Herramienta para el almacenamiento de datos.

Los Sistemas de Gestión de Base de Datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Postgresql está considerado como el sistema de gestión de bases de datos de código abierto más avanzado del mundo, debido a que proporciona un gran número de funcionalidades que normalmente solo se encontraban en sistemas de bases de datos comerciales tales como Oracle.

Algunas de sus principales características son:

- Aproxima los datos a un modelo objeto-relacional y es capaz de manejar complejas rutinas y reglas.
- Contiene consultas SQL declarativas, control de concurrencia multi-versión, soporte multiusuario, optimización de consultas, herencia y arreglos.
- Es un SGBD altamente extensible ya que soporta operadores funcionales, métodos de acceso y tipos de datos definidos por el usuario.
- Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos.
- Basa su funcionamiento en una arquitectura proceso-por-usuario cliente/servidor.

PostgreSQL fue seleccionado como sistema de gestión de base de datos para el desarrollo de esta herramienta, por los buenos resultados que en la práctica se han obtenido al desarrollar aplicaciones web con Python. Además soporta distintos tipos de datos, permite la declaración de funciones propias, así como la definición de disparadores, la gestión de diferentes usuarios como también los permisos asignados a cada uno de ellos (17). También se utiliza ya que es la herramieta de almacenamiento de datos utilizada en el centro de Telemática.

1.5 Lenguaje de programación.

Se decide utilizar Python ya que es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (18).También se utiliza Python ya que Xilema-Base-Web esta desarrollada con este lenguaje.

1.6 Herramienta de Desarrollo (IDE).

Como herramienta de desarrollo se utiliza PyCharm pues es un entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Además proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django.

1.7 Framework Web.

Se utiliza Django pues es un framework web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código. Además este es el framework de desarrollo del centro de Telemática.

1.8 Xilema-Base-Web

Xilema-Base-Web es un marco de trabajo desarrollado en el Centro de TLM que está constituido por Django como framework base, librerías de javascript como son Jguery y Backbone y que cuenta con las pautas de diseño de la Universidad (19).

Conclusiones

Después del estudio y análisis realizado del objeto de investigación, apoyado en los métodos científicos definidos, se pudo construir el marco teórico-conceptual que soporta dicho proceso. El estudio del marco teórico aportó elementos sólidos para la realización de la herramienta para la generación automática de los diseños de casos de prueba a las aplicaciones desarrolladas con Xilema-Base-Web. Se adquirieron los conocimientos necesarios sobre las herramientas, metodología y lenguajes utilizados en el desarrollo de soluciones informáticas, seleccionándose los más adecuados para el cumplimiento del objetivo propuesto.

En el estudio realizado no se encontró ninguna herramienta que genere diseños de casos de prueba automáticamente por lo que se decide diseñar una herramienta que permita automatizar dicho proceso.

Como metodología de desarrollo de software se escoge XP y como herramienta CASE el Visual Paradigm en su versión 8.0. El lenguaje de programación seleccionado fue Python 3.4.3, el framework web de desarrollo utilizado Django en su versión 1.7.7, el IDE de desarrollo es PyCharm en su versión 3.0.1 y como gestor de base de datos PostgreSQL 9.4 para el desarrollo de la aplicación.

CAPÍTULO 2: Propuesta de Solución

Introducción

En este capítulo se describen las fases principales de la metodología XP: Planificación y Diseño de la solución propuesta. Se desarrollan los artefactos correspondientes a estas fases como las Historias de Usuarios, Plan de Iteraciones, Plan de Duración de Iteraciones y el Plan de Entrega.

2.1 Usuarios relacionados con el sistema

La aplicación que se desarrolla muestra un grupo de funcionalidades que permiten cumplir con el objetivo trazado. Se denomina usuario a cualquier persona relacionada con el sistema, ya sea vinculada al desarrollo del mismo o que de una forma u otra interactúa con la aplicación, incluyendo a los que mantendrán el sistema funcionando y actualizado (20).

En la implementación se establecen dos roles uno para el analista principal del proyecto, el cual tendrá acceso a todas las funcionalidades del sistema y el otro para el analista del proyecto el cual tendrá acceso a las funcionalidades del sistema según los permisos que el analista principal le otorgue.

Tabla 1. Usuarios relacionados con el sistema

Usuario	Descripción
Analista Principal	Es el que interactúa directamente con el sistema y administra todas sus funcionalidades.
Analista	Tendrá acceso a las funcionalidades del sistema según los permisos que el analista principal le otorgue.

2.2 Lista de requisitos.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. De forma general estos requisitos indican lo que debe hacer el sistema. Los requerimientos funcionales que se identificaron para el desarrollo de la aplicación fueron los siguientes:

RF1-Gestionar Usurario.

RF1.1 - Adicionar Usuario.

RF1.2 - Modificar Usuario.

RF1.3 - Eliminar Usuario.

RF1.4 - Autenticar Usuario.

RF2 - Gestionar Proyecto.

RF2.1 - Adicionar Proyecto.

RF2.2 - Modificar Proyecto.

RF2.3 - Eliminar Proyecto.

RF3 - Gestionar datos de Casos de Prueba.

RF3.1 - Adicionar datos de Caso de Prueba.

RF3.2 - Modificar datos de Caso de Prueba.

RF3.3 - Eliminar datos de Casos de Prueba .

RF4 - Generar Diseño de Casos de Prueba.

2.3 Historias de Usuarios

La metodología XP utiliza la técnica de las Historias de Usuario (HU) para sustituir los documentos de especificación funcional y los casos de uso. Estas HU son escritas por el cliente en su propio lenguaje como descripciones cortas de lo que el sistema debe realizar. El tratamiento de las HU es muy dinámico y flexible, permite que en cualquier momento se puedan romper, reemplazar por otras más específicas o generales, añadirse nuevas o ser modificadas. Para ser implementadas las HU, el cliente y los desarrolladores se reúnen para detallar las funcionalidades de cada una. El tiempo de desarrollo ideal para una HU varía entre 1 y 5 semanas (21).

A continuación se muestran las HU más importantes del sistema.

Tabla 2. HU. Gestionar usuario

Historia de Usuario	
Número: 1	**Nombre de la Historia de Usuario:** Gestionar usuario.
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista Principal	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 0.2
Riesgo en desarrollo: Alto	**Puntos reales: 0.2**
Descripción: El sistema permitirá al analista principal la posibilidad de adicionar, modificar y eliminar un usuario en el sistema.	

Tabla 3. HU. Gestionar proyecto

Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Gestionar proyecto.
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados: 0.4**
Riesgo en desarrollo: Alto	**Puntos reales: 0.4**
Descripción: El sistema permitirá al analista la posibilidad de adicionar, modificar y eliminar un proyecto en el sistema.	

Tabla 4. HU. Gestionar datos de Caso de Prueba

Historia de Usuario	
Número: 3	**Nombre de la Historia de Usuario:** Gestionar datos de Caso de Prueba.
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 4
Riesgo en desarrollo: Alto	**Puntos reales: 4.2**
Descripción: El sistema debe permitir al analista la posibilidad de crear, modificar y eliminar datos de Caso de Prueba.	

Tabla 5. HU. Generar Diseño de Casos de Prueba

Historia de Usuario	
Número: 4	**Nombre de la Historia de Usuario:** Generar Diseño de Casos de Prueba.
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados: 4.2**
Riesgo en desarrollo: Alto	**Puntos reales: 4.5**
Descripción: El sistema debe permitir al analista la opción de generar Diseño de Casos de Pruebas a partir de los datos entrados por el usuario.	

2.4 Estimación de esfuerzos por HU

Las HU deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios (19). Inicialmente, el equipo de desarrolladores estima el esfuerzo necesario para implementar las HU y los clientes aprueban los objetivos y tiempos de entrega, la estimación temporal se basa en un cálculo estimado por parte de los desarrolladores de cada una de las HU (22). Las estimaciones realizadas en esta fase son primarias y podrían variar cuando se analicen más en detalle en cada iteración (21). Para la realización de la herramienta se utilizarán las semanas como medida para la estimación de esfuerzos establecidas por los programadores. A continuación se muestra la tabla Estimación de esfuerzos:

Tabla 6. Estimación de esfuerzo por HU

No	Historia de Usuario	Estimación(por semana)
1	Gestionar Usuario	0.2
2	Gestionar Proyecto	0.4
3	Gestionar datos de Caso de Prueba.	4.2
4	Generar Diseños de Casos de Pruebas.	4.5

2.5 Plan de Iteraciones

Todo proyecto que emplea metodología XP debe dividirse en iteraciones. Las iteraciones son fases o etapas de la implementación donde se obtienen resultados en un tiempo estimado (23). En el plan de iteraciones se especifican detalladamente el orden de desarrollo de las HU dentro de cada iteración, conjuntamente con la duración de las mismas.

Iteración 1: En esta iteración se implementan las HU que tienen prioridad alta en el negocio, las principales son las HU 1 y 2 las cuales son de vital importancia para la aplicación ya que conforman la

base de la estructura del sistema. Con esta iteración se obtiene la primera versión de la aplicación la cual se utilizará para mostrar al cliente y permitirá al grupo de trabajo tener una retroalimentación.

Iteración 2: En esta iteración se realiza la implementación de la HU 3, se corrigen errores o inconformidades del cliente con las HU implementadas en la iteración anterior y se obtiene la segunda versión del sistema.

Iteración 3: En esta iteración se realiza la implementación de la HU 4 corrigiéndose los errores de las iteraciones anteriores.

2.6 Plan de duración de iteraciones

El plan de duración de las iteraciones se realiza luego de tener el estimado en días que demora implementar cada HU (21). El objetivo del plan de duración de iteraciones es especificar detalladamente el orden de desarrollo de las HU dentro de cada iteración y la duración de las iteraciones.

Tabla 7. Plan de duración de iteraciones

Iteración	Descripción de la iteración	Duración total
1	Se tuvieron en cuenta las HU pertenecientes al Gestionar usuario y Proyectos.	1 semanas
2	Se tuvieron en cuenta las HU pertenecientes al Gestionar Datos del Caso de Prueba. También en esta iteración se corregirán los errores encontrados en la iteración anterior obteniéndose una nueva versión del sistema.	6 semanas
3	Se tuvo en cuenta la HU Generar Diseño de Casos de	6 semanas

	Prueba. Además se corrigieron los errores encontrados en la iteración anterior.	

2.7 Plan de entregas

En el plan de entregas se establecen que HU son agrupadas para conformar una entrega y el orden de las mismas.

Tabla 8. Plan de entregas

Historia de Usuario	Primera Iteración	Segunda Iteración	Tercera Iteración
Gestionar Usuario	V1.0	Finalizado	-
Gestionar Proyecto	V1.0	Finalizado	-
Gestionar datos casos de uso y especificación de requisitos de software.	-	V1.0	Finalizado
Generar Diseño de Casos de Prueba.	-	-	Finalizado

2.8 Prototipos de Interfaz de usuarios

Los prototipos de interfaz de usuarios son elementos de diseño visual que permiten al usuario tener una idea de las interfaces que mostrará el sistema para obtener una retroalimentación sobre los requerimientos del mismo (24).

A continuación se muestran algunos prototipos de interfaz de usuarios.

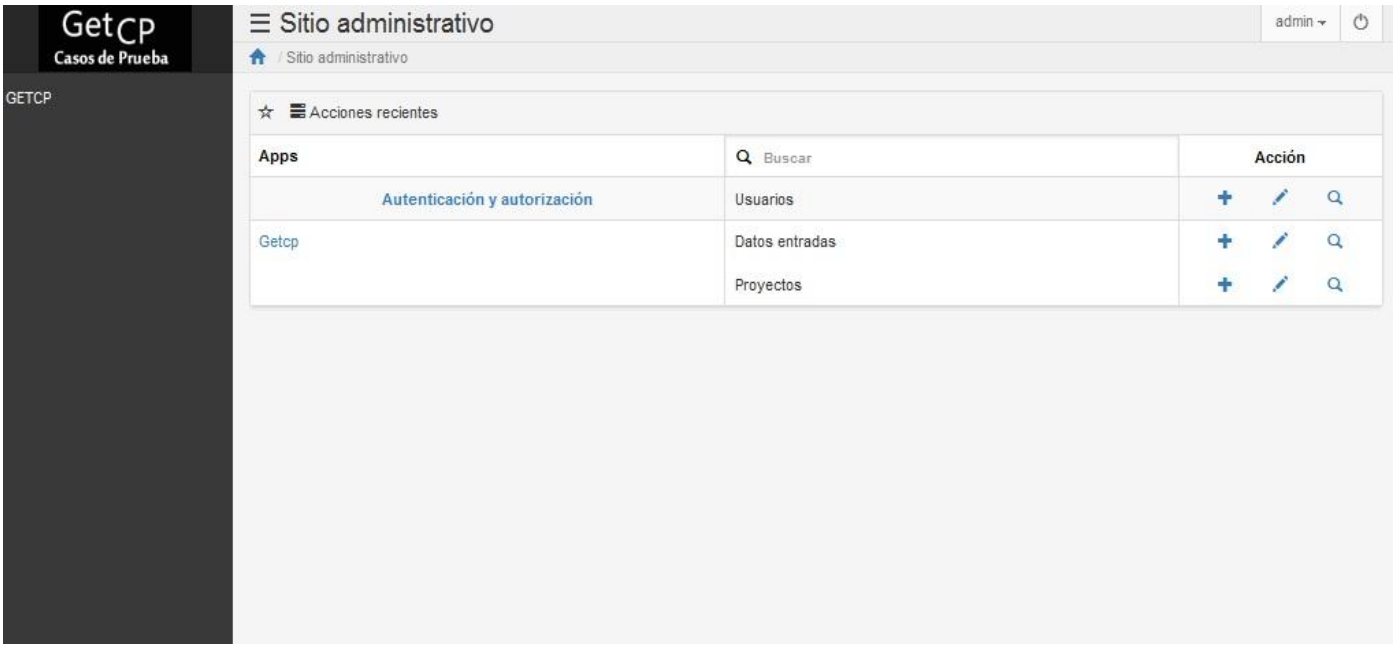

Ilustración 1. Prototipo de Interfaz de Usuario. Sitio Administrativo

Ilustración 2. Prototipo de Interfaz de Usuario. Añadir proyecto

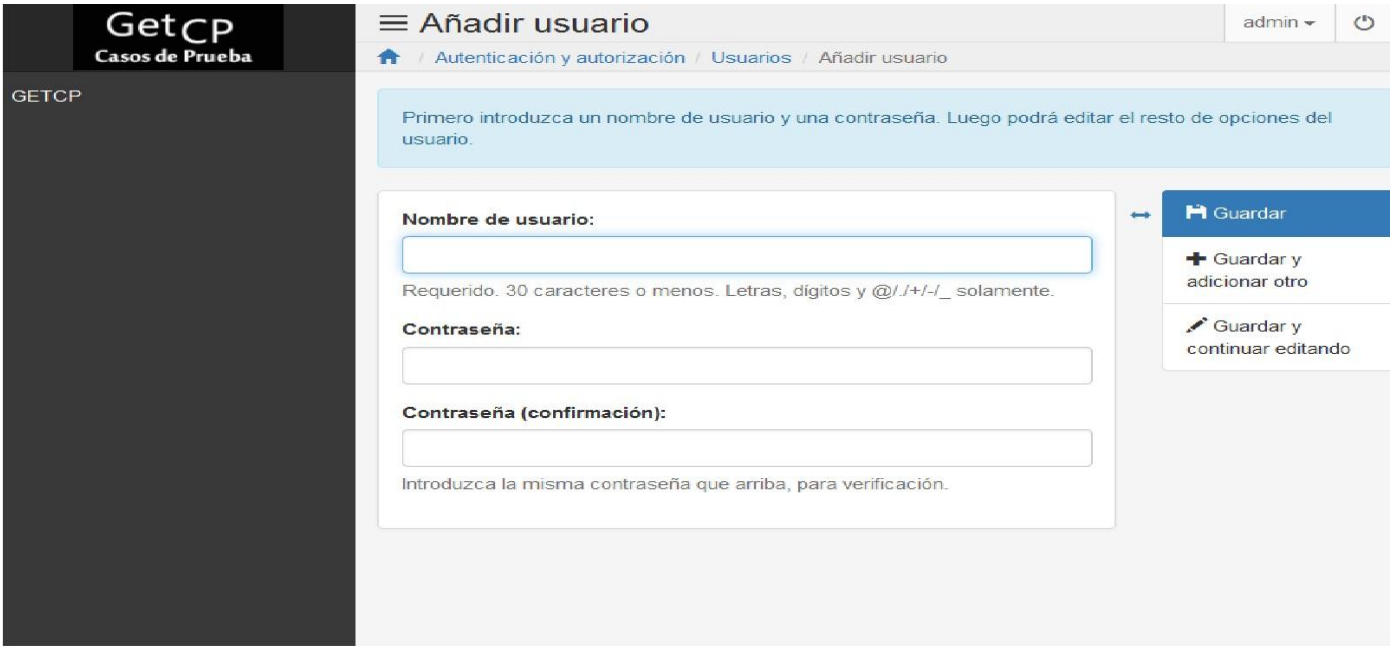

Ilustración 3. Prototipo de Interfaz de Usuario. Añadir usuario

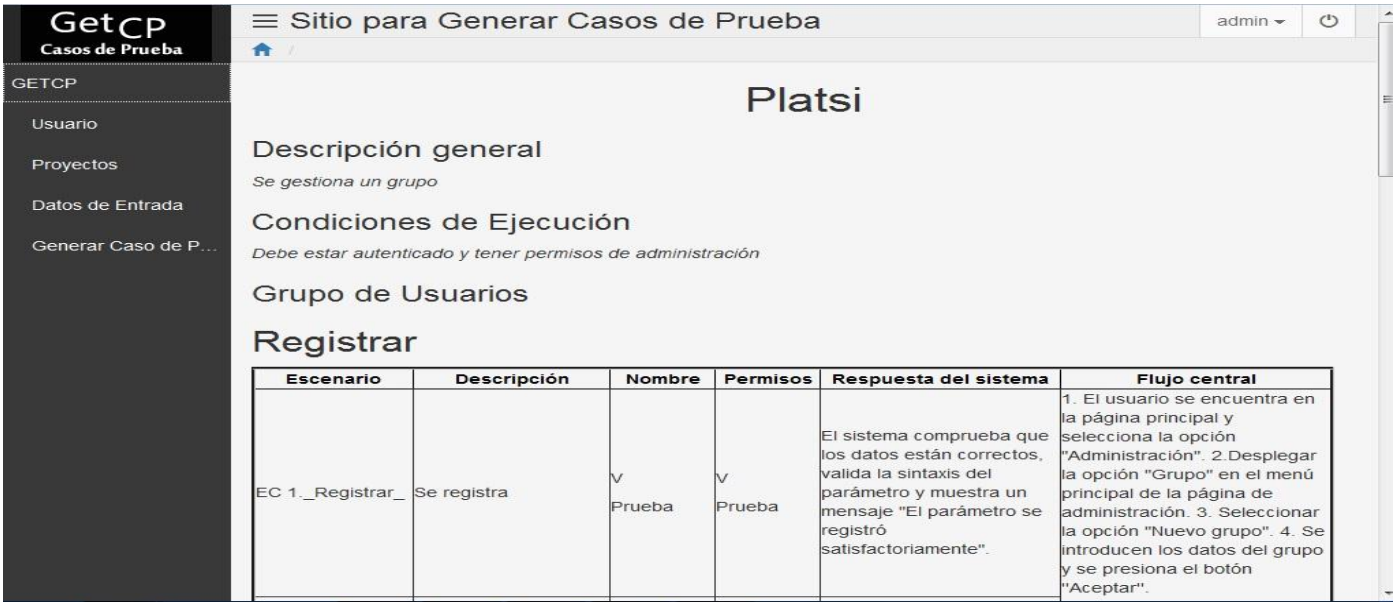

Ilustración 4. Prototipo de Interfaz de Usuario. Generar casos de prueba

2.9 Tareas de Ingeniería.

Las tareas de ingeniería son las distintas funcionalidades operativas que conforman una historia de usuario y que permiten testear si se está trabajando bien. Esto constituye un paso decisivo para comenzar la implementación del software, pues permite organizar el trabajo en secuencias lógicas, de acuerdo a la planificación correspondiente a esa historia de usuario (25).

Las tareas de la ingeniería se realizan con el objetivo de resolver las HU que pueden tener una o más funcionalidades, en dependencia de la complejidad de la HU a desarrollar.

Tabla 9. Tareas de Ingeniería

No. HU	Nombre HU	No TI	Tarea de Ingeniería
1	Gestionar usuario	1	Adicionar Usuario
		2	Modificar Usuario
		3	Eliminar Usuario
		4	Autenticar usuario
2	Gestionar proyecto	5	Adicionar proyecto
		6	Modificar proyecto
		7	Eliminar proyecto
3	Gestionar datos de Caso de Prueba	8	Adicionar datos de Caso de Prueba
		9	Modificar datos de Caso de Prueba
		10	Eliminar datos de Caso de Prueba

4	Generar Diseños de Casos de Pruebas	11	Generar Diseños de Casos de Pruebas

Tabla 10. Tareas de Ingeniería 1 Gestionar Usuario

Tarea de Ingeniería	
Número de la tarea: 1	**Número de Historia de Usuario:** 1(Gestionar Usuario).
Nombre de Tarea: Adicionar Usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha Inicio: 4 de febrero 2015	**Fecha Fin:** 5 de febrero 2015
Programador responsable: Osmel Roján	
Descripción: El analista principal accede a la opción Añadir Usuario. Se muestra una interfaz requiriendo un nombre de usuario y contraseña, el sistema verifica que los datos estén correctos y este le otorga los privilegios al nuevo usuario creado.	

Tabla 11.Tareas de Ingeniería Gestionar Proyecto

Tarea de Ingeniería	
Número de la tarea: 5	**Número de Historia de Usuario:** 2(Gestionar Proyecto).
Nombre de Tarea: Adicionar Proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2

Fecha Inicio: 6 de febrero 2015	**Fecha Fin:** 7 de febrero 2015
Programador responsable: Yenisleidi Serrano	
Descripción: El sistema permite adicionar un proyecto mostrando la opción: Adicionar proyecto, el usuario lo selecciona y muestra una interfaz de formulario requiriendo los campos para crear un nuevo proyecto.	

Conclusiones.

En este capítulo se propuso la solución para la elaboración de la Herramienta automática de diseños de casos de prueba para las aplicaciones desarrolladas con Xilema Base Web, empleando una metodología que permite realizar el análisis y diseño de la aplicación. Se definieron un total de 14 HU que describen los aspectos principales a tener en cuenta para el desarrollo de la solución. Se precisó la prioridad de cada una de las HU puntualizando el orden de su implementación y las iteraciones en que serán implementadas. A partir de las HU se construyó el plan de entregas, las tareas de ingeniería y el plan de iteraciones para una mejor organización en el trabajo de los desarrolladores.

Capítulo 3 Implementación y Pruebas

Introducción

En este capítulo se especifican las fases llevadas a cabo durante la elaboración del sistema, además de mostrar las pruebas realizadas al software, las tarjetas CRC, el patrón arquitectónico empleado, los patrones de diseño utilizados y el modelo de la base de datos correspondiente a la solución.

3.1 Tarjetas CRC (Cargo o Clase, Responsabilidad y Colaboración).

Las tarjetas CRC son en la práctica pequeñas tarjetas de cartón que se elaboran para ser mostradas al cliente, de manera que se pueda llegar a un acuerdo sobre la validez de las abstracciones propuestas, lo que ayuda al equipo durante el diseño e implementación del sistema. Estas constituyen documentación adicional que será adjuntada a las HU(21).

Tabla 12.Tarjeta CRC Gestionar Datos de Caso de Prueba.

Clase: DatosEntrada	
Responsabilidad	**Colaboración**
Gestionar los Datos de Entrada	• Proyecto

Tabla 13.Tarjeta CRC Generar Casos de Prueba.

Clase: GenerarCasosDePrueba	
Responsabilidad	**Colaboración**
Generar Casos de Prueba	• Datos de Entrada • Proyecto

3.2 Patrones de arquitectura

Un patrón es un modelo a seguir y surgen de las experiencias de seres humanos al tratar lograr ciertos objetivos. Capturan la experiencia existente y probada para promover buenas prácticas (26).

Los patrones arquitectónicos definen la estructura del software y son a la vez patrones de alto nivel que fijan la arquitectura global de una aplicación (27).

3.2.1 Patrón Modelo-Vista-Controlador

El patrón Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Este patrón se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio (28).

Django es un framework de desarrollo web de código abierto, escrito en Python, que implementa el patrón de diseño antes mencionado. A continuación se muestra una breve descripción de cada una de las capas de la arquitectura definida.

Modelo: esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos. En la herramienta se ve claramente en la interfaz de entrada de datos y su relación en la base de datos, además de las consultas que se realizan y su validación.

Controlador: esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción dicha capa, bien sea por cambios en la información del Modelo o por alteraciones de la Vista.

Vista: contiene las decisiones relacionadas a la presentación. Maneja de forma visual los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

En la siguiente figura se muestra cómo funciona la arquitectura implementada en el sistema).

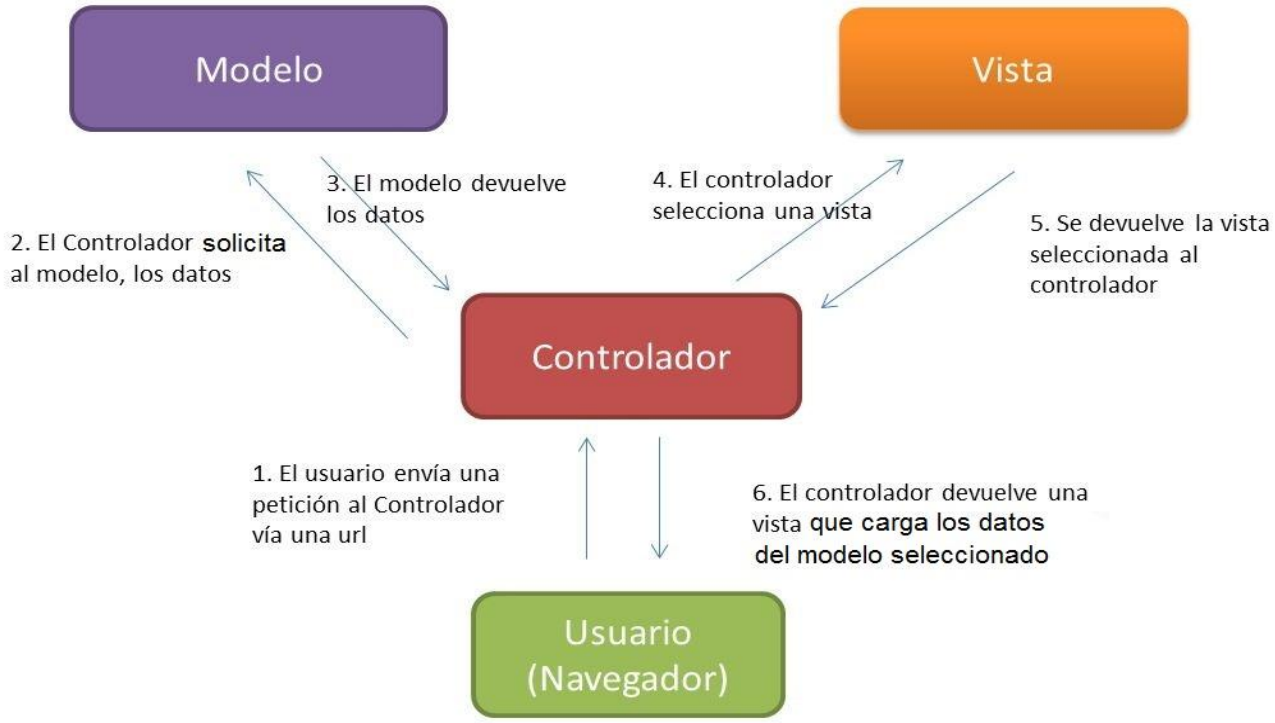

Ilustración 5. Patrón Arquitectónico Modelo-Vista-Controlador.

3.3 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Estos patrones identifican clases, instancias, roles, colaboraciones y la distribución de responsabilidades (26).

Los patrones de diseño se dividen en dos grandes grupos los GRASP (del inglés General Responsibility Asignment Software Patterns) (patrones generales de software para asignar responsabilidades) y los GOF (del inglés Gang of Four). A continuación se realiza un estudio de cómo fueron utilizados en la Herramienta para la generación Automática de Diseños de Casos de Prueba a las aplicaciones desarrolladas con Xilema-Base-Web.

3.3.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (29).

En el diseño del sistema se destaca el uso de 4 patrones principales que son:

Alta Cohesión: en la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Como ejemplo en el sistema está la clase Proyecto que se le asignan responsabilidades con las demás clases del sistema.

Bajo Acoplamiento: el acoplamiento es una medida de la fuerza con que está conectada a otras clases y con que recurre a ellas. Una clase con bajo o débil acoplamiento no depende de muchas otras. Un ejemplo es la clase Usuario la cual va a ser usada cuando solo se necesite gestionar un usuario.

Experto: es la clase que tiene la información necesaria para realizar la responsabilidad. Como ejemplo se tiene al propio framework, que genera las clases de acceso a datos asignándoles las responsabilidades de ejecutar todas las funcionalidades de las entidades que representan y de la cual poseen información. Otro ejemplo es la clase DatosEntrada que contiene la información necesaria para ejecutar los procedimientos que la identifican.

Controlador: lo utiliza la clase controladora como manipuladora de las peticiones o eventos del sistema.Como ejemplo se tiene la clase GenerarDatosCasoPrueba que se encarga de manejar un evento del sistema, definiendo además el método de su operación.

3.3.2 Patrones GOF

Los patrones de diseño Gof se clasifican en tres categorías: de creación, estructurales y de comportamiento. Los patrones de creación abstraen el proceso de creación de instancias, los estructurales se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño y los de comportamiento atañen a los algoritmos y a la asignación de responsabilidades entre objetos. (30).

Algunos de los patrones utilizados en la aplicación son:

Abstract Factory: pertenece a la familia de los patrones creacionales, el cual se refiere al proceso de creación de objetos. Este patrón permite que el sistema sea independiente a cómo se creen sus objetos y refleja cómo crear familias de objetos relacionados con productos sin instanciar clases directamente, se ve reflejado en las clases modelos al crear los formularios. Ejemplo:

```
class DatosEntrada(models.Model):
    nombre_de_proyecto= models.ForeignKey(Proyecto)
    nombre_caso_prueba = models.CharField(max_length=250)
    descripcion_general = models.TextField('Descripción General')

    condiciones_ejecucion = models.TextField('Condiciones de Ejecución')
    flujo_central = models.TextField()
    def get_datosentrada(self, proj):
        return DatosEntrada.objects.filter(nombre_de_proyecto = proj)
    def __str__(self):
        return self.nombre_caso_prueba
```

Factory Method: este patrón define una interfaz para la creación de un objeto, pero permitiendo a las subclases decidir de qué clase instanciarlo. Permite, por tanto, que una clase difiera la instanciación en favor de sus subclases (30). Un ejemplo de la utilización de este patrón se pone en práctica en las clases entidades que son las encargadas de crear objetos de un tipo determinado.

```
class GenerarDatosCasoPrueba(models.Model):
    caso_prueba= models.ForeignKey(DatosEntrada)
    nombre_escenario                                                    =
models.CharField(max_length=200,editable=False,blank=True, null= True)
    descripcion = models.TextField(editable=False,blank=True, null= True)
    respuesta_sistema   =   models.TextField(editable=False,blank=True,   null=
True)
    def __str__(self):
        return self.nombre_escenario
```

3.4 Diseño de la base de datos

Una de las tareas más importantes para la elaboración de un sistema web es la construcción de la base de datos. A continuación se muestra el modelo de datos del sistema:

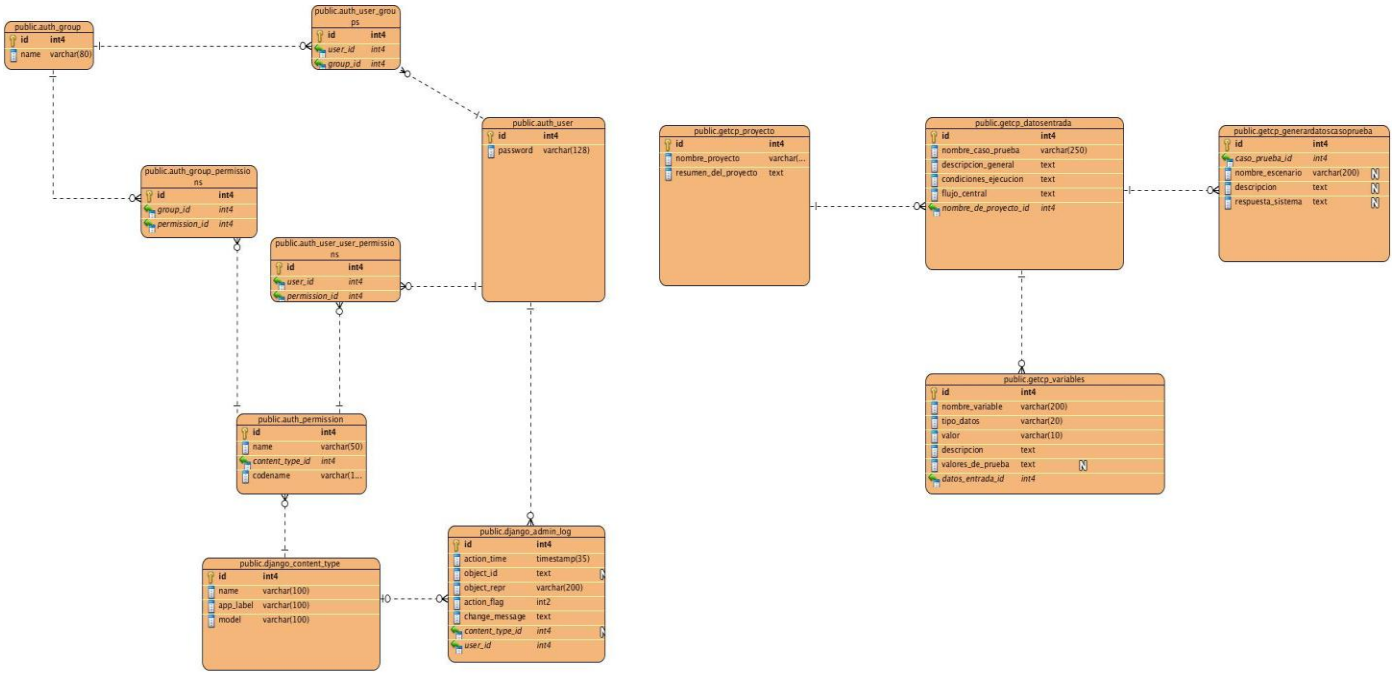

Ilustración 6. Modelo de datos

3.5 Pruebas

En la metodología XP el uso de las pruebas es fundamental, es la forma de comprobar que las funcionalidades que se implementan funcionan correctamente y cumplen con lo requerido por el cliente. Existen diferentes estrategias de pruebas, pero el estudio de este trabajo se ha centrado en las pruebas de la metodología de desarrollo de software empleada en el presente trabajo de diploma, la cual divide las pruebas de sistema en dos grupos: pruebas unitarias y pruebas de aceptación (31).

3.5.1 Pruebas Unitarias

Las pruebas unitarias son una de las piedras angulares de XP, se realizan para controlar el funcionamiento de pequeñas porciones de código. Generalmente son realizadas por el mismo programador, debido a que al conocer con mayor detalle el código, se le simplifica la tarea de elaborar conjuntos de datos de prueba para testearlo (32).

A continuación se muestran algunos códigos de las pruebas unitarias realizadas a cada clase implementada en la aplicación:

```
class TestProyecto(TestCase):
def test_get_proyecto(self):
self.fail()

class TestDatosEntrada(TestCase):
    def test_get_datosentrada(self):
        self.fail()

class TestVariables(TestCase):
    def test_get_variables(self):
        self.fail()
```

Luego de realizar las pruebas unitarias estas arrojaron a los siguientes resultados, teniéndose 5 pruebas satisfactorias y 0 no conformidades como se muestra a continuación en la gráfica:

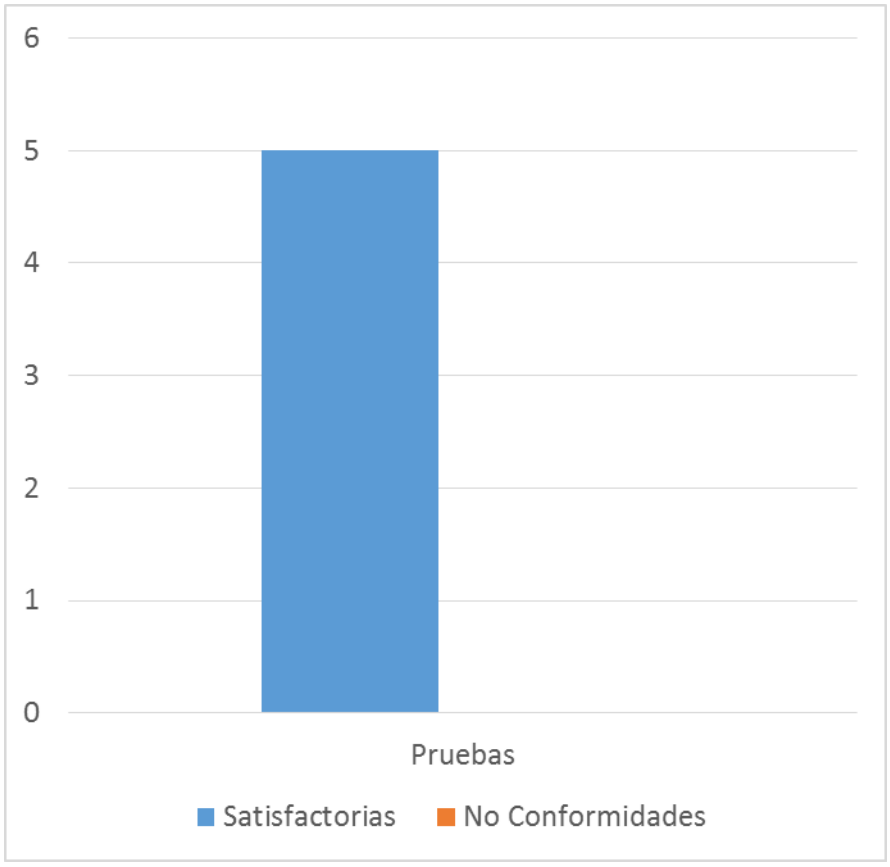

Ilustración 7. Gráfico del resultado de las Pruebas Unitarias

3.5.2 Pruebas de aceptación

Las pruebas de aceptación, se realizan sobre el producto terminado e integrado, están concebidas para que sea un usuario final quien detecte los posibles errores. Este tipo de pruebas son comúnmente realizadas por estos usuarios, quienes deben informar todas las deficiencias o errores que encuentre antes de dar por aprobado el sistema definitivamente. Se clasifican en dos tipos: pruebas Alfa y pruebas Beta.

- Pruebas Alfa: Estas pruebas consisten en hacer pruebas funcionales como clientes en un entorno de desarrollo. Se trabaja en un entorno controlado y siempre con la presencia de un miembro del

equipo de desarrollo para ayudarle a usar el sistema, para analizar los resultados y para que tenga validez la prueba realizada.

- Pruebas Beta: Las pruebas beta vienen después de las pruebas alfa, y se desarrollan en el entorno de producción; un entorno que está fuera del control de los desarrolladores. Aquí el cliente trabaja de forma independiente con el producto y trata de encontrar fallos (reales o imaginarios) de los que informa al desarrollador. Las pruebas alfa y beta son habituales en productos destinados al mercado masivo (33).

Para asegurarse que la aplicación desarrollada cumple sus requisitos, se definió realizar pruebas de aceptación, puesto que representan la satisfacción del cliente con el producto desarrollado. Estas pruebas conllevan a precisar lo que la aplicación debe hacer en determinadas circunstancias, por esto el cliente es la persona adecuada para diseñar las pruebas.

Las pruebas de aceptación son creadas en base a las HU y en cada ciclo de desarrollo. Una HU no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información (34).

A continuación se exponen los casos de pruebas de aceptación utilizados para probar el sistema divididos en las iteraciones correspondientes a cada HU:

Iteración 1:

Tabla 14. Caso de Prueba de Aceptación 1 HU Gestionar usuario

Caso de Prueba de Aceptación	
Código: HU1_p1	Historia de Usuario (No1): Gestionar usuario.
Nombre: Entrada al sistema con datos erróneos.	
Descripción: Se desea probar que el sistema no permita el acceso a un usuario con datos erróneos.	
Condiciones de ejecución: Juego de datos incorrectos o incompletos.	
Entrada/ Pasos de ejecución: Datos incorrectos. El sistema comprueba que los datos estén	

correctos para dar acceso al usuario.
Resultados esperados: Muestra un mensaje de error: Nombre de usuario/contraseña inválidos.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 15. Caso de Prueba de Aceptación 2 HU Gestionar usuario

Caso de Prueba de Aceptación	
Código: HU1_p2	Historia de Usuario (No1): Gestionar usuario.
Nombre: Entrada al sistema con datos válidos.	
Descripción: Se desea probar que el sistema permita el acceso a un usuario.	
Condiciones de ejecución: Juego de datos válidos.	
Entrada/ Pasos de ejecución: El sistema comprueba que los datos estén correctos para dar acceso al usuario.	
Resultados esperados: Entrada al sistema del usuario.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 16. Caso de Prueba de Aceptación 3 HU Gestionar usuario

Caso de Prueba de Aceptación	
Código: HU1_p3	Historia de Usuario (No1): Gestionar usuario.
Nombre: Adicionar un nuevo usuario con datos incorrectos o incompletos.	
Descripción: Se desea probar que el sistema no permita registrar un nuevo usuario con datos incorrectos o incompletos.	
Condiciones de ejecución: Se introducen datos de los nuevos usuarios incompletos o incorrectos.	
Resultados esperados: Muestra un mensaje de respuesta del sistema “Por favor, corrija los errores” y señala el campo o los campos con errores.	

Evaluación de la prueba: Prueba satisfactoria.

Tabla 17. Caso de Prueba de Aceptación 4 HU Gestionar usuario

Caso de Prueba de Aceptación	
Código: HU1_p4	Historia de Usuario (No1): Gestionar usuario.
Nombre: Adicionar un nuevo usuario con datos correctos.	
Descripción: Se desea probar que el sistema permita registrar un nuevo usuario con datos correctos.	
Condiciones de ejecución: Se introducen datos correctos del nuevo usuario.	
Resultados esperados: Muestra un mensaje de respuesta del sistema “Se ha añadido con éxito el nuevo usuario, puede continuar editándolo abajo”.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 18. Caso de Prueba de Aceptación 5 HU Gestionar usuario

Caso de Prueba de Aceptación	
Código: HU1_p5	Historia de Usuario (No1): Gestionar usuario.
Nombre: Modificar usuario introduciendo nuevos datos incorrectos o incompletos.	
Descripción: Se desea probar que el sistema no permita modificar un usuario introduciendo datos incorrectos o incompletos.	
Condiciones de ejecución: Se introducen nuevos datos de usuarios incompletos o incorrectos.	
Resultados esperados: Muestra un mensaje de respuesta del sistema “Por favor, corrija los errores” y señala el campo o los campos con errores.	

Evaluación de la prueba: Prueba satisfactoria.

Tabla 19. Caso de Prueba de Aceptación 6 HU Gestionar usuario

Caso de Prueba de Aceptación	
Código: HU1_p6	Historia de Usuario (No1): Gestionar usuario.
Nombre: Modificar usuario introduciendo nuevos datos correctos.	
Descripción: Se desea probar que el sistema permita modificar un usuario introduciendo datos correctos.	
Condiciones de ejecución: Se introducen nuevos datos usuarios correctos.	
Resultados esperados: Muestra un mensaje de respuesta del sistema “El usuario ha sido modificado de forma correcta”.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 20. Caso de Prueba de Aceptación 7 HU Gestionar usuario

Caso de Prueba de Aceptación	
Código: HU1_p7	Historia de Usuario (No1): Gestionar usuario.
Nombre: Eliminar usuario.	
Descripción: Se desea probar que el sistema elimine un usuario de forma correcta.	
Condiciones de ejecución: Se accede a la opción de eliminar un usuario.	
Resultados esperados: Muestra un mensaje de respuesta del sistema “El usuario ha sido eliminado de forma correcta”.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 21. Caso de Prueba de Aceptación 1 HU Gestionar proyecto

Caso de Prueba de Aceptación	
Código: HU2_p1	Historia de Usuario (No2): Gestionar proyecto.
Nombre: Crear un nuevo proyecto con juego de datos incorrectos o incompletos.	
Descripción: Probar que no se pudo crear el nuevo proyecto con los datos incorrectos o incompletos.	
Condiciones de ejecución: Deben existir errores en el juego de datos del nuevo proyecto.	
Resultados esperados: El sistema debe mostrar un mensaje de error ¨Existen datos Incorrectos¨.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 22. Caso de Prueba de Aceptación 2 HU Gestionar proyecto

Caso de Prueba de Aceptación	
Código: HU2_p2	Historia de Usuario (No2): Gestionar proyecto.
Nombre: Crear un nuevo proyecto con juego de datos correctos.	
Descripción: Probar que se puede crear un nuevo proyecto con juego de datos correctos.	
Condiciones de ejecución: No existen errores en el juego de datos del nuevo proyecto.	
Resultados esperados: El sistema muestra el siguiente mensaje “Se ha creado el nuevo proyecto de forma satisfactoria”	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 23. Caso de Prueba de Aceptación 3 HU Gestionar proyecto

Caso de Prueba de Aceptación	
Código: HU2_p3	Historia de Usuario (No2): Gestionar proyecto.

Nombre: Modificar un proyecto con juego de datos incorrectos o incompletos.
Descripción: Probar que no se pudo modificar el proyecto con los datos incorrectos o incompletos.
Condiciones de ejecución: Deben existir errores en el juego de datos que se modifican.
Resultados esperados: El sistema debe mostrar un mensaje de error ¨Existen datos Incorrectos¨.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 24. Caso de Prueba de Aceptación 4 HU Gestionar proyecto

Caso de Prueba de Aceptación	
Código: HU2_p4	Historia de Usuario (No2): Gestionar proyecto.
Nombre: Modificar un proyecto con juego de datos correctos.	
Descripción: Probar que se puede modificar un proyecto con juego de datos correctos.	
Condiciones de ejecución: No existen errores en el juego de datos que se va a introducir.	
Resultados esperados: El sistema muestra el siguiente mensaje “Se ha modificado el proyecto de forma satisfactoria”.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 25. Caso de Prueba de Aceptación 5 HU Gestionar proyecto

Caso de Prueba de Aceptación	
Código: HU2_p5	Historia de Usuario (No2): Gestionar proyecto.
Nombre: Eliminar proyecto.	
Descripción: Se desea probar que el sistema elimine un proyecto de forma correcta.	
Condiciones de ejecución: Se accede a la opción de eliminar proyecto.	
Resultados esperados: Muestra un mensaje de respuesta del sistema “El proyecto ha sido	

eliminado satisfactoriamente".
Evaluación de la prueba: Prueba satisfactoria.

Iteración 2:

Tabla 26. Caso de Prueba de Aceptación 1 HU Gestionar datos del caso de prueba

Caso de Prueba de Aceptación	
Código: HU3_p1	Historia de Usuario (No3): Gestionar datos de caso de prueba.
Nombre: Adicionar datos del caso de prueba con juego de datos incorrectos o incompletos.	
Descripción: Probar que no se pueda adicionar datos incorrectos o incompletos del caso de prueba.	
Condiciones de ejecución: Datos de entrada del caso de prueba son incorrectos.	
Resultados esperados: El sistema muestra un mensaje de error ¨Existen datos Incorrectos¨.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 27. Caso de Prueba de Aceptación 2 HU Gestionar datos del caso de prueba

Caso de Prueba de Aceptación	
Código: HU3_p2	Historia de Usuario (No3): Gestionar datos de caso de prueba.
Nombre: Adicionar datos del caso de prueba con juego de datos correctos.	
Descripción: Probar que se pueda adicionar datos del caso de prueba.	
Condiciones de ejecución: Datos de entrada del caso de prueba correctos.	
Resultados esperados: El sistema muestra un mensaje "Se han insertado los datos".	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 28. Caso de Prueba de Aceptación 3 HU Gestionar datos del caso de prueba

Caso de Prueba de Aceptación	
Código: HU3_p3	Historia de Usuario (No3): Gestionar datos de caso de prueba.
Nombre: Modificar datos del caso de prueba con juego de datos incorrectos o incompletos.	
Descripción: Probar que no se pueda modificar datos incorrectos o incompletos del caso de prueba.	
Condiciones de ejecución: Datos de entrada de los casos de prueba incorrectos o incompletos.	
Resultados esperados: El sistema muestra un mensaje de respuesta del sistema ¨Existen datos Incorrectos¨.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 29. Caso de Prueba de Aceptación 4 HU Gestionar datos del caso de prueba

Caso de Prueba de Aceptación	
Código: HU3_p4	Historia de Usuario (No3): Gestionar datos de caso de prueba.
Nombre: Modificar datos del caso de prueba con juego de datos correctos.	
Descripción: Probar que se pueda modificar datos del caso de prueba de forma correcta.	
Condiciones de ejecución: Datos de entrada del caso de prueba correctos.	
Resultados esperados: El sistema muestra un mensaje de información “Se han modificado los datos de forma satisfactoria”.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 30. Caso de Prueba de Aceptación 5 HU Gestionar datos del caso de prueba

Caso de Prueba de Aceptación	
Código: HU3_p5	Historia de Usuario (No2): Gestionar datos de caso de prueba.
Nombre: Eliminar datos de caso de prueba.	
Descripción: Se desea probar que el sistema elimine datos de caso de prueba de forma correcta.	
Condiciones de ejecución: Se accede a la opción de eliminar datos de caso de pruebas.	
Resultados esperados: Muestra un mensaje de información “Los datos han sido eliminado satisfactoriamente”.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 31. Caso de Prueba de Aceptación 6 HU Gestionar datos del caso de prueba

Caso de Prueba de Aceptación	
Código: HU3_p6	Historia de Usuario (No2): Gestionar datos de caso de prueba.
Nombre: Eliminar datos de caso de prueba.	
Descripción: Se desea probar que el sistema elimine datos de caso de prueba de forma correcta.	
Condiciones de ejecución: Se accede a la opción de eliminar datos de caso de pruebas.	
Resultados esperados: Muestra un mensaje de información “Los datos han sido eliminado satisfactoriamente”.	
Evaluación de la prueba: Prueba satisfactoria.	

Iteración 3:

Tabla 32. Caso de Prueba de Aceptación 1 HU Generar Diseños de casos de Pruebas

Caso de Prueba de Aceptación	
Código: HU4_p5	Historia de Usuario (No2): Generar Diseños de Casos de Pruebas.
Nombre: Genera diseños de casos de prueba.	
Descripción: Se desea probar que el sistema genere diseños de casos de prueba.	
Condiciones de ejecución: Se accede a la opción de generar diseños de casos de prueba. Existe al menos un proyecto con los datos de un caso de prueba.	
Resultados esperados: Muestra una interfaz con los diseños de casos de prueba.	
Evaluación de la prueba: Prueba satisfactoria.	

3.5.3 Resultados de las pruebas

Se realizaron las pruebas de aceptación, dividiéndose en 3 iteraciones para probar el correcto funcionamiento del sistema. Se detectaron un total de 8 no conformidades y 18 recomendaciones. A continuación se muestra en el gráfico el resultado antes mencionado.

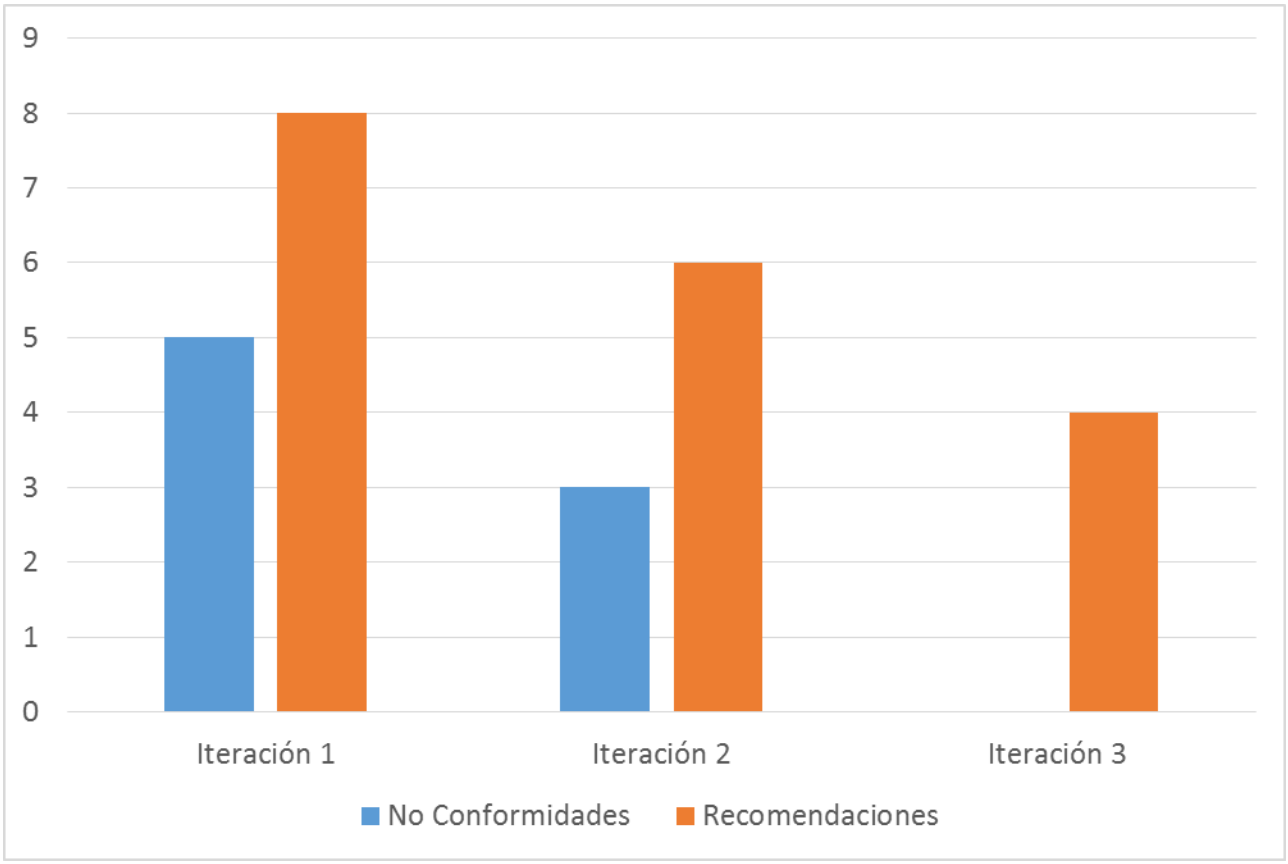

Ilustración 8. Gráfico de las no conformidades

3.6 Conclusiones

En el presente capítulo se plantearon las tarjetas CRC, los patrones arquitectónicos y de diseños utilizados en el desarrollo de la aplicación. Se realizó el modelo de datos del sistema, además de las pruebas de aceptación que se llevaron a cabo con el fin de brindar un producto que cumpla con los objetivos trazados. Con la conclusión de este capítulo se considera terminada la propuesta de solución.

Conclusiones Generales

Después de elaborar la Herramienta para la Generación Automática de Diseños de Casos de Prueba se llegó a las siguientes conclusiones:

Los métodos científicos utilizados permitieron desarrollar los conceptos y teorías que sustentan la investigación para el desarrollo de la solución. La selección de la metodología de desarrollo, lenguaje de programación, tecnologías y herramientas posibilitaron cumplir con el objetivo general definido.

La implementación de la Herramienta para la Generación de Diseños de Casos de Prueba para las aplicaciones desarrolladas con Xilema-Base-Web posibilitó obtener un producto funcional capaz de satisfacer las necesidades del cliente y las pruebas realizadas a la solución demostraron el correcto funcionamiento del sistema.

De forma general, se cumplió con el objetivo propuesto llegando a una solución que facilitará el proceso de diseño de casos de prueba para las aplicaciones desarrolladas con Xilema-Base-Web.

Recomendaciones

Como parte del proceso de desarrollo de la investigación se recomiendan los siguientes aspectos:

1. Mantener el desarrollo continuo e incremental de la herramienta realizada, atendiendo a los cambios que puedan surgir a lo largo de su utilización.
2. Adicionarle nuevas funcionalidades para un mejor empleo de la herramienta, ejemplo de estas funcionalidades puede ser generar los casos de prueba de forma separada.
3. Añadir nuevos casos de pruebas a generar, por ejemplo el Importar.
4. Utilizar un componente para exportar automáticamente los diseños de casos de prueba a un documento con formato xls que presente un mejor formato que el actual.

Referencias bibliográficas

1. Brito Viñas BC, Hernández Pérez G, Álvarez González A. Ciencia, gestión tecnológica y desarrollo sostenible y solidario en los países latinoamericanos: experiencia cubana. Espacios. [En línea] 19 de 2 de 2000. [Citado el: 15 de 12 de 2014.] http://www.revistaespacios.com/a98v19n02/40981902.html.

2. Gestión de Calidad y Pruebas de Software.[En línea] [Citado el: 16 de 12 de 2014.] http://pruebasdesoftware.com/.

3. Portal Universidad de las Ciencias Informáticas. [En línea] 2012. [Citado el: 16 de 12 de 2014.] http://www.uci.cu..

4. Aenor. *Sistemas de gestión de la calidad.* Barcelona : s.n., 2011. (ISO 9000:2000).

5. Pressman, Roger S. *Ingeniería de Software.* Sexta Edición. New York : s.n., 2010. págs. 768-770.

6. Quality Assurance & Software Testing. [En línea] 2008. [Citado el: 18 de 12 de 2014.] http://calidadysoftware.com/..

7. Pressman, Roger S. *Ingeniería de Software.* Sexta Edición. New York : s.n., 2010. págs. 770-775.

8. *Ingeniería de Software.* Sexta Edición. New York : s.n., 2010. págs. 419-433.

9. *Ingeniería de Software.* Sexta Edición. New York : s.n., 2010. págs. 433-434.

10.*Ingeniería de Software.* Sexta Edición. New York : s.n., 2010. págs. 436-438.

11. Sommerville, Roger. *Ingeniería de Software.* Octava Edición. China Hong Kong : s.n., 2007. págs. 551-561.

12. autores, Colectivo de. *Actas de los Talleres de Ingeniería de Software y Base de Datos.* 2009.

13. Quality Assurance & Software Testing. . [En línea] 2008. [Citado el: 12 de 1 de 2015.] http://calidadysoftware.com/.

14. Pérez, Isaías Carrillo. *Metodología de desarrollo de Software .* Valencia Universidad Politécnica : s.n., 2008. ISBN.

15. Procesos de Software - Metodología Extreme Programming(XP). [En línea] 20 de marzo de 2013. [Citado el: 20 de 1 de 2015.] http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP.

16. Visual Paradigm. [En línea] 1995. [Citado el: 25 de enero de 2015.] http://www.visual-paradigm.com. ISBN.

17. Portal oficial de Postgres [Citado el: 25 de enero de 2015.] . [En línea] 2013 de 10 de 2013. [Citado el: 25 de enero de 2015.] http://www.postgressql.org.es. ISBN.

18. Comunidad de software libre Paraguay - Framework de Desarrollo . [En línea] 24 de marzo de 2013. [Citado el: 27 de enero de 2015.] http://www.pti.org.py/csl/index.php....

19. Plugin para Xilema Base Web. [En línea] 18 de marzo de 2015. [Citado el: 18 de abril de 2015.] Plugin para Xilema Base Web-Wiki GRHS.html.

20. Usuario(Infomática). [En línea] 2013. [Citado el: 19 de abril de 2015.] http://www.ecured.cu/index.php/Usuario_%28Inform%C3%A1tica%29.

21. Joskowicz, José. *Reglas y Prácticas en eXtremeProgramming.* 2008. ISBN.

22. Riola, José Carlos Carvajal. *Metodologías ágiles: Herramientas y modelo de desarrollo para aplicaciones Java EE como métodología empresarial.* 2008. ISBN.

23. Ramírez, Ing.Danay Pérez. *METODOLOGÍAS ÁGILES.¿CÓMO DESARROLLO UTILIZANDO XP.* Habana,Cuba,Cujae : s.n., 2008. ISBN.

24. Prototipo de la Interfaz de Usuario - MeRinde. [En línea] 2013. [Citado el: 5 de mayo de 2015.] http://merinde.net/index.php?option=com_content&task=view&id=490&Itemid=291.

25. García, Arlee Enedina Paneque y Noel González. *Sistema de Gestión de Auditoría de la Dirección de Supervisión y Control de la Universidad de las Ciencias Informáticas.* Ciudad Habana,Cuba : s.n., 2010.

26. C, Larman. *UML y patrones págs.* 2010. págs. 185-215. Vol. Vol. Tomo I.

27. Patrones arquitectónicos . [En línea] 2013. [Citado el: 10 de mayo de 2015.] http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos...

28. Henney, Kevlin. *¿What is Software Architecture? .* 2007.

29. Alvarez, Miguel Angel. Novedades de HTML 5. ¿ Qué es HTML 5? [En línea] 2009. [Citado el: 29 de 1 de 2015.] http://www.desarrolloweb.com/articulos/que-es-html5.html.. . ISBN.

30. Madrid, Facultad de Informática_Universidad Politecnica de. *Unidad Docente Ingeniería del software Patrones del Gang of Four.* Madrid : s.n., 2003. ISBN.

31. J Gutíerrez, M.J Escalona, M.Mejías, J.Torres. *Pruebas del sistema en Programación Extrema.* Sevilla : s.n.

32. Malforá, Dayvis, y otros. *Testing en eXtreme Programing.* 2006.

33. Pruebas de aceptación. Pruebas de aceptación. [En línea] [Citado el: 18 de Mayo de 2015.] http://www.es.testhouse.net/pruebas-de-aceptacion/..

34. Joskowicz, José. *Reglas y Prácticas en eXtremeProgramming.* España : s.n., 2008.

Bibliografía

1. Pressman, Roger S. Ingeniería de Software.
2. Sistemas de gestión de la calidad(ISO 9000:2000).
3. Estrategia para la aplicación de Pruebas de Caja Blanca y Caja Negra al proyecto Registros y Notarías. La Habana, Cuba : s.n., 2008.
4. Valbuena, Sonia Jaramilla. Programación Orientada a Objetos. Armenia, Quindío : s.n., 2010.
5. Portal oficial de selenium. Selenium. Portal oficial de selenium. Selenium. [En línea] 2010. http://www.selenium.hq.org. ISBN. Marzo de 2013. [Citado el: 23 de Enero de 2015.]
6. Pérez, Isaías Carrillo. Metodología de desarrollo de Software. Valencia: Universidad Politécnica : s.n., 2008. ISBN.
7. Ramírez, Ing.Danay Pérez. METODOLOGÍAS ÁGILES.¿CÓMO DESARROLLO UTILIZANDO XP? Habana,Cuba,Cujae : s.n., 2008.
8. Henney, Kevlin. ¿What is Software Architecture? . 2007.
9. ISBN., Larman C. UML y patrones págs. 185-215. Vol. Tomo I. 2010. ISBN.
10. Unidad Docente Ingeniería del software Patrones del Gang of Four. Facultad de Informática_Universidad Politecnica de Madrid : s.n., 2003. ISBN.
11. Pruebas de aceptación. Pruebas de aceptación 2005.http://www.es.testhouse.net/pruebas-de-aceptacion/.
12. J Gutíerrez, M.J Escalona, M.Mejías, J.Torres. Pruebas del sistema en Programación Extrema. Sevilla
13. Aenor. *Sistemas de gestión de la calidad.* Barcelona : s.n., 2011. (ISO 9000:2000).
14. Ramírez, Ing.Danay Pérez. *METODOLOGÍAS ÁGILES.¿CÓMO DESARROLLO UTILIZANDO XP.* Habana,Cuba,Cujae : s.n., 2008. ISBN.
15. Malforá, Dayvis, y otros. *Testing en eXtreme Programing.* 2006.
16. Joskowicz, José. *Reglas y Prácticas en eXtremeProgramming.* España : s.n., 2008.
17. C, Larman. *UML y patrones págs.* 2010. Vol. Tomo I.
18. Henney, Kevlin. *¿What is Software Architecture? .* 2007.
19. Riola, José Carlos Carvajal. *Metodologías ágiles: Herramientas y modelo de desarrollo para aplicaciones Java EE como métodología empresarial.* 2008. ISBN.

20. Pérez, Isaías Carrillo. *Metodología de desarrollo de Software .* Valencia Universidad Politécnica : s.n., 2008. ISBN.
21. autores, Colectivo de. *Actas de los Talleres de Ingeniería de Software y Base de Datos.* 2009.

Anexos

Tabla 33.HU. Adicionar Usuario

Historia de Usuario	
Número: 5	**Nombre de la Historia de Usuario:** Adicionar Usuario
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados: 0.05**
Riesgo en desarrollo: Alto	**Puntos reales: 0.05**
Descripción: El sistema deber permitir al usuario la opción de adicionar un nuevo usuario.	

Tabla 34.HU. Modificar Usuario

Historia de Usuario	
Número: 6	**Nombre de la Historia de Usuario**: Modificar Usuario
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados: 0.05**
Riesgo en desarrollo: Alto	**Puntos reales: 0.05**
Descripción: El sistema deber permitir al usuario la opción de modificar un usuario.	

Tabla 35.HU. Eliminar Usuario

Historia de Usuario	
Número: 7	**Nombre de la Historia de Usuario**: Eliminar Usuario
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados: 0.05**
Riesgo en desarrollo: Alto	**Puntos reales: 0.05**
Descripción: El sistema deber permitir al usuario la opción eliminar un usuario.	

Tabla 36.HU. Autenticar Usuario

Historia de Usuario	
Número: 8	**Nombre de la Historia de Usuario**: Autenticar Usuario
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados: 0.05**
Riesgo en desarrollo: Alto	**Puntos reales: 0.05**

Descripción: El sistema deber permitir al usuario la opción de autenticar un usuario.

Tabla 37.HU. Adicionar Proyecto

Historia de Usuario	
Número: 9	**Nombre de la Historia de Usuario:** Adicionar Proyecto
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados: 0.13**
Riesgo en desarrollo: Alto	**Puntos reales: 0.13**
Descripción: El sistema permitirá al usuario la posibilidad de adicionar un proyecto en el sistema.	

Tabla 38.HU. Modificar Proyecto

Historia de Usuario	
Número: 10	**Nombre de la Historia de Usuario:** Modificar Proyecto
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados: 0.13**
Riesgo en desarrollo: Alto	**Puntos reales: 0.13**
Descripción: El sistema permitirá al usuario la posibilidad de modificar un proyecto en el sistema.	

Tabla 39.HU. Eliminar Proyecto

Historia de Usuario	
Número: 11	**Nombre de la Historia de Usuario:** Eliminar Proyecto
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados: 0.13**
Riesgo en desarrollo: Alto	**Puntos reales: 0.13**
Descripción: El sistema permitirá al usuario la posibilidad de eliminar un proyecto en el sistema.	

Tabla 40.HU. Adicionar datos de Caso de Prueba

Historia de Usuario	
Número: 12	**Nombre de la Historia de Usuario:** Adicionar datos de Caso de Prueba.
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos reales: 2.25**
Descripción: El sistema debe permitir al usuario la posibilidad de adicionar los datos de un Caso de Prueba.	

Tabla 41.HU. Modificar datos de Caso de Prueba

Historia de Usuario	
Número: 21	**Nombre de la Historia de Usuario:** Modificar datos de Caso de Prueba.
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales: 1.6**
Descripción: El sistema debe permitir al usuario la posibilidad de modificar los datos de Caso de Prueba.	

Tabla 42.HU. Eliminar datos de Caso de Prueba

Historia de Usuario	
Número: 30	**Nombre de la Historia de Usuario:** Eliminar datos de Caso de Prueba.
Cantidad de modificaciones a la Historia de usuario: 0	
Usuario: Analista	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 0.3
Riesgo en desarrollo: Alto	**Puntos reales: 0.35**
Descripción: El sistema debe permitir al usuario la posibilidad de eliminar los datos del Caso de Prueba.	

Tabla 43. Tarea de Ingeniería 2 Gestionar usuario

Tarea de Ingeniería	
Número de la tarea: 2	**Número de Historia de Usuario:** 1(Gestionar Usuario).
Nombre de Tarea: Modificar Usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha Inicio: 4de febrero 2015	**Fecha Fin:** 5 de febrero 2015
Programador responsable: Yenisleidi Serrano	
Descripción: El sistema permite modificar un usuario mostrando la opción: modificar usuario, el usuario lo selecciona y muestra una interfaz de formulario requiriendo los campos para modificar un usuario determinado.	

Tabla 44. Tarea de Ingeniería 3 Gestionar usuario

Tarea de Ingeniería	
Número de la tarea: 3	**Número de Historia de Usuario:** 1(Gestionar Usuario).
Nombre de Tarea: Eliminar Usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha Inicio: 4de febrero 2015	**Fecha Fin:** 5 de febrero 2015
Programador responsable: Yenisleidi Serrano	

Descripción: El sistema permite eliminar un usuario mostrando la opción: eliminar usuario, el usuario selecciona eliminar y muestra un mensaje diciendo si está seguro de eliminar el usuario.

Tabla 45. Tarea de Ingeniería 4 Gestionar usuario

Tarea de Ingeniería	
Número de la tarea: 4	**Número de Historia de Usuario:** 1(Gestionar Usuario).
Nombre de Tarea: Autenticar Usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha Inicio: 4de febrero 2015	**Fecha Fin:** 5 de febrero 2015
Programador responsable: Yenisleidi Serrano	
Descripción: El sistema permite autenticar un usuario mostrando la opción: autenticar usuario, el usuario lo selecciona y muestra un mensaje diciendo que el usuario ha sido autenticado.	

Tabla 46. Tarea de Ingeniería 6 Gestionar proyecto

Tarea de Ingeniería	
Número de la tarea: 6	**Número de Historia de Usuario:** 2(Gestionar Proyecto).
Nombre de Tarea: Modificar Proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2

Fecha Inicio: 6 de febrero 2015	**Fecha Fin:** 7 de febrero 2015
Programador responsable: Yenisleidi Serrano	
Descripción: El sistema permite modificar un proyecto mostrando la opción: Modificar proyecto, el usuario lo selecciona y muestra una interfaz de formulario requiriendo los campos para modificar un nuevo proyecto.	

Tabla 47. Tarea de Ingeniería 7 Gestionar proyecto

Tarea de Ingeniería	
Número de la tarea: 7	**Número de Historia de Usuario:** 2(Gestionar Proyecto).
Nombre de Tarea: Eliminar Proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha Inicio: 6 de febrero 2015	**Fecha Fin:** 7 de febrero 2015
Programador responsable: Yenisleidi Serrano	
Descripción: El sistema permite eliminar un proyecto mostrando la opción: Eliminar proyecto, el usuario lo selecciona y muestra un mensaje preguntando que si está seguro de eliminar el proyecto.	

Tabla 48. Tarea de Ingeniería 8 Gestionar datos de caso de prueba

Tarea de Ingeniería	
Número de la tarea: 8	**Número de Historia de Usuario:** 3(Gestionar

	Datos de Caso de Prueba).
Nombre de Tarea: Adicionar Datos de Caso de Prueba.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.25
Fecha Inicio: 8 de febrero 2015	**Fecha Fin:** 20 de febrero 2015
Programador responsable: Osmel Roján Herrera	
Descripción: El sistema permite eliminar un proyecto mostrando la opción: Eliminar proyecto, el usuario lo selecciona y muestra un mensaje preguntando que si está seguro de eliminar el proyecto.	

Tabla 49. Tarea de Ingeniería 9 Gestionar datos de caso de prueba

Tarea de Ingeniería	
Número de la tarea: 9	**Número de Historia de Usuario:** 3(Gestionar Datos de Caso de Prueba).
Nombre de Tarea: Modificar Datos de Caso de Prueba.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.25
Fecha Inicio: 1 de marzo 2015	**Fecha Fin:** 9 de marzo 2015
Programador responsable: Osmel Roján Herrera	
Descripción: El sistema permite modificar los datos de caso de prueba mostrando la opción: Modificar Datos de Caso de Prueba, el usuario lo selecciona y muestra una interfaz con los datos de caso de prueba.	

Tabla 50. Tarea de Ingeniería 10 Gestionar datos de caso de prueba

Tarea de Ingeniería	
Número de la tarea: 10	**Número de Historia de Usuario:** 3(Gestionar Datos de Caso de Prueba).
Nombre de Tarea: Eliminar Datos de Caso de Prueba.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.25
Fecha Inicio: 10 de marzo 2015	**Fecha Fin:** 12 de marzo 2015
Programador responsable: Osmel Roján Herrera	
Descripción: El sistema permite eliminar los datos de caso de prueba mostrando la opción: Eliminar Datos de Caso de Prueba, el usuario lo selecciona y muestra un mensaje diciendo que si está seguro de eliminarlo.	

Tabla 51. Tarea de Ingeniería 11 Generar diseño de caso de prueba

Tarea de Ingeniería	
Número de la tarea: 11	**Número de Historia de Usuario:** 4(Generar Diseño de Casos de Prueba).
Nombre de Tarea: Generar Diseño de Casos de Prueba.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4.5
Fecha Inicio: 1 de abril 2015	**Fecha Fin:** 23 de abril 2015
Programador responsable: Osmel Roján Herrera	

Descripción: El sistema permite generar los datos de caso de prueba mostrando la opción: Generar Datos de Caso de Prueba, el usuario lo selecciona y se generan automáticamente los casos de prueba.

Tabla 52.Tarjeta CRC Gestionar Proyecto

Clase: Proyecto	
Responsabilidad	**Colaboración**
Gestionar Proyecto.	

Tabla 53.Tarjeta CRC Variables

Clase: Variables	
Responsabilidad	**Colaboración**
Gestionar Variables	• Datos de Entrada

Tabla 54.Tarjeta CRC Gestionar Usuarios

Clase: Usuario	
Responsabilidad	**Colaboración**
Gestionar Usuario.	