Facultad 4

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

Título: "Herramienta de Administración de requisitos para los proyectos del centro FORTES**"**.

Autor: Ancel Eulalio Núñez Martínez.

Tutor(es): Ing. Enerys Mesa Morales.
Ing. José Antonio Soto Pérez.

La Habana, 2011.
"Año 53 de la Revolución".

Declaración de autoría.

Declaro ser el único autor de este trabajo y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Firma del autor:
Ancel Eulalio Núñez Martínez.

_____

Firma del tutor(a):
Ing. Enerys Mesa Morales.

_____

Firma del tutor:
Ing. José Antonio Soto Pérez.

Datos de contacto.

Nombre y apellidos: Enerys Mesa Morales.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: emesa@uci.cu

Ingeniera en Ciencias Informáticas, en la Universidad de Ciencias Informáticas (UCI) en el 2009, Adiestrada, dos años de experiencia en la temática, dos años de graduada.

Nombre y apellidos: José Antonio Soto Pérez.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: jasoto@uci.cu

Ingeniero en Ciencias Informáticas, en la Universidad de Ciencias Informáticas (UCI) en el 2009, Adiestrado, dos años de experiencia en la temática, dos años de graduado.

DEDICATORIA

La realización de este gran sueño se lo dedico a:

Las dos mujeres más importantes en mi vida, las que me han demostrado su amor incondicional, su cariño y confianza. Para mí han sido un gran ejemplo a seguir en todo momento:

A mi abuelita Esther, que no tuve la alegría de seguir creciendo a su lado, ya que solo pude disfrutar siete años del inmenso amor que tenía para darme, esos años junto a ella fueron más que suficiente para conocer lo importante que es ser un hombre de bien, hacer las cosas correctamente y enmendar los errores que se pueden cometer en la vida. Ella me enseñó a ser la persona que hoy soy. Gracias abuelita por convertirme en ese hombre que siempre quisiste ver en mí, por dibujarme el camino hasta donde he llegado. Yo sé que en estos momentos me debes estar mirando en donde quiera que estés y sintiéndote orgullosa de mí, pero ojalá y la vida te hubiese dado la oportunidad de verme convertido en todo un ingeniero. Te quiero mucho.

A mi mamá por ser esa madre ejemplar que todos queremos tener, por ser esa persona maravillosa, llena de amor, de detalles y de dulzura. Por brindarme su confianza y cariño de madre, por ser amiga, compañera, madre, padre, consejera, por guiarme siempre por el camino correcto, por darme una excelente formación, por enseñarme que la perseverancia y el esfuerzo son el camino para lograr los objetivos propuestos, por darme amor y brindarme su apoyo en todo momento, por ser la única en confiar en mí y concretar el sueño de mi abuela. Para ti va dedicado este triunfo. Mami si por estar lejos de ti en 10 años de estudio o por mi forma de ser no te he demostrado todo lo que siento por ti, quiero que veas mi cariño reflejado a través de este logro, que solo yo sé todo lo que has luchado para verme hoy aquí; decirte lo mucho que te quiero y te admiro no bastaría. Gracias por ser mi madre y una guía para mí.

AGRADECIMIENTOS

Agradecerle especialmente a mi tutora Enerys, por llegar en el momento indicado, cuando mi vida transitaba por un período muy difícil, cuando me hacía falta que alguien creyera en mí de la forma en que ella lo hizo. Más que mi tutora ha sido mi amiga, mi compañera de tesis y mi hermana, me ha apoyado en todo momento desde los inicios de esta investigación, ha sido una persona muy especial con la cual he contado para la realización de este trabajo; por lo que gran parte de que hoy sea un ingeniero se lo debo a ella. Por su dedicación y entrega incondicional; muchas gracias mi hermanita.

A mi abuelita Esther por dibujarme el camino correcto con su amor y cariño.

A mi mamá Deisy por brindarme su amor incondicional en todo momento, y por ser la única persona que confió en mí desde el principio.

A mi papá Lázaro por siempre estar ahí, apoyándome y siendo la bujía inspiradora en los últimos años de mi carrera.

A mi hermanito Yancel por crecerse antes de tiempo y ser el hombre de la casa en mi ausencia.

A mis hermanos Bárbara, Joel y Joan por siempre estar ahí y apoyarme en todo momento.

A Mailyn por ser mi muñequita y regalarme los momentos más lindos de mi vida en la UCI.

A Evelio por estar siempre al lado de mi mamá apoyándola y cuidándola en mi tiempo de estudios.

A mi familia en general por estar ahí cuando los necesitaba.

A mis hermanos del preuniversitario Margot, Frank, Yasser y Lorenzo por su apoyo y confianza.

A todos los profesores que aportaron su granito de arena a lo largo de mi vida.

A todos los profesores que me formaron como Ingeniero en Ciencias Informáticas durante estos 5 años de la UCI, en especial Rogelio, Sandra, Dianelys, Fidel, Dailyn, Osdalme y Bartolo.

Al grupo 8101 que fue mi primer grupo y los considero como una gran familia.

A mis hermanos del antiguo grupo 8101 Carlos, Mario, Guille, Alejandro, Joel y Miguel por compartir buenos y malos momentos, aquí y en Venezuela.

AGRADECIMIENTOS

Al grupo 8202 que me acogieron como uno más de ellos.

Al grupo 4405, que desde Venezuela conformamos una linda familia.

Al conjunto folklórico Ilú Ashé, donde estuve 4 años maravillosos, descubriendo una nueva faceta de mi vida: el baile.

Al dream team Anett, Zuly, Clary, Mily y Agustín que sin ellos no hubiera obtenido todos los logros y alegrías de estos años en la UCI.

A mi hermanita Yolaida por ayudarme y aconsejarme en los momentos que lo necesité.

A Ilenis por apoyarme y estar siempre pendiente de mí.

A José porque en el momento difícil de la tesis me dio su apoyo incondicional, como tutor y amigo.

Al tribunal por apoyarme y darme fuerza para seguir hacia delante en los momentos difíciles de la tesis.

A Mercy, Julito y Dianelys por brindarme su ayuda en todo momento durante mi estancia en la universidad.

Al piquete del apto Ale, Julio, Anita, Idalmis, Lianet, Adrián, Yerandy, el Flaco, Yunior, Leo, Yadima, Noslén, Yane y las peques que siempre estuvieron ahí, para brindarme una mano cuando la necesitaba.

A mi equipo de balonmano en especial a Kenier y Javier.

A mi piquete de la facultad 2 Aimé Regla, Maqui, Yaineris, Yanko y Los trigueños que con ellos en estos 5 años compartí momentos muy buenos.

Al piquete de los ranchones, discotecas, macumbas, amanezcos, entre otros, que hicieron mis tiempos libres de la uci más amenos.

En fin a todos los que tuvieron que ver de una forma u otra en mi formación como profesional en los 5 años de estancia en la universidad.

Resumen.

La gestión de requisitos es un proceso que puede definir la calidad de un software y constituye además un componente medular durante el ciclo de desarrollo de un producto. La gestión de requisitos tiene dos subprocesos fundamentales: la ingeniería de requisitos que se encarga de definir todas las actividades involucradas en el descubrimiento, documentación y organización de los requisitos para un producto de software determinado y la administración de requisitos que realiza el control de los mismos. En los proyectos del Centro de Tecnologías para la Formación (FORTES) no se realiza una eficiente administración de requisitos, debido a que dicha actividad actualmente se registra solo a nivel de documentos, sin tener un seguimiento constante de los requisitos del sistema a lo largo del ciclo de vida, ni el control de los cambios que se realizan en los mismos. Tal situación surge, porque no existe una herramienta automatizada que se adapte a las características específicas que presenta el centro.

El resultado final de esta investigación es una aplicación Web que cubre las necesidades del proceso de administración de requisitos en los proyectos del centro FORTES, desarrollada con tecnologías y herramientas libres; ayudando a garantizar la calidad del proceso de desarrollo de software, así como los productos resultantes de este.

Palabras clave: administración de requisitos, aplicación Web, FORTES, gestión de requisitos, ingeniería de requisitos, requisito.

Tabla de contenido.

Introducción.

En la industria del software, el proceso de creación es un elemento complejo. Hoy en día la ausencia de un proceso de desarrollo o la gestión no eficiente de él, constituye la principal causa de fracaso de un proyecto, atentando contra los objetivos previstos. Un proceso de desarrollo de software define quién está haciendo qué, cuándo y cómo. Además, para funcionar de manera efectiva, concentra su esquema de trabajo definiendo un conjunto de técnicas para las diferentes áreas del conocimiento, encaminadas a minimizar el riesgo de producción.

La ingeniería de software constituye una de las vertientes fundamentales que contiene el proceso de desarrollo, se encuentra encaminada principalmente a facilitar el control del proceso y a aumentar la productividad y el trabajo. Además se vincula en la planificación del proyecto y estimación de los costos. Dentro de este marco se gestionan los requisitos de un software, que ayudan a cumplir la finalidad de los procesos que lo engloban; mediante ellos y los cambios que puedan requerir, se puede estimar la complejidad, tiempo de desarrollo real y costos finales que pueda tener un proyecto en un momento determinado.

El proceso de gestión de requisitos (GR) se considera un proceso crítico dentro de la gestión de un proyecto y se divide para su organización en dos subprocesos: ingeniería de requisitos y la administración de los mismos. El proceso de ingeniería se encarga principalmente de la identificación de los requisitos, y el de administración se encarga del registro y control de cambios sobre ellos. Este último constituye un proceso de importancia, porque en dependencia de la variación en que incurran los requisitos de un sistema, así variarán todos los elementos que dependan de ellos, dígase estimaciones de costo, tiempo y calidad final de un software.

La Universidad de las Ciencias Informáticas (UCI) presenta un carácter innovador, que mantiene el vínculo entre docencia, investigación y producción. Aunque ha sufrido ajustes sobre sus esquemas productivos, siempre ha tenido presente la definición de sus procesos de desarrollo de software enfocados a la mejora continua.

En la UCI la producción de software está basada en una infraestructura productiva compuesta por 13 centros de desarrollo de software pertenecientes a las diferentes facultades de la sede central, cuenta

también con centros de desarrollo regionales, una vicerrectoría de producción y la empresa comercializadora ALBET.

El proceso de reestructuración de la vida productiva en la universidad ha sido enfocado por áreas del conocimiento o líneas temáticas entre las cuales se encuentran: identificación y seguridad, salud y bioinformática, educación, informática industrial, telecomunicaciones y señales, gestión empresarial y banca, además del turismo; permitiendo la especialización de las diferentes zonas de desarrollo de software. Cada una de las líneas temáticas mencionadas tiene definidos los esquemas de trabajos para las áreas del conocimiento dentro del proceso de desarrollo de software, prestando singular atención a la gestión de requisitos.

El Centro de Tecnologías para la Formación (FORTES) perteneciente a la Facultad 4, tiene como línea temática el desarrollo de soluciones para el sector de la educación. Presenta como misión desarrollar tecnologías que permitan ofrecer servicios y productos para la implementación de soluciones de formación aplicando las tecnologías de la información y las comunicaciones, a todo tipo de instituciones con diferentes modelos de formación y condiciones tecnológicas.

FORTES encamina sus esfuerzos a garantizar la calidad de las soluciones y la formación de los recursos humanos a partir de investigaciones que combinen los elementos pedagógicos y tecnológicos más avanzados, integrando así los procesos de formación, producción e investigación. En dicho centro radican varios proyectos productivos, donde actualmente no se realiza de forma eficiente la administración de requisitos (AR).

Tal situación se debe a que los requisitos tanto del cliente como del producto y los elementos que se derivan de ellos, se registran solamente a nivel de plantillas o documentos digitales; definidos según el expediente que utilice el proyecto. El control de cambios sobre estos elementos se realiza de manera individual a través del rol que corresponda según el flujo de trabajo al que pertenezca ese artefacto. Estos cambios se realizan sin evaluar el desarrollo e impacto evolutivo que pudiera tener sobre el producto final. Provocando además, que no estén disponibles en todo momento para el equipo de proyecto durante el proceso de producción; lo que trae consigo retrasos en el tiempo, aumento de costos y planificaciones irreales.

De manera colateral a esta eventualidad, el actual estado de los proyectos del centro FORTES no permite que cada rol actualice la trazabilidad entre los elementos de su fase y la anterior por lo que no se mantiene una dependencia continua durante el desarrollo.

Aparejada a la situación descrita, la UCI está enfocada en un proceso de migración hacia el uso de tecnologías libres y FORTES se encuentra inmerso en tales políticas, alineando sus procesos de producción con los definidos centralmente.

Como resultado de lo expuesto anteriormente se define como **problema investigativo**: ¿Cómo apoyar la administración de requisitos en los proyectos del centro FORTES, alineado con las políticas de migración de la Universidad de las Ciencias Informáticas?

La investigación tiene como **objeto de estudio** el desarrollo de herramientas de administración de requisitos para proyectos de desarrollo de software.

El **objetivo general** de la presente investigación es, desarrollar una herramienta de administración de requisitos para los proyectos del centro FORTES.

En correspondencia con el objeto de estudio y el objetivo planteado, el **campo de acción** queda enmarcado en las herramientas de administración de requisitos para proyectos del centro FORTES.

Del objetivo general se desglosan los siguientes **objetivos específicos:**

1. Estudiar el estado del arte relacionado con las herramientas de AR para definir posición del investigador.
2. Realizar el análisis, diseño e implementación de la herramienta de AR para los proyectos del centro FORTES.
3. Validar la correcta implementación de los requisitos mediante pruebas al sistema.

De acuerdo con el problema investigativo planteado, la **idea a defender** es la siguiente: con la automatización de la AR para los proyectos del centro FORTES, se permitirá organizar y documentar los requisitos del sistema, así como controlar de manera automatizada los cambios que se realicen sobre los mismos.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas investigativas:**

1. Análisis del estado del arte referente a teorías y sistemas similares dedicados a la AR.
2. Definición de las funcionalidades con las que debe contar la aplicación a implementar.
3. Realización del análisis y diseño del sistema.
4. Implementación de las funcionalidades seleccionadas para el sistema.
5. Validación del producto para comprobar su correcto funcionamiento.

Para la realización de la investigación se utilizarán varios **métodos científicos** en la búsqueda y procesamiento de la información**,** como son:

A nivel teórico:

- **Análisis histórico–lógico:** este método se empleó en el estudio de las tecnologías que existen actualmente, con el fin de seleccionar la más adecuada a utilizar en la investigación, de acuerdo con las características propias del sistema a desarrollar.

- **Análisis-síntesis:** este método fue usado para la caracterización del subproceso, así como para el levantamiento de los requisitos de la herramienta a proponer, señalándose principalmente que debe permitir gestionar requisitos, tipos de ellos, atributos, así como mostrar vistas de trazabilidad.

- **Modelación:** se utilizó este método para el modelado de los diagramas de la vista interna del sistema, permitiendo diseñar el mismo para su posterior implementación.

A nivel empírico:

- **Observación:** con este método se estudiaron las diferentes herramientas de AR, para observar su funcionamiento y la manera en que organiza su información, ayudando a esclarecer el objeto de estudio, así como la futura herramienta que se propone obtener.

La estructura capitular de la presente investigación está definida de la siguiente forma:

Capítulo 1 Fundamentación teórica.

Enfocado en el basamento teórico de la AR, incluyendo soluciones informáticas similares en el mercado, así como las herramientas y tecnologías que se emplean en el desarrollo de la investigación.

Capítulo 2 Análisis, diseño e implementación del sistema.

En este capítulo se realiza el levantamiento de las funcionalidades que debe cumplir la aplicación y se generan los artefactos resultantes de desarrollar los flujos de trabajo de especificación de requisitos, análisis, diseño e implementación.

Capítulo 3 Pruebas a la solución.

En este capítulo se diseñan y ejecutan los casos de prueba, se corrigen las no conformidades encontradas, para asegurar la calidad y el correcto uso de la aplicación.

1 *Capítulo.*

Fundamentación teórica.

1.1 Introducción.

El proceso de gestión de requisitos (GR) enmarca dos grandes subprocesos: la ingeniería y la administración de requisitos (AR), este último tiene gran importancia dentro del ciclo de desarrollo de software. Esclarecer el marco teórico durante la investigación es fundamental, por tal motivo se definirán conceptos y características relacionadas con el objeto de estudio; así como la caracterización de las herramientas automatizadas para la AR y selección de las tecnologías necesarias para el desarrollo de la investigación.

1.2 Proceso de desarrollo de software.

Un proceso de desarrollo de software tiene como objetivo la producción exitosa de un producto informático que contempla los requisitos de los clientes. Se caracteriza por ser intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas. El proceso de desarrollo de software no es único, por lo que no existe un proceso de software universal que sea efectivo para todos los contextos de los proyectos de desarrollo. Debido a esta diversidad, es difícil automatizarlo (1).

Un proceso de desarrollo de software se caracteriza por (1):

- **Un marco común del proceso**, definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos de software, con independencia del tamaño o complejidad.
- **Un conjunto de tareas**, cada uno es una colección de tareas de ingeniería del software, hitos de proyectos, entregas y productos de trabajo del software, y puntos de garantía de calidad, que permiten que las actividades y el marco de trabajo se adapten a las características del proyecto de software y a los requisitos del equipo del proyecto.

- **Las actividades de protección** tales como, garantía de calidad del software, gestión de configuración del software y medición, abarcan el modelo del proceso. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso.

Existe un conjunto de actividades fundamentales que se encuentran presentes en este proceso (2):

- **Especificación de software:** se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.
- **Diseño e implementación:** se diseña y construye el software de acuerdo con la especificación.
- **Validación:** el software debe validarse, para asegurar que cumpla con lo que quiere el cliente.
- **Evolución:** el software debe evolucionar, para adaptarse a las necesidades del cliente.

Los requisitos son la pieza fundamental en un proceso de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que constituyen uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo. Además, la especificación de requisitos es la base que permite verificar si se alcanzaron o no los objetivos de los clientes del sistema, comprobándose así, el cumplimiento de las metas trazadas.

1.3 Definición general de requisito.

Un requisito es una descripción de una condición o capacidad que debe cumplir un sistema, ya sea derivada de una necesidad de usuario identificada, o bien, estipulada en un contrato, estándar, especificación u otro documento formalmente impuesto al inicio del proceso (3).

Existen varias definiciones de requisito a nivel mundial, entre las cuales se pueden citar las siguientes:

- Condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo (4).
- Condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta (4).
- Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste (2).

El autor de la presente investigación teniendo en cuenta las definiciones citadas anteriormente considera que un requisito puede definirse como: un elemento necesario dentro de un sistema, que puede representar una capacidad, una característica o un factor de calidad, de tal forma que le sea beneficioso el producto a los clientes y a los usuarios finales.

1.3.1 Características de los requisitos.

Los requisitos juegan un papel crucial en la vida del desarrollo de un software, de ahí que necesiten cumplir con algunos aspectos o variables que configuran su estado e identidad. Para que se encuentren formulados de manera correcta, deben satisfacer las siguientes características (1):

- **Necesario:** cuando su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.
- **Conciso:** cuando es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- **Completo:** cuando no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- **Consistente:** si no es contradictorio con otro requisito.
- **No ambiguo:** cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.
- **Verificable:** cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas.

1.4 Gestión de requisitos.

La GR es un conjunto de actividades que ayuda al equipo de proyecto a identificar, controlar y rastrear tanto a los requisitos del sistema, como a los cambios sobre ellos en cualquier momento durante el desarrollo del proyecto.

La GR es una parte vital en el desarrollo de proyectos de software ya que define el propósito, la dirección y el tamaño del proyecto, condicionando así el éxito global del mismo. Mantiene la comunicación iterativa con el cliente, con el fin de definir y registrar adecuadamente qué se espera del proyecto. Además, esta actividad asegura la consistencia entre los requisitos y el sistema construido o en construcción. Se

caracteriza por consumir grandes cantidades de tiempo y esfuerzo; abarcando así todo el ciclo de vida del producto (2).

1.4.1 Administración de requisitos.

La AR es el proceso que establece las capacidades básicas de un proyecto. Es efectiva para mejorar los flujos de trabajo, desde el diseño de software hasta la estimación de costo. Es difícil que requisitos adicionales no sean descubiertos durante el ciclo de vida en un proyecto, y es así que la investigación continua y la documentación de requisitos es esencial para la administración efectiva de estos (5).

La administración de requisitos implica (2)**:**

- Definir procedimientos que establezcan los pasos y los análisis que se realizarán antes de aceptar los cambios propuestos.
- Cambiar los atributos de los requisitos afectados.
- Mantener la trazabilidad hacia atrás, hacia delante y entre requisitos.
- Controlar las versiones del documento de requisitos.

Los beneficios que aporta son (5)**:**

- Productos que satisfacen las necesidades del cliente.
- Cumplimiento de objetivos de ingresos y rendimiento.
- Aprovechamiento de las oportunidades de mercado.
- Satisfacción del cliente.
- Estandarización y reutilización.
- Mejoramiento de la productividad y la calidad.
- Garantía del cumplimiento de normativas.
- Eliminación de costosos problemas en fases tardías del ciclo de vida.
- Aprovechamiento de los conocimientos y las inversiones en el software existente.
- Minimización de riesgos.

La trazabilidad es una de las actividades fundamentales en la AR, implica que un requisito debe ser rastreable desde que se define y durante todo el desarrollo del software, garantizando una adecuada administración del cambio con el fin de evaluar el impacto en el resto del sistema. Además, contribuye al

seguimiento de un requisito hacia adelante o hacia atrás; reflejando los efectos que puede tener, la inclusión o exclusión de un nuevo requisito. La trazabilidad también ayuda en la integración de nuevos módulos reflejando las implicaciones que puedan tener en el sistema.

La trazabilidad puede ayudar en algunos casos a conocer con anterioridad, cuáles serían las consecuencias de cambiar o eliminar un requisito en el sistema, lo que en estrategias carentes de trazabilidad, no se podía visualizar tan fácilmente.

Después de identificados los requisitos se desarrollan las tablas de rastreabilidad, cada una de ellas relaciona los requisitos con uno o más aspectos del sistema o de su ambiente. Entre muchas tablas posibles están las siguientes (1).

- **Tabla de rastreabilidad de las características:** muestra la manera en que los requisitos se relacionan con las características del sistema o producto observables para el cliente.
- **Tabla de rastreabilidad de dependencia:** indica la forma en que los requisitos están relacionados entre sí.
- **Tabla de rastreabilidad del subsistema:** establece categorías entre los requisitos de acuerdo con el (los) subsistema (s) que gobierna (n).
- **Tabla de rastreabilidad de la interfaz:** muestra la forma en que los requisitos se relacionan con las interfaces internas y externas del sistema.

1.5 Herramientas automatizadas para la administración de requisitos.

El uso de herramientas para la AR mejora la productividad y la calidad en el desarrollo de un proyecto de software. Un inadecuado entendimiento de las necesidades de los usuarios hace que la mayoría de los proyectos fracasen, por lo que se hace necesario crear un hilo continuo entre requisitos, diseño e implementación.

Las herramientas automatizadas proporcionan algunas ventajas como (6)**:**

- Mayor control de proyectos complejos.
- Reducen costos y retrasos en la liberación de un proyecto.
- Mejoran la comunicación en los equipos de trabajo.
- Ayudan a determinar la complejidad del proyecto y esfuerzos necesarios.

- Suministran una trazabilidad completa de la especificación.
- Reducen las no-conformidades del sistema.

Existe una gran cantidad de herramientas para la administración de requisitos en el mercado y muchas de ellas están ampliamente difundidas, la mayoría tienen un conjunto de características básicas que se asocian a herramientas de este tipo.

En general, todas se basan en sistemas centralizados de gestión de bases de datos para almacenar la información correspondiente a los requisitos, que suele consistir en párrafos de texto libre con una serie de atributos predefinidos y a los que la mayoría de las herramientas permiten asociar nuevos tipos de atributos por parte del usuario. Todas las herramientas asumen que la estructura de los requisitos es jerárquica, de forma que un requisito puede estar formado o tener asociados otros requisitos de nivel inferior. Otras de las características comunes de la mayoría de las herramientas es la posibilidad de realizar consultas sobre los requisitos en función de determinados valores de sus atributos.

1.5.1 Característica de la herramienta de AR para el centro FORTES.

El centro FORTES se encuentra actualmente en pleno proceso de migración hacia software libre. La aplicación final no debe estar ajena a dicha directriz, por tal razón debe ser desarrollada con herramientas y tecnologías libres. FORTES necesita una herramienta para la administración de requisitos que presente alta disponibilidad, que permita a los equipos de proyectos de manera íntegra consultar y editar información a cualquier hora del día.

La aplicación no debe interferir con el proceso normal de desarrollo de un producto en específico. Debe permitir su instalación independiente de la configuración de software o hardware que pueda tener una computadora; donde los usuarios finales no necesiten tener conocimiento previo sobre una tecnología o lenguaje en específico para poder configurarla durante su instalación.

Además, de las características externas, también la herramienta debe estar orientada a cumplir con las funcionalidades específicas de las aplicaciones de su tipo: posibilitar gestionar atributos, requisitos y sus clasificaciones, así como mantener la trazabilidad de los mismos y presentar diferentes reportes de información sobre ellos.

1.5.2 Características de las herramientas consultadas.

Teniendo en cuenta las características genéricas que deben tener las herramientas para la AR en el centro FORTES, se realizó un estudio sobre las siguientes.

RequisitePro.

Es la herramienta que ofrece la empresa IBM Rational Software para tener un mayor control sobre los requisitos existentes o nuevos para el usuario, que surjan durante el ciclo de vida del proyecto de software.

En RequisitePro los requisitos se encuentran documentados bajo un esquema organizado de documentos. Estos esquemas cumplen completamente con los estándares requeridos por algunas de las instituciones más reconocidas a nivel mundial en el desarrollo de software, tales como: el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), la Organización Internacional de Estandarización (ISO), el Modelo de Capacidad de Madurez (CMM) y el Proceso Unificado de Desarrollo de Software (RUP). Además, proporciona a los equipos la posibilidad de comprender el impacto de los cambios. Garantiza que todos los componentes del equipo estén informados de los requisitos más actuales para asegurar la coherencia (7).

El RequisitePro a pesar de brindar diferentes funcionalidades que cubren todo el campo de la AR y contar con un soporte empresarial, posee varios inconvenientes que propician la decisión de descartarla como herramienta a utilizar. Estos problemas se exponen a continuación (7):

- Solo funciona en el sistema operativo Microsoft Windows pero en el centro FORTES se necesita una herramienta multiplataforma.
- No publica su código fuente, haciendo imposible que el centro FORTES pueda ajustarla a sus necesidades.
- Consta de tres tipos de licencias donde el precio es elevado: una licencia de usuario autorizado, una licencia de término fijo de usuario autorizado (FTL) y una licencia flotante.

Gestión de requisitos (REM, por sus siglas en inglés).

Es una herramienta experimental desarrollada por la Universidad de Sevilla. Es gratuita, muy simple y sencilla de utilizar, por tanto, las funcionalidades y características de las que dispone son

muy básicas. REM permite crear únicamente documentos normalizados, donde se incluyen los requisitos necesarios para el desarrollo de un sistema de información. Dispone de una estructura muy rígida y unos servicios ofrecidos muy limitados que no permiten adaptarse a cambios, ni necesidades (8).

REM a pesar de ser una herramienta de código libre, solamente funciona sobre el sistema operativo Microsoft Windows. Además no posee actualizaciones después de haber sido creada, por lo que cuenta con pocas funcionalidades y no contempla el uso de la trazabilidad, actividad que es fundamental en la AR. Por estos inconvenientes fue descartada, ya que no cumple con la migración total a software libre que pretende la UCI.

Herramienta de gestión de requisitos de código abierto (OSRMT, por sus siglas en inglés).

Es una herramienta de software libre y multiplataforma. Fue desarrollada en el lenguaje de programación Java. Permite la descripción de los requisitos, además de los documentos relacionados con la ingeniería de requisitos. También comprende las distintas matrices de trazabilidad, funcionalidades y casos de uso (CU) (9).

Esta aplicación presenta varios inconvenientes referente a la herramienta que se pretende en el centro FORTES, uno de ellos es, que se compone para realizar la AR de 2 herramientas:

- La primera es una aplicación de escritorio que está desarrollada sobre Java y es la encargada de la administración de requisitos. Si se desea utilizar en el centro FORTES, se debe instalar esta aplicación en cada una de las computadoras de los integrantes del proyecto junto con la máquina virtual de Java que requiere para su funcionamiento. Esto trae consigo un desaprovechamiento de los recursos y una dependencia de la tecnología Java.
- La segunda aplicación es una extensión Web que es de carácter informativo, y solo permite la visualización de la información. No permite al usuario intercambiar datos, propiciando que el trabajo se torne engorroso.

Teniendo en cuenta lo anteriormente planteado se determina que esta herramienta no es la más adecuada, debido a que tiene dependencias de tecnologías que no son libres, y el despliegue de

esta traería consigo un gasto de recursos de procesamiento y almacenamiento en las computadoras de los integrantes del proyecto.

A continuación se muestra una tabla que relaciona las características de las herramientas consultadas con las que se pretende que cumpla la herramienta AR en el centro FORTES.

Características	RequisitePro	REM	OSRMT
Manejo de varios proyectos.	X		X
Manejo de trazabilidad.	X		X
Incluye interfaz Web.	X	X	X
Software libre.		X	X
Multiplataforma.			X
Nivel de seguridad.	Alta	Baja	Alta
Fácil instalación y configuración Web.	X		
Sin costo de licencia.		X	X
Independencia de tecnologías propietarias.			

Tabla 1: "Matriz de comparación entre las herramientas".

De acuerdo con las características de las herramientas analizadas, ninguna de ellas se adapta totalmente a las necesidades existentes en el centro FORTES. Por lo que se decidió como propuesta de solución, desarrollar una herramienta de código abierto, moldeable a la situación de dicho centro, que cumpla con las principales características de las herramientas más utilizadas a nivel mundial.

En principio la herramienta debe proporcionar monitoreo de varios proyectos, trazabilidad entre los diferentes elementos definidos, facilidad tanto en el control de cambios como en la instalación y configuración de la herramienta. Además, debe ser una herramienta multiplataforma, que pueda ser mantenida por el centro, permitiendo a los proyectos productivos el ahorro por términos de licencia, obteniendo así mayores ganancias y mejoras en la calidad del producto de software final.

1.6 Herramientas y tecnologías.

Durante el desarrollo de un software la selección de las herramientas y tecnologías, es un elemento importante porque constituye la base para cualquier fase del proyecto.

1.6.1 Metodología de desarrollo de software.

Una metodología de desarrollo de software define la distribución de los recursos dentro de un proyecto y cómo alcanzar un objetivo específico. Proporciona normas para el desarrollo eficiente del software con calidad, al captar las mejores prácticas que el estado actual de la tecnología permite (10).

Características más comunes de las metodologías:

- Uso de técnicas gráficas.
- Diccionario de datos.
- Documentación.
- Análisis del riesgo.
- Análisis estructurado.
- Análisis orientado a objetos.

Para la realización del presente trabajo de diploma se estudiaron diferentes metodologías con el fin de seleccionar la más adecuada para un equipo de desarrollo integrado por una sola persona y la necesidad de desarrollar un software en el menor tiempo posible.

Proceso Unificado de Desarrollo de Software (RUP, por sus siglas en inglés).

Está centrado en la arquitectura, dirigido por CU, y es iterativo e incremental. Los casos de uso representan los requisitos funcionales del sistema, guían el diseño, la implementación y las pruebas. Constituyen un elemento integrador y una guía de trabajo. Está constituido por cuatro fases: inicio, elaboración, construcción y transición.

Durante la fase de inicio se realizan iteraciones, donde se enfatiza en las actividades de modelado de los flujos de trabajos negocio y requisitos. En la fase de elaboración, las iteraciones están orientadas al desarrollo de la línea base de la arquitectura, abarcando más flujos de trabajo: requerimiento, modelo de negocio (refinado), análisis, diseño y una parte de implementación. En la fase de construcción, se lleva a cabo la implementación del producto por medio de un conjunto de iteraciones. Por último, en la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios. Cada fase concluye con un hito bien definido (11).

Programación Extrema (XP, por sus siglas en inglés).

Es la primera metodología ágil y la que proveyó de conciencia al movimiento actual de metodologías ágiles. Se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los programadores, y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, al igual que donde existe un alto riesgo técnico. El ciclo ideal de XP consiste de 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto (12).

Se diferencia de las demás metodologías en que tiene un alto nivel de disciplina de las personas que participan en el proyecto. Constituye la práctica de la programación, usualmente orientada a objetos y con un fuerte uso de patrones de diseño (12)**.**

Proceso Unificado Abierto (Open UP, por su nombre en inglés).

Constituye un proceso unificado de desarrollo de corta duración, aplicado de manera iterativa e incremental dentro de un ciclo de vida estructurado en tres capas. Open UP adopta una pragmática y ágil filosofía centrada en el proceso colaborativo de desarrollo de software que puede ser aplicada a una gran variedad de proyectos en diferentes direcciones (13).

El Open UP es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. Por lo tanto, no provee lineamientos para todos los elementos que se manejan en un proyecto, pero tiene los componentes básicos que pueden servir de base a procesos específicos. La mayoría de los elementos de Open UP están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances (13).

En un proyecto organizado sobre esta metodología, los esfuerzos personales se convierten en micro incrementos. Estos proveen un ciclo de retroalimentación relativamente corto que permite flexibilidad y mejor adaptación a las decisiones tomadas dentro de cada iteración. Esta metodología divide el proyecto en iteraciones que son planeadas sobre intervalos de tiempo definidos en semanas. Dichas iteraciones se centran dando cumplimiento a los objetivos definidos

previamente en el plan para cada una por parte del equipo de desarrollo, donde cada ciclo iterativo debe concebir como resultado un demo o un ejecutable con funcionalidades específicas (13).

Open UP en cada iteración del ciclo de vida, incrementa progresivamente los objetivos de las iteraciones anteriores, añadiendo nuevas funcionalidades a las versiones establecidas del software que se tiene hasta el momento. Dentro del ciclo de vida tiene 4 fases: inicio, elaboración, construcción y transición (13).

Scrum.

Define un proceso empírico, iterativo e incremental de desarrollo que intenta obtener ventajas respecto a los procesos definidos (cascada, espiral, prototipos, etc.) mediante la aceptación de los riesgos del desarrollo. Método que enfatiza prácticas y valores de gestión de proyectos por sobre las demás disciplinas del desarrollo. Se definen algunos roles y artefactos que contribuyen a la obtención de un proceso que maximiza la retroalimentación para mitigar cualquier riesgo que pueda presentarse (14).

Scrum se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad. Su intención es la de maximizar la retroalimentación sobre el desarrollo pudiendo corregir problemas y mitigar riesgos de forma temprana. No propone el uso de ninguna práctica de desarrollo en particular; sin embargo, es habitual emplearlo como un framework ágil de administración de proyectos. Suministra un marco de gestión basado en patrones organizacionales (14).

De las metodologías mencionadas anteriormente fue seleccionada para la investigación Open UP, en función del número de artefactos que genera. Además del enfoque iterativo e incremental, contiene la cantidad mínima y suficiente de artefactos para un desarrollo de software ágil enfocado fundamentalmente en la implementación del producto final, pero sin excluir los activos significativos para documentar el sistema.

Además, se asume esta metodología para el desarrollo de la investigación porque se han obtenido resultados relevantes en cuanto a la agilización de producción de software para equipos de desarrollo. La

propuesta a desarrollar no constituye un software de alta complejidad ni está determinado por procesos críticos.

1.6.2 Lenguaje unificado de modelado (UML).

Es un lenguaje de modelado para la especificación, visualización, construcción y documentación de los artefactos de un proceso de desarrollo de software. Fue originalmente concebido por la industria de la tecnología y sistemas de información. Constituye un lenguaje para especificar y no para describir métodos o procesos. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar. UML cuenta con varios tipos de diagramas que muestran diferentes aspectos de las entidades representadas, pueden ser categorizados en: diagramas de estructura, de comportamiento y de iteración (15).

1.6.3 Herramienta de modelado.

Las herramientas que sirven de apoyo a la ingeniería de software, son conocidas como herramientas CASE (ingeniería de software asistida por computadora), estas apoyan las actividades que tienen lugar a lo largo de todo el ciclo de vida del proyecto, incluyendo actividades como la gestión de proyectos y la estimación.

Visual Paradigm.

Es una herramienta profesional que soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Proporciona abundantes tutoriales, demostraciones interactivas y proyectos UML (16).

Visual Paradigm posee las siguientes características (16):

- Crea y visualiza diagramas UML.
- Genera código C++, CORBA, Java, esquemas XML a partir de los diagramas.
- Genera documentación.
- Realiza ingeniería inversa, código a modelo y código a diagrama.
- Genera bases de datos.
- Multiplataforma.

Rational Rose.

Herramienta de modelación visual que cubre todo el ciclo de vida de desarrollo de un software. Como todos los demás productos utilizados para la modelación de software proporciona un lenguaje de modelado para el equipo, que facilita la creación de software de mayor calidad y de forma más rápida. Una de las características principales de Rose es que utiliza como lenguaje de modelado a UML, permitiendo así diseñar base de datos y requerimientos de aplicación a través de diseños lógicos y físicos. Además, los arquitectos de software y desarrolladores pueden visualizar el sistema completo utilizando un lenguaje común y los diseñadores modelar componentes e interfaces de forma individual, para luego unirlos con otros componentes del proyecto (17).

Rational Rose al constituir una herramienta con plataforma independiente, ayuda a la comunicación entre los miembros de los equipos, así como a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Uno de sus aspectos fundamentales es que incluye un conjunto de herramientas de ingeniería inversa y generación de código que facilitan la implementación del producto final (17).

Realizada la valoración de las herramientas CASE se decide utilizar Visual Paradigm versión 6.4 en la investigación, por su compatibilidad con el sistema operativo Linux, donde será modelada la solución. A diferencia del Rational Rose que es soportado solamente por sistemas operativos como: Microsoft Windows 2000, NT, XP y Windows 7.

1.6.4 Sistema de gestión de contenidos.

Un sistema de gestión de contenidos (CMS, por sus siglas en inglés) permite la creación y administración de contenidos, principalmente en páginas Web, posibilitando manejar de forma independiente el contenido y el diseño, garantizando así, modificar el diseño del sitio sin tener que cambiar su contenido. Un CMS provee una infraestructura de autenticación, autorización y permisos de fácil implementación, ahorrando tiempo de desarrollo y permitiendo el control de las publicaciones de varios editores o autores (18).

A continuación se hace referencia a dos de los CMS más utilizados a nivel mundial y en la Universidad de las Ciencias Informáticas:

Joomla.

Es un CMS con las características y la funcionalidad que encuentra en la mayoría de las aplicaciones de gama alta (19).

Las principales funcionalidades son (19):

- Gestión del contenido del sitio basado completamente en una base de datos.
- Todas las secciones de noticias, productos o servicios se pueden editar y gestionar.
- Las secciones de temas pueden ampliarse mediante aportes de autores.
- Administra los usuarios con varios tipos de cuenta de usuario disponibles.
- Las características de etiquetado de los contenidos permiten un acceso flexible para cada tipo de usuario.
- Modifica completamente el sitio y la administración con el uso de plantillas simples.
- Diseños completamente personalizables, incluyendo los menúes izquierdo, derecho y central.
- Permite subir imágenes a la galería del servidor vía navegador para su uso en cualquier lugar del sitio.
- Realiza una búsqueda con texto completo a través de todas las áreas de contenido.

Drupal.

Es un sistema de gestión de contenido modular multipropósito y muy configurable que permite publicar artículos, imágenes, u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos (20).

Es un programa libre, con licencia pública general GNU/GPL (por sus siglas en inglés), desarrollado en PHP y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la Web, y un énfasis especial en la usabilidad y consistencia de todo el sistema (20).

El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante a su flexibilidad y adaptabilidad, así como a la gran cantidad de módulos adicionales disponibles, hace que sea adecuado para realizar diferentes tipos de sitios Web (20).

Entre sus características principales se encuentra (20):

- Control absoluto del flujo de trabajo.
- Control detallado de permisos y roles de usuario.
- Control sobre la visualización.
- Comunidad de desarrollo organizada, estricta y muy profesional.
- Respeto de estándares, usabilidad y accesibilidad.

Con las características anteriores se realizó una comparación, la cual arrojó como resultado que Drupal 6.20 es el CMS seleccionado para realizar la propuesta de la investigación, por los siguientes aspectos:

Permisos para usuarios:

Joomla sigue teniendo limitados los permisos predefinidos a usuarios. En Joomla sólo hay tres tipos de acceso, público, para registrados y para administradores. Además, hay sólo 5 tipos de usuarios, que se diferencian principalmente en las posibilidades de editar noticias: propias o de otros. En Drupal el sistema es configurable al gusto, permitiendo editar desde los permisos por módulo, hasta por las diferentes opciones del módulo. Pueden crearse grupos de usuarios predefinidos, y asignar a cada grupo los permisos deseados. (21).

Categorías y contenido:

Joomla permite nada más dos niveles de clasificación: secciones y categorías, mientras que Drupal permite crear tipos de contenido específicos y configurar todas las categorías para cada tipo de contenido diferente: en árbol, por jerarquía, entre otras.

Base de datos:

Joomla solo soporta MySQL, mientras que Drupal soporta además a PostgreSQL, cumpliendo así con las políticas de migración de la UCI.

1.6.5 Lenguaje de programación.

Los lenguajes de programación Web se dividen en dos grupos fundamentales, los del lado del cliente y los del lado del servidor. Los lenguajes del lado del servidor son los encargados del acceso a bases de datos y del tratamiento de la información; y los lenguajes del lado del cliente se encargan de aportar dinamismo a la aplicación en los navegadores.

Para realizar la aplicación se definió como CMS Drupal, incluyendo consigo la obligación de trabajar con los lenguajes definidos e implementados en sus módulos, para su compatibilidad y mejor funcionamiento, estos son:

PHP.

Es un lenguaje interpretado de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Al nivel más básico, PHP puede procesar la información de formularios, generar páginas con contenidos dinámicos, o mandar y recibir cookies. También soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. Puede ser desplegado en la mayoría de los servidores Web y en casi todos los sistemas operativos y plataformas sin costo alguno.

Este lenguaje tiene muchas virtudes que lo han convertido en la opción de muchos programadores debido a que (22):

- Posee una gran cantidad de funciones desarrolladas.
- Capacidad de conexión con la mayoría de los manejadores de bases de datos que se utilizan en la actualidad destacando su conectividad con MySQL y PostgreSQL.
- Permite las técnicas de programación orientada a objetos.
- El código fuente es abierto y gratuito.
- Existe una gran comunidad de desarrolladores, lo que garantiza que los fallos de funcionamiento se encuentren y reparen rápidamente.

JavaScript.

Lenguaje de scripts desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML. Se utiliza embebido en el código HTML y XHTML, entre las etiquetas <script> y </script>. Dentro de sus características más importantes se pueden encontrar (23):

- Es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente.

- Es un lenguaje orientado a eventos. Cuando un usuario presiona el cursor sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante este lenguaje se pueden desarrollar scripts que ejecuten acciones en respuesta a estos eventos.

- Es un lenguaje orientado a objetos. El modelo de objetos de JavaScript está reducido y simplificado, pero incluye los elementos necesarios para que los scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador.

HTML.

Es el lenguaje de marcado predominante para la elaboración de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos como imágenes. Se escribe en forma de "etiquetas", rodeadas por corchetes angulares. También puede describir, hasta cierto punto, la apariencia de un documento, y puede incluir un script como por ejemplo Java Script, el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

Es un lenguaje extensible, se le pueden añadir características, etiquetas y funciones adicionales para el diseño de páginas Web, generando un producto vistoso, rápido y sencillo. La mayoría de las etiquetas del lenguaje son semánticas. La interpretación de las etiquetas es realizada por el navegador Web (24).

1.6.6 Entorno de desarrollo integrado.

Un entorno de desarrollo integrado (IDE, por sus siglas en inglés) es un ambiente de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Eclipse:

Este IDE es el usado por lo general para el desarrollo de aplicaciones en Java, en la que se ha especializado, también permite programar en PHP, aunque con muy pocas opciones de revisión y corrección de errores y sin completamiento de código. Es una herramienta que necesita de mucha ayuda del hardware para realizar la compilación del código fuente escrito, además de ser multiplataforma (25).

NetBeans:

NetBeans es un IDE que posee un importante número de módulos para extender, puede ser utilizado para varios lenguajes de programación. Es un producto libre y gratuito sin restricciones de uso. Además, de ser muy potente para el desarrollo de aplicaciones de escritorio y Web (26).

Características principales (26):

- Instalación y actualización simple.
- Características visuales para el desarrollo Web.
- Estable y rápido.

Después de realizado un estudio de ambos IDE para utilizar en la investigación, se seleccionó NetBeans por sus características y su integración a la perfección con los lenguajes de programación: PHP, JavaScript y HTML. Además, acelera y optimiza el proceso de edición, eliminación de errores, análisis y publicación de aplicaciones. Esta herramienta permite ahorrar tiempo, de esta manera aumenta y mejora la productividad ya que permite acceder a las diferentes clases de funciones, así como a sus variables, etiquetas y propiedades.

1.6.7 Sistema gestor de base de datos.

Un sistema gestor de bases de datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de uno de manipulación de datos y de otro de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

PostgreSQL.

PostgreSQL es un sistema de gestión de base de datos relacional orientado a objetos y libre, publicado bajo la licencia de distribución de software Berkeley (DSB). Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y apoyada por organizaciones comerciales. Dicha comunidad es denominada grupo de desarrollo global de PostgreSQL (GDGP) (27).

Características principales (27):

- Consistencia: garantiza que sólo se empieza aquello que se puede acabar, por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- Aislamiento: avala que una operación no puede afectar a otras. Esto condiciona que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
- Durabilidad: condiciona que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Altamente adaptable a las necesidades del cliente.
- Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, entre otros.
- Multiplataforma.

MySQL.

Sistema de administración de bases de datos relacionales. Está licenciado bajo la GPL de la GNU. Como gestor de bases de datos es uno de los más usados en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de una gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

En aplicaciones Web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que lo hace ideal para este tipo de aplicaciones (28).

Las principales características de este gestor de bases de datos son las siguientes (28):

- Gran portabilidad entre sistemas.
- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.
- Múltiples motores de almacenamiento, permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.

- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.
- Fácil instalación y configuración.

La investigación realizada sobre los SGBD proyecta la selección de PostgreSQL (versión 8.4), por poseer la característica de manejar grandes volúmenes de datos. PostgreSQL está caracterizado por la durabilidad, propiedad que asegura que una vez realizada una operación, ésta persistirá y no se podrá deshacer aunque falle el sistema. Es multiplataforma y compatible con el CMS Drupal 6.20. Cuenta con una documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios. Además, es muy usado en la Universidad de las Ciencias Informáticas por cumplir las políticas de migración.

1.6.8 Servidor Web.

Un servidor Web es un servicio que se ejecuta en una computadora (servidor) que entrega a otras computadoras (los clientes) informaciones que ellos requieren bajo un lenguaje común, denominado protocolo. Por lo tanto en el servidor Web es donde se almacena la información estática accedida y/o las aplicaciones que la generan.

Apache.

Está diseñado para ser un servidor Web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos; se ha adaptado siempre a una gran variedad de estos últimos a través de su diseño modular. Este diseño permite a los administradores de sitios Web elegir qué características van a ser incluidas en el servidor, seleccionando cuales módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. Es un software de libre distribución que publica su código fuente, lo que permite que cualquiera pueda modificarlo y colaborar así en su desarrollo (29).

Ventajas (29):

- Modular.
- Código abierto.
- Multiplataforma.
- Extensible.

- Presenta entre otras características mensajes de error altamente configurables.
- Apache tiene amplia aceptación en la red.

Microsoft Internet Information Server (IIS).

Es una serie de servicios para los ordenadores que funcionan con sistemas operativos de Microsoft Windows. Este servicio convierte a un ordenador en un servidor de Internet o Intranet, es decir, en las computadoras que tienen este servicio instalado se pueden publicar páginas Web tanto local como remotamente. Además, se basa en varios módulos que le aportan la capacidad para procesar distintos tipos de páginas (30).

Realizada la comparación sobre los servidores Web, el que más se ajusta a la investigación es Apache 2.0 por las siguientes razones: es altamente confiable, estable, independiente de la plataforma, con distribución gratuita y de código fuente. Además, el CMS seleccionado lo propone como uno de los servidores Web por excelencia.

1.7 Conclusiones.

Al finalizar el capítulo quedaron claramente definidos los principales aspectos del proceso de administración de requisitos. Se arribó a la conclusión que de las herramientas para la AR estudiadas, ninguna cumple con los aspectos señalados para su aplicación en el centro FORTES y en correspondencia con las líneas de producción del centro y de la universidad, fueron seleccionadas las tecnologías para el diseño e implementación del sistema.

2 *Capítulo.*

Análisis, diseño e implementación del sistema.

2.1 Introducción.

El diseño de la vista interna de un software es el insumo inicial para la implementación final del mismo y ayuda en la descripción del alcance de la propuesta de solución, permitiendo así, tener una visión general. Para diseñar una aplicación se definen, entre otros: actores, requisitos funcionales y no funcionales, CU, diagramas y arquitectura. Precisamente por la importancia que tiene esta macro actividad, este capítulo está dedicado a tal función y a la implementación del sistema.

2.2 Fase de inicio.

En la fase de inicio de la metodología seleccionada, se define el alcance y visión del proyecto. Se identifican los conceptos fundamentales en el entorno de la aplicación, se especifican los requisitos y características a implementar, entre otras actividades que generan un conjunto de artefactos críticos que se desarrollarán a continuación (31).

2.2.1 Documento visión.

Los requisitos de un software constituyen un recurso que debe permanecer disponible para todos los integrantes del proyecto en todo momento, durante el ciclo de desarrollo del mismo; donde los cambios que se realicen sobre ellos tienen total dependencia y controlarán las funcionalidades del producto final, manteniendo un entendimiento entre el cliente y el equipo de desarrollo.

En el subproceso de AR intervienen todos los integrantes del proyecto, cada uno de ellos según el rol que ocupe, será el encargado de actualizar la trazabilidad bidireccional correspondiente a los artefactos que genera en un momento dado.

En varios de los proyectos pertenecientes al centro FORTES de la Facultad 4, actualmente no se realiza de forma eficiente la AR. Dicho proceso no se realiza de forma automatizada y no se tiene un seguimiento

constante por parte de los integrantes del equipo de proyecto de los requisitos del sistema a lo largo del ciclo de vida, ni el control de los cambios que se realizan. De ahí que se necesite una herramienta que agilice tal proceso y permita que todos los integrantes de un proyecto tengan acceso a la información actualizada.

En el centro FORTES varios proyectos desarrollan sus productos en diferentes sistemas operativos y tecnologías, incluso dentro de un mismo proyecto también se puede encontrar tal situación, por lo que se hace necesaria una herramienta que pueda solventar dicha problemática. Se propone una aplicación Web porque ofrece:

- ✓ **Compatibilidad multiplataforma:** más factible para la compatibilidad multiplataforma que las aplicaciones de software descargables.
- ✓ **Actualización:** pueden ser actualizadas sin requerir que el usuario tome acciones pro-activas e interferir con sus hábitos de trabajo.
- ✓ **Inmediatez de acceso:** no necesitan ser descargadas, instaladas y configuradas en las computadoras clientes.
- ✓ **Menos requerimientos de memoria:** tienen menos demandas de memoria RAM por parte del usuario final que los programas instalados localmente.
- ✓ **Menos errores:** menos propensas a bloquearse y crear problemas técnicos debido a software o conflictos de hardware con otras aplicaciones existentes, protocolos o software personal interno.
- ✓ **Múltiples usuarios concurrentes:** puede ser utilizada por múltiples usuarios al mismo tiempo.
- ✓ **Integración:** permite tener una herramienta común para los proyectos del centro FORTES.

2.2.2 Modelo conceptual.

El modelo de dominio es una representación visual de los conceptos u objetos significativos del mundo real para un problema o área de interés, y enlaza estos objetos unos con otros. Se aplica cuando no es posible encontrar una estructura en los procesos de negocios, es decir, estos no están bien definidos y no se pueden determinar con claridad sus fronteras ni quienes se benefician de estos (32).

Conceptos principales de la aplicación:

Proyecto: conjunto de actividades interrelacionadas, con un inicio y un final definido, que utiliza recursos limitados para lograr un objetivo deseado. Entorno donde se realiza la AR.

Equipo de proyecto: conjunto de personas que laboran en un proyecto, encargadas además de realizar las diferentes actividades dentro de la AR.

Administración de requisitos: conjunto de actividades que ayudan al equipo de proyecto a controlar y rastrear los requisitos y los cambios a éstos en cualquier momento mientras se desarrolla el proyecto.

Paquete: estructura lógica que puede agrupar requisitos pertenecientes a un paquete o módulo de funcionalidades.

Tipo de requisito: define el comportamiento de los diferentes elementos en el sistema, tiene la responsabilidad principal de agrupar requisitos del mismo tipo con una referencia única.

Requisito: elemento que defina el equipo de proyecto durante el proceso de desarrollo.

Atributo: propiedades de los requisitos, que presentan una o varias formas de medición.

Vista: forma de mostrar los elementos de importancia del sistema mediante reportes. Pueden ser de trazabilidad o de atributos.

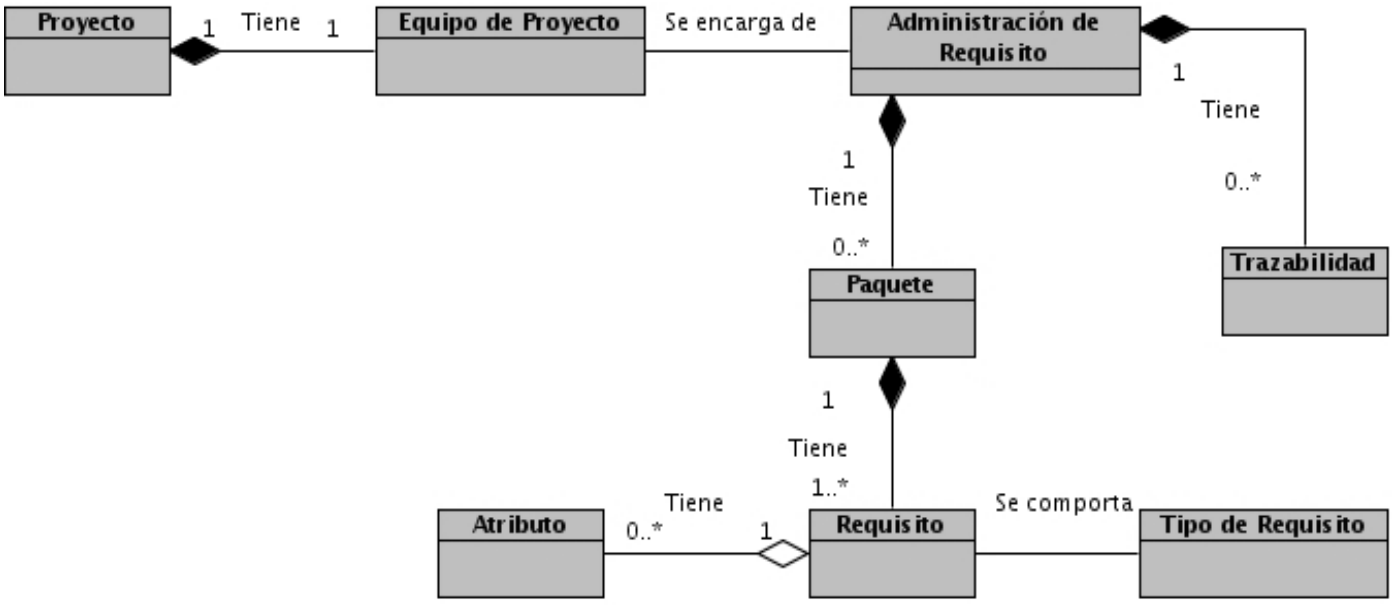

Figura 1: modelo de dominio.

2.2.3 Especificación de requisitos.

La especificación de requisitos comprende la descripción completa del comportamiento del sistema que se va a desarrollar, constituyendo una base arquitectónica que con posterioridad se agrupará en CU y describirá los procesos que realizará la futura aplicación que se propone. Además, define el funcionamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que este pueda poseer.

Requisitos funcionales.

RF1 Gestionar usuario.

RF1.1 Insertar usuario.

RF1.2 Modificar usuario.

RF1.3 Eliminar usuario.

RF1.4 Listar usuarios.

RF2 Autenticar usuario.

RF3 Gestionar proyecto.

RF3.1 Crear proyecto.

RF3.2 Modificar proyecto.

RF3.3 Eliminar proyecto.

RF3.4 Listar proyectos.

RF4 Gestionar tipo de requisito.

RF4.1 Crear tipo de requisito.

RF4.2 Modificar tipo de requisito.

RF4.3 Eliminar tipo de requisito.

RF4.4 Listar tipos de requisitos.

RF5 Gestionar atributo.

RF5.1 Crear atributo.

RF5.2 Modificar atributo.

RF5.3 Eliminar atributo.

RF5.4 Listar atributos.

RF6 Gestionar paquete.

RF6.1 Crear paquete.

RF6.2 Modificar paquete.

RF6.3 Eliminar paquete.

RF6.4 Listar paquetes.

RF7 Gestionar requisito.

RF7.1 Crear requisito.

RF7.2 Modificar requisito.

RF7.3 Eliminar requisito.

RF7.4 Listar requisitos.

RF7.5 El sistema debe permitir insertar y eliminar trazabilidad bidireccional.

RF8 Mostrar requisito.

RF8.1 Mostrar los requisitos existentes agrupados por proyectos y paquetes.

RF9 Mostrar vista.

El sistema debe permitir la existencia de dos tipos de vista, la primera para generar reporte sobre los requisitos con sus atributos y la segunda para generar reporte sobre la trazabilidad en los diferentes tipos de requisitos. No se podrá realizar cambios sobre las vistas.

RF10 Generar reporte histórico.

RF10.1 El sistema debe permitir guardar cada acción hacia un requisito que se realice.

RF10.2 El sistema debe permitir mostrar el reporte histórico de los cambios realizados.

RF11 Mostrar cambios recientes realizados sobre requisitos.

RF11.1 El sistema debe permitir mostrar los cambios recientes realizados sobre los requisitos agrupados por proyectos y paquetes.

Requisitos no funcionales.

Usabilidad.

RNFUS1 La herramienta luego de instalada debe ser compatible en los principales navegadores (Internet Explorer 8 o superior, Firefox 3.6 o superior, entre otros).

Soportabilidad.

RNFSO1 El sistema debe ser:

- De fácil instalación, configuración y puesta en marcha.
- Programado orientado a objeto.

RNFSO2 El sistema debe ejecutarse sobre un servidor Apache y utilizar PostgreSQL como gestor de base de datos.

Interfaces de usuarios.

RNFIU1 El sistema proporcionará claridad y buena organización de la información, fácil de navegar y de rápido entendimiento para los usuarios.

Implementación.

RNFIM1 El sistema se realizará utilizando el lenguaje de programación PHP.

RNFIM2 Para la modelación de los diagramas UML se empleará Visual Paradigm.

Portabilidad.

RNFPO1: El sistema debe ser multiplataforma y ejecutarse de manera óptima sin importar el sistema operativo que utilice el usuario.

Seguridad.

RNFSE1: El sistema debe garantizar que la información sea registrada, visualizada, eliminada y actualizada únicamente por la persona que posea los privilegios correspondientes.

Hardware.

RNFHW1: El servidor Web y el de base de datos deben tener un requerimiento mínimo recomendado de: procesador Dual Core, 1 Gb de RAM y disco duro 160 Gb (pero puede variar según el proyecto que utilice

la herramienta). Las computadoras clientes deben poseer un mínimo recomendado de: procesador Pentium 4 y 512 Mb de RAM.

2.2.4 Modelado de CU del sistema.

Actores del sistema.

Actor	Descripción
Usuario.	Luego de autenticarse podrá tener acceso a realizar vistas de los diferentes elementos de importancia en el sistema tales como requisitos y trazabilidad, pero sin acceso a actualizar ningún contenido.
Usuario avanzado.	Tiene la potestad para insertar, modificar y eliminar proyectos, tipos de requisitos, requisitos, atributos de requisitos y paquetes, así como todos los privilegios que presenta el usuario básico.
Administrador	Encargado de administrar el sistema en su totalidad. Además, de las acciones de los usuarios básicos y avanzados podrá insertar, modificar y eliminar los usuarios, así como generar reporte histórico de los cambios.

Tabla 2: descripción de los actores del sistema.

Casos de uso del sistema.

Los CU proporcionan uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico utilizando un lenguaje más cercano al usuario final.

Código	Nombre	Actor	Referencia
CU1	Gestionar usuario.	Administrador	RF1
CU2	Autenticar usuario.	Usuario	RF2
CU3	Gestionar proyecto.	Usuario avanzado	RF3
CU4	Gestionar tipos de requisitos.	Usuario avanzado	RF4
CU5	Gestionar atributo.	Usuario avanzado	RF5
CU6	Gestionar paquete.	Usuario avanzado	RF6
CU7	Gestionar requisito.	Usuario avanzado	RF7

CU8	Mostrar requisito.	Usuario	RF8
CU9	Mostrar vista.	Usuario	RF9
CU10	Generar reporte histórico.	Administrador	RF10
CU11	Mostrar cambios recientes.	Usuario	RF11

Tabla 3: CU del sistema.

Diagrama de casos de uso.

El modelado de CU se realiza con el objetivo de modelar de una forma simple y efectiva los requisitos del sistema desde el punto de vista del usuario. El diagrama de CU constituye una visión general que se ha identificado para satisfacer los requerimientos funcionales del sistema.

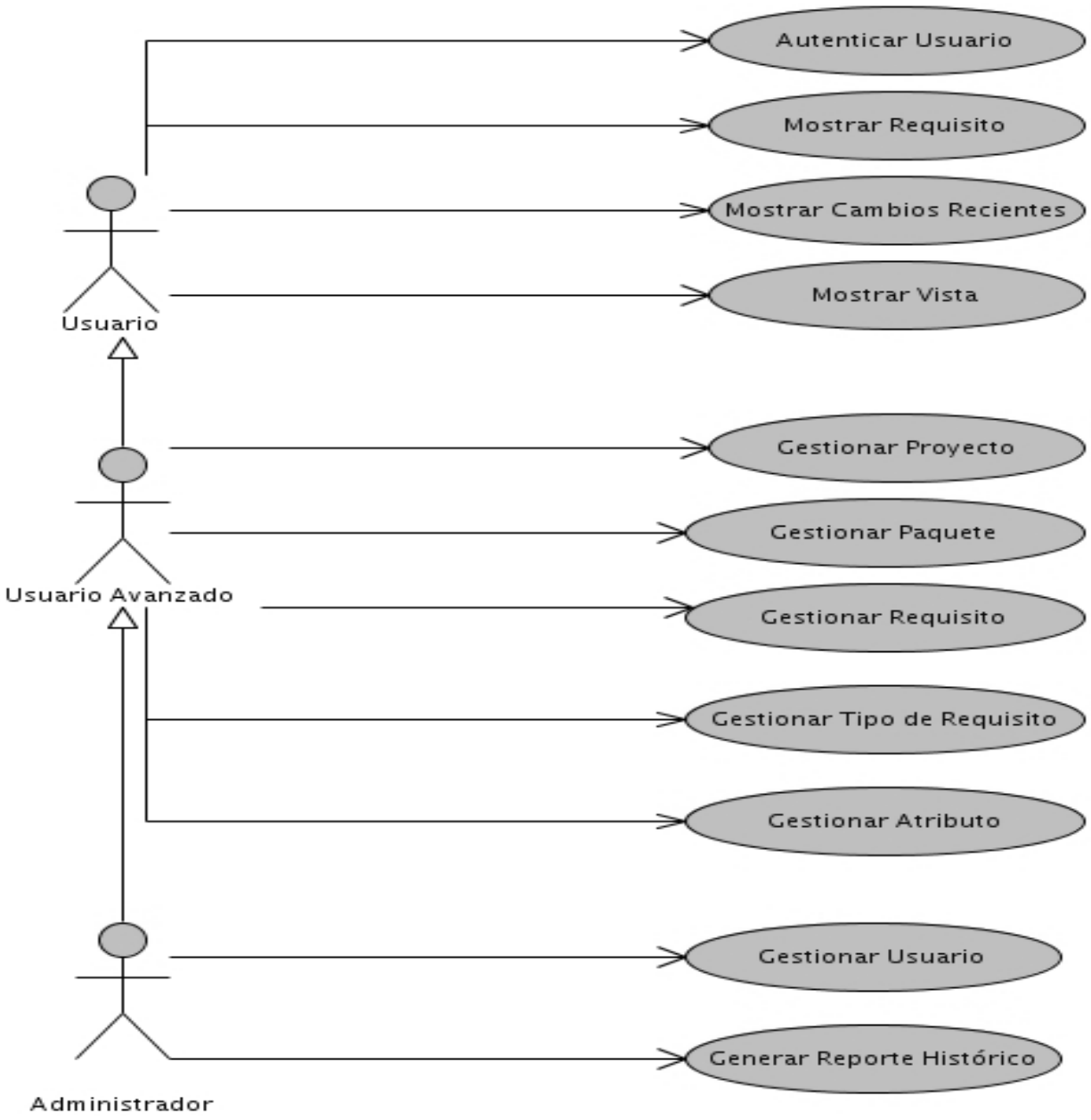

Figura 2: diagrama de CU del sistema.

2.2.5 Especificación de casos de uso del sistema.

Los CU pretenden ser herramientas simples para describir el comportamiento del software o de los sistemas. Un CU contiene una descripción textual de todas las maneras que los actores previstos podrían

trabajar con el software o el sistema. Los CU no describen ninguna funcionalidad interna del sistema, ni explican cómo se implementará. Simplemente muestran los pasos que el actor sigue para realizar una tarea (33).

A continuación se describen dos de los casos de uso más críticos de la presente investigación:

Descripción del CU Mostrar requisito.

Caso de uso:	Mostrar requisito.
Actor(es):	Usuario, usuario avanzado y administrador.
Descripción:	Permite al usuario ver lo requisitos existentes, agrupados por proyecto y paquete.
Referencia:	RF8.
Precondiciones:	El usuario debe estar autenticado para realizar esta acción.
Poscondiciones:	Se mostraron los requisitos.
Flujo normal de eventos	
1. El CU se inicia cuando el usuario selecciona la opción “Mostrar requisito”.	2. El sistema mostrará una interfaz donde los requisitos van estar agrupados por proyectos y paquetes.
	3. Termina el caso de uso.
Flujo alterno de los eventos	
2.a No existen requisitos en la base de datos.	
	2.a.1 El sistema muestra el siguiente mensaje “No existe ningún requisito para mostrar”.

Tabla 4: descripción del CU “Mostrar requisito”.

Descripción CU Gestionar requisito.

Caso de uso:	Gestionar requisito.
Actor(es):	Usuario avanzado y administrador.
Descripción:	Permite al usuario avanzado crear, mostrar, modificar y eliminar un requisito.
Referencia:	RF7.

Precondiciones:	El usuario debe estar autenticado con privilegios para realizar esta acción.
Poscondiciones:	Se creó, modificó, mostró o eliminó un requisito en el sistema, actualizándose la tabla que contiene la información referente a los mismos.
Flujo normal de eventos	
Acciones del actor	**Acciones del sistema**
1. El CU se inicia cuando el actor selecciona la opción "Gestionar requisito".	2. El sistema muestra las siguientes opciones: ✓ Crear requisito. Ver sección 1: "Crear requisito". ✓ Listar requisito. Ver sección 2: "Listar requisito". ✓ Insertar trazabilidad. Ver sección 5: "Insertar trazabilidad".
Sección 1: "Crear requisito"	
1. Selecciona la opción "Crear requisito".	2. El sistema muestra los campos. ✓ Nombre. ✓ Descripción. ✓ Proyecto. ✓ Paquete. ✓ Tipo de requisito. ✓ Adjuntar archivo. Permite seleccionar la medición de los atributos para el requisito. Además: • Crear un requisito. • Cancelar.
3. Introduce los datos del nuevo requisito y selecciona el botón "Crear".	4. El sistema valida los datos de entrada.
	5. El sistema almacena la información, posteriormente direcciona al usuario a la opción "Listar requisitos" y muestra el siguiente mensaje: "El requisito se ha creado correctamente".

	6.Termina el caso de uso.
Flujo alterno de los eventos	
3.a Selecciona la opción “Cancelar”.	
	3.a.1 El sistema regresa a la opción “Listar requisito”. Ver sección “Listar requisito”
3.b Selecciona la opción “Asignar atributos”.	
	3.b.1 El sistema permite seleccionar los atributos.
3.b.2 Selecciona los atributos.	3.b.3 Ejecuta el paso 3 del flujo básico.
4.a Existen campos vacíos.	
	4.a.1 El sistema marca el campo vacío.
	4.a.2 Muestra el siguiente mensaje: “Debe llenar el campo señalado“.
	4.a.3 Regresa al paso 3 del flujo básico.
Sección 2: “Listar requisito”	
1. Selecciona la opción “Listar requisito”.	2. El sistema muestra una tabla con todos los requisitos y los siguientes datos: ✓ Nombre. ✓ Proyecto. ✓ Paquete. ✓ Adjunto. Permite: ✓ Modificar un requisito. Ver sección 3: “Modificar requisito”. ✓ Eliminar un requisito. Ver sección 4: “Eliminar requisito”. ✓ Eliminar trazabilidad. Ver sección 6: “Eliminar trazabilidad”. Además, filtrar datos de un listado de requisitos. Ver sección 7:

	"Filtrar datos"
Sección 3: "Modificar requisito"	
1. Selecciona la opción "Modificar un requisito".	2. El sistema muestra los datos del requisito y permite modificar los campos que el actor desee. ✓ Nombre. ✓ Descripción. ✓ Proyecto. ✓ Paquete. ✓ Tipo de requisito. ✓ Adjuntar archivo. Permite seleccionar la medición de los atributos para el requisito. Además: • Modificar un requisito. • Cancelar.
3. Modifica los datos deseados y selecciona la opción "Modificar".	4. El sistema valida los datos modificados.
	5. El sistema almacena la información modificada y regresa a la opción "Listar requisito".
	6. Termina el caso de uso.
Flujo alterno de los eventos	
3.a Selecciona la opción "Cancelar".	
	3.a.1 El sistema regresa a la opción "Listar requisito". Ver sección "Listar requisito"
3.b Selecciona la opción "Asignar atributos".	
	3.b.1 El sistema permite seleccionar los atributos.

3.b.2 Selecciona los atributos	3.b.3 Ejecuta el paso 3 del flujo básico.
4.a Existen campos vacíos.	
	4.a.1 El sistema marca el campo vacío.
	4.a.2 Muestra el siguiente mensaje: “Debe llenar el campo señalado“.
	4.a.3 Regresa al paso 3 del flujo básico.
Sección 4: “Eliminar requisito”	
1. Selecciona la opción que permite eliminar un requisito.	2. El sistema muestra un formulario solicitando la confirmación al actor de eliminar un requisito específico y visualiza los siguientes datos del mismo: ✓ Nombre. ✓ Descripción. ✓ Proyecto. ✓ Paquete. ✓ Tipo de requisito. ✓ Adjunto. Permite: ✓ Eliminar. ✓ Cancelar.
3. Confirma la eliminación del requisito.	4. El sistema elimina el requisito y regresa a la opción “Listar requisito”.
	5. Termina el caso de uso.
Flujo alterno de los eventos	
3.a Selecciona la opción “Cancelar”.	
	3.a.1 El sistema regresa a la opción “Listar requisito”. Ver sección 2: “Listar requisito”.
Sección 5: “Insertar trazabilidad”	

1. Selecciona la opción “Insertar trazabilidad”.	2. El sistema muestra un campo que permite seleccionar el proyecto al cual corresponden los requisitos.
3. Selecciona un proyecto y la opción “Aceptar”.	4. Valida los datos.
	5. El sistema muestra un formulario que permite seleccionar los siguientes campos. ✓ Requisito. ✓ Requisito hacia atrás. ✓ Requisito hacia adelante. Además, permite: ✓ Adicionar la trazabilidad. ✓ Regresar a la vista anterior.
6. Selecciona los datos de la nueva trazabilidad y la opción “Adicionar”.	7. El sistema valida los datos de entrada y muestra el siguiente mensaje: “La trazabilidad se ha creado correctamente para el requisito (seleccionado en el campo requisito)”.
	8. Termina el caso de uso.
Flujo alterno de los eventos	
4.a Existen campos vacíos.	
	4.a.1 El sistema marca los campos vacíos.
	4.a.2 Muestra el siguiente mensaje: “Debe llenar los campos señalados“.
	4.a.3 Regresa al paso 3 del flujo básico.
7.a Existen campos vacíos.	
	7.a.1 El sistema marca los campos vacíos.
	7.a.2 Muestra el siguiente mensaje: “Debe llenar los campos señalados“.
	7.a.3 Regresa al paso 6 del flujo básico.

6.a Selecciona la opción “Regresar”.	
	7.a.1 El sistema regresa a la vista anterior.
6.b Trazabilidad existente.	
	6.b.1 El sistema muestra el siguiente mensaje: “No se puede hacer la trazabilidad dos veces al mismo requisito”.
	6.b.2 Ejecuta el paso 6 del flujo básico.
Sección 6: “Eliminar trazabilidad”	
1. Selecciona la opción que permite eliminar la trazabilidad de un requisito.	2. El sistema muestra un formulario para eliminar la trazabilidad a los requisitos, para lo cual se deben seleccionar los siguientes datos. ✓ Requisito hacia atrás. ✓ Requisito hacia adelante. Permite: ✓ Eliminar la trazabilidad. ✓ Cancelar.
3. Selecciona la trazabilidad a eliminar.	
	4. El sistema elimina la trazabilidad, regresa a la opción “Listar requisito” y muestra el siguiente mensaje: “La trazabilidad se ha eliminado correctamente”.
	5. Termina el caso de uso.
Flujo alterno de los eventos.	
3.a Selecciona la opción “Cancelar”.	
	3.a.1 El sistema regresa a la opción “Listar requisito”. Ver sección 2: “Listar requisito”.
Sección 7: “Filtrar datos”	
1. Selecciona la opción “Filtrar datos”.	2. El sistema muestra los siguientes campos para realizar la

	búsqueda opcional: ✓ Proyecto. ✓ Filtrar búsqueda (paquete o tipo de requisito).
3. Selecciona el campo que desee filtrar y la opción “Filtrar”.	4. El sistema muestra una tabla correspondiente al filtrado de los datos atendiendo al criterio de búsqueda seleccionado.
	5. Termina el caso de uso.
Flujo alterno de los eventos	
4.a Existen campos vacíos.	
	4.a.1 El sistema marca los campos vacíos.
	4.a.2 Muestra el siguiente mensaje “Debe llenar los campos señalados“.
	4.a.3 Regresa al paso 3 del flujo básico.

Tabla 5: descripción del CU “Gestionar requisito”.

2.3 Fase de elaboración.

En esta fase se analiza el dominio y diseño del problema. Se establece una arquitectura de base sólida y se estabilizan los requisitos y los planes de desarrollo (31).

2.3.1 Modelo de diseño.

Es un modelo físico y concreto, muy cercano a la implementación. Es una realización del diseño del sistema. Debe ser mantenido durante todo el ciclo de vida del software (34).

Diagramas de secuencia.

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada CU. Contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos (35).

A continuación se presenta el diagrama de secuencia, de uno de los CU más significativo (ver figura 3).

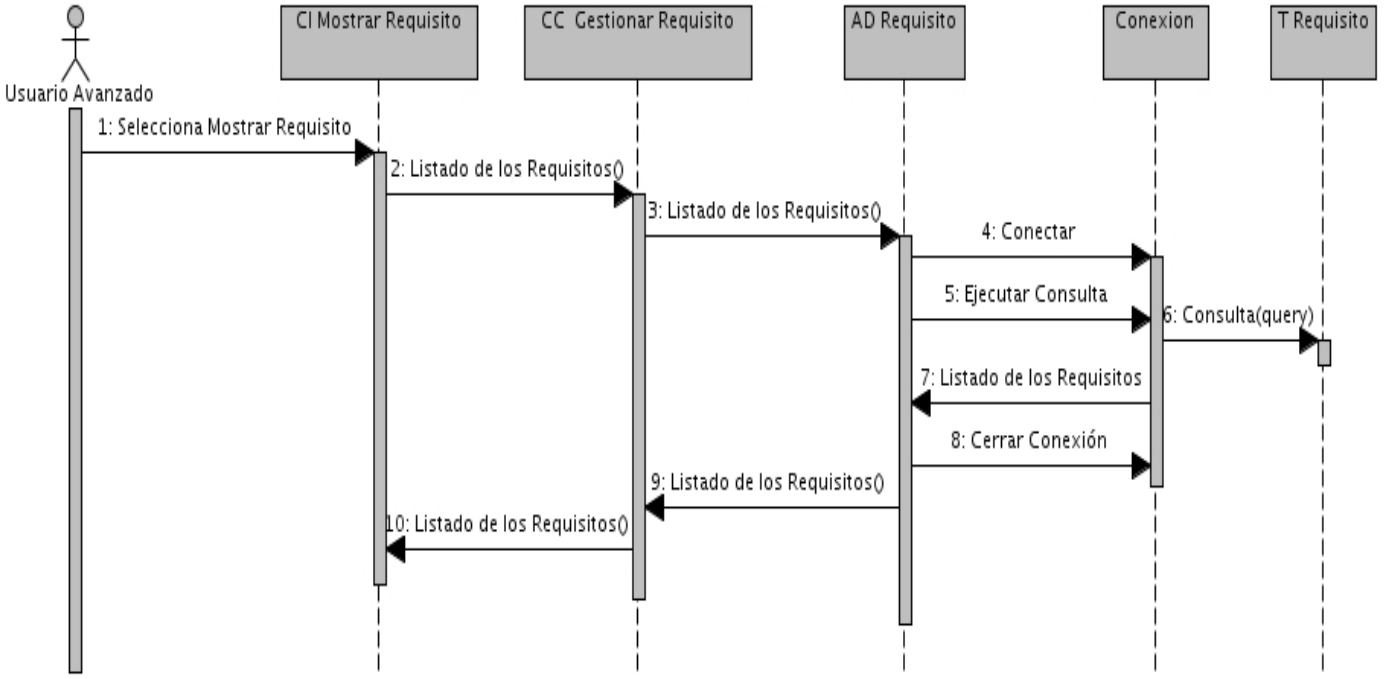

Figura 3: diagrama de secuencia para el CU "Mostrar requisito".

Diagrama de clases del diseño.

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos (36).

A continuación se visualiza el diagrama de clases de uno de los CU de la presente investigación (ver figura 4).

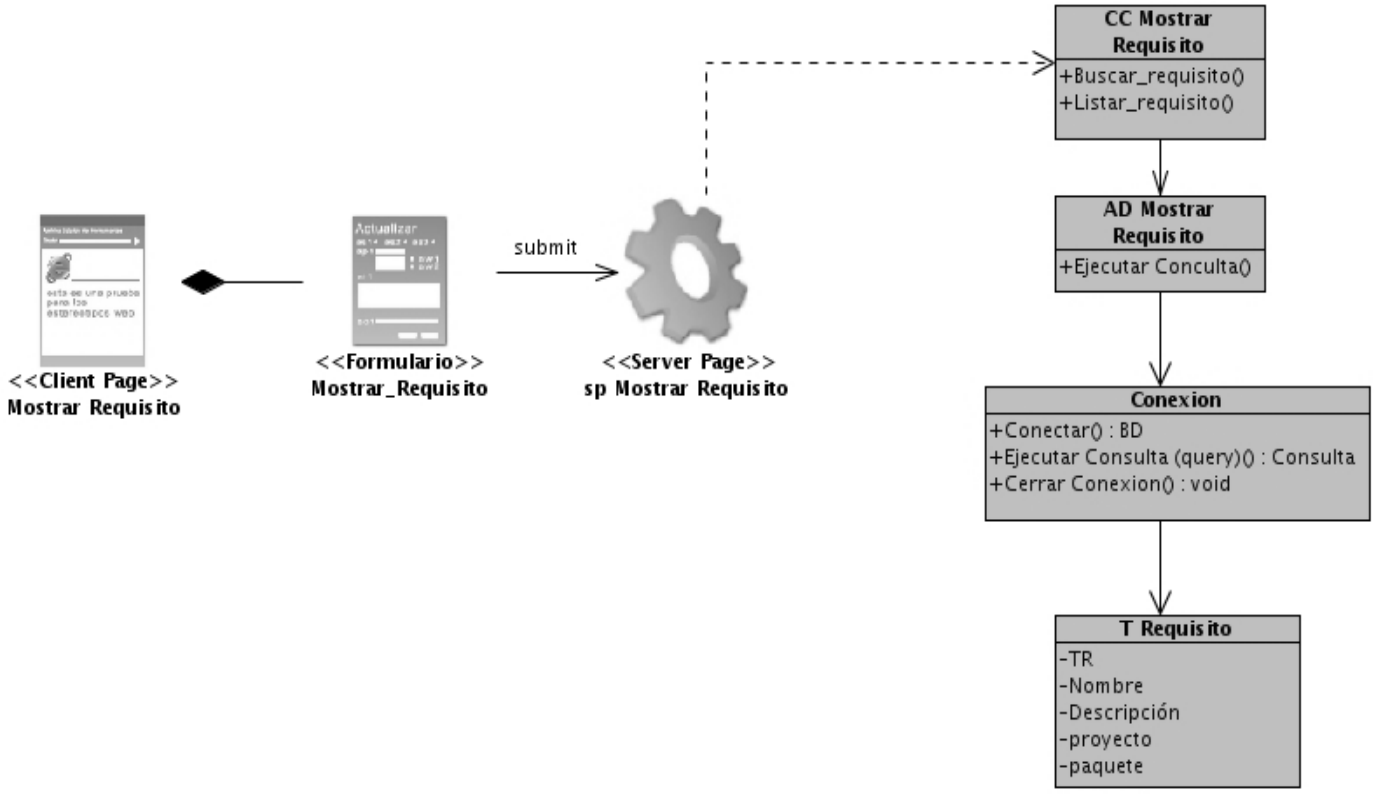

Figura 4: diagrama de clase CU "Mostrar requisito".

Modelo de datos.

Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia (ver figura 5).

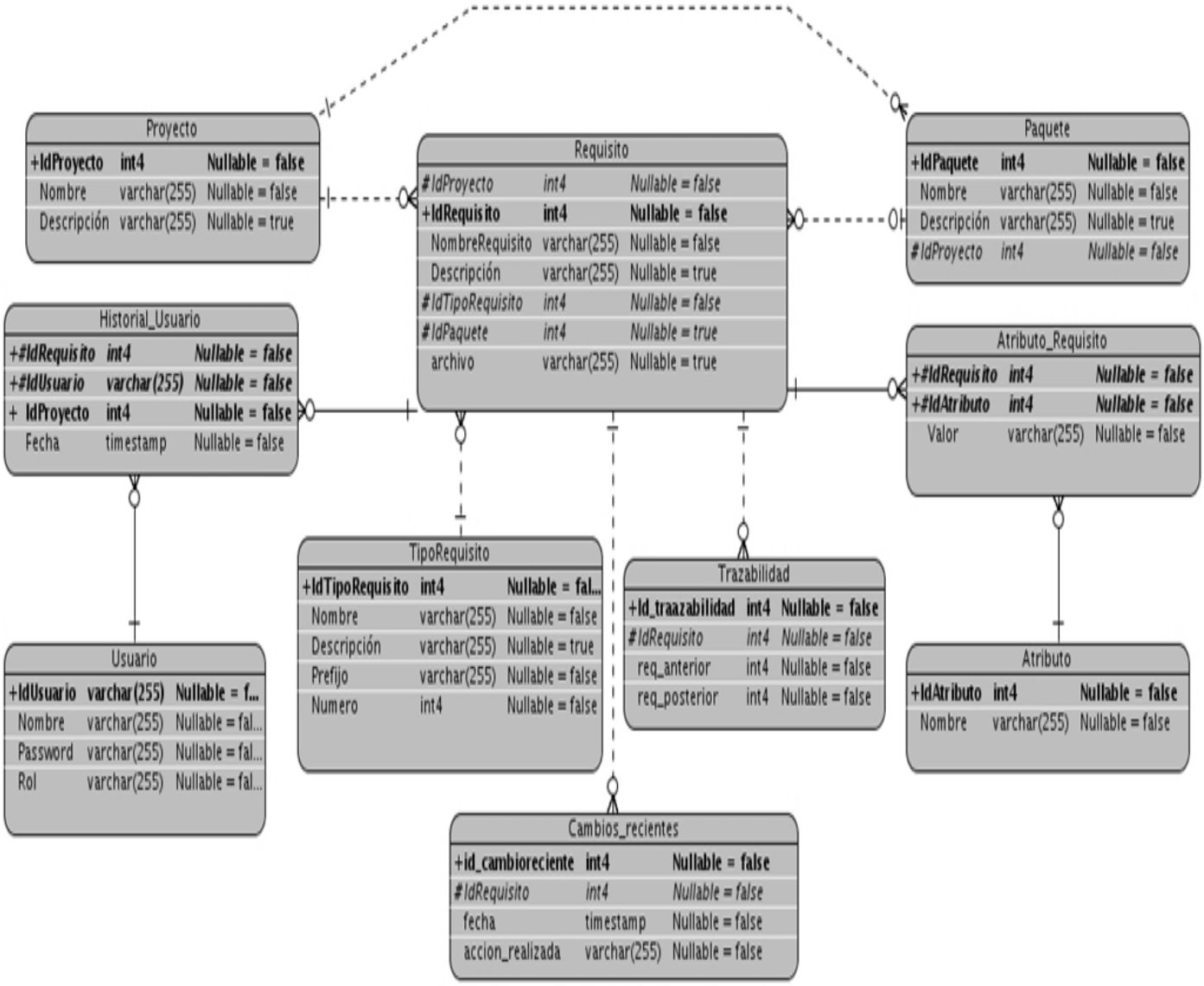

Figura 5: modelo de datos.

2.3.2 Modelo de arquitectura.

Patrón arquitectónico.

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. Para diseñar la arquitectura de la aplicación se escogió el patrón Modelo-Vista-Controlador (MVC) que actualmente es uno de los más utilizados para el desarrollo de aplicaciones Web. Este patrón separa los datos de una

aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos (ver figura 6). La vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el SGBD y el controlador representa la lógica del negocio (37).

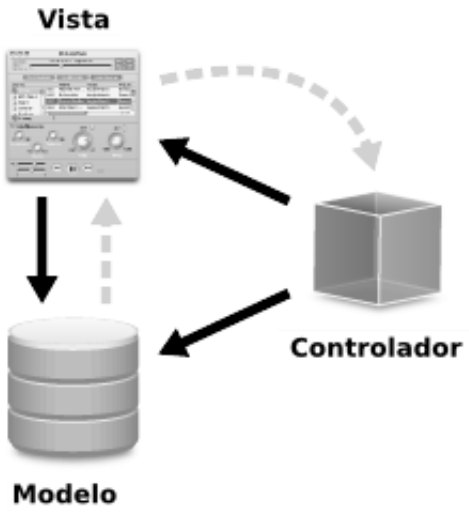

Figura 6: patrón MVC.

Este patrón de arquitectura de software se desarrolla rápidamente y de forma modular. Las funciones de la aplicación se separan en tres componentes distintos: modelo, vista y controlador; lo que permite hacer cambios en una parte de la aplicación sin que las demás se vean afectadas.

1. Modelo: este es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos. Es el responsable de acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento. Lleva un registro de las vistas y controladores del sistema.

2. Vista: esta presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Las vistas son responsables de recibir datos del modelo y mostrarlos al usuario.

3. Controlador: este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Es responsable de recibir los eventos de entrada. El controlador sirve como un intermediario entre el modelo, la vista y cualquier otro recurso necesario.

Las principales ventajas del patrón de arquitectura MVC (37):

1. Soporte para múltiples vistas ya que la vista se separa del modelo y no hay ninguna dependencia directa entre ambos, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos al mismo tiempo.

2. Proporciona un mecanismo de configuración a componentes complejos mucho más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).

3. Mayor soporte a los cambios: los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio. Puesto que el modelo no depende de las vistas, la adición de nuevos tipos de vista al sistema generalmente no afectan al modelo. Por tanto, el ámbito del cambio se limita a la vista.

Patrones de diseño.

Un patrón de diseño nombra, abstrae e identifica los aspectos claves de un diseño estructurado común, que lo hace útil para la creación de diseños orientados a objetos reutilizables. Definen una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones (38).

Los patrones de diseño tienen como propósito:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores, condensando conocimiento ya existente.

Patrones de diseños utilizados en la investigación.

Instancia única: está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Este patrón provee una única instancia global.

La utilización de este patrón posibilitará en la aplicación tener creado solamente un objeto de la clase que se encarga de establecer la conexión a la BD en todo momento.

Bajo acoplamiento: este patrón se utilizó con el objetivo de tener los módulos lo menos posible relacionados entre sí; evitando que en caso de producirse una modificación en alguno de ellos, se tenga la menor repercusión sobre el resto, potenciando así la reutilización y disminuyendo la dependencia entre los módulos.

Este patrón fue usado específicamente en el módulo atributo, que fue separado del módulo valor de medición, aunque los dos tienen una estrecha relación entre sí y con el módulo requisito. Dicha acción se realizó con el propósito de minimizar el impacto de uno sobre otro en caso de ocurrir algún cambio en cualquiera de los dos.

Alta cohesión: una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas para que no realicen un trabajo complejo y asuman más responsabilidades de las que les corresponden.

En esta investigación se utilizó este patrón para asignar a los módulos las responsabilidades correspondientes a cada uno, sin sobrecargar a un módulo con funciones de otro. Ejemplo de eso, se puso en práctica en el módulo requisito, el cual lleva el mayor peso de la aplicación ya que todos los módulos implementados se relacionan con él pero realizan sus funciones por separado.

2.4 Fase construcción.

En esta fase el propósito es completar el desarrollo del sistema basado en la arquitectura definida para asegurar la calidad del software (31).

2.4.1 Modelo de implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Además, cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación (39).

Módulos usados para desarrollar el sistema.

Los módulos en Drupal son ficheros con extensión .module que contienen funciones escritas en PHP. Estas son llamadas por Drupal durante sus procesos habituales de gestión de contenido. Por ejemplo,

cada vez que un nodo es creado, visualizado, modificado o borrado, Drupal llama a uno de estos ganchos pasándoles el contenido del nodo. De esta forma los módulos tienen la posibilidad de modificar y adaptar la información a visualizar en las páginas Web antes de que se mande definitivamente al navegador.

A continuación se mostrará los módulos que se utilizaron de Drupal para el desarrollo del sistema.

Nombre	Versión	Descripción
Bloque	6.20	Controla las cajas que se muestran alrededor del contenido principal.
Filtro	6.20	Maneja el filtrado de contenido en preparación para mostrarlo.
Nodo	6.20	Permite que se envíe contenido al sitio y que se despliegue en páginas.
Sistema	6.20	Gestión de la configuración general del sitio por administradores.
Usuario	6.20	Administra el registro de usuarios y el sistema de inicio de sesión.
Menú DHTML	6.x-3.5	Abre los menús dinámicamente para reducir refrescamientos de las páginas.
Nodo de privacidad por rol	6.x-1.6	Proporciona control de nivel de acceso de nodo basado en la pertenencia a funciones por roles.

Tabla 6: módulos de Drupal.

Aparte de estos módulos que usa el Drupal por defecto para su mejor funcionamiento, se implementaron otros más que la comunidad de desarrollo de Drupal no contempla y eran necesarios para dar cumplimiento a todas las funcionalidades que requiere el sistema en cuestión.

Nombre	Versión	Descripción
Atributo	1.0	Módulo que permite crear, modificar, listar y eliminar atributo.
Atributo requisito	1.0	Módulo que permite almacenar el valor de los atributos de los

		requisitos.
Historial	1.0	Módulo que permite tener el control de los usuarios que se relacionan con los requisitos del sistema.
Paquete	1.0	Módulo que permite crear, modificar, listar y eliminar paquete.
Proyecto	1.0	Módulo que permite crear, modificar, listar y eliminar proyecto.
Requisito	1.0	Módulo que permite crear, modificar, listar y eliminar requisito.
Requisito reciente	1.0	Módulo que permite tener información de los cambios recientes de los requisitos.
Tipo de requisito	1.0	Módulo que permite crear, modificar, listar y eliminar tipo de requisito.
Trazabilidad	1.0	Módulo que permite guardar la trazabilidad de los requisitos.
Valores de medición	1.0	Módulo que permite asignar valores a los atributos.

Tabla 7: módulos implementados.

Diagrama de componentes.

Un diagrama de componentes muestra cómo se encuentra dividido un software y las relaciones entre sus componentes. Este tipo de diagramas presentan un nivel de abstracción más elevado que los diagramas de clases, aunque los componentes usualmente se encuentran implementados por clases. Además, se utiliza para modelar las diferentes vistas de un sistema (estática y dinámica) y para mostrar qué componentes pueden ser utilizados en común por varios sistemas o entre las diferentes partes del mismo (40).

A continuación se muestra el diagrama de componentes para el módulo requisito, elaborado por el autor de la presente investigación:

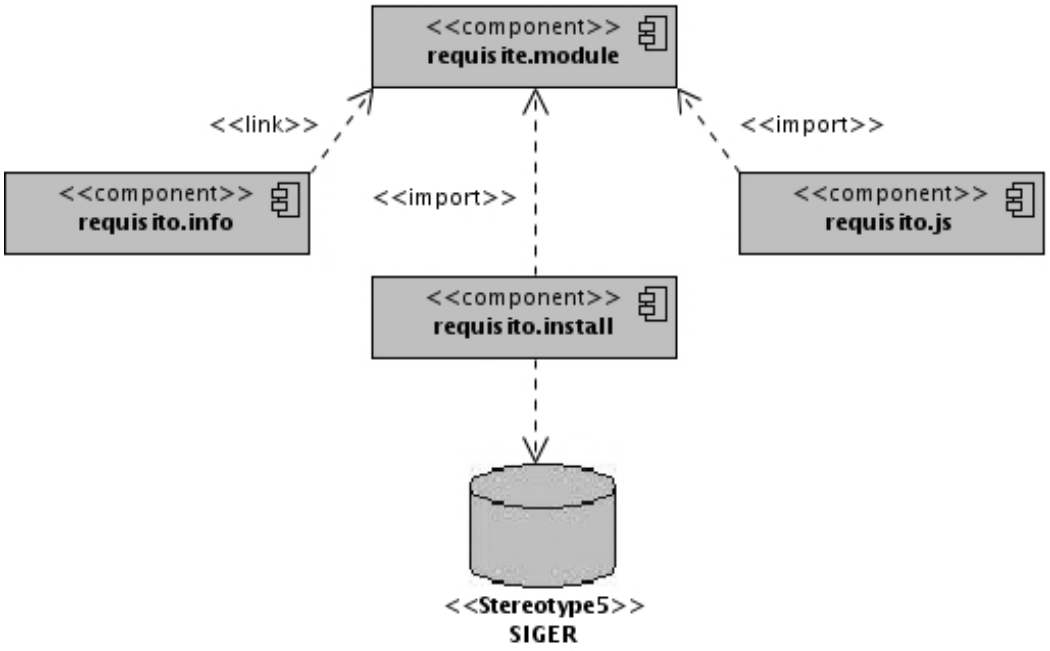

Figura 7: diagrama de componentes.

Cada módulo utilizado de Drupal cuenta con esta estructura idéntica a la figura anterior, donde todos los módulos implementados para la aplicación se agrupan en un paquete llamado “custom”. A continuación se muestra una vista general de cómo está conformado y estructurado Drupal (ver figura 8) (41).

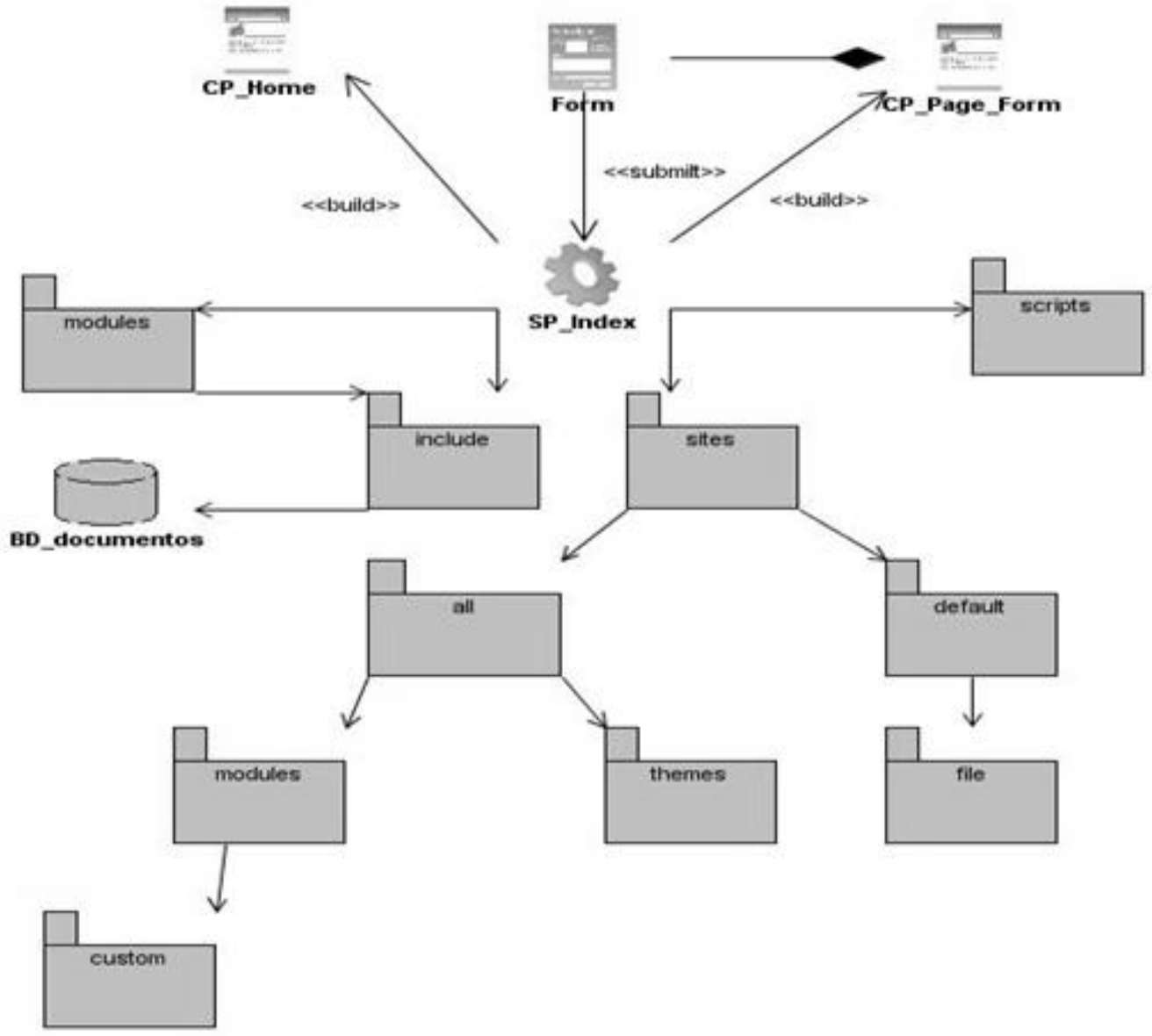

Figura 8: diseño estructural de Drupal.

Diagrama de despliegue.

En un diagrama de despliegue se muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Los nodos son elementos físicos que existen en tiempo de ejecución y representan un recurso computacional que generalmente tienen algo de

memoria y capacidad de procesamiento. El diagrama de despliegue sirve para modelar la topología de hardware sobre la cual se ejecutará el sistema (42).

A continuación se visualiza el diagrama de despliegue propuesto para la aplicación (ver figura 9).

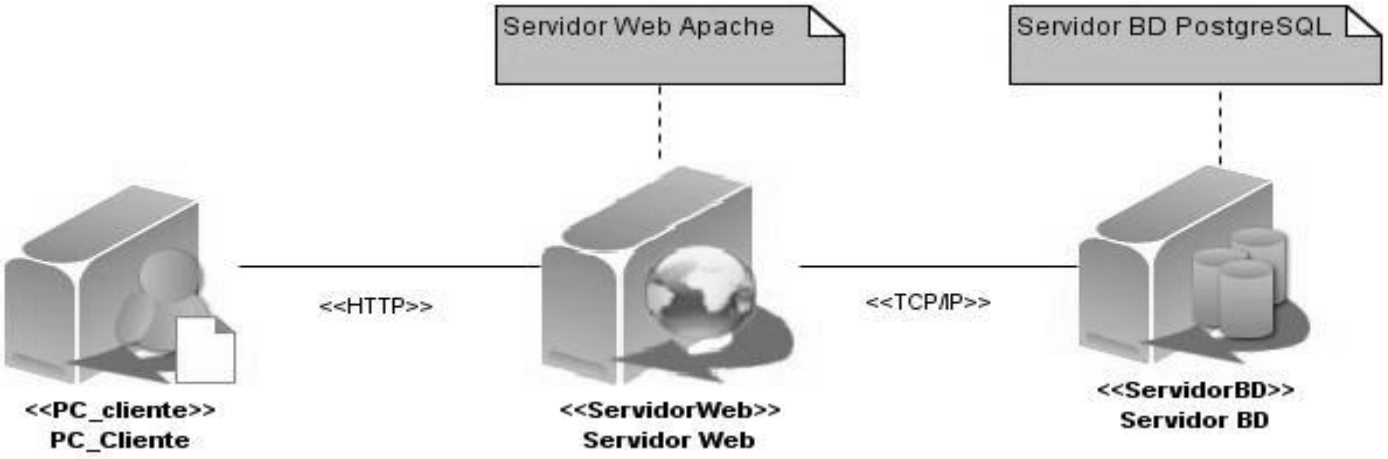

Figura 9: diagrama de despliegue.

Descripción:

La distribución física del sistema se compone por: computadoras clientes, las cuales están conectadas mediante el protocolo HTTP a un servidor donde se encuentra instalado un servidor Apache 2.0 y está a su vez se encuentra conectada por el protocolo TCP/IP con otro servidor, donde se encuentra el SGBD PostgreSQL 8.4.

2.5 Conclusiones.

Al terminar el presente capítulo se obtuvo la concepción total del sistema, definiéndose alcance y visión. Luego de un levantamiento de información se clasificaron y refinaron los requisitos, que se convirtieron luego en CU, los cuales fueron descritos. También se modelaron diagramas de: secuencia, clases y BD, así como la distribución física que debería tener la aplicación al desplegarse. Con los insumos anteriores terminados, se implementó el sistema sobre algunas iteraciones guiadas por los requisitos funcionales.

3 *Capítulo.*

Pruebas a la solución.

3.1 Introducción.

Las pruebas son un factor de vital importancia para la producción de un software, ya que persiguen el objetivo de garantizar la calidad del producto, y por tal motivo juegan un papel protagónico, pues con la ejecución de un conjunto finito de casos de prueba es posible verificar el comportamiento del producto durante su ciclo de vida. Durante este capítulo se estarán definiendo estos elementos de importancia para garantizar un funcionamiento óptimo del software final.

3.2 Fase de transición.

Esta fase tiene como objetivo realizar pruebas para determinar si se alcanzaron las expectativas de los usuarios y la concordancia entre los logros del producto final con los requisitos definidos (31).

3.2.1 Pruebas.

Este proceso garantiza la calidad del software, tiene como objetivo comprobar la satisfacción de los requisitos y localizar para subsanar, el mayor número de deficiencias antes de liberar el producto final. El tamaño y complejidad del proceso, depende del tamaño y complejidad del producto que se está desarrollando y para ejecutarlo es necesario definir nivel, método y tipo de pruebas a utilizar (43).

Como en esta investigación la metodología que se utilizó fue Open UP, la misma precisa varias tareas en funciones de la calidad del producto final por lo que se definió lo siguiente para realizar los casos de prueba y la lista de no conformidades.

Nivel de prueba: sistema.

En este nivel es verificado que cada elemento interactúe de forma adecuada, que sea alcanzado la funcionalidad y el rendimiento del sistema total, sean validados los requisitos establecidos comparándolos

con el sistema construido. El software debe integrarse con los componentes de hardware correspondientes respondiendo de manera funcional acorde con lo propuesto (43).

Tipo de prueba: funcionalidad.

Este tipo de prueba asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Verifica la apropiada aceptación de datos y se encuentra enfocada principalmente a los requisitos funcionales o CU (43).

Método de prueba: caja negra.

El método de prueba aplicado fue el de caja negra el cual se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Este tipo de prueba permite obtener un conjunto de condiciones de entrada que ejerciten de forma completa todos los requisitos funcionales de un programa, es un enfoque complementario que intenta descubrir diferentes tipos de errores como por ejemplo errores de interfaz, errores en estructuras de datos o acceso a las BD entre otros (43).

3.2.2 Diseño de casos de prueba.

La sección está dedicada a mostrar los casos de prueba (CP) diseñados que poseen la responsabilidad de probar las funcionalidades definidas para el software. En la siguiente tabla se muestra un caso de prueba realizado al CU Mostrar requisito.

CU Mostrar requisito

Nombre del CU	Mostrar requisito.
Descripción general	Permite al usuario ver lo requisitos existentes en la BD agrupados por proyecto y paquete.
Pruebas realizadas	- Mostrar requisito.

Tabla 7: descripción general del CU "Mostrar requisito".

CP 1 Mostrar requisito

Nombre del CP	Mostrar requisito.
Descripción.	Permite al usuario visualizar los requisitos existentes agrupados por proyectos y paquetes.
Flujo central.	1. El usuario avanzado selecciona en el menú la opción "Mostrar Requisito".

	2. El sistema muestra los requisitos existentes agrupados de forma jerárquica por proyectos y paquetes.
Condiciones de ejecución.	El usuario debe estar autenticado en la aplicación.

Tabla 8: CP 1 "Mostrar requisito".

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Seleccionar la opción: "Mostrar requisito."		El sistema debe mostrar los requisitos existentes agrupados de forma jerárquica por proyectos y paquetes. En caso que no exista ningún requisito en la BD, se mostrará un mensaje informándolo.	- Se mostraron los requisitos existentes, agrupados de forma jerárquica por proyectos y paquetes. - Ante la inexistencia de requisitos en la BD se mostró un mensaje de notificación.	El sistema se comportó de forma esperada.

Tabla 9: iteración CP 1 "Mostrar requisito".

Las no conformidades que se detectaron por iteración, con la ejecución de los 11 casos de prueba, son las siguientes:

Iteración	No conformidades	Descripción de las no conformidades	Clasificación	Estado
	1	El botón cancelar de la funcionalidad "Modificar proyecto" no funciona correctamente, pide escribir los campos obligatorios cuando se dejan	Critica	Resuelta

		en blanco.		
1	2	El mensaje de validación de la funcionalidad “Crear proyecto” está mal redactado.	No crítica	Resuelta
	3	El eliminar del tipo de requisito, no está eliminando los requisitos que pertenecen a él.	Crítica	Resuelta
	4	No es descriptivo el campo “Número de requisito”, debería llamarse número de inicio.	No crítica	Resuelta
	5	En la funcionalidad del tipo de medición en la opción “Asignar valores de medición” existe error ortográfico (múltiples).	No crítica	Resuelta
	6	Cuando se filtra un requisito por los criterios de búsqueda, el botón dice en su texto “Deshacer”, sería más descriptivo “Volver”.	No crítica	Resuelta
	7	En la tabla principal del listado de los requisitos la columna trazabilidad está en minúscula y no se estandariza como en las demás columnas.	No crítica	Resuelta
	8	En la funcionalidad de “Eliminar trazabilidad” el campo “Requisito hacia atrás:” y el mensaje “Por favor seleccione el requisito hacia atrás para eliminarlo” tienen errores ortográficos.	No crítica	Resuelta
	9	En la columna de la opción “Mostrar requisitos” debe estar atributos en plural.	No crítica	Resuelta
	10	El botón “Deshacer” luego del filtrado en la opción “Vista por atributos” no es descriptivo, debería decir “Volver”.	No crítica	Resuelta
	11	La opción “Valor simple” no funciona en crear atributo.	Crítica	Resuelta

	12	La funcionalidad “Mostrar trazabilidad” no genera correctamente la matriz.	Crítica	Resuelta
	13	En la funcionalidad “Crear proyecto” el mensaje mostrado cuando la descripción excede los 255 caracteres está poco descriptivo.	No crítica	Resuelta
	14	Cuando se lista los requisitos en la opción “Listar requisito” luego de realizado un filtraje, la tabla que se muestra en una columna tiene el nombre “trazabilidad” perdiendo los estándares de las otras columnas, que están en mayúsculas.	No crítica	Resuelta
	15	En la sección de valores de medición, en el campo “Segundo valor”, el mensaje de descripción tiene errores ortográficos (Valor opcional para selección múltiple.)	No crítica	Resuelta
	16	En el filtrado está mal redactado el mensaje de error cuando no se selecciona el proyecto y tiene error ortográfico.	No crítica	Resuelta
	17	Al usuario avanzado no se le muestra “Listar atributos”.	Crítica	Resuelta
	18	En el listar de la funcionalidad “Gestionar atributo”, el campo nombre está con minúscula.	No crítica	Resuelta
2	1	En la vista de atributos, la funcionalidad filtrar el botón “Deshacer” no es descriptivo, debería llamarse “Volver”.	No crítica	Resuelta
	2	En la opción “Cambios recientes” no muestra los requisitos eliminados.	Crítica	Resuelta

	3	En la funcionalidad “Crear proyecto” el mensaje mostrado cuando el nombre excede los 50 caracteres está muy grande.	No crítica	Resuelta
	4	En la funcionalidad “Generar reporte histórico” falta los puntos finales de los mensajes mostrados.	No crítica	Resuelta
	5	En la funcionalidad “Filtrar datos del paquete” cuando no se selecciona el proyecto y se presiona el botón “Filtrar”, el mensaje mostrado tiene error ortográfico (carácter) y se debería valorar poner un mensaje más atractivo.	No crítica	Resuelta
3	1	El mensaje de validación de la funcionalidad “Crear proyecto” está mal redactado.	No crítica	Resuelta

Tabla 10: no conformidades detectadas.

Para el correcto desarrollo del proceso de pruebas de la aplicación implementada se definieron un total de 11 casos de prueba, ejecutándose 3 iteraciones y utilizando el método de caja negra para probar las principales funcionalidades que debe cumplir el sistema. Fueron detectadas 24 no conformidades, de ellas 6 críticas, del resto la mayoría fueron errores ortográficos. Los problemas detectados tenían una repercusión negativa sobre el funcionamiento del sistema y la BD. Posteriormente con la erradicación de los mismos se pudo mejorar y optimizar gradualmente el funcionamiento de la aplicación.

Iteración	**Cantidad de no conformidades**	**Críticas**	**No críticas**	**Cantidad resueltas**
1	18	5	13	18
2	5	1	4	5
3	1	0	1	1
Total	24	6	18	24

Tabla 11: resumen de las pruebas.

3.3 Conclusiones.

Durante la ejecución de los casos de prueba diseñados en correspondencia con los casos de uso, se detectaron un conjunto de no conformidades que señalaban deficiencias del software. Estas carencias fueron todas solucionadas, garantizando un mejor funcionamiento de la herramienta propuesta.

Conclusiones generales.

Al terminar la presente investigación se puede observar que el objetivo general de conjunto con los específicos fue cumplido satisfactoriamente, además de arribarse a las siguientes conclusiones:

- ✓ La aplicación Web desarrollada, brinda soporte para la administración de requisitos en los proyectos del centro FORTES y cumple con las políticas de migración de la universidad.
- ✓ Con el uso de la herramienta implementada los proyectos del centro FORTES podrán automatizar el control de cambios sobre sus requisitos y documentar éstos sin interferir con la configuración propia que utilizan.
- ✓ Como resultado de todo el proceso investigativo, se obtiene una amplia documentación, la cual servirá para implementaciones posteriores.
- ✓ La aplicación Web obtenida en la investigación, mantiene disponible para el equipo de proyecto la información actualizada de los cambios que puedan ocurrir sobre los requisitos, con el objetivo final de contribuir al éxito de los productos finales desarrollados en el centro FORTES.

Recomendaciones.

Para obtener el máximo rendimiento de la herramienta y la calidad en el producto final se recomienda:

- ✓ Incorporar nuevas funcionalidades para aumentar la calidad en la prestación de servicio como pueden ser:
  - Trazabilidad por árbol jerárquico.
  - Trazabilidad por CU y actores del sistema.
- ✓ Extender la solución al resto de los proyectos productivos de la Universidad de las Ciencias Informáticas, ya que la herramienta funcionará como despliegue piloto en el centro FORTES pero por sus características puede aplicarse a cualquier centro de desarrollo de la UCI.

Referencias bibliográficas.

1. **Pressman, R.** *Ingeniería del Software: Un enfoque práctico.* 1997.

2. **Sommerville.** *I.Ingeniería de Software, Pearson Educación.* 2002.

3. **Alfaro, Carlos Monge.** *LA INGENRIA DE REQUERIMIENTOSY SU IMPORTANCIA EN EL DESARROLLO DE PROYECTOS DE SOFTWARE .* Costa Rica : s.n., 2005.

4. **Society, IEEE Computer.** *IEEE Standard Glossary of Software Engineering Terminology.* 1990. std 610.12-1990.

5. **Mendoza, Gonzalo Mena.** *Procesos de la Ingeniería de Requerimientos.* julio de 2005.

6. **Landazuri, Bárbara A. Mcdonald.** *Definición de Perfiles en Herramientas de Gestión de Requisitos.* España : s.n., 2005.

7. IBM - Rational RequisitePro - Software. [En línea] 2009. [Citado el: 10 de 12 de 2010.] http://www.ibm.com/software/awdtools/reqpro/.

8. REM. [En línea] [Citado el: 10 de 6 de 2011.] http://www.lsi.us.es/descargas/descarga_programas.php?id=3.

9. OSRMT. [En línea] 2010. [Citado el: 11 de 12 de 2010.] http://ipcorp.com.ar/blog/?p=15..

10. **Nosek, Palvia y.** *A Field Examination of System Life Cycle Techniques and Methodologies.* 1993.

11. **Martínez, Alejandro Martínez y Raúl.** *Guía a Rational Unified Process .* 2008.

12. **Leonardi, Maria Carmen y Leite, Julio Cesar Sampaio do Prado.** *Un Proceso para XP basado en Regla del Negocio.* 2009.

13. Open UP. [En línea] 2008. [Citado el: 8 de 12 de 2010.] http://epf.eclipse.org/wikis/openup/.

14. **Juan Palacio, 2006.** *El modelo Scrum.* 2006.

15. Tutorial de UML. [En línea] 2010. [Citado el: 14 de 12 de 2010.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

16. Visual Paradigm– UML. [En línea] 2010. [Citado el: 14 de 12 de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/ .

17. **Rational, Rose.** Rational Rose Enterprise. [En línea] 2007. [Citado el: 15 de 12 de 2010.] http://www.rational.com.ar/herramientas/roseenterprise.html..

18. ¿Que es un sistema de gestión de contenidos? [En línea] 2007. [Citado el: 20 de 1 de 2011.] http://www.ibersoluciones.com/que-es-un-sistema-de-gesti-n-de-contenidos.html..

19. Creacion de sitios web basados en Joomla CMS. Un Sistema de Gestión de Contenidos de Clase Mundial. . [En línea] 2008. [Citado el: 20 de 1 de 2011.] http://www.infocreaciones.com/index.php?option=com_content&task=view&id=57&Itemid..

20. Drupal - Open Source CMS | drupal. [En línea] 2010. [Citado el: 15 de 1 de 2011.] http://drupal.org/.

21. CMS Matrix. Tabla comparativa entre Joomla y Drupal. [En línea] [Citado el: 25 de 1 de 2011.] http://www.cmsmatrix.org/matrix/cms-matrix..

22. PHP: Hypertext Preprocessor . [En línea] [Citado el: 14 de 12 de 2010.]. [En línea] 2009. [Citado el: 14 de 12 de 2010.] http://www.php.net/.

23. *Manual de Java Script.* 2005.

24. **WebEstilo.** Conceptos básicos. *Tutorial de HTML.* [En línea] 2008. [Citado el: 16 de 12 de 2010.] http://www.webestilo.com/html/.

25. eclipse. [En línea] 2010. [Citado el: 12 de 1 de 2011.] http://www.eclipse.org/.

26. **NetBeans., Comunidad.** [En línea] 2010. [Citado el: 5 de 12 de 2010.] http://netbeans.org/community/releases/68/.

27. PostgreSQL. [En línea] 2008. [Citado el: 14 de 12 de 2010.] http://www.postgresql.org/.

28. **Internals, MySQL.** Resources for the MySQL Community. [En línea] 12 de 2007. [Citado el: 15 de 12 de 2010.] http://forge.mysql.com/wiki/MySQL_Internals.

29. Una Introdución a Apache. [En línea] 2010. [Citado el: 13 de 12 de 2010.] http://linux.ciberaula.com/articulo/linux_apache_intro.

30. Tutorial ISS- Internet Information Server. [En línea] 2009. [Citado el: 13 de 1 de 2011.] http://www.htmlpoint.com/iis/index.html.

31. **Jordana, Garcilaso.** *Introducción Open UP.* 2008.

32. **Dapena., MSc. Martha D. Delgado.** *Definición del modelo del negocio y del dominio.* 2008.

33. **Tello, Jesús Cáceres.** *Diagramas de Casos de Uso.* 2009.

34. **Jofré, Enrique.** *MODELO DE DISEÑO Y EJECUCIÓN DE ESTRATEGIAS DE NEGOCIOS.* 2002.

35. **Tello, Jesús Cáceres.** *DIAGRAMAS DE SECUENCIA.* 2009.

36. **Lázaro, Juan Carlos Gutiérrez.** *Diagramas de Clases y Casos de Uso.* 2008.

37. Patrón Modelo-Vista-Controlador. [En línea] 2009. [Citado el: 10 de 1 de 2011.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

38. **Craig, Larman.** *UML y Patrones. Introducción al análisis y diseño orientado a objeto.*

39. **eumed.net.** Enciclopedia virtual. [En línea] 2008. [Citado el: 30 de 6 de 2011.] http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm.

40. *Modelo de Implementación:Diagramas de Componentes y Despliegue.* 2008.

41. **Navarro, Anett Redondo y Zuleida.** *Sistema para la gestión de documentos.* 2011.

42. **Peñalvo, Francisco José García.** *Diseño orientado a objetos.* 2008.

43. **Juristo, Natalia, Moreno, Ana M. y Vegas,Sira.** *Técnicas de Evaluación de Software.* 2006.

Glosario de términos.

Aplicación Web: es una aplicación informática distribuida cuya interfaz de usuario es accesible desde un cliente Web, normalmente un navegador Web.

Base de datos: conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior.

Calidad: es la percepción del cliente sobre un producto o servicio, es una fijación mental del consumidor que asume conformidad con dicho producto o servicio y la capacidad del mismo para satisfacer sus necesidades.

Ciclo de vida: período de tiempo que comienza con la concepción del producto del software y termina cuando el producto está disponible para su uso.

Código abierto: permite a los desarrolladores leer, redistribuir, y modificar el código fuente de una aplicación.

Interfaz: es el conjunto de comandos y/o métodos que permiten la intercomunicación del programa con cualquier otro programa o entre partes (módulos) del propio programa, elemento interno o externo.

Microsoft Windows: es el nombre de una serie de sistemas operativos desarrollados por la empresa Microsoft desde 1981.

Scripts: secuencia de código sin compilar que puede ejecutar acciones, usualmente poco complejas.

Software: programas de sistema, utilerías o aplicaciones expresados en un lenguaje de máquina.

Software Libre: es el software, que una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente.