Universidad de las Ciencias Informáticas

Facultad 6

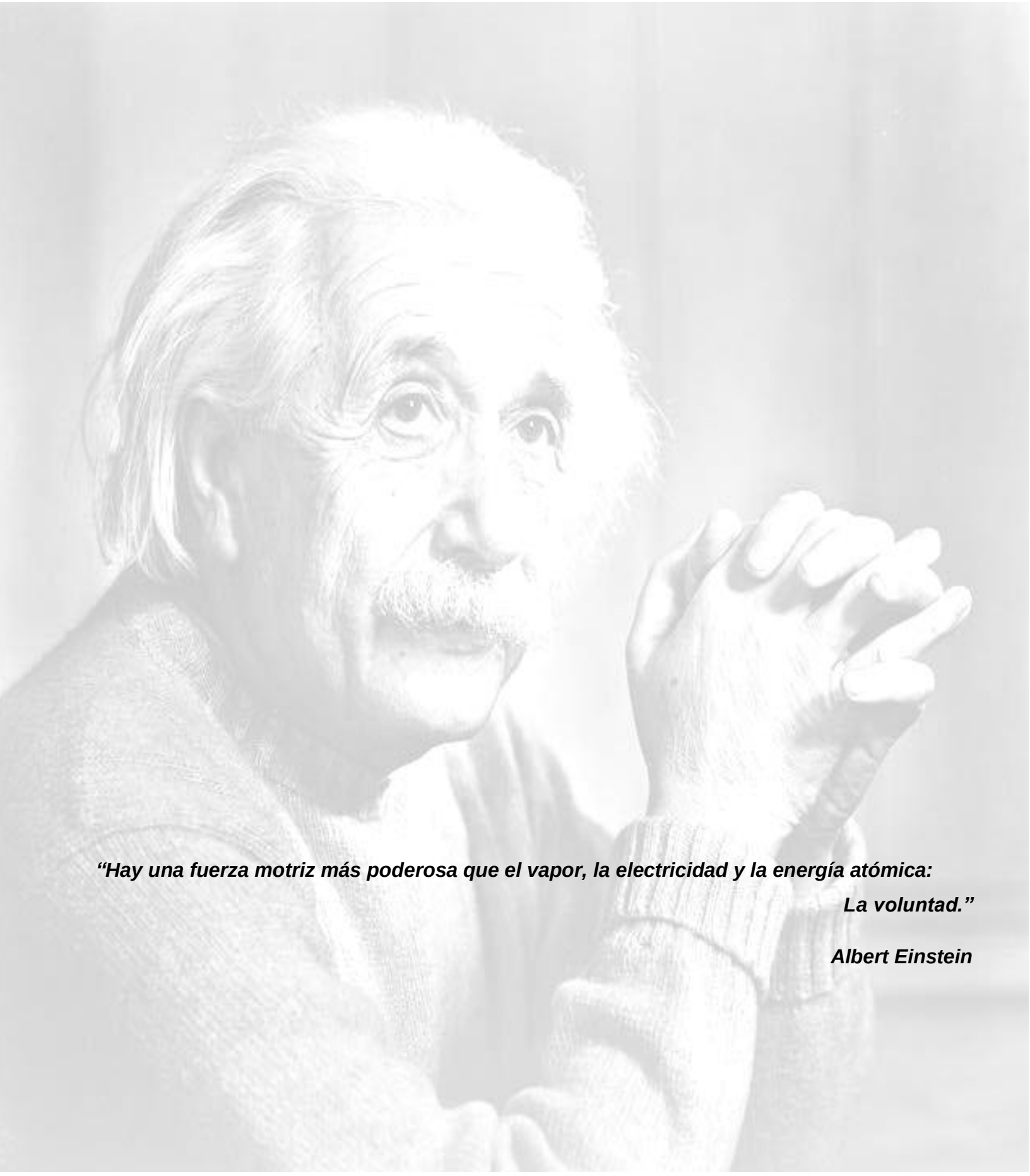
Título: “Aplicación Web que guíe la implantación del proceso base de Ingeniería de Requisitos en las empresas de producción de Software”.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor: Flavio Enrique Roche Rodríguez

Tutor: Ing. Yaima Bety Gijón.

Junio 2011



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica:

La voluntad."

Albert Einstein

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro de Calidad para Soluciones Informáticas (Calisoft) de la Universidad de las Ciencias Informáticas para que hagan el uso que estimen pertinente con este trabajo.

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

_____

Flavio Roche Rodríguez

Autor

_____

Ing. Yaima Bety Suarez Gijón

Tutor

Datos de Contacto

DIPLOMANTE

Nombre: Flavio Enrique Roche Rodríguez

E-mail: feroche@estudiantes.uci.cu

TUTOR

Nombre: Yaima Bety Suarez Gijón

E-mail: ysuarezg@uci.cu

Agradecimientos

Primero que todo mi más sincero agradecimiento a mis padres, en especial a mi querida madre que ha sido para mí ejemplo y guía en todo momento, todo lo que he logrado en esta vida se lo debo a ella. Gracias por haberme apoyado durante toda mi vida como estudiante e inculcarme los principios y valores para formarme como un buen profesional.

A mi familia que, aunque muy, muy lejos, me han apoyado y dedicado todo el tiempo que necesitaba.

Agradezco de manera especial a mi tutora Yaima que me ha ayudado mucho durante la realización de este trabajo.

A mis amigos (mis tantos y tantos amigos), por compartir conmigo los buenos y malos momentos. Por convertirse en mi familia cuando me encontraba lejos de la mía. En especial a Elio (El viejo), Vila (Nani), Jorge, Isael (El puro), Mayito, Carlos, Héctor, Abelito, Carlos (Bellcross), Reynaldo.

Agradezco también a las chicas que no podían faltar y me han ayudado mucho, Yanet, Yelena, Vania, Leidiana, Lara, Yesenia Hernández, Dayne, Lili y muy especial a las dos que han estado luchando conmigo durante estos 5 largos años, Vivi y Eli, gracias por todo.

A mi hermano de lucha durante todo mi andar por la escuela, muchos de los logros que he cosechado durante este tiempo son gracias a su ayuda y a su apoyo, más que un amigo es como un hermano para mí, Michael Marrero (El micho).

A la revolución cubana y especialmente al comandante en Jefe por darme la oportunidad de convertirme en un joven forjador del futuro.

Dedicatoria

Le dedico este momento tan importante de mi vida a las personas que le debo lo que soy:

A mi madre Blanquita gracias por estar siempre ahí para mí, todo lo que he logrado en esta vida es gracias a tu apoyo y al amor que siempre me has brindado, este título es para ti. Gracias por todo.

A mi padre Enrique y a mi abuelita Herlinda las otras dos personas más importantes para mí en esta vida, gracias por su amor y sus consejos.

A mis tíos que siempre me han apoyado desde pequeño, a mis tías María y Juana Irma, y a mi tío Juan.

Al resto de mi familia que siempre me apoyó.

RESUMEN

Debido a las dificultades que tienen las empresas cubanas de software para acceder a certificaciones de carácter internacional, el Centro de Calidad para soluciones informáticas de la UCI se ha dado a la tarea de crear un modelo para guiar a las empresas cubanas durante su proceso de desarrollo. En el trabajo se desarrolló una herramienta informática que sirve de ayuda para implantar este modelo en las empresas de Cuba, la que se encuentra enmarcada en el área de procesos de Ingeniería de Requisitos. Esta aplicación permitirá acceder a información de las diferentes actividades relacionadas con el área de procesos antes mencionada así como a los roles asociados a estas. Además permitirá el acceso y descarga a las distintas plantillas asociadas al área de procesos. De la misma forma la aplicación visualizará una serie de indicadores que permitirán observar el estado del proceso de requisitos. Para garantizar la portabilidad de la aplicación, esta se desarrolló bajo ambientes multiplataforma, fundamentalmente con herramientas de software libre.

PALABRAS CLAVE

Ingeniería de Requisitos, Requisitos, Indicadores, Actividades de Proceso.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Durante el de cursar de los años la humanidad ha sido testigo de los grandes y sorprendentes avances que la tecnología ha tenido, en la actualidad estos avances se suceden unos a otros; pero no siempre las cosas se comportaban de este modo, un factor determinante en este desarrollo ha sido el Software, que es el conjunto de los programas de cómputo, procedimientos, reglas, documentación que genera en su concepción, soporte y datos asociados que forman parte de las operaciones de un sistema de computación, mediante el cual se han facilitado y agilizado varios procesos que ya se manejaban con anterioridad.

Al percatarse de esto muchas empresas en el mundo comenzaron a dedicarse a la producción de software, los cuales en sus inicios se realizaban virtualmente y sin ninguna planificación, hasta que los planes comenzaron a descalabrarse y los costos a correr. Era muy frecuente escuchar que un gran número de los proyectos de software fracasaban por no realizar una adecuada administración y planificación.

Las industrias de software han continuado creciendo con el transcurso del tiempo. A raíz de las experiencias adquiridas muchas empresas han entendido la necesidad de incrementar la madurez en sus procesos de desarrollo para alcanzar altos niveles de productividad y comercialización, elementos fundamentales para poder sobrevivir en el mercado mundial.

Para incrementar estos niveles es necesario apoyarse en modelos que guíen los próximos pasos a realizar en vías de garantizar los objetivos planteados. Mundialmente existen muchos modelos entre los más conocidos y usados por las empresas productoras de software se encuentran CMMI (Capability Maturity Model Integration), el modelo ISO y el mexicano MOPROSOFT.

El país cuenta con la Industria Cubana de Software (ICS) la cual se caracteriza por contar con pequeñas y medianas empresas (Pymes), las cuales encuentran diversos obstáculos para poder introducirse en el mercado mundial, uno de ellos es el acceso a certificaciones de calidad internacionales debido a los escasos recursos con que cuentan, la complejidad que involucra su implantación, los altos costos de una consultoría especializada y la certificación final.

Las empresas cubanas a pesar de tener poca madurez en sus procesos de desarrollo debido a la falta de experiencia, producen software de calidad pero no la suficiente para asegurar el éxito de la mayoría de los proyectos y una amplia exportación de sus productos.

Para mitigar el problema anteriormente expuesto, el Centro de Calidad para soluciones informáticas (Calisoft) de la UCI, se ha dado a la tarea de crear un Modelo Cubano para el Desarrollo de Aplicaciones Informáticas que guíe a las empresas de software por el camino correcto.

Pero debido a la falta de experiencia y los bajos niveles de madurez en sus procesos de desarrollo de software que tienen estas empresas, así como la falta de organización, la inadecuada preparación de los recursos humanos, la abundante pero muy abstracta bibliografía que existe y el poco conocimiento que se tiene sobre el tema, es que se hace muy complicado la implantación del Modelo Cubano en las empresas de software.

Ante tal situación se plantea el siguiente **problema científico:** ¿Cómo apoyar la implantación del proceso base de Ingeniería de Requisitos del Modelo Cubano para el Desarrollo de Aplicaciones Informáticas en las empresas de producción de software?

Reconociendo como **objeto de estudio:** La Ingeniería de Requisitos.

Enmarcado en el **campo de acción:** Procesos de desarrollo y administración de requisitos del Modelo Cubano para el Desarrollo de Aplicaciones Informáticas.

En aras de solucionar el problema planteado se define como **objetivo general:** Desarrollar una Aplicación Web para guiar la implantación del proceso base de Ingeniería de Requisitos del Modelo Cubano para el Desarrollo de Aplicaciones Informáticas en las empresas de producción de Software.

A partir del análisis del objetivo general se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Definir las funcionalidades que deberá cumplir la herramienta.
- Diseñar una herramienta que guíe la implantación del proceso base de Ingeniería de Requisitos del Modelo Cubano para el Desarrollo de Aplicaciones Informáticas en las empresas de producción de Software.
- Implementar una herramienta que guíe la implantación del proceso base de Ingeniería de Requisitos del Modelo Cubano para el Desarrollo de Aplicaciones Informáticas en las empresas de producción de Software.

- Validar la herramienta.

Para dar cumplimiento a los objetivos específicos se trazaron las siguientes **tareas de investigación**:

- Revisión y análisis de las bibliografías relacionadas con la ingeniería de requisitos.
- Realización del estudio del estado del arte de herramientas que existen con funcionalidades parecidas a la que se pretende implementar.
- Definición de las herramientas de desarrollo para la implementación de la aplicación Obtención de los requisitos.
- Definición y especificación de los requisitos.
- Definición y descripción de los casos de usos.
- Análisis y propuesta de la arquitectura del sistema.
- Diseño de las clases y la base de datos.
- Elaboración del diagrama de despliegue.
- Elaboración del diagrama de componentes.
- Ejecución de pruebas de sistema, específicamente de caja negra.

El presente trabajo se encuentra estructurado en cuatro capítulos, resumidos de la siguiente forma:

Capítulo 1 Fundamento Teórico

En este capítulo se realiza el fundamento teórico del trabajo, se abordan varios conceptos asociados al problema para lograr un mejor dominio de los términos así como la selección de las herramientas de desarrollo que son utilizadas para la implementación de la aplicación.

Capítulo 2 Características del Sistema

Durante el desarrollo de este capítulo se brinda un fundamento de la solución de la propuesta, se amplía un poco más acerca de la información que se maneja y se profundiza en las características del sistema. Se presentan los requisitos funcionales y no funcionales, se identifican los casos de uso del sistema y se

describen los mismos; se priorizan estos casos de uso y se representa la vista de casos de usos.

Capítulo 3 Diseño del Sistema

Se analizan los Casos de Usos del Sistema para diseñar las clases que se implementan, se presentan los Diagramas de Secuencia del Diseño, los Diagramas de las Clases del diseño con sus relaciones, los patrones arquitectónicos y de diseño utilizados, así como el modelo de datos y la vista de despliegue.

Capítulo 4 Implementación y Prueba

Está basado fundamentalmente en la implementación del sistema y responde a los requisitos previamente establecidos, así como se prueban las diferentes funcionalidades que este brinda.

CAPÍTULO 1. FUNDAMENTO TEÓRICO

Introducción

En el presente capítulo se aborda el fundamento teórico relacionado con la Ingeniería de Requisitos. Además, se describen los principales conceptos asociados al problema, las principales aplicaciones y herramientas utilizadas para el desarrollo de este.

1.1 Ingeniería de Requisitos

La Ingeniería de Requisitos cumple un papel muy importante en el proceso de desarrollo de software, ella juega un rol fundamental, que es la definición de lo que se desea hacer. La Ingeniería de Requisitos es todavía una disciplina inmadura por lo que en la actualidad no existe una definición universalmente aceptada. Muchos son los autores que han dado su criterio al respecto, por lo que en la literatura se pueden encontrar diferentes definiciones. A continuación se citan algunas de ellas:

- Aplicación de principios científicos y técnicas para desarrollar, comunicar y gestionar requisitos. (1)
- Es el proceso sistemático de desarrollar requisitos mediante un proceso iterativo y cooperativo para analizar el problema, documentar las observaciones resultantes en varios formatos de representación y comprobar la precisión del conocimiento obtenido. (2)
- Todas las actividades relacionadas con la identificación y documentación de las necesidades del cliente y usuarios. La creación de un documento que describe la conducta externa y las restricciones asociadas (de un sistema) que satisfará dichas necesidades. El análisis y validación del documento de requisitos para asegurar consistencia y viabilidad. (3)
- Un proceso de descubrimiento, refinamiento, modelado y especificación. La Ingeniería de Requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, se analizan sus necesidades, se confirma su viabilidad, se negocia una solución razonable, se especifica la solución sin ambigüedad, se valida la especificación y se gestionan los requisitos para que se transformen en un sistema operacional. (4)

La Ingeniería de Requisitos puede considerarse como un proceso para descubrir lo que el cliente desea, así como lo gestión de los cambios que sean necesarios realizar para dar solución a dichas necesidades.

1.2 Requisitos

Una de las características de la Ingeniería de Requisitos es la falta de uniformidad en la terminología utilizada, uno de los conceptos que presenta este problema es el de requisito. A continuación algunas definiciones del mismo:

- Condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo. (5)
- Característica del sistema que es una condición para su aceptación. (6)
- Condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, especificación u otro documento formal. (7)
- Propiedad que un sistema debería tener para logar el éxito en el entorno en el que se usará. (8)

A pesar de esta aparente simplicidad del concepto, la gran cantidad de calificativos que se aplican al término requisito muestran distintos aspectos que con frecuencia se consideran aisladamente. A partir de las definiciones antes expuestas puede plantearse que los requisitos son las condiciones que el sistema debe cumplir así como las restricciones que debe tener para poder funcionar.

1.2.1 Características de los requisitos

Las características de los requisitos son las propiedades que este debe cumplir. A continuación algunas de las más importantes. (5)

- **Correcto:** Una especificación de requisitos es correcta si, y sólo si, cada requisito declarado se encuentra en el software. No existen herramientas o procedimientos que aseguren la exactitud. Al identificar los requisitos este procedimiento se hace más fácil y hay menos probabilidad al error.
- **Inequívoco:** Cada requisito declarado tiene sólo una interpretación. Se requiere que cada característica de la última versión del producto se describa usando un único término. En casos donde un término en un contexto particular tenga significados múltiples, el término debe ser incluido en un glosario donde su significado se especifique.
- **Comprobable:** Cada requisito declarado es comprobable. Un requisito es comprobable si existe algún proceso rentable finito con el que una persona o la máquina puede verificar que el producto del software reúne el requisito.
- **Modificable:** Su estructura y estilo son tales que puede hacerse cualquier cambio a los requisitos

de forma fácil, completa y consistente mientras conserve la estructura y estilo.

- **Identificable:** El origen de cada uno de sus requisitos está claro si facilita las referencias de cada requisito en el desarrollo futuro o documentación del mismo.

Otras de las características que deben cumplir los requisitos y que no se deben perder de vista son:

- **Especificado por escrito:** Como todo contrato o acuerdo entre dos partes.
- **Posible de probar o verificar:** Si un requisito no se puede comprobar, entonces ¿Cómo se sabe si se cumplió con él o no?
- **No ambiguo:** Un requisito no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición no debe causar confusiones al lector.

1.2.2 Clasificación de los requisitos

Durante el proceso de obtención o levantamiento de requisitos son muchos los que se identifican por las diversas ideas que surgen por parte de los clientes o por el equipo de desarrollo acerca de lo que el sistema debe hacer, estas ideas son las que se analizan como posibles requisitos. Estos requisitos son divididos en dos grandes categorías, las cuales cuentan con características propias que los identifican: Requisitos funcionales y Requisitos no funcionales.

Requisitos funcionales

Los requisitos funcionales son los que definen las funciones que el sistema debe ser capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa el ¿qué? y no el ¿cómo? se deben hacer esas transformaciones. A continuación algunas definiciones de requisitos funcionales:

- Estos requisitos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema. (1)
- Los requisitos funcionales especifican acciones que el sistema debe ser capaz de cumplir, sin tomar en cuenta algún tipo de restricción física. Por lo general se describen a través de casos de uso, obteniéndose las actividades que serán objeto de automatización. (2)

Resumiendo lo visto anteriormente los requisitos funcionales no son más que las condiciones o capacidades que el sistema debe cumplir.

Requisitos no funcionales

Los requisitos no funcionales tienen que ver con las características que de una u otra forma puedan limitar el sistema. Estas características son las que hacen al producto atractivo, usable, rápido y confiable. Normalmente están vinculados a requisitos funcionales, es decir una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Los requisitos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que este cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable es, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

Existen múltiples categorías para clasificar a los requisitos no funcionales, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta: (4)

Requisitos de software

Son las condiciones de software que se deben cumplir para que pueda funcionar la aplicación. Por ejemplo Sistema Operativo Windows.

Requisitos de hardware

Al igual que en la sección anterior se deben enunciar aquí los elementos de hardware de que se disponen, por ejemplo: se requiere disponer de un Modem estándar o una tarjeta digitalizadora de video.

Restricciones en el diseño y la implementación

Este tipo de requerimiento especifica o restringe la codificación o construcción de un sistema. Son restricciones que han sido ordenadas y deben ser cumplidas estrictamente. Ejemplos de ellas son:

- Estándares requeridos.
- Lenguajes de programación a ser usados para la implementación.

- Uso obligatorio de ciertas herramientas de desarrollo.
- Restricciones en la arquitectura o el diseño.
- Bibliotecas de clases.

Requisitos de apariencia o interfaz externa

Este tipo de requerimiento describe la apariencia del producto. Es importante destacar que no se trata del diseño de la interfaz en detalle sino que especifican cómo se pretende que sea la interfaz externa del producto.

Los requisitos de apariencia se vuelven más importantes a medida que los productos de software se mueven hacia áreas más orientadas al consumidor. Productos muy sofisticados como las cámaras digitales, cámaras de video u organizadores personales les dan la impresión a los clientes de ser muy fáciles de operar, sin embargo contienen un gran nivel de funcionalidad.

Requisitos de seguridad

Este es quizás el tipo de requerimiento más difícil, que provocará los mayores riesgos si no se maneja correctamente. La seguridad puede ser tratada en tres aspectos diferentes:

- **Confidencialidad:** La información manejada por el sistema está protegida de acceso no autorizado y divulgación.
- **Integridad:** la información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes; de la misma forma será considerada igual a la fuente o autoridad de los datos. Pueden incluirse también mecanismos de chequeo de integridad y realización de auditorías.
- **Disponibilidad:** Significa que a los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán la obtención de datos por parte de los usuarios.

La seguridad de un sistema no solo tiene en cuenta la seguridad de este propiamente dicho, tiene en cuenta además el ambiente en el que se usará. Es por ello que se tiene que contemplar la seguridad física del lugar donde se usa la aplicación, los controles administrativos que se establecen de acceso al sistema y las regulaciones legales que afectan o determinan el uso de este.

Requisitos de uso

Estos requisitos describen los niveles apropiados de uso dados los usuarios finales del producto, para ello deben revisarse la especificaciones de los perfiles de usuarios y ver las clasificaciones de sus niveles de experiencia.

Debe conocerse:

- ¿Qué tipo de personas son?
- ¿Qué tipo de producto necesitan para realizar su trabajo?
- ¿Qué tipo de requerimiento de uso haría el producto adecuado para ellos?

Los requisitos de uso se derivan de una combinación de lo que el cliente está tratando de lograr con el producto y de lo que los usuarios finales esperan del mismo. Estos elementos deben estar definidos antes de escribirlos.

Requisitos de soporte

Abarcan todas las acciones a realizar una vez que se ha terminado el desarrollo del software con motivo de asistir a los clientes de este, así como lograr su mejoramiento progresivo y evolución en el tiempo.
También se tienen:

- Requisitos legales
- Requisitos de confiabilidad
- Requisitos de interfaz Interna

Después de haber visto una serie de características de los requisitos no funcionales así como las diferentes clasificaciones, se llega a la conclusión que estos no son más que las propiedades o cualidades que el sistema debe cumplir.

1.3 Importancia de la Ingeniería de Requisitos

A pesar de que las herramientas para desarrollar software han ido evolucionando a lo largo de los años, se producen software que no son satisfactorios para los clientes y usuarios. Los principales problemas durante el proceso de producción de un software residen en las primeras etapas de desarrollo, cuando hay

que decidir las características del producto de software a desarrollar.

La parte más difícil de construir un sistema de software es decidir lo que se quiere hacer, ninguna otra parte del trabajo afecta más negativamente al sistema final si se realiza de manera incorrecta. Ninguna otra parte es más difícil de rectificar después.
Aplicar una buena ingeniería de requisitos brinda una serie de beneficios entre los cuales se pueden destacar: (9)

- **Permite gestionar las necesidades del proyecto en forma estructurada:** Cada actividad de la Ingeniería de Requisitos consiste de una serie de pasos organizados y bien definidos.
- **Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados:** La Ingeniería de Requisitos proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.
- **Disminuye los costos y retrasos del proyecto:** Es sabido que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro; especialmente aquellas decisiones tomadas durante la Ingeniería de Requisitos, ya que es una de las etapas de mayor importancia en el ciclo de desarrollo de software y de las primeras en llevarse a cabo.
- **Mejora la calidad del software:** La calidad en el software tiene que ver con cumplir un conjunto de requisitos (funcionalidad, facilidad de uso, confiabilidad, desempeño).
- **Mejora la comunicación entre equipos:** La especificación de requisitos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre, el proyecto no será exitoso.

El costo de una buena recogida de requisitos y análisis del sistema a desarrollar es menor comparado con el costo resultante de tener requisitos pobres. El fundamento básico de cualquier software recae sobre su proceso de Ingeniería de Requisitos. El éxito o fallo del software depende casi siempre de cuán bien se hayan capturado, entendido y usado los requisitos como base para el desarrollo.

1.4 Actividades de la Ingeniería de Requisitos

Dentro de la Ingeniería de Requisitos existen cuatro actividades básicas que se tienen que llevar a cabo para completar el proceso. Estas actividades ayudan a reconocer la importancia que tiene para el desarrollo de un proyecto de software realizar una especificación y administración adecuada de los

requisitos de los clientes o usuarios.

Las cuatro actividades son: extracción, análisis, especificación y validación, y son explicadas a continuación: (9)

- **Extracción:** Esta fase representa el comienzo de cada ciclo. Extracción es el nombre comúnmente dado a las actividades involucradas en la obtención de los requisitos del sistema. Aquí, los analistas de requisitos deben trabajar junto al cliente para descubrir el problema que el sistema debe resolver, los diferentes servicios que el sistema debe prestar y las restricciones que se pueden presentar.
- **Análisis:** Esta fase comienza después de realizada la extracción, la cual se enfoca en descubrir problemas en los del sistema identificados hasta el momento. Usualmente se hace un análisis luego de redactar una versión inicial del documento de requisitos; en esta etapa estos se leen, se conceptualizan, se investigan, se intercambian ideas con el resto del equipo, se resaltan los problemas, se buscan alternativas y soluciones, y luego se programan reuniones con el cliente para discutir los requisitos.
- **Especificación:** En esta fase se documentan los requisitos acordados con el cliente, en un nivel apropiado de detalle. En la práctica, esta etapa se realiza conjuntamente con el análisis, se puede decir que la especificación es el "pasar en limpio" el análisis realizado previamente aplicando técnicas y estándares de documentación, como la notación UML (Lenguaje de Modelado Unificado), que es un estándar para el modelado orientado a objetos, por lo que los casos de uso y la obtención de requisitos basada en casos de uso se utiliza cada vez más para la obtención de estos.
- **Validación:** La validación es la etapa final de la Ingeniería de Requisitos. Su objetivo es ratificar los requisitos, es decir, verificar todos los que aparecen en el documento especificado para asegurar que representan una descripción aceptable del sistema que se debe implementar. Esto implica verificar que sean consistentes y que estén completos.

Se puede apreciar que el proceso de Ingeniería de Requisitos es un conjunto estructurado de actividades, mediante las cuales se obtiene y se logra dar un mantenimiento adecuado al documento de especificación de requisitos, que es el documento final de carácter formal, que se obtiene de este proceso.

Es necesario recalcar que no existe un proceso único que sea válido para aplicar en todas las organizaciones. Cada organización debe desarrollar su propio proceso de acuerdo al tipo de producto que se esté desarrollando, a la cultura organizacional, y al nivel de experiencia y habilidad de las personas involucradas en la ingeniería de requisitos.

1.5 Técnicas utilizadas para la captura de requisitos

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae las necesidades que debe cubrir dicho sistema. El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él.

Por la complejidad que todo esto puede implicar, en la Ingeniería de Requisitos se ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa. Es importante resaltar que estas técnicas pueden ser aplicables a las distintas fases del proceso de la Ingeniería de Requisitos, con la salvedad de que hay que tomar en cuenta las características propias del proyecto en particular que esté desarrollándose para aprovechar al máximo su utilidad. A continuación se presentan un grupo de técnicas que han sido utilizadas en el proceso de desarrollo de software: (10)

- **Entrevistas y Cuestionarios:** Resulta una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. Esta técnica se emplea para reunir información proveniente de personas o de grupos. Durante la entrevista, el analista conversa con el encuestado; el cuestionario consiste en una serie de preguntas relacionadas con varios aspectos de un sistema. Por lo común, los encuestados son usuarios de los sistemas existentes o usuarios en potencia del sistema propuesto. En algunos casos, son gerentes o empleados que proporcionan datos para el sistema propuesto o que serán afectados por él. El éxito de esta técnica, depende de la habilidad del entrevistador y de su preparación para la misma.
- **Lluvia de ideas:** Es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy

numeroso (máximo 10 personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador. Como técnica de captura de requisitos es sencilla de usar y de aplicar.

- **Casos de Uso:** Aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores proponen casos de uso como técnica para la captura de requisitos. Los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas, otros sistemas, etc.) que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores. La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades.
- **Prototipos:** Durante la actividad de extracción de requisitos, puede ocurrir que algunos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los requisitos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final. Para validar los requisitos obtenidos, se construyen prototipos. Los prototipos son simulaciones del posible producto, que luego son utilizados por el usuario final, lo que permite conseguir una retroalimentación importante en cuanto a si el sistema diseñado con base a los requisitos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva. El desarrollo del prototipo comienza con la captura de requisitos. Desarrolladores y clientes se reúnen y definen los objetivos globales del software, identifican todos los requisitos que son conocidos, y señalan las áreas en las que será necesaria la profundización en las definiciones.
- **Mapas Conceptuales:** Los mapas conceptuales son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la Ingeniería de Requisitos pues son fáciles de entender por el usuario. Sin embargo, deben ser usados con cautela porque en algunos casos pueden ser muy subjetivos y llegan a ser ambiguos en casos complejos si no se acompaña de una descripción textual.

1.6 Indicadores

En la vida cotidiana se encuentran un gran número de aspectos los cuales se comportan como variables, es decir, tienen características que cambian de modo tanto cualitativo como cuantitativo; dada la necesidad de describir la información sobre las variables que se observan es que surgen los indicadores. En la actualidad se puede apreciar una gran variedad de conceptos al definir el término indicador. A continuación se muestran algunas de estas definiciones:

- Es una medida que puede usarse como guía para controlar y valorar la calidad de las diferentes actividades. (11)
- Son elementos del sistema de control de gestión, que proporciona información significativa sobre aspectos críticos o claves de una organización mediante la relación de dos o más datos. (11)
- Un indicador es un instrumento a partir del cual se registra, procesa y presenta la información necesaria para medir el avance o retroceso en el logro de un determinado objetivo. (12)
- Un indicador es un valor mensurable que permite seguir la evolución de un proceso para identificar el logro de un objetivo. (13)

Después de estudiar algunas de las clasificaciones del término indicador se llega a la conclusión que este no es más que un criterio para valorar, analizar y evaluar el comportamiento de variables, planificando y tomando decisiones a partir de estos.

1.6.1 Características de los indicadores

Los indicadores poseen una gran cantidad de características, a continuación algunas de las más importantes: (14)

- **Validez:** Deben reflejar y medir los efectos y resultados.
- **Demostrables:** Deben evidenciar los cambios buscados.
- **Relevancia:** Deben servir efectivamente al usuario para la toma de decisiones.
- **Representatividad:** Deben expresar efectivamente el significado que los actores le otorgan a determinada variable.
- **Confiabilidad:** Las mediciones que se realicen por diferentes personas deben arrojar los mismos resultados.

- **Sensibilidad:** Deben reflejar el cambio de la variable en el tiempo, es decir, debe cambiar de forma efectiva y persistente a lo largo del período de análisis.
- **Eficiencia:** Deben ser exactos al expresar el fenómeno.
- **Suficiencia:** Deben expresar el fenómeno sin ser redundantes.
- **Flexibilidad:** Deben adecuarse a la realidad de lo que se pretende medir.

1.6.2 Clasificación de los indicadores

Existen distintos tipos de indicadores, los cuales cuentan con diversas aplicaciones ya que pueden utilizarse en muchas de las esferas de la vida. Es válido aclarar que estos deben cumplir las características mencionadas anteriormente. A continuación la clasificación de los indicadores: (14)

- **Indicadores cuantitativos:** Son los que se refieren directamente a medidas en números o cantidades.
- **Indicadores cualitativos:** Son los que se refieren a cualidades. Se trata de aspectos que no son cuantificados directamente, como opiniones, percepciones o juicios emitidos sobre algo.
- **Indicadores directos:** Son aquellos que permiten una dirección directa del fenómeno.
- **Indicadores indirectos:** cuando no se puede medir de manera directa la condición económica, se recurre a indicadores sustitutivos o conjuntos de indicadores relativos al fenómeno que se desea medir o sistematizar.
- **Indicadores positivos:** Son aquellos en los cuales si se incrementa su valor estarían indicando un avance hacia la equidad.
- **Indicadores negativos:** Son aquellos en los cuales si su valor se incrementa estarían indicando un retroceso hacia la inequidad.

1.6.3 Importancia de los indicadores de calidad

Los indicadores tienen gran importancia ya que forman parte de una de las tantas formas que existen para medir la calidad del software. El objetivo de estos es aportar a la empresa un camino correcto para que esta logre cumplir con las metas establecidas. Entre las ventajas fundamentales se encuentran: (15)

- Permiten medir cambios a través del tiempo.
- Facilitan mirar de cerca los resultados de iniciativas o acciones.
- Son instrumentos muy importantes para evaluar y dar surgimiento al proceso de desarrollo.
- Son instrumentos valiosos para orientar cómo se pueden alcanzar mejores resultados en proyectos de desarrollo.

Los indicadores representan importantes herramientas para la toma de decisiones ya que transmiten información que permite transformarla en acción, por lo que resultan fundamentales para evaluar y predecir tendencias, así como para valorar el cumplimiento de las metas y objetivos fijados en el proyecto. Por ello cumplen una función activa en el mejoramiento de los procesos de formulación, rediseño y seguimiento de la información.

1.7 Herramientas, metodología y tecnologías propuestas

1.7.1 Metodología de desarrollo

Actualmente en el mundo existe una gran variedad de metodologías de desarrollo de software, las cuales son utilizadas en una infinidad de áreas; entre las más utilizadas se pueden mencionar: Rational Unified Process (RUP), SCRUM y Extreme Programming (XP).

RUP

RUP es un proceso de desarrollo del software orientado a objetos, el cual proporciona el conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema de software, es además un marco de trabajo genérico, lo cual permite que pueda ser utilizado en una gran variedad de sistemas de software.

Objetivos (7)

- Proporcionar una guía para ordenar las actividades de un equipo.
- Dirigir las tareas de cada desarrollador por separado y del equipo como un todo.
- Especificar los artefactos que deben desarrollarse.
- Ofrecer criterios para el control y la medición de los productos y las actividades del proyecto.

Características fundamentales. (7)

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requisitos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- **Iterativo y que permite incrementos:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

RUP está divido en 4 fases: inicio, elaboración, construcción y transición; además cuenta con 6 flujos de trabajo ingenieriles así como con 3 de soporte, estos flujos van haciendo iteraciones dentro de las fases realizando revisiones completas de los artefactos desarrollados, esto permite mitigar tempranamente los riesgos además de poder ver los resultados de cada iteración.

XP (Programación Extrema)

XP es una metodología de desarrollo ligera. Está pensada para obtener un software eficiente en un reducido tiempo de trabajo después de estar en constantes cambios los requisitos del software. A continuación se muestran algunas características que diferencian a XP de las demás metodologías: Está concebida para ser usada dentro de proyectos pequeños y de corto plazo.

Se basa en equipos de desarrollos pequeños; (2 a 10) personas trabajando en un mismo sitio y con pocos roles, lo que facilita el intercambio de responsabilidades y el trabajo en conjunto. Consiste en una programación rápida o extrema que está basada en la sencillez, la comunicación y la realimentación o reutilización del código desarrollado. Puesto que puede ser un proyecto para uso interno, aunque también puede utilizarse para otros proyectos. La arquitectura del proyecto se define y mejora a lo largo de su desarrollo.

El cliente forma parte del equipo de desarrollo. Con la utilización de XP la comunicación con el cliente es a través de diálogos cara a cara, el cual es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo y lograr un producto que satisfaga sus necesidades. (16)

SCRUM

SCRUM define un marco para la gestión de proyectos que se ha utilizado con éxito durante los últimos años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: El desarrollo de software se realiza mediante iteraciones, denominadas sprint con una duración entre 15 y 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente.

La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria (Daily Scrum Meeting) de 15 minutos, donde cada miembro del equipo contesta 3 preguntas: ¿Qué has hecho desde ayer? ¿Qué es lo que estás planeando hacer hoy? ¿Has tenido algún problema que te haya impedido alcanzar tu objetivo? (17)

SCRUM es un proceso ágil que se enfoca en la entrega del mayor valor de negocio en el menor plazo, teniendo en cuenta las funcionalidades priorizadas por el cliente. Permite inspeccionar software listo para ser liberado en forma rápida y continua (cada dos a cuatro semanas). Por tanto cada dos a cuatro semanas cualquiera puede ver el software funcionando y decidir liberarlo como está o continuar mejorándolo durante otra iteración.

Fundamento de la selección

A pesar de ser muy ágil en el proceso de desarrollo de software debido a que utiliza el mínimo de documentación, XP tiene como desventaja que el cliente debe de estar fuertemente ligado al equipo de desarrollo, en este caso el cliente no se encuentra dentro del mismo. Por su parte con Scrum es difícil hacer cambios una vez comenzada una iteración, generalmente hay que esperar a que concluya para poder realizar cambios.

Una vez analizadas las diferentes metodologías, se selecciona RUP para el desarrollo del presente trabajo ya que es la que más se adapta a las necesidades del equipo de desarrollo. Es una metodología que posibilita una mitigación temprana de los riesgos y permite que en cada iteración se realicen revisiones

completas de los artefactos desarrollados.

1.7.2 Lenguaje de modelado

Para desarrollar el análisis y diseño del software se decidió utilizar el Lenguaje unificado de modelado de software (UML) ya que está fuertemente vinculado a la metodología seleccionada. Este modelo fue creado por Grady Booch, James Rumbaugh e Ivar Jacobson y permite el modelado de sistemas con tecnología orientada a objetos. (18)

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema software. Captura decisiones y conocimientos que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios.

La decisión de utilizar UML como notación para el desarrollo del software se debe a que se ha convertido en un estándar que tiene las siguientes características: (7)

- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo.

1.7.3 Herramientas CASE

Las herramientas CASE son una gran variedad de aplicaciones informáticas cuyo objetivo es garantizar que se alcancen la consistencia, completitud y cumplimiento de los estándares. Estas se utilizan para mejorar la calidad mediante un entorno interactivo y para automatizar e integrar las tareas de las distintas etapas del ciclo de vida. Entre las herramientas más utilizadas a nivel mundial se pueden mencionar Rational Rose y Visual Paradigm.

Rational Rose

Es una de las herramientas más poderosas para el modelado visual, es utilizada para el análisis y diseño de sistemas basados en objetos. Se utiliza para modelar un sistema antes de proceder a construirlo.

Características (19)

- Mantiene la consistencia de los modelos del sistema software.
- Chequeo de la sintaxis UML.
- Ingeniería Inversa (crear modelo a partir código).
- Generación de código a partir de los modelos.
- Generación de la documentación automáticamente.

Es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas es que utiliza la notación estándar en la arquitectura de software UML, la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común.

Visual Paradigm

Visual Paradigm para UML soporta el ciclo de vida completo del desarrollo de software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas, además permite generar reportes y documentación.

Características (20)

- Soporte de UML.
- Diagramas de Procesos de Negocio (Proceso, Decisión, Actor de negocio, Documento).
- Ingeniería inversa Java.
- Generación de código (Modelo a código, diagrama a código).
- Editor de Detalles de Casos de Uso, para la especificación de los detalles de los casos de uso.
- Diagramas de flujo de datos.
- Soporte ORM (Generación de objetos Java desde la base de datos).

Esta herramienta a diferencia de la anterior es software libre, lo cual como ya se vio anteriormente es de gran importancia para el país y para la universidad, por el proceso de migración en que ambos se encuentran.

Fundamento de la selección

Durante el estudio realizado sobre las herramientas de modelado se ha podido determinar que ambas poseen muchas ventajas para realizar el modelado de un software, sin embargo Rational Rose presenta como desventaja que es propietario, por lo que no sería recomendable utilizarlo para el desarrollo de la aplicación, pues en la actualidad se aboga por la soberanía tecnológica y la utilización de software gratuitos.

Se selecciona Visual Paradigm ya que este soporta todo el ciclo de vida de desarrollo del software contribuyendo así a un mejor producto y proporciona a los desarrolladores una plataforma con interfaz amigable que les permite diseñar un producto con calidad de forma muy rápida.

1.7.4 Herramienta para el almacenamiento de datos

Estas herramientas permiten guardar datos de un mismo contexto y almacenarlos para su posterior uso, lo que posibilita acceder a los datos de forma rápida y estructurada. Mundialmente existen varios tipos de herramientas para el almacenamiento de datos, a continuación se describen dos de las más usadas.

MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL está escrito en una mezcla de C y C++.

Características de MySQL: (21)

- Múltiples motores de almacenamiento permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.
- Agrupación de transacciones, reúne múltiples transacciones de varias conexiones para incrementar el número de ellas por segundo.

PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD. Permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Esta estrategia es superior al uso de bloqueos por tabla o por filas, común en otras bases de datos, y elimina la necesidad del uso de bloqueos explícitos.

Características (22)

- Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas.
- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

Fundamento de la selección

Se selecciona PostgreSQL ya que es un sistema gestor de bases de licencia libre a diferencia de MySQL. Presenta una gran capacidad de almacenamiento, alta ocurrencia y amplia variedad de tipos nativos. Además implementa el uso de subconsultas y transacciones haciendo su funcionamiento mucho más eficaz, ofreciendo soluciones en campos en las que MySQL no podría.

1.7.5 Servidor web

Apache

Es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a las últimas versiones de protocolos. Es capaz de funcionar en la mayoría de las plataformas y entornos existentes. Está entre los servidores web más reconocidos a nivel internacional, debido a la capacidad de correr en múltiples sistemas operativos. Apache es una tecnología gratuita de código abierto y altamente configurable, de diseño modular capaz de interpretar diversos lenguajes de programación. El servidor Apache está estructurado en módulos, los cuales pueden clasificarse en tres categorías específicas: (23)

- **Módulos Base:** Son funciones básicas del Apache.
- **Módulos Multiproceso:** Responsables de la unión con los puertos de la máquina, aceptando las

peticiones y enviando a los hijos a atender las peticiones.

- **Módulos Adicionales:** Cualquier otro módulo que le adicione una funcionalidad al servidor se puede añadir sin necesidad de volver a instalar el software.

Características de Apache

- Multiplataforma
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Basado en hebras en la versión 2.0
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla de forma abierta
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.

Internet Information Services

Es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows. Originalmente era parte del paquete de opciones para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS. Estos servicios proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor web.

El servidor web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas. Por ejemplo, Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl (23).

Características de Internet Information Services

- Autenticación de texto implícita avanzada
- Comunicaciones seguras:
- Cifrado canalizado por servidor (SGC):
- Asistentes para seguridad:
- Restricciones de dominio de Internet e IP:
- Almacenamiento de certificados:
- Almacenamiento en caché de plantillas
- Mensajes de error personalizados mejorados:
- Administración remota:
- Administración centralizada

Fundamento de la selección

Se decide utilizar Apache ya que es un servidor robusto, estable y de muy fácil configuración. Tiene soporte para host virtuales y posibilita el uso de transacciones seguras. Además de que es un servidor web HTTP de código abierto a diferencia del Internet Information Services que es de licencia propietaria. Apache cuenta con una licencia BSD (Berkeley Software Distribution) la cual es permisiva y da la posibilidad de usar el código fuente en software tanto libre como no libre.

1.7.6 Entornos de desarrollo integrados (IDE)

Un IDE es una aplicación compuesta por un conjunto de herramientas útiles para un desarrollador. Puede ser exclusivo para un lenguaje de programación o utilizarse para varios. Está compuesta por un editor de código (con facilidades como resaltado de sintaxis, completamiento de código y navegación entre clases), un compilador y herramientas de automatización de la compilación, un depurador y en algunos casos un constructor de interfaz gráfica.

NetBeans

NetBeans es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portable entre las distintas plataformas, haciendo uso de la tecnología Java. Dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones Web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles y funcionalidades ampliables mediante la instalación de paquetes.

Soporta varios lenguajes, entre ellos: (24)

- Java
- C/C++
- Ruby on Rails
- PHP

Características de NetBeans: (24)

- Instalación y actualización simple.
- Diseñador visual de formularios para Swing GUI.
- Características visuales para el desarrollo web.
- Creador gráfico de juegos para celulares.

ZendStudio

Zend Studio consta de dos partes en las que se dividen las funcionalidades, la parte del cliente y la del servidor. Las dos partes se instalan por separado, la del cliente contiene la interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración.

Características (25)

- No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- Soporte para PHP 4 y PHP 5.
- Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de

funciones y métodos de clase.

- Plegado de código (comentarios, bloques de phpDoc, cuerpo de funciones y métodos e implementación de clases).
- Sangrado automático y otras ayudas de formato de código.
- Detección de errores de sintaxis en tiempo real.
- Soporte para control de versiones usando CVS o Subversion.
- Soporte para navegación en bases de datos y ejecución de consultas SQL.

Fundamento de la selección

Después de haber realizado esta comparación entre los IDE de Desarrollo se puede apreciar que ambos son potentes en el desarrollo de aplicaciones web, sin embargo el ZendStudio tiene como desventaja que es de carácter propietario. Por lo que se decidió utilizar como IDE de Desarrollo NetBeans ya que es un reconocido entorno de desarrollo integrado en múltiples plataformas, además de ofrecer soporte a otros lenguajes como HTML y JavaScript que también son utilizados a lo largo del desarrollo de la aplicación.

1.7.7 Lenguajes de programación

Partiendo de las características de la aplicación, se hace necesaria la selección de un lenguaje mediante el cual se pueda cumplir con los requisitos propuestos. Actualmente existen muchos lenguajes para el desarrollo de aplicaciones web surgidos a partir de las tendencias y necesidades de los escenarios. El análisis se centró fundamentalmente en los lenguajes Procesador de Hipertexto (PHP) y Java.

Procesador de Hipertexto (PHP)

PHP es un lenguaje interpretado de alto nivel que se ejecuta en servidores Web y permite la generación de páginas dinámicas. Puede ser desplegado en muchos servidores Web y en casi todos los sistemas operativos y plataformas sin costo alguno. Al ser un lenguaje libre, dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas Web.

Características fundamentales del lenguaje PHP: (26)

- Soporte de conexiones a bases de datos tales como: MySQL, PostgreSQL, Oracle, entre otras.
- Interactúa con uno de los servidores de Web más potentes como el Apache.

- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML
- Utiliza las sesiones de HTTP, conectividad de Java, LDAP, SNMP e IMAP.
- Tiene manejo de excepciones.

JAVA

Es un lenguaje de programación sencillo, orientado a objetos, de propósito general e independiente de la plataforma de desarrollo. Este lenguaje posibilita la "Integración externa", como el uso de herramientas, métodos y funcionalidades desarrolladas por otros programadores. Esto supone una ventaja tanto en el ámbito del desarrollo como en la repercusión final de un proyecto.

Las características principales son: (27)

- **Es robusto:** Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos ayuda a detectar errores lo antes posible, en el ciclo de desarrollo.
- **Es de arquitectura neutral:** Para establecer Java como parte integral de la red, el compilador compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará.
- **Es dinámico:** Se beneficia todo lo posible de la tecnología orientada a objetos. No intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución.
- **Es seguro:** El sistema de Java tiene ciertas políticas que evitan que se puedan codificar virus con este lenguaje.
- **Es Multihilo:** Un lenguaje que soporta múltiples hilos, es decir puede ejecutar diferentes líneas de código al mismo tiempo.

Fundamento de la selección

Por las ventajas que posee y las facilidades que proporciona relacionadas con el desarrollo y rendimiento para sistemas con características similares al de este trabajo, se decide emplear PHP en su versión 5.0 como lenguaje de programación para el desarrollo de los módulos, así como también porque se cuenta con suficiente experiencia con el uso del mismo.

1.7.8 Marco de trabajo o Frameworks

Un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, que puede servir de base para que otro proyecto se organice y desarrolle. Típicamente puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. (28)

Dojo

Nace en el 2004 gracias a un aumento de la demanda por aplicaciones web con mejores características tanto de diseño como de uso, además de esto hubo otros factores que influyeron en el nacimiento de este Framework entre los cuales se pueden encontrar: (28)

- Incompatibilidades en el cumplimiento de los estándares por parte de los Navegadores.
- El surgimiento de AJAX generó nuevos desafíos para los desarrolladores y diseñadores.
- La web 2.0 generó nuevas oportunidades tanto para el desarrollo de servicios como para la evolución de las tecnologías.

Características:

- Comunicación asíncrona.
- Sistema de paquetes.
- Almacenamiento en el cliente de datos.
- Almacenamiento en el servidor.
- Manipulación de DOM.
- Animaciones.
- Manejo de Eventos.
- Gráficos.

Se decide utilizar Dojo ya que como se pudo observar anteriormente, este cuenta con una serie de características que pueden ser utilizadas para enriquecer sitios web; cuenta con varias librerías totalmente software libre las cuales permiten una fácil adapatación a la herramienta a utilizar.

Conclusiones del Capítulo

En este capítulo se abordaron algunos conceptos referentes a la Ingeniería de Requisitos para lograr un mejor entendimiento de lo que se quiere realizar. Se analizaron las tendencias y tecnologías actuales ofreciendo las definiciones necesarias para comprender las herramientas que serán utilizadas en el desarrollo de la aplicación. Se utilizará RUP como metodología de desarrollo, la cual utiliza como lenguaje de modelado UML y como herramienta de modelado Visual Paradigm. Se utilizará PHP como lenguaje de programación y el entorno de desarrollo será NetBeans, el cual será integrado al framework Dojo y al gestor de base de datos PostgreSQL.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

Introducción

En este capítulo se describen los requisitos funcionales y requisitos no funcionales de la aplicación a desarrollar. Se realiza una descripción de los actores que interactúan con la herramienta así como de los casos de usos identificados. Se presenta el diagrama de clases del modelo de dominio, el diagrama de casos de uso del sistema y el diagrama de casos de uso arquitectónicamente significativos.

2.1 Descripción del sistema

La herramienta que se propone desarrollar es una aplicación web que servirá de guía a las empresas de software durante todo su proceso de desarrollo para de esta forma facilitar y hacer más eficiente el trabajo de ellas. Esta herramienta tiene la ventaja que es totalmente gratuita, además de ser multiplataforma, es decir que se puede acceder a ella desde cualquier puesto de trabajo. La aplicación brindará el acceso a las diferentes actividades del proceso de Ingeniería de Requisitos, así como los roles y los documentos asociados a estas actividades permitiendo su visualización o descarga.

Además la herramienta permitirá el cálculo de tres indicadores: el por ciento de utilidad del proceso, el por ciento de inestabilidad de los requisitos y el indicador elementos inconsistentes, lo que permitirá reflejar el estado del proceso de requisitos y de esta forma hacer más fácil el desarrollo del proyecto, estos indicadores se representarán de forma gráfica para hacer más fácil su entendimiento.

2.2 Modelo del dominio

El Modelo de Dominio es una representación visual estática del entorno real del problema, incluye el vocabulario del dominio significativo desde el punto de vista de la arquitectura para de esta forma ayudar al entendimiento de ella. (4)

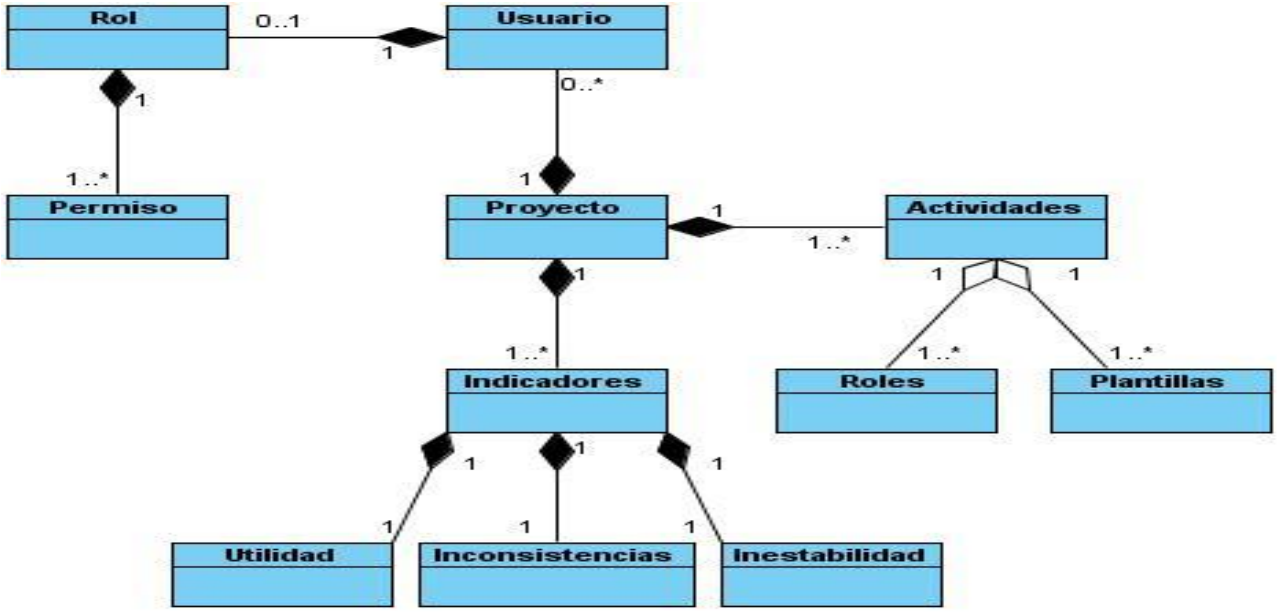


Figura 1 Modelo del dominio

Para un mejor entendimiento se describen cada una de las clases existentes en el diagrama de clases del dominio.

- **Usuario:** Se encarga de gestionar las personas que interactúan con el sistema.
- **Rol:** Permite que los usuarios o actores tengan un calificativo de acuerdo a la especialidad que desempeñan.
- **Proyecto:** Permite agrupar todas las funcionalidades del sistema.
- **Permiso:** Permite a los actores o usuarios del sistema acceder a determinadas funcionalidades.
- **Indicadores:** Representa los diferentes indicadores para medir la calidad del proceso de requisitos.
- **Plantillas:** Se encarga de agrupar los documentos que se utilizan durante las actividades del área de procesos Ingeniería de Requisitos.
- **Actividades:** Representa las diferentes actividades del área de proceso Ingeniería de Requisitos.
- **Utilidad:** Representa al indicador utilidad del proceso.
- **Inestabilidad:** Representa al indicador inestabilidad de los requisitos.
- **Inconsistencias:** Representa al indicador elementos inconsistentes.

2.3 Especificación de los requisitos de software

2.3.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Definen las funciones que el sistema será capaz de realizar.

RF1 – Autenticar Usuario.

RF2 – Gestionar Usuario.

RF2.1 – Insertar Usuario.

RF2.2 – Eliminar Usuario.

RF2.3 – Modificar Usuario.

RF2.4 – Listar Usuario.

RF3 – Visualizar información de las actividades del proceso.

RF4 – Visualizar información de los Roles.

RF5 – Visualizar y descargar plantilla.

RF6 – Visualizar información sobre los Indicadores de calidad.

RF7 – Adicionar respuesta a la encuesta.

RF8 – Adicionar inconsistencias.

RF9 – Modificar estado inconsistencia.

RF10 –Graficar indicadores**.**

RF10.1- Graficar indicador utilidad del proceso.

RF10.2- Graficar indicador inestabilidad de los requisitos.

RF10.3- Graficar indicador elementos inconsistentes.

RF11 – Incrementar cambios a los requisitos.

RF12 – Mostrar revisiones.

RF13 – Adicionar revisión.

2.3.2 Requisitos no funcionales

Son las restricciones que afectan a los servicios o funciones del sistema, tales como restricciones de tiempo, definición de estándares y otras condiciones necesarias para que el sistema pueda funcionar.

Son propiedades o cualidades que el producto debe tener, estas son las que hacen al producto atractivo, usable, rápido y confiable.

- **Apariencia o interfaz externa.**

  La aplicación deberá tener una interfaz externa sencilla, amigable y fácil de entender para el usuario, de esta forma se evita que este se pierda dentro de la aplicación. Además su funcionamiento debe ser de fácil comprensión para todo tipo de usuario.

- **Uso**

  El sistema podrá ser usado por cualquier persona con conocimientos básicos de computación, específicamente sobre páginas web, las interfaces deben garantizar un intercambio de datos de manera fácil para el usuario.

- **Seguridad:**
  - **Confidencialidad**
    - La autenticación será una de las acciones del usuario en el sistema y consistirá en proveer un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica.
  - **Integridad**
    - Se debe garantizar que la información sensible sólo pueda ser vista por los usuarios con el nivel de acceso adecuado y que las funcionalidades del sistema se muestren de acuerdo al usuario que esté activo.
  - **Disponibilidad**
    - El sistema debe estar disponible para su utilización las 24 horas del día, durante los siete días de la semana, con el menor tiempo posible de recuperación ante fallos.
- **Software**

  Al ser una aplicación cliente servidor, los requisitos de software de la herramienta son diferentes para cada una de las partes. En el lado del cliente debe existir un navegador web que soporte Java Script. Por el lado del servidor debe estar instalado PostgreSQL como gestor de Base de Datos y

Apache como servidor web.

- **Hardware**

  Se debe contar con 256 MB de memoria RAM como mínimo, aunque lo ideal sería 512 MB o superior, y el procesador debe ser un Pentium II o superior en el caso de las PC clientes. Para el caso del servidor web se debe contar con 1 GB de memoria RAM y el procesador debe ser un Pentium IV o superior.

2.4 Diagrama de Casos de Uso del sistema

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema, de esta forma se representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa. (7)

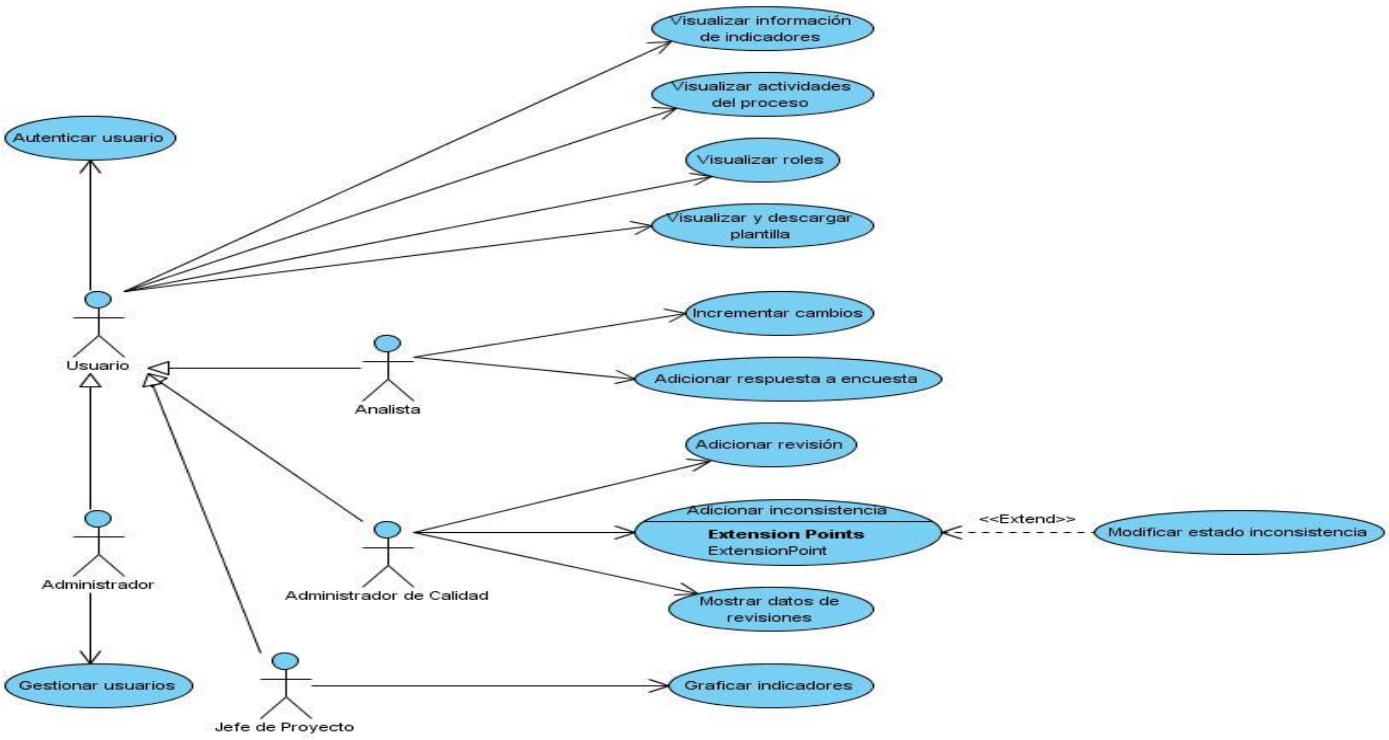


Figura 2 Diagrama de Casos de Uso del sistema

2.5 Definición de los Casos de Uso del sistema

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad de funcionalidad coherente, son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor. (4)

2.5.1 Actores del sistema

Los actores pueden ser sistemas, hardware externo o personas que interactúan con el sistema, ya sea para inicializar una funcionalidad o brindarle información al mismo. Son los que se benefician de las funcionalidades dentro de los casos de uso.

Nombre del Actor	Descripción
Analista	Se refiere al analista de sistema de cualquier proyecto de desarrollo de software, tendrá acceso a funcionalidades del sistema que usuarios comunes no tendrán.
Administrador	Es el encargado de gestionar los usuarios, así como la configuración de los permisos para acceder a determinadas funcionalidades.
Usuario	Se refiere a cualquier persona que quiera acceder a la aplicación para consultar la descripción de las actividades del proceso de ingeniería de requisitos, de los roles y plantillas asociadas a dicho proceso.
Administrador de la Calidad	Se refiere al Administrador de la Calidad de cualquier proyecto de desarrollo de software, tendrá a acceso a funcionalidades del sistema que otros usuarios no tendrán.
Jefe de Proyecto	Se refiere al Jefe de Proyecto de cualquier proyecto de desarrollo de software, tendrá acceso a todas las funcionalidades del sistema

Tabla 1 Descripción de los actores del sistema

2.5.2 Descripción de los Casos de Uso del sistema

La descripción textual de los casos de uso describe paso a paso la interacción entre el actor y el sistema, es cómo responde el sistema ante las peticiones de los diferentes actores. A continuación la descripción textual del caso de uso Graficar Indicadores.

Caso de uso Graficar Indicadores

Caso de Uso:	Graficar Indicadores
Actores:	Jefe de Proyecto
Resumen:	El caso de uso se inicia cuando el jefe de proyecto trata de realizar algunas de las siguientes secciones. **Inestabilidad en los requisitos:** El sistema grafica el por ciento de inestabilidad de cambios de los requisitos. **Utilidad del proceso:** El sistema grafica el por ciento de utilidad del proceso de acuerdo a las respuestas de la encuesta. **Elementos Inconsistentes:** El sistema grafica la cantidad de elementos inconsistentes e inconsistencias encontrados de acuerdo a la revisión en que se hayan realizado.
Precondiciones	Tener los privilegios de autenticación requeridos
Referencias	RF 10.1, RF 10.2, RF 10.3
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1.1 El jefe de proyecto selecciona en el sistema la opción a realizar.	1.2. La aplicación muestra una interfaz para que el Jefe de proyecto seleccione la acción a realizar. a) Si selecciona la opción Inestabilidad de los requisitos ir a la **Sección "Inestabilidad de los requisitos"** b) Si selecciona la opción Utilidad del Proceso ir a la **Sección "Utilidad del Proceso"** c) Si selecciona la opción Elementos Inconsistentes ir a la **Sección "Elementos Inconsistentes"**
Sección "Inestabilidad de los requisitos"	
Acción del Actor	**Respuesta del Sistema**
1.1 El jefe de proyecto selecciona la opción Obtener resultados dentro del menú Inestabilidad de los Requisitos	1.2. Muestra la interfaz para elegir los requisitos a graficar.
1.3 El jefe de proyecto selecciona los requisitos a los que desea acceder. Habilitándose la opción 'Resultado' una vez que el jefe de proyecto seleccione al menos un requisito.	1.4 La aplicación muestra la gráfica del por ciento de inestabilidad de los requisitos de acuerdo a los requisitos previamente seleccionados.
Sección "Utilidad del Proceso"	
Acción del Actor	**Respuesta del Sistema**
2.1 El jefe de proyecto selecciona la opción Obtener resultados dentro del menú Utilidad del Proceso.	2.2. Muestra la interfaz donde se elige la encuesta para obtener sus resultados.

2.3 El jefe de proyecto selecciona la encuesta deseada.	2.4 Muestra la gráfica del por ciento de utilidad del proceso de acuerdo a la encuesta previamente seleccionada.
Sección "Elementos Inconsistentes"	
Acción del Actor	**Respuesta del Sistema**
3.1 El jefe de proyecto selecciona la opción Obtener resultados dentro del menú Elementos Inconsistentes.	3.2. La aplicación muestra una gráfica del número de elementos inconsistentes e inconsistencias identificados en cada revisión.
Poscondiciones	Se grafica el indicador en dependencia de la opción elegida por el usuario.

Tabla 2 Descripción del Caso de Uso Graficar Indicadores

2.6 Vista de Casos de Uso arquitectónicamente significativos

Esta vista describe los procesos más significativos, presenta los actores y los casos de uso más importantes para el sistema, además presenta la percepción que tiene el usuario de las funcionalidades del sistema. En ella se presentan los procesos del negocio más importantes para el sistema y los casos de uso críticos que se derivan de estos. (9)

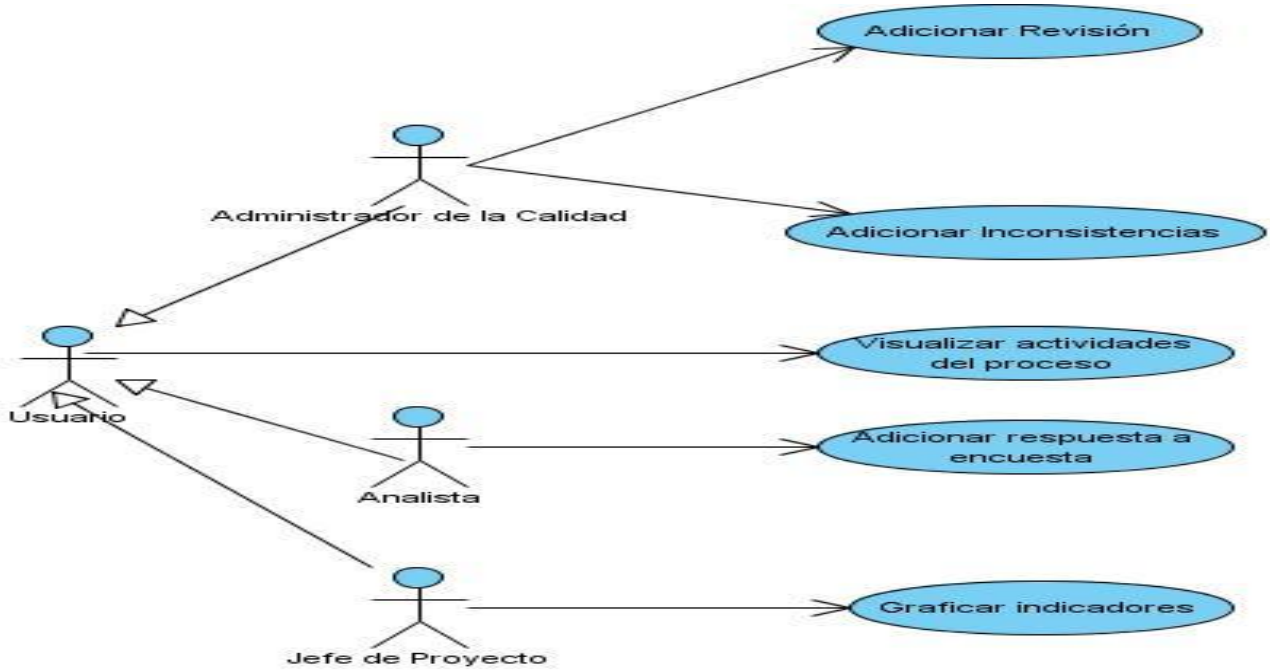


Figura 3 Vista de Casos de Uso

Conclusiones

En este capítulo se identificaron los requisitos funcionales y no funcionales, así como los actores del sistema. Para una mejor comprensión de las funcionalidades de la aplicación se realizó la descripción textual de uno de los Casos de Uso de mayor significación para la aplicación. Además se representó el diagrama de Casos de Uso del sistema y la Vista de Casos de Uso.

CAPÍTULO 3. DISEÑO DEL SISTEMA

Introducción

En este capítulo se traducen los requisitos a una especificación que describe cómo implementar el sistema, dando paso al diseño de la aplicación generando los artefactos necesarios para cada fase. Además se realizan los diagramas de clases y de interacción para uno de los casos de usos más relevantes definidos en el capítulo anterior. También se muestra el modelo datos, la vista de despliegue y se explican los principales patrones arquitectónicos y de diseño utilizados.

3.1 Diagrama de clases del diseño

Los diagramas de clases del diseño se utilizan para modelar la vista de diseño estática de un sistema, estos describen gráficamente las especificaciones de las clases además de visualizar las relaciones entre estas.

A continuación se muestra el diagrama de clases del diseño para el caso de uso Graficar Indicadores.

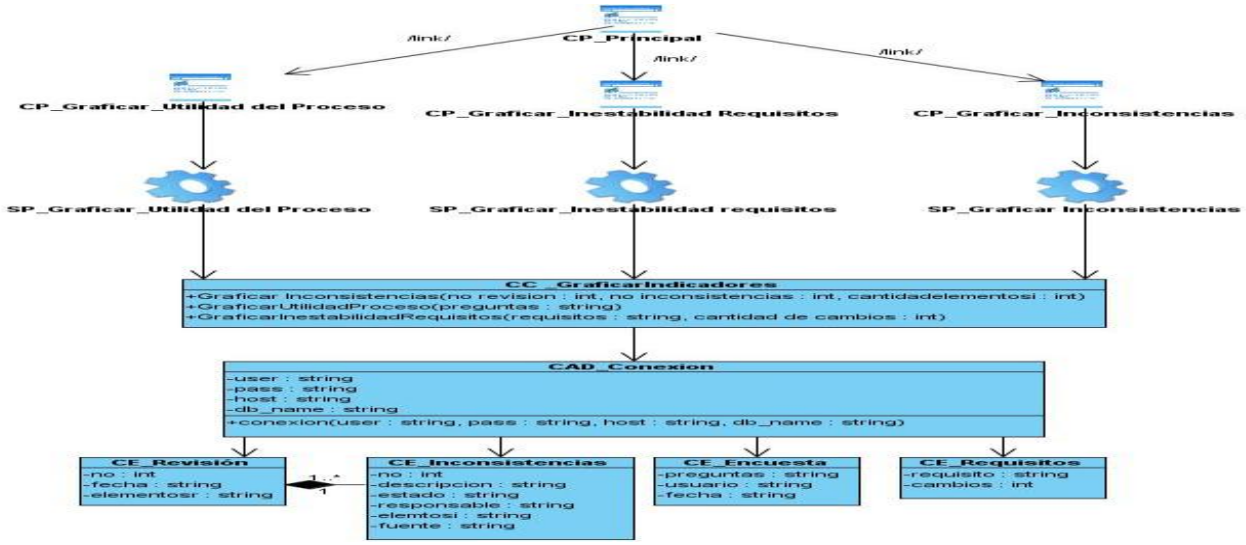


Figura 4 Diagrama de clases del diseño Graficar Indicadores

Descripción de las principales clases: Graficar Indicadores.

- **CP_Principal**: Es la interfaz principal de la aplicación a partir de esta se puede acceder al resto de las interfaces.
- **CP_Graficar_Utilidad del Proceso:** Es la interfaz donde se muestra el gráfico correspondiente al indicador Utilidad del Proceso.
- **CP_Graficar_Inestabilidad de los Requisitos:** Es la interfaz donde se muestra el gráfico correspondiente al indicador Inestabilidad de los Requisitos.
- **CP_Graficar_Inconsistencias:** Es la interfaz donde se muestra el gráfico correspondiente al indicador Elementos Inconsistentes.
- **SP_Graficar_Utilidad del Proceso:** Esta clase tiene la responsabilidad de ejecutar todas las funcionalidades relacionadas con el indicador Utilidad del Proceso.
- **SP_Graficar_Inestabilidad de los Requisitos:** Esta clase tiene la responsabilidad de ejecutar todas las funcionalidades relacionadas con el indicador Inestabilidad de los Requisitos.
- **SP_Graficar_Inconsistencias:** Esta clase tiene la responsabilidad de ejecutar todas las funcionalidades relacionadas con el indicador Inconsistencias.
- **CC_GraficarIndicadores:** Esta clase contiene las principales funcionalidades para poder graficar los diferentes indicadores.

Nombre: CC_ GraficarIndicadores	
Tipo de Clase: Controladora.	
Funcionalidades	
Nombre	**Descripción**
+Graficar_Inconsistencias(nrevisión, ninconsistencias,elementosi)	Permite cargar los datos para graficar el indicador Elementos Inconsistentes.

+Graficar_UtilidadProceso(preguntas)	Permite cargar los datos para graficar el indicador Utilidad del Proceso.
+Graficar_InestabilidadRequisitos (requisitos, cantidad de cambios)	Permite cargar los datos para graficar el indicador Inestabilidad de los Requisitos.

Tabla 3 Clase Controladora Graficar_Indicadores

- **CAD_Conexión:** Esta clase permite el acceso a los datos
- **CE_Revisión:** Esta clase contiene los datos de una revisión.
- **CE_Inconsistencias:** Esta clase contiene los datos de una inconsistencia.
- **CE_Requisitos:** Esta clase contiene los datos de los requisitos.
- **CE_Encuestas:** Esta clase contiene los datos de las encuestas.

3.2 Diagramas de interacción

Un diagrama de interacción, representa la forma en cómo un Cliente (Actor) u Objetos (Clases) se comunican entre sí en petición a un evento. Representan el dinamismo que se modela en el diagrama de clases del diseño de manera estática, esto implica recorrer toda la secuencia de llamadas de donde se obtienen las responsabilidades. Hay dos tipos de diagramas de interacción: Diagramas de secuencia y Diagramas de colaboración. (4)

3.2.1 Diagramas de secuencia

Los diagramas de secuencia muestran las interacciones entre objetos ordenadas en secuencia temporal durante un escenario concreto. Proporciona una representación de la realización de casos de uso en términos de clases del diseño. En el diseño es más factible el empleo de los diagramas de secuencia ya que representan con más claridad el flujo de las acciones que debe realizar el sistema. (4)
A continuación se muestran los diagramas de secuencia para los tres escenarios del caso de uso Graficar Indicadores.

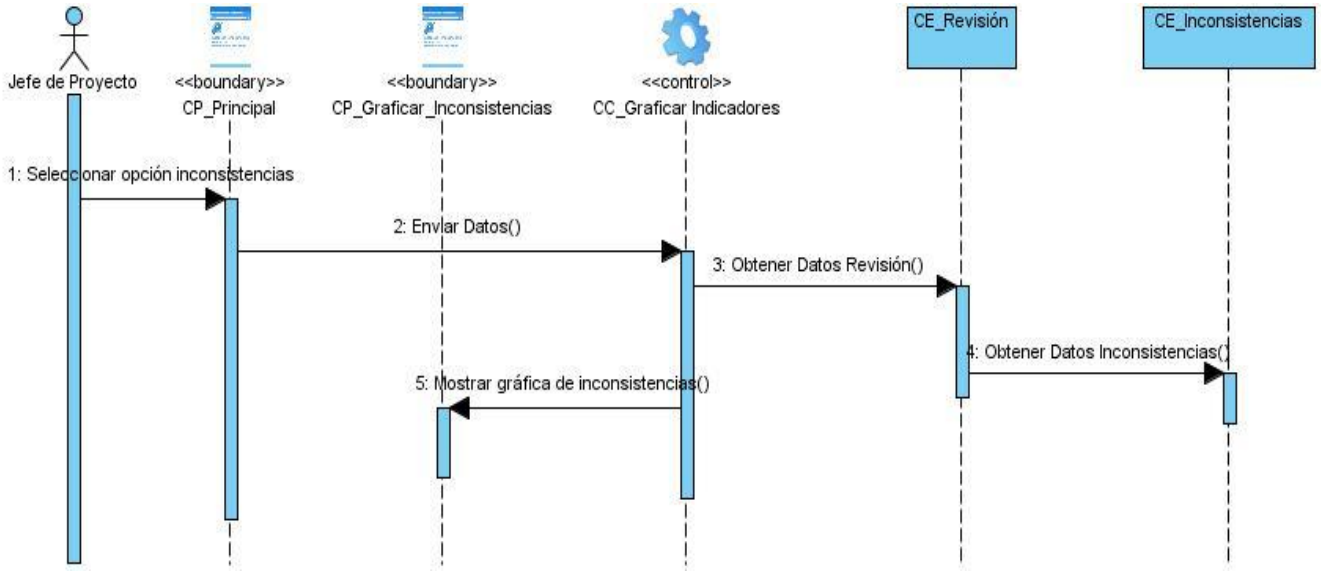


Figura 5 Diagrama de secuencia escenario Elementos Inconsistentes

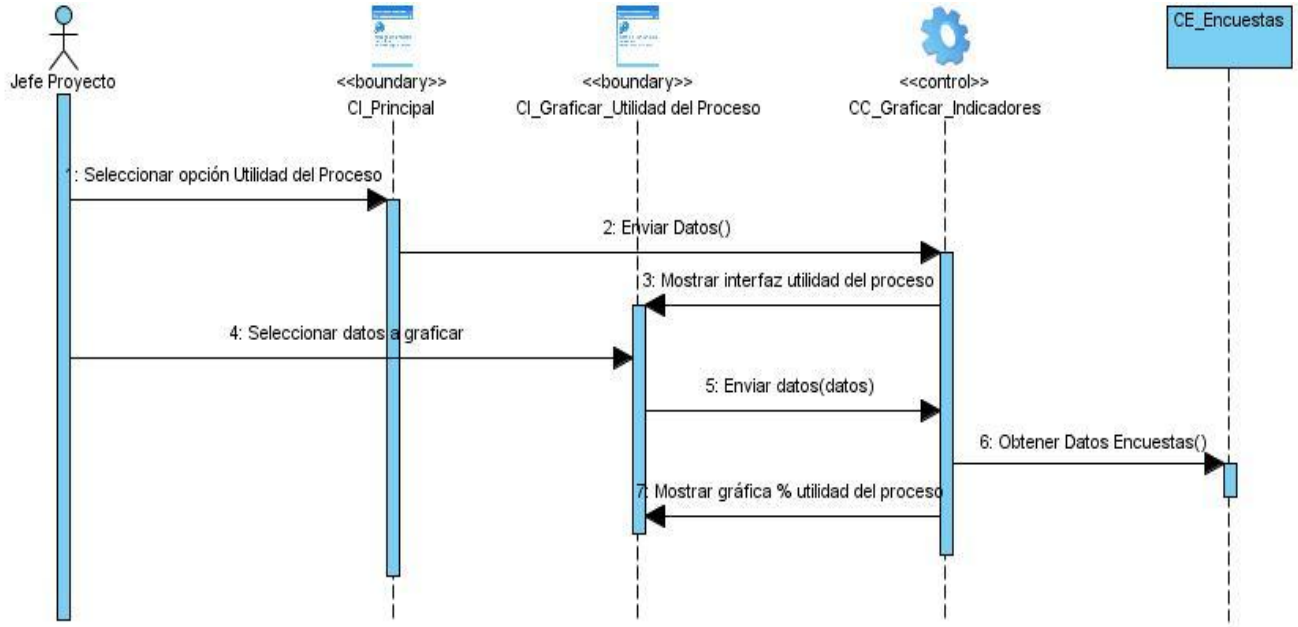


Figura 6 Diagrama de secuencia escenario Utilidad del Proceso

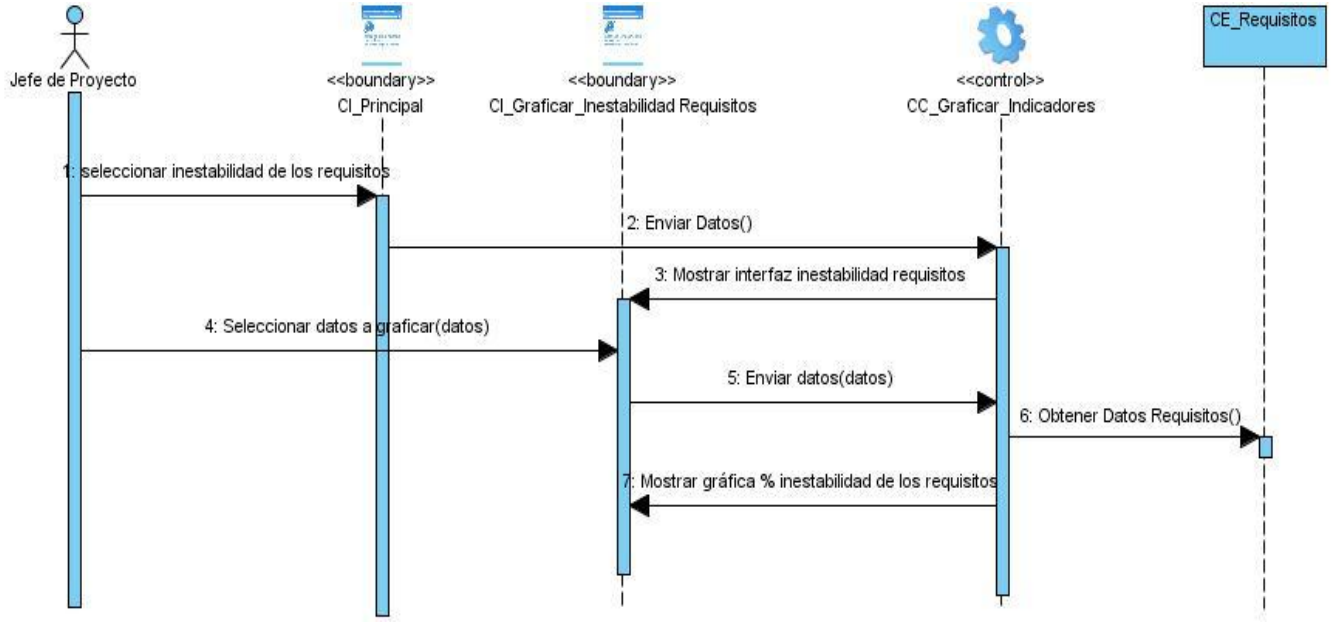


Figura 7 Diagrama de secuencia escenario Inestabilidad de Requisitos

3.3 Descripción de estilos arquitectónicos y patrones de diseño

3.3.1 Patrón arquitectónico

Los patrones expresan un paradigma fundamental para estructurar u organizar un sistema de software. Los patrones arquitectónicos inciden sobre aspectos fundamentales de la estructura de un sistema de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. (29)

Patrón Modelo – Vista – Controlador (MVC) (30)

Modelo Vista Controlador es un patrón que considera dividir una aplicación en tres módulos claramente identificables y con funcionalidades bien definidas: El Modelo, las Vistas y el Controlador.

El modelo

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar sin tomar en cuenta ni la forma en la que esta información va a ser mostrada, ni los mecanismos que hacen que estos datos estén dentro del modelo, es decir, sin tener relación con ninguna otra entidad dentro de la aplicación. Es independiente de cualquier representación de salida o comportamiento de entrada.

Las vistas

Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, y pueden existir varias vistas asociadas al mismo modelo.

El controlador

El controlador es un objeto que se encarga de dirigir el flujo de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador.

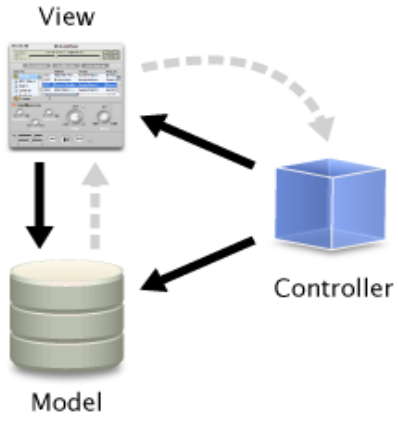


Figura 8 Patrón Arquitectónico Modelo-Vista-Controlador

A continuación se muestra cómo se evidencia este patrón arquitectónico dentro de la aplicación, se ejemplifica con la clase del diseño Graficar_Indicadores mostrada en epígrafes anteriores.

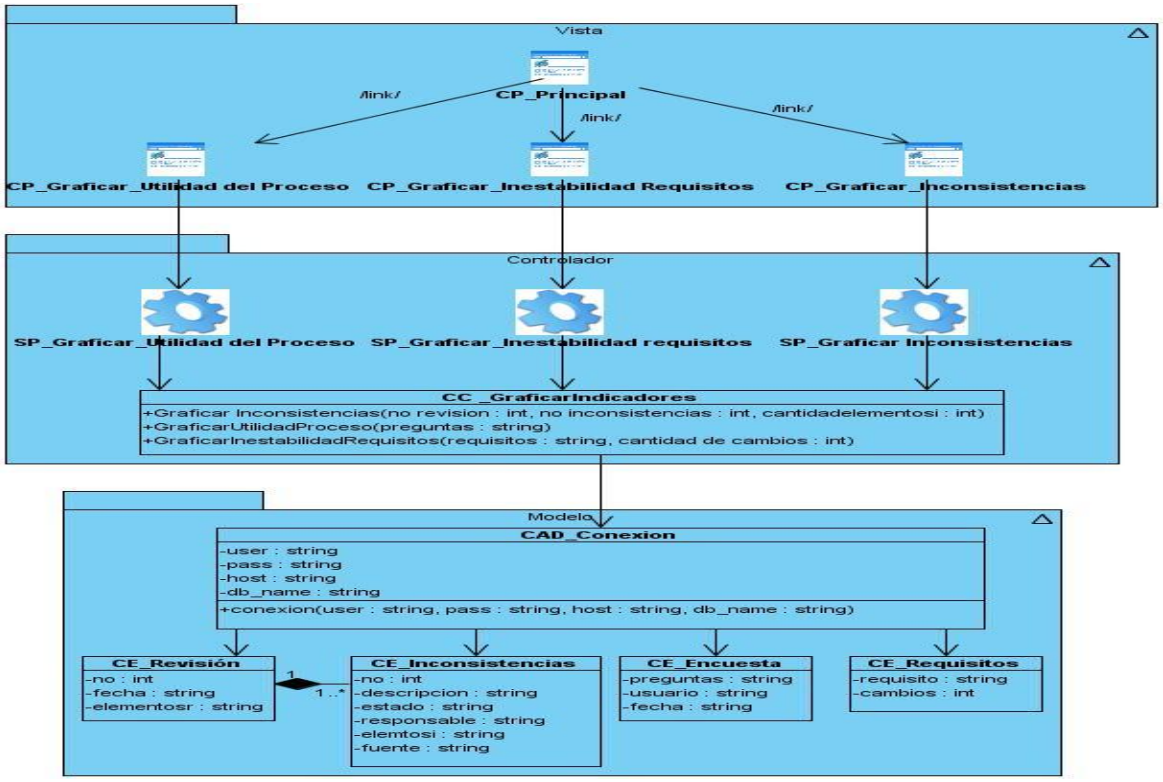


Figura 9 Clase del Diseño Graficar Indicadores (Modelo-Vista-Controlador)

3.3.2 Patrones de diseño

Un patrón describe un problema que ocurre varias veces y describe también el núcleo de la solución al problema, de forma que puede utilizarse en ilimitadas ocasiones sin tener que hacer dos veces lo mismo.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. En otras palabras, un patrón de diseño no es más que una solución estándar para un problema común de programación. Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios. (31)

Patrones Grasp

GRASP es un acrónimo de General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para sugerir la importancia de aprender estos principios para diseñar con éxito el software orientado a objetos. Los patrones GRASP son utilizados para describir los principios fundamentales del diseño y la asignación de responsabilidades. (31)

Experto en Información

Un Modelo de diseño podría definir cientos o miles de clases software, y una aplicación podría requerir que se realicen cientos o miles de responsabilidades. Durante el diseño de objetos, cuando se definen las interacciones entre los objetos, se toman decisiones sobre la asignación de responsabilidades a las clases. Haciendo un resumen de lo anteriormente planteado este patrón consiste en asignar la responsabilidad a la clase que tiene la información necesaria para realizarla. Esto se evidencia en muchas partes de la aplicación, ¿Cuál es la clase encargada de adicionar una nueva inconsistencia? En este caso la clase SP_Graficar_Inconsistencias.php es la que maneja este tipo de información, por lo que la clase CI_Inconsistencia.html envía los datos para adicionar la inconsistencia hacia ella. Evidenciándose de esta forma el patrón Experto en Información.

Creador

La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad y reutilización. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento. En la aplicación se pone de manifiesto este patrón, por ejemplo cuando la clase SP_Graficar_Inconsistencias.php crea una instancia de CC_Graficar_Indicadores.php para invocar al método de graficar inconsistencias.

$graficar=new Graficar_Indicadores (); $graficar->GraficarInconsistencias ().

Alta Cohesión

La cohesión es la medida de la fuerza que une a las responsabilidades de una clase. Una clase con baja cohesión hace muchas cosas no relacionadas, o hace demasiado trabajo. Tales clases no son convenientes ya que son difíciles de mantener, de reutilizar y de entender. Una clase con alta cohesión mejora la claridad y la facilidad de su uso, su mantenimiento se simplifica y es fácil de reutilizar.

Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que un elemento está conectado a otro, un elemento con bajo acoplamiento no depende demasiado de otros elementos. Una clase con alto acoplamiento confía en muchas otras clases, tales clases podrían no ser deseables ya que son muy complicadas de entender, son difíciles de reutilizar y los cambios realizados en las clases relacionadas fuerzan cambios locales. Este patrón se evidencia dentro de la aplicación en la capa modelo ya que las clases de acceso a los datos tienen bastante independencia de las clases de abstracción de datos. Hay poca dependencia entre esas clases lo que permite una mayor reutilización.

Después de vistos los patrones GRASP se verán otros patrones utilizados en la aplicación.

DAO

Patrón de Diseño que centraliza todo el acceso a datos en una capa independiente, aislándolo del resto de la aplicación. Su principal beneficio es que reduce la complejidad de los objetos de negocio al abstraerlos de la implementación real de la comunicación con la fuente de datos y de esta forma permitir una migración más fácil de fuente de datos. El uso de este patrón en el sistema se evidencia en la capa de Acceso a Datos (Clase Conexión.php), Ella se encarga de realizar la gestión de los datos entre las clases que contienen la lógica de negocio y las entidades.

Decorador

El uso de este patrón se pone de manifiesto en la relación que se establece entre el layout o plantilla global y las diferentes plantillas que forman parte de la vista. El archivo llamado Index.php que contiene el layout de la página, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. Este comportamiento es una implementación del patrón Decorador.

3.4 Modelo de datos

El modelo de datos es una colección de conceptos que sirven para describir la estructura de una base de datos, además de proporcionar los medios necesarios para conseguir la abstracción de los datos. Un modelo de datos es la estructura o representación física de las tablas de la base de datos**.** (7) A continuación el modelo de datos de la aplicación.

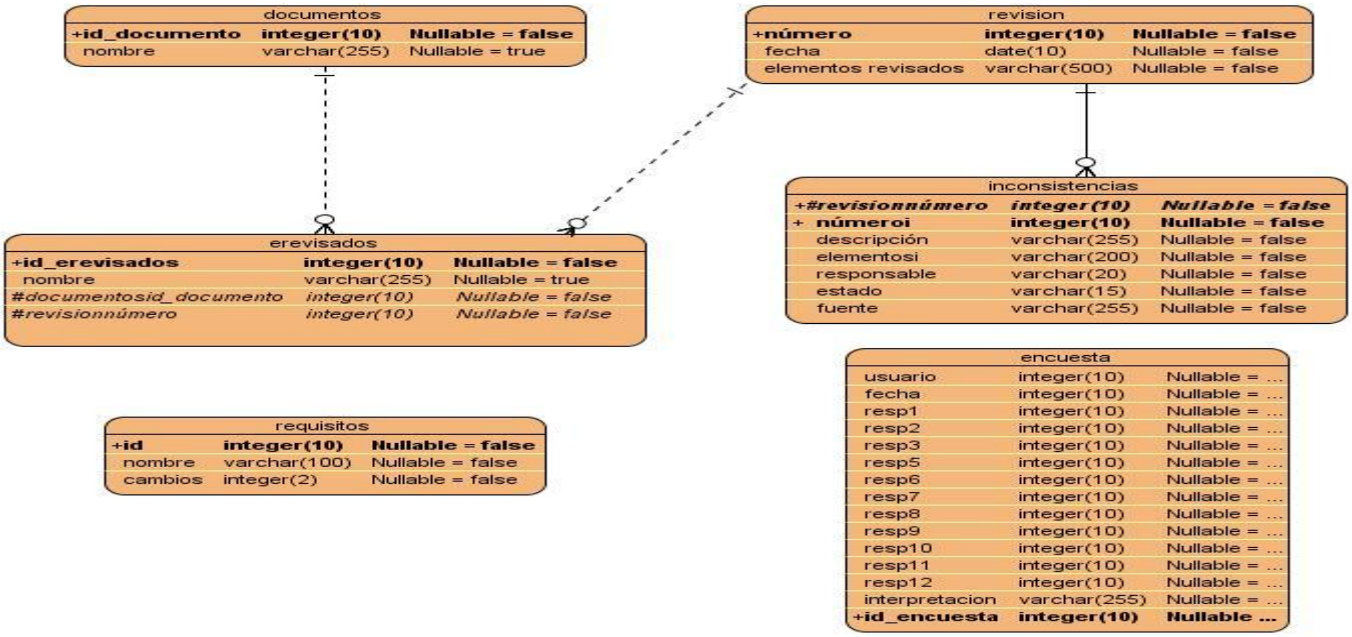


Figura 10 Modelo de datos

3.5 Vista de despliegue

La Vista de despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada con el despliegue del sistema propuesto. Esta vista representa el mapeo de componentes de software ejecutables con los nodos de procesamiento.

La Vista de despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre estos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. (4)

A continuación se presenta la Vista de despliegue de la aplicación.

Figura 11 Vista de despliegue

Se reflejan tres nodos principales, el nodo procesador PC Cliente, el Servidor de BD y el nodo procesador Servidor Web. La PC Cliente estará conectada mediante el Protocolo de Transferencia de Hipertexto HTTP al nodo procesador que representa al Servidor Web y este mediante el protocolo de control de transmisión tcp/ip estará conectado al servidor de Base de Datos.

Conclusiones del Capítulo

En el capítulo se definió como patrón arquitectónico Modelo-Vista-Controlador y se analizó cómo se evidencia este dentro de la aplicación. De la misma forma se explicaron los patrones de diseño a utilizar. Se realizó el diagrama de clases del diseño de uno de los Casos de Uso críticos y los diagramas de secuencia asociados a este. Además, se obtuvo el modelo de datos y la vista de despliegue.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

Introducción

Durante este capítulo se realiza el flujo de trabajo de implementación, se analizan los artefactos principales como el Modelo de implementación que incluye componentes, subsistemas de implementación y diagramas de componentes. Se comienza a implementar el sistema en términos de componentes, se muestran fragmentos de las clases e implementaciones más importantes para la herramienta y las pruebas realizadas para su validación.

4.1 Modelo de implementación

El Modelo de implementación representa la composición física de la implementación en términos de subsistemas de implementación y elementos de implementación. Describe cómo se organizan los componentes de acuerdo a los mecanismos de estructuración disponibles en el entorno de implementación y cómo dependen los componentes unos de otros. Dicho modelo se considera el artefacto más significativo del flujo de trabajo de implementación debido a la importancia que tiene para los desarrolladores. Este modelo está conformado por el diagrama de componentes. (7)

4.1.1 Diagrama de componentes

Dentro del Modelo de implementación se encuentran los diagramas de componentes. Un diagrama de componentes representa cómo un sistema es dividido en componentes y muestra las dependencias entre estos, además describe los elementos físicos y sus realizaciones en el entorno de implementación y en el lenguaje de programación utilizado.(4) A continuación se representa el diagrama de componentes utilizado en la presente investigación.

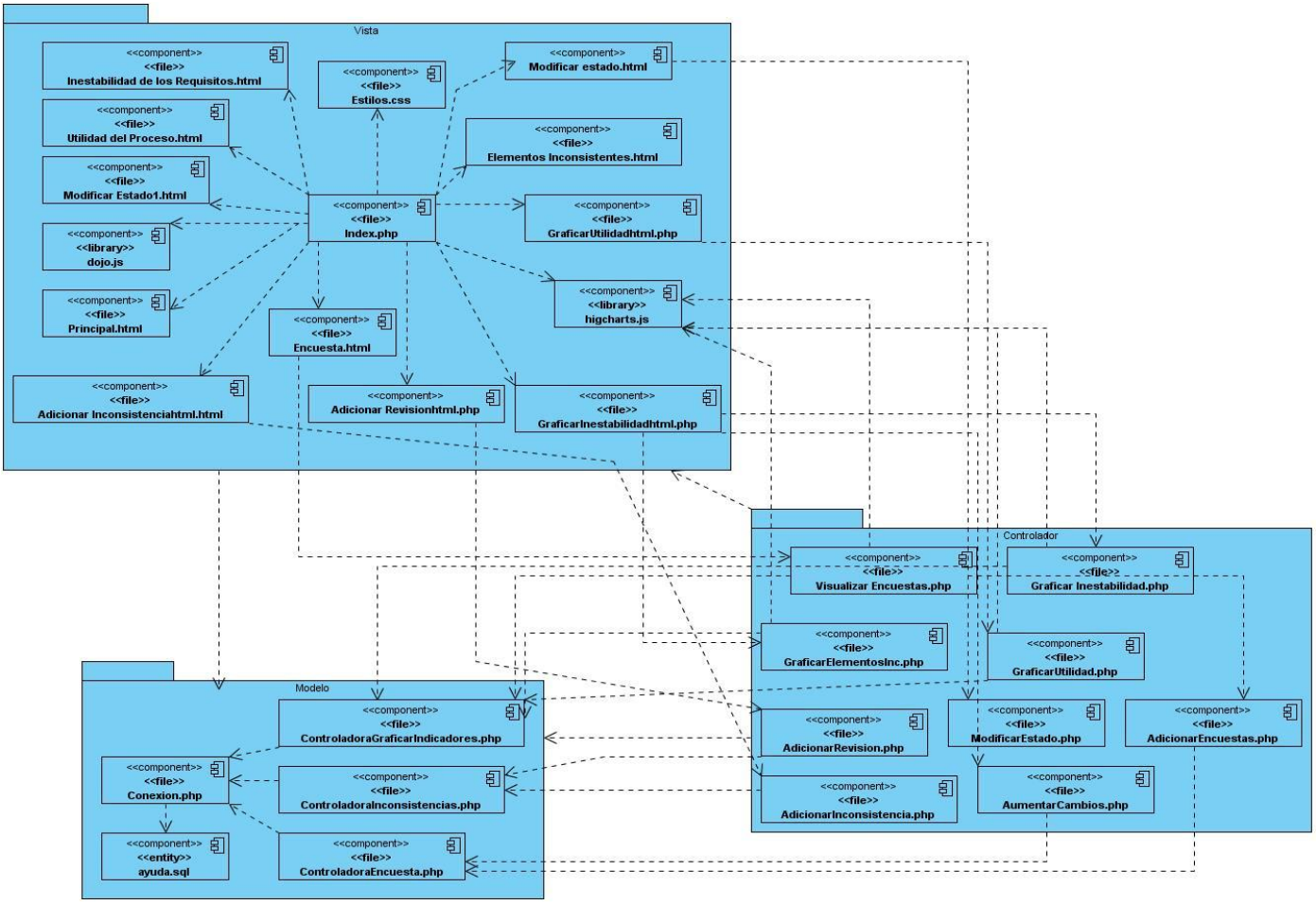


Figura 12 Diagrama de componentes

4.2 Código fuente

Para obtener una versión funcional de la aplicación se deben implementar los componentes que se han definido; como resultado se obtienen archivos que contienen el código fuente de la aplicación. Este no es más que un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. En el código fuente de un programa está descrito por completo su funcionamiento. (32)

4.2.1. Estándares de codificación

La combinación de técnicas de codificación sólidas y las buenas prácticas de programación con el objetivo de lograr un código robusto, es de vital importancia para la calidad del software. La aplicación continua de un estándar de codificación puede contribuir a obtener un sistema de software fácil de comprender y de mantener. (7)

Nomenclatura

- Los nombres de cada uno de los elementos del programa deben ser significativos; su nombre debe explicar en lo posible el uso del elemento.
- La mayoría de los elementos se deben nombrar usando sustantivos.
- La forma de construir los nombres será colocando primero el verbo o el sustantivo, y luego su complemento con la primera letra en mayúscula.

Estilo de codificación utilizado

- Todas las etiquetas php deben ser completas (<? php ?>).
- Los bloques de código siempre deben estar encerrados por llaves (excepto si constan de una línea).
- Tamaño = 3 (espacios) para:
  - Declaraciones dentro de las clases.
  - Enunciado dentro de métodos y funciones.
  - Enunciados dentro de bloques de comandos.

4.2.2. Ejemplo de código fuente

A continuación se muestra un fragmento de código para graficar el indicador Utilidad del Proceso y se ofrece una breve descripción de su implementación.

Ejemplo de código fuente Graficar Indicadores: Funcionalidad Utilidad del Proceso

```php
<?php

require_once('../Controladora/Controladora_Resultados.php');
$idencuesta=$_POST['encuesta'];
$control = new Controladora();
$result=$control->SeleccionarEncuestaID($idencuesta);


while($datos = pg_fetch_array($result))
{
	$categ1=round(($datos[3]+$datos[4]+$datos[5])*100/3,1);
	$categ2=round(($datos[6]+$datos[7]+$datos[8])*100/3,1);
	$categ3=round(($datos[9]+$datos[10]+$datos[11])*100/3,1);
}

	$nombres[] = "'Categoria1'";
	$nombres[] = "'Categoria2'";
	$nombres[] = "'Categoria3'";

	$cambios1[] = $categ1;
	$cambios1[] = $categ2;
	$cambios1[] = $categ3;

	$cambios2[]=100-$categ1;
	$cambios2[]=100-$categ2;
	$cambios2[]=100-$categ3;

?>
 <script type="text/javascript">

		var chart;
		$(document).ready(function() {
			chart = new Highcharts.Chart({
				chart: {
					renderTo: 'grafico',
					defaultSeriesType: 'bar'
				},
				title: {
					text: 'Utilidad del Proceso'
				},
		subtitle: {
			text: 'CALISOFT'
		},
				xAxis: {
					categories: [<?php echo implode(', ', $nombres) ?>]
				},
				yAxis: {
					min: 0,
					title: {
						text: '% de las respuestas'
					}
				},
				legend: {
					backgroundColor: '#FFFFFF',
					reversed: true
				},
				tooltip: {
					formatter: function() {
						return ''+
							this.series.name +': '+ this.y +'';
					}
				},
				plotOptions: {
					series: {
						stacking: 'normal'
					}
				},
				series: [{
					name: 'No',
					data: [<?php echo implode(', ', $cambios2) ?>]
				}, {
					name: 'Si',
					data: [<?php echo implode(', ', $cambios1) ?>]
				}]
			});


		});

	</script>
```

Tabla 4 Ejemplo de código fuente Graficar Indicadores: Funcionalidad Utilidad del Proceso

Esta funcionalidad permite, una vez seleccionada la encuesta a la que se quiere acceder a sus resultados, calcular las categorías en que se dividen las respuestas a esta y mostrar los resultados de forma gráfica.

4.3 Validación

Validar datos es una tarea común para cualquier tipo de aplicación, desde la capa de presentación hasta la capa de Acceso a Datos. A menudo, la misma lógica de validación es aplicada varias veces por lo que se emplea mucho tiempo en la implementación, para evitar esto en las clases interfaces se validan los tipos de datos y para no introducir datos erróneos en la aplicación la validación se realiza en las clases controladoras.

En la aplicación las validaciones a nivel de cliente se realizan utilizando el lenguaje JavaScript, el cual se incluye en los formularios que necesiten validar los datos antes de enviarlos al servidor, para lograr esto se crearon expresiones regulares que verifiquen que los valores de entrada sean los correctos. En el caso de las validaciones realizadas a nivel de servidor estas fueron realizadas utilizando el lenguaje PHP.

Ejemplo de código validación Graficar Indicadores: Inestabilidad de los Requisitos

Validaciones.js

```
function TotalMarcados()
{
        var checks=document.getElementsByTagName('input');
        var totalchecks=checks.length;
        var totalmarcados=0;
        for(var pos=0;pos<totalChecks;pos++)
        {
           if(checks[pos].type=="checkbox")
           {
              if(checks[pos].checked)
                 totalmarcados++;
           }
        }
        if(totalmarcados==0)
           {
              alert("Marque al menos uno");
              return false
           }
         else
              return true
 }
```

Tabla 5 Ejemplo de código validación Graficar Indicadores: Inestabilidad de los Requisitos

4.4 Pruebas

Un instrumento adecuado para determinar la calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que este cumple con los requisitos.

Una forma de verificar la consistencia de los datos de la aplicación es mediante la introducción de datos erróneos o dejar en blanco campos que son de carácter obligatorio. Deben realizarse pruebas a todos los artefactos generados durante la construcción de un producto, lo que incluye especificaciones de requisitos, casos de uso, diagramas de diversos tipos y, por supuesto, el código fuente y el resto de productos que forman parte de la aplicación.

Las pruebas realizadas a la aplicación son de caja negra al software, para realizar estas se diseñaron casos de pruebas a través de la descripción de los casos de usos. El siguiente es un ejemplo del Caso de Prueba realizado al Caso de Uso Graficar Indicadores.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1-Inestabilidad de los requisitos	EC 1.1: Obtener resultados correctamente.	Se deben mostrar los resultados de forma correcta.	1. El jefe de proyecto selecciona la opción Obtener Resultados dentro del menú Inestabilidad de los Requisitos. 2. El jefe de proyecto selecciona los datos correctamente. Habilitándose la opción 'Resultado' una vez que el jefe de proyecto seleccione al menos un requisito.
SC2-Utilidad del Proceso	EC 2.1 Obtener resultados correctamente.	Se deben mostrar los resultados de forma correcta.	1. El jefe de proyecto selecciona la opción Obtener Resultados dentro del menú Utilidad del Proceso. 2. El jefe de proyecto selecciona los datos correctamente.

SC3-Elementos Inconsistentes	EC 3.1 Obtener resultados correctamente.	Se deben mostrar los resultados de forma correcta.	1. El jefe de proyecto selecciona la opción Obtener Resultados dentro del menú Elementos Inconsistentes.

Tabla 6 Secciones de prueba Caso de Uso Graficar Indicadores

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Requisitos	Campo de selección	No	Variable para especificar los requisitos a lo que se quieren acceder.
2	Encuesta	Campo de selección	No	Variable para especificar las respuestas de la encuesta.
3	Revisión	Campo de selección	No	Variable para especificar las revisiones realizadas.

Tabla 7 Descripción de variables Caso de Uso Graficar Indicadores

Escenario	V1	Respuesta del sistema	Resultado de la Prueba	Flujo Central
SC1- Inestabilidad de los requisitos	V (2,4,5,7)	El sistema muestra los resultados de forma satisfactoria.	Satisfactorio	1- El jefe de proyecto selecciona la opción Obtener Resultados dentro del menú Inestabilidad de los Requisitos.
	V (1,2,5,3,6)	El sistema muestra los resultados de forma satisfactoria.	Satisfactorio	2- El jefe de proyecto selecciona los datos correctamente.

Tabla 8 Matriz de datos SC1: Inestabilidad de los requisitos

Escenario	V2	Respuesta del sistema	Resultado de la Prueba	Flujo Central
SC2-Utilidad del Proceso.	V (Flavio,29-05-2011)	El sistema muestra los resultados de forma satisfactoria.	Satisfactorio	1-El jefe de proyecto selecciona la opción Obtener Resultados dentro del menú Utilidad del Proceso.
	V (Blanca,22-05-2011)	El sistema muestra los resultados de forma satisfactoria.	Satisfactorio	2-El jefe de proyecto selecciona los datos correctamente.

Tabla 9 Matriz de datos SC2: Utilidad del Proceso

Escenario	V3	Respuesta del sistema	Resultado de la Prueba	Flujo Central
SC3-Elementos Inconsistentes	V(1,2,3)	El sistema muestra los resultados de forma satisfactoria.	Satisfactorio	1-El jefe de proyecto selecciona la opción Obtener Resultados dentro del menú Elementos Inconsistentes.

Tabla 10 Matriz de datos SC3: Elementos Inconsistentes

4.5 Interfaces principales de la aplicación

A continuación se muestran algunas interfaces de la aplicación:

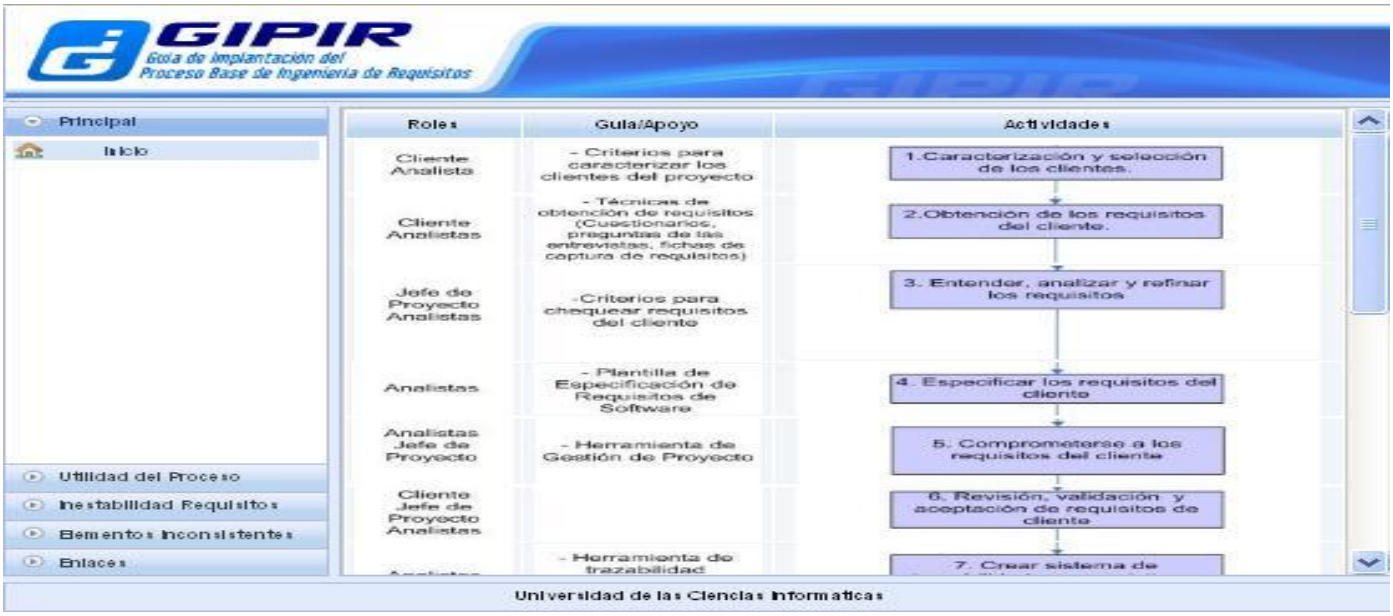


Figura 13 Interfaz Principal

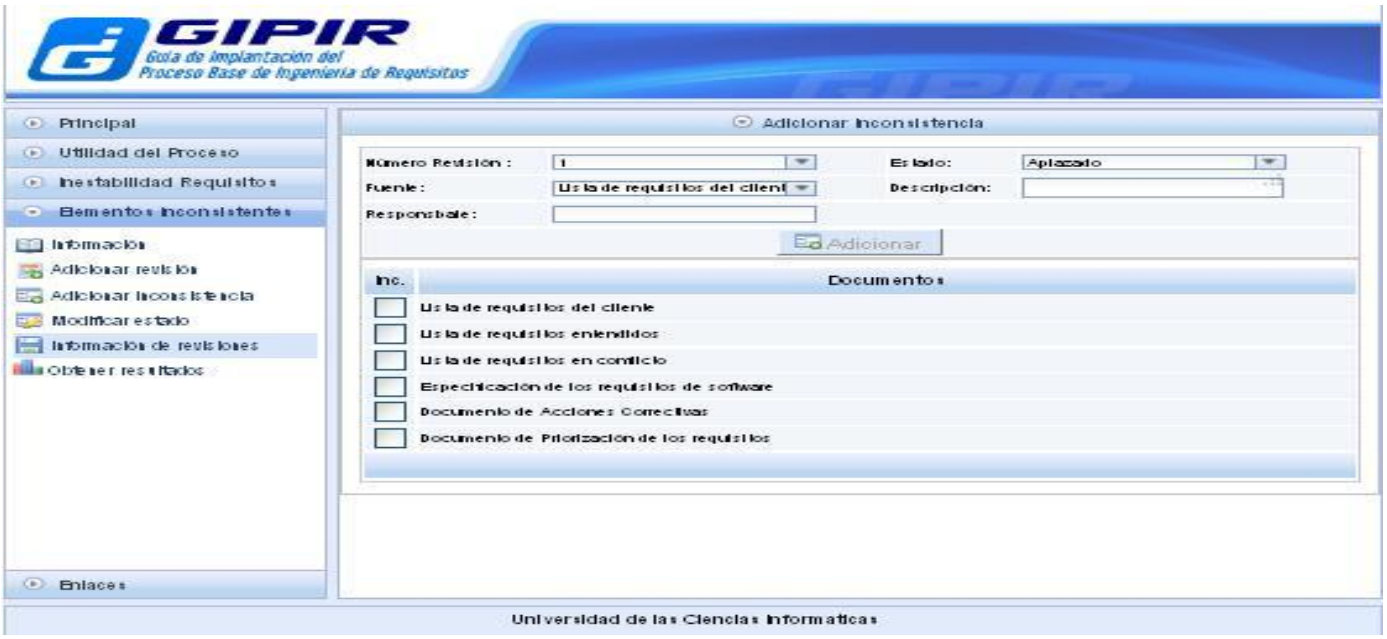


Figura 14 Interfaz Adicionar Inconsistencias

Figura 15 Interfaz Resultados Utilidad del Proceso

Conclusiones del Capítulo

En este capítulo se elaboró el diagrama de componentes, se confeccionó completamente la propuesta que trae la aplicación y se brindó una breve descripción de algunos de los fragmentos de código utilizados en la herramienta para lograr un mejor entendimiento de estos. Además se citó el ejemplo de una validación realizada que permite detectar y solucionar los errores identificados así como la realización de pruebas de caja negra al software.

CONCLUSIONES

Una vez culminado el trabajo es posible afirmar se cumplieron los objetivos trazados y puede concluirse que:

- La aplicación fue implementada utilizando herramientas, lenguajes y tecnologías en su mayoría distribuidas bajo licencias de software libre en correspondencia con las políticas de la Universidad y del país.
- Se definieron las funcionalidades que debía tener la aplicación y se logró el desarrollo de una herramienta informática.
- La implementación de esta propuesta posibilitará guiar a las empresas durante la implantación del proceso base de Ingeniería de Requisitos del Modelo Cubano para el Desarrollo de Aplicaciones Informáticas.
- Las pruebas realizadas para validar las funcionalidades que fueron implementadas, demuestran que la aplicación cumple satisfactoriamente con los requisitos que garantizan su correcto funcionamiento.

RECOMENDACIONES

Después de haber logrado los objetivos que se trazaron al principio de este trabajo y como el producto informático se encuentra en su primera versión se plantean las siguientes recomendaciones:

- Implementar una nueva versión basada en ontologías, debido a que el trabajo con estas permite una formalización en la representación del conocimiento y al mismo tiempo una mejor organización y reutilización de este.
- Agregarle la implementación de nuevos indicadores, lo que permitirá reflejar de forma más simple el estado del proceso de requisitos y de esta forma hacer más fácil el desarrollo del proyecto.
- Integrar la aplicación web del presente trabajo con el sistema de administración de requisitos.

REFERENCIAS BIBLIOGRÁFICAS

1 **Christel, M. G., Kang,** Issues in Requirements ElicitationPittsburgh 1992

2 **Durán Toro, A.** Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de InformaciónSevilla 2006

3 **P.Hsia, A. Davis, D.Kung** Requirements Engineering. IEEE Software 1993

4 **Pressman, R.** Ingeniería de Software. Un enfoque práctico 2000

5 **IEEE. web** IEEE Recommended Practice for Software Requirements Specificationshttp://arantxa.ii.uam.es/~sacuna/is1/normas/IEEE_Std_830_1998.pdf

6 **DOD. Military Standard**Software Development and Documentation.http://www.library.itsi.disa.mil/mil_std/498_win3.exe

7 **Jacobson, Ivar, Booch, Grady y Rumbauch, James.** El Proceso Unificado de Desarrollo del Softwaremadrid2000

8 **J.A.Goguen.** Requirement Engineering as the Reconciliation of Social and Technical issues http://www.cse.ucsd.edu

9 **Herrera J., Lizka Johany** Ingeniería de Requerimientos, Ingeniería de Software http://www.monografias.com/trabajos6/resof/resof.shtml

10 **Escalona, M.E.** Ingeniería de Requisitos en Aplicaciones para la Web Un estudio comparativo. http://www.lsi.us.es/docs/informes/LSI-2002-4.pdf

11 **Lieter, P** Proyecto docente e investigador2003

12 **Departamento nacional de planeación** Guía metodológica para la formulación de indicadoresBogota2009

13 **Indicadores** Indicadores http://es.kioskea.net/contents/qualite/indicateurs.php3

14 **indicadores** Metodología para la formulación de indicadoreshttp://www.desarrolloeconomico.gov.co/documentos/produccion/MetodologiaIndicadores.pdf

15 **Castrillo, Ricardo Gonzalez** Taller sobre indicadores de calidad2008

16 **H. Canós, José, Letelier, Patricio y Carmen Penadés** Métodologías Ágiles en el Desarrollo de Software.Valencia2002

17 **Serrano, Jorge.** Explicando SCRUM2003

18 **Lucas** modelado-sistemas-UML http://lucas.hispalinux.es/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf

19 **IBM.** IBM http://www.ibm.com.

20 **Paradigm, Visual**Paradigm, Visual http://www.visual-paradigm.com/

21 **Corporation, MySQL.**Corporation, MySQL.http://MySQL.org

22 **postgre** PostgreSql http://www.postgresql.org/docs/8.3/static/ddl-schemas.html

23 **schermbeek, Mariangela Pocaterra Van** APACHE http://ddd.uab.cat/pub/trerecpro/2009/hdl_2072_41829/Treball+de+recerca.pdf

24 **Fundation, The netbeans** Netbeanshttp://www.netbeans.org

25 **ZendStudio** Zend.comhttp://www.zend.com/en/products/studio/

26 **PHP Hinostroza, Raul Rodas.**Caracteristicas de PHP http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP

27 **javaHispano**Tu lenguaje, tu comunidad http://www.javahispano.org/contenidos/es/exprimiendo_javawebstart

28 **DojoToolkit** http://www.dojotoolkit.org

29 **Perrone Paul J and Krishna** Handbook, J2EE Developer's2008

30 **Bergin, Joseph** Building Graphical User Interfaces with the MVC pattern http://csis.pace.edu/bergin/mvc/mvcgui.html

31 **Larman, Craig** UML y Patrones Indiana2004

32 **Código** El código fuente http://www.proyectoautodidacta.com/comics/que-es-el-codigo-fuente/

BIBLIOGRAFÍA

1.**Alarcon, José Manuel.** Manual-de-PostgreSQL-Server . [Online] 2 23, 2009. [Cited: 2 15, 2011.] http://es.scribd.com/doc/32185176/Manual-de-PostgreSQL-Server.

2.**Bergin, Joseph.** Building Graphical User Interfaces with the MVC pattern. [Online] 2 4, 2009. [Cited: 3 12, 2011.] http://csis.pace.edu/ bergin/mvc/mvcgui.html.

3.**Burbeck, Steve.** Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). [Online] 3 21, 2011. http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

4.**Castrillo, Ricardo Gonzalez.** Taller sobre indicadores de calidad. 2008.

5.**Christel, M. G., Kang,.** Issues in Requirements Elicitation. Pittsburgh : s.n., 1992.

6.**Departamento nacional de planeación de.** Guía metodológica para la formulación de indicadores. Bogota : s.n., 2009..

7.**DOD. Military Standard.** Software Development and Documentation. [Online] 10 20, 2005. [Cited: 2 20, 2011.] http://www.library.itsi.disa.mil/mil_std/498_win3.exe.

8.**Dojo.** DojoToolkit. [Online] 12 5, 2006. [Cited: 1 23, 2001.] http://www.dojotoolkit.org.

9.**Durán Toro, A.** Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información. Sevilla : s.n., 2006.

10.**Escalona, M.E.** Ingeniería de Requisitos en Aplicaciones para la Web Un estudio comparativo. [Online] 3 20, 2002. [Cited: 1 12, 2011.] http://www.lsi.us.es/docs/informes/LSI-2002-4.pdf.

11.**Fundation, The netbeans.** Netbeans. [Online] 5 12, 2004. http://www.netbeans.org.

12.**Gamma, Erich, y otros.** Addison-Wesley Design Patterns: Elements . 1995.

13.**H. Canós, José, Letelier, Patricio y Carmen Penadés.** Métodologías Ágiles en el Desarrollo de Software. Valencia : s.n., 2002.

14.**Herrera J., Lizka Johany.** Ingeniería de Requerimientos, Ingeniería de Software. [Online] 4 12, 2003. [Cited: 2 23, 2011.] http://www.monografias.com/trabajos6/resof/resof.shtml

15.**IBM. .** IBM. [Online] 9 25, 2004. [Cited: 1 26, 2011.] http://www.ibm.com. **Lucas.** modelado-sistemas-UML. [Online] 10 2, 2007. [Cited: 1 20, 2001.] http://lucas.hispalinux.es/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.

16.**IEEE. web.** IEEE Recommended Practice for Software Requirements Specifications. [Online] 7 25,

2001. [Cited: 2 1, 2011.] http://arantxa.ii.uam.es/~sacuna/is1/normas/IEEE_Std_830_1998.pdf.

17.**Indicadores.** Indicadores. [Online] http://es.kioskea.net/contents/qualite/indicateurs.php3.

18.**Indicadores.** Metodología para la formulación de indicadores. [Online] 1 2, 2007. [Cited: 1 21, 2011.] http://www.desarrolloeconomico.gov.co/documentos/produccion/MetodologiaIndicadores.pdf.

19.**ISO.9000. 2000.** Ayuda Extendida del Rational Unified Process. 2002.

20.**Jacobson, Ivar, Booch, Grady y Rumbauch, James.** El Proceso Unificado de Desarrollo del Software. madrid : s.n., 2000.

21.**J.A.Goguen.** Requirement Engineering as the Reconciliation of Social and Technical issues. [Online] 8 25, 2002. [Cited: 1 2011, 23.] http://www.cse.ucsd.edu.

22.**JavaScript.** Conocimientos básicos de Javascript. [Online] 12 23, 2008. [Citado: 4 23, 2011.] http://perso.wanadoo.es/javascript_12/.

23. **Larman, Craig. UML y Patrones.** Indiana: s.n., 2004.

24. **Lieter, P.** Proyecto docente e investigador. 2003.

25. **Mora, Roberto Canales.** Tutorial: Patrones GRASP. Adictos al trabajo. [Online] 12 5, 2003. [Cited: 2 23, 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

26. **MySQL.** Corporation, MySQL. [Online] 10 27, 2005. [Cited: 1 23, 2011.] http://MySQL.org.

27. **MVC.** Building Graphical User Interfaces with the MVC pattern. [Online] 15 3, 2008. [Cited: 3 25, 2011.] http://csis.pace.edu/ bergin/mvc/mvcgui.html.

28.**Paradigm, Visual.** Paradigm, Visual. [Online] 10 27, 2005. [Cited: 1 21, 2011.] http://www.visual-paradigm.com.

29.**Perrone Paul J and Krishna.** Handbook, J2EE Developer's. 2008

30.**PHP Hinostroza, Raul Rodas.** Caracteristicas de PHP. [Online] 9 21, 2007. http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

31. **P.Hsia, A. Davis, D.Kung.** Requirements Engineering. IEEE Software. 1993.

32.**Postgre.** PostgreSql. [Online] 10 26, 2010. [Cited: 1 14, 2011.] http://www.postgresql.org/docs/8.3/static/ddl-schemas.html.

33. **Pressman, R.** Ingeniería de Software. Un enfoque práctico. 2000.

34. **Reynoso, C B.** *Introducción a la Arquitectura de Software.* Universidad de Buenos Aires : s.n., 2004.

35.**Saha, Amit Kumar.** MySQL. [Online] 2 23, 2007. [Cited: 4 25, 2011.] http://www.gedlc.ulpgc.es/docencia/abd/Recursos/MySQL-Intro-features-benefits-SPANISH.pdf.

36.**Schermbeek, Mariangela Pocaterra Van.** APACHE. [Cited: 1 23, 2011.] http://ddd.uab.cat/pub/trerecpro/2009/hdl_2072_41829/Treball+de+recerca.pdf.

37.**Serrano, Jorge.** Explicando SCRUM. 2003.

38.**Software, Departamento de Ingeniería de.** Ingeniería de Software II. [Online] 2 25, 2011. http://eva.uci.cu.

39. **Vázquez, Roberto Hugo.** Taller de Calidad de Software. [Online] 10 12, 2004. [Cited: 3 12, 2011.] http://gridtics.frm.utn.edu.ar/docs/introduccion%20a%20la%20calidad%20de%20software%20Vazquez.pdf

40. **Vila, Fermi.** JavaScript. [Online] 12 3, 2007. [Cited: 3 25, 2011.] www.softdowload.com.ar.

41. **ZendStudio.** Zend.com. [Online] 23 6, 2006. http://www.zend.com/en/products/studio/.

.

GLOSARIO DE TÉRMINOS

1- **Apache:** Es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

2- **APIs:** Una interfaz de programación de aplicaciones o API (del inglés application programming interface).

3- **Código abierto u Open source:** Es el término con el que se conoce al software distribuido y desarrollado libremente.

4- **Entidades:** Objetos concretos o abstractos que presentan interés para el sistema.

5- **GUI:** La interfaz gráfica de usuario, conocida también como GUI es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

6- **Hardware:** Componentes físicos que constituyen las Computadoras y demás dispositivos periféricos.

7- **Multihilo:** Es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente. Un hilo es básicamente una tarea que puede ser ejecutada en paralelo con otra tarea.

8- **PHP:** Es un lenguaje de programación usado generalmente para la creación de contenido para sitios web.

9- **Plugins:** Es una aplicación informática que añade funcionalidades específicas a un programa principal. Su nombre significa enchufable y su presencia es muy habitual en los navegadores web, en reproductores de música y en sistemas de gestión de contenidos. Los plugins no son parches ni actualizaciones, sino propiedades añadidas a los programas originales, aparecidas por primera vez a mediados de los años 70.