INSTITUTO SUPERIOR MINERO METALÚRGICO

"Dr. Antonio Núñez Jiménez".

Facultad de Metalurgia-Electromecánica.

Departamento de Informática

"SISTEMA DE GESTIÓN DE LA CALIDAD PARA LA EVALUACIÓN DE LAS APLICACIONES DE ESCRITORIO. ''

Trabajo de diploma para optar por el título de Ingeniero informático

Autor: Ramón Alejandro Velázquez Díaz.

Tutor: Ing. Eloy Rafael Jiménez Iglesias.

Moa, Cuba

Junio, 2012

Año 53 de la Revolución.

DECLARACIÓN DE AUTORÍA.

Declaro que soy el único autor de este trabajo y autorizo al Instituto Superior Minero Metalúrgico de Moa “Dr. Antonio Núñez Jiménez” para que hagan el uso que estimen pertinente del mismo.

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

Ramón Alejandro Velázquez Díaz

_____

Firma del autor

Ing. Eloy R. Jiménez Iglesias.

_____

Firma del tutor

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título:

Autor:

El tutor del presente Trabajo de Diploma considera que durante su ejecución elestudiante mostró las cualidades que a continuación se detallan.

<Aquí el tutor debe expresar cualitativamente su opinión y medir (usando laescala: muy alta, alta, adecuada) entre otras las cualidades siguientes:

• Independencia

• Originalidad

• Creatividad

• Laboriosidad

• Responsabilidad>

<Además, debe evaluar la calidad científico-técnica del trabajo realizado(resultados y documento) y expresar su opinión sobre el valor de los resultados obtenidos (aplicación y beneficios) >

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de <nota 2-Desaprobado, 3-Aprobado, 4-Bien, 5-Excelente>. <Además, si considera que los resultados poseen valor para ser publicados, debe expresarlo también>

_____

Nombre completo del primer tutor

<Grado científico, Categoría docente

y/o investigativa>

(Si procede)

_____

Nombre completo del segundo tutor

<Grado científico, Categoría docente y/o investigativa>

Fecha:_____

Pensamiento:

"Sólo los necios se encuentran satisfechos y confiados con la calidad de su trabajo".

Mercedes Milá.

Dedicatoria:

Dedico trabajo a mis padres Guillermo Rodríguez Castañeda y Teresa Díaz Jardines porque gracias a ellos existo y soy un buen ser humano.

A mis abuelos, que pese a la gran distancia siempre pensé en ellos durante la realización del trabajo.

A toda mi familia, que me brindó el apoyo que he necesitado para dar un feliz término a este trabajo.

Agradecimientos:

En primer lugar agradecer a mis padres por siempre estar junto a mí pese a los problemas y dificultades. Por haber tratado de educarme de la mejor manera posible. Gracias por su ayuda y cariño.

A mi tío Julio Esteban Díaz por su gran ayuda y por haber sido siempre un ejemplo a seguir profesionalmente.

A mi familia por todo su afecto y ayuda que me brindaron.

A mi tutor gracias por la ayuda y orientación que me diste.

A cada compañero de aula que perdí en el camino, me hubiera gustado que todos estuvieran conmigo en este momento tan especial.

A las personas que desee que estuvieran conmigo para compartir esta alegría, que no es la misma sin ellas y que en un momento determinado de mi vida fueron fuentes de inspiración para mi tesis.

Agradezco a esta Revolución que me ha dado la oportunidad de formarme como profesional.

Y a todos aquellos que de una forma u otra me tendieron desinteresadamente la mano.

A todos, gracias.

Resumen:

En la actualidad la industria productora de software en nuestro país tiene como principal objetivo desarrollar productos y servicios Informáticos de alta calidad. Para poder obtener estos resultados y lograr un posicionamiento y un reconocimiento en el mercado, es necesaria la implantación de Modelos o Estándares de Calidad que garanticen la comprobación objetiva de la evaluación de la calidad de los productos de software, desarrollados en cada una de las entidades.

En el siguiente proyecto se pretende crear un sistema automatizado que permita evaluar las aplicaciones de escritorio y los productos no conformes de las diferentes áreas del ISMM. Con el desarrollo del mismo se desea obtener mayor calidad rapidez y organización a la hora de corregir los problemas encontrados en la entidad. Esta Aplicación puede llevarse a cabo por los especialistas en la temática o cualquier otra persona interesada en esta actividad.

Palabras Claves: Calidad, Evaluación.

Summary

At the present time the producing industry of software in our country must like main objective develop to products and Computer science services of high quality.

In order to be able to obtain these results and of obtaining a positioning and a recognition in the market, the implantation of Models or Standards of Quality that guarantees the objective verification of the evaluation of the quality of software products, developed in each one of the organizations is necessary.

In the following project it is tried to create an automated system that allows evaluating the desktop applications and the products you do not conform of the different areas from the ISMM. With the development of the same one it is desired to obtain greater quality rapidity and organization at the time of correcting the problems found in the organization. This Application can be carried out by the specialists in thematic or the any other person interested in this activity.

Key words: Quality, Evaluation.

Índice de contenido

INTRODUCCIÓN:

Una especificación y evaluación integral y detallada de la calidad de los productos de software es un factor clave para asegurar que la calidad sea la adecuada. Esto se puede lograr definiendo de manera apropiada las características de calidad, teniendo en cuenta el propósito del uso del producto de software en la institución.

Es importante especificar y evaluar cada característica relevante de la calidad de los productos de software, cuando esto sea posible, utilizando mediciones validadas o de amplia aceptación, que hagan técnicamente transparente esta actividad.

La calidad es un activo estratégico clave, del que dependen la mayor parte de las organizaciones, no sólo para mantener su posición en el mercado sino incluso para asegurar su supervivencia. El desarrollo de un producto que satisfaga, en la mayor medida posible, los requerimientos del cliente, es la medida de calidad buscada en la producción.

A nivel mundial en la industria del software hay tendencia al crecimiento del volumen y complejidad de los productos, los proyectos están excesivamente tardes, se exige mayor calidad y productividad en menos tiempo y hay suficiente personal calificado.

Esto trae consigo que sea un reto para la Industria del Software desarrollar las estrategias que le permitan alcanzar un nivel de calidad realmente alto, por lo que se hace necesario todo un estudio para la elección e implantación del Modelo o Estándar de Calidad indicado. Aunque la empresa productora de software de nuestro país es muy joven se esfuerza para desarrollar software de calidad que puedan ser insertados en el mercado mundial. Para lograr estos objetivos es necesario garantizar la calidad de un producto de software, determinado por el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (software) satisfará los requisitos dados de calidad.

Actualmente en el ISMMM el proceso de evaluación de los software se realiza de forma manual, lo que hace el proceso más lento, no solo a la hora de su redacción, sino también en el momento de darles seguimiento a las acciones

tomadas, además esto aumenta los gastos de papel y otros materiales de oficina e incrementa la morosidad del seguimiento de las mismas, puesto que no permite que la actividad sea interactiva. También se torna un poco engorroso en el momento de generar diversos reportes correspondientes a las no conformidades.

Partiendo de esta situación problemática se decidió llevar a cabo la presente investigación definiendo la siguiente pregunta:

¿Cómo agilizar y facilitar el proceso de evaluación para la calidad de las aplicaciones de escritorios en el ISMMM teniendo en cuenta la Funcionalidad, Usabilidad y Confiabilidad?

Como **objeto de estudio** se tiene: El Sistema de Gestión de la calidad en el proceso de evaluación de las aplicaciones de escritorios.

De ahí se desprende nuestro **campo de acción,** o sea la automatización de los procesos de evaluación de la calidad de las aplicaciones de escritorio en el ISMMM.

Como **Objetivo General** de la investigación es desarrollar un Sistema que permita agilizar y facilitar el proceso de evaluación de la calidad de las aplicaciones de escritorio en el ISMMM.

Para dar respuesta al Problema Científico se plantea como **Idea a defender** que: Si se desarrolla una aplicación para la evaluación de las aplicaciones de escritorio se agilizará esta actividad y además permitirá realizar el seguimiento y comprobar la eficacia de las mismas en el ISMMM, mejorando así la calidad del producto.

Como **Objetivos específicos:**

- Confección del Marco teórico conceptual.
- Elaboración del análisis, diseño e implementación del sistema.
- Realización del Estudio de Factibilidad.
- Implantación del software

Para lograr los objetivos propuestos se han trazado una serie de **tareas** a realizar:

- Revisar de la literatura relacionada con el tema y entrevistas realizadas a especialistas.
- Buscar referencias bibliográficas
- Lograr un mayor conocimiento de las tecnologías y herramientas actuales necesarias para la posterior implementación del sistema.
- Analizar la Metodología para el desarrollo del software.
- Desarrollar el análisis y diseño del Software
- Realizar las pruebas al Software.
- Efectuar el desarrollo de la aplicación.
- Implantar el Software.

Para complementar estas tareas se han empleado **métodos empíricos y teóricos** de la investigación científica. **Los métodos empíricos** ayudan en el descubrimiento de los hechos, información, Procesamiento de datos y en el conocimiento de las características fundamentales del problema, los que posibilitan su estudio y explicación.

Entre los **métodos empíricos** usados podemos citar:

Observación: Se utiliza para ver la funcionalidad de las diferentes áreas de la entidad y el comportamiento del problema.

Entrevista: Se realizó una conversación planificada con el fin de obtener información individual o colectiva y determinar los principales requerimientos del sistema.

Análisis de documentos: Permitió conocer cómo funcionan actualmente el procedimiento de evaluación del software.

Los métodos teóricos se tuvieron en cuenta durante el transcurso de la investigación; pues crearon las condiciones para la interpretación y desarrollo de las teorías de interpretación de los datos obtenidos.

Los **métodos teóricos** utilizados fueron:

Análisis y Síntesis: Se utilizó para la recopilación y el procesamiento de la Información obtenida en los métodos empíricos y arribar a las conclusiones de la investigación.

Hipotético _ deductivo: Para la elaboración de la idea a defender.

El trabajo consta de introducción, 4 capítulos, conclusiones, recomendaciones, bibliografía, glosario de términos y anexos:

En el Capítulo 1, Fundamentación teórica, se ofrece una breve descripción del objeto de estudio y algunos conceptos fundamentales. Se realiza un estudio acerca de los diferentes sistemas existentes vinculados al campo de acción, además se presenta la metodología y las herramientas a utilizar en el desarrollo del sistema propuesto.

En el Capítulo 2, Planeación y Diseño, se hace uso de la metodología expuesta en el capítulo inicial para el desarrollo del proyecto, abordando en detalles cada uno de sus fases.

En el Capítulo 3, Desarrollo y Pruebas, se presentan los principales métodos y definiciones dentro de la implementación de los flujos de trabajo. También se muestran las interfaces gráficas diseñadas para la interacción de los flujos de trabajo con los usuarios. Se describen además las pruebas realizadas y sus resultados.

En el Capítulo 4, Estudio de Factibilidad, se realiza un estudio de los esfuerzos requeridos para la construcción del sistema, y se valora la sostenibilidad del producto.

CAPÍTULO I – FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

La Industria Cubana del Software La Industria Cubana del Software (ICSW) está llamada a convertirse en una significativa fuente de ingresos nacional, como resultado del correcto aprovechamiento de las ventajas del considerable capital humano disponible. Las Universidades cubanas juegan un papel importante en el desarrollo de la ICSW, y en la materialización de los proyectos asociados al programa cubano de informatización.

Aunque la empresa productora de software de nuestro país es muy joven se es fuerza para desarrollar software de calidad que puedan ser insertados en el mercado mundial. Para lograr estos objetivos es necesario garantizar la calidad de un producto de software, determinado por el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (software) satisfará los requisitos dados de calidad.

El siguiente capítulo es el resultado de una investigación acerca de las herramientas y metodologías existentes, con las cuales se le pudiera dar solución a nuestro problema planteado, se realiza una breve referencia sobre el estado del arte del sistema, además de una descripción sobre las tendencias y tecnologías actuales sobre las que se apoya el sistema.

1.2 Estado del Arte

Después de una profunda búsqueda en numerosas fuentes de información y entre

ellas Internet (nacional e internacional), se pudo observar que existen varios sistemas automatizados vinculados al campo de acción que le ocupa a este trabajo. Consideremos los siguientes:

TickIT: Es un programa de certificación de administración de la calidad para el software apoyado sobre todo por las industrias suecas de software. Además de

mejorar la calidad del software, uno de los principios de TickIT es mejorar y regular el comportamiento de auditores que trabajan en el sector de tecnología de información a través de entrenamiento, y la certificación subsiguiente de auditores. El registro internacional de auditores certificados maneja el registro para los auditores de TickIT.

1.2.1 Sistemas Automatizados existentes vinculados al campo de acción

Luego de una minuciosa búsqueda en diferentes páginas web, tanto nacionales como internacionales, se encontró que existen algunos sistemas automatizados vinculados al campo de acción referenciado a este trabajo:

SUMI (Software Usability Measuring Inventory) es un software utilizado para medir la satisfacción y valorar la percepción del usuario de la calidad del software, fue desarrollado por la Universidad College Cork como parte del proyecto MUSiC como una solución a los problemas de medición de la percepción de usabilidad del software por parte del usuario. Proporciona un método válido para la comparación tanto de productos como de diferentes versiones del mismo producto.

SUMI debe ser aplicado a una muestra de usuarios con alguna experiencia con el tipo de software que va a evaluarse para poder obtener resultados confiables. Establece el uso de 10 usuarios representativos para conseguir resultados satisfactorios. Puede ser utilizado para evaluar un producto o serie de productos con el fin de realizar una comparación producto-producto o comparar el producto contra la base de datos estandarizada que permita ver como el producto que esta siendo evaluado compara contra el perfil promedio establecido en el mercado. (ALVA, 2005)

1.3 Conceptos Fundamentales

La gestión de software(GS): Es la disciplina de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el

proyecto dentro del alcance, el tiempo, y coste definidos; garantizando la calidad del producto. La GS comprende la gestión de configuración y la gestión de proyecto.

Calidad de Software: Se define como la ausencia de errores de funcionamiento, la adecuación a las necesidades del usuario, y el alcance de un desempeño apropiado (tiempo, volumen, espacio), además del cumplimiento de los estándares. Los objetivos que la calidad persigue son: La aceptación (utilización real por parte del usuario) y la Mantenibilidad (posibilidad y facilidad de corrección, ajuste y modificación durante largo tiempo. La Calidad del Software es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente. Es el conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas.

1.3.1 Evaluación de Calidad de Aplicaciones de Escritorio.

Este modelo se ha desarrollado en un intento de identificar los atributos más importantes para la calidad interna y externa en un producto software. El modelo identifica seis características claves de calidad [NC-ISO/IEC 9126-1, 2005] donde cada una de ellas se descomponen en un conjunto de sub-características como se muestra en la Figura:

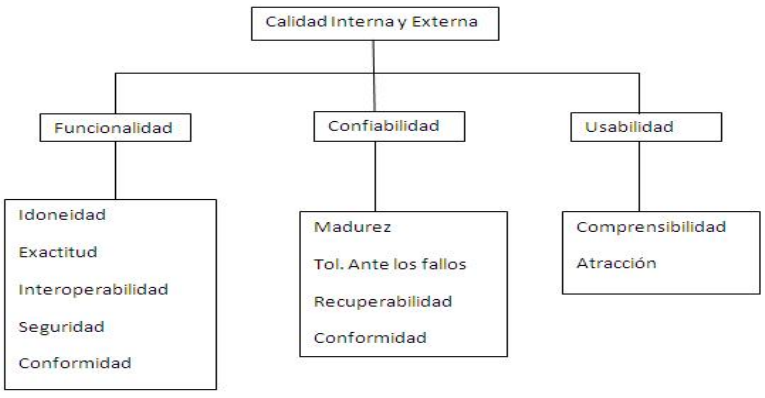


Figura 3.1 Características a evaluar

1.3.2 Las métricas para la medición de la característica Funcionalidad

Las métricas externas de funcionalidad deben ser capaces de medir un atributo como es el comportamiento funcional del sistema en el cual el software está presente. Según [Peña, 2009] estas son:

- **Métricas de idoneidad:** Las métricas externas de idoneidad deben ser capaces de medir un atributo como es la ocurrencia de un funcionamiento insatisfactorio o la ocurrencia de una operación insatisfactoria.

Un funcionamiento u operación insatisfactoria puede ser:

❖ Funcionamiento u operación que no se desempeña de la forma especificada en el Manual de usuario o la especificación de requisitos.

❖ Funcionamiento u operación que no provee una salida aceptable o razonable al tomar en consideración un objetivo específico de las tareas del usuario.

- **Métricas de exactitud:** Las métricas externas de precisión deben ser capaces de medir un atributo como es la frecuencia con que los usuarios se encuentren con la ocurrencia de una falta de exactitud o de precisión, como pueden ser:

❖ Resultados incorrecto o imprecisos causados por datos inadecuados; por ejemplo, un dato con pocos dígitos significativos para un cálculo de precisión.

❖ Inconsistencia entre el procedimiento de operación actual y el descrito en el manual de operación.

❖ Diferencias entre el resultado actual y el razonablemente esperadoproducto de una tarea ejecutada durante la operación.

- **Métricas de interoperabilidad:** Las métricas externas de interoperabilidad deben ser capaces de medir un atributo como es el número de funciones o la ocurrencia de la menor incomunicación que involucre a datos y comandos o instrucciones que sean transferidos entre el producto de software y otros sistemas, otros productos de software u otros equipos a los cuales está conectado.

- **Métricas de seguridad (informática):** Las métricas externas de seguridad (de la información o informática) no se han incluido para el primer nivel de maduración objeto del presente trabajo.

- **Métricas de conformidad de la funcionalidad:** Las métricas externas de conformidad de la funcionalidad deben ser capaces de medir un atributo como lo es el número de funciones con dificultades en la conformidad (o la ocurrencia de problemas de conformidad) con las regulaciones, normas u otras convenciones relacionadas, lo cual haga que el producto de software falle en adherirse a las mismas. Estas métricas no se incluyen para el primer nivel de maduración objeto de este trabajo.

1.3.3 Las métricas para la medición de la característica Confiabilidad

Las métricas externas de confiabilidad deben ser capaces de medir atributos relacionados con el comportamiento del sistema del cual el software forma parte durante la ejecución de las pruebas para indicar la magnitud de la confiabilidad, o sea, seguridad de funcionamiento del software durante la operación del sistema, con las que en la mayor parte de los casos no se distingue entre el software y el sistema. Ellas son:

- **Métricas de madurez:** Las métricas externas de madurez deben ser capaces de medir un atributo como la exención de fallas en el software, causados por la ocurrencia de fallos existentes en el propio software.

- **Métricas de tolerancia ante fallos:** Las métricas externas de tolerancia ante fallos deben estar relacionadas con la capacidad del software de mantener un nivel de ejecución específico en casos de fallos de operación, o se infrinjan las interfaces especificadas.

- **Métricas de recuperabilidad:** Las métricas externas de recuperabilidad deben ser capaces de medir aquellos atributos como son el software y sistemas capaces de restablecer su nivel adecuado de ejecución y recuperar los datos directamente afectados en casos de fallos totales.

- **Métricas de conformidad de la confiabilidad:** Las métricas externas de conformidad de la confiabilidad deben ser capaces de medir un atributo como lo es la cantidad de funciones con dificultades en la conformidad, o la ocurrencia de problemas de conformidad con las regulaciones, normas u otras convenciones relacionadas con la confiabilidad o seguridad de funcionamiento. Dichas métricas no se incluyen para un primer nivel de maduración del presente trabajo.

1.3.4 Las métricas para la medición de la característica Usabilidad

Las métricas externas de usabilidad miden la dimensión con que el software puede ser comprendido, estudiado, operado, atractivo y concordante con las regulaciones y guías relativas a la usabilidad. Resulta recomendable que la evaluación de estas métricas se haga por un grupo (7, 8, aunque menores pueden obtener información de utilidad) de usuarios o evaluadores usuarios simulados o

clonados (pero representativos de un rango de usuarios) sin que reciban asistencia externa alguna. A continuación se brindan las que en una primera etapa serán objeto de utilización.

- **Métricas de comprensibilidad:** Las métricas externas de comprensibilidad deben ser capaces de valorar cómo un nuevo usuario podría comprender:

❖ Si el software es idóneo para la aplicación a la cual lo destina.
❖ Cómo el software puede ser usado para una tarea en particular.

- **Métricas de atracción:** Las métricas externas de atracción deben ser capaces de evaluar la apariencia del software, y van a estar influenciadas por factores tales como el color en la pantalla y su diseño.

- **Métricas de conformidad de la usabilidad:** Las métricas externas de conformidad de la usabilidad deben ser capaces de evaluar la adherencia del software a las regulaciones, normas, convenciones, guías y estilos relativos la usabilidad. Estas métricas no se incluyen en este trabajo.

1.5 Sistemas Automatizados existentes vinculados al campo de acción

En la actualidad existen software que resultan ser eficientes en este aspecto como es el caso de: TickIT

1.6 Propuesta de Solución

Desarrollar una aplicación web que permita agilizar y facilitar el proceso para la evaluación de la calidad de aplicaciones de escritorio, que permita la gestión de los software y el cálculo de las métricas utilizadas, además que de la posibilidad a los usuarios finales a interactuar con dicho sistema para así lograr una mejor calidad. Con el desarrollo del mismo se desea obtener mayor calidad rapidez y organización a la hora de corregir los problemas

encontrados en la entidad. Este software puede llevarse a cabo por los especialistas en la temática o cualquier otra persona interesada en esta actividad.

1.7 Tendencias y tecnologías actuales para el desarrollo de aplicaciones.

Ante el incesante avance de las tecnologías, la sociedad, ávida de nuevas herramientas y funcionalidades, exige a los desarrolladores de software nuevos retos y nuevas concepciones para satisfacer sus exigencias, cada vez más ambiciosas. Para satisfacer estas exigencias, los desarrolladores deben buscar nuevas ideas surgiendo así nuevas metodologías y formas de desarrollo que permitan confeccionar productos cada vez más complejos.

1.8 Lenguajes de programación

Personal Home Page - (PHP) es el acrónimo de procesador hipertexto (Hipertexto Preprocessor). Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Es también un lenguaje interpretado y embebido en el HTML. Fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores. (HERNÁN, 2006)

PHP es un lenguaje de programación de estilo clásico, esto significa que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones, etc. No es un lenguaje de marcas como podría ser HTML, XML o WML.

A diferencia de Java o Java Script que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso permite acceder a los recursos que tenga el servidor, como por ejemplo podría ser, una base de datos. (HERNÁN, 2006)

PHP es la gran tendencia en el mundo de Internet, últimamente puede observar un ascenso imparable, puesto que cada día son muchas las páginas Web que lo utilizan para su funcionamiento, según las estadísticas, PHP se utiliza en más de 10 millones de páginas, y cada mes realiza un aumento del 15%. Como síntesis, PHP corre en 7 plataformas, funciona en 11 tipos de servidores, ofrece

soporte sobre unas 20 bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite; lo cual permite la creación de aplicaciones Web muy robustas, y contiene unas 40 extensiones estables sin contar las que se están experimentando, igualmente tiene soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros, además de que:

- Es software libre, lo que implica menos costes y servidores más baratos que otras alternativas.

- Es muy rápido. Su integración con la base de datos MySQL y el servidor Apache, le permite constituirse como una de las alternativas más atractivas del mercado.

- Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja, de modo que si está familiarizado con esta sintaxis, resultará un poco mejor aprender PHP.

- Su librería estándar es realmente amplia, lo que permite reducir los llamados “costes ocultos”, uno de los principales defectos de ASP.

- PHP tiene una de las comunidades más grandes en internet, esto permite encontrar fácilmente ayuda, documentación, artículos, noticias y otros recursos.

- Permite las técnicas de Programación Orientada a Objetos (POO).

- Posibilita crear los formularios para la Web. (HERNÁN, 2006)

HTML – El lenguaje HTML es el idioma de la Web. Se basa en el uso de “Etiquetas” para la definición del formato del texto, los distintos elementos que conforman la página, sus propiedades y disposición. Este lenguaje es interpretado

por los navegadores, procesado y convertido en una Web tal como la vemos en la pantalla, con imágenes, tablas, texto, videos y toda clase de elementos. El lenguaje está compuesto por etiquetas o marcas, gracias a ellos es posible darles forma a todos los componentes de una página o un documento HTML.

Las etiquetas de HTML están divididas en etiquetas de apertura y de cierre, aunque no siempre existen estas últimas. (HERNÁN, 2006)

Java Script – Es un lenguaje pensado para agregar interactividad con el usuario a las páginas HTML. Permite ejecutar secuencias de comandos en el mismo navegador del usuario. Con Java Script se puede realizar cálculos rápidos y complejos, verificar formularios antes de enviarlos, crear calendarios, convertir divisas. Es un lenguaje que distingue entre minúscula y mayúscula, no exige la declaración explícita de las variables, es posible crear las variables. Es importante saber que Java Script no lo soportan todos los navegadores por lo que nos vemos en la situación de probar el código resultante en más de un navegador. La sintaxis es muy parecida a C o C++, por lo que se convierte en un lenguaje fácil para el que lo domine. (HERNÁN, 2006)

1.9 Sistemas Gestores de Base de Datos (SGBD)

Actualmente existen muchos sistemas gestores de bases de datos, entre ellos, analizaremos las características y facilidades que brinda cada uno de los que se han tenido en consideración, los que siguen son: PostgreSQL y MySQL.

PostgreSQL - es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostGreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, no es un sistema de gestión de bases de datos puramente orientado a objetos. PostgreSQL está considerado como la base de datos de código abierto más

avanzada del mundo y proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. Entre las que se encuentran:

- Implementación del Standard SQL92/SQL99.

- Permite que mientras un proceso escribe en una tabla, otros accedan a la

- Misma tabla sin necesidad de bloqueos. (HERNÁN, 2006)

MySQL - es un sistema de gestión de base de datos relacional, multi-hilo y multiusuario, con más de seis millones de instalaciones. Al contrario de proyectos como el Apache, donde el software es desarrollado por una comunidad pública, y el copyright del código está en poder del autor individual, MySQL está poseído y patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. (HERNÁN, 2006)

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson, y Michael Widenius. (HERNÁN, 2006)

MySQL funciona sobre múltiples plataformas, incluyendo AIX, BSD, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, Novell Netware, OpenBSD, OS/2 Warp, QNX, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista y otras versiones de Windows. (HERNÁN, 2006)

1.10 Herramientas CASE

Embarcadero ER/Studio - es una herramienta de modelado de datos, se usa para el diseño y la construcción lógica y física de bases de datos. Su ambiente es de gran alcance y multinivel. Se diseña para hacer más fácil de entender el estado

actual de los datos de las empresas. Simple y fácil al usuario, ayuda a organizaciones para tomar decisiones en cómo resolver embotellamientos de los datos, elimina redundancia y alcanza en última instancia usos de más alta calidad que entreguen datos más eficientes y exactos a la empresa. (ER/Studio*, 2008*)

Rational Rose - proporciona un lenguaje de modelado común para permitir la creación más rápida de la calidad de software. Incluye Unified Modeling Language (UML) de apoyo y es una de la más completa de productos de la familia Rational Rose. Proporciona la base de datos para el modelado UML diseños, con la capacidad de representar a la integración de los datos y los requisitos de las aplicaciones a través de diseños físicos y lógicos. (Rational Rose*, 2008*)

1.11 Metodologías para el desarrollo de sistemas informáticos

Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en muchos otros.

Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuáles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software. Algunas de las metodologías para el desarrollo informático son:

RUP. (Proceso Unificado del Racional)-Es una metodología para proyectos más largos, debido a su gran cantidad de diagramas que lleva consigo, además se documenta poco sobre el Sistema que se está llevando a cabo.

- Las principales características de este proceso unificado son:
- Dirigido por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.
- Y además utiliza un solo lenguaje de modelación (UML).

XP. (Programación Extrema)-Metodología que adopta 12 prácticas que se pueden utilizar todas o no, eso lo deciden el programador y el cliente según las necesidades de este último o si la aplicación no requiere de todas. Se centra especialmente en documentar en forma de plantillas, tiene cuatro fases:

Planeación, Diseño, Desarrollo o Implementación y Pruebas. En la primera fase se generan como artefactos los usuarios del negocio, las historias de usuarios, la lista de reserva del producto, el plan de iteraciones, entre otros. En la segunda se tiene el modelo de datos, tarjetas CRC. En tercera fase se desarrollaron las tareas de ingeniería y la cuarta fase son efectuadas las pruebas al software para verificar que el mismo cumpla con todos las funcionalidades acordadas, estas pruebas pueden ser aceptadas por el cliente o denegadas por el mismo.

SXP. (Scrum + Programación. Extrema). Esta Metodología fue elaborada en la Universidad de las Ciencias Informáticas, escogieron una parte de Scrum y otra parte de XP, para una mayor organización dividen en carpetas todas sus plantillas. Pero no existe una documentación amplia de esta metodología como en XP.

1.12 Herramientas y tecnologías a emplear en la propuesta de solución.

Macromedia Dreamweaver – Es un editor de texto o un entorno de desarrollo donde el Webmaster puede olvidarse de las partes más tediosas del diseño, como tablas, formularios, y demás elementos.

Es una de las herramientas más utilizadas para el trabajo de aplicaciones visuales, el programa se adapta increíblemente a las necesidades de todo tipo de

profesional del diseño Web tanto como para los que deseen programar el código como para los que gustan de una metodología totalmente visual.

Soporta varios lenguajes tales como: PHP, ASP. HTML, JavaScript o CSS. Otra característica interesante del programa es su integración con Flash y Fireworks también productos de Macromedia.

Permite insertar algunos elementos básicos en Flash sin necesidad de tener este programa instalado, como botones, viñetas y textos. Finalmente si queremos potenciar el programa podemos instalarle gran cantidad de plug-ins, o extensiones, los cuales pueden ser descargados del sitio de Macromedia o bien podemos programarlos nosotros mismos. (HERNÁN, 2006)

XAMPP - Es un paquete independiente de plataforma, software libre, incluye el servidor Web Apache que fue el que se utilizó en la aplicación, además trae el servidor de datos MySQL, sus respectivos gestores phpMyAdmin y phpSQLiteAdmin, el intérprete del lenguaje homónimo PHP con los extras incluidos en PEAR, el intérprete del lenguaje Perl. El programa esta liberado bajo la licencia GNU y actúa como un servidor Web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux. (BASULTO, 2010)

- Incluye chequeo de seguridad.
- Contiene un panel de control.

MYSQL Workbench 5.1 OSS - Después de un estudio de las herramientas que se abordaron anteriormente se escogió esta ya que nos proporciona muchas ventajas.

Si se está comenzando un nuevo diseño o está manteniendo una base de datos existente, se combina con las características para ayudarle a conseguir el trabajo hecho con eficacia.

- La creación de diagramas es clara y rápida.
- Tiene la posibilidad de realizar diagramas con desempeño rápido.

- Te da la posibilidad de darle un nombre a la Base de Datos ante de exportarla.

1.13 Fundamentación de la selección del lenguaje a utilizar.

PHP v5.0 - Las iníciales PHP significan "PHP Hipertexto Pre-processor y se trata de un lenguaje de programación que es usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web. Es un lenguaje de programación usado generalmente para la creación de contenido para sitios o aplicaciones Web. La versión 5 de PHP presenta un magnífico trabajo con el paradigma orientado a objeto que permite la reutilización de código entre otras facilidades.

Ventajas de trabajar con PHP comparado con otros lenguajes similares:

- Es un lenguaje multiplataforma.
- Rapidez de ejecución.
- Mantiene un bajo consumo de recursos de máquina.
- Gran seguridad, muy poca probabilidad de corromper los datos.
- Capacidad de conexión con la mayoría de los manejadores de bases de datos que se emplean en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Posee una amplia documentación en internet, incluyendo una gran variedad de ejemplos y de ayudas.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de programación orientada a objetos.
- Permite crear formularios para la Web.
- No requiere definición de tipos de variables ni manejo detallado de bajo nivel.

1.14 Fundamentación del SGBD a utilizar.

MySQL v5.0 - MySQL es muy rápido, fiable y fácil de usar, surge para manipular bases de datos muy grandes. Es un sistema multiplataforma de base de datos relacionales, lo que da velocidad y flexibilidad, cuenta con un sistema de privilegios contraseñas muy seguro que permite la autenticación básica para el acceso al servidor. MySQL es un sistema de administración de base de datos. Opera en una arquitectura cliente/servidor.

- Es el sistema gestor de bases de datos más popular, además que cualquiera puede estudiar su código y comprenderlo fácilmente.
- Luego de analizadas las características y facilidades del SGBD presentado, y la de la herramienta a desarrollar se decide usar el MySQL como SGBD, por las siguientes razones:
- No se necesitará de un manejo complejo de la información.
- El PHP maneja más fácil al MySQL, debido a la gran cantidad de funciones que tiene explícita.
- El MySQL es multiplataforma.

1.15 ¿Por qué elegir XP?

Actualmente XP es la metodología ágil más documentada (hay una colección de libros “XP Series” de Addison Wesley) y extendido. Existe una gran comunidad de desarrolladores XP. Otra de las ventajas de XP es que no es necesario adoptarlo en forma completa, sino que pueden utilizarse varias de sus prácticas en forma independiente. Esto hace que el costo de su implementación sea mucho más accesible que el de otras metodologías. Algunas de las ventajas que tiene XP:

- Puede ser implementado en forma parcial (elegir sólo algunas de las prácticas)
- Puede ser implementado en forma gradual.
- Puede adaptarse a las necesidades de cualquier equipo de desarrollo. De hecho, Kent Beck recomienda a los equipos que lo adapten a sus necesidades (BECK, 1999).

- Exige que se establezca una comunicación más fluida con el cliente y que este tenga mayor participación en el proceso de desarrollo. La consecuencia de esto es que el cliente se involucre más en el desarrollo del producto.
- Actualmente es la metodología ágil más extendida y documentada.
- Se realizan pruebas constantemente del sistema.

1.16 Patrones Arquitectónicos

Patrón de arquitectura MCV (Modelo vista controlador) - Para el diseño de aplicaciones con sofisticadas interfaces se emplea el patrón de diseño MCV. (BASULTO, 2010)

Si se realiza un diseño ofuscado, es decir, una forma de mezclar los componentes de interfaz y de negocio, entonces, la consecuencia será que, cuando se necesite cambiar la interfaz, tendrá que modificarse trabajosamente los componentes de negocio, por lo que propiciará mayor trabajo y más riesgo de error. (BASULTO, 2010)

Se trata de realizar un diseño que desacople la vista del modelo, con el fin de de perfeccionar la reusabilidad.

Elementos del patrón:

- Modelo: datos y reglas de negocio (Base de Datos).

- Vista: muestra la información del modelo a los usuarios.
- Controlador: gestiona las entradas de los usuarios.

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información en una base de datos, que puede presentarse de diversas formas: diagrama de pastel, de barras, tabular, etc.

El modelo es responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Definir las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: “si Ud. es un profesor y no entra a la aplicación, consultar el Administrador de la misma”.
- Llevar un registro de las vistas y controladores del sistema.
- Si se está en presencia de un modelo activo, el mismo notificará a las vistas los cambios que en los datos pueda producir un agente externo.

El controlador es responsable de:

- Recibir los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo “si el evento z, entonces acción w”. estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada a actualizar.

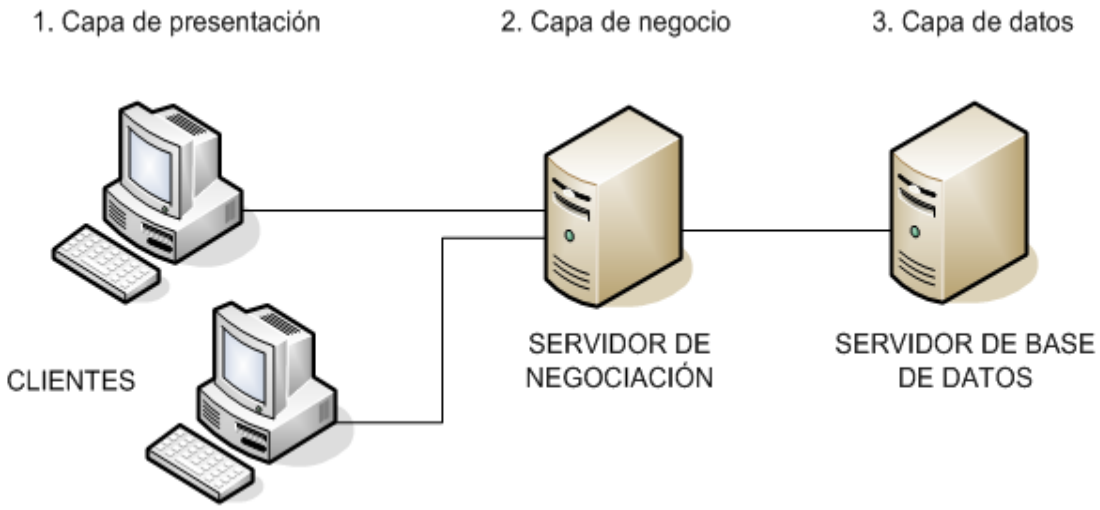
Las vistas son responsables de:

- Recibir datos del modelo y mostrarlo al usuario.
- Pueden suministrar el servicio de actualizar, para que sea invocado por el controlador o por el modelo, (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Arquitectura en tres capas

El diseño de sistemas informáticos suele usar las arquitecturas multinivel o programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite lo que permite el diseño de arquitecturas escalables,

(que pueden ampliarse con facilidad en caso de que las necesidades aumenten). (BASULTO, 2010)



Capas o niveles

- Capa de presentación o interface: es la capa que le permite al usuario interactuar con el sistema, captura y le comunica la información al mismo, dando un mínimo de proceso, (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la del negocio. (BASULTO, 2010)
- Capa de lógica o de negocio: es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio e incluso lógica del negocio, pues es aquí donde se establecen las reglas que deben cumplirse. Esta capa se comunica con la de presentación para recibir las solicitudes y presentar los resultados, y con la capa de datos para solicitar al gestor de bases de datos para almacenar o recuperar datos de él. (BASULTO, 2010)
- Capa de datos: es donde se ubican los datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de los mismos, reciben solicitudes de almacenamiento o de recuperación de información desde la lógica del negocio. (BASULTO, 2010)

Todas estas capas pueden residir en un único ordenador, esto no sería lo normal, lo más usual es que haya una multitud de ordenadores donde reside la capa de interface (son los clientes de la arquitectura cliente/servidor).

Las capas de negocio y de datos pueden residir en un mismo ordenador, y si el crecimiento de las necesidades lo aconseja, pueden dividirse en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, pueden separarse en varios ordenadores los cuáles recibirán las peticiones del ordenador en que resida la capa de negocio. Si por el contrario, la complejidad fuese en la capa de negocio lo que obligase a la separación, esta lógica del negocio podría residir en uno o más ordenadores que realizarían las solicitudes a una única base de datos. (BASULTO, 2010)

En una arquitectura de tres niveles, los términos “capas” y “niveles” no significan lo mismo ni son similares. El término capa hace referencia a la forma como una solución es segmentada desde el punto de vista lógico: interface/lógica del negocio/datos.

En cambio, el término nivel, corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo: Una solución de tres capas (interface, lógica, datos), que residen en un solo ordenador (interface+lógica+datos), se dice que la arquitectura de la solución es de tres capas y un nivel.

Una solución de tres capas que residen en dos ordenadores (interface+lógica, lógica+datos), se dice que la arquitectura de la solución es de tres capas y dos niveles. Una solución de tres capas que residen en tres ordenadores, la arquitectura que la define es: una solución de tres capas y tres niveles. (BASULTO, 2010)

Para la implementación de la aplicación se escogió la arquitectura en tres capas, porque aumenta la escalabilidad del sistema antes futuros cambios y adición de funcionalidades, y nos proporciona la reutilización de soluciones.

1.17 Conclusiones del Capítulo.

En este capítulo quedaron reflejados todos los conceptos relacionados con el problema. Se ha podido apreciar todo lo referente a la base teórica que fundamenta esta investigación, dando a conocer al lector los principales conceptos que se manejan, las diferentes tecnologías para el desarrollo de aplicaciones, los sistemas automatizados que existen en el mundo vinculados al campo donde se proyecta el objeto de estudio así como la propuesta de solución con las herramientas a utilizar, fundamentación del leguaje de programación a utilizar, el Sistema Gestor de Base de Datos y la metodología que se eligió para resolver el problema planteado, así como los patrones definidos para el desarrollo de este trabajo.

CAPÍTULO II – PLANEACIÓN Y DISEÑO

2.1 Introducción

En el presente capitulo, llega a la fase de planeación y diseño, donde se detallan las necesidades del cliente, se describen las funcionalidades que serán objeto de automatización mediante el empleo de las historias de usuarios (HU), se realiza una estimación del esfuerzo necesario para las mismas y se establece un plan de iteraciones necesarias sobre el sistema para su terminación.

Símbolo	Intención	Extensión
especialistas	Los especialistas, serán los usuarios que interactuaran con el sistema, estos serán los administradores y usuarios comunes.	Personas
Usuarios		Personas
Características del software	Se refiere a las características por las cuales se realiza la evaluación de las aplicaciones de escritorio	Recolección de datos
Evaluaciones	Se refiere a todos los tipos de evaluaciones que se le realizaran al software en cuestión antes de llegar a la evaluación final.	-

Tabla 2.1 Definición de entidades y conceptos fundamentales

2.2. Lista de reserva.

De acuerdo a lo antes expuesto, el sistema debe ser capaz de:

- Autenticar usuarios

- Insertar usuarios.
- Eliminar usuarios.
- Mostrar listado de usuarios.
- Cambiar contraseña.
- Insertar proyecto
- Eliminar proyecto.
- Listar proyecto.
- Insertar parámetros.
- Evaluar proyecto.
- Mostrar listado de evaluaciones.
- Comparar proyecto.
- Mostrar resumen de evaluaciones
- Exportar resumen de evaluaciones.

2.2.2 Historias de usuario.

Las HU, son las técnicas utilizadas en XP para detallar los requisitos del software. Son el resultado directo del intercambio entre los usuarios y desarrolladores a través de reuniones donde las conocidas *tormenta de ideas* (*brain storm*) arrojan no solo los requerimientos, sino también las posibles soluciones. Las HU permiten responder ágilmente a los requerimientos cambiantes y aunque se redactan desde

las perspectivas de los clientes, también los desarrolladores pueden brindar ayuda en la identificación de las mismas. Para definirlas se emplea la siguiente plantilla:

Historia de usuario	
Número: No. Historia de usuario	**Usuario:** Usuario entrevistado para obtener la función requerida a automatizar.
Nombre: nombre de la historia de usuario que sirve para identificarla mejor entre los desarrolladores y el cliente.	
Prioridad en el negocio: Importancia: Alta / Media / Baja	**Riesgo en desarrollo:** Dificultad: Alta / Media / Baja
Puntos estimados: Estimación: de 1 a 3 puntos	**Iteración asignada:** Iteración a la que corresponde
Programador responsable: Nombre de encargado de programación.	
Descripción: Se especifican las operaciones por parte del usuario y las respuestas del sistema.	
Observaciones: Algunas observaciones de interés, como glosario, información sobre usuario etc.	

Tabla 2.2. Planilla de historia de usuario.

Las historias de usuario descritas por el cliente:

Historia de usuario	
Número: 1	**Usuario:** Cliente.
Nombre : Autenticación de usuarios	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Humberto Torres Cabrera	
Descripción: Los usuarios del sistema ingresan sus datos para necesitar entrar al sistema (nombre de usuario y contraseña). El sistema verifica que los datos estén correctos, en caso de que nos sean correctos los mismos la	

aplicación muestra un mensaje de error.
Observaciones: Confirmado.

Tabla 2.3 HU No. Autenticar usuarios

Historia de usuario	
Número: 2	**Usuario:** cliente.
Nombre: Gestión de usuarios.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: El Administrador debe de insertar los usuarios, ya sean usuarios que trabajaran directamente en las evaluaciones de los proyectos así como administradores. También puede eliminarlos, cambiarles la contraseña y mostrar un listado de los mismos. El sistema guarda los datos.	
Observaciones: Confirmado.	

Tabla 2.4 HU No. Gestionar Usuario.

Historia de usuario	
Número: 3	**Usuario:** Cliente.
Nombre: Gestión de proyectos	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Los encargados de las evaluaciones de los proyectos podrán insertar un proyecto tecleando su nombre tipo y autor. Además eliminarlos y mostrar un listado de los mismos. El sistema guarda los datos mismos.	

Observaciones: Confirmado.

Tabla 2.5. Gestionar proyectos.

Historia de usuario	
Número: 5	**Usuario:** Cliente.
Nombre: Insertar parámetros.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Los encargados de las evaluaciones podrán insertar los parámetros que se necesitan para poder evaluar un software.	
Observaciones: Confirmado.	

Tabla 2.6. Insertar parámetros.

Historia de usuario	
Número: 6	**Usuario:** Cliente.
Nombre: Evaluar proyectos.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Los encargados de las evaluaciones deben de evaluar los proyectos mediantes la métrica a utilizar.	

Observaciones: Confirmado.

Tabla 2.7. Evaluar proyectos.

Historia de usuario	
Número: 7	**Usuario:** Cliente.
Nombre: Mostrar listado de evaluaciones.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Los encargados de las evaluaciones pueden mostrar el listado de las evaluaciones, así como reevaluar un proyecto.	
Observaciones: Confirmado.	

Tabla 2.8. Mostrar listado de evaluaciones.

Historia de usuario	
Número: 8	**Usuario:** Cliente.
Nombre: Comparar proyecto.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Los encargados de las evaluaciones podrán comparar los proyectos.	

Observaciones: Confirmado.

Tabla 2.9. Comparar proyecto.

Historia de usuario	
Número: 9	**Usuario:** Cliente.
Nombre: Mostrar resumen de evaluaciones.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** media
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Los encargados de las evaluaciones podrán mostrar un resumen del software que desee.	
Observaciones: Confirmado.	

Tabla 2.10. Mostrar resumen de evaluaciones.

Historia de usuario	
Número: 9	**Usuario:** Cliente.
Nombre: Exportar resumen de evaluaciones**.**	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** media
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Los encargados de las evaluaciones podrán exportar el resumen de las evaluaciones.	

Observaciones: Confirmado.

Tabla 2.11. Exportar resumen de evaluaciones**.**

2.3 Planificación de entregas.

En esta fase el cliente establece la prioridad de cada historia de usuario, y seguidamente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Las

estimaciones de esfuerzo asociado a la implementación de las HU las establecen los programadores utilizando como medida el punto de estimación. Un punto de estimación se considera como una semana ideal de trabajo, donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción, este punto de estimación que se utiliza para representar la semana ideal, es de 5 días. Por otra parte, en esta fase, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos de estimación correspondientes a las historias de usuario que fueron terminadas en la última iteración. Esta fase dura solamente unos pocos días.

2.3.1-Estimación de esfuerzos por historias de usuarios

Para el buen desarrollo del sistema propuesto, se realizó una estimación para cada una de las historias de usuario identificadas, llegando a los resultados que se muestran a continuación:

Historias de Usuario	Puntos de Estimación
Autenticar usuarios.	1 semana
Gestionar usuarios.	1 semana
Gestionar proyectos.	1 semana
Insertar parámetros.	1 semana

Evaluar el proyecto.	2 semana
Mostrar listado de evaluaciones.	1 semana
Comparar proyecto.	2 semanas
Mostrar resume de evaluaciones.	2 semanas
Exportar resume de evaluaciones.	2 semanas

Tabla 2.13- Estimación de esfuerzo por historia de usuario

2.3.2- Planificación de Iteraciones

Una vez descritas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se procede a realizar la planificación de la etapa de implementación del sistema. En este plan se establece cuántas iteraciones serán necesarias realizar sobre el sistema para su terminación. El plan de iteraciones puede incluir indicaciones sobre cuáles HU se incluirán en un release, lo cual debería ser consistente con el contenido de una o dos iteraciones. En relación con lo antes tratado se decide realizar el sistema en 3 iteraciones, las cuales se describen detalladamente a continuación:

Iteración 1:

Esta iteración tiene como objetivo darle cumplimiento a las HU que se consideraron de mayor importancia para el desarrollo del framework. Al concluir dicha iteración se contará con todas las funcionalidades descritas en las HU 1, 2, 3, 4 y 5, las cuales hacen alusión al tipo de navegación que pueden tener los modelos de desarrollo (ya sea lineal o en cascada), al módulo de actividades interactivas, a todo lo referente a la carga de la configuración, tipos de fondos musicales y todo lo involucrado con la carga de medias. Además se tendrá la primera versión de prueba, que contará con dos modelos de desarrollo que incorporan todas las funcionalidades antes vistas, éstos modelos se le presentarán al cliente con el objetivo de obtener una retroalimentación del mismo para posteriores iteraciones del producto.

Iteración 2:

Esta iteración tiene como finalidad desarrollar las HU que responden a los números 6 y 7. Dichas HU son las que brindan las funcionalidades de reproducción de contenidos y visualización de imágenes a través de una galería de imágenes. Además permiten a los usuarios controlar el estado de la reproducción de los contenidos, mediante algunas funciones básicas como detener, pausar o comenzar la reproducción, subir y bajar el volumen de la misma así como brindar la posibilidad de que el usuario vaya a un punto específico del archivo mediante una barra de desplazamiento. La versión que se obtendrá de esta iteración en unión con la entregada en la iteración anterior se le dará al cliente para verificar si cumple con las necesidades antes acordadas con él.

Iteración 3:

Esta es la última iteración del framework, en la que se dará cumplimiento a las HU 8 y 9. Dichas HU cumplen con las funcionalidades de impresión de documentos contenidos en el producto multimedia y brindan la posibilidad de hacer búsquedas de información referente a algún tema que sea de interés para los usuarios que interactúen con el software. Estas HU son integradas con el resultado de las iteraciones anteriores y como fruto de esta integración se obtendrá la versión 1.0 del producto final, que contiene tres modelos de desarrollos, que muestran la utilidad del framework y la arquitectura desarrollada. A partir de este momento el software será puesto a un proceso de prueba para evaluar el desempeño del mismo.

2.3.3-Plan de duración de las iteraciones

Iteración	Historias de usuario	Duración total
Iteración 1	Autenticar usuarios.	4 semanas
	Gestionar usuarios.	
	Gestionar proyectos.	
	Insertar parámetros.	
Iteración 2	Evaluar el proyecto.	5 semanas
	Mostrar listado de evaluaciones.	

	Comparar proyecto.	
Iteración 3	Exportar resumen de evaluaciones.	4 semanas
	Mostrar resume de evaluaciones.	

Tabla 2.14 Plan de duración de las iteraciones

2.4-Tarjetas CRC

Las tarjetas CRC forman parte de las técnicas propuestas por algunos de los creadores de la metodología ágil XP (Ward Cunningham y Kent Beck), con el objetivo de obtener un diseño simple y que no incurra en la implementación de funcionalidades que no son necesarias. Esta técnica de modelado permite entender las características del sistema pensando en términos de objetos y clases. Debido a la gran facilidad de uso y entendimiento, que propician dichas tarjetas, el equipo de desarrollo del presente trabajo, decidió utilizarlas para diseñar el framework que se desea desarrollar. Además de crear un pequeño diagrama de clases con el objetivo de que se tenga una mejor visión del diseño del framework, para aquellos desarrolladores que en un futuro pudieran trabajar en él, y no tenga mucha experiencia en el uso de las tarjetas CRC.

Clases	
Responsabilidades	**Colaboraciones**
Clase A	Clase B

Tabla 2.15 Plantilla de tarjeta CRC.

Clases: Autenticar _ usuarios	
Responsabilidades	**Colaboraciones**
Autenticar usuarios	Conexion.

Tabla 2.16 Tarjeta CRC No.1 Autenticar usuarios

Nombre de la clase: Autenticar Usuarios.	
Tipo de la clase: Lógica del negocio.	
Responsabilidades:	**Colaboradores:**
Insertar nombre _ usuario.	
Insertar contraseña	

Tabla 2.17 Tarjeta CRC No.1 Autenticar usuarios.

Nombre de la clase: Gestionar Usuario	
Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Insertar los datos del usuario. Eliminar los datos del usuario. Cambiar contraseña del usuario. Mostrar listado de los usuarios	insertar_usuarios. modificar_usuarios. modificar_usuarios. listar_usuarios.

Tabla 2.18 Tarjeta CRC No.1 Gestionar usuarios

Nombre de la clase: Gestionar Proyecto	
Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Insertar los datos del proyecto.	insertar_proyectos.
Eliminar los datos del proyecto.	Listar_proyectos.
Mostrar listado del proyecto.	Listar_proyectos.

Tabla 2.19 Tarjeta CRC No.1 Gestionar Proyecto

Nombre de la clase: Insertar parámetros.	
Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Insertar los parámetros requeridos.	insertar_parametros.

Tabla 2.20 Tarjeta CRC No.1 Insertar parámetros.

Nombre de la clase: Evaluar Proyecto	
Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Evaluar el proyecto	Evaluar_soft.

Tabla 2.21 Tarjeta CRC No.1 Evaluar el proyecto.

Nombre de la clase: Mostrar listado de Evaluaciones.	
Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Mostrar las evaluaciones	Listar_soft_eval

Tabla 2.22Tarjeta CRC No.1 Mostrar listado de Evaluaciones.

Nombre de la clase: Comparar Proyecto.	
Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Comparar Proyecto	comparar_proyectos.

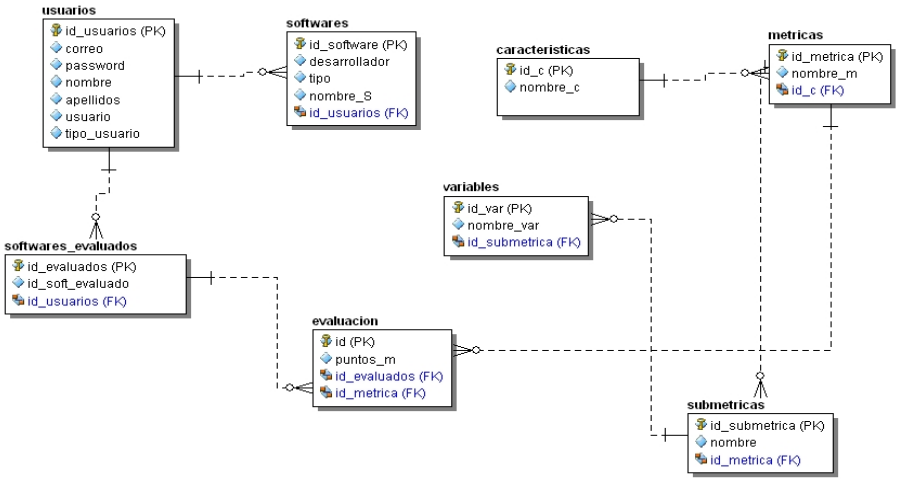
Tabla 2.23Tarjeta CRC No.1 Mostrar listado de Evaluaciones.

Nombre de la clase: Mostrar resumen de evaluaciones.	
Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Mostrar resumen de evaluaciones...	Generar_ resumen_ valorativo.

Tabla 2.13 Tarjeta CRC No.24 Mostrar resumen de evaluaciones.

2.5 - Modelo de datos

Es el modelo físico de la Base de Datos con la cual trabaja esta Aplicación Web, nos sirve para entender con más facilidad como está diseñada nuestra base de datos y las relaciones de las mismas, así como el tipo de relación que tienen estas.



2.6- Conclusiones

En este capítulo se abordo la fase de planeación y diseño donde se delinearon las HU con la participación del cliente, se llevó a efecto la planificación de iteraciones de cada HU a partir de la estimación del esfuerzo necesario de las mismas. Además presentando las principales clases mediante el empleo de las tarjetas CRC, estamos listos para pasar a la siguiente etapa de desarrollo y pruebas.

CAPÍTULO III – DESARROLLO Y PRUEBAS

3.1 Introducción

En este capítulo se inicia la etapa de desarrollo y pruebas correspondiente a la metodología XP. Se presenta el modelo de datos empleado para la aplicación concluyente, y se realiza el desarrollo de las iteraciones a partir del desglose de las historias de usuario en tareas. Asimismo aparecen las interfaces graficas de usuario diseñadas para la aplicación final.

3.2-Desarrollo de las iteraciones

Durante la fase planificación y diseño fueron detalladas las historias de usuario correspondientes a cada una de las iteraciones a desarrollar, teniendo en cuenta las prioridades y restricciones de tiempo, previstas por el cliente.

Para darle cumplimiento a cada HU, primeramente se debe realizar una revisión del plan de iteraciones, y si es necesario, se le hacen modificaciones a este.

3.2.1-Tareas por historias de usuario

Dentro del contenido de este plan, las HU se descomponen en tareas de programación o ingeniería, y a su vez, estas son asignadas al equipo de desarrollo para su implementación. Las tareas no tienen que ser entendidas necesariamente por el cliente, pues las mismas, sólo son utilizadas por los miembros del equipo de desarrollo, por lo que pueden ser escritas en lenguaje técnico. Las mismas se representan mediante las tarjetas de tareas. (BASULTO, 2010)

Historia de usuario	**Tareas**
Autenticar usuarios	Insertar los datos para entrar al sistema.
Gestionar usuarios	Insertar usuarios. Eliminar usuarios. Mostrar listado de usuarios. Cambiar contraseña.

Gestionar proyecto	Insertar proyecto. Eliminar proyecto. Mostrar listado de proyecto.
Gestionar parámetros	Insertar parámetros.
Evaluar el proyecto	Evaluar el proyecto.
Gestionar evaluaciones	Mostrar las evaluaciones.
Comparar proyecto	Comparar proyecto
Mostrar resumen valorativo	Mostrar resumen valorativo
Exportar resumen valorativo	Exportar resumen valorativo

Tabla 3.1 Tareas por Historias de usuarios

3.2.2-Iteración No.1

En esta iteración se le dio cumplimiento a la implementación a las HU que se consideraron de mayor importancia para el desarrollo de la aplicación, con el fin de obtener una versión del software.

Historias de usuario abordadas en la primera iteración.

Historias de usuario	**Tiempo de estimación (semanas)**
Autenticar usuarios.	1 semana
Gestionar usuarios.	1semana
Gestionar proyectos.	1semana
Gestionar parámetros	1semana

Tabla 3.2 Iteración 1

Tarea ingeniería	
Número tarea: 1	**Número historia**: 2
Nombre tarea: Insertar usuario.	

Tipo de tarea : Desarrollo	Puntos estimados: 1
Fecha inicio:	**Fecha fin:**
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Esta tarea facilita insertar los usuarios.	

Tabla 3.3 Tarjeta de tarea No. 1 Insertar usuario.

Tarea ingeniería	
Número tarea: 2	**Número historia:** 2
Nombre tarea: Eliminar Usuario.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio:	**Fecha fin:**
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Esta tarea facilita eliminar los usuarios.	

Tabla 3.4 Tarjeta de tarea No. 2 Eliminar Usuario.

Tarea ingeniería	
Número tarea: 3	**Número historia:** 2
Nombre tarea: Cambiar contraseña.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio:	**Fecha fin:**
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Esta tarea facilita cambiar la contraseña.	

Tabla 3.5 Tarjeta de tarea No. 3 Cambiar contraseña.

Tarea ingeniería	
Número tarea: 4	**Número historia:** 2
Nombre tarea: Mostrar listado de usuarios.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio:	**Fecha fin:**
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Esta tarea muestra un listado de los usuarios.	

Tabla 3.6 Tarjeta de tarea No. 4 Mostrar listado de usuarios.

Tarea ingeniería	
Número tarea: 5	**Número historia:** 3
Nombre tarea: Insertar Proyecto.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio:	**Fecha fin:**
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Esta tarea inserta un proyecto.	

Tabla 3.7 Tarjeta de tarea No. 5 Insertar Proyecto.

Tarea ingeniería	
Número tarea: 6	**Número historia:** 3

Nombre tarea: Eliminar Proyecto.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio:	**Fecha fin:**
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Esta tarea elimina los Proyecto.	

Tabla 3.8 Tarjeta de tarea No. 6 Eliminar Proyecto.

Tarea ingeniería	
Número tarea: 7	**Número historia:** 3
Nombre tarea: Mostrar listado de los Proyectos.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio:	**Fecha fin:**
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Esta tarea muestra un listado de los proyectos.	

Tabla 3.9 Tarjeta de tarea No. 7 Mostrar listado de los Proyectos.

Tarea ingeniería	
Número tarea: 8	**Número historia:** 4
Nombre tarea: Insertar parámetros.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio:	**Fecha fin:**
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: Esta tarea inserta los parámetros para la evaluación.	

Tabla 3.10 Tarjeta de tarea No. 8 Insertar parámetros.

3.2.3-Iteración No.2

Historias de usuario abordadas en la segunda iteración.

Historias de usuario	**Tiempo de estimación (semanas)**
Evaluar el proyecto.	2
Mostrar listado de evaluaciones.	1
Comparar proyecto.	2

Tabla 3.11 Iteración 2

Historia de usuario	
Número: 6	**Usuario:** Cliente.
Nombre: Evaluación de los proyectos.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Ramón Alejandro Velázquez Díaz.	
Descripción: El encargado deberá evaluar los proyectos a través de los parámetros que deberá insertar.	
Observaciones: Confirmado.	

Tabla 3.12 Tarjeta de tarea No. 9 Evaluación de los proyectos

Historia de usuario	
Número: 8	**Usuario:** Cliente.
Nombre: Comparación de proyectos	

Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Humberto Torres Cabrera.	
Descripción: El especialista y el estudiante podrán comparar los proyectos.	
Observaciones: Confirmado.	

Tabla 3.13 Tarjeta de tarea No. 10 Insertar parámetros.

3.2.4-Iteración No.3

Historias de usuario abordadas en la tercera iteración.

Historias de usuario	Tiempo de estimación (semanas)
Mostrar resumen valorativo.	2
Exportar resumen valorativo	2

Tabla 3.20 Iteración 2

3.3- Principales Interfaces de la aplicación

Aquí se verán algunas de las principales interfaces de la aplicación llevadas a cabo en la fase de implementación de la metodología expuesta en el capítulo inicial.

- **Interfaz principal de los profesores**
- **Interfaz de evaluación.**
- **Interfaz del resultado de la evaluación**
- **Interfaz de la comparación de los software.**

3.4-Pruebas

En la metodología XP las pruebas juegan un papel fundamental, pues esta permite la comprobación continua del código. El desarrollo constante de las pruebas da lugar a que se desarrolle un software con mayor calidad dando una mayor seguridad de lo que se está haciendo. En esta metodología hay dos tipos

de testing; las unitarias o desarrollo dirigido por pruebas, desarrolladas por los programadores verificando su código de forma automática, y las pruebas de aceptación, las cuáles son evaluadas luego de culminar una iteración verificando así que se cumplió la funcionalidad requerida por el cliente.

3.4.1-Desarrollo dirigido por pruebas

El desarrollo dirigido por pruebas, se enfoca en la implementación orientada a pruebas. El código debe ser probado paso a paso para lograr un resultado, aunque no con lógica para el negocio, pero si funcional. Algunas personas confunden este término con las llamadas “pruebas de caja blanca” las cuáles se les practican a los métodos u operaciones para medir la funcionalidad del mismo, desde el punto de vista de validez del cliente. Sin embargo, el TDD se aplica antes de comenzar a implementar cada paso de la tarea en desarrollo, asumiendo que la prueba es insatisfactoria desde un inicio. Sólo una vez que se haya cumplido de la forma más sencilla posible la lógica del código a probar se asume como cumplida. Luego se realiza un proceso conocido como refactorización de código perteneciente a una de las doce prácticas planteadas por la metodología XP, el cual consiste en mantener el código en buen estado, modificándolo activamente para que conserve claridad y sencillez. (BASULTO, 2010)

3.4.2- Pruebas de aceptación

Las pruebas de aceptación en XP, se pueden asociar con las pruebas de caja negra que se aplican en la metodología RUP, sólo que se crean a partir de las historias de usuario y no por un listado de requerimientos. Durante las iteraciones, las HU se traducen a pruebas de aceptación. En ellas se especifican desde la perspectiva del cliente, los escenarios para probar que una historia de usuario ha sido implementada correctamente. La misma puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo que persiguen estas pruebas, es garantizar que las funcionalidades solicitadas por el cliente han sido realizadas satisfactoriamente. Una HU no se considera completa hasta que no ha transitado por sus pruebas de aceptación.

Luego de ver lo plasmado anteriormente y el autor reunirse con el cliente para su análisis, se decidió llevar a cabo el proceso mediante las pruebas de aceptación.

La planilla utilizada para plasmar el contenido de las pruebas de aceptación se muestra a continuación. (COLUMBIÉ, 2010)

Prueba de aceptación
HU: Nombre de la historia de usuario que va a comprobar su funcionamiento.
Nombre: Nombre del caso de prueba.
Descripción: Descripción del propósito de la prueba.
Condiciones de ejecución: Precondiciones para que la prueba se realice.
Entrada/Pasos ejecución: Pasos para probar la funcionalidad.
Resultado: Resultado que se desea de la prueba.
Evaluación de la prueba: (Aceptada o denegada)

Tabla 3.24 Planilla de prueba de aceptación.

Prueba de aceptación
HU: Autenticar usuarios
Nombre: Prueba para comprobar la autenticación de usuarios. (Nombre de usuario y contraseña).
Descripción: Validación de entrada de los datos de los usuarios.
Condiciones de ejecución: El usuario debe introducir su nombre de usuario y contraseña.
Entrada/Pasos ejecución: El usuario escribe su nombre de usuario y contraseña y luego da clic en el botón Entrar.
Resultado: • Si el usuario tiene acceso para entrar a la aplicación e inserta sus datos correctamente entrará sin problemas al Sistema. Se emite un mensaje de error en caso de que: • Se inserte los datos de un usuario no válido para el Sistema. (Ya sea su nombre de usuario o su contraseña).

• Se dé clic en el botón Entrar sin llenar todos los campos de texto.
Evaluación de la prueba: Aceptada.

Tabla 3.25 Prueba para comprobar la autenticación de usuarios.

Prueba de aceptación
HU: Gestión de usuarios
Nombre: Prueba para verificar la gestión de usuarios.
Descripción: Validación de la gestión de usuarios.
Condiciones de ejecución: El usuario debe de entrar a la aplicación y el mismo tiene que ser administrador de esta para poder insertar usuarios, eliminarlos, cambiarles su contraseña sin necesidad de pedir su contraseña actual, así como listar todos los usuarios con acceso al Sitio.
Entrada/Pasos ejecución: El administrador escribe todos los datos que se le piden para poder insertar un usuario determinado asignándole el tipo de usuario, luego presiona el botón Insertar. Después que el usuario este insertado es que se le puede cambiar su contraseña, eliminarlo y ver sus datos en el listado de usuarios.
Resultado: • Si se insertó el usuario correctamente muestra un mensaje de confirmación. • Cuando se cambia la contraseña se emite otro mensaje de confirmación. Se emite un mensaje de error en caso de que: • Falten datos del usuario a la hora de insertarlo. • Se inserte un usuario que ya exista, es decir que tenga ese nombre. • A la ahora de confirmar la contraseña no coincida con la que se puso anteriormente. • Cuando se va a eliminar el usuario administrador, ya que este está definido por el Sistema. • Se dé clic en botón eliminar sin antes seleccionar ningún usuario.

• Cuando se le va a cambiar la contraseña a un usuario que no exista o se escriba incorrectamente.
Evaluación de la prueba: Aceptada.

Tabla 3.26 Prueba para comprobar la Gestión de usuarios.

Prueba de aceptación
HU: Gestionar Proyectos
Nombre: Prueba para verificar la gestión de proyectos
Descripción: Validación de gestión de proyectos.
Condiciones de ejecución: El usuario debe de entrar a la aplicación y el mismo tiene que ser un usuario comun de esta para poder insertar proyectos, eliminarlos, y tanto el cómo los estudiantes podrán listar todos los proyectos que han sido insertados
Entrada/Pasos ejecución: El usuario escribe todos los datos que se le piden para poder insertar un nuevo software determinado, luego presiona el botón Insertar. Después que el software este insertado es que se le puede eliminar y ver sus datos en el listado de software.
Resultado: Se emite un mensaje de error en caso de que: • Falten datos del software a la hora de insertarlo. • Se dé clic en el botón insertar sin llenar los campos de texto. • Si se entra el nombre de un software que ya este insertado. • Se dé clic en botón eliminar sin antes seleccionar ningún software.
Evaluación de la prueba: Aceptada.

Tabla 3.27 Prueba para comprobar la Gestión de proyectos

.

Prueba de aceptación
HU: Insertar parámetros.

Nombre: Prueba para verificar la inserción de los parámetros.
Descripción: Validación de la inserción de los parámetros.
Condiciones de ejecución: El usuario debe de entrar a la aplicación y el mismo tiene que ser un usuario común de esta para poder insertar un cada parámetros con lo que se trabajará en la evaluación.
Entrada/Pasos ejecución: El usuario escribe todos los parámetros que se necesitan para la evaluación, luego presiona el botón evaluar.
Resultado: Se emite un mensaje de error en caso de que: • Falten datos. • Se dé clic en el botón evaluar sin llenar los campos de texto.
Evaluación de la prueba: Aceptada.

Tabla 3.27 Prueba para comprobar la Inserción de parámetros.

Prueba de aceptación
HU: Evaluar proyectos.
Nombre: Prueba para verificar la evaluación de los proyectos.
Descripción: Validación de la evaluación de los proyectos.
Condiciones de ejecución: Podrán evaluar el proyecto todas las personas que sean usuarios del sistema..
Entrada/Pasos ejecución: El usuario una vez dentro de la aplicación se dirigen al menú de navegación, a la pestaña Evaluar proyecto, selecciona el proyecto a evaluar, y una vez seleccionado podrá insertar todos los parámetros del sistema a evaluar.
Resultado: Se emite un mensaje de error en caso de que: • No se seleccione un software apara la evaluación. • No se encuentren software insertados

Evaluación de la prueba: Aceptada.

Tabla 3.28 Prueba para comprobar la Evaluación de proyectos.

Prueba de aceptación
HU: Comparar proyecto
Nombre: Prueba para verificar la comprobación de los proyectos
Descripción: Validación de comparar proyecto.
Condiciones de ejecución: El usuario debe de entrar a la aplicación y el mismo tiene que ser un usuario comun de esta para poder compara proyectos, tendrán que elegir los software para la realización de la evaluación.
Entrada/Pasos ejecución: El usuario tienen que elegir los dos software que quieran comparar y presionar el botón comparar.
Resultado: Se emite un mensaje de error en caso de que: • Falten seleccionar los dos software a la hora de compararlo. • Falten seleccionar uno de los software a la hora de compararlo. • Se seleccionen dos software iguales para la comparación.
Evaluación de la prueba: Aceptada.

Tabla 3.29 Prueba para comprobar la Evaluación de proyectos.

Prueba de aceptación
HU: Mostrar resumen valorativo.
Nombre: Prueba para verificar si se mostró el resumen.
Descripción: Validación de la mostración del resumen valorativo.

Condiciones de ejecución: El usuario podrá mostrar un resumen valorativo del software que desee.
Entrada/Pasos ejecución: Los usuarios una vez dentro de la aplicación se dirigen al menú de navegación, a la pestaña resumen, y damos clic, ahí para mostrar el resumen del software deseado.
Resultado: • Se muestra un resumen detallado de las evaluaciones de los software que se han evaluados.
Evaluación de la prueba: Aceptada.

Tabla 3.30 Prueba para comprobar la Evaluación de proyectos.

Prueba de aceptación
HU: Exportar Resumen
Nombre: Prueba para verificar si se exportó el resumen
Descripción: Validación de la exportación del resumen
Condiciones de ejecución: El usuario tiene que tener privilegios de usuario común, para poder texportar el resumen de evaluaciones.
Entrada/Pasos ejecución: Los usuarios una vez dentro de la aplicación se dirigen al menú de navegación, a la pestaña exportar resumen, y damos clic. Ahí para exportar el resumen de todas las evaluaciones realizadas.
Resultado: • Se exportó un resumen detallado con todas las evaluaciones que se han hecho y aún están guardadas en la Base de Datos.
Evaluación de la prueba: Aceptada.

Tabla 3.31 Prueba para comprobar la Evaluación de proyectos.

3.5-Conclusiones

En este capítulo se llevó a cabo la fase de desarrollo y pruebas, Se realizó el desarrollo de las iteraciones a partir de la distribución de tareas por historias de usuarios y se le practicó las pruebas de aceptación a las mismas para verificar que las funcionalidades de la aplicación estén correctamente implementadas, siendo todas estas aceptadas por el cliente.

CAPÍTULO IV – ESTUDIO DE FACTIBILIDAD

4.1 Introducción

En la actualidad para un satisfactorio desarrollo de cualquier proyecto se hace imprescindible el estudio de factibilidad para tener en cuenta una estimación de los costos a incurrir logrando así definir si será factible o no desarrollar el mismo. Para el estudio de factibilidad de este proyecto se utilizará la metodología Costo Efectividad (Beneficio), la cual plantea que la conveniencia de la ejecución de un proyecto se determina por la observación de dos factores en conjunto:

- El costo que involucra la implementación de la solución informática, adquisición y puesta en marcha del sistema hardware/software y los costos de operación asociados.
- La efectividad que se entiende como capacidad del proyecto para satisfacer la necesidad, solucionar el problema o lograr el objetivo por el cual se ideó, es decir, un proyecto será más o menos efectivo con relación al mayor o menor cumplimiento que alcance en la finalidad para la cual fue ideado (costo por unidad del cumplimiento del objetivo). (PÉREZ, 2008)

4.2 Efectos Económicos.

- Efectos directos.
- Efectos indirectos.
- Efectos externos.
- Intangibles.

Efectos directos:

Positivos:

- Los usuarios con acceso al sistema tendrán la posibilidad de evaluar de forma personalizada cualquier software.
- Se mejora la eficiencia y calidad del proceso de gestión de evaluación de aplicaciones de escritorio.

- Se agilizará el proceso de evaluación en cada departamento.
- Se obtendrá un resumen con todas las evaluaciones realizadas (usuario que evalúa, resultado, aspectos evaluados y clasificación).
- Se facilitará el proceso de llevar todas las evaluaciones a un formato para imprimir.

Negativos:

- Para el uso de esta aplicación implementada en plataforma Web se necesitará que la misma sea ejecutada con el navegador Mozilla Firefox, porque es con el que se trabajó en la elaboración del producto por lo que el diseño está adaptado a este tipo de navegador.

Efectos indirectos:

- Los efectos económicos observados que pudiera repercutir sobre otros mercados no son perceptibles, aunque este proyecto no está construido con la finalidad de venta.

Efectos externos:

- Se obtendrá un producto disponible que le facilitará gran parte del trabajo a los profesores encargados de la gestión para la evaluación de la calidad de aplicaciones de escritorio.

Intangibles:

- En la valoración económica siempre hay elementos como perjuicio o beneficio, pero al momento de ponderar en unidades monetarias esto resulta difícil o prácticamente imposible.

A fin de medir con precisión los efectos, deberán considerarse dos situaciones:

- **Situación sin el producto.**

Para llevar a cabo la gestión de evaluación de la calidad de aplicaciones de escritorio se deben tener en cuenta los siguientes pasos:

1. Los usuarios deben de realizar la evaluación al software a través de la evaluación de las características a evaluar y el cálculo de las métricas correspondiente a cada característica..
2. Hacerle llegar esta información a los encargados del proceso.

- **Situación con el producto.**

Para la entrada de los datos al sistema propuesto debemos seguir los siguientes pasos:

1. Cargar la aplicación en el navegador de la computadora.
2. Entrar los datos correspondientes del usuario que hará uso de la aplicación.
3. Realizar la evaluación al software deseado..
4. Mostrar un resumen de las evaluaciones.
5. Exportar el resumen con las evaluaciones.

4.3 Beneficios y costos Intangibles en el proyecto

Costos

- Resistencia al cambio.

Beneficios

- Mejor comodidad para los usuarios.
- Mejora la calidad de evaluación.
- Menor tiempo empleado en el proceso de evaluación.
- Conectividad desde cualquier PC que esté conectada a la red.

4.4 Ficha de costo

Para determinar el costo económico del proyecto se utilizará el procedimiento para elaborar una ficha de costo de un producto. Para la elaboración de la ficha se consideran los siguientes elementos de costo, desglosados en moneda libremente convertible y moneda nacional. (ZALDIVAR, 2010)

4.4.1 Costos en Moneda Libremente Convertible (C.U.C)

Ficha de Costo.	
	Precio(s)
Costos Moneda Libremente Convertible	
Costos Directos	
Compra de equipos de cómputo	*0,00*
Alquiler de equipos de cómputo	*0,00*
Compra de licencia de Software	*0,00*
Depreciación de equipos	*0,43*
Materiales directos	*0,00*
Subtotal	*0,43*
Costos Indirectos	
Formación del personal que elabora el proyecto	*0,00*
Gastos en llamadas telefónicas	*0,00*
Gastos para el mantenimiento del centro	*0,00*
Know How	*0,00*
Gastos en representación	*0,00*
Subtotal	*0,00*
Gastos de Distribución y Venta	
Participación en ferias o exposiciones	*0,00*
Gastos en transportación	*0,00*
Compra de materiales de propagandas	*0,00*
Subtotal	*0,00*
Total	*0,43*

Tabla 4.1: Costo en Moneda Libremente Convertible

4.4.2 Costos en moneda nacional (M.N):

Ficha de Costo.	
	Precio(s)
Costos Moneda Nacional	
Costos Directos	
Salario del personal que laborará en el proyecto	*600,00*
12,5% del total de gastos por salarios se dedica a la seguridad social	*0,00*
9.09% de salario total, por concepto de vacaciones a acumular	*0,00*
Gasto por consumo de energía eléctrica	*275,93*
Gastos en llamadas telefónicas	*0,00*
Gastos administrativos	*0,00*
Subtotal	*875,93*
Costos Indirectos	
Gasto por consumo de energía eléctrica	*0,00*
Know How	*0,00*
Subtotal	*0,00*
Total	*875,93*

Tabla 4.2: Costo en Moneda Nacional.

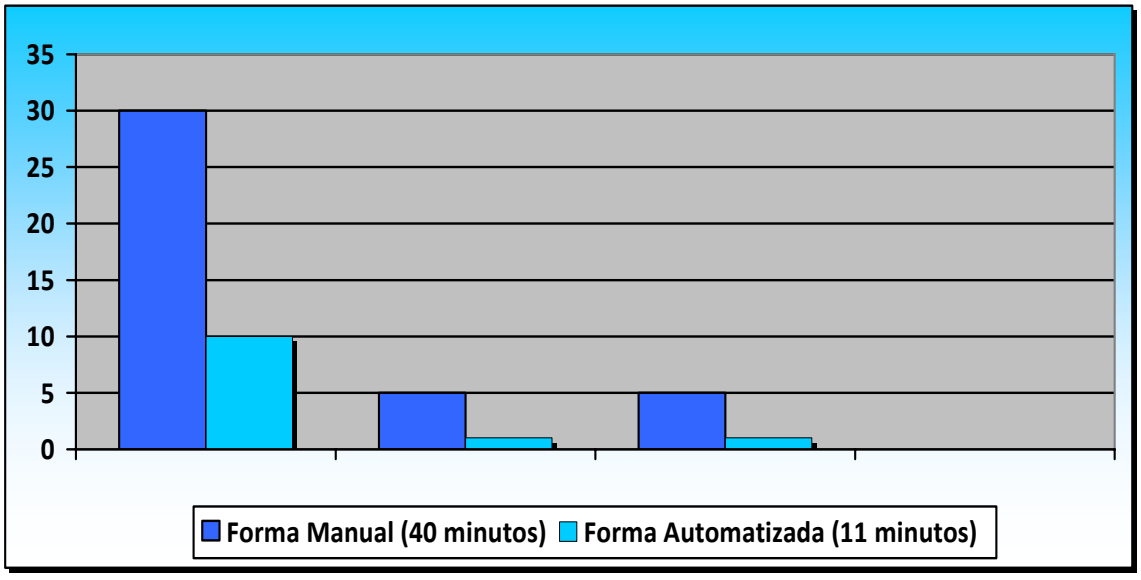
Como se hizo referencia anteriormente, la técnica seleccionada para evaluar la factibilidad del proyecto es la metodología Costo- Efectividad. Para esta técnica es imprescindible definir una variable discreta que haga variar los costos. Teniendo en cuenta que el costo para este proyecto es despreciable, tomaremos como costo el tiempo en minutos empleado para resolver la gestión de evaluación de la calidad de aplicaciones de escritorio y la variable sería la complejidad de las pruebas que se realizan durante este proceso.

Valores de la variable (Solución manual):

- Realizar la evaluación a las aplicaciones de escritorio a través de métricas. 3 variables (30 min.).
- Elaborar en resumen de las evaluaciones. 1 variable (5 min.).
- Imprimir resumen de evaluaciones. 1 variable (5 min.).

Valores de la variable (Solución con el software):

- Cargar cada métrica con los datos necesarios para la realización de la evaluación. 3 variables (10 min.).
- Cargar formulario con los datos necesarios para mostrar un resumen de todas las evaluaciones. 1 variable (0.033 min.).
- Exportar resumen con las evaluaciones realizadas. 1 variable (0.16 min.).



Teniendo en cuenta los resultados reflejados en la gráfica en cuanto al punto de equilibrio queda demostrada la factibilidad del sistema evidenciado por la relación entre la complejidad del problema (cantidad de variables) y el tiempo que demora la solución del mismo de forma manual y automatizada.

4.5 Conclusiones del Capítulo

En este capítulo se hace un estudio profundo del costo real en que se incurrió durante el diseño e implementación de la aplicación, software mediante la metodología Costo Efectividad (Beneficios), se analizaron todos los factores directos, indirectos, externos e intangibles, además se calculó el costo de ejecución del producto software mediante la ficha de costo arrojando como resultado $25,00 C.U.C y $335,5 M.N, así como el tiempo que demora realizar una evaluación de forma manual y de forma automatizada, demostrándose la factibilidad del proyecto.

CONCLUSIONES GENERALES:

Hemos llegado a la meta de este trabajo el cual consta de cuatro capítulos donde recogen ampliamente de una forma u otra todo lo realizado en la investigación y el Sistema que está arrojaron las siguientes conclusiones:

- Se desarrolló el producto final que consistía en la automatización de una aplicación informática para la para la evaluación de la calidad de los aplicaciones de escritorio.
- Se realizó la planeación y diseño de la aplicación, en los cuales se identificaron y especificaron los requisitos funcionales, así como se llevaron a cabo las posteriores fases de codificación y pruebas, que se definen en la metodología de desarrollo utilizada.
- Se llevó a cabo un estudio de las principales metodologías, lenguajes y herramientas que se consideraron factibles para el desarrollo del sistema.
- Se realizaron las pruebas de aceptación definidas por el usuario, lo que arrojó como resultado su aceptación, con lo cual se demostró el cumplimento satisfactorio de las historias de usuarios.

Se efectuó un estudio detallado de la factibilidad del producto final, el cual arrojó los resultados esperados.

Como resultado de los objetivos propuestos en la investigación se logró la implementación del producto software que se esperaba. Con el propósito fundamental de simplificar las demoras que se producen en el tratamiento manual de la información, disminuir el grado de errores y para contribuir a elevar la calidad del desarrollo del trabajo.

RECOMENDACIONES:

Se recomienda lo siguiente:

- Primeramente la utilización del sistema.
- Agregar nuevas funcionalidades acorde a nuevos requisitos que pudiesen surgir por alguna causa o para aumentar el rendimiento del Sistema.
- Realizar un estudio más profundo de este sistema en vista a perfeccionarlo en versiones futuras.
- Extender el software a cualquier entidad de nuestro país.
- Trabajar en la aplicación con navegadores Open-Source ya que traducen los estilos de diseño con mayor calidad.

REFERENCIA BIBLIOGRÁFICA:

ALVA OBESO, María Elena. *"Metodología de Medición y Evaluación de la Usabilidad de Sitios Web Educativos"* (Tesis Doctoral). Universidad de Oviedo. España. 2005.

(BASULTO Jorge Mario). *Sistema de Gestión integral de la empresa Empleadora del NIQUEL "EMPLENI*". MODULO GESTION DE CONTRATOS DE COMPRAS. ISMM. Moa, 2010

(COLUMBIÉ Rafael Wilian). *Evaluación de Calidad de Sitios Web*. ISMM. Moa, 2010.

De Miguel, Enrique. *"Usabilidad el Nuevo Paradigma del Software de Gestión"*.2008.[Consultado2011-02-25]. Disponible en: **http://mejornegocios.com/**

ER/Studio [Consultado 2011-03-30]; Disponible en: **http://www.monografias.com/trabajos14/modelobase/modelobase.shtml**

(HERNÁN RUIZ Marcelo, 2006), *Programación Web Avanzada*, La Habana, Editorial Félix Varela, 2006

Introducción a herramientas CASE y System Architect. Universidad Politécnica de Valencia. . [Consultado: 2011-03-28], 2004. Disponible en: **http://www.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf**

(MONMANY, J). Aplicaciones web. [Consultado: 2011-03-20]. Disponible en: **http://www.webvillage.info**

MySQL [Consultado: 2011-03-15].Disponible en **http://es.wikipedia.org/wiki.**

PÉREZ GARCÍA, Dra. Ana María. *Procedimientos para la elaboración de la ficha de costo de un producto informático.* Facultad MFC UCLV. Villa Clara.

Rational Rose Enterprise. [Consultado 2011-03-30]; Disponible en: **http://www-306.ibm.com/software/awdtools/developer/rose/eneterprise**

(ZALDIVAR Yusimi).*Sistema automatizado de control de acciones correctivas y preventivas de la Empresa Empleadora del Níquel*. ISMM. Moa, 2010.

GLOSARIO DE TERMINOS:

HTML: Hyper Text Markup Language, o simplemente HTML, es un lenguaje de etiquetas de programación muy sencillo que se utiliza para crear los textos y las páginas web. Si se hace la traducción de su nombre del inglés al castellano, sería “Lenguaje de Marca de Hipertextos”, ya que es justamente un lenguaje que se basa en las marcas para crear los hipertextos.

Testing: Las pruebas de software, en inglés *testing* son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador o videojuego. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas.

Open source: Código abierto o código libre. Software que distribuye de forma libre su código fuente, de forma que los desarrolladores pueden hacer variaciones, mejoras o reutilizarlo en otras aplicaciones. También conocido como software libre.

UML: (*Unified Model Language*): Es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de una gran parte de la comunidad informática. Constituye una técnica utilizada para el modelado de diferentes tipos de sistemas, para describir una información bien detallada sobre el funcionamiento de los mismos mediante símbolos estándares.

Principales interfaces del sistema.

Interfaz principal

Listado de software que están en la base de datos

Evaluación de las métricas de idoneidad

Resumen evaluativo de la característica de funcionalidad