“Instituto Superior Minero Metalúrgico”
Dr. Antonio Núñez Jiménez
Facultad Metalurgia - Electromecánica
Moa, Holguín

Trabajo de Diploma

Para optar por el título de de Ingeniería en Informática

SISTEMA DE GESTION Y CONTROL DE LAS PRACTICAS LABORALES

Autor:
Osmany Santana Díaz

Tutor:
Ing. Octavio Martínez Ramírez

Moa, Cuba
Julio, 2010

DECLARACION DE AUTORIA

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

Para que así conste firmamos la presente a los _____ días del mes de _____ del 2010.

Osmany Santana Díaz

_____

Firma autor

Ing. Octavio Martínez Ramírez

_____

Firma tutor

OPINION DEL USUARIO DEL TRABAJO DE DIPLOMA

Yo, Osmany Santana Díaz, estudiante del Instituto Superior Minero Metalúrgico de Moa (ISMMM), declaro que soy el único autor de la presente investigación titulada Sistema de Gestión y Control de las Prácticas Laborales para la carrera de Ingeniería Informática en el ISMM. Se considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

Totalmente

Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____

_____

_____

_____

Como resultado de la implantación de este trabajo se reporta un efecto económico que asciende a <_____> MN y/o <_____> CUC. (Este valor es REAL, no indica lo que se reportará, sino lo que reporta a la entidad. Puede desglosarse por conceptos, tales como: cuánto cuesta un software análogo en el mercado internacional, valor de los materiales que se ahorran por la existencia del software, valor anual del (de los) salario(s) equivalente al tiempo que se ahorra por la existencia del software).

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____

_____ _____

Nombre del representante de la entidad Cargo

_____

Firma Cuño

OPINION DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Sistema de Gestión y Control de las Prácticas Laborales para la carrera de Ingeniería Informática en el ISMM.

Autor: Osmany Santana Díaz.

El tutor del presente Trabajo de Diploma considera que durante su ejecución, el estudiante mostró en alta medida las cualidades que a continuación se declaran:

- Independencia
- Originalidad
- Creatividad
- Laboriosidad
- Responsabilidad

Es necesario destacar además que:

El documento puede ser tomado como referencia o material de consulta de posteriores investigaciones por la calidad profesional y científico-técnica del trabajo realizado.

El producto obtenido tributa directamente a las necesidades de la Universidad que le formó como ingeniero, por lo que este trabajo rinde un humilde tributo a todos aquellos que de forma directa invirtieron en su persona para hacer de quien fuera un egresado de la Enseñanza Media un Ingeniero Informático.

La implantación del software puede hacerse sin dificultad en nuestro centro, por lo que sus beneficios pueden ser verificables fácil y rápidamente.

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: 5-Excelente.

_____

Octavio Alexei Ramírez Martínez
Ing. Informático

Agradecimientos

*A mi **MADRE**: por ser la madre, la amiga, la confidente. Gracias por todas y cada una de las noches de desvelo y preocupación por mí y por mis cosas.*

*A mi **PAPÁ** por estar a mi lado brindándome su apoyo incondicional y comprendiéndome, siempre dispuesto ayudarme*

*A mi **hermana** por quererme y soportarme como soy y por siempre estar a mi lado.*

*A mi **tutor**, Octavio Martínez Ramírez, gracias por la ayuda y orientación que me diste.*

*A mis **compañeros** sobre todos los de quinto año por la unidad que los caracterizó y por los tantos momentos que compartimos.*

*A mis **amigos** por estar siempre ahí brindarme su apoyo incondicional.*

*A todo el **claustro de profesores**, que durante estos cinco años han contribuido a mi formación como profesional.*

*A mi **familia** que se ha preocupado por mí en todo momento y han ayudado de una manera u otra a conquistar este logro.*

*A mis **compañeros de tesis** por la cooperación que los caracterizó.*

*A la **Universidad** por haberme formado como profesional a la altura de nuestros tie mpos.*

A los que se me puedan haber quedado involuntariamente (Discúlpenme, por favor, no ha sido mi intención en lo absoluto).

A los que de una forma u otra han contribuido con la realización de este trabajo.

A todos, gracias.

Dedicatoria

*A mis **padres** que fueron la luz primera, mis primeros maestros. A ellos que son la llama eternizada que vive en mis venas, que vive en mi alma.*

*A una de las mejores personas que he conocido jamás; ejemplo de sacrificio, ternura y abnegación. A mi **Mamá**; por ser la madre ejemplar, sacrificada, dura y sobre todo, tierna y cariñosa que siempre has sido.*

*Al ejemplo de **padre**, de hombre y amigo que es mi papá, créeme que intentar ser como tú es difícil; pero lo intento.*

A mi abuela, tíos, tías, primos, primas, amigos y amigas gracias por su apoyo incondicional.

Resumen

La implantación en la sociedad de las denominadas TIC, produce cambios insospechados en todas las facetas de la vida. Cada vez son más las organizaciones e instituciones que se inclinan por incorporar aplicaciones que gestionen su información, obteniendo así una mayor dinámica en sus procesos.

En el Departamento Informática del Instituto Superior Minero Metalúrgico de Moa (ISMMM) surgió la necesidad de informatizar el proceso de las Prácticas Laborales mediante la creación de una aplicación Web, ya que esta actividad no es realizada de forma automatizada lo que provoca demora durante el proceso de gestión de la información. Con este sistema trataremos de solucionar gran parte de los problemas actuales relacionados con la gestión de las Prácticas Laborales. La aplicación muestra funcionalidades básicas como la captación de datos de los estudiantes, donde se especificaran todos los datos asociados a las Prácticas Laborales que hoy se recogen en formato Excel (Institución que le atiende, nombre, apellidos, año, tema, tutor, tribunal, día y hora de discusión). El sistema debe brindar reportes asociados al problema. Adicionalmente a esto debe contar con un módulo de administración y un banco de problemas diferenciado para los distintos años de la carrera. Debe presentar una propuesta en formato que sirva de guía para el informe de las Prácticas Laborales personalizada según las características y necesidades de cada año.

Abstract

The implantation in the society of named TIC, make change unsuspected in all the facets of life. Each time organizations are further and institutions that applications fall for incorporating themselves that they try to obtain his information, obtaining a bigger dynamics in his processes thus.

At the Department Computer of Moa's Institute Superior Minero Metalúrgico (ISMMM) rose the need to computerize the process of intervening Labor Practices the creation of application Web, since this activity is not accomplished of automated form what he provokes takes time in the process of steps of information. With this system we will try to solve great part of the present-day problems related with the steps of Labor Practices at the department, where no application form does not exist to this end. Application evidences basic functionalities like the student's data entry, where they specify all of the data correlated to the Labor Practices that today one pick up in format Excel to (Institution that attends him, name, surnames, year, theme, tutor, tribunal, day and hour of discussion). System must offer reports associates to the problem. Additionally hereto it must count with a module of administration and a bench of problems told apart for the distinct years of race. He must render a proposal in format that he be good at the personalized report of Labor Practices according to the characteristics and needs out of every year guiding.

Índice

Índice de Figuras y Tablas

Introducción

En el mundo de hoy las mayorías de las actividades y operaciones giran en torno a una computadora. La informática soporta importantes y continuos cambios que repercuten en nuestra sociedad.

En la actualidad, no conciben entidades que no utilicen las Tecnologías de la Información y las Comunicaciones (TIC) ya que éstas aportan soluciones que permiten actuar con rapidez, sacar el máximo rendimiento del personal y tomar decisiones. Las Tecnologías de la Información y las Comunicaciones (TIC), producto de sus avances y como integrantes de esta ciencia, han posibilitado el incremento rápido del desarrollo de la economía y la sociedad. Estos avances vienen dados gracias a la ambición del conocimiento del ser humano, a su naturaleza creadora, y son aplicados en gran medida en nuestra sociedad mediante la producción de software.

Desde Enero de 1959 las universidades cubanas han dado firmes pasos en lograr una universidad para los revolucionarios, cada día se va consolidando su papel activo dentro de la sociedad, entre una de las causas, por permitir a sus ciudadanos cursar estudios superiores, incrementar el número de centros e instituciones a lo largo de todas las provincias, elevar la calidad de la formación de los estudiantes y vincularse activamente a las necesidades más apremiantes del desarrollo económico y social de la nación.

Como ha expresado nuestro Comandante en Jefe, **“Cuba se ha convertido en capital pedagógica del mundo”**, basándose en el profundo desarrollo en el campo de la educación que hemos alcanzado en las últimas décadas y que es reconocido internacionalmente. Este desarrollo abarca todo el sistema educacional desde el preescolar hasta el postgrado.

Las Prácticas Laborales es la forma organizativa específica de la actividad laboral donde se aplica, fundamentalmente, el principio de combinar el estudio con el trabajo relacionando la teoría con la práctica. El principal objetivo de las Prácticas Laborales es contribuir a la adquisición de conocimientos y al desarrollo de habilidades que caracterizan la actividad profesional. Contribuye, además, al desarrollo de la disciplina y la responsabilidad ante el trabajo. Se integra, como un sistema, con las actividades académicas e investigativas y contribuye a la rápida introducción, en el proceso

docente educativo, de los últimos logros científicos y técnicos que se aplican. Las Prácticas Laborales, por ser el elemento de mayor peso dentro del proceso de formación de los estudiantes, se organizarán como una de las asignaturas principales dentro del plan de estudio.

La Prácticas Laborales desempeñan un importante papel en función de la formación de un egresado que sea capaz de enfrentarse a un conjunto de situaciones, que puedan ser modificadas mediante la acción transformadora de él. Esta es parte del complemento práctico de la formación técnica, científica y social de los estudiantes donde el estudiante adquiere una serie de habilidades que redundan en su propio beneficio y en el de la institución donde está insertado. Las Prácticas Laborales adquieren un carácter obligatorio una vez que el estudiante se ubica en cualquier entidad, donde debe cumplir con una serie de requisitos, entre ellos, la asistencia y permanencia en las entidades.

Nuestro país no está ajeno al desarrollo de software y en los últimos años ha dado un salto cuantitativo y cualitativo en busca de la informatización de la sociedad. Las Universidades son las principales artífice en el desarrollo de software que cada día se adentra más en la sociedad, sometidas a procesos de cambios y transformaciones ante los retos que le plantea la actual sociedad del conocimiento, revelándose cada vez más como componente estratégico en la construcción de una sociedad por el rol social que juegan en la producción y transferencia de conocimientos. La importancia del vínculo entre las universidades, las entidades de producción y la sociedad en general, forman parte del discurso de la sociedad de nuestros días.

Nuestra Universidad no está ajena a esta actividad sobre todo en nuestro Departamento de Informática donde los estudiantes y sus conocimientos van encaminados a la producción de software. En este momento no se cuenta con una herramienta informática que permita la realización del proceso de gestión de las Prácticas Laborales lo que entorpece el desarrollo de esta actividad. Por tales motivos se decidió realizar un sistema informático que integre el proceso de gestión de las Prácticas Laborales.

Como consecuencia a lo planteado anteriormente se tiene la siguiente **situación problemática.** Las Prácticas Laborales juegan un papel de gran importancia en las universidades por ser unas de las asignaturas principales en la formación profesional

de los estudiantes pese a su importancia esta actividad no es realizada de forma automatizada lo que provoca serias dificultades durante el proceso de gestión de la información por tal motivo en el Departamento de Informática del ISMM se dio la tarea de desarrollar un sistema informático para controlar los estudiantes en el transcurso de las Prácticas Laborales.

Dada esta situación define como **problema científico** como la necesidad de desarrollar un sistema automatizado que permita la gestión y el control de los estudiantes de Informática en el ISMM durante las Prácticas Laborales. Para darle solución al problema el **objeto de estudio** que se propone es realizar un estudio de los Sistemas de Gestión y Control de las Prácticas Laborales.

Paralelo a objetivo que persigue la presente investigación, el **campo de acción** sería los Sistemas automatizado para la Gestión y Control de las Prácticas Laborales.

De manera que el **objetivo general** de la investigación, sería desarrollar una aplicación Web que permita la gestión y el control de la información de los estudiantes de Informática en el ISMM durante las Prácticas Laborales.

Para guiar nuestra investigación las **ideas a defender** que se plantea es la siguiente:

La implementación de un sistema informático que favorecerá la celeridad y calidad en el control de los estudiantes de Informática en el ISMM durante las Prácticas Laborales.

Los **objetivos específicos** a plantearse son los siguientes:

- Realizar un análisis del estado actual del funcionamiento de las Prácticas Laborales.
- Realizar un estudio acerca de las tecnologías y herramientas actuales necesarias para la posterior implementación del sistema.
- Realizar el análisis, diseño e implementación del sistema.
- Garantizar la seguridad de los datos que se gestionan.

Para lograr un mejor desarrollo de la investigación y darle seguimiento al objetivo trazado se plantearon las siguientes **tareas:**

- Detallar el proceso de gestión de las Prácticas Laborales.

- Fundamentación y análisis bibliográfico que permita la familiarización con las principales tecnologías y herramientas utilizadas actualmente para el desarrollo de aplicaciones informáticas.
- Seguir cada una de las etapas de la ingeniería del software hasta la implementación de un sistema que permita el control de las Prácticas Laborales.
- Estudio de la Factibilidad y Sostenibilidad.
- Prueba y Documentación del Sistema
- Desarrollo del Manual de Usuarios.

Para el cumplimiento de las tares los métodos de investigación empleados fueron **Métodos Teóricos** y **Métodos Empíricos.**

Los siguientes **Métodos Teóricos** sustentan la investigación:

Analítico-Sintético: Permitió integrar y descomponer el conocimiento, pues se hizo una breve investigación y estudio del sistema, determinando los aspectos esenciales y el arribo a conclusiones prácticas y teóricas y así identificar el problema concreto existente.

Inductivo-deductivo: Permitió pasar de lo particular a lo general y viceversa favoreciendo objetivamente el enlace que se establece en la realidad entre lo singular y lo general ya que ambas se complementan mutuamente en el proceso de desarrollo científico.

Modelación: Pues se crean abstracciones que explican la realidad, por ejemplo, todos los modelos y diagramas presentados.

Los **Métodos Empíricos** empleados son:

Entrevistas: Nos permite obtener la información adecuada para adentrarnos más en el problema y en las necesidades del Departamento y así determinar los principales requerimientos del sistema.

Observación: Para ver la funcionalidad que existe en el Departamento de Informática y el comportamiento del sistema.

El documento consta de 5 capítulos:

Capítulo 1: "FUANDAMENTACION TEORICA": Se exponen conceptos y criterios principales necesarios para el entendimiento del problema, además se realiza un estudio del proceso de gestión de las Prácticas Laborales y una caracterización de las metodologías de desarrollo a utilizar, el lenguaje de programación, los sistemas gestores de base de datos las herramientas para el desarrollo del software y las técnicas empleadas durante el proceso de desarrollo.

Capítulo 2: "PLANIFICACION - DEFINICION": En este capítulo se presenta parte de la propuesta solución de la investigación, haciendo uso de la metodología de desarrollo SXP propuesta por la Universidad de Ciencias Informáticas en el que se aborda la fase de planificación y definición en la que se explica toda la dinámica del proyecto basándose en el expediente del proyecto.

Capítulo 3: "DESARROLLO": En este capítulo se abordan los elementos pertenecientes a la fase de desarrollo. Se presenta las tarjetas CRC para apreciar el desarrollo orientado a objetos. También aparecen las tareas de ingeniería para llevar a cabo el desarrollo de las Historias de Usuario.

Capítulo 4: "PRUEBAS": Este capítulo está dedicado a las pruebas que se les realizan al software las pruebas de aceptación del cliente. Estas pruebas fueron llevadas a cabo antes de cada entrega que se realizó durante todo el desarrollo del proyecto.

Capítulo 5: "ESTUDIO DE FACTIBILIDAD": En este capítulo se realiza un estudio de factibilidad del proyecto, se utilizará la Metodología Costo Efectividad (Beneficio), la cual plantea la conveniencia de la ejecución del proyecto.

CAPITULO 1: FUNDAMENTACION TEORICA

1.1 Introducción

En este capítulo se realiza un proceso investigativo de los aspectos teóricos necesarios para la elaboración y concepción del Trabajo de Diploma. Se describen los principales conceptos asociados al problema y que son necesarios para un mejor entendimiento y darle solución al mismo. Además se realiza un estudio del proceso de gestión de las Prácticas Laborales en Cuba y la estructura que debe tener con sus funciones asociadas; estableciendo la estrategia a llevar a cabo y los beneficios esperados. Se hace una caracterización de cada tipo de herramientas y del lenguaje de programación, así como las metodologías de desarrollo escogida para el desarrollo del software.

1.2 Funcionamiento de las Prácticas Laborales en Cuba

Las Prácticas Laborales dentro de sus propósitos tienen ofrecer ideas que permitan al estudiante comprender el proceso de enseñanza aprendizaje, de manera que logren vincular la práctica con el estudio, donde el estudiante sea capaz de ser creador en las diversas situaciones donde realiza su actividad. Para el éxito de esta actividad, por una parte, debemos lograr que los estudiantes se involucren en actividades auténticas y, por la otra, construir nuevo conocimiento sobre la base que ya posee y profundizar en el aprendizaje mediante el hacer parte de un equipo.

Para desarrollar este propósito existe un proceso de formalización de convenio entre la parte docente y la parte laboral, que deben cumplir una serie de requisitos por ambas partes, donde los estudiantes puedan permanecer ocho horas diarias desarrollando sus habilidades teórico-prácticas, durante el curso académico, cumpliendo con diferentes obligaciones y responsabilidades.

Las Prácticas Laborales adquieren un carácter obligatorio una vez que el estudiante se ubica en cualquier entidad, donde debe cumplir con una serie de requisitos entre ellos, la asistencia y permanencia en las entidades. Así como las entidades deben cumplir una serie de obligaciones **(Matos, 2004)**.

1.2.1 Estructura de las Prácticas Laborales (PL)

- Coordinadores de las Prácticas Laborales
- Tutores
- Técnicos o personal especializado

Funciones:

- Coordinadores
  - ✓ Visitas a las organizaciones para establecer convenios.
  - ✓ Coordinar el trabajo que se desarrollará con los estudiantes.
  - ✓ Chequear la asistencia de los estudiantes.
  - ✓ Coordinar con los profesores que imparten las asignaturas que darán salida a las Prácticas Laborales.
- Tutores
  - ✓ Velar por la incorporación y asistencia de los estudiantes a las Prácticas Laborales.
  - ✓ Visita a las entidades donde están insertados los estudiantes.
  - ✓ Preparación metodológica sobre asesoramiento de las asignaturas que se imparten.
- Técnicos
  - ✓ Elabora el programa de trabajo a desarrollar el estudiante.
  - ✓ Supervisa, dirige y evalúa el proceso de las Prácticas Laborales.
  - ✓ Asesoramiento a los estudiantes durante el desarrollo de las Prácticas Laborales.

1.3 Estrategia de las PL

La propuesta de esta estrategia está encaminada a la formación integral del estudiante donde desde su plano personal se desarrollen rasgos de su personalidad como es la motivación, el interés y la responsabilidad.

La **misión** de esta actividad es alcanzar una interrelación entre lo académico, lo investigativo y lo laboral que contribuya a establecer un vínculo estrecho entre los estudiantes y las problemáticas de las entidades de información.

La **visión** consolida un buen desarrollo del proceso enseñanza aprendizaje que permita alcanzar una formación profesional con la calidad que así lo requiere.
Los **objetivos** de las Prácticas Laborales alcanzados:

- ✓ Dotar a los estudiantes de los conocimientos técnicos básicos de la especialidad, incluyendo la práctica y el desarrollo de las nuevas tecnologías de la información.
- ✓ Alcanzar habilidades para localizar bibliografía e información.
- ✓ Emplear las normas y procedimientos para la descripción de documentos e información en distintos soportes.
- ✓ Conocer los diferentes procesos de información: Búsqueda, clasificación y recuperación **(Matos, 2004).**

1.4 Beneficios y resultados esperados

- ➢ Permite que los estudiantes piensen y actúen por sí mismo, fuera de los parámetros educativos.
- ➢ Permite la formación integral del futuro profesional.
- ➢ Suministra a los docentes una herramienta para que los estudiantes trabajen en equipos.
- ➢ Estimula el crecimiento emocional, intelectual y personal en el estudiante.
- ➢ Ofrece a los estudiantes la oportunidad de aprender diferentes técnicas para la solución de problemas **(Matos 2004).**

1.5 Tendencias y tecnologías actuales

Informatización equivale a cambios de mentalidad y de hábitos para asimilar con provecho e incluso con placer el cambio, la sustitución de lo viejo por lo nuevo; para que la creatividad se multiplique haciendo el mayor y mejor uso de las muy diversas posibilidades que ofrece “la máquina”, la tecnología. La asimilación de las nuevas tecnologías informáticas, como base para el logro de la sociedad de la información, se convierte en tarea de primer orden. No se trata de estar a la moda o con el último grito de la tecnología, sino de una necesidad inminente de incorporar, en este caso, el

paradigma Cliente - Servidor al mayor número de procesos posibles; es evidente que se trata del próximo escalón a subir.

1.5.1 Tecnología Cliente - Servidor

La arquitectura cliente servidor se basa en la distribución de tareas que desde los años 70 se estableció entre un banco central y sus sucursales El cliente (un usuario de PC) solicita un servicio (como por ejemplo imprimir) que un servidor le proporciona (un procesador conectado a la LAN). Este enfoque común de la estructura de los sistemas informáticos se traduce en una separación de las funciones que anteriormente formaban un todo. Los detalles de la realización van desde los planteamientos sencillos hasta la posibilidad real de manejar todos los ordenadores de modo uniforme. Para establecer un canal de comunicación entre dos programas que se ejecutan en dos computadoras distintas o incluso en una misma computadora, un programa debe iniciar la conexión y el otro aceptarla. En esto se basa el modelo Cliente-Servidor. El servidor indica al sistema operativo que está en condiciones de aceptar solicitudes de conexión al mismo tiempo que espera por ellas. La mayoría de los servidores atienden solicitudes de varios clientes. Cuando un cliente o computadora remota necesita enviar o descargar información hacia o desde el servidor, éste establece una conexión por donde circula la información y luego la cierra. El cliente interactúa directamente con el usuario, procesa sus solicitudes y muestra los resultados. Así por ejemplo la World Wide Web utiliza un modelo cliente-servidor, los visualizadores o navegadores (clientes) tienen la función de manipular las solicitudes de documentos hechas por los usuarios. Ellos determinan con qué computadora se realizará la conexión, descargan los documentos solicitados y se lo muestran al usuario en su pantalla. Los servidores Web son los responsables de la otra parte de la conexión, esperan una solicitud proveniente de un usuario y una vez hecha ésta, transmiten el documento al visualizador. La conexión entre el visualizador y el servidor dura exactamente el tiempo justo para que el primero envíe una solicitud y el segundo una respuesta. Esto quiere decir que no existe ninguna conexión entre estos elementos una vez que el usuario ya está leyendo el documento solicitado. Ese modo de trabajo los hace extremadamente eficientes y descongestiona el tráfico en la red **(Lamas 2009).**

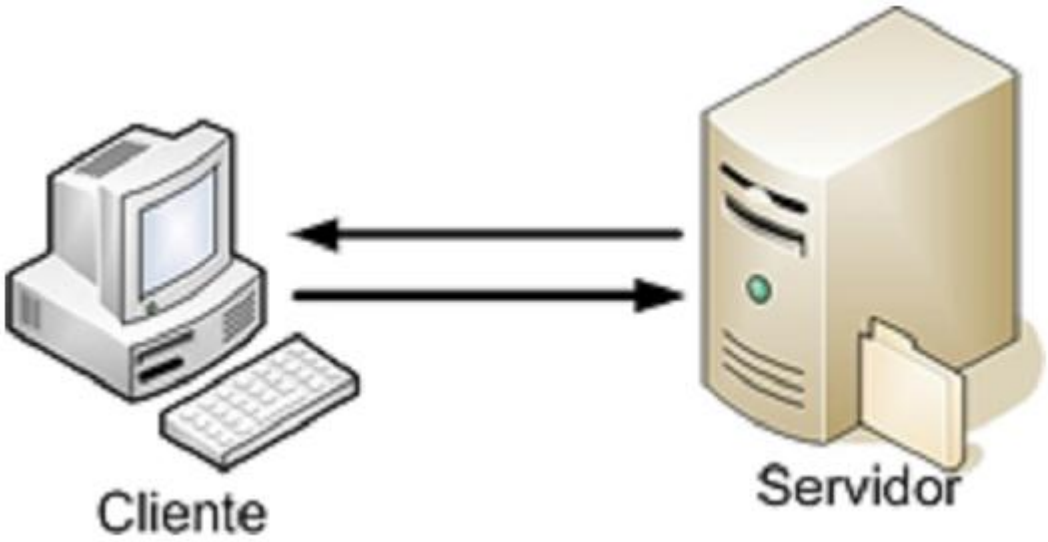

Figura 1: Arquitectura Cliente – Servidor

1.5.2 Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina **(Díaz 2009).**

LENGUAJES DEL LADO DEL SERVIDOR

PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas Web dinámicas. Es usado principalmente en interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando distintas bibliotecas. El Pre-procesador de hipertextos (*Hypertext Pre-processor*, PHP por sus siglas en inglés) inicialmente se llamó PHP *Tools*, siendo publicada bajo licencia de software libre. Sus principales características son:

- Es un lenguaje multiplataforma.
- Soporte para una gran cantidad de bases de datos
- Integración con varias bibliotecas externas, permite generar documentos en PDF.

- Ofrece una solución simple y universal para las paginaciones dinámicas Web de fácil programación.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Producto de código abierto
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones

ASP

ASP (*Active Server Pages*) es la tecnología desarrollada por Microsoft para la creación de páginas dinámicas del servidor. ASP se escribe en la misma página Web, utilizando el lenguaje Visual Basic Script o Jscript (Javascript de Microsoft). Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la página ASP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores. El tipo de servidores que emplean este lenguaje son, evidentemente, todos aquellos que funcionan con sistema Windows NT, aunque también se puede utilizar en un PC con *Windows* 98 si instalamos un servidor denominado Personal Web Server. Incluso en sistemas Linux podemos utilizar las ASP si instalamos un componente denominado *Chilisoft*, aunque lo ideal es trabajar sobre el servidor Web para el que está pensado: IIS.

Con las ASP podemos realizar muchos tipos de aplicaciones distintas. Nos permite acceso a bases de datos, al sistema de archivos del servidor y en general a todos los recursos que tenga el propio servidor. También existe la posibilidad de comprar componentes ActiveX fabricados por distintas empresas de desarrollo de software que sirven para realizar múltiples usos, como el envío de correo y generar gráficas dinámicamente **(Lamas 2009).**

PERL

Es un lenguaje de programación muy utilizado para construir aplicaciones CGI para el Web. Perl es un acrónimo de *Practical Extracting and Reporting Language*, que viene a indicar que se trata de un lenguaje de programación muy práctico para extraer información de archivos de texto y generar informes a partir del contendido de los ficheros. Es un lenguaje libre de uso, eso quiere decir que es gratuito. Antes estaba muy asociado a la plataforma Unix, pero en la actualidad está disponible en otros sistemas operativos como Windows. Perl es un lenguaje de programación interpretado, al igual que muchos otros lenguajes de Internet como JavaScript o ASP.

JSP

JSP es un acrónimo de *Java Server Pages*, que en castellano vendría a decir algo como Páginas de Servidor Java. Es pues, una tecnología orientada a crear páginas Web con programación en Java. Con JSP podemos crear aplicaciones Web que se ejecuten en variados servidores Web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Por tanto, las JSP podremos escribirlas con nuestro editor HTML/XML habitual.

LENGUAJES DEL LADO DEL CLIENTE

JavaScript

Javascript es un lenguaje de programación interpretado que permite a los desarrolladores crear acciones en sus páginas Web. Es utilizado para crear pequeños programas que luego son insertados en una página Web y en programas más grandes, orientados a objetos pero mucho más complejos.

HTML

El Lenguaje de Marcas de Hipertexto **(***HyperText Markup Language*, HTML por sus siglas en inglés) es el lenguaje de marcado predominante para la construcción de páginas Web. Usado para describir la estructura y el contenido en forma de texto, así

como para complementar el texto con objetos tales como imágenes. PHTML es una extensión para un tipo de páginas Web que llevan código PHP y HTML para ser generadas. Cuando una página está escrita en PHP podemos encontrarla con varios tipos de extensiones como por ejemplo .php, .php4, .php5.

1.5.3 Justificación del lenguaje de programación a utilizar

Luego de realizar un estudio de los diferentes lenguajes de programación planteados anteriormente se decide optar **PHP** del lado del servidor y **HTML** del lado del cliente.

¿Por qué utilizar PHP?

Es el acrónimo de *Hypertext Preprocessor*. Es software libre, lo que implica menores costes y servidores más baratos que otras alternativas, a la vez que el tiempo entre el hallazgo de un fallo y su resolución es más corto. Es muy rápido. Su sintaxis está inspirada en C. Su librería estándar es realmente amplia, lo que permite reducir los llamados 'costes ocultos', uno de los principales defectos de ASP.

PHP se puede considerar multiplataforma. Funciona en toda máquina que sea capaz de compilar su código, entre ellas diversos sistemas operativos para PC y diversos Unix. El código escrito en PHP en cualquier plataforma funciona exactamente igual en cualquier otra. El acceso a las bases de datos de PHP es muy heterogéneo, pues dispone de un juego de funciones distinto por cada gestor. PHP es suficientemente versátil y potente como para hacer tantas aplicaciones grandes que necesiten acceder a recursos a bajo nivel del sistema como pequeños *scripts* que envíen por correo electrónico un formulario rellenado por el usuario.

1.5.4 Sistemas Gestores de Base de Datos (SGBD)

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y

DataBase Management System. Actualmente existen muchos sistemas gestores de bases de datos, entre ellos, están: MySQL, PostgreSQL y Microsoft SQL Server.

PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, dentro de sus características destacan las siguientes:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo cambios. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases de datos, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos

PostgreSQL suministra nativamente soporte para:

- ✓ Números de precisión arbitraria.
- ✓ Texto de largo ilimitado.
- ✓ Figuras geométricas (con una variedad de funciones asociadas)
- ✓ Direcciones IP (IPv4 e IPv6).
- ✓ Arrays.

Como características adicionales tiene:

Claves ajenas también denominadas Llaves ajenas o **Claves Foráneas**

Disparadores: Un disparador se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

Todos los disparadores se definen por seis características:

- ✓ El nombre del disparador.
- ✓ El momento en que el disparador debe arrancar.
- ✓ La tabla donde el disparador se activara.
- ✓ La frecuencia de la ejecución.

- ✓ La función que podría ser llamada.

PostgreSQL posee integración con un gran número de lenguajes tales como PHP, Java, C, C++ **(Díaz 2009).**

MySQL

Es la base de datos open source más popular, esta se encuentra entre las mejores del mundo. Su continuo desarrollo y su creciente popularidad están haciendo de MySQL un competidor cada vez más directo de gigantes en la materia de las bases de datos como Oracle. MySQL es un sistema de administración de bases de datos (*Database Management System*, DBMS) para bases de datos relacionales.

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario. El modelo relacional se caracteriza a muy grandes rasgos por disponer que toda la información debe estar contenida en tablas, y las relaciones entre datos deben ser representadas explícitamente en esos mismos datos. Es un software de código abierto esto quiere decir que es accesible para cualquiera, para usarlo o modificarlo. MySQL es muy rápido, confiable, robusto y fácil de usar tanto para volúmenes de datos grandes como pequeños. Además tiene un conjunto muy práctico de características desarrolladas en cooperación muy cercana con los usuarios. Sin embargo bajo constante desarrollo, MySQL hoy en día ofrece un rico y muy útil conjunto de funciones. La conectividad, velocidad y seguridad hace de MySQL altamente conveniente para acceder a bases de datos **(Lamas 2009).**

SQL Server

Microsoft SQL Server, propietario de Microsoft, pertenece a la familia de los sistemas de administración de base de datos, operando en una arquitectura cliente/servidor de gran rendimiento. Su desarrollo fue orientado para hacer posible manejar grandes volúmenes de información, y un elevado número de transacciones. SQL Server es una aplicación completa que realiza toda la gestión relacionada con los datos. El servidor sólo tiene que enviarle una cadena de caracteres (la sentencia SQL) y esperar a que le devuelvan los datos. SQL Server permite la creación de procedimientos almacenados,

los cuales consisten en instrucciones SQL que se almacenan dentro de una base de datos de SQL Server, realizados en lenguaje SQL, se trata de procedimientos que se guardan semicompilados en el servidor y que pueden ser invocados desde el cliente. SQL Server puede manejar perfectamente bases de datos de *TeraBytes* con millones de registros y funciona sin problemas con miles de conexiones simultáneas a los datos, sólo depende de la potencia del hardware del equipo en el que esté instalado y solamente corre sobre Windows NT- 2000 Server.

1.5.5 Justificación del SGBD utilizar

Luego de realizar un estudio de las diferentes características de los gestores de base de datos expuesto anteriormente se decide optar por **MySQL** por las siguientes razones:

MySQL es una de las bases de datos más populares desarrolladas bajo la filosofía de código abierto. Su principal objetivo de diseño fue la velocidad. Otra característica importante es que consume muy pocos recursos, tanto de CPU como de memoria. Tiene licencia GPL a partir de la versión 3.23.19. Algunas de sus ventajas respecto a otros gestores de base de datos son:

Mayor rendimiento. Mayor velocidad tanto al conectarse con el servidor como al servir *selects* y demás. Aunque se cuelgue, no suele perder información ni corrompe los datos. Mejor integración con PHP. No tiene límites en el tamaño de los registros. Mejor control y acceso, en el sentido de qué usuarios tienen acceso a qué tablas y con qué permisos **(Lamas 2009).**

Principales Características

- El principal objetivo de MySQL es velocidad y robustez.
- Escrito en C y C++.
- Clientes C, C++, JAVA, Perl, TCL.
- Multiproceso, es decir puede usar varias CPU si éstas están disponibles.
- Puede trabajar en distintas plataformas y S.O. distintos.
- Sistema de contraseñas y privilegios muy flexibles y seguros.
- Utilidad (*Isamchk*) para chequear, optimizar y reparar tablas.

- Todos los datos están grabados en formato ISO8859_1.
- Los clientes usan TCP o UNIX *Socket* para conectarse al servidor.
- El servidor soporta mensajes de error en distintas lenguas.
- Todos los comandos tienen -*help* o - Para las ayudas.

1.5.6 Tecnología RIA (*Rich Internet Aplication*)

AJAX

AJAX Acrónimo de ***A**synchronous **J**avaScript **A**nd **X**ML* (JavaScript y XML asíncronos, donde XML es un acrónimo de e**X**tensible **M**arkup **L**anguage), es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

Algunas tecnologías por la que está compuesto AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

Sus ***características*** principales son:

- Mayor rapidez e interactividad, al estilo aplicaciones de escritorio.
- Reduce significativamente la carga de información continua del servidor, actualizando solamente porciones de la página.

1.5.7 Entorno de trabajo

Zend Studio o Zend Development Environment

Es un completo entorno de desarrollo integrado para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac

OS y GNU/Linux. Junto con su contraparte *Zend Platform*, son la propuesta de *Zend Technologies* para el desarrollo de aplicaciones Web utilizando PHP, actuando *Zend Studio* como la parte cliente y *Zend Platform* como la parte servidora. Se trata en ambos casos de software comercial, lo cual contrasta con el hecho de que PHP es software libre.

Características:

- No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- Soporte para PHP 4 y PHP 5.
- Resaltado de sintaxis, auto completado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.
- Plegado de código (comentarios, bloques de phpDoc, cuerpo de funciones y métodos e implementación de clases).
- Inserción automática de paréntesis y corchetes de cierre.
- Sangrado automático y otras ayudas de formato de código.
- Emparejamiento (*matching*) de paréntesis y corchetes (si se sitúa el cursor sobre un paréntesis (corchete) de apertura (cierre), *Zend Studio* localiza el correspondiente paréntesis (corchete) de cierre (apertura)).
- Detección de errores de sintaxis en tiempo real.
- Funciones de depuración: Botón de ejecución y traza, marcadores, puntos de parada (*breakpoints*), seguimiento de variables y mensajes de error del intérprete de PHP. Permite también la depuración en servidores remotos (requiere *Zend Platform*).
- Instalación de barras de herramientas para Internet Explorer y Mozilla Firefox.
- Manual de PHP integrado.

Zend Studio fue diseñado para usarse con el lenguaje PHP; sin embargo ofrece soporte básico para otros lenguajes Web, como HTML, Javascript y XML.

Case Studio

Es una herramienta profesional con la que podrás diseñar tus propias bases de datos, facilitándote herramientas para la creación de diagramas de relación, modelado de datos y gestión de estructuras (Es compatible con ambos SGBD, MySQL y PostgreSQL).

Nusphere o PHPEd

Es un editor para programadores con soporte para múltiples formatos, similar a otras aplicaciones como PHP Coder. PHPEd facilita tu trabajo de programación con numerosas características de gran utilidad entre las que destacan:

- Completo sistema de ayuda
- Plantillas de documento y de fragmentos de código frecuentes
- Código de colores para comandos en PHP, Perl, Javascript, SQL, HTML y más.

Además, esta herramienta incluye un cliente de FTP y un servidor Web integrados, totalmente configurables según tus necesidades de trabajo.

Adobe Dreamweaver

Es una aplicación en forma de estudio (basada en la forma de estudio de *Adobe Flash*) enfocada en la construcción y edición de sitios y aplicaciones Web basadas en estándares. Creado inicialmente por Macromedia (actualmente producido por *Adobe Systems*). Es el programa de este tipo más utilizado en el sector del diseño y la programación Web, por sus funcionalidades, su integración con otras herramientas como *Adobe Flash* y, recientemente, por su soporte de los estándares del *World Wide Web Consortium*.

Servidor Apache

Apache, es un servidor de protocolo para la transferencia de hipertextos (*Hypertext Transfer Protocol*, HTTP por sus siglas en inglés) de software libre para plataformas Unix, Windows, y Macintosh, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Las características que lo definen son las siguientes:

- Multiplataforma
- Es un servidor de Web conforme al protocolo HTTP/1.1
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.

- Basado en hebras en la versión 2.0.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla de forma abierta.

1.6 Metodologías ágiles

Actualmente para desarrollar un proyecto con éxito, debe estar regido por metodología de desarrollo, la cual puede seguir uno o varios modelos de ciclo de vida, o sea, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto. Una metodología de desarrollo de software es un conjunto de técnicas, herramientas, procedimientos y soporte documental que permite a los desarrolladores definir los elementos necesarios para la construcción de un nuevo producto de software. Es la que durante el proceso de desarrollo del software define "quién está haciendo qué, cuándo y cómo para alcanzar un determinado objetivo." Mediante la metodología de desarrollo de software se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.
Las Metodologías Ágiles se basan en los siguientes principios:

- Realizar entregas cortas en el tiempo y continuas.
- Dar la bienvenida a los cambios.
- Entregas periódicas y frecuentes que funcionen.
- La comunicación directa es el método más eficiente y efectivo para comunicar información. Intenta evitar el teléfono, correos electrónicos, fax, etc.
- La medida principal de progreso es el software que funciona.
- Buen diseño y calidad técnica.
- La simplicidad es algo básico.

Actualmente existen varias metodologías de desarrollo de software, las cuales se deben estudiar con detenimiento para definir cuál es la más adecuada a usar en el software a construir.

Entre las más conocidas están:

- **Metodología Crystal:** Conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación.
- **Dynamic Systems Development Method (DSDM):** DSDM empezó en Gran Bretaña en 1994 como un consorcio de compañías del Reino Unido que querían construir sobre RAD [N. del T. Desarrollo Rápido de Aplicaciones] y desarrollo iterativo. Habiendo empezado con 17 fundadores ahora tiene más de mil miembros y ha crecido fuera de sus raíces británicas. Siendo desarrollado por un consorcio, tiene un sabor diferente a muchos de los otros métodos ágiles. Tiene una organización de tiempo completo que lo apoya con manuales, cursos de entrenamiento, programas de certificación y demás. También lleva una etiqueta de precio, lo qué ha limitado mi investigación sobre su metodología. Sin embargo Jennifer Stapleton ha escrito un libro que da una apreciación global de la metodología.
- **Adaptive Software Development (ASD):** Promovido por Jim Highsmith. Sus principales características son:

✓ Iterativo.

✓ Orientado a los componentes software más que a las tareas y tolerante a los cambios.

✓ El ciclo de vida que propone tiene tres fases esenciales:

  ✓ Especulación.

  ✓ Colaboración y aprendizaje.

En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

- **Feature -Driven Development (FDD):** Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.
- **Lean Development (LD):** Definida por Bob Charretes en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. Los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios.
- **SCRUM:** Esta metodología se basa en una filosofía del desarrollo ágil, creado por Hirotaka Takeuchi e Ikujiro Nonaka ahí por el año 1986, para desarrollo de software. SCRUM aunque puede ser usado para otro tipo de proyectos y tiene demostrada efectividad en otras áreas, aunque generalmente es funcional solo para desarrollos de software porque para eso fue diseñado. La idea es desarrollar aplicaciones mucho más rápido y eficazmente.
- **XP:** Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

¿Por qué aplicar Scrum y XP?

Antes de plantearse un cambio tan importante como el que puede suponer usar una metodología ágil es necesario mirar a tu alrededor y tener claro donde estas y hacia donde quieres ir. Si estas pensando en meterte en el largo (y posiblemente tortuoso) camino de cambiar tu metodología de trabajo lo primero es identificar que problemas tienes y evaluar si una metodología ágil te va a ayudar a resolver estos problemas; si esta metodología encaja en tu organización y es posible aplicarla, si el equipo de

desarrollo que las va a aplicar cuenta con los conocimientos y los medios necesarios, vamos que lanzarse a lo loco a usar esto o aquello simplemente porque está de moda es como poco “arriesgado” y podemos terminar consiguiendo que el remedio sea peor que la enfermedad.

Para el desarrollo de este trabajo se propone el uso de dos de las muchas metodologías ágiles enunciadas anteriormente, **Scrum** para la parte de planificación, y **XP** para la parte de desarrollo.

1.6.1 Scrum

Desarrollada por *Ken Schwaber*, *Jeff Sutherland* y *Mike Beedle*. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos diez años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de treinta días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de quince minutos del equipo de desarrollo para coordinación e integración.

¿Qué significa SCRUM?

Pues la palabra se usa en rugby y prácticamente significa melé. Si ustedes han visto un juego de rugby cuando va a iniciar una jugada se aglomeran los equipos empujándose con tal de hacerse de la pelota para recuperarla y pasársela a otro que espera atrás. Pues eso es SCRUM no son siglas ni nada por el estilo.

Características de Scrum

Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el *ScrumMaster*, que mantiene los procesos y trabaja de forma similar al director de proyecto, el

ProductOwner, que representa a los *stakeholders* (clientes externos o internos), y el *Team* que incluye a los desarrolladores.

El conjunto de características que forma parte de cada sprint viene del *Product Backlog*, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del *Product Backlog* que forman parte del sprint se determinan durante la reunión de *Sprint Planning*. Durante esta reunión, el *Product Owner* identifica los elementos del *Product Backlog* que quiere ver completados. Entonces, se determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente *sprint*. Durante el *sprint*, nadie puede cambiar el *Sprint Backlog*, lo que significa que los requisitos están congelados durante el *sprint*.

Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan (a menudo llamado *requirements churn*), y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por lo tanto, Scrum adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes.

1.6.2 XP (Extreme Programming)

XP es una de las metodologías ágiles de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo y corto equipo. Centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software. XP se basa en realimentación continua entre el cliente y el desarrollador, comunicación fluida, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. *Kent Beck*, el padre de XP, describe la filosofía de XP sin cubrir los detalles técnicos y de implantación de las prácticas **(Jeffries, Ferrer 2001).**

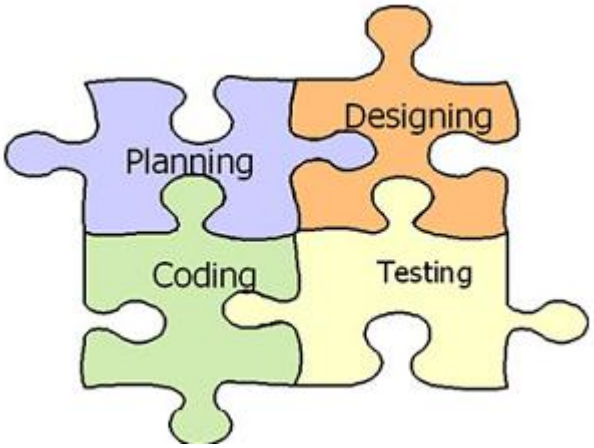

Figura 2: Extreme Programming

1.6.3 Valores de la metodología XP

Simplicidad

XP propone el principio de hacer la cosa más simple que pueda funcionar, en relación al proceso y la codificación. Es mejor hacer hoy algo simple, que hacerlo complicado y probablemente nunca usarlo mañana.

El programador tiene que estar en constante comunicación con el cliente para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.

Comunicación

El Extreme Programming se nutre del ancho de banda más grande que se puede obtener cuando existe algún tipo de comunicación: la comunicación directa entre personas. Es muy importante entender cuáles son las ventajas de este medio. Cuando dos (o más) personas se comunican directamente pueden no solo consumir las palabras formuladas por la otra persona, sino que también aprecian los gestos, miradas, etc. que hace su compañero. Sin embargo, en una conversación mediante el correo electrónico, hay muchos factores que hacen de esta una comunicación, por así decirlo, mucho menos efectiva. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.

Retroalimentación (*Feedback*)

Retroalimentación concreta y frecuente del cliente, del equipo y de los usuarios finales da una mayor oportunidad de dirigir el esfuerzo eficientemente. Mediante la retroalimentación se ofrece al cliente la posibilidad de conseguir un sistema adecuado a sus necesidades. Se le va mostrando el proyecto a tiempo para sugerir cambios y poder retroceder a una fase anterior para rediseñarlo a su gusto.

Coraje

El coraje es un valor muy importante dentro de la programación extrema. Un miembro de un equipo de desarrollo extremo debe de tener el coraje de exponer sus dudas, miedos, experiencias sin "embellecer" éstas de ninguna de las maneras. Esto es muy importante ya que un equipo de desarrollo extremo se basa en la confianza para con sus miembros. Faltar a esta confianza es una falta más que grave. El coraje existe en el contexto de los otros tres valores.

Cada uno de ellos se apoya en los demás:

- Se requiere valor para comunicarse con los demás cuando eso podría exponer la propia ignorancia.
- Se requiere valor para mantener el sistema simple, dejando para mañana las decisiones de mañana.
- Se requiere coraje para confiar en que la retroalimentación durante el camino es mejor que tratar de adivinar todo con anticipación.
- Y, sin un sistema simple, comunicación constante y retroalimentación, es difícil mantenerse valiente.

1.6.4 Prácticas de la metodología XP

Planificación incremental

La Programación Extrema asume que la planificación nunca será perfecta, y que variará en función de cómo varíen las necesidades del negocio. Por tanto, el valor real reside en obtener rápidamente un plan inicial, y contar con mecanismos de *feedback* que permitan conocer con precisión dónde estamos. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar. El objetivo de la XP es generar

versiones de la aplicación tan pequeñas como sea posible, pero que proporcionen un valor adicional claro, desde el punto de vista del negocio. A estas versiones se las denomina *releases*. Una *release* cuenta con un cierto número de historias. La historia es la unidad de funcionalidad en un proyecto XP, y corresponde a la mínima funcionalidad posible que tiene valor desde el punto de vista del negocio. Durante cada iteración se cierran varias historias, lo que hace que toda iteración añada un valor tangible para el cliente. Es fundamental en toda esta planificación la presencia de un representante del cliente, que forma parte del equipo y que decide cuáles son las historias más valiosas. Estas historias son las que se desarrollarán en la iteración actual.

Como se puede ver, y como siempre ocurre con la Programación Extrema, el enfoque utilizado para llevar a cabo la planificación es eminentemente pragmático. Gran parte de la eficacia de este modelo de planificación deriva de una división clara de responsabilidades, que tiene en cuenta las necesidades del negocio en todo momento. Dentro de esta división, el representante del cliente tiene las siguientes responsabilidades:

- ✓ Decidir qué se implementa en cada release o iteración.
- ✓ Fijar las fechas de fin de la *release*, recortando unas características o añadiendo otras.
- ✓ Priorizar el orden de implementación, en función del valor de negocio.

Testing

La ejecución automatizada de tests es un elemento clave de la XP. Existen tanto tests internos (o tests de unidad), para garantizar que el mismo es correcto, como tests de aceptación, para garantizar que el código hace lo que debe hacer. El cliente es el responsable de definir los tests de aceptación, no necesariamente de implementarlos. Él es la persona mejor cualificada para decidir cuál es la funcionalidad más valiosa.

El hecho de que los tests sean automatizados es el único modo de garantizar que todo funciona: desde el punto de vista de la XP, si no hay tests, las cosas sólo funcionan en apariencia. Aún más, si un test no está automatizado, no se le puede considerar como tal. El objetivo de los tests no es corregir errores, sino prevenirlos. Por ejemplo, los tests siempre se escriben antes que el código a testear, no después: esto aporta un

gran valor adicional, pues fuerza a los desarrolladores a pensar cómo se va a usar el código que escriben, poniéndolos en la posición de consumidores del software.

Elaborar los tests exige pensar por adelantado cuáles son los problemas más graves que se pueden presentar, y cuáles son los puntos dudosos. Esto evita muchos problemas y dudas, en lugar de dejar que aparezcan “sobre la marcha”. Un efecto lateral importante de los tests es que dan una gran seguridad a los desarrolladores: es posible llegar a hacer cambios más o menos importantes sin miedo a problemas inesperados, dado que proporcionan una red de seguridad. La existencia de tests hace el código muy maleable.

Refactorización continúa

Es el proceso de modificar el código de un sistema de software de modo que no se altere su comportamiento externo pero se mejore su estructura interna. Se refactoriza el sistema eliminando duplicación, mejorando la comunicación y agregando flexibilidad sin cambiar la funcionalidad. El proceso consiste en una serie de pequeñas transformaciones que modifican la estructura interna preservando su conducta aparente. La práctica también se conoce como Mejora Continua de Código o Refactorización implacable. Se lo ha parafraseado diciendo: “Si funciona bien, arréglelo de todos modos”.

Diseño simple

Otra práctica fundamental de la Programación Extrema es utilizar diseños tan simples como sea posible. El principio es “utilizar el diseño más sencillo que consiga que todo funcione”. Se evita diseñar características extra porque a la hora de la verdad la experiencia indica que raramente se puede anticipar qué necesidades se convertirán en reales y cuáles no.

La XP nos pide que no vivamos bajo la ilusión de que un diseño puede resolver todas o gran parte de las situaciones futuras: lo que parece necesario cambia con frecuencia, es difícil acertar a priori. Es obvio que, si no vamos a anticipar futuras necesidades, debemos poder modificar el diseño si alguna de estas se materializa. La XP soporta estas modificaciones gracias a los tests automatizados. Estos permiten hacer cambios importantes gracias a la red de protección que proporcionan. La refactorización, que

hace que el código existente sea claro y sencillo, también ayuda a hacer factibles las modificaciones.

La XP define un “diseño tan simple como sea posible” como aquél que:

- ✓ Pasa todos los tests.
- ✓ No contiene código duplicado.
- ✓ Deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código.
- ✓ Contiene el menor número posible de clases y métodos.

Propiedad colectiva del código

La XP aboga por la propiedad colectiva del código. En otras palabras, todo el mundo tiene autoridad para hacer cambios a cualquier código, y es responsable de ellos. Esto permite no tener que estar esperando a otros cuando todo lo que hace falta es algún pequeño cambio. Por supuesto, cada cuál es responsable de las modificaciones que haga. El principio básico es “tú lo rompes, tú lo arreglas, no importa si está en el código propio o en el de otros”. Por último, vale la pena tener en cuenta que la existencia de tests automatizados impide que se produzca un desarrollo anárquico, al ser cada persona responsable de que todos los tests se ejecuten con éxito al incorporar los cambios que ha introducido al programa.

Integración continúa

En muchos casos la integración de código produce efectos laterales imprevistos, y en ocasiones la integración puede llegar a ser realmente traumática, cuando dejan de funcionar cosas por motivos desconocidos. La Programación Extrema hace que la integración sea permanente, con lo que todos los problemas se manifiestan de forma inmediata, en lugar de durante una fase de integración más o menos remota. La existencia de una fase de integración separada tiene dos efectos laterales indeseables: se empieza a hacer codificación “yo-yo”, en la que todo el mundo modifica código “sólo para que funcione, ya lo ajustaremos”, y hace que se acumulen defectos. Evitar que se acumulen defectos es muy importante para la XP, como lo es el conseguir que los defectos que cada programador inyecta los elimine él mismo.

Entregas pequeñas

Siguiendo la política de la XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia. Éstas deben ser tan pequeñas como sea posible, aunque deben añadir suficiente valor como para que resulten valiosas para el cliente. Una entrega no debería tardar más tres meses.

Estándares de programación

XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible, facilitando los cambios.

1.6.5 Fases de la metodología XP

Hay diversas prácticas inherentes a la Programación Extrema, en cada uno de los ciclos de desarrollo del proyecto.

Planificación

Historias de Usuario: El primer paso de cualquier proyecto que siga la metodología XP es definir las Historias de Usuario con el cliente.

Las Historias de Usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: Constan de tres ó cuatro líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base datos adecuados.

Release planning: Tras definir las Historias de Usuario es necesario crear un plan de publicaciones, donde se indiquen las Historias de Usuario que se implementarán para cada versión de la aplicación y las fechas en las que se publicarán dichas versiones.

Iteraciones. Todo proyecto que siga la metodología XP se ha de dividir en iteraciones de aproximadamente 3 semanas de duración. Al comienzo de cada iteración los clientes deben seleccionar las Historias de Usuario definidas en el "*Release planning*" que serán implementadas.

Diseño

- ✓ **Diseños simples:** La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácil de entender e implementar que a la larga costará menos tiempo y esfuerzo desarrollar.
- ✓ **Glosarios de términos:** Usar una correcta especificación de los nombres de clases, métodos y propiedades ayudará a comprender el diseño y facilitará futuras ampliaciones y la reutilización del código.
- ✓ **Tarjetas CRC:** El uso de las tarjetas CRC (*Class, Responsabilities and Collaboration*) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica.

Codificación

El cliente es una parte más del equipo de desarrollo; su presencia es indispensable en las distintas fases de XP. A la hora de codificar una historia de usuario su presencia es aún más necesaria. No olvidemos que los clientes son los que crean las Historias de Usuario y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario el cliente debe especificar detalladamente lo que ésta hará y también tendrá que estar presente cuando se realicen los tests que verifiquen que la historia implementada cumple la funcionalidad especificada. La codificación debe hacerse ateniendo a estándares y patrones de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y la escalabilidad. Crear test que prueben el funcionamiento de los distintos códigos implementados nos ayudará a desarrollar dicho código. Crear estos test antes nos ayuda a saber qué es exactamente lo que tiene que hacer el código a implementar y sabremos que una vez implementado pasará dichos test sin problemas ya que dicho código ha sido diseñado para ese fin.

Pruebas

Un punto importante es crear test que no tengan ninguna dependencia del código que en un futuro evaluará. Hay que crear los tests abstrayéndose del futuro código, de esta forma aseguraremos la independencia del test respecto al código que evalúa. El uso de los test es adecuado para observar la refactorización. Los test permiten verificar que un cambio en la estructura de un código no tiene por qué cambiar su funcionamiento.

Para asegurar el funcionamiento final de una determinada historia de usuario se deben crear “Test de aceptación”; estos tests son creados y usados por los clientes para comprobar que las distintas Historias de Usuario cumplen su cometido.

1.7 Conclusiones

En este capítulo se hace un estudio sobre las Prácticas Laborales y se exponen los conceptos necesarios para el entendimiento del software, se toman decisiones importantes, luego de un estudio de las distintas herramientas para el desarrollo de software tales como la elección de los lenguajes de programación y metodologías utilizadas.

CAPITULO 2: PLANIFICACION - DEFINICION

2.1 Introducción

En este capítulo se presenta parte de la propuesta de solución de la investigación, haciendo uso adecuado de las metodologías de desarrollo SXP en la que se aborda la fase de planificación y definición en el cual se explicará toda la dinámica del proyecto basándose en el expediente del proyecto.

La metodología está dividida en cuatro fases importante para el desarrollo del proyecto las cuales son las siguientes:

- Planificación-Definición
- Desarrollo
- Entrega
- Mantenimiento

Cada una de estas fases está compuesta por una serie de actividades que son las que generan los artefactos que quedan incluidos en el expediente de proyecto.

Procedimientos

Explica la estructura del proyecto planteando una serie de planillas que son responsabilidad de los diferentes roles que deben existir a la hora del desarrollo el software.

2.2 Planificación del proyecto por roles

El desarrollo de software con metodologías ágiles exige de la creación de pequeños grupos de trabajo, donde los roles son pocos, pero están bien definidas sus actividades. El principal aspecto antes de comenzar el proceso de documentación es distribuir las tareas por cada uno de los roles existentes, lo que garantiza un trabajo organizado, de ahí la necesidad de tenerlos bien definidos.

- Gerente (*Management*):

Es el responsable de tomar las decisiones finales, acerca de estándares y acuerdos a seguir durante el proyecto. Participa en la definición de objetivos y requerimientos. Tiene la responsabilidad de controlar el avance del software.

- Cliente (*Customer*):

El cliente contribuye a definir las Historias de Usuario y los casos de prueba de aceptación, para validar su implementación. Además, asigna la prioridad a las Historias de Usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio y participa en la concepción inicial del sistema.

El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema. Contribuye a definir las tareas que involucra la lista de reserva del producto.

- Programador (*Programmer*):

El programador define las tareas de ingeniería y produce el código del sistema. Además selecciona el estándar de programación a utilizar, controlando incluso la gestión recambios. Además dedica parte de su tiempo a la confección de los Manuales de usuario y de desarrollo.

- Analista (*Analyst*):

Escribe la concepción del sistema y las Historias de Usuario. Crea el Modelo de historia de usuario del negocio y la LRP. Además, asigna la prioridad a las Historias de Usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio, todo esto lo realiza junto con el cliente.

- Diseñador (*Designer*):

Son los encargados del diseño del sistema, así como el de los prototipos de interfaces, máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción.

- Encargado de Pruebas (*Tester*):

Escribe los casos de prueba de aceptación. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Es importante tener en cuenta, que a pesar de aplicar este tipo de organización por roles en los proyectos productivos, no se debe de crear un esquematismo, ni pensar que un integrante de un grupo de desarrollo, en todo el ciclo de vida de un software solamente se relacionará con su funcionalidad, pues en la mayoría de los casos ocurre un solapamiento de roles, es decir una persona puede desempeñar en diferentes etapas, en dependencia de su responsabilidad inicial, varios roles.

2.3 Planificación y Definición del proyecto

La fase de Planificación - Definición, es la primera que define la metodología Scrum - XP. En esta fase se generan todos los documentos que se encuentran relacionados con la concepción inicial del sistema, así como la definición del mismo. También se incluyen algunos que están vinculados a la primera parte de los procesos de Ingeniería de Software, tales como los relacionados con el negocio, los requisitos y el diseño. En la fase de planificación se incluyen aquellos documentos que están relacionados con la estimación inicial de esfuerzos, y la valoración de los riesgos. Cada una de las plantillas que han sido incluidas en esta fase se genera de una actividad en específico, y tienen su importancia en el proceso de documentación de software, por lo que son analizadas detalladamente a continuación, describiendo los objetivos que persiguen **(Susel 2008).**

2.3.1 Plantilla Concepción del sistema

La Plantilla Concepción del sistema, es el primer documento generado en la fase de Planificación-Definición, este queda elaborado luego de realizarse la actividad de entrevista con el cliente, momento en el que se define la concepción inicial del sistema. Este documento además de reflejar la visión general del producto a implementar, también recoge los diferentes roles que intervendrán en el desarrollo del software, así como las responsabilidades que tendrán en dicho proceso. Se recoge además cuales herramientas serán utilizadas para el desarrollo de la aplicación, el alcance que va a tener, una descripción de los involucrados en el negocio, cuales son los motivos de la necesidad del desarrollo del software y la propuesta de solución. Esta es la plantilla principal dentro del la documentación por ser la guía para los demás documentos que se generan durante el ciclo de desarrollo de software **(Ver anexo 1)**.

Roles: Analista y Cliente

2.3.2 Plantilla Modelo de Historias de Usuario del negocio

La plantilla del Modelo Historias de Usuario del negocio, es un artefacto que se genera del Juego de la planificación, luego de estar definida la concepción del sistema, se hace mucho más fácil comprender el negocio.

Se definen las características específicas del negocio, así como la forma en que interactúa el sistema con el cliente y viceversa. El Modelo de negocio cuando se trabaja con metodologías ágiles, es diferente al ya conocido en el proceso unificado, ya que en este caso se trabaja con Historias de Usuarios, en vez de con casos de uso. Pero independientemente de los cambios técnicos que puedan existir, el negocio se modela igual en cualquier metodología **(Ver anexo 2)**.

Roles: Analista

2.3.3 Plantilla Lista de Reserva del Producto (LRP)

La plantilla de Lista de Reserva del Producto, es el primer artefacto generado en la etapa de captura de requisitos, está conformada por una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes que casi siempre son más que suficientes para una iteración.

Esta lista puede crecer y modificarse a medida que se obtienen más conocimientos acerca del producto y del cliente. Con la restricción de que sólo puede cambiarse entre iteraciones. El objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto. Esta lista puede estar conformada por requerimientos técnicos y del negocio, funciones, errores a reparar, defectos, mejoras y actualizaciones tecnológicas requeridas **(Ver anexo 3)**.

Roles: Analista y Cliente

2.3.4 Plantilla Historias de Usuario

Las Historias de Usuario son la técnica utilizada en XP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el proceso unificado. El tratamiento de las Historias de Usuario es muy dinámico y flexible y tienen el mismo propósito que los casos de uso, las escriben los propios clientes, tal y como ven ellos las necesidades del sistema así como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas.

Las Historias de Usuario guían la construcción de las pruebas de aceptación, elemento clave en XP (deben generarse una serie de pruebas para verificar que la historia ha sido implementada adecuadamente) y son utilizadas para estimar tiempos de desarrollo. En este sentido, sólo proveen detalles suficientes para hacer una estimación razonable del tiempo que llevará implementarlas. En el momento de implementar una historia de usuario, se debe detallar a través de la comunicación con el cliente. En esta plantilla los campos de puntos estimados y puntos reales son llenados luego del desarrollo de la actividad de estimación de esfuerzo, en el cual se decide el tiempo que se le dedicará a cada historia de usuario **(Ver anexo 4)**.

Las Historias de Usuario proporcionan ventajas, tales como:

- Están escritas en lenguaje del cliente, por lo que es muy fácil su compresión.
- Especifican cada uno de los requisitos del sistema, sin necesidad de documentaciones extensas.
- Reflejan todas las características del sistema.
- Si se definen correctamente, guían el proceso de implementación.

Roles: Analista y Cliente

2.3.4.1 Estimación de esfuerzo

Las Historias de Usuario servirán para crear el plan estimado de entrega. Se convocará una reunión para crear el plan de entregas. El plan de entregas se usará para crear los planes de iteración para cada iteración. En esta reunión estará presente el desarrollador y el usuario. Con cada historia de usuario previamente evaluada en tiempo de desarrollo ideal, el cliente las agrupará en orden de importancia. Una

semana ideal es el tiempo que costaría implementar dicha historia si no tenemos nada más que hacer, incluyendo la parte de test correspondiente. Por otra parte, se mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las Historias de Usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las Historias de Usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

De esta forma se puede trazar el plan de entregas en función de estos dos parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente. Las iteraciones individuales son planificadas en detalle justo antes de que comience cada iteración.

Tabla 1: Estimación de esfuerzo

Historias de Usuario	Número	Puntos estimados en semanas
Visualizar información	1	1
Reportes asociados	2	2
Administración	3	2
Documentos	4	3
Gestionar prácticas	5	2
Gestionar tribunales de las PL	6	2
Gestionar lugares	7	2
Gestionar usuarios	8	2

2.3.5 Plantilla Lista de riesgos

La Plantilla de Lista de riesgos, es el documento que se genera de la actividad de Valoración de Riesgos. En ella quedan definidos los posibles riesgos que actuarán sobre el proceso de desarrollo de software, así como la estrategia trazada para mitigarlos, además de un plan de contingencia que describe que curso seguirán las acciones si el riesgo se materializa.

Los riesgos del proyecto amenazan al plan del proyecto. Es decir, si los riesgos del proyecto se hacen realidad, es probable que la planificación temporal del proyecto se retrase. Los riesgos del proyecto identifican los problemas potenciales de planificación temporal, personal (asignación y organización), recursos, cliente y requisitos y su impacto en un proyecto de software.

Los riesgos técnicos amenazan la calidad y la planificación temporal del software que hay que producir. Si un riesgo técnico se convierte en realidad, la implementación puede llegar a ser difícil o imposible. Los riesgos técnicos identifican problemas potenciales de diseño, implementación, de interfaz, verificación y de mantenimiento. Además las ambigüedades de especificaciones, incertidumbre técnica, técnicas anticuadas y las “tecnologías punta” son también factores de riesgo. Los riesgos técnicos ocurren porque el problema es más difícil de resolver de lo que pensábamos.

Esta plantilla posee una gran importancia, pues a pesar que es imposible definir desde un inicio todos los riesgos que pueda atravesar un proyecto, si se tendrán algunos en cuenta, fundamentalmente si se trata de un equipo de desarrollo con experiencia. Esta plantilla propicia algunas ventajas, tales como:

- Se definen los posibles riesgos, así como la forma de mitigarlos, lo que disminuye el efecto de los mismos, si ocurrieran.
- Se lleva un control de todos los problemas que han azotado al proyecto, así como de la manera que fueron enfrentados y el impacto que tuvieron en el proceso de desarrollo.
- Se incrementa la capacidad y probabilidades de éxito.
- Facilita el desarrollo del proyecto.
- Disminuye drásticamente las sorpresas en los proyectos.

Cuando se pone mucho en juego en un proyecto de software el sentido común nos aconseja realizar un Análisis de Riesgo. El tiempo invertido identificando, analizando y gestionando el riesgo merece la pena por muchas razones:

- Menos trastornos durante el proyecto.
- Una mayor habilidad de seguir y controlar el proyecto y la confianza que da planificar los problemas antes de que ocurran.

El Análisis de Riesgos puede absorber una cantidad significativa del esfuerzo de panificación del proyecto **(Ver anexo 5)**.

Roles: Gerente

2.4 Diseño

El Diseño es la única manera de materializar con precisión los requerimientos del cliente, es un proceso y un modelado a la vez, con un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del sistema a construir. El diseño debe implementar todos los requisitos explícitos contenidos en la Lista de Reserva. Debe ser una guía que puedan leer y entender los que construyan el código y los que prueban y mantienen el software. El diseño debe proporcionar una completa idea de lo que es el software, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista de la Implementación.

La Metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e implementable que a la larga costará menos tiempo y esfuerzo desarrollar.

En la fase de Diseño se define el proceso de aplicar ciertas técnicas y principios con el propósito de definir un Sistema, con suficientes detalles como para permitir su interpretación y realización física. Este transforma elementos estructurales de la arquitectura del programa, la importancia del diseño del software se puede definir en una sola palabra calidad, dentro del diseño es donde se fomenta la calidad del proyecto.

2.4.1 Plantilla Modelo de diseño

El Modelo de Diseño identifica los objetos que el sistema contendrá y las actividades que el sistema va a automatizar. El modelo de diseño es la combinación de estos dos aspectos. El modelo de diseño es también un modelo abstracto en el que no se incluye un alto nivel de detalle. La plantilla del Modelo de diseño, es el documento que se genera del diseño con las metáforas, donde se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema que se asume de forma evolutiva y los posibles inconvenientes que se pueden generar por no contar con ella en el comienzo del proyecto. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema.

Teniendo en cuenta las características anteriores, se define en esta plantilla, un esbozo inicial del diseño del sistema, sin entrar en especificaciones, ni detalles, solo lo que el diseñador necesita para hacer un primer entregable del sistema **(Ver anexo 6)**.

Roles: Diseñador

2.5 Planificación de las iteraciones

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 Historias de Usuario se incluirán en un *release*, lo cual debería ser consistente con el contenido de una o dos iteraciones.

2.5.1 Primera iteración: Visualizar información y Reportes asociados

En esta primera iteración tiene como objetivo darle cumplimiento a las Historias de Usuario que se consideraron de importancia inicial para el desarrollo del sistema. Las Historias de Usuario mencionadas brindan funcionalidades como obtener la información que el usuario necesita.

Además se tendrá la primera versión de prueba, que contará con una vista inicial del sistema esta primera versión se le presentarán al cliente con el objetivo de obtener una retroalimentación del mismo para posteriores iteraciones del producto.

2.5.2 Segunda iteración: Administración

En esta iteración se les dará cumplimiento a la Historia de Usuario administración del sistema en la que se establece la política de autenticación y autorización que les permite a los administradores del sistema autenticarse, así como Gestionar documentos, Gestionar usuarios del sistema, Gestionar prácticas, Gestionar lugares y Gestionar los tribunales de las Prácticas Laborales.

El resultado de esta iteración integrada a la iteración anterior dará como resultado la segunda versión del sistema, además de la segunda versión de prueba del mismo. Esta versión será entregada al cliente para verificar si cumple con los requisitos acordados.

2.5.3 Tercera iteración: Documentos, Gestionar prácticas, Gestionar lugares, Gestionar usuarios y Gestionar tribunales de las PL

En esta iteración se tiene como objetivo darle cumplimiento las Historias de Usuario Documentos, Gestionar prácticas, Gestionar lugares ,Gestionar usuarios y Gestionar tribunal de las PL las cuales tiene algunas funcionalidades tales como subir y descargar archivo, insertar, eliminar y modificar la información correspondiente con la gestión deseada. Estas funcionalidades solo podrán ser ejecutadas por los administradores del sistema.

El resultado alcanzado en esta iteración será integrado a los resultados de las iteraciones anteriores obteniéndose la tercera versión del sistema. Esta versión será presentada al cliente para su verificación.

2.5.4 Plan de duración de las iteraciones

Partiendo de las Historias de Usuario planteadas en la planilla correspondiente a la misma se realiza una planificación en tres iteraciones basándose en el tiempo y procurando agrupar la funcionalidad relacionada en la misma iteración.

Tabla 2: Planificación de la iteración en semanas

Iteración	Orden de implementación por historia de usuario	Duración total de la iteración en semanas
1	• Visualizar información • Reportes asociados	1 + 2 = 3
2	• Administración	2
3	• Documentos • Gestionar prácticas • Gestionar tribunales de las PL • Gestionar usuarios • Gestionar lugares	3 + 2 + 2 + 2 + 2 = 11

2.6 Conclusiones

En este capítulo se abordado la fase de planificación y definición en el cual se explicará parte de la dinámica del proyecto basándose en el expediente del proyecto, se planificaron las Historias de Usuario que se deben tener en cuenta para la construcción del sistema, así como las iteraciones a partir de la estimación de esfuerzos de cada Historia de Usuario. El desarrollo del sistema fue realizado en tres iteraciones. Se concluye que este capítulo está listo para arribar a la siguiente etapa de desarrollo.

CAPITULO 3: DESARROLLO

3.1 Introducción

En este capítulo se abordan los elementos pertenecientes a la fase de desarrollo acorde a la metodología de desarrollo. Se presenta las tarjetas CRC les permiten al programador centrarse y apreciar el desarrollo orientado a objetos También aparecen las tareas de ingeniería para llevar a cabo el desarrollo de las Historias de Usuario.

3.2 Diseño de la Solución Propuesta

La metodología de desarrollo XP establece prácticas especializadas, que inciden directamente en la realización y elaboración del diseño de un software, sin embargo no requiere que la representación del sistema sea mediante diagramas de clases basados en UML, sino que pueden emplearse indistintamente sencillos esquemas descritos en pizarras u otras técnicas como las tarjetas CRC. No obstante el empleo de los diagramas UML pueden ser utilizados siempre y cuando contribuyan en el mejoramiento de la comunicación del equipo de desarrollo, no sean muy extensos y no requieran de mucho tiempo para su creación.

3.2.1 Tarjetas CRC

El uso de las tarjetas CRC (*Class, Responsabilities and Collaboration*) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Las tarjetas CRC representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad. Las tarjetas CRC fueron propuestas por *Ward Cunningham and Kent Beck*.

Tabla 3: Tarjeta CRC Principal

Nombre de la clase: Principal

Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Visualizar información	Visualizar
Reportes asociados al problema	Reportes

Tabla 4: Tarjeta CRC Visualizar

Nombre de la clase: Visualizar	
Tipo de la clase: Utilitario	
Responsabilidades:	**Colaboradores:**
Visualizar estudiantes	
Visualizar tribunales de las PL	
Visualizar los usuarios del sistema	
Visualizar los lugares de las PL	
Visualizar documentos	
Visualizar banco de problemas	

Tabla 5: Tarjeta CRC Reportes

Nombre de la clase: Reportes	
Tipo de la clase: Utilitario	
Responsabilidades:	**Colaboradores:**
Generar reportes asociados al problema	

Tabla 6: Tarjeta CRC Administrar

Nombre de la clase: Administrar	
Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Administración del sistema	Admin_sistema
Loguearse	Login

Tabla 7: Tarjeta CRC Admin_ sistema

Nombre de la clase: Admin_ sistema	
Tipo de la clase: Utilitario	
Responsabilidades:	**Colaboradores:**
Encargado de administrar el sistema	

Tabla 8: Tarjeta CRC Login

Nombre de la clase: Login	
Tipo de la clase: Controladora	
Responsabilidades:	**Colaboradores:**
Permitir a los usuarios autorizados loguearse	Admin_sistema

Tabla 9: Tarjeta CRC Gestión _ datos

Nombre de la clase: Gestión _ datos

Tipo de la clase: Lógica del negocio	
Responsabilidades:	**Colaboradores:**
Insertar un estudiante con información de las PL	Insertar
Eliminar estudiante de las PL	Eliminar
Modificar estudiante con la información de las PL	Modificar
Buscar estudiante	Buscar
Crear usuario	
Eliminar usuario	
Cambiar contraseña usuario	
Eliminar tribunal	
Insertar tribunal	
Insertar lugares	
Eliminar lugares	
Subir documentos	
Eliminar documentos	

Tabla 10: Tarjeta CRC Eliminar

Nombre de la clase: Eliminar	
Tipo de la clase: Utilitario	
Responsabilidades:	**Colaboradores:**

Eliminar estudiante	
Eliminar tribunal	
Eliminar usuarios deseados	
Eliminar lugares	
Eliminar documentos deseado	

Tabla 11: Tarjeta CRC Insertar

Nombre de la clase: Insertar	
Tipo de la clase: Utilitario	
Responsabilidades:	**Colaboradores:**
Adicionar estudiante a las Prácticas	
Adicionar un tribunal	
Crear cuenta de usuario	
Insertar lugar	
Subir documentos	

Tabla 12: Tarjeta CRC Modificar

Nombre de la clase: Modificar	
Tipo de la clase: Utilitario	
Responsabilidades:	**Colaboradores:**
Modificar estudiante con la información de las Prácticas Laborales	

Cambiar contraseña del usuario deseado	

Tabla 13: Tarjeta CRC Buscar

Nombre de la clase: Buscar	
Tipo de la clase: Utilitario	
Responsabilidades:	**Colaboradores:**
Realizar búsqueda de un estudiante	

3.3 Patrón de desarrollo Modelo – Vista – Controlador

El Modelo Vista Controlador (Model View Controller, MVC por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la interface de usuario y el código es el que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista.

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista **(Díaz 2009).**

3.4 Desarrollo del proyecto

En la metodología XP se convierte en un integrante más del equipo de desarrollo el cliente pues él crea las Historias de Usuario bajo la supervisión de los desarrolladores. Estas historias quedan confeccionadas cuando el cliente es capaz de identificar con precisión la funcionalidad deseada, además, también debe estar presente cuando se realicen las pruebas de aceptación para cada historia, por lo que su presencia es imprescindible.

En XP generalmente cada historia de usuario se divide en tareas de ingeniería (TI) o tareas de programación. Estas se crean para obtener una mejor planificación de la historia; con ellas se pretende cumplir con las funcionalidades básicas que luego conformarán las funcionalidades generales de cada historia. . Las tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguajes técnicos, pues las mismas son usadas únicamente por los programadores. A continuación se muestran algunas de las tareas de ingeniería en cada iteración del sistema.

3.4.1 Plantilla Tarea de ingeniería

La plantilla Tarea de ingeniería, tiene gran importancia, pues permite definir cada una de las actividades asociadas a las Historias de Usuario y que consentirán su implementación. Así se estimar el tiempo que se llevará cada historia de usuario en implementarse, de acuerdo a su complejidad.

Esta plantilla proporciona ventajas tales como:

- Permite organizar el proceso de implementación, pues las tareas se van implementando de acuerdo a su prioridad.
- Posibilita conocer el grado de complejidad de cada historia de usuario, teniendo en cuenta la cantidad de tareas asociadas.

Roles: Programador

Tabla 14: Plantilla Tarea de ingeniería

	Tarea de Ingeniería

Número Tarea: [Los números deben ser consecutivos]	**Número Historia de Usuario:** [Número de la historia de usuario a la que pertenece la tarea]
Nombre Tarea: [Nombre que identifica a la tarea.]	
Tipo de Tarea : [Las tareas pueden ser de: Desarrollo, Corrección, Mejora, Otra(Especificar)]	**Puntos Estimados:** [Tiempo en semanas que se le asignará. (Estimado)]
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: [Nombre y Apellidos del programador]	
Descripción: [Breve descripción de la tarea.]	

3.4.2 Primera iteración

En esta iteración se les dio cumplimiento a la implementación de la Historia de Usuario número 1 y 2 dicha HU brinda funcionalidades como visualizar la información, generar reportes del sistema y descargar documentos.

Tabla 15: Historias de Usuario primera iteración

Historias de Usuario	Tiempo de Implementación (semanas)	
	Estimación	Real
Visualizar información	**1**	**1**
Reportes asociados	**2**	**1**

Tareas de ingeniería vinculadas a esta iteración

Tabla 16: Tareas de ingeniería # 1 Visualizar información

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Visualizar información	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha inicio: 15 de abril 2010	**Fecha fin:** 18 de abril 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe visualizar toda la información relacionada con las Prácticas Laborales.	

Tabla 17: Tareas de ingeniería # 2 Descargar documentos

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Descargar documentos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha inicio: 15 de abril 2010	**Fecha fin:** 18 de abril 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir descargar los documentos deseados.	

Tabla 18: Tareas de ingeniería # 3 Reportes asociados

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 2
Nombre Tarea: Reportes asociados	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha inicio: 19 de abril 2010	**Fecha fin:** 23 de abril 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe mostrar reportes con los datos asociados a los estudiantes.	

3.4.3 Segunda iteración

En esta iteración del sistema se les dará cumplimiento a la Historia de Usuario número 3 dicha HU tiene como funcionalidad principal permitir la autenticación de los usuarios, y la gestión de la información.

Tabla 22: Historias de Usuario segunda iteración

Historias de Usuario	Tiempo de Implementación (semanas)	
	Estimación	Real
Administración	**2**	**2**

Tareas de ingeniería vinculadas a esta iteración

Tabla 19: Tareas de ingeniería # 4 Visualizar administración

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 5
Nombre Tarea: Visualizar administración	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 24 de abril 2010	**Fecha fin:** 25 de abril 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir mostrar la interfaz administración para autenticarse y gestionar los datos deseados.	

3.4.4 Tercera iteración

En esta iteración del sistema se les dará cumplimiento a las Historias de Usuario número 4,5,6,7 y 8 las cuales tienen como funcionalidades principales insertar, eliminar, modificar y buscar los datos de la gestión correspondiente, dicha funcionalidad solo podrán ser realizadas por los administradores.

Tabla 20: Historias de Usuario tercera iteración

Historias de Usuario	Tiempo de Implementación (semanas)	
	Estimación	Real
Documentos	**3**	**2**
Gestionar prácticas	**2**	**3**
Gestionar tribunales de las PL	**2**	**2**
Gestionar usuarios	**2**	**2**
Gestionar lugares	**2**	**2**

Tareas de ingeniería vinculadas a esta iteración

Tabla 21: Tareas de ingeniería # 5 Subir documentos

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 4
Nombre Tarea: Subir documentos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 26 de abril 2010	**Fecha fin:** 27 de abril 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir subir documentos.	

Tabla 22: Tareas de ingeniería # 6 Eliminar documentos

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 4
Nombre Tarea: Eliminar documentos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 28 de abril 2010	**Fecha fin:** 29 de abril 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir eliminar los documentos deseados.	

Tabla 23: Tareas de ingeniería # 7 Insertar estudiante

	Tarea de Ingeniería
Número Tarea: 7	**Número Historia de Usuario:** 5
Nombre Tarea: Insertar estudiante	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 5 de mayo 2010	**Fecha fin:** 7 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir insertar un estudiante con toda la información relacionada.	

Tabla 24: Tareas de ingeniería # 8 Eliminar estudiante

	Tarea de Ingeniería
Número Tarea: 8	**Número Historia de Usuario:** 5
Nombre Tarea: Eliminar estudiante	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 8 de mayo 2010	**Fecha fin:** 10 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir eliminar un estudiante.	

Tabla 25: Tareas de ingeniería # 9 Modificar estudiante

	Tarea de Ingeniería
Número Tarea: 10	**Número Historia de Usuario:** 5
Nombre Tarea: Modificar estudiante	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 11 de mayo 2010	**Fecha fin:** 13 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir modificar un estudiante de las Prácticas Laborales.	

Tabla 26: Tareas de ingeniería # 10 Buscar estudiante

	Tarea de Ingeniería
Número Tarea: 10	**Número Historia de Usuario:** 5
Nombre Tarea: Buscar estudiante	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 15 de mayo 2010	**Fecha fin:** 16 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir buscar un estudiante.	

Tabla 27: Tareas de ingeniería # 11 Insertar tribunal

	Tarea de Ingeniería
Número Tarea: 11	**Número Historia de Usuario:** 6
Nombre Tarea: Insertar tribunal	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1

Fecha inicio: 17 de mayo 2010	**Fecha fin:** 19 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir insertar un tribunal a las Prácticas Laborales.	

Tabla 28: Tareas de ingeniería # 12 Eliminar tribunal

	Tarea de Ingeniería
Número Tarea: 12	**Número Historia de Usuario:** 6
Nombre Tarea: Eliminar tribunal	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 20 de mayo 2010	**Fecha fin:** 21 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir eliminar un tribunal de las Prácticas Laborales.	

Tabla 29: Tareas de ingeniería # 13 Insertar lugar

	Tarea de Ingeniería
Número Tarea: 13	**Número Historia de Usuario:** 7
Nombre Tarea: Insertar lugar	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 22 de mayo 2010	**Fecha fin:** 24 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir insertar un lugar donde el estudiante realizará las Prácticas Laborales.	

Tabla 30: Tareas de ingeniería # 14 Eliminar lugar

	Tarea de Ingeniería
Número Tarea: 14	**Número Historia de Usuario:** 7
Nombre Tarea: Eliminar lugar	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 24 de mayo 2010	**Fecha fin:** 25 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitirle eliminar el lugar deseado.	

Tabla 31: Tareas de ingeniería # 15 Crear cuenta de usuario

	Tarea de Ingeniería
Número Tarea: 15	**Número Historia de Usuario:** 8
Nombre Tarea: Crear cuenta de usuario	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 27 de mayo 2010	**Fecha fin:** 28 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitirle al administrador crear usuarios para gestionar la información.	

Tabla 32: Tareas de ingeniería # 16 Eliminar usuario

	Tarea de Ingeniería
Número Tarea: 16	**Número Historia de Usuario:** 8
Nombre Tarea: Eliminar usuario	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 28 de mayo 2010	**Fecha fin:** 30 de mayo 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir a los administradores eliminar usuarios.	

Tabla 33: Tareas de ingeniería # 17 Cambiar contraseña usuario

	Tarea de Ingeniería
Número Tarea: 17	**Número Historia de Usuario:** 8
Nombre Tarea: Cambiar contraseña usuario	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 1 de junio 2010	**Fecha fin:** 3 de junio 2010
Programador responsable: Osmany Santana Díaz	
Descripción: El sistema debe permitir cambiar la contraseña de los usuarios creados por el administrador.	

3.4.5 Plantilla Cronograma de producción

El cronograma de producción no es más que la plantilla que recoge las actividades realizadas a desarrollo durante la iteración. Se realiza un cronograma para cada iteración planificada durante el proceso **(Ver anexo 8)**.

3.5 Conclusiones

Con la realización de este capítulo hemos abordado la fase de desarrollo planteada por la metodología SXP en el cual se hace referencia a las tareas de ingeniería correspondientes a cada historia de usuario, se elaboraron las tarjetas CRC para apreciar el desarrollo orientado a objetos y se realizo el diseño de los datos.

CAPITULO 4: PRUEBAS

4.1 Introducción

En este capítulo se aborda las pruebas de aceptación que fueron realizadas por el cliente para demostrar que el sistema funcione adecuadamente. Estas pruebas fueron llevadas a cabo antes de cada entrega que se realizó durante todo el desarrollo del proyecto.

4.2 Pruebas de software

Las pruebas de software son elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.

La creciente percepción del software como un elemento del sistema y la importancia de los costos asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien planificadas. No es raro que una organización de desarrollo de software emplee entre el treinta y cuarenta por ciento del esfuerzo total del proyecto en las pruebas.

Una vez generado el código fuente, el software debe ser probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente. El objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores. Aquí es donde se aplican las técnicas de pruebas del software. Estas técnicas facilitan una guía sistemática para diseñar pruebas que:

- Comprueban la lógica interna de los componentes software.
- Verifican los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento **(Pressman 2002).**

En la metodología XP es esencial el desarrollo de las pruebas, permitiendo probar constantemente el código. Cada vez que se quiere implementar las funcionalidades que tendrá el software, XP propone una redacción sencilla de prueba, para ser pasada por el código posteriormente. XP posee dos tipos de pruebas; las unitarias o TDD (desarrollo dirigido por pruebas, del inglés *Test Driven Development*), desarrolladas por los programadores verificando su código de forma automática y las pruebas de aceptación que son las que se utilizaran en este capítulo, dichas pruebas

serán evaluadas luego de culminar una iteración verificando así si se cumplió la funcionalidad requerida por el cliente.

4.2.1 Desarrollo Dirigido por Pruebas

El desarrollo dirigido por pruebas (TDD), se enfoca en la implementación orientada a pruebas. El código debe ser probado paso a paso y obtener un resultado, aunque aún no con lógica para el negocio, pero sí funcional. Algunas personas confunden este término con las nombradas “pruebas de caja blanca”, las cuales son pruebas que se realizan a los métodos u operaciones para medir la funcionalidad del mismo desde la perspectiva de la validez para el 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. Solo 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 informalmente como “refactorización” de código, el cual consiste en limpiarlo, organizarlo y adaptarlo a los patrones. En esencia, TDD se centra en la lógica del código y las pruebas de caja blanca en la del negocio.

4.2.2 Pruebas de Aceptación

Las pruebas de aceptación en la metodología XP, se pueden asociar con las pruebas de caja negra que se aplican en otras metodologías de desarrollo, sólo que en XP se crean a partir de las Historias de Usuario y no por un listado de requerimientos. Durante las iteraciones las Historias de Usuarios seleccionadas serán traducidas 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. Una Historia de Usuario puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de éstas es garantizar que las funcionalidades requeridas por el cliente han sido cumplidas. Una Historia de Usuario no se considera completa hasta que no ha pasado por sus pruebas de aceptación.

4.3 Objetivos de las pruebas

El objetivo de las pruebas, expresado de forma sencilla, es encontrar el mayor número posible de errores con una cantidad razonable de esfuerzo, aplicado sobre un lapso de tiempo realista. Se debe ejecutar el programa antes de que llegue al cliente, con la intención especifica de descubrir todos los errores, de manera que el cliente no experimente la frustración asociada con un producto de baja calidad.

Con el propósito de encontrar el mayor número posible de errores, las pruebas deben conducirse sistemáticamente.

Otro de sus objetivos son que:

- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.
- Reducir costos de mantenimiento.
- Obtener información concreta acerca de fallas, que pueda usarse como apoyo en la mejora de procesos, y en la de los desarrolladores.

Los objetivos anteriores suponen un cambio dramático de punto de vista. Nos quitan la idea que, normalmente, tenemos de que una prueba tiene éxito si no descubre errores. Nuestro objetivo es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo. Si la prueba se lleva a cabo con éxito (de acuerdo con el objetivo anteriormente establecido), descubrirá errores en el software. Como ventaja secundaria, la prueba demuestra hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento **(Pruebas 2008).**

4.4 Alcance de las pruebas

La prueba de software tiene limitantes, tanto teóricos como prácticos. Desde el punto de vista teórico, la prueba es un problema que llamamos no-decidible; esto implica, de algún modo, que no podemos escribir un programa que pruebe los programas sin intervención humana. Sin embargo, como mencionábamos

anteriormente, la prueba sí es automatizable en muchos aspectos. Desde el punto de vista práctico, la cantidad de posibilidades para probar exhaustivamente un sistema es sencillamente inmanejable; es necesario entonces utilizar técnicas adecuadas para maximizar la cantidad de fallas importantes encontradas con los recursos asignados. Cada método que se utilice para detectar defectos deja un residuo de defectos más sutiles contra los cuales ese método es ineficaz (la llamada “Paradoja del Pesticida”).

La prueba de software implica pues, la aplicación de técnicas y herramientas apropiadas en el marco de un proceso bien definido, determinado por el tipo de proyectos de desarrollo de software que se abordan.

4.5 Plantilla Caso de prueba de aceptación

La plantilla de Caso de prueba de aceptación, se genera de la etapa de pruebas. El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.

Roles: Cliente y Tester.

Tabla 34: Prueba de aceptación para la HU “Visualizar información”

	Caso de Prueba de Aceptación
Código Caso de Prueba: PL_ HU1_P1	**Nombre Historia de Usuario:** Visualizar información
Nombre de la persona que realiza la prueba: Osmany Santana Díaz	
Descripción de la Prueba: Se realiza una prueba para comprobar que el sistema muestra la información de las Prácticas Laborales relacionada con los estudiantes, además debe permitir descargar problema, guía de informe, P4, modelo de solicitud y carta de aceptación según al año a que pertenece.	
Condiciones de Ejecución: El cliente debe probar que se muestre toda la información.	
Entrada / Pasos de ejecución: Se solicita mostrar toda la información deseada y luego se verifica que la información sea la adecuada.	
Resultado Esperado: El sistema no presenta errores.	

Evaluación de la Prueba: Satisfactoria

Tabla 35: Prueba de aceptación para la HU "Reportes asociados"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU2_P2	**Nombre Historia de Usuario:** Reportes asociados
Nombre de la persona que realiza la prueba: Osmany Santana Díaz	
Descripción de la Prueba: Se realiza una prueba para comprobar que se muestren los Reportes asociados con la información de los estudiantes.	
Condiciones de Ejecución: El cliente debe probar que se muestren todos los Reportes asociados.	
Entrada / Pasos de ejecución: Se solicita mostrar todos los Reportes asociados deseados y luego se verifica que la información sea la adecuada.	
Resultado Esperado: El sistema no exhibe errores.	
Evaluación de la Prueba: Satisfactoria	

Tabla 36: Prueba de aceptación para la HU "Administración"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU3_P3	**Nombre Historia de Usuario:** Administración
Nombre de la persona que realiza la prueba: Osmany Santana Díaz	
Descripción de la Prueba: Se realiza una prueba para verificar la administración de los datos.	
Condiciones de Ejecución: El cliente debe probar que la Administración cumple con la expectativa esperada.	
Entrada / Pasos de ejecución: Se solicita Administración de los datos y luego se verifica que la información funcione correctamente.	
Resultado Esperado: El sistema no exhibe errores.	
Evaluación de la Prueba: Satisfactoria	

Tabla 37: Prueba de aceptación para la HU "Documentos"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU4_P4	**Nombre Historia de Usuario:** Documentos
Nombre de la persona que realiza la prueba: Osmany Santana Díaz	
Descripción de la Prueba: Se realiza una prueba para gestionar los documentos de las Prácticas Laborales.	
Condiciones de Ejecución: El cliente debe probar que se muestren los documentos necesarios.	
Entrada / Pasos de ejecución: Se solicita mostrar la información y luego se verifica que la información trabaje correctamente.	
Resultado Esperado: El sistema no exhibe errores.	
Evaluación de la Prueba: Satisfactoria	

Tabla 38: Prueba de aceptación para la HU "Gestionar prácticas"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU5_P5	**Nombre Historia de Usuario:** Gestionar prácticas
Nombre de la persona que realiza la prueba: Osmany Santana Díaz	
Descripción de la Prueba: Se realiza una prueba para verificar la Gestión de las Prácticas con su información relacionada.	
Condiciones de Ejecución: El cliente debe probar que todos los aspectos de la Gestión cumplan con las perspectivas esperadas.	
Entrada / Pasos de ejecución: Se solicita la Gestión de las prácticas y luego se verifica que la información funcione correctamente.	
Resultado Esperado: El sistema no exhibe errores.	
Evaluación de la Prueba: Satisfactoria	

Tabla 39: Prueba de aceptación para la HU "Gestionar tribunales de las PL"

Caso de Prueba de Aceptación	
Código Caso de Prueba: PL_ HU6_P6	**Nombre Historia de Usuario:** Gestionar tribunales de las PL

Nombre de la persona que realiza la prueba: Osmany Santana Díaz
Descripción de la Prueba: Se realiza una prueba para verificar la Gestión del los tribunales de las Prácticas Laborales.
Condiciones de Ejecución: El cliente debe probar que todos los aspectos de la Gestión del tribunal de las Prácticas Laborales cumplan con las perspectivas esperadas.
Entrada / Pasos de ejecución: Se solicita la Gestión del tribunal y luego se verifica que la información funcione correctamente.
Resultado Esperado: El sistema no exhibe errores.
Evaluación de la Prueba: Satisfactoria

Tabla 40: Prueba de aceptación para la HU "Gestionar lugares"

	Caso de Prueba de Aceptación
Código Caso de Prueba: PL_ HU7_P7	**Nombre Historia de Usuario:** Gestionar lugares
Nombre de la persona que realiza la prueba: Osmany Santana Díaz	
Descripción de la Prueba: Se realiza una prueba para verificar la Gestión del los lugares donde se realizaran las Prácticas Laborales.	
Condiciones de Ejecución: El cliente debe probar que todos los aspectos de la Gestión de los lugares de las Prácticas cumplan con las perspectivas esperadas.	
Entrada / Pasos de ejecución: Se solicita la Gestión del los lugares de las Prácticas Laborales y luego se verifica que la información funcione correctamente.	
Resultado Esperado: El sistema no exhibe errores.	
Evaluación de la Prueba: Satisfactoria	

Tabla 41: Prueba de aceptación para la HU "Gestionar usuarios"

	Caso de Prueba de Aceptación
Código Caso de Prueba: PL_ HU8_P8	**Nombre Historia de Usuario:** Gestionar usuarios
Nombre de la persona que realiza la prueba: Osmany Santana Díaz	
Descripción de la Prueba: Se realiza una prueba para verificar la Gestión del los usuarios del sistema.	

Condiciones de Ejecución: El cliente debe probar que todos los aspectos de la Gestión de usuarios del sistema cumplan con las perspectivas esperadas.
Entrada / Pasos de ejecución: Se solicita la Gestión del los usuarios y luego se verifica que la información funcione correctamente.
Resultado Esperado: El sistema no exhibe errores.
Evaluación de la Prueba: Satisfactoria

4.6 Conclusiones

Con la elaboración de este capítulo hemos abordado la fase de pruebas planteada por la metodología SXP, con la realización de las pruebas de aceptación en las que el cliente se asegura de que las funciones implementadas cumplan su objetivo satisfactoriamente, probando individualmente cada HU y asignándole la evaluación correspondiente. Todas las pruebas que se realizaron fueron efectivas y el cliente estuvo satisfecho, cumpliendo entonces el sistema con las Historias de Usuarios definidas inicialmente.

CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD

5.1 Introducción

Para estudiar la 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 conjunta de dos factores:

- 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 la capacidad del proyecto para satisfacer la necesidad, solucionar el problema o lograr el objetivo para el cual se ideó, es decir, un proyecto será más menos efectivo con relación al mayor o menor cumplimiento que alcance en la finalidad para la cual fue ideado (costo por unidad de cumplimiento del objetivo) **(Subiros 2009).**

5.1 Efectos económicos

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

5.1.1 Efectos directos

- ➢ Positivos
  - ✓ Se gestiona la información necesaria a la que los usuarios finales del sitio podrán acceder.
  - ✓ Mayor integración usuario- artefactos, ya que por medio de este el usuario siente necesidades de interactuar con el portal, debido a que este facilita y le brinda la información necesaria.
  - ✓ Se cuenta con una herramienta capaz de mantener la seguridad e integridad de la información difundida.

- ✓ Permite al usuario estar informado y organizado ante cualquier tarea tanto de ámbito nacional como institucional.
- ✓ Facilita a usuarios con determinadas roles difundir información por medio del portal.

➢ Negativos

- ✓ Para usar la aplicación es imprescindible el uso de un ordenador, aparejado a los gastos que este trae de consumo de corriente eléctrica y mantenimiento.

5.1.2 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 comercializarse.

5.1.3 Externalidades

- Se contará con una herramienta disponible que permitirá a los usuarios acceder a la información de las Prácticas Laborales de una forma segura y rápida.

5.1.4 Intangibles

En la valoración económica siempre hay elementos perceptibles por una comunidad 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 Proyecto (Solución Manual):**

  La información de las Prácticas Laborales en el Departamento de Informática es recogida mediante documentos Excel y de forma manual lo que provoca confusión y perdida de información al manejar todo el contenido de los estudiantes que realizan sus Prácticas Laborales.

- ✓ **Situación con Proyecto (Solución Automatizada):**

Los usuarios podrán ver el contenido solo con entrar al sistema. Para la gestión de los datos se debe autenticarse, si es administrador del sistema, siguiendo los pasos correspondientes:

- Los usuarios deben seleccionar las opciones de su conveniencia.
- Si desea gestionar la información y está autorizado debe autenticarse.
- El sistema pedirá usuario y contraseña para la gestión de la información.
- Luego de autenticado el sistema permitirá buscar, insertar, eliminar, actualizar y mostrar la información deseada.

5.2 Beneficios y Costos Intangibles en el proyecto

COSTOS:

- ✓ Resistencia al cambio.

BENEFICIOS:

- ✓ Mayor comodidad para los usuarios.
- ✓ Mejora en la calidad de la información.
- ✓ Menor tiempo empleado en la introducción de los datos.
- ✓ Facilidad a la hora de buscar la información.

5.3 Ficha de costo

Para determinar el costo económico del proyecto se utilizará el procedimiento para elaborar Una Ficha De Costo de un Producto Informático [Dra. Ana María Gracia Pérez, UCLV]. Para la elaboración de la ficha se consideran los siguientes elementos de costo, desglosados en moneda libremente convertible y moneda nacional.

Costos en Moneda Libremente Convertible:

- ✓ Costos Directos.

1. Compra de equipos de cómputo: No procede.
2. Alquiler de equipos de cómputo: No procede.
3. Compra de licencia de Software: No procede.

4. Depreciación de equipos: $ 60.78
5. Materiales directos: No procede.

Total: $ 60.78

✓ Costos Indirectos.

1. Formación del personal que elabora el proyecto: No procede.
2. Gastos en llamadas telefónicas: No procede.
3. Gastos para el mantenimiento del centro: No procede.
4. Know How: No procede.
5. Gastos en representación: No procede.

Total: $0.00

✓ Gastos de distribución y venta.

1. Participación en ferias o exposiciones: No procede.
2. Gastos en transportación: No procede.
3. Compra de materiales de propagandas: No procede.

Total: $0.00

Costos en Moneda Nacional:

✓ Costos Directos.

1. Salario del personal que laborará en el proyecto: $100.00
2. El 5% del total de gastos por salarios se dedica a la seguridad social: No procede.
3. El 0.09% de salario total, por concepto de vacaciones a acumular: No procede.
4. Gasto por consumo de energía eléctrica: $ 5.94
5. Gastos en llamadas telefónicas: No procede.
6. Gastos administrativos: No procede.

✓ Costos Indirectos.

1. Know How: $ 108.75

Total: $ 214.69

Como se hizo referencia anteriormente, la técnica seleccionada para evaluar la factibilidad del proyecto es la Metodología Costo-Efectividad. Dentro de esta metodología la técnica de punto de equilibrio aplicable a proyectos donde los beneficios tangibles no son evidentes el análisis se basa exclusivamente en los costos. 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, tómese como costo el tiempo empleado para desarrollar el proceso de gestión de las Prácticas Laborales y la variable sería la cantidad de pasos a realizar, para llevar a cabo el proceso, para lo cual tenemos 2 valores.

Valores de la variable (Solución manual)

1. El usuario entra la dirección del sistema. (2 min).
2. Determinar la información de su privilegio que será visualizada en el sistema. (3 min).
3. El sistema devuelve la información que ha sido seleccionada. (7 min).
4. El usuario hace uso de la información que necesita. (10 min).

Valores de la variable (Solución con el programa)

1. Cargar formulario con la información de los estudiantes durante el transcurso de las Prácticas Laborales, 11 variables. (5 min).
2. Cargar formulario con la información de los tribunales que estarán presente en la discusión de las Prácticas laborales, 2 variables. (10 min).
3. Proponer los diferentes lugares donde se realizan las Prácticas Laborales y de no existir incluir uno nuevo, 2 variables. 15 min).
4. Asignarle a cada estudiante un tribunal donde el profesor tutor no debe estar incluido en el mismo. (10 min).

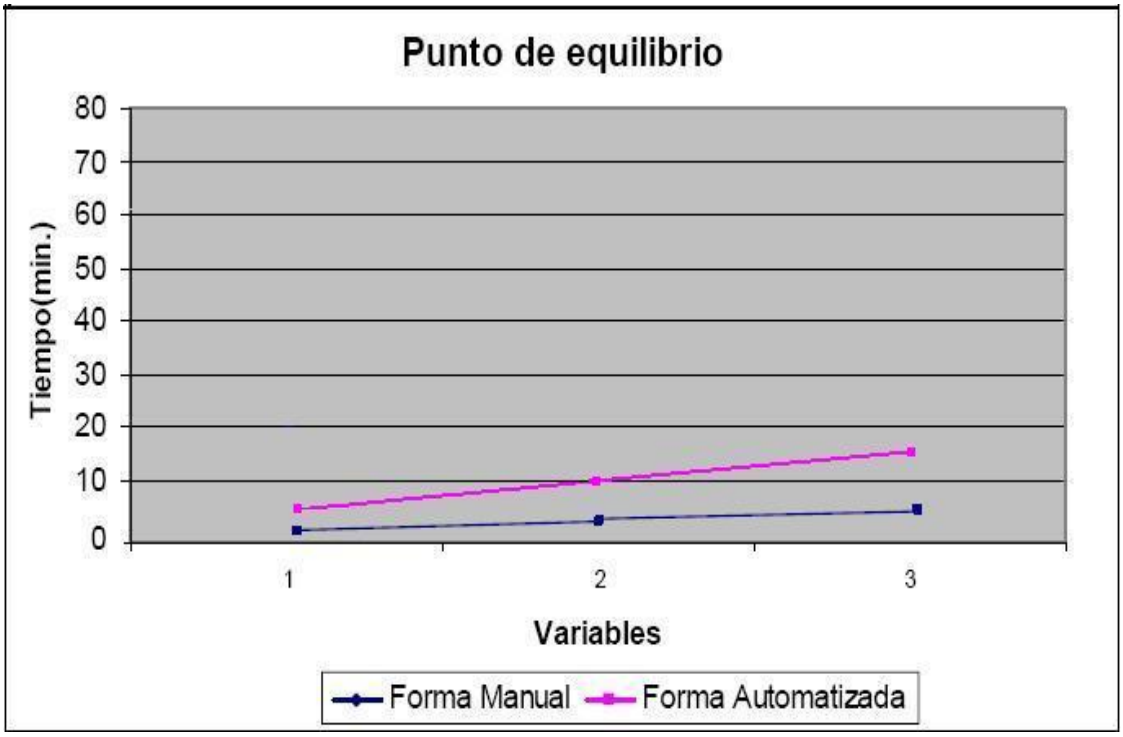

Figura 3: Gráfico de punto de equilibrio de soluciones

5.4 Conclusiones

En este capítulo se realizó el estudio de factibilidad mediante La Metodología Costo Efectividad (Beneficio), se analizó los efectos económicos, los beneficios y costos intangibles, así como se calculó el costo de ejecución del proyecto mediante la ficha de costo arrojando como resultado $ 60.78 CUC y $ 214.69 MN demostrándose la factibilidad del proyecto.

CONCLUSIONES GENERALES

Durante la realización de este trabajo se dio cumplimiento a las tareas concebidas para el desarrollo de la aplicación dando así desempeño al objetivo general del trabajo. Se hizo un análisis detallado de las diferentes metodologías ágiles para la elaboración del sistema, teniendo como resultado el uso de las metodologías SXP en las que se les dio cumplimiento a las Historias de Usuarios del sistema, se realizaron las tarjetas CRC, las tareas de ingenierías para cada Historia de Usuario y las pruebas de aceptación realizadas por el cliente. Luego del análisis de factibilidad podemos concluir que el trabajo responde como propuesta de solución al problema planteado. A demás el sistema se encuentra disponible para ser usado.

RECOMENDACIONES

- Utilizar el sistema de una forma óptima en el Departamento de Informática del Instituto Superior Minero Metalúrgico de Moa.
- Al Departamento de Informática que amplíe el alcance del sistema, incluyendo una nueva versión el cual abarque las diferentes carreras del Instituto Superior Minero Metalúrgico de Moa (ISMMM).

BIBLIOGRAFIAS

1. Díaz Cobas, Walberto, Implementación del módulo Caja del Sistema Integral de Gestión CEDRUX, Trabajo de Diploma, UCI, 2009, 13.

2. Etapa: Pruebas, 2008. [Disponible en: http://lsi.ugr.es /~arroyo/inndoc/doc/pruebas/pruebas_d.php]

3. García Avilés, Meyquel, Implementación del Módulo de Estimación de Costos del ERP Nazim, Trabajo de Diploma, ISMM, 2008, 51.

4. Mercedes Matos M. "ESTRATEGIA PARA LA FORMACIÓN INTEGRAL", SEDE Universitaria Municipal Camaguey, CIGET, 2004.

5. Pino García, Susel, Propuesta de un expediente, para los proyecto productivos del Polo de Software Libre, Trabajo de Diploma, UCI, 2008, 55.

6. Subiros Muñoz, Dariel Raúl, Desarrollo de una interfaz gráfica de usuario para el preprocesador meteorológico AERMET, Trabajo de Diploma, ISMM, 2009, 44.

7. Sitio oficial metodología XP http://www.extremeprogramming.org (Consultado: 24/enero/2010).

8. Sitio oficial de MySQL http://www.mysql.org. (Consultado: 24/enero/2010).

9. Jeffries, Canos Letelier Penades, Ferrer, C. "Extreme Programming", Addison-Wesley. 2001.

Wikipedia interactiva 2009 http://es.wikipedia.org/wiki/Scrum (Consultado: 24/enero/2010).

10. Lamas Luperón, Lisandro Manuel, Sistema de Control de Estudiantes Becados para la Residencia Estudiantil en el ISMM, Trabajo de Diploma, ISMM, 2009, 13.

11. Roger S. Pressman. Un enfoque práctico. Ciudad Madrid, editorial Mc Graw Hill, 2002. 640 páginas

12. Web Oficial de PHP. http://www.php.net. (Consulta: 08/febrero/2010).

GLOSARIO DE TERMINOS

Estándares: Es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad.

Herramientas: Son los ambientes de apoyo necesario para automatizar las prácticas de Ingeniería de Software.

Iteraciones: En el contexto de un proyecto se refieren a la técnica de desarrollar y entregar componentes incrementales de funcionalidades de un negocio. Una iteración resulta en uno o más paquetes atómicos y completos del trabajo del proyecto que pueda realizar alguna función tangible del negocio. Múltiples iteraciones contribuyen a crear un producto completamente integrado.

XP(Programación Extrema): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

Scrum: Define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones y la segunda característica importante son las reuniones a lo largo proyecto.

UML: Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software, se usa para detallar los artefactos en el sistema, para documentar y construir.

Software: Es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo.

Metodología Ágil: Nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.

Testing: Proceso de pruebas usado 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.

Metodologías de Desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

Ajax: Unión de varias tecnologías que juntas pueden lograr cosas realmente impresionantes.

XML: Sigla en inglés de Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas, una manera de definir lenguajes para diferentes necesidades.

HTML: Siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas Web.

XHTML: Reformulación del lenguaje HTML utilizado para crear páginas Web.

CCS: hojas de estilo en cascada (Cascading Style Sheets,) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML.

Release: Versión candidata definitiva de un producto de software y se refiere a un producto final, preparado para lanzarse como versión definitiva a menos que aparezcan errores que lo impidan.

HU: Historias de Usuario

Interfaz de Usuario: Es la parte de una aplicación que se encarga de interactuar con el usuario.

PL: Prácticas Laborales

UCI: Universidad de las Ciencias Informáticas

ANEXOS

Tabla 42: Plantilla Concepción del Sistema

1. **Clasificación del proyecto.**

   Desarrollo Web

2. **Tipo de proyecto.**

   Nacional

3. **Resumen:**

   Pequeña descripción de lo que trata el documento.

   Palabras claves:

4. **Surgimiento**

   Las razones por las cuales surge el producto.

5. **¿Qué es?**

   Descripción del sistema a desarrollar

6. **Metodología a utilizar**

   Descripción de la metodología a utilizar

7. **Roles**

Rol	Nombre y Apellidos
Gerente	
Cliente	
Programador	
Analista	
Diseñador	
Encargado de Pruebas	

8. **Misión**

   Misión que comprende el sistema

9. **Visión**

   Aspiraciones futuras para el sistema a desarrollar

10. **Herramientas utilizadas**

    Herramientas a utilizar para la realización del sistema

Tabla 43: Plantilla Modelo de HU del negocio

1. **Actores del negocio**

Se especifican todos los actores del negocio y se les asocia una simple descripción de cada uno de ellos.

Actor	Descripción

2. Trabajadores del negocio

Se especifican todos los actores del negocio y se les asocia una simple descripción de cada uno de ellos.

Trabajadores	Descripción

3. Diagrama de HU del Negocio

Tabla 44: Plantilla Lista de Reserva del Producto

Prioridad	Ítem *	Descripción	Estimado	Estimado por
Muy Alta				
	Números en secuencia según la cantidad de requerimiento.	Nombre de los requerimientos, ordenados según la prioridad de implementación, ubicados en Muy Alta, Alta, Media, Baja (aparecen los requerimientos de menor complejidad además de los requerimientos no funcionales del sistema a desarrollar.)	Estimación de cada uno de los requerimientos para su implementación en semanas.	Iniciales del rol que hizo la estimación del requerimiento.
Alta				
Media				
Baja				

Tabla 45: Plantilla Historia de usuario

Historia de Usuario	
Número: [Número de la historia]	**Nombre Historia de Usuario:** [Nombre que identifica la historia.]
Modificación de Historia de Usuario Número: [Cantidad de modificaciones que se le ha realizado a la historia de usuario (de no tener modificaciones se pone ninguna, sino la cantidad de veces que ha sido modificada).]	
Usuario: [Programador responsable de su implementación]	**Iteración Asignada:** [Que iteración se desarrollará. (Según su importancia)]
Prioridad en Negocio: [Prioridad puede ser Alta, Media o Baja(Según Cliente)]	**Puntos Estimados:** [Tiempo en semanas que se le asignará. (Estimado)]
Riesgo en Desarrollo: [Riesgo puede ser Alto, Medio o Bajo (Según Programadores)]	**Puntos Reales:** [Tiempo real dedicado a la realización de la HU en semanas.]
Descripción: [Breve descripción del proceso que define la historia.]	
Observaciones: [Alguna acotación importante de señalar acerca de la historia.]	
Prototipo de interface: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla 46: Plantilla Lista de riesgos

Riesgo	Tipos de riesgos	Impacto	Descripción	Probabilidad	Efectos	Mitigación del riesgo
[Enunciado del riesgo.]	[Los tipos de riesgos pueden ser: Tecnológico, Personal, Organización, Herramientas, Requerimientos, Estimación.]	[Lista de impactos en el proyecto o producto.]	[Breve descripción del riesgo.]	[La probabilidad puede ser: Muy alta, Alta, Media, Baja. Similar a la clasificación de la prioridad en las funcionalidades de la LRP]	[Los efectos pueden ser: Catastrófico, Serias, Tolerable, Insignificante.]	[Describir como se puede evitar el riesgo.]

Tabla 47: Plantilla Modelo de diseño

1. Diagrama de Paquetes

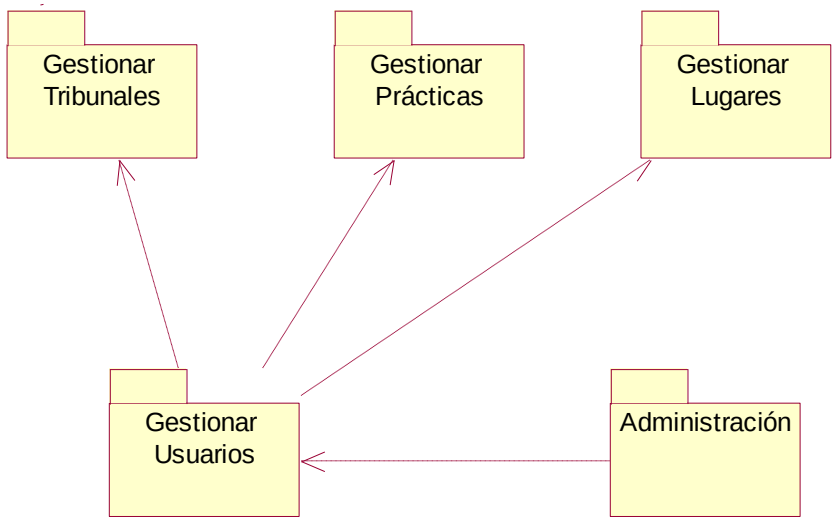

Diagrama de Paquetes.

Observaciones

- <<Observación 1>>
- <<Observación 2>>

Tabla 48: Plantilla Cronograma de producción

No	Hito	Descripción	Inicio	Fin	% ejec.	Ejecutor
[número consecutivo]	[Actividad planificada para esa iteración.]	[Breve descripción de la actividad a desarrollar]	[fecha de inicio estimada]	[fecha de fin estimada]	[% en que se encuentra realizada]	[Rol que la realiza]