UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

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

Título: Generador automático de plantillas en la *Web*.

Autores: Adrián García Ferrero.

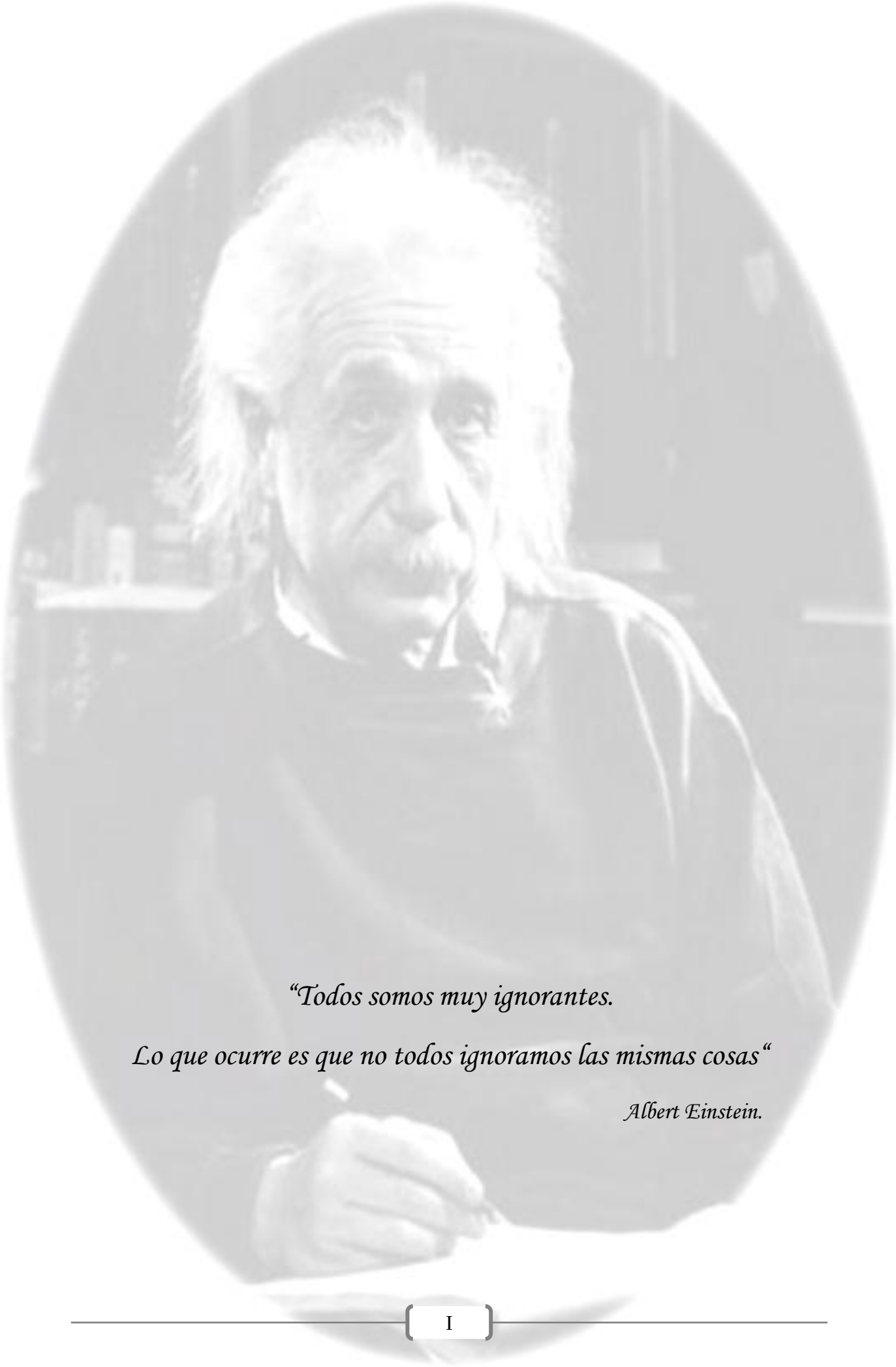
Yasiel Fernández Castellano.

Tutor: Ing. Antonio Marrero Palomino.

Cotutor: Ing. Alejandro Pablo Guerra Coello.

Ciudad de La Habana, Junio del 2010

"Año 52 de la Revolución"



"Todos somos muy ignorantes.

Lo que ocurre es que no todos ignoramos las mismas cosas"

Albert Einstein.

Declaración de autoría:

Declaramos ser autores de la presente investigación científica y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de junio del año 2010.

_____	_____
Adrián García Ferrero.	Yasiel Fernández Castellano.
Firma del autor.	Firma del autor.
_____	_____
Ing. Antonio Marrero Palomino.	Ing. Alejandro Pablo Guerra Coello.
Firma del tutor.	Firma del cotutor.

Agradecimientos:

Yasiel Fernández Castellano:

Quisiera agradecer a todas las personas que de una forma u otra se han hecho parte de este sueño hecho realidad:
A mis padres (Esteban Fernández Velásquez y Juana Castellano Compte):
Por estar siempre a mi lado, por ayudarme a levantarme ante cada caída y enseñarme que en la vida siempre hay una salida, por confiar en mí y por siempre estar a mi lado.
A mi hermano (Grasiel Fernández Castellano):
Por quererme tanto y ayudarme cada vez que lo necesité.
A mis hermanos: Esteban y Saúl Fernández por formar parte de mi vida.
A mi familia en general que siempre me han apoyado tanto y que nunca dudaron de mí:
A mis abuelos queridos (Petrona Estela Compte Rojas y Juan Castellano Favier).
A mis tías y tíos.
A mis primas.
A mis primos.
A mis sobrinitas.
A la profesora Matilde Montes De Oca Boicet por aconsejarme y estar siempre disponible para cuando lo necesitara.

A Marlon Jorge Remedios González.

A mi novia (Yalina Rodriguez Rodriguez) y compañera a lo largo de toda la carrera.
A mis fieles amigos de tantas batallas.

Adrián García Ferrero:

A la mama, la bro, la abuela, el primo, el tío y el chama.

A los que ya no están y aún se quieren.

A todos los que han formado parte de mi vida, todos importantes, unos más y otros menos, ellos lo saben, porque de un modo u otro les ha llegado mi cariño; a otros, parte de mi amor.

A los amigos de allá y los pelús de acá. A los de paso.

Al producto de María y la buena música que nos ha dado esos momentos…

Al padre y a Patry; María y Rogelio.

Adrián García Ferrero y Yasiel Fernández Castellano:

Quisiéramos agradecerle a nuestro tutor (Antonio) y a nuestro cotutor (Alejandro) por todo el tiempo que nos han dedicado, y por habernos ayudado a realizar este trabajo, sin ustedes este sueño no hubiese sido posible.

Gracias…

Dedicatoria:

Yasiel Fernández Castellano:

Dedico este Trabajo de Diploma a las personas más importantes de mi vida:
A mis padres y a mi hermanito, porque se lo merecen y ya era hora de que vieran el fruto del trabajo realizado desde hace 26 años.
Los quiero mucho y quiero que sepan que son mi razón de ser.

Adrián García Ferrero:

"A ti..."

Resumen:

El desarrollo de la informática en todo el mundo y en especial en Cuba, ha logrado que la informatización se lleve a cabo en todas las esferas de la sociedad, para lograr los procesos fundamentales de las instituciones, alcanzando mayor rapidez, control y confiabilidad del manejo de la información. Actualmente las empresas y entidades realizan actividades generando gran volumen de información, haciendo necesario la automatización y perfección de sus procesos, agilizando la manipulación y obtención de los datos.

En la Universidad de las Ciencias Informáticas (UCI) se realizan una serie de actividades como parte de la integralidad de sus estudiantes. Estas se hacen de forma engorrosa pues la captación de los estudiantes se realiza a través de hojas, o solicitando una enorme cantidad de datos mediante el correo electrónico.

El objetivo de este trabajo es presentar una solución a este problema. Siguiendo la metodología XP, se desarrolló una aplicación que genera plantillas a través de la *Web,* logrando que los usuarios creen los formularios deseados, para poder recoger y consultar la información necesaria de una actividad. Se realizó un estudio dando a conocer la tendencia actual al uso de este tipo de aplicación en el mundo, así como las herramientas más utilizadas para llevar a cabo su construcción.

Índice

Índice de tablas

Índice de figuras

Introducción

En la actualidad el mundo se ha enmarcado en el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), las empresas e instituciones automatizan y perfeccionan los procesos y actividades que realizan. Las TIC son las tecnologías que se disponen para la gestión de la información, los recursos necesarios para almacenar, proteger y manipular esa información. Actualmente esta revolución científica y tecnológica se está aplicando en todos los campos de la sociedad, se presentan cada vez más como una necesidad, es una realidad y uno de los motores principales de la sociedad actual. Mejorar y simplificar los procesos, integrar procesos internos, ahorrar tiempo y dinero a través de los sistemas de información se hace hoy tarea primordial.

Cuba, con un proyecto de desarrollo que tiene como principios la igualdad social, la participación popular y la solidaridad, ha diseñado medidas y estrategias que permitan convertir los conocimientos y las tecnologías, en instrumentos a disposición de las profundas transformaciones, logrando cada vez más eficacia y eficiencia en los procesos y por consiguiente, un aumento en la calidad de vida de los ciudadanos.

Con el triunfo de la Revolución, el gobierno se ha trazado dentro de sus objetivos, adoptar decisiones que permitan universalizar los beneficios de la llamada Sociedad de la Información y la informatización de los sectores fundamentales de la sociedad, métodos sabios y justos dado el avance tecnológico existente. Se han logrado los objetivos parcialmente, con la ayuda de las empresas productoras de software y universidades cubanas. Es por ello que surgen dentro de la Universidad de Ciencias Informáticas (UCI) muchos proyectos capaces de crear sistemas y herramientas destinados a ser aplicados en las diferentes esferas de la sociedad.

La universidad juega un papel importante en el desarrollo de la Industria Cubana del Software, y en la materialización de los proyectos asociados al programa cubano. En esta institución se realizan una serie de actividades extradocentes como parte de la integralidad de sus estudiantes, tales como el movimiento de artistas aficionados, juegos deportivos, jornadas científicas y otros. Siendo los estudiantes los protagonistas de las actividades referidas, los dirigentes de la FEU y la UJC se ven obligados a pasar por cada aula con una hoja para anotar los interesados en participar en cualquiera de estas manifestaciones y recoger todos los datos pertinentes; por ejemplo, obtener los datos de los estudiantes que participarán en el Marabana (*) se les solicita enviarlos por correo, así como cuando un profesor quiere hacer una encuesta.

Este proceso de captación de estos estudiantes interesados en cualquiera de estas actividades se hace muy engorroso en la manera que funciona, lo que crea la siguiente **situación problémica:** existe demora y desorganización en el proceso de captación de los estudiantes para actividades extradocentes; participación en copas, ya sean de programación o ingeniería de *Software*, captación para el movimiento de alumnos ayudantes, o sea, cualquier proceso de obtención de datos tanto de estudiantes como de profesores se hace de manera rudimentaria, ya sea a través de hojas, o solicitando una enorme cantidad de datos mediante el correo electrónico.

Todo lo antes expuesto demuestra la necesidad de una mejora en el proceso de obtención de datos.

Se propone la implementación de un sistema seguro y eficiente capaz de realizar la solicitud y la salva de dichos datos de manera automática, donde un usuario a través de la red acceda a un formulario (*) o plantilla, generado también vía *Web*, donde guardará los datos necesarios que defina el interesado de manera práctica y sencilla a través de dicha aplicación.

Teniendo en cuenta lo anteriormente expuesto, el **problema científico** queda formulado de la siguiente manera: ¿Cómo generar plantillas para facilitar el proceso de obtención de datos a través de la *Web*?

El **objeto de estudio** lo constituyen las herramientas generadoras de plantillas en la *Web*.

Se define como **campo de acción** a los procesos y herramientas para la generación de plantillas de datos, en la UCI.

Para dar solución a la problemática planteada anteriormente se ha definido como **objetivo general** de la investigación: Desarrollar una aplicación *Web,* fácil de usar, que permita a un cliente la generación de plantillas para la obtención y el almacenamiento de los datos solicitados.

Derivándose los siguientes **objetivos específicos:**

- Permitir almacenar los datos entrados por un usuario a una plantilla generada por un cliente, vía *Web*.
- Realizar un producto genérico que no sólo sea aplicable a la universidad, también a cualquier necesidad de cualquier cliente que se ajuste a los servicios que brinde la aplicación.

Para cumplir los objetivos trazados se realizarán las siguientes **tareas:**

- Realizar un estudio de tecnologías y herramientas para la realización de plantillas y formularios en la *Web* (*) de forma dinámica.
- Seleccionar las herramientas necesarias para el desarrollo del generador automático de plantillas.
- Desarrollar el ciclo completo de *Software*.
- Evaluar los resultados obtenidos.

Se plantea como **hipótesis:** con la implementación del generador automático de plantillas a través de la *Web*, se facilitará el proceso de obtención de los datos de un usuario interesado en una actividad definida.

Variable independiente: Generador automático de plantillas en la *Web*.

Variable dependiente: Proceso de obtención de los datos de un usuario.

Operacionalización de las variables:

Variable Conceptual	Dimensión	Indicadores	UM
Generador automático de plantillas en la *Web*.	Factibilidad	Tiempo de desarrollo	Largo Moderado Breve
		Costo	Costoso Moderado Barato
		Esfuerzo	Alto Moderado Despreciable
Proceso de obtención de los datos de un usuario.	Mejor rendimiento	Complejidad	Alta Media Baja
		Control	Bueno Malo
		Organización del trabajo	Bueno Malo
		Tiempo de Ejecución	Rápido Medio Lento

Tabla 1. Operacionalización de las variables.

En la investigación se usará un sistema conformado por métodos teóricos y empíricos. En relación con los **métodos teóricos** se citan:

- Analítico-Sintético: Se realizará un estudio de los requerimientos de la aplicación permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.
- Análisis Histórico-Lógico: Partiendo de la observación y análisis de los antecedentes del objeto de estudio se dio inicio a esta investigación y se realizará un estudio de la existencia de otros sistemas de similar funcionamiento, si es que existen.

Por otra parte, se utilizarán los **métodos empíricos** siguientes:

- Encuestas y entrevistas: Se realizarán entrevistas a dirigentes de la FEU y UJC, para generalizar los componentes que deben necesitarse en la aplicación.
- Observación: Se percibirán los distintos casos en que se realice un proceso de obtención de datos y en función de esto se creará el sistema.

Se obtendrá como **posible resultado:** El ciclo completo de desarrollo de una aplicación generadora de plantillas, logrando que los usuarios creen los formularios deseados, para poder recoger y consultar la información necesaria de una actividad.

La investigación está estructurada en:

- Capítulo 1: Fundamentación Teórica.

  Incluye el estado del arte de los sistemas de similar funcionamiento, si es que existen, en el ámbito internacional, nacional y en la UCI, y una caracterización de las herramientas y metodologías a utilizar para darle solución a la propuesta.

- Capítulo 2: Descripción y análisis de la solución propuesta. Características del sistema.

  Describe la solución de la propuesta, mostrando la especificación de los requisitos funcionales y no funcionales, definiéndose las historias de usuarios del sistema, se confecciona un plan de entrega en conjunto con un plan de iteraciones. Aborda los detalles relacionados con el diseño del generador de plantillas, y se especifican funcionalidades orientadas a responsabilidades a

través de las tarjetas CRC (clase, responsabilidad y colaboración) para facilitar la implementación de la propuesta.

Capítulo 3: Desarrollo y pruebas.

Describe cómo se desarrolla la aplicación, con las tareas asignada por iteración para darle solución a las historias de usuario. Además para confirmar que al finalizar cada iteración se obtuvo el resultado esperado, se le hacen pruebas de aceptación a las historias de usuario.

- Capítulo 4: Análisis de la factibilidad del sistema.

  Se hace un análisis de la factibilidad del sistema, el cual está guiado por del tiempo, esfuerzo y costos necesarios para el desarrollo del sistema, enfocándose en los beneficios de este.

Nota: La palabra que se encuentre seguida del operador (*) es para identificar que se encuentra en el glosario de términos.

Capítulo 1. Fundamentación teórica

Introducción

En el presente capítulo se abordan aspectos de las diferentes categorías de *Software*, especificando los sistemas de gestión de plantillas, así como elementos que se deben tener en cuenta para su funcionamiento. También las características de las tecnologías, herramientas y metodología utilizadas para el desarrollo del sistema propuesto.

1.1 Marco conceptual

Un sistema de información es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio.

Un sistema de información (SI) es un conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su posterior uso, generados para cubrir una necesidad (objetivo). Dichos elementos formarán parte de alguna de estas categorías:

- Personas.
- Datos.
- Actividades o técnicas de trabajo.
- Recursos materiales en general (típicamente recursos informáticos y de comunicación, aunque no tienen por qué ser de este tipo obligatoriamente).

Todos estos elementos interactúan entre sí para procesar los datos (incluyendo procesos manuales y automáticos) dando lugar a información más elaborada y distribuyéndola de la manera más adecuada posible en una determinada organización en función de sus objetivos.

Normalmente el término es usado de manera errónea como sinónimo de *sistema de información informático*, en parte porque en la mayoría de los casos los recursos materiales de un sistema de información están constituidos casi en su totalidad por sistemas informáticos, pero siendo estrictos, un sistema de información no tiene por qué disponer de dichos recursos (aunque en la práctica esto no suela ocurrir). Se podría decir entonces que los sistemas de información informáticos son una subclase o un subconjunto de los sistemas de información en general.

Formulario

Un formulario es una plantilla o página que permite mostrar y solicitar información, presentando espacios vacíos que pueden ser llenados siguiendo algún objetivo. Es utilizado en informática para referirse al conjunto de campos solicitados por un determinado programa, los cuales se almacenarán para su posterior uso o manipulación.

1.2 Estado del arte

Los generadores automáticos de formularios on-line son herramientas en la *Web*, creadas para que usuarios medios puedan crear sus formularios de manera sencilla y sin tener que escribir código. Con estas herramientas, un usuario puede crear en un breve espacio de tiempo un formulario, para luego agregarlo a una página *Web* o un blog.

En el mundo existen varios generadores automáticos de formularios en línea, herramientas en la red fáciles de usar, creados para usuarios de nivel medio sin muchos conocimientos de programación que necesiten agregar un formulario a su sitio *Web* o blog, estos permiten crear formularios de manera rápida y sencilla, sin tener que escribir código, invirtiendo mucho menos tiempo.

Web* Form Factory (**Fábrica de formularios *Web)** es una aplicación de código abierto que hace la función de **generador en línea de formularios** para sitios *Web*. Se encarga además de generar el código necesario para enlazar el formulario con una base de datos de manera completamente automática, lo que se traduce en un importante ahorro de tiempo a la hora de desarrollar una *Web*. (1)

PHP FormMailGenerator (Generador de formularios de correo**)** es otro generador automático de formularios. Los formularios creados con este generador en línea cuentan con las siguientes características:

- Número ilimitado de campos de formulario estándar, también campos de tarjeta de crédito y campos para adjuntar / subir archivos.
- Validación de los campos introducidos por el usuario (campos requeridos, número de tarjetas de crédito y fecha de expiración, subida de archivos, etc.)
- Envío de correos electrónicos personalizados y soporte de archivos adjuntos.
- Envío de correos de respuesta automática personalizados y copia para el usuario.
- Posibilidad de guardar los datos enviados vía formulario como un fichero final.
- Página de agradecimiento personalizada o página de redirección.

- Anti email spider (Previene la recolección de emails por parte de ambos).
- Anti-Spam (*) mediante el uso de imágenes de seguridad.
- Guarda un log de todo el tráfico. (Para verificar si existe o no actividad relacionada con spam).
- Panel de Administrador para ver y descargar los datos del formulario del usuario. (2)

***Web*siteContactFormGenerator** (Sitio *Web* Contacto generador de formularios) es una aplicación en línea gratuita que permite crear el código necesario para incorporar formularios de contacto en sitios *Web* basados en ASP, PHP o Perl y sin necesidad de tener conocimiento alguno de programación. También basta con seguir los pasos indicados en la página para **crear formularios** de contacto **en línea**, el proceso a realizar es muy sencillo e intuitivo. (3)

Wufoo es un generador en línea de formularios HTML que permite crear formularios de contacto, encuestas en línea, etc. para que puedas recopilar información, registros y pagos en línea sin necesidad de escribir una sola línea de código. (4)

Estos sistemas para la creación de formularios presentan como característica principal, permitir crear formularios sin necesidad de escribir código, pero se encuentran alojados en servidores fuera del país, no crean la estructura necesaria para guardar los datos de manera persistente, por lo que no posibilitan el almacenamiento de la información, sólo se usan para agregar formularios a otras aplicaciones y no quedan guardados de manera funcional.

1.3 Estudio de las metodologías de desarrollo

Una metodología es necesaria para el proceso de desarrollo de *Software* debido a que ayuda a construir un *Software* con calidad, en el tiempo y costo esperado.

1.3.1 Proceso Unificado de Desarrollo de *Software* (RUP)

El Proceso Unificado de Desarrollo de Software (RUP, Rational Unified Process) “se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes”.

Esta metodología utiliza el Lenguaje Unificado de Modelado (UML, Unified Modeling Language) para modelar todos los esquemas de un sistema de *Software*. RUP posee tres características fundamentales que lo distinguen del resto de las metodologías:

- Dirigido por casos de uso: A partir de la identificación de los casos de uso todos los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.
- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Iterativo-incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

RUP consta de cuatro fases: Inicio, Elaboración, Construcción y Transición y nueve disciplinas a realizar en cada fase:

- Modelado del negocio.
- Análisis de requerimientos.
- Análisis y diseño.
- Implementación.
- Pruebas.
- Despliegue.
- Gestión de configuración y cambios.
- Gestión del proyecto.
- Gestión del entorno.

(5)

1.3.2 Extreme Programing (XP)

Los objetivos de XP (Programación extrema) son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el *Software* que él necesita y cuando lo necesita. Por tanto, se debe

responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final del ciclo de la programación.

El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del *Software*.

La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica.

El objetivo de la XP es generar versiones de la aplicación tan pequeñas como sea posible, pero que proporcionen un valor adicional desde el punto de vista del negocio, basándose en los siguientes principios:

- Los individuos (*) e interacciones son más importantes que los procesos y herramientas.

  Las personas son el principal factor de éxito de un proyecto de *Software*. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que este configure su propio entorno de desarrollo en base a sus necesidades.

- Un *Software* (*) que funcione es más importante que documentación exhaustiva.

  La regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental.

- La colaboración con el cliente es más importante que la negociación de contratos.

  Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

- La respuesta ante el cambio es más importante que el seguimiento de un plan.

  La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Las características anteriormente expuestas justifican la selección de esta metodología para ser usada en el desarrollo de la propuesta. (6)

1.4 Arquitectura

La arquitectura es el arte de crear una estructura sólida para el desarrollo de un sistema informático o de un *Software*.

1.4.1 Arquitectura Multicapa

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica facilitando una manera muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada.

El estilo de arquitectura basado en capas se identifica por las siguientes características:

- Describe la descomposición de servicios, de forma que, la mayoría de la interacción ocurre solamente entre capas vecinas.
- Las capas de una aplicación pueden residir en la misma máquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas).
- Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas.
- Este modelo ha sido descrito como una “pirámide invertida de re-uso” donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella.
-

1.4.2 Arquitectura de Tres Capas

Un ejemplo de multicapa lo constituye la Arquitectura de Tres Capas, que contiene funcionalidades relacionadas con la interfaz (*) de usuario (capa de presentación), otras encargadas del procesamiento de reglas de negocio (capa de negocio), y las relacionadas con el acceso a datos(capa de acceso a datos).

Beneficios de la Arquitectura Tres capas:

- **Escalabilidad**. (*) Como las capas están basadas en diferentes máquinas, el escalamiento de la aplicación hacia afuera es razonablemente sencillo.

- **Disponibilidad**. Las aplicaciones pueden aprovechar la arquitectura modular de los sistemas habilitados usando componentes que escalan fácilmente lo que incrementa la disponibilidad.
- **Mejoras en las posibilidades de mantenimiento**. Debido a que cada capa es independiente de la otra, los cambios o actualizaciones pueden ser realizados sin afectar la aplicación como un todo.
- **Flexibilidad**. Como cada capa puede ser manejada y escalada de forma independiente, la flexibilidad se incrementa.
- **Carga de base de datos reducida**. El servidor de base de datos se beneficia de menos conexiones.
- **Independencia de plataforma**. Las PCs del cliente pueden estar en plataformas diferentes. Así esto no exige tener una versión de la plataforma específica de *Software* a cada cliente.
- **Facilidad de Actualización**. También permite actualizaciones muy fáciles de las versiones más nuevas del producto.

(7)

1.5 Lenguajes

Un lenguaje de programación es la notación para la descripción precisa de algoritmos o programas informáticos. Son el conjunto de instrucciones que permiten al programador pensar claramente sobre la complejidad del problema a resolver, de manera que pueda ordenarlas convenientemente para la creación de un programa ejecutable por la computadora.

La programación *Web*, parte de las siglas WWW, que significan World Wide *Web* o telaraña mundial. Para realizar una página con la programación *Web*, se deben tener claros tres conceptos fundamentales, los cuales son: el URL o (Localizadores Uniformes de Recursos), es un sistema con el cual se localiza un recurso dentro de la red, este recurso puede ser una página *Web*, un servicio o incluso una aplicación. En resumen el URL no es más que un nombre, que identifica una computadora, dentro de esa computadora un archivo que indica el camino al recurso que se solicita. A continuación se describen varios lenguajes de programación que se utilizan para la construcción de sitios *Web*.(8)

1.5.1 Lenguajes de programación del lado del cliente

HTML

El Lenguaje de Marcado de Hipertexto (HyperText Markup Language), es el lenguaje que se utiliza para crear las páginas *Web*. Este lenguaje indica a los navegadores cómo deben mostrar el contenido de una página *Web*.

El lenguaje HTML contiene dos partes:

- El contenido, que es el texto que se verá en la pantalla de un ordenador,
- Las etiquetas y atributos que estructuran el texto de la página *Web* en encabezados, párrafos, listas, enlaces, etc. y normalmente no se muestra en pantalla.

Lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces (*hyperlinks*) que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia (gráficos, sonido...) La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc) así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho hipertexto se realice por un programa especializado.

(9)

CSS

Es el acrónimo de Cascading Style Sheets (es decir, hojas de estilo en cascada). Es un mecanismo o lenguaje de estilo que define la presentación de los documentos HTML. Por ejemplo, CSS abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas. (10)

JavaScript

Es un lenguaje de programación que se utiliza principalmente para crear páginas *Web* dinámicas.

Una página *Web* dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (11)

1.5.2 Lenguajes de programación del lado del servidor

JAVA

El lenguaje para la programación Java (*), es un lenguaje orientado a objeto de plataforma independiente. Fue desarrollado por la compañía Sun Microsystems, con la idea original de usarlo para la creación de páginas *Web*. Esta programación tiene muchas similitudes con el lenguaje C y C++. El código generado por el compilador Java es independiente de la arquitectura y podría ejecutarse en un entorno UNIX, Mac o Windows. Esto es posible ya que el código generado por el compilador se ejecuta mediante una máquina virtual y por el procesador del ordenador directamente, lo que permite que los Applets (que son aplicaciones especiales que se ejecutan dentro de un navegador al ser cargada una página HTML(*) en un servidor *Web*. Por lo general los Applets son programas pequeños y de propósitos específicos) de una *Web* pueda ejecutarlos cualquier máquina que se conecte a ella independientemente de qué sistema operativo emplee, siempre y cuando el ordenador en cuestión tenga instalada una máquina virtual de Java. (12)

PERL

El lenguaje Perl toma su nombre de PracticalExtraction and ReportLanguage(Extracción Práctica e Informe de idiomas) y fue ideado por Larry Wall. La primera versión sale en el año de 1987. Originalmente Larry Wall lo describe como un excelente lenguaje optimizado para leer archivos de texto, extraer información de esos archivos y crear reportes basados en esa información, combinando lo mejor de los lenguajes C, sed, awk y sh.

Perl se considera un lenguaje interpretado, es decir, no es necesaria una previa compilación para poder ejecutarse, lo único que se necesita es darle al intérprete Perl, el código que se quiere ejecutar. Desde sus inicios Perl ha avanzado mucho y ahora se encuentra en la versión 5 con la versión 6 en pleno desarrollo. Cabe mencionar que Perl hoy en día es usado para una gran variedad de cosas, desde avanzados programas de seguridad hasta sencillos CGIs para administrar formularios. Una de las posibilidades que brinda Perl es la existencia de CPAN, un directorio de módulos que podemos integrar a nuestros scripts (*). (13)

PHP

PHP (*) es uno de los lenguajes del lado del servidor más extendidos en la *Web*. Nacido en 1994, se trata de un lenguaje de creación relativamente creciente que ha tenido una gran aceptación en la

comunidad de *Web*masters debido sobre todo a la potencia y simplicidad que lo caracterizan. PHP permite embeber fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas programados íntegramente en un lenguaje distinto al HTML. Por otra parte, PHP ofrece un sinfín de funciones para la explotación de bases de datos de una manera llana, sin complicaciones (Super hosting). PHP es el acrónimo de HipertextPreprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con gran variedad de librería de funciones y mucha documentación.

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor *Web*, justo antes de que se envíe la página al cliente a través de Internet al cliente. 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 PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.(*)

Luego del análisis de los lenguajes anteriores se decide utilizar para el desarrollo del sistema el lenguaje PHP, más específico PHP 5, que presenta ciertas ventajas entre las que se encuentran:

- Es un lenguaje multiplataforma.
- Tiene la capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. Además de ser ampliamente usado en la universidad e impartido en el programa docente de la facultad.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- 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.
- Está basado en el lenguaje C++ y la sintaxis usada es muy similar a C/C++ el cual es considerado aún el mejor lenguaje de programación por muchos programadores. (14)

1.5.3 Lenguaje de Modelado

El lenguaje de modelado es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de *Software*.

UML

Es el lenguaje de modelado de sistemas de *Software* más conocido y utilizado en la actualidad; está respaldado por el OMG (Grupo de Dirección de Objeto). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (15)

1.6 Servidores *Web*

Un servidor *Web* es un programa que permite desarrollar y administrar sistemas de hospedaje de sitios. Entre sus funciones está la de atender las solicitudes que se piden vía *Web*, a través de un protocolo http o https, dependiendo del tipo de la petición. El servidor *Web* buscará una página *Web* o bien ejecutará un programa en el servidor que devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición. (16)

1.6.1 Apache

El servidor *Web* Apache es un servidor *Web* gratuito desarrollado por el Apache Server Project (Proyecto Servidor Apache) cuyo objetivo es la creación de un servidor *Web* fiable, eficiente y fácilmente extensible con código fuente abierto gratuito. Este proyecto es conjuntamente manejado por un grupo de voluntarios localizados alrededor del mundo que a través de Internet planean y desarrollan el servidor y la documentación relacionada con este. Estos voluntarios son conocidos como el grupo Apache.

Ventajas:

- Corre en varios sistemas operativos, lo que lo hace prácticamente universal.
- Apache es gratuito de código fuente abierto. El hecho de ser gratuito es importante pero no tanto como que se trate de código fuente abierto, lo que le da una transparencia a este

Software de manera que si se desea se puede ver que es lo que se está instalando como servidor, se puede saber, sin ningún secreto, sin ninguna puerta trasera.

- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables a este
- Apache trabaja con varios lenguajes entre los que se encuentran: PHP, PERL y Java, teniendo todo el soporte que se necesita para las páginas dinámicas.
- Apache permite personalizar la respuesta ante los posibles errores que puedan ocurrir en el servidor. Es posible configurar Apache para que ejecute un determinado *script* cuando ocurra un error en concreto.
- Posee varias opciones de configuración para la creación y gestión de *logs*. Apache permite la creación de ficheros de log a medida del administrador, de este modo, se puede tener un mayor control sobre lo que sucede en su servidor. (17)

1.6.2 Servidor Lighttpd

Lighttpd es un servidor seguro y rápido que respeta estándares y consume muy pocos recursos, todo ello con un código limpio y elegante.

Es ideal para ser usado en entornos donde la carga es máxima y se requieren respuestas rápidas y alta escalabilidad. Anurix (*) lo utiliza como único servidor *Web* para contenido dinámico, generalmente sobre estaciones x86 o sparc64 con sistemas NetBSD.

Algunas características básicas destacables que posee Lighttpd son:

- Soporte de varios lenguajes de programación como PHP.
- Admite certificados SSL y por lo tanto puede servir https(conexión segura).
- Autentificación con htpasswd, LDAP o MySQL.
- Tiene un módulo de reescritura y de redirección de URLs.
- Genera estadísticas mediante rrdtool.
- Permite Módulos Externos.

(18)

1.6.3 IIS (Internet information server)

Es un servidor *Web* con una serie de servicios para los ordenadores que funcionan con Windows. Proporciona las herramientas y funciones necesarias para administrar de forma sencilla un servidor *Web* seguro.

- Security Features_IIS_SP1_Ops: describe nuevas características diseñadas para proteger los sitios *Web*, los datos y el servidor IIS.
- Características de rendimiento: describe las características de rendimiento y de escalabilidad.
- Tecnologías de aplicaciones *Web*: Describe la implementación de IIS 6.0 de Microsoft® ASP.NET, ASP y el proveedor ADSI.
- Herramientas administrativas y características: describe los cambios en la metabase y las nuevas herramientas administrativas.
- Estándares de Internet: describe la compatibilidad con los protocolos estándar *Web* y de Internet.
- Características de Protocolo Internet versión 6: Describe las funciones de IIS 6.0 compatibles con el Protocolo Internet versión 6 (IPv6). (19)

1.6.4 Servidor Thttpd

Es un servidor que utiliza los requerimientos mínimos de un servidor HTTP, es *Software* libre y diseñado para la simplicidad, una huella pequeña de la ejecución y la velocidad. (20)

- Simple, porque esto maneja sólo el mínimo necesario para poner en práctica el protocolo HTTP, algunas veces un poco más que el mínimo.
- Pequeño, porque tiene un periodo de explotación corto, ya que esto no se divide en dos partes y es muy cuidadoso sobre la asignación de memoria.
- Portátil, porque se compila limpiamente sobre la mayoría de sistemas operativos.
- Rápido, porque en el empleo típico es, sobre todo, más rápido que los mejores servidores “destacados” (APACHE), y bajo la carga extrema es mucho más rápido.

- Seguro, porque este se extiende a grandes longitudes para proteger el servidor *Web* contra ataques de otros sitios.

(21)

Luego de hacerse un profundo análisis de Apache, Lighttpd, IIS y Thttpd se ha decidido que para el entorno a desarrollar el mejor servidor *Web* es el Apache por las siguientes razones: La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

1.7 Sistemas Gestores de Base de datos

Los Sistemas Gestores de Bases de Datos (SGDB) tienen como objetivo servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, con el propósito de manipular los datos que serán utilizados en una organización. **(22)**

1.6.1 MySQL

MySQL es una base de datos relacional, almacena datos en tablas separadas en lugar de poner todos los datos en un gran almacén. Esto añade velocidad y flexibilidad. La parte SQL de "MySQL" se refiere a "Structured Query Language". SQL es el lenguaje estandarizado más común para acceder a bases de datos y está definido por el estándar ANSI/ISO SQL.

El *Software* MySQL tiene una doble licencia. Los usuarios pueden elegir entre usar el *Software* MySQL como un producto Open Source bajo los términos de la licencia GNU General Public License o pueden adquirir una licencia comercial estándar de MySQL AB.

Características.

- Interioridades y portabilidad
  - Probado con un amplio rango de compiladores diferentes.
  - Funciona en diferentes plataformas.
  - APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
  - Pueden usarse fácilmente múltiples CPU si están disponibles.

- Proporciona sistemas de almacenamientos transaccionales y no transaccionales.
- Usa tablas en disco muy rápidas con compresión de índice.
- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

➢ Seguridad

Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

➢ Escalabilidad y límites

- Soporte a grandes bases de datos. Se usa MySQL Server con bases de datos que contienen 50 millones de registros. También se conoce de usuarios que usan MySQL Server con 60.000 tablas y cerca de 5.000.000.000.000 de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

(23)

1.7.1 Firebird

Firebird es una base de datos relacional que ofrece muchas características de SQL ANSI estándar y que funciona en Linux, Windows, MacOSX y una variedad de plataformas UNIX. Ofrece una concurrencia excelente, alto rendimiento y un poderoso lenguaje de procedimientos almacenados y disparadores. Ha estado usándose en producción bajo varios nombres desde 1981.

Sus principales características son:

- Completo soporte para procedimientos almacenados y disparadores.
- Integridad referencial.
- Bajo consumo de recursos.
- Completo lenguaje interno para procedimientos almacenados.
- Soporte para Funciones Externas.
- Prácticamente no requiere configuración.
- Gran comunidad y muchos sitios donde se puede encontrar excelente soporte gratuito.
- Versión incrustada– ideal para crear catálogos en CDROM, versiones mono usuario, de evaluación o portátiles de las aplicaciones.
- Muchas formas de acceso a la base de datos.
- Soporte nativo para todos los principales sistemas operativos, incluyendo Windows, Linux, Solaris, MacOS.
- Copias de seguridad incrementales.
- Tablas Temporales.

(24)

1.7.2 PostgreSQL

El Sistema Gestor de Bases de Datos Relacionales Orientada a Objetos comnocido como PostgreSQL(*), está derivado del paquete Postgre escrito en Berkeley. Con cerca de una década de desarrollo, PostgreSQL es el gestor de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación(incluyendo C, C++Java, perl, tel y python).

Características.

- Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Comunidades muy activas, varias comunidades en castellano.
- Bajo "Costo de Propiedad Total" (TCO) y rápido "Retorno de la Inversión Inicial" (ROI).
- Altamente adaptable a las necesidades del cliente.
- Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, etc.
- Soporte de todas las características de una base de datos profesional (triggers, procedimientos de almacenados, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.)
- Soporte de protocolo de comunicación encriptado por SSL.
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.
- Utilidades para limpieza de la base de datos.
- Utilidades para análisis y optimización de consultas.
- Almacenaje especial para tipos de datos grandes (TOAST).

(25)

Gestor seleccionado: MySQL

Elementos que lo hacen el elegido, además de que tiene licencia de código abierto, es uno de los gestores de base de datos más usados en el *Software* libre así como en la universidad, es de fácil uso y muy rápido.

1.8 Tecnologías

Tecnología es el conjunto de conocimientos que permiten construir un *Software* o sistemas informáticos.

1.8.1 AJAX (Asynchronous JavaScript and XML)

Es utilizado para comunicarse con el servidor de la aplicación, antes de AJAX no había aplicación que fuera capaz de comunicarse directamente con el servidor, con ella, el cliente y el servidor pueden comunicarse libremente.

Características de cómo funciona:

1. Alguna acción desencadena el evento, tal y como darle clic a un botón.
2. Una vez que se desencadena el evento envía una petición al servidor utilizando XML.
3. El script de lado del servidor (PHP, ASP, etc.), reconoce el pedido por medio de JavaScript, y puede tener acceso a la base de datos en caso necesario y procesar la información.
4. Vuelve a utilizar XML y el script envía la información al cliente que hizo la petición inicial.
5. Una función secundaria de JavaScript denominada *callback* captura la información y actualiza la página *Web*.

AJAX permite a otras aplicaciones *Web*, actuar como si fueran de escritorio. Es capaz de proveer funcionalidad adicional a una página *Web*, que de otra manera no podría ser posible. Es por eso que esta tecnología es escogida para el desarrollo del *Software*.

(26)

1.8.2 Marcos de trabajo:

ExtJS

Es un marco de trabajo de JavaScript muy flexible, que permite realizar de una manera muy rápida, interfaces con apariencia profesional, tiene una gran comunidad de desarrolladores y una documentación mayor cada día. Además cuenta con la licencia “Open Source” o de código abierto.

ExtJS tiene como características:

- Gran desempeño.
- Componentes de interfaz de usuario personalizables.
- Buen diseño.
- Documentación.
- Un modelo de licencia dual. Para aplicaciones *Web* comerciales hay que adquirir una licencia y para aplicaciones *Web* de código abierto, que sean compatibles con GNU GPL licencia v3, se puede utilizar la licencia gratuita. (27)

CodeIgniter.

CodeIgniter es un poderoso entorno de desarrollo de aplicaciones en PHP, que permite crear *Web*s dinámicas. Es de código abierto, muy pequeño y posee un acceso a sus librerías bien estructurado, ayuda a los desarrolladores a crear proyectos con rapidez y desde cero.

Los controladores son uno de los componentes que forman parte del modelo de programación MVC (Modelo - Vista - Controlador) que sirven como engranaje principal a la hora de crear aplicaciones *Web*.

Un controlador en CodeIgniter es un archivo que contiene el código de una clase, de programación orientada a objetos, que se coloca en un directorio específico del esquema de carpetas del sitio.

Los controladores en CodeIgniter se guardan en la carpeta "system/application/controllers/", que se encuentra dentro de los archivos de CodeIgniter.

Algunas características importantes de CodeIgniter:

- Compatible tanto para PHP4 como para PHP5.
- Magnífica documentación y enorme comunidad de desarrolladores.
- Fácil de usar, no requiere de mucho estudio para comprenderlo.
- Clases de base de datos llenas de características con soporte para varias plataformas.
- Paginación.
- Gerencia de sesión, las sesiones típicamente correrán globalmente con cada carga de la página, la clase de sesión correrá desatendida en el trasfondo.
- Ayuda de base de datos.
- Validación de la forma y de los datos.
- Seguridad y filtración XSS.
- Encriptación de datos.

(28)

1.9 Herramientas a utilizar

1.9.1 IDE Zend Studio:

Zend Studio o Zend Development Environment es un entorno de desarrollo integrado para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Junto con su contraparte ZendPlatform, son la propuesta de tecnologías Zend

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. Está disponible también una versión de Zend Studio para Eclipse.

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, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.
- Ayudas de formato de código, como la sangría automática.
- Detección de errores de sintaxis en tiempo real.
- Manual de PHP integrado.
- Cliente FTP integrado.
- Soporte para navegación en bases de datos y ejecución de consultas SQL.
- Ofrece soporte básico para lenguajes *Web* como: HTML, Java script y XML.

1.9.2 Aptana Studio

Es un entorno integrado de desarrollo para la elaboración de aplicaciones *Web* dinámicas que empleen PHP, Ruby, Ruby onRails y Python.

Su integración con dos nuevas herramientas de Aptana amplían sus posibilidades casi al infinito. Estas son AptanaJaxer, un servidor AJAX, y Aptana Cloud, un servicio de hosting complementario.

Aptana Studio posee un asistente de código que ayuda al programador en la escritura de los diferentes lenguajes, CSS y Javascript entre ellos. En el caso del HTML, puede mostrar todos los elementos pertenecientes a este lenguaje y sus propiedades. Aptana Studio contiene también información de soporte para los principales navegadores *Web*: IE, Firefox, Opera, Netscape y Safari.

Entre las funciones más destacadas de Aptana Studio se pueden mencionar:

- Asistente de código para HTML y JavaScript.
- Librerías AJAX (jQuery, prototype, scriptaculous, ExtJS, dojo, YUI y Spry entre otras).
- Herramientas para trabajo con base de datos.
- Marcado de sintaxis mediante colores.
- Compatible con extensiones para Eclipse (existen más de 1000).
- Explorador de código en forma de árbol.

- Librerías populares AJAX/JavaScript.
- Extensión de funcionalidad mediante macros y acciones.
- Visor de errores y advertencias.
- Servidor local para probar el código. (29)

1.9.3 DBDesigner

DBDesigner es un sistema totalmente visual de diseño de bases de datos, que combina características y funciones profesionales con un diseño simple, muy claro y fácil de usar, a fin de ofrecer un método efectivo para gestionar bases de datos.

Permite administrar la base de datos, diseñar tablas, hacer peticiones SQL manuales y mucho más, como ingeniería inversa en MySQL, Oracle, MSSQL y otras bases de datos ODBC, modelos XML y soporte para la función drag-and-drop.

El programa dispone además de una interfaz profesional y de detallados manuales de uso.

DBDesigner 4 integra diseño de base de datos, modelado, creación y mantenimiento en un entorno único, sin fisuras.

Combina características profesionales y una interfaz de usuario clara y sencilla para ofrecer la manera más eficiente para gestionar sus bases de datos.

Es un proyecto de código abierto disponible para Microsoft Windows 2k/XP y Linux KDE / Gnome.

1.9.4 Visual Paradigm

Es una herramienta para UML profesional, que soporta el ciclo de vida completo del desarrollo de *Software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

- ✓ Presenta un diseño centrado en casos de uso y enfocado al negocio.
- ✓ Usa un lenguaje estándar común a todo el equipo de desarrollo y facilita la comunicación.
- ✓ Tiene capacidades de ingeniería directa (versión profesional) e inversa.
- ✓ Tiene modelos y códigos que permanecen sincronizados en todo el ciclo de desarrollo.
- ✓ Presenta disponibilidad de múltiples versiones, para cada necesidad y múltiples plataformas.

(30)

Conclusiones parciales

A lo largo del desarrollo del capítulo:

- Se argumenta la necesidad de implementar un sistema que dé solución a la problemática planteada, pues ninguno de los estudiados se ajusta a los requisitos requeridos.
- Se seleccionaron las herramientas, metodología, lenguajes y tecnologías que se necesitan para el desarrollo de la propuesta y que más se adapten a las condiciones de desarrollo.

Capítulo 2. Descripción y análisis de la solución propuesta. Características del sistema.

Introducción.

Este capítulo está enmarcado en las tres primeras fases definidas en la metodología de desarrollo de *Software* XP: exploración, planificación e iteraciones. Se abordarán temas relacionados con el funcionamiento del sistema. Se hará una descripción del flujo actual de captación de participantes a determinadas actividades de la universidad y se determinan los objetos que serán automatizados para conformar una propuesta del sistema. Se mostrarán las historias de usuarios escritas por el cliente y se definen las iteraciones por programadores, constituyéndose el plan de entrega.

2.1.1 Flujo actual de eventos.

El proceso de captación de personas para actividades definidas se realiza de la siguiente manera:

Una persona (estudiante o profesor) encargada de recoger los datos necesarios de los interesados en participar en una actividad específica, recorre cada aula de la facultad a la que pertenece, con una hoja donde recoge la información solicitada.

También se les puede orientar a los presidentes de brigadas que reúnan esta información y, una vez realizada tal tarea se encargue de hacer llegar los datos a los organizadores de la actividad, a los que luego también les resulta difícil estructurar bien la actividad con tantos papeles.

Si existe alguien interesado en alguna actividad de cuya existencia haya sabido una vez recogidos los datos de los posibles participantes, se verá obligado a localizar a alguno de sus organizadores para ser incluido en la lista, lo cual le resultará algo complicado.

Otro modo de recoger los datos es solicitándolos vía correo, una o varias personas encargadas de divulgación de la actividad piden a través de listas de distribución que los interesados les envíen, también vía correo los datos que se especifican, a ciertos usuarios (organizadores de la actividad), llenando así buzones y cargando innecesariamente el servidor de correos.

2.1.2 Objetos de automatización.

El proceso de captación de estudiantes y profesores de nuestra universidad para la participación en una actividad, necesita que varias de sus acciones sean automatizadas.

Se automatizará la creación de plantillas de datos a través de la *Web* para que cualquier usuario pueda crear sus formularios para la recogida de los datos que considere necesarios, la posibilidad de modificarlo y eliminarlo, si se desea.

Se automatizará la disponibilidad de cada plantilla creada para cualquier usuario que se interese en llenar sus datos, y se automatizará también el almacenamiento de los datos de los interesados.

2.1.3 Propuesta del sistema.

El desarrollo del presente trabajo se proyecta a la implementación de un sistema que permita la obtención de un conjunto de datos y facilite su gestión.

Se permitirá crear formularios a través de la *Web*, de manera sencilla y sin tener que escribir código utilizando funciones de arrastrar y soltar para sus componentes. Para ello existirá una opción “Crear plantilla”, la cual mostrará un área de edición (para el formulario) y un panel con los componentes necesarios.

Los formularios creados estarán disponibles para el acceso de todo aquel que desee insertar sus datos, mostrándose una lista de las solicitudes creadas por un usuario específico, permitiendo el acceso al formulario en cuestión sin posibilidad de modificación, sólo de entrada de datos, los cuales serán respectivamente almacenados en tablas de una base de datos centralizada.

Para el usuario que tenga creado un formulario, se brindará la opción “Ver datos almacenados”, que le permitirá ver los datos existentes de los interesados en su propuesta.

Para la descripción de objetos y clases de la propuesta, serán utilizados algunos patrones de diseño con el fin de darle solución a un problema de diseño general, identificando clases, instancias, roles, colaboraciones, y la distribución de responsabilidades.

El uso de estos patrones de diseño para el desarrollo del sistema, aporta las ventajas que se muestran a continuación:

- Ofrecen una manera de reutilizar la experiencia de los desarrolladores, describiendo cómo solucionar los problemas que se presentan normalmente durante el desarrollo del producto,
- Se basan en recopilar todos los conocimientos de los especialistas en el desarrollo del *Software*.

- Mediante los patrones de diseño es más probable no consumar los mismos errores.

Entre los patrones de diseño a usar se encuentran:

El **patrón MVC** (Modelo, Vista y Controlador) es un patrón de diseño del *Software* en el que todo el proceso está dividido en 3 capas. Disminuye en gran medida la duplicación de código, y centralizando el control hace el sistema más extensible.

- El **Modelo** es el encargado de guardar los datos en un medio persistente. Es el proveedor de los recursos, independiente al sistema de almacenamiento de datos.
- La **Vista** se encarga de la interfaz de usuario, en la que se realizan operaciones simples y la mayoría de las validaciones, recibe datos del modela y los muestra al usuario, como también recoge datos del usuario, para enviar al modelo
- El **Controlador** es el que le envía a la vista las respuestas desde el modelo, realizando casi todo el trabajo funcional.
-

2.1.4 Patrones de diseño GRASP

Los patrones GRASP son de gran relevancia, los mismos son utilizados para la descripción de los principios fundamentales de la concesión de responsabilidades a objetos, pero formulados en forma de patrones.

Alta Cohesión

El uso de este patrón permite que cada clase tenga como responsabilidad trabajar en una misma parte de la aplicación y sin que tenga mucha complejidad. En otras palabras una clase tiene definida una responsabilidad en un área específica que le permite colaborar con otras realizando así las tareas.

Bajo Acoplamiento

Este patrón consiste en asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible.

2.1.5 Personas relacionadas con el sistema.

Las personas relacionadas con el sistema son todas aquellas que de una forma u otra van a interactuar con la aplicación, incluyendo a los que mantienen el sistema actualizado en correcto funcionamiento y los que utilizan sus servicios.

Personas relacionadas con el sistema	Descripción
Usuario	Es la persona que navega por el sistema con privilegios específicos. Tiene la posibilidad de acceder a los distintos contenidos y servicios que estén disponibles.

Tabla 2 Personas relacionadas con el sistema

2.2 Fase I: Exploración.

En esta etapa se definen las historias de usuario (HU), las cuales constan de 3 o 4 líneas escritas por el cliente en un lenguaje no técnico y describen algo que el sistema debe hacer. Además se produce el contacto con las herramientas y tecnologías que se emplearán para construir el sistema.

Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. Esta fase dura típicamente un par de semanas, el resultado es una visión general del sistema y un plazo total estimado.

Historias de usuario (HU).

Las HU son la técnica utilizada en XP para especificar los requisitos del *Software*. Son escritas por el propio cliente y describen brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

Si en un principio no se identifican todas las historias de usuario, al comienzo de cada iteración estarán registrados los cambios en las historias de usuario y así se planificará la siguiente iteración.

Las historias de usuario son divididas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración, y son clasificadas de la siguiente forma:
Teniendo en cuenta la **Escala Nominal de Prioridad en el Negocio:**
Alta: Se le otorga a las historias de usuario que resultan funcionalidades fundamentales en el desarrollo del proyecto, a las que el cliente define como principales para el control integral del negocio.
Media: Se le otorga a las historias de usuario que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación directa sobre el proyecto que se esté desarrollando.
Baja: Se le otorga a las Historias de Usuario que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura.
Teniendo en cuenta la **Escala Nominal de Riesgo en Desarrollo:**

Alta: Cuando para la implementación de la HU se considera la posible existencia de errores que lleven a inoperatividad del código.

Media: Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.

Baja: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

A continuación un ejemplo de historia de usuarios:

Historia de usuario	
Número: 4	**Nombre:** Crear plantilla
Usuario: Usuario	
Modificación de la historia de usuario:	**Iteración asignada: 1**
Prioridad en negocio: Alta **(Alta / Media / Baja)**	**Puntos estimados:** 1.0
Riegos en desarrollo: Alta **(Alta / Media / Baja)**	**Puntos reales:**
Descripción: El usuario tiene la posibilidad de crear un formulario que conformará la plantilla de recogida de datos.	
Observaciones: Se debe facilitar la creación del formulario con propiedades de arrastrar y soltar.	

Tabla 3 Ejemplo de Historia de usuario

Requerimientos:

Los requerimientos son la condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo, estos pueden dividirse en requerimientos funcionales y requerimientos no funcionales.

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

Las características de un requerimiento son sus propiedades principales. Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individualmente como en grupo. A continuación se presentan las más importantes:

Necesario: Un requerimiento es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características física o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.

Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.

Completo: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.

Consistente: Un requerimiento es consistente sino es contradictorio con otro requerimiento.

Verificable: Un requerimiento es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: Inspección, Análisis, Demostración o Pruebas.

(31)

Requisitos funcionales.

RF1: Autenticar y registrar usuario
RF2: Crear y eliminar formulario.
RF3: Mostrar lista de solicitudes.
RF4: Acceder a llenar datos.
RF5: Salvar datos de interesado.
RF6: Mostrar lista propia de solicitudes.
RF7: Mostrar datos almacenados.

Requisitos no funcionales del sistema.

- **Requisitos de apariencia o interfaz externa:**

La aplicación propuesta será usada por quien tenga o no los conocimientos básicos de informática, la interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo con la misma. La comunicación entre el servidor de base de datos y el servidor *Web* será mediante TCP/IP, entre las máquinas clientes y el servidor *Web* será por HTTP y entre el servidor y el directorio activo mediante LDAP.

- **Requisitos de rendimiento:**

Para un funcionamiento óptimo de la aplicación se seguirán las diferentes técnicas de elaboración de sitios *Web*, que faciliten el acceso rápido a sus páginas. La eficiencia del producto estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo Cliente Servidor, y la velocidad de las consultas de la base de datos. La aplicación propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible, adecuado a la rapidez con que el cliente requiere la respuesta a su petición.

- **Requisitos de seguridad:**

Confiabilidad: La información manejada por el sistema debe estar protegida de acceso no autorizado.

Integridad: La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados de inconsistencia.

- **Disponibilidad:** La aplicación deberá estar disponible en todo momento para aquellas personas con acceso a la información y los mecanismos utilizados para lograr la seguridad no deben ser un obstáculo a los usuarios para obtener los datos deseados en un momento dado.
- **Requisitos de *Software*:** En las máquinas clientes sólo se requiere un navegador *Web*, bajo cualquier sistema operativo Windows NT en adelante o cualquier distribución de Linux. La aplicación deberá ser multiplataforma.
- **Requerimientos de hardware:** En el cliente se requiere una máquina con 128 MB de RAM como mínimo, el servidor *Web* junto con el servidor de base de datos debe tener 256 MB de RAM y 20 GB de disco duro mínimo, todas las máquinas implicadas en la funcionalidad de la aplicación deben estar conectadas a la red de al menos 100Mbps de velocidad.

2.3 Fase II: Planificación y entrega.

En esta fase el cliente establece la prioridad de cada HU y a su vez, los programadores realizan una estimación del esfuerzo necesario para la realización de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto (semana ideal de programación).

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 el 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.

Plan de iteraciones.

Una vez identificadas las historias de usuario del sistema y estimado el esfuerzo requerido para la implementación de cada una de estas, se procede a realizar el Plan de iteraciones del proyecto. Teniendo en cuenta el riesgo para desarrollar cada una de las historias de usuario, el tamaño del equipo de desarrollo, así como otros factores subjetivos, se decidió dividir el proyecto en tres iteraciones, las cuales se detallan a continuación.

- **Iteración 1**

  En esta iteración se dará cumplimiento a las historias de usuario 4, 5, 7, 8 y 9 para intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto, aunque pueden surgir cambios, ya que el cliente puede sugerir las HU que considere necesarias para maximizar el valor del negocio. Las HU mencionadas anteriormente son las que cubren las funcionalidades básicas del sistema como la gestión de los formularios, y como el almacenamiento de sus características y la posibilidad de acceder a estos para insertar los datos necesarios para que sean almacenados en la base de datos.

- **Iteración 2**

  Esta iteración se centra en una parte de los requerimientos de prioridad media, los cuales no presentan riesgos elevados para los programadores, en su mayoría. Se conforma por las historias de usuarios 1, 6 y 10. Se encarga de la parte relacionada con el registro de los

  usuarios y su autenticación, así como la posibilidad que tiene un usuario de ver sus formularios creados en una lista y los datos almacenados de su captación.

- **Iteración 3**

Agrupa las historias de usuario que poseen baja prioridad en el negocio según la clasificación del cliente. Se conforma por las historias de usuario 2 y 3, que corresponden a mostrar los usuarios con formularios creados para poder eliminar los que no cumplan objetivo.

Historias de usuarios	Iteración	Tiempo real
Crear plantilla. Salvar plantilla. Mostrar lista de plantillas. Acceder a llenar datos. Salvar datos del interesado.	1	4 semanas
Autenticar. Mostrar lista propia de plantillas. Mostrar datos almacenados.	2	2 semanas
Mostrar usuarios Eliminar plantilla.	3	2 semanas

Tabla 4 Historias de usuarios de baja prioridad

Entregas	Historias de usuarios
Entrega 1	Crear plantilla. Salvar plantilla. Mostrar lista de plantillas. Acceder a llenar datos. Salvar datos del interesado.
Entrega 2	Autenticar. Mostrar lista propia de plantillas. Mostrar datos almacenados.
Entrega 3	Mostrar usuarios Eliminar plantilla.

Tabla 5 Plan de entrega

Entrega	Iteración 1 Cuarta semana de abril	Iteración 2 Segunda semana de mayo	Iteración 3 Cuarta semana de mayo
1	Versión 1.0	Versión 2.0	Concluido
2		Versión 1.0	Concluido
3		Versión 1.0	Concluido

Tabla 6 Plan de entrega de las versiones

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

Para poder diseñar el sistema como un equipo se deberá cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permitirán desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos, modelando al dominio en términos de los objetos que componen la aplicación, sirviendo de base para el desarrollo de software orientado a objetos. La naturaleza física de las tarjetas enfatiza la división de responsabilidades a través de los objetos. El tamaño físico de las tarjetas también ayuda a establecer límites para el tamaño y complejidad de las clases.

Una tarjeta CRC representa un objeto. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Diseñador_controlador.	
Responsabilidades	**Colaboraciones**
Realizar las funciones para crear y eliminar los formularios, de un usuario específico, permite: • Crear un formulario correspondiente a una plantilla. • Salvar fichero con la configuración del formulario.	Usuario_modelo Formulario_modelo

Tabla 7 Ejemplo de Tarjeta CRC

Interfaces propuestas.

- **Autenticar usuario:** Interfaz mediante la cual el usuario se autentica en el sistema, o se registra si no existe.
- **Gestionar plantillas:** Interfaz mediante la cual el usuario una vez autenticado podrá ver su lista de plantillas (formularios) creadas, tendrá las opciones de crear una nueva, así como editar las

ya existentes, eliminarlas o mostrar los datos que han sido almacenados correspondientes a una específica.

- **Nueva plantilla:** Una vez seleccionada la opción de crear una nueva plantilla debe mostrarse esta interfaz, la cual consta de un panel en la parte izquierda de la aplicación con los componentes necesarios para la construcción del formulario (label, textbox, checbox, radiobuttons…), la aplicación permitirá arrastrar dichos componentes hacia un panel de edición ubicado a la derecha del segundo panel. El botón “crear” que estará ubicado en la parte inferior derecha de esta interfaz permitirá salvar el formulario confeccionado creando las tablas y campos correspondientes en la base de datos.
- **Eliminar plantilla:** Esta interfaz mostrará un mensaje de confirmación, si desea o no eliminar la plantilla seleccionada.
- **Mostrar datos de plantilla:** En esta interfaz se muestra una lista de los datos que existen ya almacenados para los que se creó un formulario especificado.
- **Ver todas las plantillas:** En esta interfaz se muestra una lista de todas las plantillas creadas, para que el usuario si lo desea inserte sus datos en una escogida.
- **Insertar datos:** En esta interfaz se carga el formulario seleccionado por un usuario para insertar los datos requeridos.

Diagrama de despliegue.

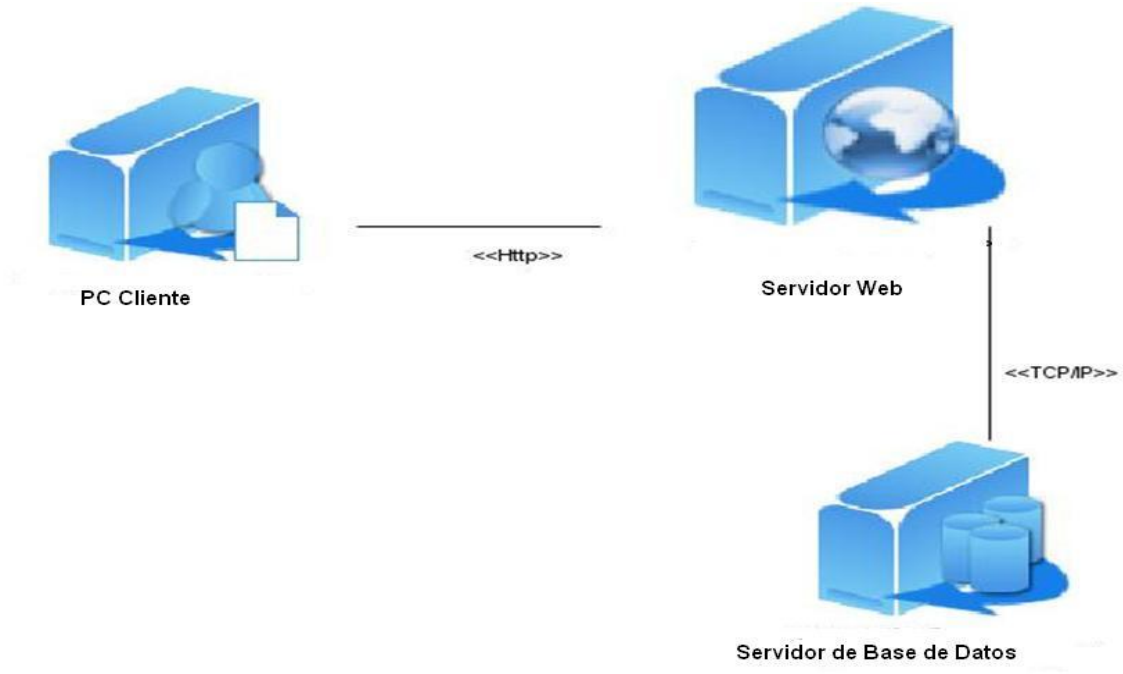


Figura 1 Diagrama de despliegue.

Modelo de datos

Una de las características fundamentales de los sistemas de bases de datos es que proporcionan cierto nivel de abstracción, al ocultar las características sobre el almacenamiento físico que la mayoría de usuarios no necesita conocer. Estos modelos son el instrumento principal para ofrecer dicha abstracción.

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos, las relaciones y las restricciones que deben cumplirse. Los modelos de datos contienen también un conjunto de operaciones básicas para la realización de consultas (lecturas) y actualizaciones. Además, los más modernos incluyen conceptos para especificar comportamiento, permitiendo especificar un conjunto de operaciones definidas por el usuario.

Los modelos de datos se pueden clasificar dependiendo de los tipos de conceptos que ofrecen para describir la estructura de la base de datos. Los modelos de alto nivel, o modelos conceptuales, disponen de conceptos muy cercanos al modo en que la mayoría de los usuarios percibe los datos, mientras que los de bajo nivel, o modelos físicos, proporcionan conceptos que describen los detalles de cómo se almacenan en el ordenador. Los conceptos de los modelos físicos están dirigidos al personal informático, no a los usuarios finales. Entre estos dos extremos se encuentran los modelos lógicos, cuyos conceptos pueden ser entendidos por los usuarios finales, aunque no están demasiado alejados de la forma en que se organizan físicamente. Los modelos lógicos ocultan algunos detalles de cómo se almacenan los datos, pero pueden implementarse de manera directa en un ordenador.

Los modelos conceptuales utilizan conceptos como entidades, atributos y relaciones. Una entidad representa un objeto o concepto del mundo real como, por ejemplo, un empleado de la empresa inmobiliaria o una oficina. Un atributo representa alguna propiedad de interés de una entidad como, por ejemplo, el nombre o el salario del empleado. Una relación describe una interacción entre dos o más entidades, por ejemplo, la relación de trabajo entre un empleado y su oficina.

Tipos de entidades.

Independientes: no dependen de ninguna otra entidad para su identificación.

Dependientes: dependen de una o más entidades para su identificación.

Atributos:

.1- Clave primaria (PK): atributo o grupo de atributos elegido como el único identificador de una entidad.

2.- Clave candidata (CK): atributo o grupo de atributo que pueden ser elegidos como PK.

3.- Clave ajena (FK): clave primaria de otra entidad.

3.- Atributo no clave: no puede ser elegido como PK.

4.- Atributo derivado: calculado a partir de otros atributos. (No necesita ser almacenado físicamente).

5- Rol-nombre atributo (FK): Nombre del rol de un atributo. Se toma un nuevo nombre para la FK que tenga una connotación con su uso.

6.- Grupo (c1, c2, c3): El atributo es un grupo y se listan los componentes.

7.- Clave ajena unificada (fk1, fk2, fk3) (FK): La lista de claves se unen en una simple FK.

(32)

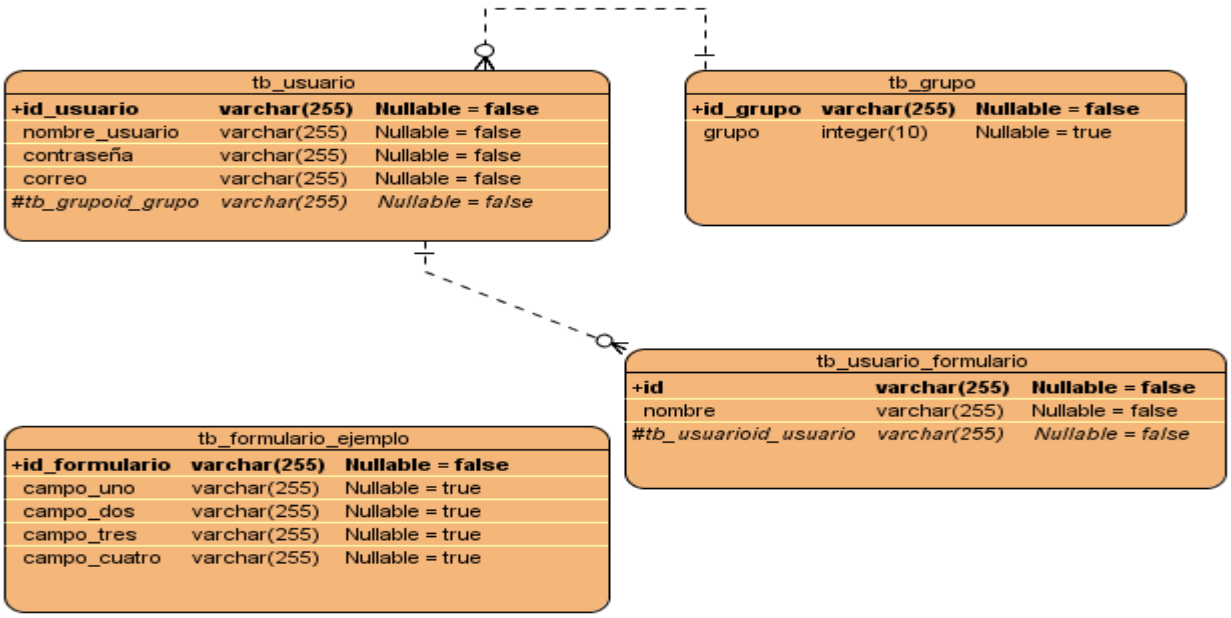


Figura 2 Modelo lógico.

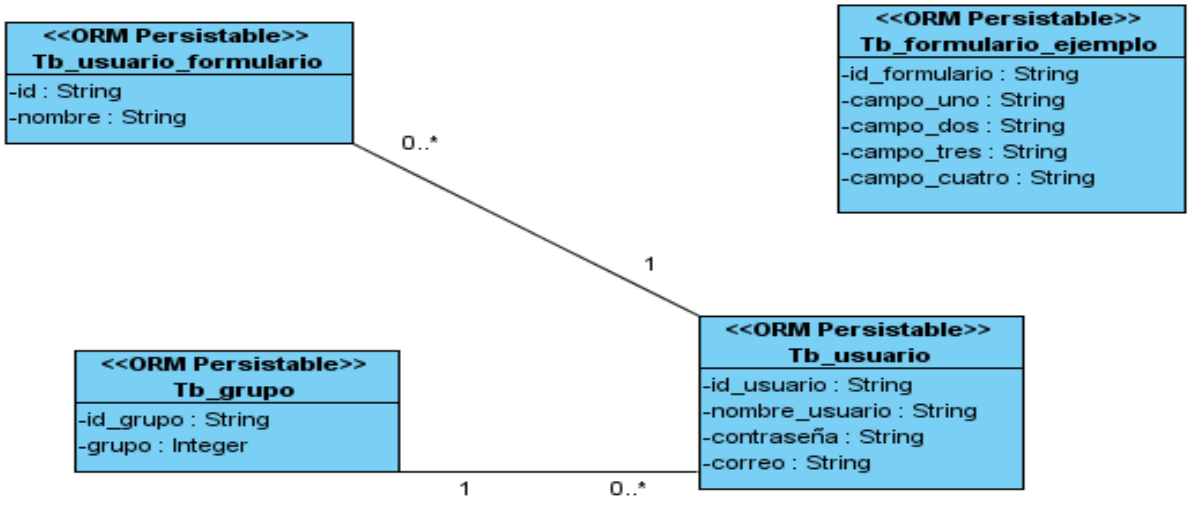


Figura 3 Modelo físico.

Nota: La tabla formulario_ejemplo representa uno de los formularios que se crean por cada plantilla de manera dinámica, con los campos que se le definen al formulario, no es una tabla estática del sistema y pueden ser muchas otras, en dependencia de la cantidad de plantillas creadas.

2.4 Fase III: Iteraciones.

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema está listo para ser explotado.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior.

Conclusiones parciales

En el presente capítulo:

- Se abordaron los temas referentes a las fases de exploración y planificación del proyecto basado en la metodología XP.
- Como resultado de la aplicación de las técnicas de ingeniería propuestas por esta metodología para estas fases, se obtuvieron los **artefactos** necesarios para comenzar el diseño del sistema.
- Una mirada exhaustiva a los artefactos generados brinda el primer acercamiento al sistema a construir, aunque sólo de una manera superficial.

Capítulo 3. Desarrollo y pruebas.

Introducción

En el presente capítulo de describen las fases de Implementación y Pruebas propias de la metodología de desarrollo XP. Se detallan las dos iteraciones realizadas durante todo el proceso de codificación del proyecto y se hace referencia a las pruebas de aceptación realizadas sobre el sistema.

3.1 Implementación.

4 Durante el inicio de cada iteración se realizan una revisión del Plan de iteraciones y se modifica si es necesario. El trabajo completo de la iteración se expresa en tareas de programación y cada una de ella es asignada a un programador en específico como máximo responsable. Estas tareas pueden interpretarse utilizando un lenguaje técnico y no necesariamente el cliente debe poder entenderlas.

5 A continuación se muestra detalladamente las tareas en las que se descomponen las historias de usuario en cada una de las iteraciones, según la planificación propuesta inicialmente.

6 **Iteración 1.** Se desarrollan 5 historias de usuario, las de más valor en la escala nominal de prioridad en el negocio.

Historias de usuario	Tiempo de implementación	
	Estimación	Tiempo real
Crear plantilla.	1	1.5
Salvar plantilla.	1	1
Mostrar lista de plantillas.	0.5	0.5
Acceder a llenar datos.	0.5	0.5
Salvar datos del interesado.	1	1

Tabla 8 Historias de usuarios desarrolladas en la primera iteración.

Iteración 2. Se desarrollan tres historias de usuario.

Historias de usuario	Tiempo de implementación	
	Estimación	Tiempo real
Autenticar.	1	1
Mostrar lista propia de plantillas.	0.1	0.1
Mostrar datos almacenados.	1	1

Tabla 9 Historias de usuario desarrolladas en la segunda iteración.

3.2 Pruebas.

Las unidades de test o pruebas constituyen unos de los pilares básicos de la Programación Extrema (XP). Estas están directamente relacionadas con el concepto de posesión del código. En cierta manera, una parte del código no será reemplazado si no supera los test que existen para ese código.
La realización de este tipo de pruebas y la publicación de los resultados debe realizarse lo más rápido posible, para que los desarrolladores puedan, con la mayor rapidez posible, aplicar los cambios que sean necesarios. A las pruebas de aceptación también se les conoce con el nombre de pruebas de funcionalidad, y constituyen la garantía de que los requerimientos fijados por los usuarios han sido reflejados en el sistema.

3.2.1 Especificación de prueba: Autenticar

Descripción:

En esta historia el usuario tendrá la opción de autenticarse y navegar según sus privilegios.

Condiciones de ejecución:

Debe existir en la base de datos el usuario que se autentica.

Entrada:

El usuario introducirá su usuario y contraseña.

Resultado esperado:

El usuario se autentica satisfactoriamente.

Evaluación de la prueba:

Prueba satisfactoria.

3.2.2 Especificación de prueba: Mostrar usuarios

Descripción:

Mostrará los usuarios que tienen creadas plantillas de datos.

Condiciones de ejecución:

Debe existir en la base de datos el usuario que se autentica.

Entrada:

El usuario introducirá su usuario y contraseña.

Resultado esperado:

La lista se muestra junto con las plantillas creadas.

Evaluación de la prueba:

Prueba satisfactoria.

3.2.3 Especificación de prueba: Eliminar formulario

Descripción:

El usuario puede eliminar un formulario creado por él.

Condiciones de ejecución:

Que exista el formulario.

Entrada:

De la lista de plantillas propias se seleccionará una y se escogerá la opción: “Eliminar plantilla”.

Aparecerá un cuadro de diálogo pidiendo confirmación sobre la eliminación de la plantilla.

El usuario seleccionará “Sí”

El formulario será eliminado del sistema.

Resultado esperado:

El formulario ha sido eliminado de la base de datos.

Evaluación de la prueba:

Prueba satisfactoria.

3.2.4 Especificación de Prueba: Crear plantilla

Descripción:

El usuario tiene la posibilidad de crear, un formulario.

Condiciones de ejecución:

El usuario debe estar autenticado

Entrada:

El usuario introducirá su usuario y contraseña.

Del menú principal seleccionará: “Crear plantilla”.

Resultado esperado:

El usuario puede arrarstrar los componentes para crear una plantilla,

Evaluación de la prueba:

Prueba satisfactoria.

3.2.5 Especificación de prueba: Salvar plantilla.

Descripción:

El usuario tendrá la opción de guardar la plantilla creada.

Condiciones de ejecución:

No debe existir una plantilla guardada con el mismo nombre.

Entrada:

El usuario introducirá su usuario y contraseña.

Una vez creada la plantilla del menú principal seleccionará “Salvar Plantilla”.

Resultado esperado:

Se genera una tabla en la base de datos con el nombre y los campos asignados.

Evaluación de la prueba:

Prueba satisfactoria.

3.2.6 Especificación de Prueba: Mostrar lista propia de plantillas

Descripción:

Cada usuario podrá ver la lista de sus propios formularios, con las opciones de mostrar datos almacenados y eliminar.

Condiciones de ejecución:

El usuario debe estar autenticado.

Entrada:

El usuario introducirá su usuario y contraseña.

Del menú principal seleccionará “Mostrar lista propia de formularios”.

Se mostrará un listado con todos los formularios que hayan sido creados por el usuario.

Resultado esperado:

Se muestran todos los formularios.

Evaluación de la prueba:

Prueba satisfactoria.

3.2.7 Especificación de Prueba: Mostrar lista de plantillas

Descripción:

Los usuarios tienen acceso a la lista de todos los formularios creados, para la solicitud de datos.

Condiciones de ejecución:

Debe existir algún formulario en la base de datos.

Entrada:

Los usuarios introducirán su usuario y contraseña.

Se mostrará un listado con todos los formularios que hayan sido creados.

Resultado esperado:

Se muestran las plantillas creadas.

Evaluación de la prueba:

Prueba satisfactoria.

3.2.8 Especificación de Prueba: Acceder a llenar datos.

Descripción:

Cualquier usuario interesado en una solicitud, podrá entrar al formulario de la plantilla a llenar sus datos.

Sólo puede llenar los campos, no puede modificar, ni eliminar información.

Condiciones de ejecución:

Debe existir alguna plantilla en la base de datos.

Entrada:

El usuario introducirán su usuario y contraseña.

Se mostrará un listado con todas las plantillas que hayan sido creadas.

El usuario elige la deseada.

El usuario llena los datos del o de los formularios que esta posea.

Resultado esperado:

Se llenan los datos de la plantilla.

Evaluación de la prueba:

Prueba satisfactoria.

3.2.9 Especificación de prueba: Salvar datos de interesado

Descripción:

Una vez llenado los datos de un formulario específico el usuario puede almacenarlos.

Condiciones de ejecución:

El usuario seleccione la opción “Llenar datos”

Entrada:

El usuario escoge la opción “Enviar datos”

Aparecerá un cuadro de diálogo informándole que sus datos han sido guardados.

Resultado esperado:

Se guardan los datos en la plantilla.

Evaluación de la prueba:

Prueba satisfactoria.

3.2.10 Especificación de prueba: Mostrar datos almacenados.

Descripción:

Los usuarios tienen acceso a la lista de los datos almacenados en una plantilla propia.

Condiciones de ejecución:

Deben existir plantillas.

Entrada:

El usuario introducirá su usuario y contraseña.

Del menú principal seleccionará “Mostrar lista propia”.

Se mostrará un listado con todas las plantillas que hayan sido creadas por él

Seleccionará una plantilla.

Escogerá la opción “Mostrar datos almacenados”.

Resultado esperado:

Se los datos almacenados en la plantilla escogida.

Evaluación de la prueba:

Prueba satisfactoria.

Resultado de la encuesta realizada.

Se realizó una encuesta que se encuentra publicada en conjunto con los objetos para que sus usuarios ya sean estudiantes o profesores expongan su criterio con respecto a los mismos, comenten si les fueron útiles, y que brinden alguna sugerencia para mejorarlos.

Esta encuesta se le aplicó a un total de unos 37 estudiantes de las distintas facultades de la universidad, para verificar también la amplitud de la necesidad del generador de plantillas en la *Web*. A continuación se muestran los resultados obtenidos.

Encuesta para usuarios del sistema		
Generar plantillas		
Obtención de datos.		
	Sí	No
1. ¿Alguna vez has necesitado de recoger datos de personas para alguna actividad cualquiera?		
2. ¿El generador de plantillas te posibilita crear el formulario que deseas?		
3. ¿Puedes acceder a las plantillas creadas por otros para insertar tus datos?		
4. ¿Tienes acceso a los datos almacenados referentes a alguna de tus plantillas?		
5. ¿Crees que la aplicación facilite la obtención de datos en un proceso de captación de cualquiera?		

Tabla 10 Encuestas.

En los resultados de la pregunta 1 se obtuvo un 94.6% de respuestas positivas por parte de los encuestados, por lo que se puede decir que es necesario un modo de obtener datos de modo simple, práctico y variado para cualquier tipo de actividad.

En los resultados de la pregunta 2 se obtuvo un 91.9 % de aprobación por parte de los encuestados, luego, el generador de plantillas posibilita crear casi cualquier tipo de formulario necesitado.

Con un 100% de aprobación de los encuestados en la pregunta 3 y 4 se demuestra que existe el acceso de un interesado en cierta actividad a llenar sus datos, los cuales estarán disponibles para ser consultados por el organizador de la misma.

Y un 97.3% expresa que el generador automático de plantillas en la *Web* facilita la obtención de datos de un proceso de captación cualquiera.

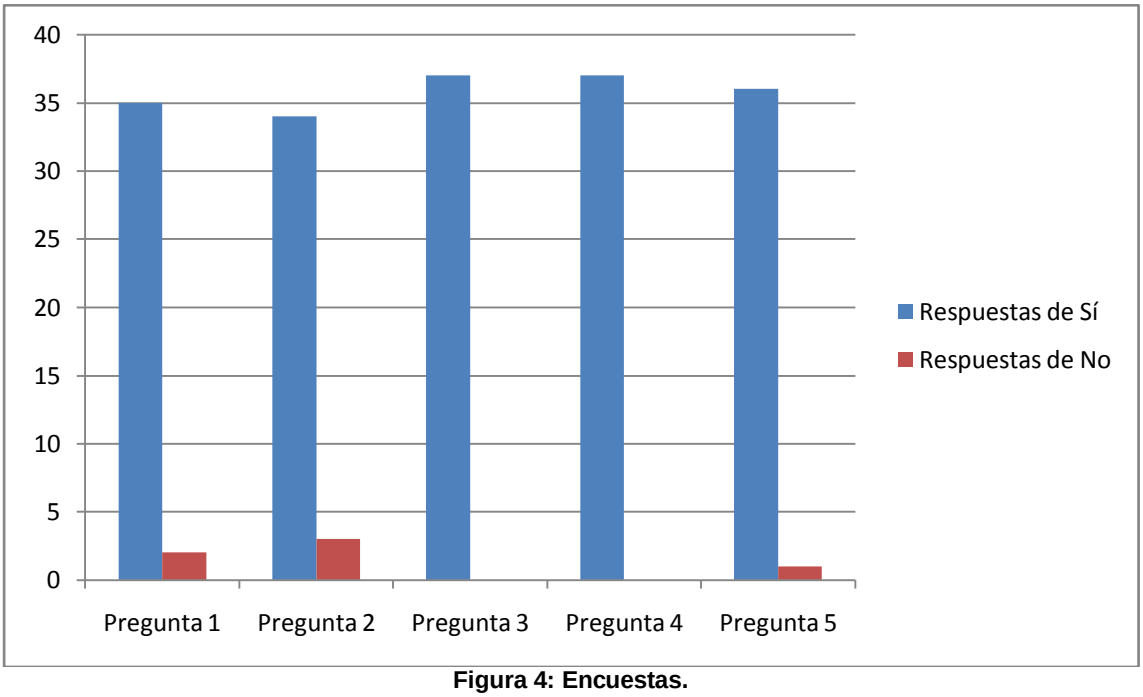


Figura 4: Encuestas.

De este modo se demuestra que con la implementación del generador automático de plantillas a través de la *Web*, se facilita el proceso de obtención de los datos de un usuario interesado en una actividad definida, dándole así validez a la hipótesis planteada.

Conclusiones parciales

- Las fases de Implementación y Prueba tienen un gran peso dentro de la metodología XP debido a que son las que cierran y verifican el correcto desarrollo de la aplicación.
- Los artefactos que se generan en esta fase son de vital importancia para comprobar si ha sido satisfactorio el trabajo realizado a lo largo de la vida de la investigación.
- Se generaron los artefactos como resultado de las actividades realizadas en las fases de Implementación y Prueba, las tareas de ingeniería y las pruebas de aceptación.
- Se realizó una encuesta donde se muestra la satisfacción de los usuarios con el producto obtenido.

Capítulo 4. Factibilidad del sistema

Introducción

COCOMO, propuesto y desarrollado por Barry Boehm, es uno de los modelos de estimación de costos mejor documentados y utilizados. El modelo permite determinar el esfuerzo y tiempo que se requiere en un proyecto de *Software* a partir de una medida del tamaño del mismo expresada en el número de líneas de código que se estimen generar para la creación del producto *Software*.

Consiste básicamente en la aplicación de ecuaciones matemáticas sobre los Puntos de Función sin ajustar o la cantidad de líneas de código (SLOC*, Source Lines of Code*) estimados para un proyecto. Estas ecuaciones se encuentran ponderadas por ciertos factores de costo (*cost drivers*) que influyen en el esfuerzo requerido para el desarrollo del *Software*. Está compuesto por tres modelos que se adaptan tanto a las necesidades de los diferentes sectores descritos, como al tipo y cantidad de información disponible en cada etapa del ciclo de vida de desarrollo. (33)

4.1 Costo en desarrollo y tiempo (Programación Extrema)

En la práctica se observa que, el enfoque tradicional con respecto al tiempo, incrementa los costos en el proceso de desarrollo: Análisis, Diseño, Implementación, Integración, Pruebas, Producción y Capacitación del sistema.

Figura 5 Costo en desarrollo vs tiempo (XP).

Mientras que la programación extrema, al manejar los principios y prácticas con pocos artefactos, entregas continuas, juego de planificación, retroalimentación con el cliente en casa, metáforas de desarrollo, estándares de programación, 40 horas semanales, programación en parejas, *stand up*

meeting, tarjetas CRC, pruebas unitarias y refactorización disminuye el tiempo de desarrollo e implementación obteniendo un producto conocido por el usuario y a su medida, cabe resaltar que XP no es recomendado para proyectos grandes y equipos amplios.

La rentabilidad, estimación de esfuerzo y costo de un proyecto de *Software* son, sin lugar a dudas, una de las tareas de mayor importancia para la producción de *Software*, ya que cada vez es mayor la necesidad de obtener datos que permitan evaluar, predecir y mejorar la calidad del producto. En el presente capítulo se realizará un estudio de factibilidad para la realización del sistema propuesto mediante una estimación de tamaño, esfuerzo y planificación necesaria para llevar a cabo el mismo.

.4.1 Estimación de esfuerzos por historia de usuario.

Historias de usuarios	Tiempo estimado(semana ideal de trabajo)	Iteración asignada	Tiempo real
Autenticar.	1.0		1.5
Mostrar usuarios.	1.0		1.5
Eliminar formulario.	0.7		1.0
Crear plantilla.	0.5		1.0
Salvar plantilla.	0.7		1.0
Mostrar lista propia de formularios.	0.6		1.0
Mostrar lista de solicitudes.	0.4		1.0
Acceder a llenar datos.	1.0		1.6
Salvar datos de interesado.	0.2		0.9
Mostrar datos almacenados.	1.0		1.5

Tabla 11 Estimación de esfuerzos por historia de usuario.

4.2 Cálculo de esfuerzo utilizando COCOMO.

El primer paso a llevar a cabo para la estimación del proyecto consiste en la obtención de los puntos de función desajustados, los cuales están dados por la suma de cada una de las entradas, las salidas y las consultas externas del sistema, así como los archivos lógicos internos y de interfaz externos. A continuación se muestran cada una de estas características aplicadas al *Software* en cuestión.

4.2.1 Entradas externas.

Es el proceso mediante el cual los datos pasan la frontera del sistema desde afuera hacia adentro. En este caso muy particular se cuenta con 2 entradas externas las que se muestran en la tabla siguiente.

Nombre de la entrada externa	Cantidad de ficheros	Cantidad de elementos	Clasificación (Simple, Media, Compleja)
Autenticar	1	2	Simple
Acceder a llenar datos	1	3	Media
Total		5	

Tabla 12 Entradas externas.

4.2.2 Salida externa.

Se define como un proceso fundamental con componentes de entrada y de salida mediante el cual los datos simples y derivados pasan del sistema desde adentro hacia afuera. Las salidas externas vinculadas al proyecto se describen en la tabla que se muestra a continuación.

Nombre de la salida externa	Cantidad de ficheros	Cantidad de elementos	Clasificación (Simple, Media, Compleja)
Mostrar lista propia de formularios.	1	3	Media
Mostrar lista de solicitudes.	1	2	Media
Mostrar datos almacenados.	1	4	Media
Total		9	

Tabla 13 Salidas externas.

4.2.3 Consultas externas.

Un proceso elemental con componentes de entradas y de salida donde el actor del sistema busca los datos de uno o más archivos lógicos internos o archivos de interfaz externa.

Nombre de la petición.	Cantidad de ficheros.	Cantidad de elementos.	Clasificación (Simple, Media, Compleja).
Total		0	

Tabla 14 Consultas externas.

4.2.4 Archivos lógicos Internos.

Son un grupo de datos relacionados lógicamente e identificables por el usuario, que residen enteramente dentro de los límites del sistema y se mantienen a través de entradas externas.

Nombre del fichero interno	Cantidad de récords.	Cantidad de elementos de datos.	Clasificación (Simple, Media, Compleja).
Plantilla llenada por el usuario	1	3	Simple
Total		3	

Tabla 15 Archivos lógicos internos.

4.2.5 Archivos de interfaz externos.

Son un grupo de datos relacionados e identificables por el usuario que sólo se utilizan para referencias. Los datos residen fuera del sistema y se mantienen por la entrada externa de otras aplicaciones, o sea cada archivo de interfaz externo es un archivo lógico interno de otra aplicación.

En este caso no fueron identificados.

Nombre de la entrada externa	Cantidad de records.	Cantidad de elementos de datos.	Clasificación (Simple, Media, Compleja).
Total		0	

Tabla 16 Archivos de interfaz externos.

4.2.6 Puntos de función desajustados.

La tabla siguiente muestra las características del sistema anteriormente expuestas, el producto de la cantidad existentes de cada una de ellas y los pesos correspondientes a las mismas dan como resultado final, los puntos de función desajustados pertenecientes al proyecto en general.

Elementos	Simple		Medio		Complejo		Subtotal
	No.	Peso	No.	Peso	No.	Peso	
Entrada externa	5	3	0	4	0	6	15
Salida externa	9	4	0	5	0	7	36
Consultas externas	0	3	0	4	0	6	0
Fichero lógico interno	3	7	0	10	0	15	21
Fichero de interfaz externa	0	5		7		10	0
Total (UFP)							72

Tabla 17 Puntos de función desajustados.

4.2.7 Cálculo de instrucciones fuentes.

Después de haber obtenido el total de puntos de función desajustados se calcula la cantidad de instrucciones fuentes con la ecuación que se muestra a continuación:

SLOC = UFP x Ratio

SLOC va a ser la cantidad de instrucciones fuentes, y el UFP: los puntos de función desajustados (72).

Ratio: Conversión de puntos de función desajustados a líneas de código para el lenguaje PHP (59).

SLOC= 72 x 25= 4248

Y al dividir el SLOC entre 1000 resulta el KSLOC

KSLOC= 4.24

KSLOC es la cantidad de instrucciones fuentes expresado en millares.	**Valor**
Puntos de función desajustados.	72
Lenguaje PHP.	59
Instrucciones fuentes por puntos de función.	4248 SLOC
Instrucciones fuentes	4.24 KSLOC

Tabla 18 Datos del Cocomo

4.2.8 Cálculo de esfuerzo nominal.

PM nominal= A x (SIZE) ^ B

PM nominal=2.94 x ((4.24) ^ 0.95)

PM nominal=11.59 meses/hombre

PM nominal: Esfuerzo nominal requerido en meses.

SIZE: Tamaño estimado del *Software* en puntos de función sin ajustar (KSLOC).

A: constante que se utiliza para capturar los efectos multiplicativos en el esfuerzo requerido de acuerdo al crecimiento del tamaño del S*oftware*. (Valor: 2.94).

B: Constante denominada Factor escalar y su valor esta dado por las resultantes de los aspectos positivos sobre los negativos que presenta el proyecto.

B= 0.91 + 0.01 x sumatoria de W

B= 0.91 + 0.01 x (4. 91)

B= 0.95

W: Variable escalar que indica las características que el proyecto presenta en cuanto a complejidad y entorno de desarrollo.

Nombre	Valor	Justificación
Prec	1.24	Existen varios proyectos similares a nivel internacional, pero no se encontró ninguno a nivel nacional.
Flex	1.01	Cuenta con alta flexibilidad en cuanto a los requisitos establecidos inicialmente.
Team	1.10	El equipo de desarrollo presenta una alta cohesión.
Resl	2.83	Se identificaron riesgos críticos en un porciento elevado.
Pmat	1.56	Se cuenta con la experiencia necesaria como para que el *Software* cumpla con las funcionalidades requeridas.
Total (SF)	4. 91	

Tabla 19 Factores de escala.

4.2.9 Ajuste del esfuerzo nominal

El resultado calculado anteriormente es un esfuerzo nominal que debe ser ajustado en base a las características del proyecto.

Nombre	Valor	Justificación
RCPX	1.74	La complejidad del sistema es alta.
RUSE	1.0	Pretende reutilizarse sólo una parte del código.
PDIF	1.0	Uso de memoria y almacenamiento normal, plataforma estable.
PREX	1.12	Baja experiencia del personal en la utilización del lenguaje, así como de la plataforma.
PERS	1.0	La capacidad del personal es alta.
FCIL	1.0	La utilización de entornos de desarrollo integrados, así como de herramientas CASE simples facilitan en gran medida el trabajo.
SCED	1.0	El sistema se desarrollo en el tiempo establecido.
Total (EM)	3.12	

Tabla 20 Ajuste del esfuerzo nominal.

PM ajustado = PM nominal x EM.

PM ajustado =11.59 x 1.95= 22.60 hombres/mes

Cálculo del tiempo de desarrollo del sistema

El tiempo requerido para el desarrollo de la aplicación se calcula por la siguiente ecuación.

TDEV= C x (PM ajustado) ^ F

TDEV=3.64 x (22.60) ^ 0.24

TDEV=7.69 meses.

F= D + 0.2 x 0.01 x sumatoria de SF

F= 0.24+ 0.2 x 0.01 x 4.91= 0.24982

C: Constante de valor 3.64

Donde

D: constante cuyo valor es 0.24

SF: valor de los factores de escala de 4.91

4.2.10 Cálculo del costo total del proyecto

Cocomo para calcular el costo total del proyecto define la ecuación que se muestra a continuación.

C= CHM x PM

C= 100 x 22.60

C= $ 2260.

C: Costo total

CHM: Costo teniendo en cuenta el salario de todos los integrantes del grupo de desarrollo.

CHM= CH x salario.

CHM= 2 x 50.

Salario: Salario de un integrante del grupo de desarrollo de *Software*.

CH= Cantidad de personas que participan en el proyecto.

4.2.11 Resultados finales.

Cálculo de :	**Valor**
Esfuerzo	22.60 hombres/mes
Tiempo de desarrollo	7.69 meses
Cantidad de hombres	2.93
Salario medio	$ 100
Costo	$ 2260

Tabla 21 Resultados Finales.

Conclusiones parciales

Durante el desarrollo del capítulo se logró:

- Determinar el esfuerzo y tiempo que se necesita para realizar el proyecto, así como el costo total del mismo.
- Concretar que la aplicación propuesta es factible, debido a que el gasto más significativo es el salario que se les dará a los desarrolladores.

Conclusiones generales

Luego de desarrollada la investigación se tiene como resultado

- Se seleccionaron las herramientas, tecnologías y metodología más adecuadas para dar cumplimiento a los objetivos propuestos.
- Las aplicaciones generadoras de formularios encontradas no satisfacen las necesidades existentes en la Universidad de Ciencias Informáticas.
- Se identificaron los requerimientos funcionales y no funcionales, logrando una primera visión del sistema.
- Mediante el análisis y diseño se generaron los artefactos necesarios para el desarrollo de la aplicación Web.
- Se realizó la implementación del sistema genérico, que permite al usuario crear una plantilla de recogida de datos con funciones de arrastrar y soltar, sin tener que escribir código, brindando también la facilidad de que otro interesado llene sus datos y pueda salvarlos.
- Con la realización de las pruebas se demostró que la aplicación “Generador Automático de Plantillas en la *Web”* se encuentra en correcto funcionamiento.
- Se le dio validez a la hipótesis planteada a través de las encuestas realizadas.

Recomendaciones:

- Crear un módulo de administración, donde un usuario del grupo administradores pueda eliminar cualquier plantilla que se encuentre en desuso.
- Agregar componentes al generador para aumentar las opciones de diseño de las plantillas.

Bibliografía referenciada.

1. Generador de Formularios. [En línea] [Citado el: 12 de Noviembre de 2009.] http://www.Webformfactory.com/.

2. FormMailGenerator. [En línea] [Citado el: 13 de Diciembre de 2009.] http://www.xeduced.com/2009/05/22/generador-online-de-formularios-html-y-php/.

3. WebsiteContactFormGenerator . [En línea] [Citado el: 16 de Noviembre de 2009.] http://www.tele-pro.co.uk/scripts/contact_form/.

4. Wufoo. [En línea] [Citado el: 20 de Noviembre de 2009.] http://wufoo.com/ .

5. Software. [En línea] [Citado el: 1 de Diciembre de 2009.] http://www-01.ibm.com/Software/awdtools/rup/.

6. SlideShare. [En línea] [Citado el: 2 de Diciembre de 2009.] http://www.slideshare.net/edgarespinoza/programacion-extrema.

7. Geeks. [En línea] [Citado el: 4 de Diciembre de 2009.] http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx .

8. Lenguajes de Programación. [En línea] [Citado el: 5 de Diciembre de 2009.] http://www.lenguajes-de-programacion.com/programacion-web.shtml.

9. HTML. [En línea] [Citado el: 6 de Diciembre de 2009.] http://www.mailxmail.com/b-hipervinculos.

10. CSS. [En línea] [Citado el: 6 de Diciembre de 2009.] http://html.conclase.net/w3c/html401-es/present/styles.html.

11. JavaScript. [En línea] [Citado el: 7 de Diciembre de 2009.] http://www.slideshare.net/Darklink/programacin-java-script.

12. Java. [En línea] [Citado el: 8 de Diciembre de 2009.] http://www.java.com/es/ .

13. Java. [En línea] [Citado el: 10 de Diciembre de 2009.] http://www.java.com/es/ .

14. PHP. [En línea] [Citado el: 12 de Diciembre de 2009.] http://php.net/index.php .

15. UML. [En línea] [Citado el: 13 de Diciembre de 2009.] http://www.buenastareas.com/ensayos/Lenguaje-Unificado-De-Modelado/145516.html.

16. Servidor Web. [En línea] [Citado el: 14 de Diciembre de 2009.]
http://members.fortunecity.com/computerize/servidor1.htm.

17. Apache. [En línea] [Citado el: 15 de Diciembre de 2009.]
http://www.apache.org/ .

18. Servidores Web. [En línea] [Citado el: 18 de Diciembre de 2009.]
http://www.anurix.com/docs/Introduccion_a_lighttpd.pdf .

19. Servidor. [En línea] [Citado el: 20 de Diciembre de 2009.]
http://technet.microsoft.com/es-es/library/cc740244%28WS.10%29.aspx

20. Cibernetia. [En línea] [Citado el: 16 de Diciembre de 2009.]
http://www.cibernetia.com/manuales/instalacion_servidor_web/3_otros_servidores_web.php).

21. Sat-Linux. [En línea] [Citado el: 18 de Enero de 2010.]
http://www.sat-linux.com/forum/showthread.php?p=11257.

22. Sistemas. [En línea] [Citado el: 20 de Enero de 2010.]
http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

23. MySQL. [En línea] [Citado el: 24 de Enero de 2010.]
http://www.mysql.com .

24. FirebirdSQL. [En línea] [Citado el: 25 de Enero de 2010.]
http://www.firebirdsql.org/.

25. PostgreSQL. [En línea] [Citado el: 26 de Enero de 2010.]
http://www.postgresql.org/.

26. Sip. [En línea] [Citado el: 28 de Enero de 2010.]
http://www.sip.gob.mx/tecnologia/416-tecnologias-ajax.

27. ExtJS. [En línea] [Citado el: 30 de Enero de 2010.]
http://www.desarrolloweb.com/wiki/ext-js.html.

28. DesarrolloWeb. [En línea] [Citado el: 1 de Febrero de 2010.]
http://www.desarrolloweb.com/articulos/controllers-codeigniter.html.

29. Aptana. [En línea] [Citado el: 3 de Febrero de 2010.]
http://www.aptana.org/.

30. VisualParadigm. [En línea] [Citado el: 5 de Febrero de 2010.]
http://www.visual-paradigm.com/.

31. Requerimientos. [En línea] [Citado el: 15 de Febreo de 2010.]
http://www.buenastareas.com/ensayos/Documento-De-Requerimientos-De-Usuarios-Para/77375.html.

32. Modelo de Datos. [En línea] [Citado el: 3 de Marzo de 2010.]
http://www3.uji.es/~mmarques/f47/apun/node32.html.

33. Factibilidad. [En línea] [Citado el: 25 de Marzo de 2010.]
http://www.worldlingo.com/ma/enwiki/es/COCOMO.

Bibliografía:

- **. 1999.** Planning Extreme Programming. s.l. : Addison-Wesley, 1999. 0201710919.
- Acebal, Cesar F. 2005.**Extreme Programming (XP):**Un nuevo método de desarrollo de *Software.* España : Universidad de Oviedo, 2005.
- Beck K. Fowler M. (2000).“**Planeando en Programación Extrema***”*. Addison Wesley. Título original: Planning Extreme Programming.
- C. Batini, S. C. (1994).**Diseño Conceptual de Bases de Datos.** Un enfoque de entidades-interrelaciones. Addison-Wesley / Díaz de Santos.
- Date, C. J. (2003).**Introducción a los Sistemas de Bases de Datos.** La Habana: Féliz Varela.
- Erika Camacho, Fabio Cardeso, Gabriel Núñez.**Arquitecturas De *Software*.** *Guía De Estudio***.** Abril – 2004 [En línea citado el 10-03-2009 Http://Prof.Usb.Ve/Lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.Pdf]
- Grady Booch, James Rumbaugh, Ivar Jacobson. **The UML Modeling Language**User Guide. Addison-Wesley 1999.
  http://ikor.eui.upm.es/jbobi/jbobi/LibroJava2.htm
- Len Bass, Paul Clements, Rick Kazman.***Software* Architecture In Practice**, SecondEdition. Addison-Wesley. Abril 2003.
- **Licencia de Ext JS and Ext GWT**. Recuperado de http://extjs.com/products/license.php
- **Los mejores 12 FrameWorks JavaScript.** Recuperado 20 de diciembre de 2009 de http://www.xperimentos.com/2007/09/04/los-mejores-12-frameworks-javascript
- **Programación orientada a objetos** [En línea]. [Sun, Citado el: 22 de marzo de 2010.]
- Roger S. Pressman**. Ingeniería Del *Software:* Un Enfoque PrácticoParte1.** La Habana 2005.
- Solís, Manuel Calero.**Una explicación de la programación extrema** *(XP).* Madrid : s.n., 2003.

Glosario de Términos

- **Anti-Spam**: Es una aplicación o herramienta informática que se encarga de detectar y eliminar el spam.
- **Anurix:** Empresa de Sistema Informáticos.
- **Escalabilidad:** Propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para extender el margen de operaciones sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.
- **Formulario:** Es una plantilla o página con espacios vacíos que han de ser rellenados con alguna finalidad, por ejemplo una solicitud de empleo.
- **Framework:** Estructura de soporte definida en la cual otros proyectos de *Software* pueden ser organizados y desarrollados.
- **HTML:** (Hyper Text MarkupLanguage) Lenguaje de etiquetas de hipertextos. Es un lenguaje de etiquetas diseñado para estructurar textos y presentarlos en forma de hipertextos, es el formato estándar de las páginas *Web*.
- **Individuo:** Es cada miembro de una especie considerado aisladamente.
- **Interfaz:** Permite el flujo de información entre un usuario y la aplicación.
- **JAVA:** Lenguaje de programación multiplataforma desarrollado por Sun Microsystem.
- **Log:** Archivo en el cual se registra toda la actividad realizada en la red.
- **Marabana:** Popular carrera, dedicada al aniversario 490 de la capital cubana en la que participan corredores tanto cubanos como extranjeros.
- **Navegador:** Es un programa que permite visualizar la información que contiene una página *Web*.
- **PHP:** Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas *Web* dinámicas.
- **Script:** Un script es un tipo de programa que consiste de una serie de instrucciones que serán utilizadas por otra aplicación.
- **SLOC:** (Source Lines of Code) Líneas de códigoFuente.
- ***Software*:** Se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos.
- **Spam**: Correo no deseado.

- **TCP/IP** (Transmission Control Protocol/Internet Protocol) es un conjunto de protocolos de comunicación utilizado para conectar sistemas informáticos a través de Internet (Microsoft Corporation, 1996).
- **URL:** (Uniform Resource Locator Localizador Uniforme de Recursos). Es una secuencia de caracteres de acuerdo a un formato estándar que se usa para nombrar recursos, como documentos e imágenes en internet por su localización.
- **URL:** Es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, videos o presentaciones digitales.
- *Web*: Es una ayuda a la navegación que cataloga páginas que de otra forma compartirían un mismo título.
- **XML:** (Extensible MarkupLanguage). Lenguajes de Marcas Extensibles. Es un metalenguaje extensible de etiquetas.

XP: (Extreme Programming). Programación extrema. Es una metodología ágil de desarrollo de *Software*.