UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

HERRAMIENTA GENERADORA DE PLANTILLAS HTML

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

AUTOR:

Elvis Hernández Pérez

TUTORES:

Ing. Arlan Gálvez Alonso

Ing. José Antonio Soto Pérez

Ciudad de La Habana, junio de 2011

"Año 53 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del presente trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Elvis Hernández Pérez

_____

Firma del Autor

José Antonio Soto Pérez

_____

Firma del Tutor

Arlan Gálvez Alonso

_____

Firma del Tutor

Dedicatoria

Dedicado:

A la memoria de mi abuela Cila y mi abuelo Pipo:

Quienes fueron partícipe del inicio de un sueño que hoy se convierte en realidad.

A mis queridos padres:

Por sacrificar tantos fines de semana sin mi compañía, por apoyarme sin descanso en el largo camino para llegar hasta aquí.

A mi siempre despierto hermano:

Por haberme enseñado incondicionalmente que la perseverancia y el esfuerzo son el camino para lograr los objetivos.

Agradecimientos

Cuando comencé a escribir estos agradecimientos pensé que por descuido podía dejar a alguien importante fuera de la mención, por eso desde ya, pido las disculpas correspondientes en caso de que suceda.

Enrolarme en el desarrollo de una tesis, como lo sería un viaje en galeón hacia un "*Nuevas Mundo*"*,* ha sido para mí un excitante desafío. Tal aventura conlleva años de preparación y esfuerzo continuado, que sin lugar a dudas, sería difícilmente abordable e imposible de llevar a buen puerto sin la colaboración y el aliento de muchas personas.

A veces nos quejamos de las pruebas y las dificultades que nos pone la vida pero en verdad tendríamos que estar agradecidos y ver las cosas positivas de la misma, agradecer por tener una vida, por respirar, por tener una familia, por estar rodeados de personas que nos quieren y nos aman.

A todos aquellos que me han prestado su ayuda de un modo u otro a lo largo de la travesía quiero agradecerles su colaboración y dedicarles un pedacito de esta tesis que sin dudas merecen.

Gracias en primer lugar a mis padres, capitanes experimentados y mentores de este grumete con los que aprendí a esquivar golpes. Les doy gracias además por la financiación de las velas con las que abandoné la tierra firme para echarme a la mar.

Gracias a mi hermano Erick, quien constituye mi ejemplo, porque tal pareciera que estoy predestinado a seguir sus pasos y como siempre le tocó llegar primero, cuando me tocó mi turno el camino era más claro.

Mención especial merecen mis abuelos (los que están y los que no), por quererme tanto, por ser siempre faros en mi vida.

A mis tías, tíos, primas y primos por acompañarme en todos los momentos importantes y por todos los momentos compartidos. Sé que cuento con ellos siempre.

Puedo asegurar sin temor a equivocarme que la Universidad de Ciencias Informáticas ha sido la nave ideal que me ha permitido atravesar las aguas para alcanzar el ansiado objetivo. Quiero expresar mi agradecimiento a sus integrantes, especialmente a los veteranos marinos con los que he combatido mano a mano en mil batallas y a todos los profesores que hicieron de mí un buen ingeniero y una mejor persona.

Gracias también a los críticos y sobre todo pacientes tutores José Antonio Soto y Arlan Gálvez por hacerme rectificar el rumbo cuando me desviaba de la ruta trazada.

Agradecimientos

Por último, pero no menos importante, gracias a los amigos de verdad que me han animado a continuar con tesón cuando el azar era adverso y las fuerzas podían flaquear.

Hoy hemos avistado un pedazo de tierra. Parece que el viaje llega a su fin: ¿Serán estas las cálidas playas del *Nuevas Mundo*?...Pronto lo sabremos.

GRACIAS A CADA UNO DE USTEDES…!!!

Resumen

El desarrollo de plantillas HTML es una tarea que consume grandes cantidades de recursos. En la Universidad de las Ciencias Informáticas no se ha desarrollado ningún producto que contribuya a la creación de plantillas HTML. Para dar solución al problema mencionado anteriormente se realizó la investigación que lleva como título “Herramienta generadora de plantillas HTML” con el objetivo de reducir drásticamente los tiempos de desarrollo así como los recursos necesarios gracias a las técnicas de generación de código.

Para el logro del objetivo propuesto se realizó una amplia revisión bibliográfica, que permitió elaborar y exponer la fundamentación teórica, las soluciones similares así como las herramientas y tecnologías utilizadas. Se efectuó el análisis, diseño e implementación del sistema, guiado por la metodología de desarrollo XP y utilizando como lenguaje de programación Java. La construcción del sistema propuesto tiene un carácter iterativo, que permite la inserción de nuevas funcionalidades y la rectificación de errores.

Palabras claves: Plantillas, HTML, Interfaz, Java

Índice

Introducción

Con el paso del tiempo la Informática va tomando auge y desarrollándose, a tal punto que actualmente se lleva a cabo una etapa de informatización masiva, en la que la mayoría de los países buscan soluciones rentables a sus problemas apoyándose principalmente en esta ciencia.

El vertiginoso avance de las Tecnologías de la Información y las Comunicaciones (TICs) y su influencia en todos los ámbitos de la sociedad, han permitido no solo el incremento en los resultados de la ciencia, la producción y los servicios, sino también que se reflejan en la forma de actuar y pensar de los individuos, abriendo las puertas a una era totalmente revolucionaria.

Las aplicaciones basadas en sistemas de información cuidan, cada vez más, sus interfaces de usuario.

Como consecuencia directa, las interfaces de usuario entregadas en productos finales están cada vez más y más cuidadas para ofrecer al usuario sensaciones de seguridad, fiabilidad, sencillez de uso y precisión en la aplicación suministrada.

En este sentido, una cuidada presentación es esencial para promover su venta. No es de extrañar que, ante tal panorama, en la producción de software comercial cada vez sea más frecuente la inversión de mayores esfuerzos en el desarrollo de interfaces de usuario de alta calidad.

Con el surgimiento de la WEB (*World Wide Web*, Red Mundial de páginas o Documentos de texto entrelazados) las posibilidades son ilimitadas, la utilizan los profesionales, consultores, escritores, clubes, clínicas, hospitales, centros de servicio y soporte técnico, compañías de seguros, bancos, compañías de ventas de cualquier tipo de producto, universidades, centros profesionales y empresas en general.

Estas páginas Web pueden estar desarrolladas en el lenguaje HTML (*Hyper Text Markup Language*, Lenguaje de Marcado de Hipertexto). La preocupación de las empresas es la estética, en lugar de cómo se va a hacer el negocio.

La Universidad de las Ciencias Informáticas (UCI), cuenta con varias facultades enfocadas a la producción de software para uso nacional y de exportación.

Para el desarrollo de plantillas HTML las herramientas más utilizadas son propietarias lo que genera un problema para la universidad dado que la innovación para adaptarlas a necesidades particulares es ilegal y un derecho exclusivo de la compañía fabricante trayendo como consecuencia que no se optimice ni se

facilite el trabajo. La misma se encuentra inmersa en el proceso de migración a software libre y no cuenta con ninguna herramienta de su propiedad que permita generar las plantillas HTML para el desarrollo de aplicaciones.

Para ello se necesita desarrollar un producto que permita generar esas plantillas HTML de forma sencilla reduciendo el número de parámetros que deba introducir el usuario y proporcionando un alto nivel de abstracción para el diseño de las mismas.

Expuesta la situación problémica existente en la Universidad de Ciencias Informáticas se plantea el siguiente **problema a resolver:**

¿Cómo facilitar la creación de plantillas HTML?

Objetivo general:

Desarrollar una herramienta libre para generar plantillas HTML.

Objeto de estudio:

Procesos de generación de plantillas HTML.

Campo de acción:

Procesos de generación de plantillas HTML para aplicaciones relacionadas con la formación estudiantil.

Objetivos específicos:

- Investigar el estado del arte y definir la posición del investigador respecto al tema.
- Analizar los métodos de generación de plantillas HTML.
- Identificar los requisitos funcionales y no funcionales que debe tener la herramienta.
- Analizar y diseñar la herramienta, de forma que cumpla con los requerimientos planteados.
- Implementar la herramienta.
- Validar la propuesta de solución sometiéndola a disímiles pruebas.

Tareas de la investigación:

- Investigación sobre herramientas destinadas a la generación de plantillas HTML.
- Análisis de cómo funcionan los métodos de generación de plantillas HTML.

- Determinación de las características y funcionalidades que debe tener la herramienta que se desea implementar para la generación de plantillas HTML.
- Selección de las herramientas y tecnologías informáticas que se van a utilizar y justificación para el uso de las mismas.
- Análisis y diseño de la herramienta a desarrollar.
- Desarrollo de la herramienta basada en el análisis y diseño propuesto.
- Comprobación de las funcionalidades de la herramienta de forma que cumpla con los requisitos planteados para la misma.

Idea a defender:

Una herramienta libre para generar plantillas HTML y que permita con pocos conocimientos técnicos, realizar diseños de calidad y estandarizados.

Métodos científicos utilizados:

Métodos teóricos

Analítico-Sintético: facilitó el resultado del trabajo con el análisis de generación de plantillas HTML, y posibilitó sintetizar las características esenciales, como también permitió arribar a las conclusiones de la investigación.

Histórico-Lógico: permitió estudiar la evolución de las herramientas generadoras de plantillas HTML además de las diferentes formas de generarlas.

Métodos empíricos

Entrevista: se empleará para obtener información sobre las necesidades en el Laboratorio de Producción de Recursos Didácticos, con el propósito de conocer las funcionalidades que debe tener la herramienta a desarrollar. Esta se emplea al inicio y durante todo el desarrollo del trabajo, pues a medida que se desarrolla la herramienta se analiza si cumple las expectativas requeridas.

Estructura capitular:

Capítulo 1: Fundamentación teórica

Contiene todo lo referente al análisis y definición de las herramientas y la metodología a utilizar para el desarrollo de la solución, además se incluyen algunos conceptos importantes para facilitar el entendimiento del trabajo.

Capítulo 2: Desarrollo de la solución

En este capítulo se muestra el de cursar del software con las fases de desarrollo Exploración y Planificación, propias de la metodología seleccionada para guiar el proceso, obteniéndose los artefactos propios de cada fase.

Capítulo 3: Implementación y pruebas

En este capítulo se aborda la fase de implementación, con la cual se finaliza el proceso de desarrollo y por tanto la propuesta que trae el presente trabajo. También se escribirán pruebas para chequear el correcto funcionamiento del programa dado que no puede existir ninguna característica que no pueda ser probada.

Capítulo 1 Fundamentación Teórica

1.1 Introducción

En el presente capítulo se hace referencia a los elementos teóricos que fundamentan el desarrollo del sistema informático propuesto. También contempla un análisis efectuado sobre el estudio del arte de las posibles herramientas y tecnologías que pudieran ser utilizadas para el desarrollo, posibilitando la toma de decisiones sobre qué es más factible utilizar en función de concluir los objetivos generales de la investigación y solución del problema planteado.

1.2 Marco teórico

Definición de una interfaz gráfica de usuario

En Ciencias de la Computación, dentro de la disciplina de la interacción Persona-Computador, se define la GUI (*Graphics User Interface,* Interfaz Gráfica de Usuario), como el medio de interacción entre un usuario y un sistema informático que se realiza mediante el lenguaje visual. Esta interfaz debe proveer al usuario un ambiente agradable y sencillo para el correcto entendimiento y ejecución del programa. (1)

La interfaz gráfica de usuario es la evolución de los antiguos programas de líneas de comandos hasta lo que se observa en nuestros días.

Definición de los diseñadores de GUI

Los diseñadores de interfaces gráficas de usuarios, son herramientas de software que permiten generar interfaces gráficas de usuario mediante un lenguaje cercano al lenguaje natural. (1)

Los diseñadores de GUI modernos utilizan, a su vez, una interfaz gráfica para comunicarse con el desarrollador, es decir, el desarrollador crea su interfaz mediante el lenguaje gráfico proveído por la herramienta (paletas de componentes, acciones de “arrastrar” y “soltar”, presión de botones, selectores de colores, etc.).

1.3 Estado del arte

Aplicaciones dedicadas a la generación de plantillas HTML

Artisteer

Artisteer es un programa de ordenador para hacer plantillas de sitios web. Con él, se pueden crear plantillas sin tener conocimientos de programación, permite elegir entre las plantillas de la propia biblioteca o hacer otras nuevas, el programa genera el código XHTML (este es el acrónimo de *Extensible Hypertext Markup Language,* Lenguaje Extensible de Marcado de Hipertexto) y el CSS (*Cascading Style Sheets,* Hojas de Estilo en Cascada) para nuevas plantillas.

Dentro de las principales características que brinda se encuentran:

- Creación de plantillas para páginas web sencillas.
- Creación de plantillas para Wordpress, Drupal o Joomla.
- Código HTML y CSS en conformidad con estándares web.
- Interfaz simple e intuitiva.
- Es una aplicación fácil de utilizar.
- Soporte en diferentes idiomas.
- Actualización automática.
- Soporte para instalación y desinstalación.
- No consume muchos recursos del sistema y no ocupa mucho espacio en el disco duro. (2)

Amaya

Editor y navegador web de código abierto. Amaya es una herramienta que tiene por objetivo la creación y actualización de documentos XHTML directamente en la web.

Comenzó como un editor de HTML + CSS, sus capacidades se fueron extendiendo hasta soportar XML (*Extensible Markup Language,* Lenguaje de Marcas Extensible). Se pueden manipular páginas web complejas, con formularios, tablas y las características más avanzadas del XHTML. Brinda la posibilidad de crear y editar expresiones matemáticas complejas dentro de las páginas web, además de asociar estilos a los documentos mediante CCS.

Es importante destacar de este editor que casi siempre genera un código limpio, algo esencial para aprender a crear sitios web. (3)

Dentro de sus principales características se encuentran:

- Multiplataforma.
- Renderizado de imágenes.
- Herramientas para imágenes vectoriales.
- Edición WYSIWYG (*What You See Is What You Get,* Lo que ves es lo que obtienes).
- Gratuito. (4)

NVU

Ofrece una amplia variedad de herramientas para crear de forma cómoda una página web, un entorno de edición WYSIWYG intuitivo.

Para los colores dispone de un editor muy fácil de utilizar junto a un editor CSS eficaz para principiantes con escasas nociones de hojas de estilo.

Incluye otras interesantes características como la gestión de proyectos web o la posibilidad de personalizar la interfaz y barras de herramientas. Además NVU está disponible para Linux, Mac OS X y Microsoft Windows. (5)

Como principales funcionalidades se encuentran:

- Multiplataforma
- Cliente FTP.
- Edición de código fuente.
- Marcos, formularios, tablas, plantillas de diseño, hojas de estilo CSS.
- Gratis.

Dreamweaver

Esta es una herramienta que soporta gran cantidad de tecnologías, además muy fáciles de utilizar:

- Hojas de estilo y capas.
- Javascript para crear efectos e interactividades.
- Inserción de archivos de multimedia.

Además, es un programa que se puede actualizar con componentes, que fabrica tanto Macromedia como otras compañías, para realizar otras acciones más avanzadas.

En resumen, el programa es realmente satisfactorio, incluso el código generado es de buena calidad. (6)

Dentro de sus principales características se pueden mencionar las siguientes:

- Versiones para Windows y MacOS-X.
- Ventana de visualización.
- Sugerencias de código.
- Herramientas avanzadas para imágenes de Photoshop.
- Utiliza CSS.
- Diseño sin escritura de código (aunque el usuario puede modificar el código manualmente).
- Software propietario.

Aportes del estudio de otras aplicaciones a la propuesta de solución

Se puede llegar a la conclusión de que el uso de herramientas de diseño de GUI trae importantes ventajas y desventajas. Las ventajas principales son el desarrollo rápido y la estandarización de componentes, las desventajas principales se refieren a la restricción del dominio funcional con respecto a los lenguajes y librerías subyacentes y la necesidad de ajustarse al paradigma proporcionado por la herramienta.

El desarrollador de software debe conocer estas herramientas porque es necesario sopesar si para un proyecto determinado pueden ser aplicables dado que pueden aumentar el tiempo de desarrollo de la aplicación que se desee realizar.

1.4 Tendencias y tecnologías actuales

1.4.1 Tipos de aplicaciones

Existen distintos tipos de aplicaciones, dentro de estas se encuentran las aplicaciones Web y las de escritorio.

Aplicaciones Web

Una aplicación Web es una aplicación informática que los usuarios pueden utilizar accediendo a un servidor Web a través de Internet o de una intranet mediante un navegador. Es un software que se

codifica en un lenguaje soportado por los navegadores Web en la que se confía la ejecución al navegador. Las aplicaciones web presentan numerosas ventajas y a su vez desventajas.

Ventajas

- Se pueden utilizar desde cualquier lugar.
- No requieren hacer actualizaciones en los clientes.
- No hay problemas de incompatibilidad entre versiones, porque todos trabajan con la misma.
- No necesitan instalar nada en el cliente, agregar una nueva terminal solo requiere poner una computadora nueva.
- No se obliga a utilizar determinado sistema operativo. (7)

Desventajas

- Requiere conexión a la red.
- Se pierde tiempo de desarrollo haciéndola compatible con los diferentes navegadores, los frameworks[1] ayudan a solventar estos problemas.
- Su tiempo de respuesta es más lento, esto ha mejorado utilizando tecnologías como AJAX. (8)

Aplicaciones de escritorio

Las aplicaciones de escritorio también poseen ventajas y desventajas.

Ventajas:

- Pueden ser más robustas.
- Tiempo de respuesta más rápido.
- Mayor capacidad gráfica visual.
- Mayor personalización. (9)

Desventajas:

- Requieren instalación.
- Generalmente se hacen para un sistema operativo específico.
- Se requiere actualizar en cada cliente.

[1] un ***framework*** es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de *software* concretos, con base en la cual otro proyecto de *software* puede ser organizado y desarrollado.

Selección del tipo de aplicación a utilizar

Se seleccionan las aplicaciones de escritorio ya que mantienen un contacto permanente entre los procesos internos del programa y lo que sucede en la interfaz de usuario, además, le proporcionan al usuario un mayor control sobre las configuraciones y el trabajo con documentos.

1.4.2 Arquitectura

Patrón de diseño Modelo Vista Controlador (MVC)

Describe una forma muy utilizada en el Web de organizar el código de una aplicación separando los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Modelo: Encargado del acceso a datos.

Vista: Define la interfaz de usuario, HTML (Lenguaje de Marcado Hipertextual) +CSS (Hojas de Estilo en Cascada). Recibe datos del modelo y la muestra al usuario.

Controlador: Responde a eventos y modifica la vista y el modelo.

Este modelo presenta ventajas dado que:

- Podemos modificar uno de los componentes sin conocer cómo funcionan los demás involucrados.
- El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.
- Soporte de múltiples vistas dado que la vista se halla separada del modelo y no existe una dependencia directa entre ambos, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos mostrado de modos diferentes.

Pero a su vez presenta desventajas:

- El tiempo de desarrollo de una aplicación que implementa el patrón de diseño modelo-vista-controlador es mayor, al menos en la primera etapa, que el tiempo de desarrollo de una aplicación que no lo implementa.

- El costo de actualizaciones frecuentes: Si el modelo experimenta cambios frecuentes podría desbordar las vistas con una lluvia de requerimientos de actualización. (8)

Arquitectura 3 capas o programación 3 capas

Consiste literalmente en separar un proyecto en Capa de Presentación, Capa de Negocio y Capa de Datos. Esto permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API (Interfaz de Programación de Aplicaciones) que existe entre niveles. (10) Su ventaja principal es que en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

Capas

Capa de presentación: Esta es la parte que se le muestra al usuario, las pantallas que se le muestran para que él interaccione con el programa.

Capa de negocio: Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todos los procesos que deben realizarse.

Capa de datos: Es donde se encuentran los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

A continuación se muestra una imagen representativa la arquitectura en 3 capas.

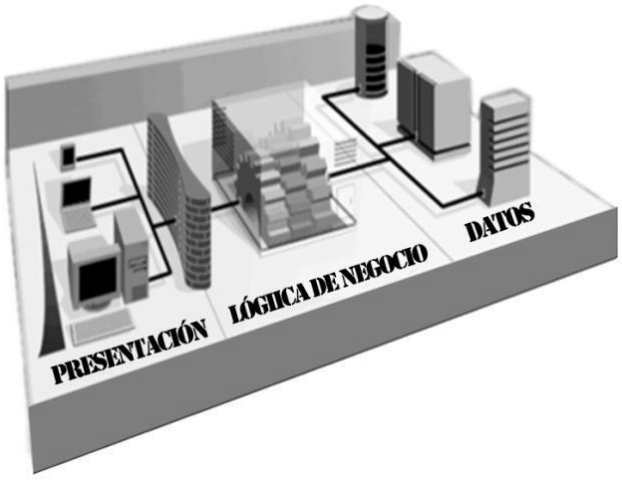

Figura 1 Arquitectura 3 capas

Arquitectura Cliente-Servidor

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. (11)

En la arquitectura cliente-servidor el remitente de una solicitud es conocido como cliente. Sus características son:

- Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo).
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Al receptor de la solicitud enviada por cliente se conoce como servidor. Sus características son:

- Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales.

Ventajas:

- **Centralización del control**: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- **Escalabilidad**: se puede aumentar la capacidad de clientes y servidores por separado.
- **Fácil mantenimiento**: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio.

Desventajas

- La congestión del tráfico ha sido siempre un problema
- El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la aplicación es una Web, no podemos escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

Selección del tipo de arquitectura de software a utilizar

Se selecciona para la arquitectura del software el patrón de diseño MVC dado que separa hacia fuera el proceso y es más escalable. Además, si se desea hacer una modificación al modelo del dominio, como aumentar funcionalidades o datos contenidos, solo debe modificarse el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos. Una de las causas por la que no se escoge la arquitectura cliente-servidor es porque al separar dos de las tres capas el cliente puede integrar parte de la funcionalidad del sistema.

1.4.3 Metodologías de desarrollo de software

Todo desarrollo de software es difícil y riesgoso de controlar, y si no se sigue una metodología de por medio, lo que se obtiene son clientes insatisfechos, lo cual crearía una mala reputación para la empresa. Para lograr buenos resultados en el proceso de desarrollo de software es necesario hacer una correcta Ingeniería de Software, aplicando un conjunto de métodos y de herramientas, con el objetivo de obtener un software fiable, de modo rentable, fácil de mantener, a base de un desarrollo sistemático.

Existen un gran número de metodologías utilizadas en estos últimos años, las cuales difieren entre sí, pero todas tienen en común la división del proceso en etapas, coincidiendo en el ciclo de desarrollo de un software: análisis, diseño, implementación y prueba.

Programación Extrema (XP)

XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

- El cliente define el valor de negocio a implementar.
- El programador estima el esfuerzo necesario para su implementación.
- El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
- El programador construye ese valor de negocio.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración. El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.

- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

SCRUM

SCRUM es una de las metodologías ágiles más conocidas para la gestión de proyectos. Es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Se utiliza como marco para otras prácticas de ingeniería de software como RUP (Proceso Unificado de Desarrollo) o XP.

Potencia la formación de equipos de trabajos autosuficientes y multidisciplinares, reduciendo la carga de gestión y proporcionando a los miembros del equipo un entorno amigable y productivo para desarrollar sus habilidades al máximo. Este entorno proporciona además mayor calidad de vida a los trabajadores y mejora drásticamente la moral en las organizaciones.

RUP

Es una metodología para la ingeniería de software que va más allá del mero análisis y diseño orientado a objetos, para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. El resultado es un proceso basado en componentes, dirigido por casos de uso, iterativo e incremental y centrado en la arquitectura. Es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software, sin embargo, el RUP es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y tamaños de proyecto. El Proceso Unificado se basa en componentes, lo que significa que el sistema en construcción está hecho de componentes de software interconectados por medio de interfaces bien definidas. Utiliza el Lenguaje Unificado de Modelado (UML) en la preparación de todos los planos del sistema. Entre sus principales características se pueden encontrar:

- Unifica los mejores elementos de metodologías anteriores.
- Orientado a Objetos.
- Utiliza el UML como lenguaje de representación visual.

Como es un proceso en su modelación describe entre sus principales elementos a trabajadores, artefactos, actividades y flujo de actividades o sea quién, qué, cómo y cuándo.

El ciclo de vida

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. La arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desenvuelve fundamentalmente algunos más que otros.

Selección de la metodología de desarrollo de software

Dentro de las metodologías mencionadas anteriormente no se decide escoger SCRUM debido a que no genera toda la evidencia o documentación de otras metodologías, no es apto para todos los proyectos, tal vez sea necesario complementarlo con otros procesos.

La metodología RUP no se escoge porque está diseñada para proyectos de desarrollo grandes y con un amplio equipo de desarrolladores, donde la documentación es un factor importante así como la realización de gran volumen de artefactos, aspectos que demorarían la entrega del producto en un plazo de tiempo corto.

Se decide seleccionar XP como metodología de desarrollo, ya que el plazo de entrega del proyecto es corto, además los requerimientos actuales pueden estar sujetos a futuros cambios y éste es flexible a los mismos, ya que permite administrar las modificaciones de una forma óptima.

Se escoge en gran medida XP porque es la metodología ágil de más renombre en la actualidad, se diferencia de las demás metodologías que forman este grupo en un aspecto en particular: el alto nivel de

disciplina de las personas que participan en el proyecto. Propone que la comunicación y la satisfacción del cliente es lo principal, y no se hace mucho énfasis en la documentación, es más importante definir los requerimientos y las pruebas de calidad. También está diseñada para el trabajo de equipos pequeños.

1.4.4 Lenguaje unificado de modelado (UML)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software orientado a objetos (OO). Entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. Es un sistema notacional destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Puede utilizarse para modelar distintos tipos de sistemas: de software, de hardware y organizaciones del mundo real. (12)

De sus características se puede decir que divide cada proyecto en un número de diagramas que representan las distintas vistas del proyecto y juntos representan la arquitectura del mismo. Permite describir un sistema en diferentes niveles de abstracción, simplificando la complejidad sin perder información, para que los usuarios y desarrolladores comprendan las características de la aplicación. Ofrece nueve diagramas para modelar sistemas:

- De Casos de Uso para modelar los procesos de negocio.
- De Secuencia para modelar el paso de mensajes entre objetos.
- De Colaboración para modelar interacciones entre objetos.
- De Estado para modelar el comportamiento de los casos de uso, objetos y operaciones.
- De Actividad para modelar el comportamiento de los casos de uso, objetos u operaciones.
- De Clases para modelar la estructura estática de las clases en el sistema.
- De Objetos para modelar la estructura estática de los objetos en el sistema.
- De Implementación para modelar la distribución del sistema.
- De Componentes para modelar componentes.

Ventajas

Hoy en día, UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mientras más complejo es el sistema que se desea crear más beneficios presenta el uso de UML, las razones de esto son evidentes:

- Diseño y documentación.
- Código reutilizable.
- Descubrimiento de fallas.
- Ahorro de tiempo en el desarrollo del software.
- Mucho más fáciles las modificaciones.
- Más fácil comunicación entre programadores.

UML es un lenguaje expresivo, claro y uniforme, que no garantiza el éxito de los proyectos, pero si mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios.

1.4.5 Herramientas de modelado

CASE es una sigla, que corresponde a las iniciales *Computer Aided Software Engineering*; y en su traducción al español significa Ingeniería de Software Asistida por Computación. El concepto de herramienta CASE es muy amplio; y una buena definición genérica sería la de considerarla, como la aplicación de métodos y técnicas a través de las cuales se les hace útil a las personas comprender las capacidades de las computadoras por medio de programas, de procedimientos y su respectiva documentación. Las herramientas CASE representan una forma que permite modelar los procesos del negocio de las empresas y desarrollar los Sistemas de Información Gerenciales. Debido a la gran demanda que tienen estas herramientas, la exigencia en cuanto a su uso ha ido aumentando, por lo que toda herramienta CASE debe entre otras cosas:

- Proporcionar topologías de aplicaciones flexibles y portátiles.
- Brindar un control de versiones.
- Crear código compilado en el servidor.
- Dar un soporte multiusuario.
- Ofrecer seguridad.

Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. (13) Esta herramienta posee unas características gráficas muy cómodas que facilitan la realización de los diagramas siguiendo como lenguaje de modelado UML.

Algunas de sus principales características son:

- Producto de calidad.
- Soporta aplicaciones web.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDE´s (*Integrated Development Environment*, Entorno Integrado de Desarrollo).
- Disponibilidad de múltiples plataformas.
- Ofrece una serie de facilidades para generar informes que permiten documentar el proyecto.

Rational Rose

Es una de las más poderosas herramientas de modelado visual para el análisis y diseño de sistemas orientados a objetos (14). Se utiliza para modelar un sistema antes de proceder a construirlo. Cubre todo el ciclo de vida de un proyecto:

- Concepción y formalización del modelo.
- Construcción de los componentes.
- Transición a los usuarios.
- Certificación de las distintas fases.

Es la herramienta CASE que comercializan los desarrolladores de UML, la cual permite el completamiento de una gran parte de los flujos fundamentales del RUP como son:

- Modelado del negocio.
- Captura de requisitos.
- Análisis y Diseño.
- Implementación.
- Gestión de Configuración y Control de Cambios.

Características

- Permite especificar, analizar, diseñar el sistema antes de codificarlo.
- Mantiene la consistencia de los modelos del sistema de software.
- Chequeo de la sintaxis UML.
- Generación y documentación automática.
- Generación de código a partir de los modelos.
- Ingeniería Inversa.
- Esta herramienta contiene 4 vistas:
  - Vista de Caso de Uso.
  - Vista Lógica.
  - Vista de Componente.
  - Vista de Despliegue.

Selección de la herramienta de modelado

Después de haber analizado las características de las herramientas de modelado expuestas anteriormente, se concluye que ambas son muy utilizadas a nivel mundial y facilitan el trabajo durante el desarrollo de un producto de software garantizando su calidad final. Por tanto, ambas son dos herramientas potentes y fáciles de usar.

Se selecciona Visual Paradigm en su versión 6.4 como herramienta CASE por su integración a UML, además de portar con las características de ser multiplataforma, amigable en su uso y poseer interoperabilidad con otras aplicaciones e integración con distintos Entornos de Desarrollo

Integrado.Visual Paradigm es un producto que facilita a las organizaciones la creación de sus bases de datos y acelera al máximo la implementación del mismo.

1.4.6 Lenguajes de programación

Actualmente existen diferentes lenguajes de programación que han ido surgiendo debido a las tendencias y necesidades que presentan las plataformas.

Un lenguaje de programación es una construcción del ser humano para expresar programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos con sentido único y una regla principal que resume las demás. Para que esta construcción sea operable en un computador debe existir otro programa que controle la validez o no de lo escrito. A este programa se le llama traductor (15).

CSharp

C# es un lenguaje de propósito general orientado a objetos creado por Microsoft para su plataforma .NET. Su sintaxis básica se deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes. C# fue diseñado para combinar el control abajo nivel de lenguajes como C y la velocidad de programación de lenguajes como Visual Basic. Dentro de sus principales características encontramos:

- Orientado a objetos.
- Orientado a componentes.
- Recolección de basura.
- Seguridad de tipos.
- Instrucciones seguras. (16)

Java

Java es un lenguaje de programación orientado a objetos. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Ventajas

- Es independiente de la plataforma de desarrollo.

- Existen dentro de su librería clases gráficas como awt y swing, las cuales permiten crear objetos gráficos comunes altamente configurables y con una arquitectura independiente de la plataforma.
- Java permite a los desarrolladores aprovechar la flexibilidad de la Programación.
- Orientada a Objetos en el diseño de sus aplicaciones.
- Simple: Elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. Orientado a Objetos.
- Familiar: Como la mayoría de los programadores están acostumbrados a programar en C o en C++, la sintaxis de Java es muy similar al de éstos.
- Robusto: El sistema de Java maneja la memoria de la computadora. No se tiene que preocupar por apuntadores, memoria que no se esté utilizando, etc.
- Seguro: El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje.
- Portable: Como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.
- Dinámico: Java no requiere que se compilen todas las clases de un programa para que funcione. (17)

Desventajas

- Hay diferentes tipos de soporte técnico para la misma herramienta, por lo que el análisis de la mejor opción se dificulta.
- Para manejo a bajo nivel deben utilizarse métodos nativos, lo que limita la portabilidad.
- El diseño de interfaces gráficas con awt y swing no son simples.

Python

Es un lenguaje de programación comparado habitualmente con Perl. Los usuarios lo consideran como un lenguaje bueno para programar. Permite la creación de todo tipo de programas incluyendo los sitios web.

Su código no necesita ser compilado. Es un lenguaje de programación multiplataforma, lo cual fuerza a que los programadores adopten por un estilo de programación particular:

- Programación orientada a objetos.

- Programación estructurada.
- Programación orientada a aspectos.

Ventajas:

- Libre y fuente abierta.
- Gran cantidad de funciones y librerías.
- Sencillo y rápido de programar.
- Multiplataforma.
- Licencia de código abierto (Open Source).
- Orientado a Objetos.
- Portable. (18)

Desventajas:

- Lentitud por ser un lenguaje interpretado.

Selección del lenguaje de programación

Una vez analizadas las principales características de los lenguajes antes mencionados, se decidió que la solución propuesta se desarrollará en lenguaje de programación Java versión 6, ya que este ofrece una amplia gama de componentes reutilizables que facilitan y agilizan el desarrollo, se puede ejecutar en cualquier sistema (siempre y cuando dicho sistema posea una implementación de la máquina virtual de Java), es libre, robusto y de muy alto nivel, lo que posibilita que el desarrollador se centre más en cubrir los requerimientos del negocio, que en las trabas y especificidades del lenguaje.

C# no genera código nativo y para ejecutar los programas que se realizan, la computadora tiene que tener instalado .NET, además está algo atado a Microsoft y a Windows. Otra de las razones por la que se decide utilizar para el desarrollo de la aplicación el lenguaje Java es porque Python al no ser un lenguaje compilado se hace más lento.

1.4.7 Entorno de Desarrollo Integrado (IDE)

Un IDE es un entorno de programación creado como un programa de aplicación con un conjunto de herramientas para el programador: un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Estos pueden ser aplicaciones independientes o pueden formar parte de aplicaciones

existentes, para un lenguaje de programación específico o multilenguaje. Proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación y puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. (19)

Netbeans

Es un IDE gratuito. Puede obtener todas las herramientas que necesite para crear aplicaciones profesionales para el escritorio, la empresa, la web y equipos móviles con el lenguaje Java, C/C++, y Ruby. Se ejecuta en varias plataformas incluyendo Windows, Linux y Mac OS X y Solaris. Provee varias características como: funciones de edición enriquecida de Javascript, soporte para el uso del framework web Spring, y mejor integración con MySQL.

Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .net 2002).

Selección del IDE

Se selecciona como IDE al Netbeans 6.9 dado que está orientado hacia la librería gráfica Swing de Java que es precisamente lo que se necesita para el desarrollo de la aplicación, además de ser una herramienta gratuita y brindar numerosos servicios a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación.

1.5 Conclusiones

El profundo análisis realizado por la constante búsqueda de bibliografía y el constante estudio de las tecnologías actuales, han posibilitado que se alcance el conocimiento necesario acerca de las ventajas de las herramientas más utilizadas para el desarrollo de la aplicación en cuestión. Asentándose en este conocimiento se puede concluir que para el desarrollo de la aplicación se utilizará como lenguaje de

programación Java y como IDE de desarrollo el Netbeans, todo esto rigiéndose por la metodología XP para lograr una buena documentación.

Capítulo 2 Desarrollo de la solución

2.1 Introducción

Este capítulo está enmarcado en las tres primeras fases definidas en la metodología XP: exploración, planificación e iteraciones. Abordará temas relacionados con las historias de usuario y la planificación del proyecto. Se determinan los objetos que serán automatizados para conformar una propuesta del sistema, y se definen las iteraciones, constituyéndose el Plan de Entrega.

2.2 Propuesta del sistema

El presente trabajo propone la implementación de un sistema que permita trabajar sobre plantillas prediseñadas.

La aplicación constará inicialmente con un diseño de plantilla con el fin de ahorrar tiempo. El usuario podrá crear una plantilla donde se le brindará la posibilidad de crear una nueva estructura y personalizar el fondo de la misma. También permitirá cambiar la imagen del banner[2], personalizar el menú superior cambiando el color, al igual que la tipografía. De los menús laterales se podrá modificar el color así como la cantidad.

Se podrán gestionar los elementos y espacios en el contenido del trabajo, se le ofrecerá la posibilidad de agregar fotos junto al texto que él disponga.

La aplicación le permitirá al usuario la exportación del trabajo en formato HTML. Si no se ha finalizado un trabajo se le brindará la opción de salvarlo con el fin de poder continuar en otro momento. También presentará las opciones de ayuda y la de ver lo que se va desarrollando.

2.3 Requisitos no funcionales (RNF)

Los requisitos no funcionales no representan qué debe hacer el sistema pero tienen igual importancia porque a través de ellos se pueden ver las diferentes características del sistema, como apariencia, software, hardware, herramientas utilizadas que están asociados. A continuación se mencionan los requerimientos no funcionales.

RNF Apariencia o interfaz gráfica:

[2]Un banner (en español: banderola) es un formato publicitario en Internet. Consiste en incluir una pieza publicitaria dentro de una página web.

La aplicación propuesta será utilizada por personas que tengan un conocimiento medio de informática, por lo que la interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo:

- Utilizar botones que expresen su función.
- Idioma de la aplicación en español.
- Área de edición.
- Menú con diferentes elementos.

RNF Software:

Es necesario contar con una máquina virtual de java instalada en el equipo.

La PC (*Personal Computer,* Computadora Personal) del cliente puede tener instalado cualquier sistema operativo debido a que el sistema será multiplataforma.

RNF de diseño e implementación:

- Se utilizará el lenguaje Java o extensiones de este.
- Como estándar de codificación se utilizarán las convenciones de código inherentes del lenguaje java.
- Se debe generar la documentación de todas las clases, métodos y recursos creados.
- La solución debe ser extensible de acuerdo a las necesidades de uso.
- La solución deberá poder ejecutarse en diferentes ambientes de desarrollo.

RNF Portabilidad:

- La solución debe ser independiente de la plataforma en que se utilice, propiedad que hereda del lenguaje utilizado.

2.4 Personas relacionadas con el sistema

Se define como persona relacionada al sistema toda aquella que obtiene un resultado del valor de uno o varios procesos que se ejecutan en el mismo, además de aquellas que se encuentran involucradas en dichos procesos, pues participan en ellos pero no obtienen ningún resultado de valor. (20)

Personas relacionadas con el sistema

Personas relacionadas con el sistema	Justificación
Usuario	Es la persona que va a interactuar con el producto

Tabla 1: Personas relacionadas con el sistema

2.5 Fase de exploración

En esta etapa se definen las Historias de Usuarios (HU), las cuales constan de 3 ó 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 y el resultado es una visión general del sistema.

2.5.1 Historias de Usuario

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.

No hay que preocuparse si en un principio no se identifican todas las HU. Al comienzo de cada iteración estarán registrados los cambios en las HU y según eso se planificará la siguiente iteración.

HU: Crear plantilla

Historia de Usuario	
Número: 1	**Nombre:** Crear plantilla
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Riegos en Desarrollo:** Alto
Descripción: El usuario para poder comenzar el desarrollo de su interfaz debe crear una plantilla, estas ya están prediseñadas para ahorrarle tiempo de trabajo al mismo.	

Observaciones:

Tabla 2: Representación de la HU Crear plantilla

HU: Personalizar colores

Historia de Usuario	
Número: 2	**Nombre:** Personalizar colores
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Riegos en Desarrollo:** Medio
Descripción: El usuario puede cambiar los colores de las diferentes áreas de la página	
Observaciones:	

Tabla 3: Representación de la HU Personalizar colores

HU: Personalizar imágenes

Historia de Usuario	
Número: 3	**Nombre:** Personalizar imágenes
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Riegos en Desarrollo:** Medio
Descripción: El usuario puede modificar las imágenes del encabezado de la página así como añadir algunas nuevas.	

Observaciones:

Tabla 4: Representación de la HU Personalizar imágenes

HU: Cambiar tipografía

Historia de Usuario	
Número: 4	**Nombre:** Cambiar tipografía
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Riegos en Desarrollo:** Bajo
Descripción: El usuario puede modificar el tipo de letra del título, de la barra de navegación, del menú superior, de los menús laterales.	
Observaciones:	

Tabla 5: Representación de la HU Cambiar tipografía

HU: Gestionar elementos

Historia de Usuario	
Número: 5	**Nombre:** Gestionar elementos
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Riegos en Desarrollo:** Alto
Descripción: El usuario puede gestionar los elementos y espacios en el diseño para cada tipo de	

plantilla.
Observaciones: Esto incluye agregarlos, modificarlos o eliminarlos.

Tabla 6: Representación de la HU Gestionar elementos

HU: Exportar plantilla

Historia de Usuario	
Número: 6	**Nombre:** Exportar plantilla
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Riegos en Desarrollo:** Alto
Descripción: El usuario puede exportar la plantilla diseñada en formato HTML.	
Observaciones:	

Tabla 7: Representación de la HU Exportar plantilla

HU: Salvar plantilla

Historia de Usuario	
Número: 7	**Nombre:** Salvar plantilla
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Riegos en Desarrollo:** Bajo
Descripción: El usuario puede salvar las plantillas que diseñe.	

Observaciones:

Tabla 7: Representación de la HU Salvar plantilla

HU: Cargar plantilla

Historia de Usuario	
Número: 8	**Nombre:** Cargar plantilla
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Riegos en Desarrollo:** Bajo
Descripción: El usuario puede cargar cualquier plantilla que se haya realizado y salvado previamente con dicha herramienta.	
Observaciones:	

Tabla 8: Representación de la HU Cargar plantilla

HU: Ver

Historia de Usuario	
Número: 9	**Nombre:** Ver
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 2
Prioridad en Negocio: Baja	**Riegos en Desarrollo:** Alto
Descripción: El usuario puede el trabajo a medida que lo va desarrollando.	

Observaciones:

Tabla 9: Representación de la HU Ver

HU: Ayuda

Historia de Usuario	
Número: 10	**Nombre:** Ayuda
Usuario: Usuario	
Modificación de la Historia de Usuario:	**Iteración Asignada:** 2
Prioridad en Negocio: Baja	**Riegos en Desarrollo:** Bajo
Descripción: El usuario puede auxiliarse de la ayuda que la herramienta brinda para un mejor conocimiento de la misma.	
Observaciones:	

Tabla 10: Representación de la HU Ayuda

2.6 Fase de Planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo (5 días) donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción.

2.6.1 Estimación de esfuerzo por historia de usuario

Para el desarrollo del sistema propuesto en este trabajo se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, llegándose a los resultados que se muestran en la siguiente tabla.

Estimación de esfuerzo por HU

Historia de usuario	Puntos Estimados
Crear Plantilla	2
Personalizar colores	1
Personalizar imágenes	1
Cambiar tipografía	1
Gestionar elementos	1
Exportar Plantilla	2
Salvar Plantilla	1
Cargar Plantilla	1
Ver	2
Ayuda	1

Tabla 10: Estimación de esfuerzo por HU

2.6.2 Plan de iteraciones

Una vez identificadas las historias de usuario del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas se procede a la planificación de la etapa de implementación del proyecto. De acuerdo con lo mencionado anteriormente se decidió realizar esta en dos iteraciones, detalladas a continuación.

Iteración 1

En esta iteración se realizarán las historias de usuarios que van a dar una idea de cómo quedará la aplicación aunque todavía estará en sus inicios.

Iteración 2

La implementación de las historias de usuarios en esta iteración proporcionará una idea completa de la aplicación la cual al terminar la implementación quedará terminado el sistema.

2.6.3 Plan de duración de las iteraciones

Como parte del ciclo de vida de un proyecto utilizando XP se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con que se cuenta. Este plan se encarga de mostrar las historias de usuario que serán abordadas en cada una de las iteraciones, así como la duración estimada de estas últimas y el orden en que se implementarán.

Plan de duración de las iteraciones

Iteración	Orden de las historias de usuarios	Duración de las iteraciones
Iteración 1	Crear plantilla Personalizar colores Personalizar imágenes Cambiar tipografía Gestionar elementos Exportar Plantilla	8 semanas
Iteración 2	Salvar Plantilla Cargar Plantilla Ver Ayuda	5 semanas

Tabla 11: Plan de duración de las iteraciones

2.6.4 Plan de entregas

En el Plan de Entrega se crea una planificación donde entre los desarrolladores y clientes se deciden los tiempos de implementación para cada una de las iteraciones y la prioridad con la que serán implementadas

Plan de entregas

Entrega	Segunda semana de marzo	Primera semana de mayo
Iteración 1	Versión 1.0	Concluido
Iteración 2	Versión 0.1	Concluido

Tabla 12: Plan de Entregas

2.7 Conclusiones

Durante el desarrollo de este capítulo se hizo referencia a todo lo concerniente a la fase de Exploración y Planificación del sistema, haciendo una descripción de cada uno de los artefactos generados en el transcurso de la misma. Se asume una implementación por etapas la cual fue concebida y debidamente detallada en cada una de las iteraciones expuestas anteriormente para darle cumplimiento a las historias de usuario con que cuenta el producto.

Capítulo 3Implementación y pruebas

3.1 Introducción

Según lo planteado en la metodología XP, la implementación debe realizarse de forma iterativa, obteniéndose después de cada iteración un producto que debe de ser probado y mostrado al cliente, permitiendo lograr una constante retroalimentación entre los desarrolladores y clientes, donde se hace posible que los primeros puedan ampliar su visión del producto apoyados en la visión de los segundos. En este capítulo se detallan las 2 iteraciones llevadas durante la construcción del sistema, exponiéndoselas tareas generadas por las HU, así como las pruebas de aceptación realizadas al sistema.

3.2 Diseño del Sistema

La metodología XP propone entre sus prácticas la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto, la complejidad innecesaria y el código extra deben ser removido inmediatamente. El diseño adecuado para el software es aquel que supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos. Si alguna parte de la implementación resulta especialmente compleja, se replantea. De esta manera, cualquier cambio y modificación serán mucho más sencillos, en eso se basa la simplicidad del diseño.

Para el diseño de las aplicaciones, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se utilizan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración), se trata de simples tarjetas de papel para fichado que reemplazan a los diagramas en la representación de modelos.

A continuación se brindan los módulos desarrollados mediante sus respectivas tarjetas CRC.

Tarjeta CRC Clase Controller

Clase: Controller	
Responsabilidad	**Colaboración**
Cambiar los colores.	CssGen

Cambiar letra Cambiar estructura Cambiar imagen Exportar Guardar Nuevo Ver	FileProperties HtmlGen IVisitor

Tabla 13: Tarjeta CRC Clase Controller

Tarjeta CRC Clase FileProperties

Clase: FileProperties	
Responsabilidad	**Colaboración**
Cambiar valores en las propiedades del archivo (Crear y modificar nombre y ubicación)	----

Tabla 14: Tarjeta CRC Clase FileProperties

Tarjeta CRC Clase CssGen

Clase: CssGen	
Responsabilidad	**Colaboración**
Generar CSS	IVisitor

Tabla 15: Tarjeta CRC Clase CssGen

Tarjeta CRC Clase HtmlGen

Clase: HtmlGen	
Responsabilidad	**Colaboración**
Generar el HTML	IVisitor

Tabla 16: Tarjeta CRC Clase HtmlGen

Tarjeta CRC Clase IVisitor

Clase: IVisitor	
Responsabilidad	**Colaboración**
Obtener y modificar los valores de las etiquetas de la plantilla	Body Br Head Div Html Title A

Tabla 17: Tarjeta CRC Clase IVisitor

3.3 Fase de Implementación

En esta fase se realiza la implementación de las HU que fueron seleccionadas por cada iteración. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones. Como parte de este plan se crean tareas de programación para ayudar a organizar la implementación exitosa de las HU.

3.3.1 Tareas de programación por iteraciones

Cada HU como funcionalidad de la aplicación está compuesta por una o varias tareas de programación, éstas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una HU. A continuación se detallan para cada una de las iteraciones las tareas a desarrollar por cada HU.

Iteración 1

Tiempo real de implementación (iteración 1)

Historias de Usuario	Estimación	Real
Crear plantilla	2	1
Personalizar colores	1	1
Personalizar imágenes	1	2
Cambiar tipografía	1	1
Gestionar elementos	1	2
Exportar Plantilla	2	1

Tabla 18: Tiempo real de implementación (iteración 1)

Tareas de programación por Historias de Usuario

Historias de Usuario	Tareas de programación por historias de usuario
Crear plantilla	– Mostrar interfaz “Crear plantilla” – Cargar Plantilla
Personalizar colores	– Mostrar interfaz “Color” – Cambiar color
Personalizar imágenes	– Mostrar interfaz para seleccionar donde desea cambiar la imagen – Abrir cuadro de búsqueda para que el

	usuario seleccione la imagen que desea – Cambiar imagen
Cambiar tipografía	– Mostrar los distintos tipos de letras – Cambiar tipo de letra
Gestionar elementos	– Mostrar interfaz con las diferentes estructuras que podría tener la plantilla – Cambiar la estructura
Exportar Plantilla	– Abrir cuadro de búsqueda para que el usuario seleccione la ubicación

Tabla 19: Tareas de programación por Historias de Usuarios

Iteración 2

Tiempo real de implementación (iteración 2)

Historias de Usuario	Estimación	Real
Salvar Plantilla	1	1
Cargar Plantilla	1	1
Ver	2	3
Ayuda	1	1

Tabla 20: Tiempo real de implementación (iteración 2)

Tareas de programación por Historias de Usuario

Historias de Usuario	Tareas de programación por historias de usuario
Salvar Plantilla	– Abrir cuadro de búsqueda para que el usuario seleccione la ubicación

Cargar Plantilla	– Abrir cuadro de búsqueda para que el usuario seleccione la ubicación – Cargar plantilla
Ver	– Mostrar vista previa de la plantilla
Ayuda	– Mostrar interfaz de ayuda

Tabla 21: Tareas de programación por Historias de Usuarios

3.4 Fase de Pruebas

Una de las características principales de XP es su fuerte énfasis en las pruebas, poniendo la comprobación como el fundamento del desarrollo. Las pruebas se integran en el proceso de integración continua y construcción lo que rinde una plataforma altamente estable para el desarrollo futuro.

Las pruebas *(test)* constituyen unos de los pilares básicos de XP. No debe existir ninguna característica en el programa que no pueda ser probada, los programadores escriben pruebas para chequear el correcto funcionamiento del programa, los clientes realizan pruebas funcionales. El resultado es un programa más seguro que con el paso del tiempo sea capaz de aceptar nuevos cambios. (21)

Las pruebas se dividen en dos grupos: pruebas unitarias y pruebas de aceptación. Las pruebas unitarias son desarrolladas por los programadores y realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir, es como adelantarse a obtener los posibles errores. (22)

Por su parte las pruebas de aceptación están destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además comprobar que dicha funcionalidad sea la esperada por el cliente.

3.4.1 Pruebas de Aceptación

Cada HU lleva asociada criterios de aceptación y para cada criterio de aceptación se definen casos de prueba. Las pruebas de aceptación se escriben en las fases tempranas del desarrollo y antes de que el sistema se implemente para validar las necesidades de los usuarios y para dirigir la implementación.

Las pruebas de aceptación son creadas a partir de las HU. Durante una iteración la HU seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a probar cuando una HU ha sido correctamente implementada. Una HU puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento. Cada prueba de aceptación representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas. Ninguna HU se considera completa hasta que no supera sus pruebas de aceptación. (23)

Prueba 1 a la HU Crear plantilla

Caso de Prueba de Aceptación	
Código:HU1_P1	**Historia de Usuario:** 1
Nombre: Crear plantilla	
Descripción: Prueba para funcionalidad crear plantilla	
Condiciones de ejecución: ---	
Entrada/Pasos de Ejecución: El usuario seleccionará la opción ”Nuevo”	
Resultados Esperados: Se crea una plantilla nueva	
Evaluación de la prueba: Satisfactoria	

Tabla 22: Prueba 1 a la HU Crear plantilla

Prueba 1 a la HU Personalizar colores

Caso de Prueba de Aceptación	
Código:HU2_P1	**Historia de Usuario:** 2
Nombre: Personalizar colores	
Descripción: Prueba para funcionalidad personalizar colores	
Condiciones de ejecución: ---	

Entrada/Pasos de Ejecución: El usuario selecciona las opciones para cambiar color.
Resultados Esperados: Se muestran los colores existentes
Evaluación de la prueba: Satisfactoria

Tabla 23: Prueba 1 a la HU Personalizar colores

Prueba 2 a la HU Personalizar colores

Caso de Prueba de Aceptación	
Código:HU2_P2	**Historia de Usuario:** 2
Nombre: Personalizar colores	
Descripción: Prueba para funcionalidad personalizar colores	
Condiciones de ejecución: ---	
Entrada/Pasos de Ejecución: El usuario selecciona las opciones para cambiar color. Se muestran los colores existentes	
Resultados Esperados: Se cambia el color en la posición que el usuario haya deseado.	
Evaluación de la prueba: Satisfactoria	

Tabla 24: Prueba 2 a la HU Personalizar colores

Prueba 1 a la HU Personalizar imágenes

Caso de Prueba de Aceptación	
Código:HU3_P1	**Historia de Usuario:** 3
Nombre: Personalizar imágenes	
Descripción: Prueba para funcionalidad personalizar imágenes	
Condiciones de ejecución: ---	

Entrada/Pasos de Ejecución: El usuario selecciona las opciones para cambiar imágenes.
Resultados Esperados: Se muestra un cuadro de búsqueda para que el usuario seleccione la imagen que desea
Evaluación de la prueba: Satisfactoria

Tabla 25: Prueba 1 a la HU Personalizar imágenes

Prueba 2 a la HU Personalizar imágenes

Caso de Prueba de Aceptación	
Código:HU3_P2	**Historia de Usuario:** 3
Nombre: Personalizar imágenes	
Descripción: Prueba para funcionalidad personalizar imágenes	
Condiciones de ejecución: ---	
Entrada/Pasos de Ejecución: El usuario selecciona las opciones para cambiar imágenes. Se muestra un cuadro de búsqueda para que el usuario seleccione la imagen que desea	
Resultados Esperados: Se cambia la imagen en la posición que el usuario haya deseado	
Evaluación de la prueba: Satisfactoria	

Tabla 26: Prueba 2 a la HU Personalizar imágenes

Prueba 1 a la HU Cambiar tipografía

Caso de Prueba de Aceptación	
Código:HU4_P1	**Historia de Usuario:** 4
Nombre: Cambiar tipografía	
Descripción: Prueba para funcionalidad Cambiar tipografía	

Condiciones de ejecución: ---
Entrada/Pasos de Ejecución: El usuario selecciona las opciones para cambiar el tipo de letra
Resultados Esperados: Se cambia el tipo de letra
Evaluación de la prueba: Satisfactoria

Tabla 27: Prueba 1 a la HU Cambiar tipografía

Prueba 1 a la HU Gestionar elementos

Caso de Prueba de Aceptación	
Código:HU5_P1	**Historia de Usuario:** 5
Nombre: Gestionar elementos	
Descripción: Prueba para funcionalidad Gestionar elementos	
Condiciones de ejecución: ---	
Entrada/Pasos de Ejecución: El usuario selecciona la estructura que desea para la plantilla	
Resultados Esperados: Se modifica la estructura de la plantilla	
Evaluación de la prueba: Satisfactoria	

Tabla 28: Prueba 1 a la HU Gestionar elementos

Prueba 1 a la HU Exportar plantilla

Caso de Prueba de Aceptación	
Código:HU6_P1	**Historia de Usuario:** 6
Nombre: Exportar plantilla	
Descripción: Prueba para funcionalidad Exportar plantilla	

Condiciones de ejecución: ---
Entrada/Pasos de Ejecución: El usuario selecciona la opción para exportar plantilla
Resultados Esperados: Se muestra un cuadro para que el usuario le dé ubicación a la plantilla
Evaluación de la prueba: Satisfactoria

Tabla 29: Prueba 1 a la HU Exportar plantilla

Prueba 2 a la HU Exportar plantilla

Caso de Prueba de Aceptación	
Código:HU6_P2	**Historia de Usuario:** 6
Nombre: Exportar plantilla	
Descripción: Prueba para funcionalidad Exportar plantilla	
Condiciones de ejecución: ---	
Entrada/Pasos de Ejecución: El usuario selecciona la opción para exportar plantilla. Se muestra un cuadro para que el usuario le dé ubicación a la plantilla	
Resultados Esperados: Se exporta la plantilla con formato HTML	
Evaluación de la prueba: Satisfactoria	

Tabla 30: Prueba 2 a la HU Exportar plantilla

Prueba 1 a la HU Salvar plantilla

Caso de Prueba de Aceptación	
Código:HU7_P1	**Historia de Usuario:** 7
Nombre: Salvar plantilla	
Descripción: Prueba para funcionalidad Salvar plantilla	

Condiciones de ejecución: ---
Entrada/Pasos de Ejecución: El usuario selecciona la opción para exportar plantilla
Resultados Esperados: Se muestra un cuadro para que el usuario le dé ubicación a la plantilla
Evaluación de la prueba: Satisfactoria

Tabla 31: Prueba 1 a la HU Salvar plantilla

Prueba 2 a la HU Salvar plantilla

Caso de Prueba de Aceptación	
Código:HU7_P2	**Historia de Usuario:** 7
Nombre: Salvar plantilla	
Descripción: Prueba para funcionalidad Salvar plantilla	
Condiciones de ejecución: ---	
Entrada/Pasos de Ejecución: El usuario selecciona la opción para exportar plantilla. Se muestra un cuadro para que el usuario le dé ubicación a la plantilla	
Resultados Esperados: Se guarda la plantilla	
Evaluación de la prueba: Satisfactoria	

Tabla 32: Prueba 2 a la HU Salvar plantilla

Prueba 1 a la HU Cargar plantilla

Caso de Prueba de Aceptación	
Código:HU8_P1	**Historia de Usuario:** 8
Nombre: Cargar plantilla	
Descripción: Prueba para funcionalidad Cargar plantilla	

Condiciones de ejecución: ---
Entrada/Pasos de Ejecución: El usuario selecciona la opción para cargar plantilla
Resultados Esperados: Se muestra un cuadro de búsqueda para que el usuario seleccione la plantilla
Evaluación de la prueba: Satisfactoria

Tabla 33: Prueba 1 a la HU Cargar plantilla

Prueba 2 a la HU Cargar plantilla

Caso de Prueba de Aceptación	
Código:HU8_P2	**Historia de Usuario:** 8
Nombre: Cargar plantilla	
Descripción: Prueba para funcionalidad Cargar plantilla	
Condiciones de ejecución: ---	
Entrada/Pasos de Ejecución: El usuario selecciona la opción para cargar plantilla. Se muestra un cuadro de búsqueda para que el usuario seleccione la plantilla	
Resultados Esperados: Se carga la plantilla	
Evaluación de la prueba: Satisfactoria	

Tabla 34: Prueba 2 a la HU Cargar plantilla

Prueba 1 a la HU Ver

Caso de Prueba de Aceptación	
Código:HU9_P1	**Historia de Usuario:** 9
Nombre: Ver	
Descripción: Prueba para funcionalidad Ver	

Condiciones de ejecución: ---
Entrada/Pasos de Ejecución: El usuario selecciona la opción para pre-visualizar el contenido de la plantilla
Resultados Esperados: Se ejecuta un navegador y se muestra el contenido de la plantilla
Evaluación de la prueba: Satisfactoria

Tabla 35: Prueba 1 a la HU Ver

Prueba 1 a la HU Ayuda

Caso de Prueba de Aceptación	
Código:HU10_P1	**Historia de Usuario:** 10
Nombre: Ayuda	
Descripción: Prueba para funcionalidad Ayuda	
Condiciones de ejecución: ---	
Entrada/Pasos de Ejecución: El usuario selecciona la opción de ayuda	
Resultados Esperados: Se muestra la ayuda	
Evaluación de la prueba: Satisfactoria	

Tabla 36: Prueba 1 a la HU Ayuda

3.5 Conclusiones

En el presente capítulo se trazaron las tareas de programación correspondientes para dar solución a las historias de usuario planteadas, así como el modelo de datos y las pruebas de aceptación que se llevaron a cabo con el objetivo de brindarle al cliente un producto funcional que cumpla con las necesidades especificadas por el mismo. Con la conclusión de este capítulo se considera terminada la propuesta que ofrece el trabajo.

Conclusiones generales

El presente trabajo se enfocó en el estudio y desarrollo de una aplicación para la generación de plantillas HTML. Como resultado del trabajo realizado se logró el diseño e implementación de un sistema de software con las siguientes ventajas:

- Ahorro en tiempo y recursos para las personas que utilicen la aplicación.
- Fácil mantenimiento, de modo que se le puedan adicionar nuevas funcionalidades.
- Facilidad de uso.

El uso de la metodología XP para guiar el proceso y los artefactos generados en cada una de sus fases propició una mayor comunicación entre el equipo de desarrollo y el cliente, logrando obtener un mejor funcionamiento del sistema y que éste responda a todas las necesidades del cliente. Por lo que se considera concluida la investigación y dado por cumplido todos los objetivos establecidos al comienzo de la investigación.

Recomendaciones

Debido a los resultados de la investigación efectuada y de la experiencia adquirida durante la realización de este trabajo, y con el propósito de asegurar la posterior ampliación, modificación y mejora del sistema propuesto, se exponen a continuación algunas recomendaciones:

- Utilizar el presente trabajo como bibliografía para futuras investigaciones que utilicen la misma línea de implementación expuesta.
- Aplicar el software en los departamentos docentes de la Facultad 4 para obtener un mayor rendimiento del mismo.
- Extender las funcionalidades acordes a nuevos requisitos que surjan en los departamentos docentes.
- Desarrollar una ayuda para mejorar el trabajo con la herramienta.

Trabajos citados

1. **B., Alejandro Alvarez.** *Generadores de Interfaces de Usuario: QT Designer, NetBeans y Windows Forms Designer.*

2. **Softfree.** Softfree. *Softfree.* [En línea] 2007. [Citado el: 15 de 11 de 2010.] http://www.softfree.eu/es/windows/imagen/artisteer.php.

3. **ABCdatos.** ABCdatos. *ABCdatos.* [En línea] 1999. [Citado el: 12 de 12 de 2010.] http://www.abcdatos.com/webmasters/programa/z4422.html.

4. **LaWebera.** LaWebera.es. *LaWebera.es.* [En línea] [Citado el: 14 de 12 de 2010.] http://www.lawebera.es/manual-diseno-web/.

5. **Trujillo, Abner.** Maestros del Web. *Maestros del Web.* [En línea] [Citado el: 16 de 12 de 2010.] http://www.maestrosdelweb.com/editorial/nvu/.

6. **Alvarez, Miguel Angel.** Desarrolloweb.com. *Desarrolloweb.com.* [En línea] [Citado el: 16 de 12 de 2010.] http://www.desarrolloweb.com/articulos/332.php.

7. Solcre Technology Solutions. *Solcre Technology Solutions.* [En línea] [Citado el: 6 de 2 de 2011.] Solcre Technology Solutions.

8. **Urquiza, Pedro Castillo.** *ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DEL PORTAL WEB PARA LA ESCUELA CUBANA DE BOXEO.* 2010.

9. **GNU.** Desarrollo de aplicaciones de escritorio. *Desarrollo de aplicaciones de escritorio.* [En línea] 2010. [Citado el: 2 de 6 de 2011.] http://www.gnuconsultores.com/es/ingenieria/desarrollo/escritorio.

10. **Kernel Error.** Arquitectura 3 Capas. *Arquitectura 3 Capas.* [En línea] [Citado el: 15 de 1 de 2011.] http://kernelerror.net/programacion/php/arquitectura-3-capas/.

11. La arquitectura cliente servidor . *La arquitectura cliente servidor .* [En línea] 4 de 11 de 2009. [Citado el: 15 de 1 de 2011.]

12. **Leyton, E.** Ingeniería de software con UML. [En línea] [Citado el: 14 de 1 de 2011.] http://www.eduardoleyton.com/apuntes/Uml.pdf.

13. **Free Download Manager.** Free Download Manager. *Free Download Manager.* [En línea] [Citado el: 8 de 1 de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

14. **GSINNOVA, G.D.S.** Rational Rose Data Modeler. [En línea] [Citado el: 17 de 1 de 2011.] http://www.rational.com.ar/herramientas/rosedatamodeler.html.

15. **Lobos, M.E.D.** Aprender a programar. *Aprender a programar.* [En línea] [Citado el: 16 de 1 de 2011.] http://www.mailxmail.com/curso-aprende-programar/concepto-lenguaje-programacion.

16. **Martínez, Yainicet Nogueras.** Análisis y Diseño del Sistema Integral de Seguridad Social del MININT. *Análisis y Diseño del Sistema Integral de Seguridad Social del MININT.* 2009.

17. **Izquierdo., Anniel Rodríguez.** *Sistema para la recuperación de información aplicando técnicas de trabajo colaborativo.* 2009.

18. **Duvergel, Yoannia Castillo.** *Implementación de algoritmos de recomendaciones de imágenes para textos noticiosos.* 2009.

19. **Berroa, Geldri.** *Análisis, diseño e implementación de la Capa de Presentación del sub-módulo Dotación de Equipos Policiales perteneciente al módulo Registro y Control de SIIPOL.* Ciudad Habana : s.n., 2009.

20. **Neira, Yaneisy Pérez.** *Sistema para el control de la información y evaluación de los profesores en los departamentos docentes de la facultad 8.* C. Habana : s.n., 2009.

21. **Escribano, G.F.** Introducción a Extreme Programming. *Introducción a Extreme Programming.* [En línea] [Citado el: 2 de 5 de 2011.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf..

22. **Sanchez, M.A.M.** Metodologías De Desarrollo De Software. *Metodologías De Desarrollo De Software.* [En línea] [Citado el: 2 de 5 de 2011.] http://www.willydev.net/Descargas/cualmetodologia.pdf.

23. **Riva, C.D.** Actas de los Talleres de Ingeniería del Software y Base de Datos. *Actas de los Talleres de Ingeniería del Software y Base de Datos.* [En línea] [Citado el: 3 de 5 de 2011.] http://www.sistedes.es/TJISBD/Vol-3/No-4/PRIS09.pdf.

Bibliografía

1. **B., Alejandro Alvarez.** *Generadores de Interfaces de Usuario: QT Designer, NetBeans y Windows Forms Designer.*

2. **Softfree.** Softfree. *Softfree.* [En línea] 2007. [Citado el: 15 de 11 de 2010.] http://www.softfree.eu/es/windows/imagen/artisteer.php.

3. **ABCdatos.** ABCdatos. *ABCdatos.* [En línea] 1999. [Citado el: 12 de 12 de 2010.] http://www.abcdatos.com/webmasters/programa/z4422.html.

4. **LaWebera.** LaWebera.es. *LaWebera.es.* [En línea] [Citado el: 14 de 12 de 2010.] http://www.lawebera.es/manual-diseno-web/.

5. **Trujillo, Abner.** Maestros del Web. *Maestros del Web.* [En línea] [Citado el: 16 de 12 de 2010.] http://www.maestrosdelweb.com/editorial/nvu/.

6. **Alvarez, Miguel Angel.** Desarrolloweb.com. *Desarrolloweb.com.* [En línea] [Citado el: 16 de 12 de 2010.] http://www.desarrolloweb.com/articulos/332.php.

7. Solcre Technology Solutions. *Solcre Technology Solutions.* [En línea] [Citado el: 6 de 2 de 2011.] Solcre Technology Solutions.

8. **Urquiza, Pedro Castillo.** *ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DEL PORTAL WEB PARA LA ESCUELA CUBANA DE BOXEO.* 2010.

9. **GNU.** Desarrollo de aplicaciones de escritorio. *Desarrollo de aplicaciones de escritorio.* [En línea] 2010. [Citado el: 2 de 6 de 2011.] http://www.gnuconsultores.com/es/ingenieria/desarrollo/escritorio.

10. **Kernel Error.** Arquitectura 3 Capas. *Arquitectura 3 Capas.* [En línea] [Citado el: 15 de 1 de 2011.] http://kernelerror.net/programacion/php/arquitectura-3-capas/.

11. La arquitectura cliente servidor . *La arquitectura cliente servidor .* [En línea] 4 de 11 de 2009. [Citado el: 15 de 1 de 2011.]

12. **Leyton, E.** Ingeniería de software con UML. [En línea] [Citado el: 14 de 1 de 2011.] http://www.eduardoleyton.com/apuntes/Uml.pdf.

13. **Free Download Manager.** Free Download Manager. *Free Download Manager.* [En línea] [Citado el: 8 de 1 de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

14. **GSINNOVA, G.D.S.** Rational Rose Data Modeler. [En línea] [Citado el: 17 de 1 de 2011.] http://www.rational.com.ar/herramientas/rosedatamodeler.html.

15. **Lobos, M.E.D.** Aprender a programar. *Aprender a programar.* [En línea] [Citado el: 16 de 1 de 2011.] http://www.mailxmail.com/curso-aprende-programar/concepto-lenguaje-programacion.

16. **Martínez, Yainicet Nogueras.** Análisis y Diseño del Sistema Integral de Seguridad Social del MININT. *Análisis y Diseño del Sistema Integral de Seguridad Social del MININT.* 2009.

17. **Izquierdo., Anniel Rodríguez.** *Sistema para la recuperación de información aplicando técnicas de trabajo colaborativo.* 2009.

18. **Duvergel, Yoannia Castillo.** *Implementación de algoritmos de recomendaciones de imágenes para textos noticiosos.* 2009.

19. **Berroa, Geldri.** *Análisis, diseño e implementación de la Capa de Presentación del sub-módulo Dotación de Equipos Policiales perteneciente al módulo Registro y Control de SIIPOL.* Ciudad Habana : s.n., 2009.

20. **Neira, Yaneisy Pérez.** *Sistema para el control de la información y evaluación de los profesores en los departamentos docentes de la facultad 8.* C. Habana : s.n., 2009.

21. **Escribano, G.F.** Introducción a Extreme Programming. *Introducción a Extreme Programming.* [En línea] [Citado el: 2 de 5 de 2011.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf..

22. **Sanchez, M.A.M.** Metodologías De Desarrollo De Software. *Metodologías De Desarrollo De Software.* [En línea] [Citado el: 2 de 5 de 2011.] http://www.willydev.net/Descargas/cualmetodologia.pdf.

23. **Riva, C.D.** Actas de los Talleres de Ingeniería del Software y Base de Datos. *Actas de los Talleres de Ingeniería del Software y Base de Datos.* [En línea] [Citado el: 3 de 5 de 2011.] http://www.sistedes.es/TJISBD/Vol-3/No-4/PRIS09.pdf.

24. **Vergara, Kervin.** Blog Informático. *Blog Informático.* [En línea] [Citado el: 15 de 12 de 2010.] http://www.bloginformatico.com/amaya-editor-html-con-navegador-web-incluido.php.

25. **B., Alejandro Alvarez.** Generadores de Interfaces de Usuario: QT Designer, NetBeans y Windows Forms Designer. *Generadores de Interfaces de Usuario: QT Designer, NetBeans y Windows Forms Designer.* San José, Costa Rica : s.n.

26. **Cornejo, José Enríquez González.** DocIRS. *DocIRS.* [En línea] [Citado el: 17 de 12 de 2010.] http://www.docirs.cl/caracteristica_herramienta_uml.htm.

27. **Osmosis Latina.** Osmosis Latina. *Osmosis Latina.* [En línea] 31 de 12 de 2007. [Citado el: 7 de 1 de 2011.] http://www.osmosislatina.com/lenguajes/uml/basico.htm.

28. **Fanjul, Alejandro.** *Arquitectura de 3 capas.* 2006.

Glosario de términos

GUI: *Graphical User Interface* (Interfaz gráfica de usuarios).

Software: Equipamiento lógico o soporte lógico de un ordenador. Comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Suele ser utilizado para referirse a los programas y aplicaciones informáticas.

Tecnologías de la Información y las Comunicaciones: conjunto de servicios, redes, software y dispositivos que tienen como fin la mejora de la calidad de vida de las personas dentro de un entorno, y que se integran a un sistema de información interconectado y complementario.

Metodología de Desarrollo: Marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Código abierto: (en inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente.

HTML: es el lenguaje de marcado predominante para la construcción de páginas web.

Generación de código: es una de las fases mediante el cual un compilador convierte un programa sintácticamente correcto en una serie de instrucciones a ser interpretadas por una máquina.

Lenguaje de Programación: Conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Interfaz de Programación de Aplicaciones (API): Conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

CSS: (Significa hojas de estilo en cascada) es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML.

MVC (Modelo-Vista-Controlador): es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Glosario de términos

IDE: ***I****ntegrated* ***D****evelopment* ***E****nvironment*, Entorno de Desarrollo Integrado. Entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.