Universidad de las Ciencias Informáticas

Sistema de Reservación de los servicios de Barbería, Lavandería y Peluquería en la Universidad de las Ciencias Informáticas (UCI).

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

Autores: Reinier Jean Felipe
Oscar Rodríguez Ramos
Tutor: Ing. Arturo Luis Lara Fernández

Junio, 2010

"Año del 52 Aniversario de la Revolución"

Pensamiento

"... Milito en el grupo de los impacientes, y milito en el bando de los apurados, y de los que siempre presionan para que las cosas se hagan, y de los que muchas veces tratan de hacer más de lo que se puede..."

Fidel

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estimen pertinente con este trabajo.
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____
Firma del Autor
Reinier Jean Felipe

_____
Firma del Autor
Oscar Rodríguez Ramos

_____
Firma del Tutor
Ing. Arturo Luis Lara Fernández

Agradecimientos

Quiero agradecer:

A mis tíos que se han encargado de mí por estos 5 años de estudios.

A mi mamá que siempre ha confiado en mí y nunca ha dudado de mí.

A mis amigos que se han convertido en los pilares que me has sostenido cada vez que las fuerzas me faltaban.

A mis compañeros del edificio y a todas las personas con las que he conocido en estos 5 años aquí en la universidad.

Y en especial a toda mi familia.

Reinier

Quiero agradecer:

A mi familia en general, especialmente a mis padres por guiarme y apoyarme en todo momento.

A mi abuela por sacrificarse tanto conmigo a lo largo de estos 24 años que tengo.

A mi hermano por su ayuda constante y por haber estado siempre pendiente de mí.

A mi novia por todo el amor y el cariño que nos tenemos y por todo el apoyo que me ha brindado.

A todos mis amigos, a los del barrio y a los de la escuela. En general a todos los que de una manera u otra contribuyeron en este trabajo, muchas gracias.

Oscar

Dedicatoria

A la Revolución por brindarnos la gran posibilidad de ampliar nuestros conocimientos en está gran universidad que es la Universidad de las Ciencias Informáticas (UCI). Al comandante en jefe en especial por ser el visionario de este gran proyecto social, en el que de una forma u otra nos hemos sumergido con la disposición y la esperanza de ser hombres de futuro y de ciencias.

Los Autores

A mi familia y a la memoria de mi bisabuela Leyda (Yeyita).

Oscar

A mi abuela que, no pudo verme convertido en ingeniero.

Reinier

Resumen

Con el desarrollo de la informática y el aumento de su impacto social, son cada vez más las instituciones y organizaciones que optan por incorporar aplicaciones que gestionen su información, logrando así una mayor dinámica en el proceso de negocio de los servicios de lavandería, peluquería y barbería en la UCI.

La Universidad de las Ciencias Informáticas es una institución completamente comprometida con la Revolución Cubana, que se propone, con soluciones creativas y autóctonas brindar disímiles servicios a favor de construir una ciudad completamente informatizada. Mediante la formación integral y continua de profesionales, la actividad científico-técnica y la extensión universitaria, contribuye de forma significativa al desarrollo sostenible de la sociedad cubana y mantiene un liderazgo nacional en el campo de la tecnología informática.

El presente proyecto consiste en el desarrollo de una herramienta capaz de agilizar los procesos de reservación de los servicios de lavandería, peluquería y barbería en la Universidad de las Ciencias Informáticas (UCI). Con la aplicación se espera brindar una solución automatizada, flexible y única a todo el proceso de reservación de lavandería, peluquería y barbería para así proporcionar una mayor funcionalidad al sistema.

En la universidad no se ha desarrollado ningún sistema que gestione automáticamente los servicios anteriormente planteados, es por ello que el mismo debe estar diseñado para hacer un uso óptimo de los recursos existentes en la UCI.

Tabla de Contenido

Índice de Tablas.

Introducción

El uso de la Internet se ha generalizado en todo el mundo en los últimos años gracias al desarrollo de las infraestructuras de comunicaciones y a la aparición de diversos factores tales como, el continuo aumento de los equipos informáticos. La demanda de servicios interactivos, la liberalización de las telecomunicaciones y la introducción cada vez más general de las computadoras en el hogar.

Ofertar servicios informáticos útiles, novedosos y atractivos para la familia cubana, constituye uno de los principales objetivos del Ministerio de la Informática y las Comunicaciones en conjunto con ETECSA.

La UCI no es ajena a estos servicios, de esta forma, ofrece una variedad de accesos que hacen la vida de sus alumnos y profesores más placentera, por ejemplo la guía telefónica (http://guiatelefonica.uci.cu) y el sitio de transportación (http://transportacion.uci.cu), por lo que se plantea el siguiente problema científico.

Problema Científico:

¿Cómo automatizar los procesos de reservación de los servicios de barbería, peluquería y lavandería en la UCI?

Objeto de estudio:

Los procesos de reservación de servicios en la UCI.

Campo de Acción:

Los procesos de reservación de los servicios de barbería, peluquería y lavandería en la UCI.

Objetivo General:

Desarrollar una aplicación informática que permita la reservación de los servicios de barbería, peluquería y lavandería en la UCI.

Objetivos Específicos:

- Analizar el estado actual del proceso de reservación de la barbería, peluquería y lavandería en la UCI.
- Definir las funcionalidades que debe tener el sistema propuesto.
- Definir la metodología, las herramientas y lenguajes que se utilizará en la realización del sistema.
- Realizar el diseño e implementación del sistema.

Tareas a cumplir por el estudiante:

- Estudio detallado de como se desarrolla el proceso de reservación de la barbería, peluquería y lavandería en la UCI.
- Estudio de la metodología, las herramientas y el lenguaje propuesto para la realización del sistema.
- Planteamiento de las funcionalidades o requisitos que debe tener el sistema.
- Análisis y diseño del sistema propuesto.
- Implementación y prueba del sistema propuesto.

Métodos científicos de investigación:

Para desarrollar está investigación y lograr los objetivos planteados se utilizaron, teniendo en cuenta el tema de estudio, métodos teóricos y empíricos, mediante los cuales se pudo tener una idea más detallada de lo que se quería lograr. En la investigación se utilizan los siguientes:

Teóricos:

Histórico - Lógico: Se estudió la forma en que se realiza el proceso de reservación de servicios de barbería, peluquería y lavandería en la universidad desde sus inicios, obteniendo así los aspectos positivos y negativos del mismo, y se dedujo la necesidad de un sistema que permitiera automatizar la realización de la reservación de los servicios mencionados.

Análisis – Síntesis: Se analiza la bibliografía encontrada referente al tema en cuestión y se sintetizan los aspectos más importantes para la investigación.

Empíricos:

Observación: Se realizó una exhaustiva observación para ver que sucedía en realidad en el proceso de reservación de los servicios de barbería, peluquería y lavandería en la UCI para sacar todos los problemas que presenta el mismo.

Entrevista: Se realizó la entrevista a un gran número de estudiantes, profesores y trabajadores que hacen uso frecuente de estos servicios para ver si en realidad se necesitaba el sistema y que requisitos debía tener el mismo para que cumpliera de forma eficiente el objetivo planteado.

Idea a defender:

El desarrollo de una aplicación informática que permita la reservación de los servicios de barbería, peluquería y lavandería en la UCI.

Posibles resultados:

Aplicación informática para la reservación de los servicios de barbería, peluquería y lavandería en la Universidad de las Ciencias Informáticas. Para garantizar el uso eficiente de los mismos y aumentar la calidad de dicho proceso.

Estructura de la Aplicación:

Capítulo 1: Fundamentación teórica. En este capítulo se realizará un estudio del estado del arte revisando algunas aplicaciones con el mismo propósito de reservaciones, exponiendo las ventajas y desventajas de estos. Se realizará también un análisis de las herramientas, metodologías y lenguajes que existen en la actualidad y que sean factibles para el desarrollo de la aplicación, para de esta forma justificar la selección de los mismos.

Capítulo 2: Características de Sistema, Planificación y Diseño. Se analizarán los procesos de emisión de documentos de identificación, los cuales contribuirán a la realización de la aplicación. Se exhibirá

una descripción general del sistema propuesto. Además, se identificarán los requisitos funcionales y no funcionales, se describirán las historias de usuarios y las tarjetas CRC.

Capítulo 3: Implementación y Prueba. Se realizará la implementación de la aplicación, obteniéndose los artefactos propios como son las tareas de ingeniería de cada iteración y el diagrama de despliegue. Además, subsiguientemente se realizarán las pruebas pertinentes (aceptación y unitarias).

Capítulo 1: Fundamentación Teórica

El proceso de selección de las condiciones de ejecución, ambiente de trabajo y la forma en que se va a elaborar un *software* es fundamental para el progreso adecuado de la confección del mismo puesto que puede influir tanto en el tiempo como en la calidad del trabajo. En este capítulo se realiza un análisis de los sistemas que brindan servicios de reservaciones existentes en el mundo y sus principales procesos, de los cuales se tomaron las características que podrían ser compatibles a la universidad ya que hacía falta una visión más profesional del tipo de producto que se pensaba desarrollar. También se explican las herramientas, tecnologías y metodologías que se seleccionaron dentro de aquellas que más se adaptaban a las condiciones requeridas por la universidad, mencionando sus principales características, ventajas y desventajas.

Sistemas Automatizados de Reservaciones.

Actualmente en el mundo se ha avanzado mucho con respecto a los Sistemas automatizados de reservaciones con el fin de obtener las ventajas competitivas que el uso de estas herramientas permiten generar, convirtiéndose Internet en el escenario y el instrumento dominante que se utiliza para la promoción y la distribución de estos. Las ventajas competitivas que ofrece para las empresas un sistema automatizado de reservas en línea se traducen, entre otras, en un aumento de la calidad de los servicios prestados a sus clientes. Donde algunos de los principales servicios que se ofrecen a través de estos sistemas consisten en poner a disposición del usuario: información sobre todos los servicios disponibles; reservas; listas de espera de hoteles; tarifas confidenciales y tarifas normales; confeccionar agendas, cambios y anulaciones, entre otros servicios.

Beneficios de los sistemas automatizados de reservas estudiados:

- Mayor eficiencia y rapidez en la gestión de sus reservaciones.
- Control en tiempo real de todas vuestras disponibilidades, precios y ofertas especiales.
- Información completamente actualizada sobre los servicios prestados.
- Cálculo del precio exacto de todos los productos.
- Fácil de usar.
- Disponible las 24 horas los 365 días del año.

- Se puede utilizar desde cualquier lugar a través de Internet o Intranet. [12]

Sistemas Automatizados de reservaciones que existen en el mundo, Cuba y en la UCI.

En la actualidad y como una de las consecuencias del desarrollo alcanzado por la humanidad, las personas tienen menos tiempo dado la dinámica de la sociedad actual. Es por esto, y gracias al desarrollo de las TIC, que en el mundo cada vez toma más auge el uso de la red para gestionar ciertas actividades que tomarían demasiado tiempo llevarlas a cabo personalmente. Las reservaciones en línea han ganado mucha aceptación ya que se les permite a los usuarios, desde cualquier lugar donde se encuentre acceder a los diferentes servicios que se ofertan.

Sistema de reservación de eDreams. (http://www.edreams.es)

Software dedicado a la reservación de servicios, entre ellos, reserva de vuelos. Permite que el usuario reserve viajes de ida y vuelta, del origen y destino que escoja, así como las fechas y horarios en las que quiere viajar. Posibilita al cliente la ventaja de escoger viajar en vuelos directos o sea sin hacer escalas hasta el destino final y con las compañías aéreas de su preferencia. Facilita información sobre el viaje para que el cliente sepa las posibilidades de reservación que tiene de forma general. Ofrece al pasajero varias formas de pago, entre las que se encuentra: tarjeta, por transferencia y en efectivo. El cliente puede elegir la forma de pago más conveniente. Además de que puede realizar la compra en línea o vía telefónica.

Para confirmar la solicitud de viaje del cliente, se le envía un correo electrónico a su dirección de correo especificado por este y no es válida la reservación hasta que no se confirme la solicitud mediante esta misma vía.

Sistema de reservación de cubana de aviación. (http://www.cubana.cu)

Software dedicado a la reservación de vuelos específicamente. Esta aplicación de Cubana de Aviación S.A. posibilita al reservar el billete de viaje seleccionar la ciudad origen y destino, la fecha y clase (económica, club tropical (1ra)) en la cual desea viajar el cliente, además puede solicitar el asiento que desee ocupar en la aeronave (asiento de fumador, no fumador, ventanilla, pasillo, número y fila).

Ofrece a sus clientes información sobre la aerolínea además de facilitar las ventas directas de pasaje. Las ventas directas comprenden la reserva y el pago. El pago deberá efectuarse usando una tarjeta de crédito Visa o *MasterCard*; enviando notificación al pasajero vía correo electrónico si la transacción fue exitosa.

Sistema de reservación de transporte, UCI. (https://transportacion.uci.cu)

Este sistema informático permite la reservación de los pases de fin de semana de los estudiantes en la UCI hacia los diferentes municipios de Ciudad de La Habana.

Para reservar el pase, el estudiante solicita hasta una fecha determinada, salir el fin de semana. De esta manera, el sistema permite al estudiante reservar ida y regreso en el transporte de la universidad, atendiendo al municipio y ruta en la que quiera.

Si el estudiante no es de La Habana debe introducir el nombre del familiar, el parentesco, la dirección y el teléfono de la persona que va a visitar. Haciéndose luego la asignación del transporte (ómnibus).

Tendencias y tecnologías actuales.

Teniendo en cuenta las necesidades vistas y las características del entorno donde se aplicará la solución propuesta, se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear, descritas a continuación.

Aplicaciones Web.

Las aplicaciones web son una especialización y concreción de las aplicaciones cliente/servidor, o sea, su arquitectura general es la de un sistema cliente/servidor, donde tanto el cliente (navegador) como el servidor (servidor web), y el protocolo mediante el cual se comunican (el HTTP: *Hyper Text Transfer Protocol*) son un estándar y no deben ser creados por el desarrollador de aplicaciones.

La parte del cliente de las aplicaciones web está formada por el código HTML (*Hyper Text Markup Language*) que forma la página web, con opción a código ejecutable mediante los lenguajes *script* de los navegadores (*JavaScript, VBScript, PerlScript*) o mediante pequeños programas (*applets*) en Java. La parte del servidor está formada por un *script* que es ejecutado por el servidor Web, y cuya salida se envía al navegador del cliente.

La creciente popularidad de las aplicaciones web se debe a sus múltiples ventajas, entre las cuales se destacan las siguientes:

Multiplataforma: con un solo programa, un único ejecutable, nuestras aplicaciones pueden ser utilizadas a través de múltiples plataformas, tanto de *hardware* como de *software*.

Actualización instantánea: debido que todos los usuarios de la aplicación hacen uso de un solo programa que radica en el servidor, los usuarios siempre utilizarán la versión más actualizada del sistema.

Suave curva de aprendizaje: los usuarios, como emplean la aplicación a través de un navegador, hacen empleo del sistema tal como si estuvieran navegando por Internet, debido a esto el acceso es mucho más intuitivo.

Fácil de integrar con otros sistemas: debido a que se basa en protocolos estándares, la información manejada por el sistema puede ser accedida con mayor facilidad por otros sistemas.

Acceso móvil: el usuario puede acceder a la aplicación con la única restricción de que cuente con un acceso a la red privada de la organización o a Internet, dependiendo de las políticas establecidas por dicha organización; puede hacerlo desde una computadora de escritorio, una *LAPTOP*, desde una agenda electrónica, desde un celular, desde la oficina, desde la casa o cualquier otra parte del mundo.

El desarrollo de aplicaciones web está siendo empleado en muchas organizaciones, está situación continuará en ascenso indefinidamente. Es por ello que día a día se requieren más programadores capacitados para desarrollos basados en la *World Wide Web* (WWW).

No obstante la serie de ventajas que presenta tiene algunas desventajas, las cuales se refieren a continuación:

Acceso limitado: la necesidad de conexión a Internet hace que el acceso a estás aplicaciones no esté al alcance de todos.

La interactividad no se produce en tiempo real: en las aplicaciones web cada acción del usuario conlleva a un tiempo de espera que en algunas ocasiones es extremadamente grande hasta la respuesta del sistema.

Diferencias de presentación entre plataformas y navegadores, la falta de estándares ampliamente soportados dificulta el desarrollo de las aplicaciones.

Modelo Cliente/Servidor.

Es conocido que la arquitectura cliente/servidor es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que, definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información. Se establece así un enlace de comunicación transparente entre los elementos que conforman la estructura. [13] [20]

Entre las principales características de la arquitectura Cliente/Servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, solo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Ventajas de la arquitectura Cliente/Servidor.

- El servidor no necesita potencia de procesamiento, parte del proceso se reparte entre los clientes.
- Se reduce el tráfico de la red sustancialmente. El cliente se conecta al servidor cuando es estrictamente necesario, obtiene los datos que necesita y cierra la conexión dejando libre la red.

PHP (Hyper Text Preprocessor).

El PHP (acrónimo de "PHP: *Hyper Text Preprocessor*"), es un lenguaje interpretado de alto nivel embebido en páginas HTML (acrónimo de “HTML: *Hyper Text Markup Language*”) y ejecutado en el servidor.

El PHP inició como una modificación a PERL escrita por el programador Rasmus Lerdorf a finales de 1994. Su primer uso fue el de mantener un control sobre quien visitaba su currículum vital en su Web. En los siguientes tres años, se fue convirtiendo en lo que se conoce como PHP/FI 2.0. Esta forma de programar llegó a muchos usuarios, pero el lenguaje no tomó el peso actual hasta que dos programadores de Israel Zeev Surasky y Andi Gutmans le incluyeron nuevas características en 1997, que dio por resultado el PHP3.0. En el invierno de 1998, poco después del lanzamiento oficial de PHP 3.0, Andi Gutmans y Zeev Surasky comenzaron a trabajar en la reescritura del núcleo de PHP. Los objetivos del diseño fueron mejorar la ejecución de aplicaciones complejas, y mejorar la modularidad del código base de PHP. Estas aplicaciones se hicieron posibles por las nuevas características de PHP 3.0 y el apoyo de una gran variedad de bases de datos y APIs de terceros, pero PHP 3.0 no fue diseñado para el mantenimiento tan complejo de aplicaciones eficientemente. [13]

Una de las características más poderosas del PHP es el soporte nativo incluido para una gran cantidad de bases de datos. Entre las bases de datos soportadas se encuentran: *MySQL, Oracle, PostgreSQL, MSSQL Server, DBM, DBase, Frontbase, FilePro, SQLite* entre otras. PHP ofrece además una potente integración con muchas bibliotecas externas, las cuales permiten que el desarrollador realice prácticamente cualquier cosa desde generar documentos en formato PDF (*Portable Document Format*) hasta analizar código XML (*Extensive Markup Language*), permite además mediante su integración con la librería GTK+ crear programas GUI (*Graphics User Interface*, al igual que los programas de *Visual Basic* o *Delphi*). Dichos programas creados con PHP-GTK son altamente portables entre plataformas.

La principal ventaja de PHP viene dada por ser un lenguaje multiplataforma, funciona tanto para Unix (con el popular Apache) como para Windows (con la versión de Apache para el mismo o el servidor WWW incorporando *Internet Information Server*) permitiendo que el código que haya sido creado para cualquiera de las plataformas sea completamente funcional en la otra y no haya porque realizarse modificación alguna.

Es *software* completamente libre, lo cual implica menos costos y servidores WWW mucho más baratos que alternativas propietarias. Es sumamente rápido y su integración con Sistemas Gestores de Bases de Datos libres (*MySQL, PostgreSQL*) y servidores igualmente libres como Apache lo hacen erguirse como una de las alternativas más económicas y eficientes del actual mercado.

PHP posee una de las más grandes comunidades en Internet, de esta manera, no se hace nada difícil encontrar a lo largo de toda la red cualquier tipo de ayuda, documentación, noticias, artículos, código fuente y muchos más recursos.

Debido a su extensa distribución PHP está siendo sostenido por una inmensa comunidad de programadores. Siendo un *software* de fuente totalmente abierta, PHP se fortalece cada vez más de la ayuda de un gran grupo de desarrolladores, logrando así que los problemas de funcionamiento sean encontrados y reparados en el menor tiempo posible. El código fuente se mantiene al día de forma continua con mejoras y extensiones del mismo para lograr una ampliación de las capacidades del lenguaje.

La funcionalidad de PHP se describe a través de los siguientes pasos:

- Escritura en las páginas HTML pero con el código de PHP embebido.
- Guardar la página creada en el servidor WWW.
- Un usuario mediante un navegador solicita una página al servidor WWW.
- El servidor interpreta en código PHP embebido en la página solicitada.

El servidor envía de vuelta al usuario que realizó la petición el resultado del conjunto de código HTML y el resultado de la interpretación del código PHP incrustado que es también HTML.
Bajo ningún concepto el servidor envía al navegador del usuario que realiza la petición código PHP, todas las operaciones son realizadas de forma transparente al usuario, el código PHP es ejecutado en el servidor por el motor PHP y solo el resultado de dicha ejecución es enviado en forma de HTML al navegador del usuario. El resultado es lo que se conoce como una página puramente HTML, de esta forma, el usuario aparentemente estará visitando una página HTML que dichas páginas son interpretadas por cualquier navegador.

Por ser PHP un lenguaje ejecutado en el servidor no se hace necesario que el navegador lo soporte, es totalmente independiente del navegador empleado por cualquier usuario, no siendo así para el funcionamiento de las páginas alojadas en el servidor, este si debe soportar PHP para un correcto funcionamiento de dichas páginas. Además de que se puede encontrar de forma libre en el mercado y podemos acceder a él mediante Internet. [12]

Favorablemente, lo actual incluido a PHP 5 mejora muchas e importantes áreas en el lenguaje y también su ejecución, por ejemplo:

- Soporta una compleja Programación Orientada a objetos (POO).
- XML.
- Integración nativa con *Zend Engine*.

PHP 5 introduce tres palabras clave (*prívate, protected y public*) las cuales surgen para sustituir a la palabra clave Var en la definición de variable miembro —atributos— de las clases, y que anteceden a la definición de funciones miembro —métodos—.

Otros lenguajes como *PERL* (*Practical Extraction and Report Language*), *ASP* (*Active Server Pages*) y JSP (*Java Server Pages*) tienen características muy similares a PHP aunque además poseen características especiales de cada uno de ellos mediante las cuales son distinguidos, entre ellas se pueden destacar:

Características multiplataforma: con excepción de ASP (quien es solamente soportado por la plataforma Windows de Microsoft) los demás lenguajes están actualmente soportados en múltiples y diversas plataformas.

Velocidad de ejecución: la mayor velocidad de ejecución es alcanzada por PHP, seguido por PERL, quien a su vez es seguido por JSP.

Disponibilidad de recursos: en la actualidad los lenguajes más utilizados en toda la Internet son el PHP y el JSP, siendo los más empleados en la publicación de artículos y código fuente de ejemplo. PHP posee una de las comunidades más grandes en Internet, de igual envergadura que la comunidad de Java.

Familiaridad con el lenguaje: en la universidad los lenguajes más utilizados por los desarrolladores de aplicaciones WWW son el ASP y el PHP.

De acuerdo con estas comparaciones realizadas, el PHP resulta ser un lenguaje favorito, por lo que es preciso utilizarlo en la implementación de la propuesta de solución del presente trabajo.

Servidor Web Apache.

Un servidor de páginas web es un programa que permite acceder a páginas web alojadas en un ordenador. Hoy en día Apache es el servidor web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un *software* de código abierto que funciona sobre cualquier plataforma. Desde su origen ha evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad, surgió en abril de 1996 y ya en julio del 2002 era utilizado por el 57% de los sitios web de Internet.

Tiene capacidad para servir páginas tanto de contenido estático, para lo que serviría sencillamente un viejo ordenador 486, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información, es muy potente y altamente configurable.

Los servidores web suministran páginas web a los navegadores que lo solicitan. En términos más técnicos, los servidores web soportan el Protocolo de Transferencia de Hipertextos como HTTP (*Hyper Text Transfer Protocol*), el estándar de Internet para comunicaciones web. Usando HTTP, un servidor web envía páginas web en HTML y *Common Gateway Interface* (CGI), así como otros tipos de *scripts* a los navegadores o *browsers* cuando éstos los requieren. Cuando un usuario hace clic sobre un enlace a una página web, se envía una solicitud al servidor web para localizar los datos nombrados por ese enlace, el servidor web recibe esta solicitud y suministra los datos que le han sido solicitados o bien devuelve un mensaje de error.

El servidor Apache es un gran *software* que está completamente estructurado en módulos, está dividido en muchos fragmentos de código quienes hacen referencia a diferentes aspectos o funcionalidades del servidor web. Esta modularidad es intencionada pues la configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo.

Los módulos de Apache son clasificables en tres categorías:

- Módulos Base: módulo con las funcionalidades básicas de Apache.
- Módulos Multiproceso: son los que se responsabilizan de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.

- Módulos Adicionales: cualquier otro módulo que le añada algún tipo de funcionalidad al servidor web.

Framework (Marco de trabajo).

Desde principios de este siglo el lenguaje de programación PHP ha sido encapsulado casi en su totalidad. Aprovechando las amplias posibilidades que brinda para el trabajo con redes, servicios web, bases de datos y desarrollo general de aplicaciones, muchas comunidades de desarrollo, compañías y programadores han creado esta nueva tendencia de encapsular y organizar la mayoría de las funcionalidades a pequeños paquetes llamados Frameworks (marcos de trabajo), caracterizados por tener gran flexibilidad, reutilización, extensibilidad y rendimiento. [8]

Los marcos de trabajo soportan conexiones a varios sistemas gestores de bases de datos como MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL y SQLite. Muchos están basados en el patrón Modelo-Vista-Controlador y por lo tanto, tienen buenos subsistemas que gestionan plantillas o vistas y proveen al desarrollador facilidades para manipular peticiones y acoplar la información que brinda el modelo de datos con la forma en que es presentada.

Cakephp.

Cakephp es un framework de desarrollo de aplicaciones web escrito en PHP, creado sobre los conceptos de Ruby on Rails. Facilita al usuario la interacción con la base de datos mediante el uso de *Active Record*. Además hace uso del patrón Modelo Vista Controlador. Características de Cakephp.

- Compatible con PHP4 y PHP5.
- CRUD de la base de datos integrado.
- URLs amigables.
- Sistema de plantillas rápido y flexible.
- *Helpers* para AJAX, JavaScript, HTML, forms y más.
- Trabaja en cualquier subdirectorio del sitio.
- Validación integrada.

- Listas de control de acceso.
- Componentes de seguridad y sesión.

Symfony.

Hasta ahora es el marco de trabajo que más documentación oficial posee. Con varios libros publicados por sus creadores como *"The Definitive Guide To Symfony"* y *"Practical Symfony"*, cada programador tiene disponible una ayuda constante sobre cada detalle de la innumerable cantidad de clases, funciones y utilidades que brinda. [4]

Su grado de madurez mejora con la utilización del I18N (Internacionalización y traducción de interfaces) y un novedoso sistema de *plug-in* lo cual lo hace mucho más extensible. Se adapta a las nuevas tecnologías soportando distintos tipos de formatos para peticiones como *JSON, XML, RDF, ATOM y YAML*, es decir, una aplicación desarrollada en Symfony puede ser tan extensible como para comunicarse y brindar servicios web a dispositivos móviles de última generación como el iPhone. [5]

Su sistema de *helpers* (ayudantes) es tan amplio que abarca todas las fases y modalidades de la construcción de aplicaciones web:

- Mecanismos de escape de valores para contrarrestar ataques XSS.
- Manipulación del sistema de cache del PHP para optimizar la rapidez del sistema.
- Formateo de fechas y cifras que guían al usuario a una mejor comprensión de la información mostrada.
- Encapsulación de numerosas funciones *JavaScript* para mejorar el trabajo con las plantillas y modernizar el sistema.
- Validadores basados en expresiones regulares para el uso de formularios que manipularán datos obtenidos de los usuarios.

Es importante mencionar que los datos almacenados son muy útiles en la medida que se puedan transformar en información adecuada para las personas que los necesitan. Esta plataforma tecnológica que se utiliza tiene facilidades para convertir los datos en información y poder entregarlos a los usuarios de forma que sean útiles y aumenta su reutilización. [4]

Selección del framework a utilizar

Se decidió utilizar el framework Symfony porque posee gran cantidad de documentación disponible, es uno de los más utilizados en el mundo, incluye varias herramientas que permiten automatizar las tareas más comunes de la ingeniería del software y es recomendable para desarrollar aplicaciones webs complejas en las que es importante la productividad de su desarrollo, la facilidad de su mantenimiento y su seguridad.

***JavaScript*.**

JavaScript es un lenguaje de *scripts* desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML. Se utiliza embebido en el código HTML, entre las *tags* <*script*> y <*/script*>. [19]

Es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias *JavaScript* contenidas en una página HTML y ejecutarlas adecuadamente.

JavaScript es un lenguaje orientado a objetos. El modelo de objetos de *JavaScript* está reducido y simplificado, pero incluye los elementos necesarios para que los *Scripts* puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador.

Cuando un usuario pincha sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante *JavaScript* se pueden desarrollar *Scripts* que ejecuten acciones en respuesta a estos eventos. En este caso actúa como un lenguaje orientado a eventos. [19]

Open LDAP.

LDAP www.openldap.org (Implementación libre del Protocolo ligero de acceso a directorios) es un protocolo para comunicaciones entre servidores y clientes LDAP. Estos servidores almacenan "carpetas" las cuales son accedidas por los clientes LDAP. Se le llama ligero (*lightweight*) porque es un pequeño y fácil protocolo que fue derivado del X.500 DAP (Protocolo de acceso a directorios) definidos en la capa de red del modelo OSI (Interconexión de Sistemas Abiertos). No se limita a información de contacto o incluso a información acerca de las personas; se usa también para buscar certificados encriptados, apuntadores a impresoras y otros servicios en la red, además, tiene la posibilidad de que una contraseña de un usuario pueda ser utilizada para acceder a varios servicios. Este protocolo es

muy apropiado para información almacenada en forma de carpetas donde generalmente se usen búsquedas rápidas y se actualice de forma no frecuente. LDAP no define cómo los programas funcionan, sino el lenguaje usado por programas clientes para comunicarse con el servidor, o servidores con servidores.

La mayoría de los clientes LDAP solo leen de un servidor. Las habilidades de búsquedas que tienen algunos clientes como los programas de correos pueden variar mucho, pocos pueden actualizar o introducir información. La encriptación de datos no está incluida y además, generalmente se requiere de una conexión SSL (Protocolo de Capa de Conexión Segura) para acceder al servidor.

El directorio LDAP de la UCI provee, aparte del sistema de autenticación, información básica y oficial en cuanto a los usuarios del dominio.

***Visual Paradigm* (Paradigma Visual)**

Visual Paradigm (Paradigma Visual) para UML 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. El lenguaje de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad con menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Visual Paradigm UML Professional puede generar código de un modelo de clase y también la estructura de base de datos relacional (tablas o esquemas) adecuados a la persistencia de mantener los datos contenidos en las clases de entidad llamada. *Visual Paradigm* UML permite diferentes puntos de partida:

- Técnicas de ingeniería inversa del diagrama entidad relación de una base de datos existente, y generar juego clases de código.
- Comenzar con un modelo de clase y generar un diagrama entidad relación, tablas de bases de datos y el código de la misma.
- Comenzar con un diagrama entidad relación y generar un modelo de clase, tablas de bases de datos y el código de la misma.

El código generado consta de:

- Las propias clases para los objetos modelados en los diagramas de clases.
- Clases de controlador de prestación de lectura, escritura y eliminación de las funciones para las entradas de la base de datos.
- Código de ejemplo para la generación de datos en la base de datos, para la lectura y eliminado.

Netbeans.

El Netbeans es un *IDE,* una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE Netbeans. El IDE Netbeans es un producto libre y gratuito sin restricciones de uso.

Es un IDE de código abierto escrito completamente en Java usando la plataforma Netbeans. El Netbeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en *Ant*, control de versiones y *refactoring*. [14] [22]

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- Administración de ventanas.
- *Framework* (marcos de trabajo) basado en asistentes (diálogo paso a paso).

Sistema Gestor de bases de datos.

Un Sistema de Gestión de Bases de Datos (SGBD) podría definirse como un paquete generalizado de *software* que es ejecutado en un ordenador anfitrión, quien centraliza los accesos a los datos y actuando de interfaz mediando entre los datos físicos y el usuario que realiza la petición. Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la

base de datos, el control de acceso, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias, inconsistencias, tratar la concurrencia y mantener la integridad de los datos almacenados. [3]

Un SGBD posee los siguientes objetivos específicos:

- Independencia de los datos y los programas de aplicación.
- Minimización de la redundancia de los datos almacenados.
- Integración y sincronización de las bases de datos.
- Integridad de los datos.
- Seguridad y protección de los datos.
- Facilidad de manipulación de la información.
- Control centralizado.

La información es representada a través de tuplas, las cuales describen el fenómeno, proceso o ente de la realidad objetiva que está siendo analizada y es representada a través de tablas.
Entre los SGBD más comúnmente empleados en el mundo se encuentran *Oracle, MySQL, Microsoft SQL Server, PostgreSQL, Internase*, DB2, entre otros. Todos los aquí expuestos presentan un enfoque relacional con una excelente base matemática centrada en el Álgebra Relacional.

Todos los SGBD expuestos anteriormente facilitan el trabajo con la base de datos y poseen características comunes que los diferencian, por ejemplo:

Oracle: conocido como uno de los mejores gestores del mundo requiere de una licencia para su posterior empleo, es necesario pagar un alto precio para poder hacer uso de este SGBD. [17]

Microsoft SQL Server: no es multiplataforma, solo puede ser empleado en conjunto al sistema operativo Windows que es propiedad de la gran compañía Microsoft Corp.

MySQL: es un SGBD de fuente abierta, especial para aplicaciones de pequeña y de mediana envergadura. [16]

PostgreSQL: es también un SGBD de fuente abierta con mayores prestaciones que *MySQL*, incluye además gran potencia para aplicaciones de mayor tamaño. Soporta un subconjunto de *SQL92* mayor que el soportado por MySQL. [18]

Como SGBD es seleccionado.

PostgreSQL es el SGBD seleccionado por las múltiples ventajas que este ofrece con respecto a los demás gestores y por compatibilidad con el sistema actual.

PostgreSQL.

PostgreSQL es un SGBD Objeto-Relacionales (ORDBMS) libre, liberado bajo la licencia de BSD (Berkeley *Software Distribution*).

Es una alternativa mucho más eficaz a otros sistemas de bases de datos de código abierto (MySQL, Firebird y MaxDB), así como a sistemas propietarios como *Oracle y Microsoft SQL Server. PostgreSQL* ha sido desarrollado de varias formas desde el año 1997. En 1994, Andrew Yu y Jolly Chen añadieron un intérprete de lenguaje SQL a *Postgres*. Postgre95 fue lanzada a la WWW para encontrar su propio hueco en el mundo como un hijo de dominio público y código abierto del código original *Postgres* de *Berkeley*.

PostgreSQL es ampliamente considerado como el sistema de bases de datos de fuente abierta más avanzado del mundo. Posee un gran número de características que históricamente solo pueden ser vistas en productos comerciales de alta confiabilidad.

A continuación se refleja una lista de algunas de las principales características, a partir de *PostgreSQL* 7.1:

- DBMS Objeto-Relacional: *PostgreSQL* aproxima grandemente los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte para múltiples usuarios, transacciones, consultas optimizadas, herencia, arreglo, etc.
- Altamente Extensible: *PostgreSQL* soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.

- Soporte SQL comprensivo: PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (join) SQL92.Integridad Referencial: *PostgreSQL* soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- Lenguajes Procedurales: *PostgreSQL* tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo nominado *PL/pgSQL*. Este es comparable al lenguaje procedural de *Oracle, PL/SQL.*
- Otra de las tantas ventajas de *PostgreSQL* es su habilidad para usar *Perl, Python*, o TCL (*Tool Command Language*) como lenguaje procedural embebido.

Lenguaje Unificado de Modelado (UML)

UML (*Unified Modeling Language* o Lenguaje de modelado unificado) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de *software*.

Los creadores pretendieron mediante la creación de este lenguaje, unificar las experiencias acumuladas sobre técnicas de modelado e incorporar las mejores prácticas en un acercamiento estándar. El UML permite a los creadores de sistemas generar diseños que capturen sus ideas en forma convencional y fácil de comprender para comunicar todas las personas que estén vinculadas en el desarrollo del *software*, esto es llevado a cabo mediante un conjunto de símbolos y diagramas.

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todas aquellas personas que de una u otra forma estén involucrados en el proceso de desarrollo. Un modelo UML es lo que supuestamente hará el sistema pero no como lo hará.

Seguidamente se exponen de forma general las principales características:

- Lenguaje unificado para la modelación de sistemas.
- Tecnología orientada a objetos.
- El cliente participa en todas las etapas del proyecto.
- Corrección de errores viables en todas las etapas.

- Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y Cliente/Servidor.

Metodologías.

Ingeniería de *software*

La ingeniería de *software* es una tecnología que indica "COMO" construir técnicamente un *software*: económico, fiable y que funcione eficientemente. Dentro de esta se encuentra la metodología que se va a utilizar para el desarrollo del proyecto.

Razones de la Ingeniería de *software*

Estudiar ¿cuáles son las actividades que organizadas, haciendo uso racional de los recursos y apoyándose en técnicas y herramientas, logran la mayor eficiencia al construir un *software*?

Metodología

La **metodología**, dentro de la ingeniería de *software*, se encarga de elaborar estrategias de desarrollo de *software* que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

RUP

El **Proceso Unificado Racional** (*Rational Unified Process* en inglés, habitualmente resumido como **RUP**) es un proceso de desarrollo de *software* y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP está basado en 5 principios clave que son:

1. Adaptar el proceso.
2. Balancear prioridades.
3. Demostrar valor iterativamente.

4. Elevar el nivel de abstracción.

5. Enfocarse en la realidad.

El ciclo de vida RUP

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada una de ellos, estos a la vez se dividen en fases que finalizan con un hito donde se debe tomar una decisión importante:

1. **Inicio**: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos.

2. **Elaboración**: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.

3. **Construcción**: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.

4. **Transición**: se Instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Los elementos de RUP son:

- Actividades: Procesos que se determinan en cada iteración.
- Trabajadores: Personas o entes involucradas en cada proceso.
- Artefactos: Puede ser un documento, un modelo, o un elemento de modelo.

XP

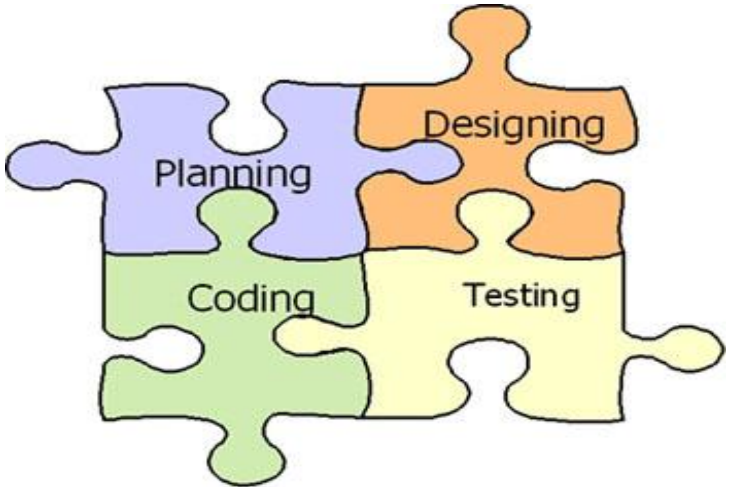

Figura #1 Representación de las fases de la metodología XP

La Programación Extrema es una metodología ligera de desarrollo de *software* que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Las cuatro variables de esta metodología son:

1. Coste: Máquinas, especialistas y oficinas.
2. Tiempo: Total y de Entregas.
3. Calidad: Externa e Interna.
4. Alcance: Intervención del cliente.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. [8]

XP se basa en retroalimentación 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. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. [8] [9]

XP surgió como respuesta y posible solución a los problemas derivados del cambio en los requerimientos. Se plantea como una metodología a emplear en proyectos de riesgo. Con ella se aumenta la productividad. [9]

Características de la Metodología XP.

El desarrollo bajo XP tiene características que lo distinguen claramente de otras metodologías:

- Los diseñadores y programadores se comunican efectivamente con el cliente y entre ellos mismos.
- Los diseños del *software* se mantienen sencillos y libres de complejidad o pretensiones excesivas.
- Se obtiene retroalimentación de usuarios y clientes desde el primer día gracias a las baterías de pruebas.
- El *software* es liberado en entregas frecuentes tan pronto como sea posible.
- Los cambios se implementan rápidamente tal y como fueron sugeridos.
- Las metas en características, tiempos y costos son reajustadas permanentemente en función del avance real obtenido.
- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustancial del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierten en miembros del equipo. [24]

Prácticas de XP

La mayoría de estas prácticas no son nuevas, sino que han sido reconocidas por la industria como las mejores prácticas durante años. En la XP, dichas prácticas son llevadas al extremo para que se obtenga algo mucho mejor que la suma de las partes, la metodología se basa en:

Planificación Incremental: La Programación Extrema asume que la planificación nunca será perfecta, y que varía en función de cómo varíen las necesidades del negocio. Por tanto, el valor

real reside en obtener rápidamente un plan inicial, y contar con mecanismos que permitan conocer con precisión la situación actual del proyecto. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar.

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

El hecho de que los test sean automatizados es el único modo de garantizar que todo funciona. Desde el punto de vista de la XP, si no hay test, las cosas sólo funcionan en apariencia.

El objetivo de los test no es corregir errores, sino prevenirlos. Los test siempre se escriben antes que el código a testear, no después: esto aporta un gran valor adicional, pues fuerza a los desarrolladores a pensar cómo se va a usar el código que escriben, situándolos en la posición de consumidores del *software*. Elaborar los test exige pensar por adelantado cuáles son los problemas más graves que se pueden presentar, y cuáles son los puntos dudosos.

Un efecto lateral importante de los test es que dan una gran seguridad a los desarrolladores: es posible llegar a hacer cambios más o menos importantes sin miedo a problemas inesperados, dado que proporcionan una red de seguridad. La existencia de test hace el código más seguro.

Diseño simple: Otra práctica fundamental de la Programación Extrema es utilizar diseños tan simples como sea posible. El principio es "utilizar el diseño más sencillo que consiga que todo funcione". Se evita diseñar características extras porque a la hora de la verdad la experiencia indica que raramente se puede anticipar qué necesidades se convertirán en reales y cuáles no. La XP exige no tener la ilusión de que un diseño puede resolver todas o gran parte de las situaciones futuras: lo que parece necesario cambia con frecuencia, es difícil acertar a priori.

La metodología XP define un "diseño tan simple como sea posible" como aquél que:

Pasa todos los test.

No contiene código duplicado.

Deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código.

Contiene el menor número posible de clases y métodos.

Pruebas Unitarias: se basa en las pruebas 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 si se adelantara a obtener los posibles errores.

Re - fabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo, cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

¿Qué es lo que propone XP?

1. Empieza en pequeño y añade funcionalidad con retroalimentación continua.
2. El manejo del cambio se convierte en parte sustancial del proceso.
3. El costo del cambio no depende de la fase o etapa.
4. No introduce funcionalidades antes que sean necesarias.
5. El cliente o el usuario se convierten en miembro del equipo.

Derechos del Cliente.

1. Decidir que se implementa.
2. Saber el estado real y el progreso del proyecto.

3. Añadir, cambiar o quitar requerimientos en cualquier momento.
4. Obtener lo máximo de cada semana de trabajo.
5. Obtener un sistema funcionando cada 3 ó 4 meses.

Lo fundamental en este tipo de metodología es:

1. La comunicación, entre los usuarios y los desarrolladores.
2. La simplicidad, al desarrollar y codificar los módulos del sistema.
3. La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Metodología seleccionada (XP)

Luego del estudio de las metodologías anteriores se decide utilizar XP principalmente porque el proyecto a desarrollar presenta poca complejidad, es una de las metodologías de desarrollo de *software* más exitoso en la actualidad, utilizadas para proyectos de corto plazo, pequeño equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Conclusiones.

En el capítulo se ha comentado el estado actual de desarrollo de sistemas automatizados de reservación en el ámbito mundial, nacional y en la universidad, exponiendo ejemplos de algunos sistemas automatizados de gestión de reservaciones que existen actualmente. Uno de los grandes problemas que se pueden presentar en la elaboración de un *software* es la selección de una herramienta inadecuada sobre todo si se trata del lenguaje que se piensa utilizar ya que cualquier inconformidad generada por insuficiencias del mismo conlleva a retrasos del flujo de actividades. Es por eso por lo que este paso dentro de la planificación de la aplicación debe hacerse de manera cuidadosa, para no tomar decisiones erróneas que puedan traer resultados catastróficos durante todo el proceso de desarrollo. Las tecnologías y formas de desarrollo evolucionan rápidamente, por lo tanto, las aplicaciones web evolucionan, son mantenidas y se adaptan a los nuevos entornos; pero es tarea

de todo buen desarrollador guiarlas teniendo en cuenta la actualización en los conocimientos adquiridos, planteándose además una propuesta de solución a la problemática que existe en la universidad.

Capítulo 2: Características del sistema. Planificación y diseño.

Introducción

En este capítulo se documentan las principales funcionalidades y procesos que ocurren en el sistema. Se plantean los requerimientos funcionales y no funcionales de la aplicación a desarrollar. Se pretende mostrar cómo está aprovechada la arquitectura y las posibilidades que proporcionan el marco de trabajo y las librerías utilizadas en la programación de aplicaciones web en el producto que se está tratando, con el objetivo de facilitar la comprensión del funcionamiento del mismo.

Objeto de estudio

Situación Problémica.

La UCI y su población crecen constantemente y con ellos los servicios que se brindan a la comunidad universitaria. Actualmente se está llevando a cabo una estrategia que posibilita la informatización y automatización de los servicios. La universidad ya cuenta con diversos sistemas que ofrecen a estudiantes, profesores y trabajadores hacer uso de algunos de estos servicios de forma óptima, pero a su vez, existen otros como son los de la peluquería, la barbería y la lavandería que no se encuentran informatizados y son de gran demanda. Se nota que la manera mediante la cual se realizan los servicios mencionados anteriormente es bastante engorroso y en ocasiones no se hace de la mejor forma, ni con la calidad requerida, resaltando además que todo se realiza de forma manual, no se cuenta con un detallado registro de las operaciones que se realizan y que los usuarios pierden demasiado tiempo esperando a ser atendidos, corriendo el riesgo en algunas ocasiones de no poder recibir dichos servicios. Por lo que no se logra una prestación placentera de estos servicios, aparecen muchas dificultades, no hay un control adecuado y se causan muchas incomodidades a los trabajadores de estos lugares y a los usuarios que los requieren. Por todos los problemas que esto trae consigo se hace necesario implementar un sistema para par la solución de los problemas planteados anteriormente.

Objeto de automatización.

Se desea garantizar un sistema que permita realizar cualquier tipo de reservación, ya sea para la peluquería, barbería y lavandería.

Información que se maneja.

La reservación de los procesos de barbería, peluquería y lavandería contiene implícita datos personales y datos propios de la reservación siendo esta información sensible. Se debe lograr, y es uno de los objetivos que persigue la implementación de la aplicación, que la manipulación de la misma sea de una manera segura y confiable, que no atente contra la autenticidad e integridad de la misma.

Propuesta del sistema.

Para resolver esta situación se propone implementar un sistema automatizado, seguro, que mejore la calidad y el tratamiento de la información de los servicios de barbería, peluquería y lavandería en la UCI.

Para lograrlo el sistema será implementado en una tecnología web, lo que hace posible el acceso a la aplicación desde todos los puntos de la UCI para los que se configure, constará con una base de datos para almacenar los valores entrados en los formularios HTML donde estarán validadas las entradas, en su diseño, se podrá ver el área de navegación en el cual se encontrarán las opciones que tendrán los usuarios según su nivel de acceso. Tendrá un área de trabajo en el cual se podrá agregar toda la información necesaria.

Modelo del negocio

El modelo de negocio se realiza con el propósito de comprender las características y actividades que se llevan a cabo en el contexto a automatizar. En él, se modelan los clientes y trabajadores del negocio y su interacción con los procesos y objetivos que lo componen.

Actores / Trabajadores del negocio

Un **actor** del negocio es algo o alguien fuera del sistema que interactúa con este, o sea, son todas aquellas personas o sistemas que obtienen un resultado de valor de uno o varios procesos del negocio.

Los **trabajadores** del negocio son aquellas personas o sistemas que están involucrados en uno o más procesos del negocio, que participan en ellos, pero no obtienen ningún resultado de valor.

Tabla #1 Actores del negocio.

Actor	Justificación
Cliente	Son todas aquellas personas que se benefician con el negocio, ya sea para obtener información o reportes de este.

Tabla #2 Trabajadores del negocio.

Trabajadores	Justificación
Recepcionista de la lavandería	Es la persona encargada de recoger los datos del cliente cuando este llega a realizar la reservación en la lavandería.
Administradora de la lavandería	Es la persona encargada de administrar la lavandería.
Recepcionista de la peluquería	Es la persona encargada de recoger los datos del cliente cuando este llega a realizar algún servicio en la peluquería.
Administradora de la peluquería.	Es la persona encargada de administrar la peluquería.
Peluquera	Persona encargada de realizar el pedido del cliente.
Recepcionista de la barbería	Es la persona encargada de recoger los datos del cliente cuando este llega a realizar algún servicio en la barbería.
Administradora de la barbería.	Es la persona encargada de administrar la barbería.
Barbero	Persona encargada de realizar el pedido del cliente.

Especificación de los requerimientos del *software*.

Requerimientos funcionales

1. Autenticar Usuario.
    - Introducir nombre de usuario y contraseña del dominio UCI.
    - Validar los datos introducidos.
    - Otorgar los accesos al usuario.
2. Gestionar Reservación.
    - Crear reservación
    - Modificar reservación
    - Cancelar reservación
3. Gestionar Roles.
    - Crear roles
    - Modificar roles
    - Eliminar roles
    - Asignación de permisos
4. Gestionar servicios
    - Crear servicios
    - Modificar servicios
    - Eliminar servicios
5. Gestionar usuarios
    - Crear usuarios
    - Modificar usuarios
    - Eliminar usuarios
    - Asignar roles
6. Gestionar Trabajadores
    - Crear trabajadores
    - Modificar trabajadores
    - Eliminar trabajadores
7. Gestionar Sesiones

- Crear sesiones
- Modificar sesiones
- Eliminar sesiones

8. Gestionar locales
    - Crear locales
    - Modificar locales
    - Eliminar locales
9. Gestionar Reportes
    - Crear reportes
    - Exportar reportes
10. Gestionar noticias
    - Crear noticias
    - Modificar noticias
    - Eliminar noticias

Requerimientos no funcionales:

1. Apariencia o Interfaz externa
    - Diseño encuadrado para una resolución de 800x600, pero preparado para verse en otras resoluciones.
    - El sistema debe tener una interfaz sencilla, intuitiva, amigable y debe mantener el formato de las páginas de otros módulos. En general fácil de usar y agradable a la vista del usuario.
2. Usabilidad
    - El sistema podrá ser usado por cualquiera persona con conocimientos básicos con el manejo de la computadora y de un ambiente web en sentido general.
3. Portabilidad
    - Permitir que el sistema se ejecute en un Sistema Operativo Windows 98, Linux o superior.
4. Rendimiento
    - Estará implementado sobre una tecnología web, facilitando su uso a través de la red.

- Tiempo de respuesta rápido. El sistema debe de tener un tiempo de respuesta rápido ante cualquier solicitud del usuario.

5. Seguridad
   - Establecer los distintos roles que establezcan las acciones que pueden realizar los usuarios.
   - Transmisión de los datos por la red a través de un protocolo seguro.
   - Verificación de acciones irreversibles (por ejemplo los eliminar).
   - Garantizar que la información sea modificada y vista solamente por aquel que tenga permiso para ello.
6. Confiabilidad:
   - La información manejada por el sistema estará protegida de accesos no autorizados.
   - Validar la entrada de los usuarios.
   - Mantenimiento. El sistema debe estar bien documentado, de forma tal que el tiempo de mantenimiento sea mínimo.
   - Preciso en la información. El sistema debe ser preciso en la información que proporciona
7. Integridad
   - La información manejada por el sistema debe será objeto de cuidadosa protección contra la corrupción y estados inconsistentes. Deberán existir mecanismos de chequeos de integridad.
8. Disponibilidad
   - Los usuarios autorizados tendrán acceso a la información en todo momento.
   - Disponible todo el tiempo. El sistema debe estar disponible todo el tiempo para trabajar de forma tal que se pueda acceder las 24 horas.
9. Funcionalidad
   - Mínima cantidad de páginas para ejecutar todas la funciones posibles (preferentemente que estén relacionadas).
10. Diseño e implementación.
    - Aplicación web escrita sobre el lenguaje de programación PHP 5.2.5
    - Usar el gestor de base de datos PostgreSQL.

- Usar como servidor web Apache.
- Desarrollar bajo el framework Symfony.
- Usar como metodología de desarrollo XP

Planificación.

Definición de los historiales de usuarios.

Tabla #3 HU Autentificar usuario.

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Autenticar Usuario
Modificación de Historia de Usuario Número:	
Usuario: Cliente	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Alta	**Puntos Reales:**
Descripción: Se brinda la posibilidad de que la persona que acceda al sistema introduzca sus datos (usuario contraseña) con la finalidad de verificar y otorgarle los permisos según el rol que ocupa dentro de la aplicación.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #4: HU Gestionar reservación.

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Gestionar reservación
Modificación de Historia de Usuario Número:	
Usuario: Cliente	**Iteración Asignada:** 1

Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Alta	**Puntos Reales:**
Descripción: Se realizan las acciones (previa autenticación) de crear, eliminar y modificar una reservación, en caso de los clientes que no tengan usuario UCI, debe acercarse al local y hacer la reservación en este.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #5: HU Gestionar roles

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Gestionar roles
Modificación de Historia de Usuario Número:	
Usuario: Súper-administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Alta	**Puntos Reales:**
Descripción: El súper-administrador del sistema tendrá la posibilidad de crear, eliminar y modificar roles, que previamente se le serán asignados a los diferentes usuarios del sistema.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #6: HU Gestionar servicios.

Historia de Usuario	
Número: 4	**Nombre Historia de Usuario:** Gestionar servicios
Modificación de Historia de Usuario Número:	
Usuario: Súper-administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Alta	**Puntos Reales:**
Descripción: El súper-administrador del sistema tendrá la posibilidad de adicionar, modificar y eliminar los servicios en dependencia a la disponibilidad de los mismos.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #7: HU Gestionar usuarios

Historia de Usuario	
Número: 5	**Nombre Historia de Usuario:** Gestionar usuarios
Modificación de Historia de Usuario Número:	
Usuario: Súper-administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Alta	**Puntos Reales:**
Descripción: El súper-administrador tendrá la posibilidad de adicionar usuarios en la aplicación, además de asignarle los permisos de este y el local donde estos ejercerán sus funciones.	

Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #8: HU Gestionar trabajadores

Historia de Usuario	
Número: 6	**Nombre Historia de Usuario:** Gestionar trabajadores
Modificación de Historia de Usuario Número:	
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Alta	**Puntos Reales:**
Descripción: El administrador podrá adicionar trabajadores en la aplicación, además de asignarle el local donde estos cumplirán con sus funciones laborales. También tendrá la posibilidad de eliminarlo o editar las funciones y el local donde este trabaja.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #9: HU Gestionar sesiones

Historia de Usuario	
Número: 7	**Nombre Historia de Usuario:** Gestionar sesiones.
Modificación de Historia de Usuario Número:	
Usuario: Administrador y Recepcionista	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:**

Riesgo en Desarrollo: Baja	**Puntos Reales:**
Descripción: Los administradores contarán con la posibilidad de insertar las diferentes jornadas laborales con las que se prestarán servicios en la aplicación.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #10: HU Gestionar locales

	Historia de Usuario
Número: 8	**Nombre Historia de Usuario:** Gestionar locales.
Modificación de Historia de Usuario Número:	
Usuario: Súper-administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Baja	**Puntos Reales:**
Descripción: El administrador contará con la posibilidad de crear nuevos locales para la prestación de los servicios que contará la aplicación. También tendrá la opción de modificarlos o eliminarlos en caso de que sea necesario.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #11: HU Gestionar reportes

	Historia de Usuario
Número: 9	**Nombre Historia de Usuario:** Gestionar reportes.
Modificación de Historia de Usuario Número:	

Usuario: Administrador y Recepcionista	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Baja	**Puntos Reales:**
Descripción: Se brinda la posibilidad a los administradores de exportar las solicitudes realizadas en formato PDF para ser trasladadas a otro lugar, además de poder sacar la recaudación de los diferentes servicios en un rango de tiempo, sabiendo las ganancias de los mismos.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Tabla #12: HU Gestionar noticias

Historia de Usuario	
Número: 10	**Nombre Historia de Usuario:** Gestionar noticias.
Modificación de Historia de Usuario Número:	
Usuario: Administrador y recepcionista	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:**
Riesgo en Desarrollo: Baja	**Puntos Reales:**
Descripción: Se posibilita la publicación de diferentes noticias en la aplicación referente a los servicios o generales, con el objetivo de informar a los usuarios que visiten el sitio.	
Observaciones:	
Prototipo de interfaz: [Imagen de cada una de las interfaces relacionadas con la HU.]	

Estimación de los esfuerzos por historia de usuario.

Durante la fase de planificación se realizará una estimación de los esfuerzos que costara implementar cada historia de usuario. Esto se expresa utilizando como medida un punto. Un punto se considera como una semana ideal de trabajo, donde los miembros del equipo de desarrollo trabajan el tiempo planificado sin ninguna interrupción. Esta estimación incluye todo el esfuerzo asociado a la implementación de la historia de usuario, por ejemplo, las pruebas unitarias, la integración y refactorización del código, y la preparación y ejecución de las pruebas de aceptación.

Tabla #13: Estimación de los esfuerzos por HU.

Historias de usuarios	Puntos de estimación
Autenticar usuario	1
Gestionar reservación	0.5
Gestionar roles	0.5
Gestionar servicios	0.5
Gestionar usuarios	0.5
Gestionar trabajadores	0.5
Gestionar sesiones	0.3
Gestionar locales	0.5
Gestionar reportes	1
Gestionar noticias	0.4

Plan de iteraciones.

Después de ser descritas las historias de usuarios y estimado el esfuerzo propuesto para la realización de cada una de ellas, se procede a la implementación del sistema. Este plan especifica exactamente cuales de las historias de usuario será implementada para cada iteración del sistema y las posibles fechas para esas liberaciones.

Sobre la base de lo anterior planteado se decide realizar el sistema en 2 iteraciones, las cuales se detallan a continuación:

Iteración #1

En esta iteración se tiene como objetivo la implementación de las historias de usuarios de importancia en el negocio pero que a su vez tenga baja prioridad en el negocio. Al finalizar se tendrá la implementación de las funcionalidades descritas en las historias de usuarios de la 1 hasta la 9 las cuales se refieren a todo el proceso de reservación en los módulos de lavandería, barbería, peluquería y el sistema de notificaciones del módulo de administración de la aplicación.

Iteración #2

En esta iteración se tendrá como objetivo la implementación de las historias de usuarios con alto nivel de prioridad para el negocio así como algunas de media prioridad. Al final se tendrá la implementación de las historias con alto y medio nivel de prioridad que no generan dependencias con otras historias de usuarios, las que se pueden comprobar una vez finalizada su implementación sin depender de otras implementaciones.

Plan de duración de las iteraciones.

Como parte del ciclo de vida de un proyecto utilizando la metodología 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 el que se cuenta. Este plan tiene como finalidad mostrar la duración de cada iteración, así como el orden en el que serán implementadas las HU (Historia de Usuario) en cada una de las mismas.

Tabla #14: Plan de duración de las iteraciones

Iteraciones.	Orden de las HU a implementar.	Duración total de la iteración.
Iteración #1	1. Autenticar usuario 2. Gestionar reservación 3. Gestionar roles 4. Gestionar servicios 5. Gestionar usuarios 6. Gestionar sesiones	2 semanas
Iteración #2	1. Gestionar noticias 2. Gestionar trabajadores 3. Gestionar locales 4. Gestionar reportes	2 semanas

Plan de entregas.

A continuación se presenta el plan de entregas elaborado para la fase implementación. Para facilitar la elaboración de dicho plan se acoplaron las funcionalidades referentes a un mismo tema en módulos, quedando estos de la siguiente manera:

Tabla #15: Plan de entregas.

Módulos	Historias de usuarios asociadas
sfGuardAuth	Autenticar usuario
solicitudes	Gestionar Reservación
usuarios	Gestionar usuarios
permisos	Gestionar Roles
sesión	Gestionar Sesiones
servicio	Gestionar servicios
grupos	Gestionar Trabajadores
local	Gestionar locales
reportes	Gestionar Reportes
noticia	Gestionar noticias

Como producto del plan de entregas se hacen raleases al sistema en las fechas que se indican a continuación:

Tabla #16: Módulos y HU asociadas.

Módulos	Principio de la 1ra iteración. 2da y 3ra semana de abril	Final de la 1ra iteración. 4ta semana de abril 1ra semana de mayo	Principio de la 2da iteración. 2da semana de mayo.	Final de la 3ra iteración. 2da y 3ra semana de mayo
solicitudes	1.0	1.1	1.2	Finalizado
usuarios	1.0	1.1	1.2	Finalizado
permisos	1.0	Finalizado	Finalizado	Finalizado
sesión	1.0	Finalizado	Finalizado	Finalizado

servicio	1.0	Finalizado	Finalizado	Finalizado
grupos	1.0	1.1	1.2	Finalizado
local	1.0	Finalizado	Finalizado	Finalizado
reportes	1.0	1.1	1.2	-
noticia	1.0	Finalizado	Finalizado	Finalizado

Diseño.

Para el diseño de aplicaciones la metodología XP no requiere de diagramas usando notación UML, en lugar de esto se utilizan otras técnicas como las tarjetas CRC (Cargo o Clase, Responsabilidad y Colaboración), a continuación se muestran las tarjetas CRC generadas en la fase de diseño de la aplicación.

Tarjetas CRC (Cargo o Clase, Responsabilidad, Colaboración)

Tabla #17: Tarjeta CRC Módulo sfGuardAuth

Módulo sfGuardAuth	
Funcionalidades	**Colaboraciones (módulos)**
1. Autenticación	Solicitudes Usuarios Permisos Sesión Servicio Grupos Local Reportes Noticia

Tabla #18: Tarjeta CRC Módulo solicitudes

Módulo solicitudes	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear solicitud. 2. Modificar solicitud. 3. Eliminar solicitud.	Sesión Servicio Local Noticia Reportes

Tabla #19: Tarjeta CRC Módulo usuarios

Módulo usuarios	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear usuarios. 2. Eliminar usuarios. 3. Modificar usuarios 4. Asignar roles	Grupos Permisos Solicitudes

Tabla #20: Tarjeta CRC Módulo permisos

Módulo permisos	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear permisos. 2. Modificar permisos. 3. Eliminar una permios.	Grupos Usuarios.

Tabla #21: Tarjeta CRC Módulo sesión.

Módulo sesión	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear sesión. 2. Modificar sesión. 3. Eliminar sesión.	Solicitudes Servicio Local

Tabla #22: Tarjeta CRC Módulo servicio

Módulo servicio	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear servicio. 2. Modificar servicio. 3. Eliminar servicio.	Solicitudes Sesión Local.

Tabla #23: Tarjeta CRC Módulo grupos.

Módulo grupos	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear grupos. 2. Modificar grupos. 3. Eliminar grupos.	Usuarios Permisos

Tabla #24: Tarjeta CRC Módulo local.

Módulo local	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear local. 2. Modificar local. 3. Eliminar local.	Servicio Sesión

	Solicitudes

Tabla #25: Tarjeta CRC Módulo reportes.

Módulo reportes	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear reportes. 2. Modificar reportes. 3. Eliminar reportes.	Solicitudes

Tabla #26: Tarjeta CRC Módulo noticia.

Módulo noticia	
Funcionalidades	**Colaboraciones (módulos)**
1. Crear noticia. 2. Modificar noticia. 3. Eliminar noticia.	Solicitudes

Diseño de la base de datos.

El diseño de la base de datos es algo que no se puede pasar por alto, producto de que uno de sus objetivos fundamentales es brindar la persistencia al modelo que se describe en el epígrafe anterior. En esta investigación fueron construidos dos modelos para la representación de los datos persistentes: el Modelo Lógico de Datos y el Modelo Físico de Datos, además de las descripciones de las tablas de estos modelos. (**Ver anexo 1).**

MVC (Modelo Vista Controlador)

Modelo Vista Controlador (MVC) 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. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página *HTML* y el código que provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos. (**Ver Anexo 2**)

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

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos. En MVC corresponde al controlador.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace)
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (*handler*) o *callback*.
- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta dirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue

sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.

- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Patrones de diseño

Un patrón es un modelo el cual se sigue para realizar algo. Los patrones surgen de la experiencia de los seres humanos de tratar y a la vez lograr ciertos objetivos. Los patrones capturan la experiencia existente y probada para promover nuevas prácticas.

Los Patrones de Diseño (*Design Patterns*) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo del software y otros ámbitos referentes al diseño de interacción o interfaz.

Estos se encuentran divididos en tres grandes categorías:

- Patrones Creacionales: solucionan problemas de creación de instancias. Y ayudan a encapsular y abstraer dicha creación.
- Patrones Estructurales: solucionan problemas de composición (agregación) de clases y objetos.
- Patrones de Comportamiento: soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.

Un Patrón de diseño es:

- Una solución estándar para un problema común de programación.
- Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- Un proyecto o estructura de implementación que logra una finalidad determinada.
- Un lenguaje de programación de alto nivel.
- Una manera más práctica de escribir ciertos aspectos de la organización de un programa.
- Conexiones entre componentes de programas.
- La forma de un diagrama de objeto o de un modelo de objeto.

Los patrones de diseño han contribuido grandemente a la flexibilidad. Y en adición, han demostrado ser una forma muy útil de reutilizar diseño, ya que ellos no solo nombran, abstraen e identifican aspectos clave de estructuras comunes de diseño, sino que generalmente son descritos en una forma

específica documental, haciendo su comprensión y aplicación fácil para el conjunto de desarrolladores.

Se pueden destacar que los beneficios que produce un patrón pueden ser medidos en varios aspectos:

Contribuyen a reutilizar diseños, identificando aspectos clave de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. La importancia de la reutilización del diseño no es despreciable, ya que esta provee numerosas ventajas: reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad, eficiencia y consistencia de los diferentes diseños, y proporciona un considerable ahorro en la inversión.

Mejoran (aumentan, elevan) la flexibilidad, modularidad y extensibilidad, factores internos e íntimamente relacionados con la calidad percibida por el usuario.

Los patrones incrementan las diferentes gamas de diseño, ayudando a diseñar desde un mayor nivel de abstracción.

Patrones GRASP usados:

Creador: Todos los módulos del sistema tienen una clase *actions.class.php* que contiene las acciones definidas para los mismos y es en ella misma donde se ejecutan las funciones que hacen al sistema funcional. En esta clase las acciones se encargan de crear los objetos de las clases que representan las entidades, evidenciando de este modo que la clase *actions.class.php* es el "creador" de las entidades.

Experto: Se evidencia este patrón puesto que Doctrine es la librería externa que utiliza Symfony para realizar su capa de abstracción al modelo de datos, encapsulando toda la lógica de los datos y generando las clases con funcionalidades comunes de las entidades. Por tanto, cada clase creada por Doctrine a partir de una entidad es experta en manejar su información.

Controlador: Todas las peticiones web son manejadas por un solo controlador frontal (*frontend.php*), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Alta Cohesión: Symfony permite la asignación de responsabilidades con alta cohesión, por ejemplo la clase *actions.class.php* tiene la responsabilidad para definir las acciones sobre las plantillas y colabora con otras para realizar diferentes operaciones y crear objetos, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, proporcionando que el software sea flexible frente a grandes cambios.

Bajo Acoplamiento: La clase *actions.class.php* hereda solamente de *sfActions* para lograr un bajo acoplamiento de clases.

Patrones GOF usados:

Creacionales:

Singlenton (Instancia Única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Es el caso del controlador frontal, donde hay una llamada a la función *sfContext::getInstance()* que garantiza que siempre se acceda a la misma instancia.

Abstrac Factory (Fábrica Abstracta): Se utiliza este patrón al trabajar con objetos de distintas familias de manera que no se mezclen entre sí, haciendo transparente el tipo de familia concreta que se esté usando. Cuando el marco de trabajo necesita, por ejemplo crear un nuevo objeto, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

Estructurales:

Decorator (Decorador): Añade funcionalidad a una clase, dinámicamente. El archivo layout.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el *Layout* decorando la misma.

Composite (Objeto compuesto): Permite tratar objetos compuestos como si de uno simple se tratase. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

Conclusiones

Durante el desarrollo de este capítulo se hace referencia a todo lo concerniente a la fase de Planificación y el Diseño del Sistema propuesto, haciendo una descripción de cada uno de los artefactos generados en el transcurso de esta fase. Se asume la implementación de iteraciones la cual fue concebida y debidamente detallada. En esta fase es muy importante y necesario el permanente diálogo entre el cliente y el equipo de desarrollo.

Capítulo 3: Implementación y prueba.

Introducción.

La Metodología XP plantea que la implementación de un *software* debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para incrementar la visión de los desarrolladores con la opinión de éste.

Durante el transcurso de las iteraciones se realiza la implementación de las HU (Historias de Usuarios) seleccionadas para ser realizadas en cada una de ellas. Al principio de estas se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de ser necesario. Como parte de este plan se descomponen las HU en tareas de desarrollo, asignando a un grupo de desarrollo (o persona) la responsabilidad de la implementación. Estas tareas son para el uso estricto de los programadores, pueden ser escritas en un lenguaje técnico y no necesariamente entendible por el usuario.

Implementación.

Iteración #1

En esta iteración se implementarán las HU con bajo nivel de complejidad con el fin de obtener algunas funcionalidades para mostrarle al cliente y obtener información de este, para tomar decisiones con respecto a los pasos a seguir para el desarrollo de las otras iteraciones.

Tabla #27: Módulos abordados en la iteración #1.

Módulos	Historias de usuarios	Tiempo de implementación (Puntos estimados)	
		Estimación	Real
Solicitud	Gestionar reservación	0.5	0.4
Usuarios	Gestionar usuarios	0.4	0.2
Permisos	Gestionar permisos	0.4	0.2
Sesión	Gestionar sesión	0.4	0.2
Servicio	Gestionar servicio	0.4	0.2

A continuación se describen las tareas efectuadas para cada uno de los módulos desarrollados en la iteración #1:

Tabla #28: Tarea de ingeniería 1.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** HU_1
Nombre Tarea: Creación de la tabla sfGuardUser en al BD.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se crea en la BD la tabla correspondiente a los usuarios registrados	

Tabla #29: Tarea de ingeniería 2.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** HU_1
Nombre Tarea: Implementar la interfaz de autentificación.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se crea una interfaz para que el usuario inserte sus datos (usuario y contraseña).	

Tabla #30: Tarea de ingeniería 3.

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** HU_2
Nombre Tarea: Creación de la tabla tb_solicitudes en la BD.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se crea en la BD la tabla correspondiente a las solicitudes (reservaciones) hechas por los clientes.	

Tabla #31: Tarea de ingeniería 4.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** HU_2
Nombre Tarea: Implementar el adicionar reservación.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se implementan las funcionalidades que permitan adicionar una reservación.	

Tabla #32: Tarea de ingeniería 5.

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** HU_2
Nombre Tarea: Implementar el cancelar reservación.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe	
Descripción: Se implementan las funcionalidades que permitan cancelar una reservación.	

Tabla #33: Tarea de ingeniería 6.

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** HU_2
Nombre Tarea: Implementar el modificar reservación.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se implementan las funcionalidades que permitan modificar una reservación.	

Tabla #34: Tarea de ingeniería 7.

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** HU_3
Nombre Tarea: Creación de la tabla tb_sf_guard_permission en la BD.	

Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se crea en la BD la tabla correspondiente a los permisos de los usuarios.	

Tabla #35: Tarea de ingeniería 8.

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** HU_3
Nombre Tarea: Implementar el adicionar rol.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se implementan las funcionalidades que permita adicionar un rol.	

Tabla #36: Tarea de ingeniería 9.

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** HU_3
Nombre Tarea: Implementar el cancelar rol.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	

Descripción: Se implementan las funcionalidades que permita cancelar un rol.

Tabla #37: Tarea de ingeniería 10.

Tarea de Ingeniería	
Número Tarea: 10	**Número Historia de Usuario:** HU_3
Nombre Tarea: Implementar el modificar reservación.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se implementan las funcionalidades que permita modificar un rol.	

Tabla #38: Tarea de ingeniería 11.

Tarea de Ingeniería	
Número Tarea: 11	**Número Historia de Usuario:** HU_4
Nombre Tarea: Creación de la tabla tb_servicio en la BD.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se crea en la BD la tabla correspondiente a los servicios con que cuenta la aplicación.	

Tabla #39: Tarea de ingeniería 12.

Tarea de Ingeniería	
Número Tarea: 12	**Número Historia de Usuario:** HU_4
Nombre Tarea: Implementar el modificar servicio.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita modificar un servicio en la aplicación.	

Tabla #40: Tarea de ingeniería 13.

Tarea de Ingeniería	
Número Tarea: 13	**Número Historia de Usuario:** HU_4
Nombre Tarea: Implementar el adicionar servicio.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita adicionar un servicio en la aplicación.	

Tabla #41: Tarea de ingeniería 14.

Tarea de Ingeniería	
Número Tarea: 14	**Número Historia de Usuario:** HU_4

Nombre Tarea: Implementar el eliminar servicios.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita eliminar un servicio en la aplicación.	

Tabla #42: Tarea de ingeniería 15.

	Tarea de Ingeniería
Número Tarea: 15	**Número Historia de Usuario:** HU_5
Nombre Tarea: Implementar el modificar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita modificar un usuario en la aplicación.	

Tabla #43: Tarea de ingeniería 16.

	Tarea de Ingeniería
Número Tarea: 16	**Número Historia de Usuario:** HU_5
Nombre Tarea: Implementar el adicionar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**

Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez.	
Descripción: Se implementa la funcionalidad que permita adicionar un usuario en la aplicación.	

Tabla #44: Tarea de ingeniería 17.

	Tarea de Ingeniería
Número Tarea: 17	**Número Historia de Usuario:** HU_5
Nombre Tarea: Implementar el eliminar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez.	
Descripción: Se implementa la funcionalidad que permita eliminar un usuario en la aplicación.	

Tabla #45: Tarea de ingeniería 18.

	Tarea de Ingeniería
Número Tarea: 18	**Número Historia de Usuario:** HU_7
Nombre Tarea: Creación de la tabla tb_sesion en la BD.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**

Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez
Descripción: Se crea en la BD la tabla correspondiente a las sesiones.

Tabla #46: Tarea de ingeniería 19.

Tarea de Ingeniería	
Número Tarea: 19	**Número Historia de Usuario:** HU_7
Nombre Tarea: Implementar el modificar sesión.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez.	
Descripción: Se implementa la funcionalidad que permita modificar un servicio en la aplicación.	

Tabla #47: Tarea de ingeniería 20.

Tarea de Ingeniería	
Número Tarea: 20	**Número Historia de Usuario:** HU_7
Nombre Tarea: Implementar el adicionar sesión.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez.	
Descripción: Se implementa la funcionalidad que permita adicionar una sesión en la	

aplicación.

Tabla #78: Tarea de ingeniería 21.

Tarea de Ingeniería	
Número Tarea: 21	**Número Historia de Usuario:** HU_7
Nombre Tarea: Implementar el eliminar sesión.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita eliminar una sesión en la aplicación.	

Iteración #2.

En esta iteración se implementan las historias de usuario con un nivel de complejidad alto y se continúan perfeccionando los módulos implementados en la iteración anterior.

Tabla #49: Módulos abordados en la iteración #2.

Módulos	Historias de usuarios	Tiempo de implementación (Puntos estimados)	
		Estimación	Real
noticias	Gestionar noticias.	0.3	0.2
Trabajadores	Gestionar trabajador	0.4	0.2
Locales	Gestionar local	0.3	0.2
Reportes	Gestionar reportes	0.7	0.5

A continuación se describen las tareas efectuadas para cada uno de los módulos desarrollados en la iteración #2:

Tabla #50: Tarea de ingeniería 22.

Tarea de Ingeniería	
Número Tarea: 22	**Número Historia de Usuario:** HU_6
Nombre Tarea: Creación de la tabla tb_trabajador en la BD.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se crea en la BD la tabla correspondiente a los trabajadores.	

Tabla #51: Tarea de ingeniería 23.

Tarea de Ingeniería	
Número Tarea: 23	**Número Historia de Usuario:** HU_6
Nombre Tarea: Implementar el modificar trabajador.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita modificar un trabajador en la aplicación.	

Tabla #52: Tarea de ingeniería 24.

Tarea de Ingeniería	
Número Tarea: 24	**Número Historia de Usuario:** HU_6
Nombre Tarea: Implementar el adicionar trabajador.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita adicionar un trabajador en la aplicación.	

Tabla #53: Tarea de ingeniería 25.

Tarea de Ingeniería	
Número Tarea: 25	**Número Historia de Usuario:** HU_6
Nombre Tarea: Implementar el eliminar trabajador.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita eliminar un trabajador en la aplicación.	

Tabla #54: Tarea de ingeniería 26.

Tarea de Ingeniería	
Número Tarea: 26	**Número Historia de Usuario:** HU_8
Nombre Tarea: Creación de la tabla tb_local en la BD.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se crea en la BD la tabla correspondiente a los locales.	

Tabla #55: Tarea de ingeniería 27.

Tarea de Ingeniería	
Número Tarea: 27	**Número Historia de Usuario:** HU_7
Nombre Tarea: Implementar el modificar local.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez.	
Descripción: Se implementa la funcionalidad que permita modificar un local en la aplicación.	

Tabla #56: Tarea de ingeniería 28.

Tarea de Ingeniería	
Número Tarea: 28	**Número Historia de Usuario:** HU_8

Nombre Tarea: Implementar el adicionar local.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita adicionar un local en la aplicación.	

Tabla #57: Tarea de ingeniería 29.

	Tarea de Ingeniería
Número Tarea: 29	**Número Historia de Usuario:** HU_8
Nombre Tarea: Implementar el eliminar local.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez**.**	
Descripción: Se implementa la funcionalidad que permita eliminar un local en la aplicación.	

Tabla #58: Tarea de ingeniería 30.

	Tarea de Ingeniería
Número Tarea: 30	**Número Historia de Usuario:** HU_9
Nombre Tarea: Implementar el obtener reportes.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**

Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez
Descripción: Se implementa la funcionalidad obtener reportes (importe) por servicio, por un rango de fecha y por local.

Tabla #59: Tarea de ingeniería 31.

Tarea de Ingeniería	
Número Tarea: 31	**Número Historia de Usuario:** HU_10
Nombre Tarea: Creación de la tabla tb_noticias en la BD.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez	
Descripción: Se crea en la BD la tabla correspondiente a las noticias publicadas en la aplicación.	

Tabla #60: Tarea de ingeniería 32.

Tarea de Ingeniería	
Número Tarea: 32	**Número Historia de Usuario:** HU_10
Nombre Tarea: Implementar el modificar noticia.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez.	

Descripción: Se implementa la funcionalidad que permita modificar una noticia en la aplicación.

Tabla #61: Tarea de ingeniería 33.

Tarea de Ingeniería	
Número Tarea: 33	**Número Historia de Usuario:** HU_10
Nombre Tarea: Implementar el adicionar noticia.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez.	
Descripción: Se implementa la funcionalidad que permita adicionar una noticia en la aplicación.	

Tabla #62: Tarea de ingeniería 34.

Tarea de Ingeniería	
Número Tarea: 34	**Número Historia de Usuario:** HU_10
Nombre Tarea: Implementar el eliminar noticia.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:**
Fecha Inicio:	**Fecha Fin**
Programador Responsable: Reinier Jean Felipe y Oscar Rodríguez.	
Descripción: Se implementa la funcionalidad que permita eliminar una noticia en la	

aplicación.

Diagrama de despliegue

El diagrama de despliegue permite apreciar de forma visual cómo se encuentran relacionados físicamente los componentes de la aplicación. En este caso la aplicación se encuentra hospedada en un servidor web (apache) y maneja los datos persistentes sobre un sistema gestor de base de datos (*PostgreSQL*).

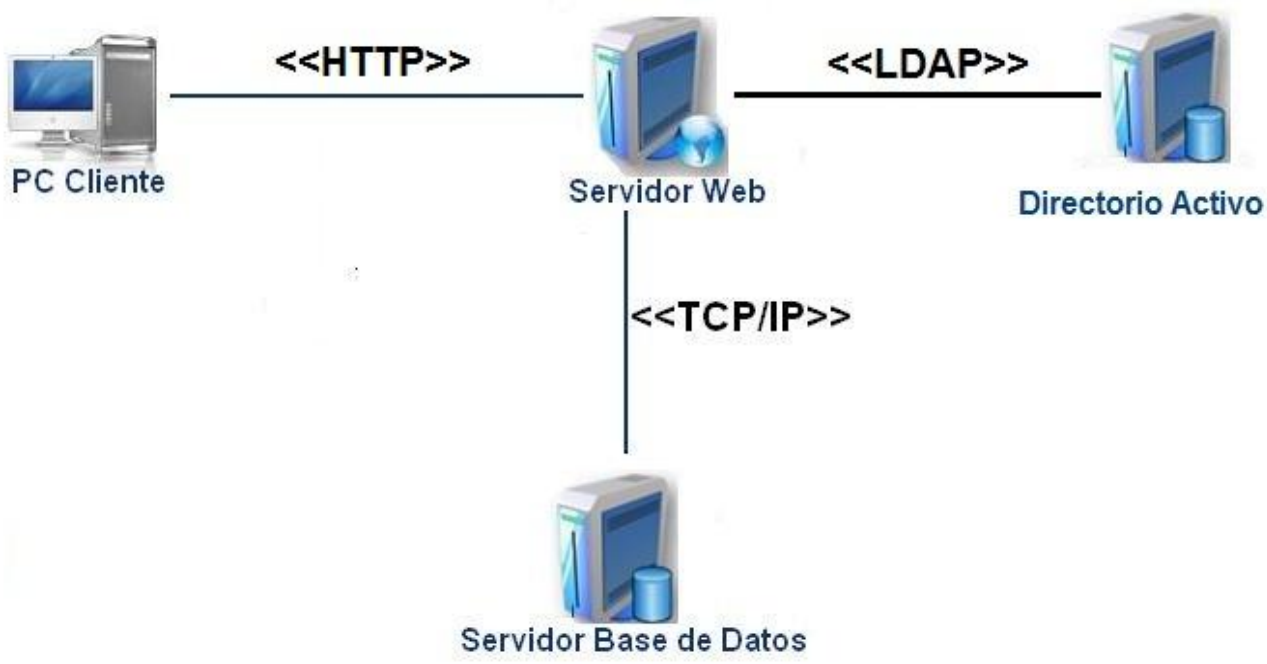

Figura #2 Diagrama de despliegue

Pruebas

Uno de los pilares fundamentales de la metodología XP lo constituye el proceso de pruebas, el cual anima a los desarrolladores a probar constantemente. Mediante está filosofía se minimiza la cantidad de errores no detectados y también el tiempo transcurrido entre la introducción de éste en la

aplicación y su detección. Todo esto contribuye a elevar la calidad del producto y a la seguridad de los desarrolladores a la hora de hacer cambios o modificaciones. La metodología XP divide las pruebas en dos grupos:

- Pruebas unitarias, desarrolladas por los programadores y encargadas de verificar el código de forma automática.
- Pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

Pruebas de aceptación.

Las pruebas de aceptación son pruebas que se crean a partir de las historias de usuario. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una HU (Historia de Usuario) ha sido implementada correctamente. Una HU puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de estas pruebas es garantizar que los requerimientos hayan sido cumplidos y que la aplicación es realmente lo que el cliente quería. Una HU no se considera terminada hasta que no ha pasado sus pruebas de aceptación. [26] (**Ver anexo 4).**

Conclusiones.

En este capítulo se hizo alusión a las fases de implementación y pruebas del proceso de desarrollo de la aplicación exponiendo los artefactos generados en cada una de estas fases y describiendo cada uno de ellos.

Conclusiones generales

En el presente trabajo se propuso la realización de una aplicación para facilitar el proceso de reservación de los servicios de barbería, peluquería y lavandería en la UCI, logrando que el mismo se desarrolle de manera más eficiente. Al terminó del mismo se concluye lo siguiente:

- El sistema desarrollado automatiza en un 85% los procesos de reservación de lavandería, peluquería y barbería en la UCI.
- Se agiliza el proceso de reservación de los servicios de barbería, peluquería y lavandería en la UCI, tanto para los estudiantes que solicitan el servicio como para los trabajadores y profesores de la universidad.
- La utilización del sistema provee un mejor control sobre las reservaciones realizadas en los diferentes servicios prestados.
- El sistema propuesto disminuye la carga de trabajo de los trabajadores de los locales por las siguientes razones:
    1. El trabajo no es realizado a mano y existe ahora la constancia de las ganancias de los diferentes servicios.
    2. El sistema administra la cantidad de solicitudes por local logrando así que no existan locales con más demanda que los demás, compensando la carga de trabajo.
- Facilita el trabajo en la oficina de barbería, peluquería y lavandería en la UCI.

Recomendaciones

Como resultado del proceso de investigación y realización de la aplicación han surgido ideas que serían recomendables tener en cuenta para un futuro perfeccionamiento del sistema, a continuación se listan las mismas:

- A los encargados de realizar las nuevas mejoras al sistema, se recomienda agregar una funcionalidad que permitan realizar reclamaciones por parte de los estudiantes, profesores y trabajadores de la universidad respecto a las reservaciones realizadas.
- Al centro de informatización y desarrollo de la UCI se recomienda dar continuidad al desarrollo y soporte del sistema implementado, además, de optimizar la codificación de la aplicación y actualizar la versión del marco de trabajo empleado.
- Continuar en el estudio de este tipo de aplicaciones con el fin de añadir mejoras funcionales enfocadas al beneficio del usuario.

Bibliografías

1. (s.f.). Recuperado el 15 de Enero de 2010, de Definición de sistema: http://www.mitecnologico.com/Main/DefinicionDeSistema
2. (s.f.). Recuperado el 16 de Enero de 2010, de Comparison of Oracle, MySQL and Postgres DBMS: http://dcdbappl1.cern.ch:8080/dcdb/archive/ttraczyk/db_compare/db_compare.html
3. (s.f.). Recuperado el 16 de Enero de 2010, de ¿Qué es un Sistema Gestor de Bases de Datos o SGBD?: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd
4. (s.f.). Recuperado el 10 de Enero de 2010, de Sympony: http://www.librosweb.es/symfony/
5. (s.f.). Recuperado el 10 de Enero de 2010, de Sympony características: http://www.symfony-project.org/
6. (s.f.). Recuperado el 13 de Enero de 2010, de Frameworks de PHP: http://www.maestrosdelweb.com/editorial/los-frameworks-de-php-agilizan-tu-trabajo/
7. (s.f.). Recuperado el 13 de Enero de 2010, de TOP 10 Framework de PHP: http://www.tufuncion.com/top10_framework
8. (s.f.). Recuperado el 13 de Enero de 2010, de Listado de frameworks de PHP: http://www.webadictos.com.mx/2007/09/13/lista-de-frameworks-mvc-en-php/
9. (s.f.). Recuperado el 18 de Enero de 2010, de Metodología ágiles: www.xprogramming.com.
10. (s.f.). Recuperado el 18 de Enero de 2010, de Metodología ágiles: www.extremeprogramming.org
11. (s.f.). Recuperado el 15 de 1 de 2010, de ¿Por qué los sistemas de gestión son necesarios? : http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/
12. Álvarez, M. (s.f.). Recuperado el 15 de Enero de 2010, de “Breve historia de PHP”: http://www.desarrolloweb.com/manuales/12/

13. anónimo. (s.f.). Recuperado el 15 de Enero de 2010, de Arquitectura cliente/servidor: http://www.ciber-tec.com/ads.htm

14. anónimo. (s.f.). Recuperado el 20 de Enero de 2010, de Neatbeans 6.8: http://netbeans.org/features/

15. García, M. (s.f.). Recuperado el 17 de Enero de 2010, de "Ventajas del HTML": http://www.ugr.es/pages/investigacion/

16. Netpecos. (s.f.). Recuperado el 17 de Enero de 2010, de "PostGreSQL vs. MySQL": http://www.netpecos.org/docs/mysql_postgres/x15.html

17. Oracle. (s.f.). Recuperado el 17 de Enero de 2010, de http://www.oracle.com/index.html

18. Postgre, O. v. (s.f.). Recuperado el 15 de Enero de 2010, de Oracle vs Postgre: http://searchoracle.techtarget.com/news/1179016/PostgreSQL-vs-Oracle-Users-speak-out

19. Saavedra, J. (s.f.). "Lenguajes de Programación". . Recuperado el 15 de Enero de 2010, de http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/

20. Lenguajes del lado servidor y del lado cliente. [En línea] 17 de enero de 2007. http://eats.wordpress.com/2007/01/17/lenguajes-del-lado-servidor-y-del-lado-cliente/.

21. Álvarez, Miguel Ángel. DesarrolloWeb.com. [En línea] 2010. http://www.desarrolloweb.com/articulos/introduccion-jquery.html.

22. Corredor, Germán. Sun libera su IDE NetBeans para Java y PHP. [En línea] julio de 2009. http://osum.sun.com/profiles/blogs/sun-libera-su-ide-netbeans-67-1..

23. Zaballa Coca, Roilán. Personalización de distribuciones basadas en la familia SUSE Linux.". 2009. Sanchez, María A. Mendoza. Metodologías De Desarrollo De Software. [En línea] http://www.willydev.net/InsiteCreation/v1.0/descargas/cualmetodologia.pdf.

24. XP Programación Extrema. Club Developers. [En línea] [Citado el: 7 de febrero de 2010.] http://www.clubdevelopers.com/index.php?p=38..

25. El único instrumento adecuado para determinar el status de la calidad. s.l.: Asociación de Técnicos, 2008.B., Ing. Alexander Oré. Pruebas Unitarias CAP 1-Sotware Testing and QA-

Pruebas Unitarias. [En línea] 2009. http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

26. La prueba de aceptación es la prueba más importante para los productos software. [En línea] 2010. http://pruebasdesoftware.com/pruebadeaceptacion.htm.
27. Symfony.es. [En línea] http://www.symfony.es/..
28. ¿Qué son las bases de datos? [En línea] 26 de octubre de 2007. [Citado el: 20 de marzo de 2010.] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos..

Glosario de términos.

Sistema[1]**:** Un sistema es una reunión o conjunto de elementos relacionados. Puede estructurarse de conceptos, objetos y sujetos. Los sistemas se componen de otros sistemas a los que llamamos subsistemas. En la mayoría de los casos, podemos pensar en sistemas más grandes o súper ordinales, los cuales comprenden otros sistemas que llamamos sistema total y sistema integral. [1]

Sistema de gestión de Reservaciones: Aplicación informática que almacena toda la información relacionada con las solicitudes de reserva de servicios realizadas por los clientes de una organización o entidad.

Gestión de Información: Acciones que realizan la aplicación y/o el usuario para mostrar y manipular información referente a las modalidades de reservación; incluye el envío de correos, generación de historiales y manipulación de todo tipo de mensajes que se le muestran al usuario.

Reportes: Listados con reservaciones y distribuciones que se generan a partir de una solicitud. Están organizados por distintos criterios como hora de la reservación, tipo de prestaciones, etcétera.

Servicio Web: Es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programaciones diferentes y ejecutadas sobre cualquier plataforma pueden utilizar los servicios web para intercambiar datos en redes.

[1] **Sistema: Del latín systema, un sistema es módulo ordenado de elementos que se encuentran interrelacionados y que interactúan entre sí. El concepto se utiliza tanto para definir a un conjunto de conceptos como a objetos reales dotados de organización.**