1 INTRODUCCIÓN

1.1 ANTECEDENTES

En la ciudad de El Ángel, Cantón Espejo, Provincia del Carchi, se constituyó el 15 de febrero del 2005, la Empresa Municipal de Agua Potable y Saneamiento Ambiental del Cantón Espejo, como persona jurídica de derecho público con autonomía administrativa, operativa, financiera y patrimonial, la misma que se regirá por las normas de la Ley Orgánica de Régimen Municipal, la presente Ordenanza que regula la prestación de los servicios de Agua Potable y Saneamiento Ambiental, las disposiciones de los reglamentos internos generales y específicos que se expidan y demás normas jurídicas aplicables.

La empresa se constituye por la presente ordenanza y se denominará Empresa Municipal de Agua Potable y Saneamiento Ambiental del Cantón Espejo, cuyas siglas son EMAPSA-E, y por ello, con este mismo nombre se identificará en todos los actos públicos, privados, judiciales, extrajudiciales y administrativos.

La EMAPSA-E ejercerá su acción en el cantón Espejo, provincia del Carchi, teniendo competencia para todo lo relacionado con la prestación de los servicios de Agua Potable, Alcantarillado y Saneamiento Ambiental, dentro del plan cantonal de desarrollo. Estos servicios que se inician en la ciudad de El Ángel, podrán extenderse a las parroquias rurales del cantón y otras jurisdicciones del régimen autónomo y entidades públicas o privadas, dedicadas a la prestación de ellos. Todo esto a través de convenios legalmente suscritos.

La Empresa tiene como objetivo la prestación de los servicios de Agua Potable y Saneamiento Ambiental, tendiente a preservar la salud de los habitantes y obtener una rentabilidad social y económica en sus inversiones.

1.1.1 VISIÓN Y MISIÓN DE LA EMAPSA-E

1.1.1.1 VISIÓN DE EMAPSA-E

La Empresa Municipal de Agua Potable y Saneamiento Ambiental del Cantón Espejo, prevee en cinco años, bajo el concepto de calidad total convertirse en una Empresa Líder de la Región Norte del País.

1.1.1.2 MISIÓN DE EMAPSA-E.

Contribuir al mejoramiento de la calidad de vida de la población del Cantón Espejo, a través de la prestación eficiente del Servicio de Agua Potable.

1.2 PROBLEMA

La Empresa Municipal de Agua Potable y Saneamiento Ambiental (EMAPSA-E), tiene como finalidad la captación, tratamiento, distribución, producción y venta de agua potable y la prestación de los servicios de alcantarillado a la ciudad El Ángel y las parroquias rurales (27 de Septiembre, La Libertad, San Isidro, El Goaltal) del cantón Espejo, garantizando eficiencia y eficacia, con criterio de equidad y justicia, comprometida con una concepción ecológica que preserve las cuencas hidrográficas y proteja el medio ambiente de todo el Cantón.

EMAPSA-E es una empresa, que brinda servicios de agua potable y alcantarillado a la población del cantón Espejo, proyectándose siempre a la satisfacción del cliente.
También tiene como finalidad dar el debido mantenimiento al sistema regional así como disponer de un sistema de agua potable que garantice cantidad y calidad las veinte y cuatro horas de servicio.

La Empresa Municipal de Agua Potable y Saneamiento Ambiental cuenta con el Departamento de Proyectos, Departamento de Medio Ambiente y Laboratorio; estos departamentos son los encargados de desarrollar proyectos para la implementación de nuevas redes de agua potable como de alcantarillado, juntamente con los otros departamentos se encargan de desarrollar plantas de tratamiento de aguas residuales así como del manejo de todos los residuos sólidos a la vez el Laboratorista es el encargado de realizar un monitoreo diario del agua para realizar el análisis correspondiente para saber cuál es el estado del agua.

Por ser una empresa con apenas cinco años, esta no cuenta con todos los sistemas informáticos necesarios que ayuden a optimizar sus procesos y cumplir con las normas requeridas por las entidades de control gubernamentales.

La Empresa Municipal de Agua Potable y Saneamiento Ambiental en la actualidad el control vehicular lo realiza de una manera informal y no acorde a las exigencias de las entidades de control gubernamentales; y lleva los procesos de control vehicular de forma manual.

El deficiente manejo de los inventarios de vehículos sus accesorios y herramientas provoca la falta de información en tiempo real, difícil acceso a registros históricos y menor control de existencias posibilitando la generación de artículos obsoletos.

Las órdenes de movilización, partes de novedades y accidentes se realizan de forma manual y en hojas pre impresas, lo cual conlleva a un registro desordenado e inexacto de la información.

La emisión de las órdenes de provisión de combustibles y lubricantes se lo realiza de forma manual y en hojas pre impresas dificultando llevar un control, lo cual puede provocar un desmesurado o escaso pedido de suministros de los mismos.

El inadecuado manejo de la administración de los vehículos de la institución, dificulta el registro de entrada / salida de los mismos; y su entrega – recepción a los conductores.

Las consecuencias de no disponer a futuro de una solución informática para el control vehicular es que al seguir llevando un manejo manual esto podría causar pérdida, inconsistencia y confusión al momento de requerir la información.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Analizar, Diseñar e Implementar un Sistema de Control Vehicular para la Empresa Municipal de Agua Potable y Saneamiento Ambiental del Cantón Espejo utilizando herramientas de software libre.

1.3.2 OBJETIVOS ESPECÍFICOS

Recolectar los requerimientos funcionales y no funcionales.

Realizar la planificación del desarrollo del proyecto.

Definir el modelo del negocio, modelo de objetos del negocio y modelo del dominio.

Implementar la solución desarrollada en la Empresa Municipal de Agua Potable y Saneamiento Ambiental del Cantón Espejo.

Controlar mediante código de barras la emisión de solicitudes.

Realizar las pruebas y estabilización de la aplicación informática.

Realizar la documentación necesaria del sistema.

1.4 JUSTIFICACIÓN

Los vehículos pertenecientes al sector público, y a las entidades de derecho privado que disponen de recursos públicos, están destinados exclusivamente para uso oficial, es decir para el desempeño de funciones públicas, en los días y horas laborables, y no podrán ser utilizados en fines personales, ni familiares, ajenas al sector público, ni en actividades electorales y políticas.

Para la movilización de los vehículos oficiales, fuera de la sede donde los funcionarios ejercen habitualmente sus funciones, las Órdenes de Movilización serán emitidas por la máxima autoridad o el servidor delegado para el efecto que podrá ser el Jefe de Transportes y tendrán una vigencia no mayor de 5 días hábiles.

Al contar con un sistema informático de control vehicular se podrá conocer la disponibilidad de un vehículo en una determinada fecha, disponer de la cantidad adecuada de lubricantes y combustibles en los vehículos, mantenimientos oportunos, mejorando la vida útil de los vehículos con un gran beneficio para la empresa. Además, al tener un mantenimiento adecuado de los vehículos se reduce de manera significativa las emisiones de contaminantes atmosféricos de origen vehicular.

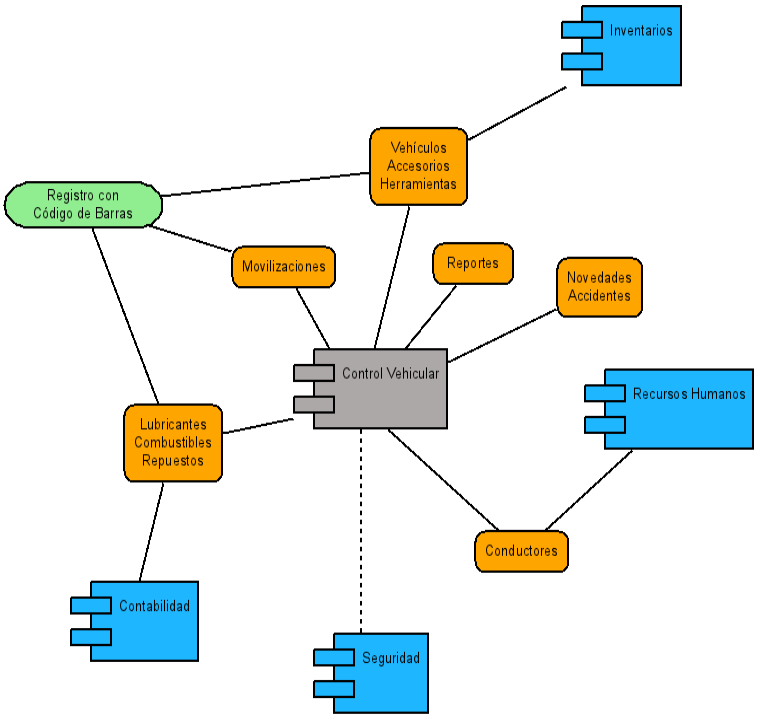
Por lo antes expuesto se hace imperiosa la necesidad de contar con una solución tecnológica para la realización de dicho control.

1.5 ALCANCE

Para el desarrollo del sistema de control vehicular se utilizarán las siguientes tecnologías:

Servidor Web: Apache HTTP Server 2.2
Lenguaje de Programación: PHP 5.3
Sistema de Gestión de Base de Datos Relacional: PostgreSQL 9.1
Framework: Symfony PHP Web 1.4
Entorno de Desarrollo: NetBeans 7.0

1.5.1 ARQUITECTURA MODULAR



ROLES DE USUARIO

Administrador: Encargado del control de los servidores, y del módulo de seguridad.

Gerente: Acceso a todos reportes.

Secretaria: Ingreso de vehículos, accesorios y herramientas; y emisión de las órdenes de movilización.

Contadora: Emisión de órdenes de consumo de lubricantes, combustibles y repuestos.

Recursos Humanos: Ingreso de Conductores.

1.5.2 ARQUITECTURA TECNOLÓGICA

La arquitectura que se utilizará es Modelo Vista Controlador (MVC); en el lado del servidor se utilizará Linux y se tendrá tanto el servidor de base de datos como el de aplicaciones, y contará con toda la infraestructura necesaria para su correcto funcionamiento; por su parte del lado del cliente se necesitará únicamente que estos cuenten con un Navegador Web compatible.

Limitaciones:

La aplicación informática será implementada en la Empresa Municipal de Agua Potable y Saneamiento Ambiental del Cantón Espejo.

Para el desarrollo de la aplicación se utilizarán las herramientas y tecnologías que los encargados de la parte informática de la institución lo dispongan.

La aplicación será parte del Sistema de Planificación de Recursos Empresariales (Sistema Integrado) que se encuentra en la etapa de desarrollo en la institución.

BENEFICIOS DE LA IMPLEMENTACIÓN DEL SISTEMA

Al contar con un sistema informático de control vehicular se podrá conocer la disponibilidad de un vehículo en una determinada fecha, disponer de la cantidad adecuada de lubricantes y combustibles en los vehículos, mantenimientos oportunos, mejorando la vida útil de los vehículos con un gran beneficio para la empresa.

Además, al tener un mantenimiento adecuado de los vehículos se reduce de manera significativa las emisiones de contaminantes atmosféricos de origen vehicular.

2 MARCO TEÓRICO

2.1 SERVIDOR WEB: APACHE HTTP SERVER

Apache HTTP2 Server es uno de los más robustos y rápidos servidores web multiplataforma que existen, ha sido creado desde la idea de open-source demostrando una vez más que ir asociado a esta idea no es signo de fracaso y lo avala el ser el servidor web más utilizado en todo el mundo (desde pequeñas y grandes empresas, pasando por instituciones y universidades).

Con Apache HTTP Server se puede ejecutar CGI, Perl, PHP más Bases de datos, SSL, soporte para host virtuales, soporte IPv6, en fin, casi todo lo que le pidamos a cualquier servidor web.

Apache es un servidor web de software libre desarrollado por la Apache Software Foundation, cuyo objetivo es servir o suministrar páginas web (en general, hipertextos) a los clientes web o navegadores que las solicitan.

La arquitectura utilizada es cliente/servidor, es decir, el equipo cliente hace una solicitud o petición al equipo servidor y éste la atiende.

En el equipo cliente se ejecuta una aplicación llamada “navegador o cliente web” que:

Sirve de interfaz con el usuario: atiende sus peticiones, muestra los resultados de las consultas y proporciona al usuario un conjunto de herramientas que facilitan su comunicación con el servidor.

Se comunica con el servidor web: transmite las peticiones de los usuarios.

El protocolo utilizado para la transferencia de hipertexto es HTTP (HiperText Transfer Protocol) que está basado en el envío de mensajes y establece el conjunto de normas mediante las cuales se envían las peticiones de acceso a una web y la respuesta de la misma.

El servidor web Apache proporciona contenidos al cliente web o navegador como:

1. Páginas estáticas: es el uso más generalizado que se hace de un servidor web. De esta forma se transfieren archivos HTML, imágenes, etc y no se requiere un servidor muy potente en lo que al hardware y software se refiere.

2. Páginas dinámicas: la información que muestran las páginas que sirve Apache cambia ya que se obtiene a partir de consultas a bases de datos u otras fuentes de datos. Son, por tanto, páginas con contenido dinámico, cambiante.

Su curioso nombre hace referencia a sus orígenes. Cuando el proyecto inicial (Centro Nacional de Actividades de Supercomputación, NCSA, Universidad de Illinois) fue abandonado por su principal desarrollador, Rob McCool, diferentes webmasters comenzaron a desarrollar “parches” para el código fuente de este servidor inicial y mediante el correo electrónico sincronizaban sus aportaciones.

De esta forma apareció el proyecto Apache, cuyo nombre se debe a: A PAtCHy server (un servidor "parcheado").
La primera versión de Apache es la 0.6 en 1995, y en la actualidad la versión disponible es la 2.2.

Y su nombre de Apache se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet.

Apache HTTP Server es un esfuerzo para desarrollar y mantener un servidor HTTP de código abierto para sistemas operativos como UNIX9 y Windows NT10.

El objetivo principal de Apache HTTP Server es el de proporcionar un servidor eficaz, seguro y extensible con servicios HTTP en sincronización con los estándares de HTTP. Apache ha sido el servidor web más popular en Internet desde 1996.

Apache HTTP Server es un esfuerzo de desarrollo de software colaborativo encaminado a crear una implementación de código abierto, sólida, de calidad comercial, completa y disponible libremente de un servidor HTTP (web).

2.1.1 ALGUNAS DE LAS CARACTERÍSTICAS DE APACHE HTTP SERVER

Es un servidor web potente, flexible y compatible.
Es muy configurable y extensible con módulos de terceros.
Puede personalizarse escribiendo "módulos" utilizando el módulo Apache API.
Proporciona código fuente completo y viene con una licencia ilimitada.
Es multiplataforma es decir, se puede ejecutar en cualquier sistema operativo.
Cuenta con una gran comunidad.

2.1.2 MOD JK

mod_jk es un conector que permite al contenedor de Java Server Pages (JSP) Jakarta Tomcat interactuar con servidores web como Apache, Netscape, iPlanet, SunOne e incluso IIS usando el protocolo AJP.

La principal funcionalidad de este módulo es permitir a servidores de aplicaciones o al servidor Jakarta Tomcat enlazarse con un servidor web. Este servidor web, típicamente el servidor HTTP Apache, introduce una mayor gestión en las conexiones de los clientes y mayor la seguridad en las transacciones del sistema. Así mismo se puede enlazar varias instancias al servidor web permitiendo así una mayor tolerancia a errores y aligerar la carga en los servidores Java.

2.2 SYMFONY PHP WEB FRAMEWORK

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente en PHP 5.3. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

2.2.1 CARACTERÍSTICAS

Symfony fue diseñado para ajustarse a los siguientes requisitos:

Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).

Independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de Propel, permiten cambiar con facilidad de SGBDR en cualquier fase del proyecto.

Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5.

Sencillo de usar en la mayoría de casos, aunque es preferible para el desarrollo de grandes aplicaciones Web que para pequeños proyectos.

Symfony emplea el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas partes que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos.
La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista.

2.2.2 AUTOMATIZACIÓN DE CARACTERÍSTICAS DE PROYECTOS WEB

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como:

La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.

La capa de presentación utiliza plantillas y layouts que pueden ser creados por diseñadores HTML sin ningún tipo de conocimiento del framework. Los helpers incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.

Los formularios incluyen validación automatizada y relleno automático de datos ("repopulation"), lo que asegura la obtención de datos correctos y mejora la experiencia de usuario.

Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.

La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.

El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.

El soporte de correo electrónico incluido y la gestión de APIs permiten a las aplicaciones web interactuar más allá de los navegadores.

Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación de datos.
Los plugins, las factorías (patrón de diseño "Factory") y los "mixin" permiten realizar extensiones a medida de Symfony.

Las interacciones con Ajax son muy fáciles de implementar mediante los helpers que permiten encapsular los efectos JavaScript compatibles con todos los navegadores en una única línea de código.

2.2.3 ENTORNO DE DESARROLLO Y HERRAMIENTAS

Symfony puede ser completamente personalizado para cumplir con los requisitos de las empresas que disponen de sus propias políticas y reglas para la gestión de proyectos y la programación de aplicaciones. Por defecto incorpora varios entornos de desarrollo diferentes e incluye varias herramientas que permiten automatizar las tareas más comunes de la ingeniería del software:

Las herramientas que generan automáticamente código han sido diseñadas para hacer prototipos de aplicaciones y para crear fácilmente la parte de gestión de las aplicaciones.

El framework de desarrollo de pruebas unitarias y funcionales proporciona las herramientas ideales para el desarrollo basado en pruebas ("test-driven development").

La barra de depuración web simplifica la depuración de las aplicaciones, ya que muestra toda la información que los programadores necesitan sobre la página en la que están trabajando.

La interfaz de línea de comandos automatiza la instalación de las aplicaciones entre servidores.

Es posible realizar cambios "en caliente" de la configuración (sin necesidad de reiniciar el servidor).

El completo sistema de log permite a los administradores acceder hasta el último detalle de las actividades que realiza la aplicación.

2.2.4 ¿POR QUÉ LO LLAMARON "SYMFONY" ?

Porque Fabien quería una nombre corto que tuviera una letra 's' (de Sensio) y una letra 'f' (de framework), que fuera fácil de recordar y que no estuviera asociado a otra herramienta de desarrollo.
Además, no le gustan las mayúsculas. "Symfony" era muy parecido a lo que estaba buscando, aunque no es una palabra correcta en el idioma inglés (la palabra correcta es "symphony"), y además estaba libre como nombre de proyecto. La otra alternativa era "baguette".

2.3 SISTEMA DE GESTIÓN DE BASE DE DATOS RELACIONAL: POSTGRESQL

PostgreSQL es un poderoso sistema de base de datos relacional de código abierto con el desarrollo activo y una arquitectura probada se ha ganado una sólida reputación de fiabilidad, integridad de

datos y corrección.

Funciona en todos los principales sistemas operativos, entre ellos: Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows. Es totalmente compatible con ACID, tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios lenguajes). Incluye la mayoría de tipos de datos de SQL92 y SQL99 entre ellos: INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL y TIMESTAMP. También soporta almacenamiento de objetos binarios grandes como imágenes, sonidos y video. Dispone de interfaces nativas de programación de C/C++, Java, .NET, Perl, Python, Ruby, Tcl, ODBC, entre otros; y una documentación de carácter excepcional.

2.3.1 CARACTERÍSTICAS

Algunas de sus principales características son, entre otras:

ALTA CONCURRENCIA

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

AMPLIA VARIEDAD DE TIPOS NATIVOS

PostgreSQL provee nativamente soporte para:

Números de precisión arbitraria.
Texto de largo ilimitado.
Figuras geométricas (con una variedad de funciones asociadas).
Direcciones IP (IPv4 e IPv6).
Bloques de direcciones estilo CIDR.
Direcciones MAC.

Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

Otras características

Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).

Disparadores (triggers): Un disparador o trigger se define como una acción específica que se realiza de acuerdo a un evento, cuando éste ocurra dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

Los disparadores se definen por seis características:

El nombre del disparador o trigger

El momento en que el disparador debe arrancar

El evento del disparador deberá activarse sobre

La tabla donde el disparador sé activará

La frecuencia de la ejecución

La función que podría ser llamada

Entonces combinando estas seis características, PostgreSQL le permitirá crear una amplia funcionalidad a través de su sistema de activación de disparadores (triggers).

Vistas.

Integridad transaccional.

Herencia de tablas.

Tipos de datos y operaciones geométricas.

Soporte para transacciones distribuidas.

Permite a PostgreSQL integrase en un sistema distribuido formado por varios recursos (p.ej, una base de datos PostgreSQL, otra Oracle, gestionado por un servidor de aplicaciones donde el éxito ("commit") de la transacción globlal es el resultado del éxito de las transacciones locales.

2.4 LENGUAJE DE PROGRAMACIÓN: PHP

PHP es un lenguaje de programación interpretado (Lenguaje de alto rendimiento), diseñado originalmente para la creación de páginas web dinámicas. Se usa principalmente para la interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica.
PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf40 en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. El lenguaje PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, el número de sitios en PHP ha compartido algo de su preponderante dominio con otros nuevos lenguajes no tan poderosos desde agosto de 2005.

El sitio web de Wikipedia está desarrollado en PHP.

Permite la conexión a diferentes tipos de servidores de bases de datos.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

2.4.1 CARACTERÍSTICAS DE PHP

Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.

El código fuente escrito en PHP es invisible al navegador web y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.

Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.

Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).

Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.

Es Software Libre, por lo que se presenta como una alternativa de fácil acceso para todos.

Permite aplicar técnicas de programación orientada a objetos.

Biblioteca nativa de funciones sumamente

amplia e incluida.

No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.

PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable.

Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.
Como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente.

El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. Es utilizado en aplicaciones Web relacionadas por algunas de las organizaciones más prominentes tales como Mitsubishi, Redhat, Ericsson y NASA.

La forma de usar php es insertando código php dentro del código html de un sitio web. Cuando un cliente (cualquier persona en la web) visita la página web que contiene éste código, el servidor lo ejecuta y el cliente sólo recibe el resultado. Su ejecución, es por tanto en el servidor, a diferencia de otros lenguajes de programación que se ejecutan en el navegador.

Generalmente los scripts en PHP se embeben en otros códigos como HTML, ampliando las posibilidades del diseñador de páginas web enormemente.

La interpretación y ejecución de los scripts PHP se hacen en el servidor, el cliente (un navegador que pide una página web) sólo recibe el resultado de la ejecución y jamás ve el código PHP.

2.5 PATRÓN DE ARQUITECTURA DE SOFTWARE: MODELO VISTA CONTROLADOR

La arquitectura MVC (Model/View/Controller) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk.

Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

Este modelo de arquitectura se puede emplear en sistemas de representación gráfica de datos o en sistemas CAD, en donde se presentan partes del diseño con diferente escala de aumento, en ventanas separadas.

Este modelo de arquitectura presenta varias ventajas:

Hay una clara separación entre los componentes de un programa; lo cual permite implementarlos por separado.

Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.

La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

2.5.1 DEFINICIÓN DE LAS PARTES

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

2.5.2 ELEMENTOS DEL PATRÓN

Modelo: datos y reglas de negocio.

Vista: muestra la información del modelo al usuario.

Controlador: gestiona las entradas del usuario.

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información de una base de datos, que se puede presentar de diversas formas: diagrama de tarta, de barras, tabular, etc. A continuación se detalla cada componente:

1. El modelo es el responsable de:

Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".

Lleva un registro de las vistas y controladores del sistema.

Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc).

2. El controlador es responsable de:

Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).

Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega(nueva_ord en_de_venta)".

3. Las vistas son responsables de:

Recibir datos del modelo e interactuar con el usuario.

Tienen un registro de su controlador asociado (normalmente porque además lo instancia).

Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Un ejemplo de MVC con un modelo pasivo (aquel que no notifica cambios en los datos) es la navegación web, que responde a las entradas del usuario, pero no detecta los cambios en datos del servidor.

El diagrama de secuencia:

Pasos:

1. El usuario introduce el evento.

2. El Controlador recibe el evento y lo traduce en una petición al Modelo (aunque también puede llamar directamente a la vista).

3. El modelo (si es necesario) llama a la vista para su actualización.

4. Para cumplir con la actualización la Vista puede solicitar datos al Modelo.

5. El Controlador recibe el control.

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), 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 modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

2.5.2.1 ¿QUÉ VENTAJAS TRAE UTILIZAR EL MVC?

1. Es posible tener diferentes vistas para un mismo modelo (ejemplo: representación de un conjunto de datos como una tabla o como un diagrama de barras).

2. Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.

3. Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).

2.5.3 FLUJO QUE SIGUE EL CONTROL EN UNA IMPLEMENTACIÓN GENERAL DE UN MVC

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

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).

2. 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.

3. 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.

4. 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.

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

3 CONCLUSIONES

1. Los objetivos planteados en mi trabajo de grado fueron comprobados ya que se logró un software de altísima calidad, que cumplió con los objetivos planteados.

2. El contar con este aplicativo realizado en aplicación web nos permite acceder a la información extrayendo reportes en línea de manera rápida, atendiendo a solicitudes realizadas por la Institución.

3. Implementar el software permitió contar con un sistema de calidad para el desempeño y productividad de sus empleados, y simplificó el control de acceso a la información del control vehicular (Órdenes de Movilización, Control de Mantenimientos, Órdenes de Lubricantes y Combustibles, Registro de Conductores, etc), contando con información organizada, centralizada y de fácil acceso.

4. Este proyecto permitió un cambio organizacional previo y durante la implementación, atendiendo los requerimientos en los servicios de secretaria, contabilidad, recursos humanos y gerencia.

5. Con la Implementación del registro con código de barras se pudo realizar de manera más ágil las búsquedas de las Órdenes de Movilización y de los Mantenimientos de los vehículos.

4 REFERENCIAS BIBLIOGRÁFICAS

The Apache Software Foundation, 2012: The Apache Software Foundation, Apache HTTP Server, 2012, http://httpd.apache.org

Código abierto, 2012: Wikipedia, Código abierto, 2012, http://es.wikipedia.org/wiki/C%C3%B3digo_abierto

HTTP, 2012: Wikipedia, Hypertext Transfer Protocol, 2012, http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Softpedia, 2012: Softpedia, Descripción de Apache HTTP Server, 2012, http://www.softpedia.es/programa-Apache-HTTP-Server-for-Windows-12027.html

Apache, 2012: The Apache Software Foundation, The Apache Tomcat Connector, 2012, http://tomcat.apache.org/connectors-doc/

SensioLabs, 2012: SensioLabs, Open-Source PHP Web Framework, 2012, www.symfony-project.org/

SGBD: Wikipedia, Sistema de Gestión de Base de Datos, 2012, http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos

phpDoc, 2012: phpDoc, phpDocumentor, 2012, http://www.phpdoc.org/

PGDG, 2012: PostgreSQL Global Development Group, PostgreSQL, 2012, http://www.postgresql.org/

PGDG, 2011: The PostgreSQL Global Development Group, PostgreSQL 9.0 Official Documentation Volumen I The SQL Langu, 2011

PHP Group, 2012: The PHP Group, PHP: Hypertext Preprocessor, 2012, http://php.net/

WL, 2012: WikiLibros, Implementación Modelo Vista Controlador (MVC), 2012, http://es.wikibooks.org/wiki/Aplicaciones_Distribuidas:_Un_enfoque_pr%C3%A1ctico#Implementaci.C3.B3n_Modelo_Vista_Controlador_.28MVC.29