Universidad de las Ciencias Informáticas

Facultad 1

Título: Sistema de información para la toma de decisiones del cierre de operaciones postales y telegráficas.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores:

Oliver Sosa Cano

Roldán Real Gómez

Tutor:

Msc. Mariano Flores López

Junio de 2008

DECLARACIÓN DE AUTORÍA

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

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

_____ _____

Oliver Sosa Cano Roldán Real Gómez

_____

Tutor Msc. Mariano Flores López

DATOS DE CONTACTO

Síntesis del tutor: Máster Mariano Flores López

Director de Informatización de la Universidad de las Ciencias Informáticas durante 1 año, dirigió el proceso inicial de los proyectos de desarrollo productivo de la UCI

Especialista Principal en Calidad y Desarrollo de la Empresa de Desarrollo de Software para el sector de la Salud, denominada SOFTEL durante 1 año.

Director de Calidad de La Empresa Nacional de Software, DESOFT, durante 8 meses.

Director del Centro de Desarrollo Integrado de Proyectos Informáticos de la Empresa Correos de Cuba, cargo que ocupa actualmente y que está desempeñado desde hace dos años.

DEDICATORIA

A mi familia, toda.

r. r. g.

A mi madre

Cuando te ríes
Las plantas se hinchan de verde
Y mis ojos te acompañan con dos
lagrimones.

A mi hermano

Ser tu ejemplo no es mi meta
Todo lo contrario
Que tú seas el mío.

o. s. c.

AGRADECIMIENTOS

A mis padres, por comprender mi lejanía.

A mis amigos, por serlo.

A Mayver, por su amistad y sus comidas.

Al retozón de Dayron por reírse de mis chistes.

A Carlos de la Rosa, Carlos Verdecia, Armando, Richar, Pedro, Enrique, Yoan. Los socios de la universidad.

Y a mi compañero de tesis Oliver.

Roldán Real Gómez.

A mi madre por mi existencia.

A mi hermano por estar.

A Roldán por compartir conmigo todo esto.

A Mayver por demostrarme que sólo a veces la idiotez es proporcional a cuanta curva guarde la armazón.

A Dayron por la risa compartida.

A Enrique por la amistad incondicional de todos estos años.

A Frank Abel por el aliento lejano.

A Frank Luis por la sabiduría.

A Yasef por los consejos y la comprensión.

A Anisbert por la temprana alegría.

A todos mis amigos de la universidad,
que no son muchos pero son los mejores.

Oliver Sosa Cano.

RESUMEN

La transición que estamos viviendo actualmente hacia una sociedad del conocimiento ha modificado profundamente las relaciones entre las personas, empresas y gobiernos: las empresas usan la Red para relacionarse con clientes y proveedores, utilizan también herramientas de gestión del conocimiento para ser más eficientes, los gobiernos mejoran su presencia en Internet y los servicios a los ciudadanos a través de la Red, los usuarios usan las herramientas para mejorar sus relaciones personales. Vamos hacia una sociedad altamente interconectada donde el eje fundamental es la información. El software es cada vez más el gran intermediario entre la información y la inteligencia humana.

Actualmente, en la Empresa de Correos de Cuba (ECC) se están utilizando varios sistemas que tributan información útil para integrar en un único sistema de información, los cuales trabajan aislados entre sí, además ninguno de ellos presenta todas las potencialidades de un sistema de información, que además de sus funciones básicas, debe incluir otras que le proporcionen una mayor eficiencia en el proceso de toma de decisiones a los directivos. Por esta razón, surge en la empresa la necesidad de implementar un sistema de información, que ilustre mediante gráficos el trabajo con grandes volúmenes de información para brindar a los usuarios diferentes soluciones.

La ECC, ha decidido encaminar este proyecto denominando “Sistema de Información para la toma de decisiones del cierre de operaciones postales y tecnológicas”, para aumentar su desarrollo en este sentido y brindar un servicio de excelencia a todos sus trabajadores. El objetivo concreto del trabajo es implementar un sistema de información.

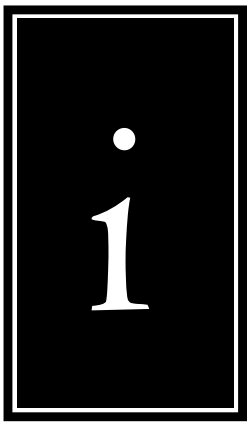
En este documento se realiza una investigación científica detallada del tema en cuestión, se plasman los resultados del estudio realizado en la ECC para la construcción de la aplicación, se explican los conceptos relacionados con la misma, se hace un análisis del sistema, se brinda una explicación profunda de la solución e implementación del producto y se dejan algunas recomendaciones para el mejoramiento futuro del mismo.

Palabras claves: sistema de información, ECC.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS



— ¿Qué sabes de este asunto?— preguntó el Rey a Alicia.

—Nada— dijo Alicia.

— ¿Absolutamente nada?— insistió el Rey.

—Absolutamente nada— dijo Alicia.

—Esto es importante— dijo el Rey, volviéndose hacia los jurados.

LEWIS CARROLL, Alicia en el país de las maravillas.

INTRODUCCIÓN

En las últimas décadas, la sociedad ha experimentado un desarrollo tecnológico acelerado. Con el surgimiento de las microcomputadoras es posible el almacenamiento de gran cantidad de información sin la utilización de papel y con poco trabajo manual humano; el desafío que plantea el momento actual es trasformar las diferentes entidades en un centro digitalizado.

En la era post-industrial, la era de la información, el enfoque de las compañías ha cambiado de la orientación hacia el producto a la orientación hacia el conocimiento, en este sentido el mercado compite hoy en día en términos del proceso y la innovación, en lugar del producto. El énfasis ha cambiado de la calidad y cantidad de producción hacia el proceso de producción en sí mismo, y los servicios que acompañan este proceso.

Con el empleo de las nuevas tecnologías en las empresas cubanas, sin duda se pretende que ayude en la toma de decisiones, en la seguridad e integridad de la información, que se disminuyan los gastos o costos, aumente la productividad, las respuestas sean rápidas y eficientes, las acciones aceptadas, la población y el personal de las empresas tengan un alto desarrollo de su capacidad y conocimiento.

La Empresa de Correos de Cuba (ECC) perteneciente al Ministerio de la Informática y las Comunicaciones (MIC), se encuentra inmersa en un proceso de automatización de todos sus servicios en aras de incrementar la eficiencia de los mismos y elevar el grado de satisfacción de sus clientes.

El Cierre de Operaciones es un proceso vital dentro de la organización y juega un papel fundamental en el control de la calidad de los servicios que se brindan. Mediante este proceso la organización conoce no sólo el nivel de utilización de los servicios por parte de los clientes: es posible conocer el estado de los diferentes indicadores de calidad y actuar en consecuencia con ello.

Este proceso está compuesto por varios subprocesos entre los que se encuentran el proceso de **Cierre de Operaciones Postales** y el **Cierre de Operaciones Tecnológicas**. A su vez, ambos subprocesos están compuestos por varias actividades como por ejemplo gestión de la información del Centro de Clasificación Nacional, Giros, Telegrafía, Dirección de Redes Telemáticas, Dirección de Comercio Electrónico, entre otros.

Las diferentes direcciones de las áreas postal y de tecnologías son las responsables del manejo de todo tipo de indicadores, así como del conjunto de reportes y resúmenes para poder tener al alcance las estadísticas útiles en la toma de decisiones de los directivos que allí laboran; cuyo monitoreo es de vital importancia para el buen funcionamiento de la entidad. En ocasiones, estos indicadores son controlados desde los distintos territorios, quienes se encargan de enviar los reportes a su dirección correspondiente. En la empresa no se cuenta con un sistema que controle todo este flujo de información y facilite el proceso de toma de decisiones.

¿Qué está ocurriendo realmente? En la **actualidad**, en las áreas postal y de tecnologías de las 28 entidades de la ECC que intercambian información con ellas, se están utilizando varios sistemas que tributan información útil para integrar en un único sistema de información, los cuales trabajan aislados entre sí, además ninguno de ellos presenta todas las potencialidades de un sistema de esta índole, que a parte de sus funciones básicas, debe incluir otras que le proporcionen una mayor eficiencia en el proceso de toma de decisiones a los directivos. Las consecuencias de estos acontecimientos dan lugar a la **situación problémica** que ocupa a este proyecto de investigación científica, cuyo contenido se expresa a continuación. En la entidad existe un atraso en la recopilación de información proveniente de todos los territorios. Esto trae consigo un incumplimiento en el tiempo esperado en la elaboración de los informes que contengan los indicadores medibles para la toma de decisiones realizada por el Centro de Dirección de Operaciones Postales (CDOP) y en la manipulación de la información, siendo el tiempo real mayor que el esperado. Esta situación provoca grandes esfuerzos por parte de los implicados así como la duplicación de la información.

Todas las limitaciones encontradas en estas áreas de la empresa dieron origen a dicha investigación ya que afecta considerablemente a la misma provocando insatisfacciones en los directivos que allí laboran; por lo que surge en la empresa la **necesidad** de implementar un sistema de información para la toma de decisiones del cierre de operaciones de las áreas postal y de tecnologías, que ilustre mediante gráficos el trabajo con grandes volúmenes de información para brindar a los usuarios diferentes soluciones.

Luego de diagnosticar la situación actual y hacer una revisión minuciosa de las faltas encontradas surge el siguiente **problema científico**: ¿Cómo realizar el cierre de operaciones en las áreas postal y de tecnologías de la ECC logrando una mayor eficiencia en la transmisión y manipulación de información para la toma de decisiones?

Este problema se enmarca en el **objeto de estudio:** Las áreas postal y de tecnologías de la ECC.

El **objetivo general** de este trabajo es implementar un sistema de información para lograr una mayor eficiencia en la toma de decisiones del cierre de operaciones en las áreas postal y de tecnologías de la ECC.

Los **objetivos específicos** que se persiguen con esta aplicación son los siguientes:

1. Estudiar el estado del arte de los sistemas de información en la actualidad. Los distintos enfoques sobre la captura, manipulación y extracción de la información, así como los tipos y usos de estos sistemas informáticos.
2. Estudiar el estado del arte de los sistemas existentes en la ECC que propicien información de interés para la toma de decisiones.
3. Definir la arquitectura base, los marcos de trabajo (*frameworks*[1]) y las herramientas de desarrollo a utilizar.
4. Diseñar y construir la Base de datos.
5. Implementar un sistema de información.

[1] *framework:* en español “marco de trabajo”. Ver epígrafe 1.4.4 para más detalles.

Los indicadores medibles para el cierre de operaciones en las áreas postal y de tecnologías de la ECC enmarcan sin lugar a dudas el **campo de acción** de este proyecto de investigación.

La **hipótesis** que se plantea es que con la realización de un sistema de información para la toma de decisiones del cierre de operaciones, aumentará en gran medida la eficiencia en la transmisión y manipulación de la información en las áreas postal y telegráfica de la ECC.

Luego del planteamiento de esta hipótesis, se procede a **operacionalizar las variables**:

Variables: Sistema de información para la toma de decisiones del cierre de operaciones, eficiencia en la transmisión y manipulación de la información.

Variable independiente: Sistema de información para la toma de decisiones del cierre de operaciones.

Variables dependientes: Eficiencia en la transmisión de la información.

Eficiencia en la manipulación de la información.

Operacionalización de las variables:

Variable conceptual	**Dimensión**	**Indicadores**	**Índice**
Eficiencia en la transmisión y manipulación de la información.	Transmisión de la información	Almacenamiento	Seguro
			Inseguro
		Recuperación	Completa
			Incompleta
			Nula
		Disponibilidad	Total
			Nula
		Rapidez	Rápida
			Lenta
	Manipulación de la información.	Búsqueda	Efectiva
			No efectiva
		Completamiento	Completa
			Incompleta
		Visualización en gráficos	Detallada

			No detallada
Sistema de información para la toma de decisiones del cierre de operaciones.	Sistema de información.	Entrada de la información.	Manual
			Por interoperabilidad con otros sistemas.
		Almacenamiento de la información.	Seguro
			Inseguro
		Procesamiento de la información.	Efectivo
			No efectiva
		Salida de la información.	Detallada
			No detallada

Tabla 1: Operacionalización de las variables.

Las **tareas** a cumplir para desarrollar cada uno de los objetivos se muestran a continuación:

1. Estudiar el estado del arte de los sistemas de información en la actualidad. Los distintos enfoques sobre la captura, manipulación y extracción de la información, así como los tipos y usos de estos sistemas informáticos.
2. Estudiar sistemas existentes en la Empresa de Correos de Cuba que propicien información de interés para la toma de decisiones.
3. Realizar un estudio del comportamiento de las tecnologías en el ámbito internacional para lograr la implementación de un sistema de alta calidad.
4. Fundamentar la metodología, herramientas y tecnologías a utilizar en el desarrollo del sistema.
5. Implementar el sistema de información respetando la arquitectura definida.
6. Definir la arquitectura base, los marcos de trabajo (*frameworks*) y las herramientas de desarrollo a utilizar.
7. Diseñar y construir la Base de datos.
8. Implementar módulo de Tecnología del Sistema.
9. Implementar módulo Postal del Sistema.
10. Implementar la Administración del Sistema.

La **población** que se aborda son las áreas postales y tecnológicas de las 28 entidades de la ECC. Para esto se ha estudiado una muestra de 7 entidades de Casa Matriz, lo que representa un 25% de la población.

Como **métodos de investigación científica** se utilizaron los siguientes:

Métodos Teóricos:

- Histórico lógico: Posibilitó el análisis histórico del proceso de gestión de información.
- Análisis y la síntesis: Se analiza la bibliografía y se realiza síntesis de la misma.

Métodos Empíricos:

- Entrevistas: Se realizan entrevistas con el fin de precisar el problema a resolver, capturar los indicadores medibles en cada área, así como para la validación de la propuesta que se presenta.
- Análisis de documentos: Se basa en la revisión a documentos utilizados en la investigación.

A continuación se muestra un **análisis financiero** para estimar los costos del proyecto:

Recursos Necesarios	**Etapa 1 ($)**	**Etapa 2 ($)**	**Etapa 3 ($)**	**Total**
Depreciación de PC	700	700	700	2100
Depreciación de Impresora	100	100	100	300
Materiales de oficina	40	30	30	100
Total	**840**	**830**	**830**	**$ 2500**

Tabla 2: Análisis financiero.

Luego de analizar los elementos materiales necesarios para desarrollar esta investigación, así como la depreciación de los mismos a lo largo del período de duración del desarrollo de la aplicación, se estima un presupuesto de $**2500**.

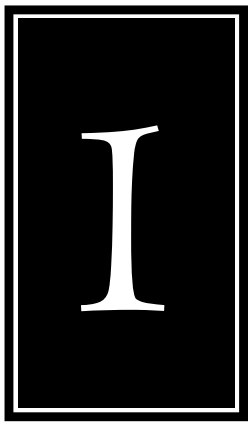
El **beneficio esperado** con esta aplicación es obtener un sistema Web de información que maneje de manera integrada la información que opera el CDOP para elevar la eficiencia en la generación de informes y en la toma de decisiones, reducir esfuerzo y eliminar la duplicación de información.

El contenido de este documento está estructurado en tres capítulos:

El **Capítulo 1** describe el marco teórico referencial del trabajo, realizando un análisis del estado del arte de los sistemas de información; el estado actual de los sistemas con que cuenta la empresa que pueden tributar información relevante a nuestro sistema de información, así como un análisis de los diferentes procesos objetos de automatización, imprescindibles para entender la lógica del negocio. Además se ofrece una breve panorámica de las características fundamentales de este tipo de sistemas y se realiza un estudio de las tecnologías en las que se apoya la realización de este software en función de un análisis de las tendencias actuales, apostando por supuesto por el Software Libre.

El **Capítulo 2** presenta la concepción de la arquitectura base del sistema. Se abordan los temas necesarios para dar la solución, como los estándares de codificación, los principios para la interfaz de usuario, el diseño y los diagramas de la base de datos; así como las clases usadas y el tratamiento de errores. Por último se tratan los beneficios obtenidos con el producto en la empresa Correos de Cuba: gracias a los despliegues de versiones beta y de la primera versión estable se han recogido estadísticas del funcionamiento del sistema y del volumen de información manejado.

Se abordan en el **Capítulo 3** las validaciones hechas al sistema de información para comprobar su correcto funcionamiento, tanto las pruebas de caja negra, las pruebas de caja blanca, y pruebas de velocidad de descarga en el cliente; por la importancia que lleva el rendimiento de la fachada de la aplicación.



La teoría del verdugo era que resultaba imposible cortar una cabeza si no había cuerpo del que cortarla; decía que nunca había tenido que hacer una cosa parecida en el pasado y que no iba a empezar a hacerla a estas alturas de su vida.

LEWIS CARROLL, Alicia en el país de las maravillas.

CAPÍTULO 1.

FUNDAMENTACIÓN TEÓRICA.

1.1- Introducción del capítulo

En este capítulo se describe el marco teórico referencial del trabajo, realizando un análisis del estado del arte de los sistemas de información; el estado actual de los sistemas con que cuenta la empresa que pueden tributar información relevante a nuestro sistema de información, así como un análisis de los diferentes procesos objetos de automatización, imprescindibles para entender la lógica del negocio. Además se ofrece una breve panorámica de las características fundamentales de este tipo de sistemas y se realiza un estudio de las tecnologías en las que se apoya la realización de este software en función de un análisis de las tendencias actuales, apostando por supuesto por el Software Libre.

1.2- Sistemas de Información

El mayor de los activos de una compañía hoy en día es su información, representada en su personal, experiencia, conocimiento, innovaciones, etcétera. Para poder competir, brindar un servicio cada vez más centrado en el usuario y lograr procesos productivos ágiles y eficientes, las organizaciones deben poseer

una fuerte infraestructura de información, en cuyo corazón se sitúa la infraestructura de la tecnología de información. De tal manera que el sistema de información se centre en estudiar las formas para mejorar los procesos dentro de la organización.

Un sistema de información es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Tratan el desarrollo, uso y administración de la infraestructura de la tecnología de la información en una organización (Langefors, 1973).

Un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información.

Entrada de Información: Es el proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas son datos o información que provienen o son tomados de otros sistemas o módulos. Esto último se denomina interfaces automáticas.

Las unidades típicas de entrada de datos a las computadoras son las terminales, las cintas magnéticas, las unidades de diskette, los códigos de barras, los escáneres, la voz, los monitores sensibles al tacto, el teclado y el mouse, entre otras.

Almacenamiento de información: El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos. La unidad típica de almacenamiento son los discos magnéticos o discos duros, los discos flexibles o diskettes y los discos compactos (CD-ROM).

Procesamiento de Información: Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otras cosas, que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados o un balance general de un año base.

Salida de Información: La salida es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, diskettes, cintas magnéticas, la voz, los graficadores y los plotters, entre otros. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada a otro Sistema de Información o módulo. En este caso, también existe una interfaz automática de salida. Por ejemplo, el Sistema de Control de Clientes tiene una interfaz automática de salida con el Sistema de Contabilidad, ya que genera las pólizas contables de los movimientos procesales de los clientes.

1.2.1- Tipos y Usos de los Sistemas de Información

Durante los próximos años, los Sistemas de Información cumplirán tres objetivos básicos dentro de las organizaciones:

1. Automatización de procesos operativos.
2. Proporcionar información que sirva de apoyo al proceso de toma de decisiones.
3. Lograr ventajas competitivas a través de su implantación y uso.

Los Sistemas de Información que logran la automatización de procesos operativos dentro de una organización, son llamados frecuentemente Sistemas Transaccionales, ya que su función primordial consiste en procesar transacciones tales como pagos, cobros, pólizas, entradas, salidas, etc. Por otra parte, los Sistemas de Información que apoyan el proceso de toma de decisiones son los Sistemas de Soporte a la Toma de Decisiones, Sistemas para la Toma de Decisión de Grupo, Sistemas Expertos de Soporte a la Toma de Decisiones y Sistema de Información para Ejecutivos; este es precisamente el grupo al que pertenece el software que se aborda. El tercer tipo de sistema, de acuerdo con su uso u objetivos que cumplen, es el de los Sistemas Estratégicos, los cuales se desarrollan en las organizaciones con el fin de lograr ventajas competitivas, a través del uso de la tecnología de información.

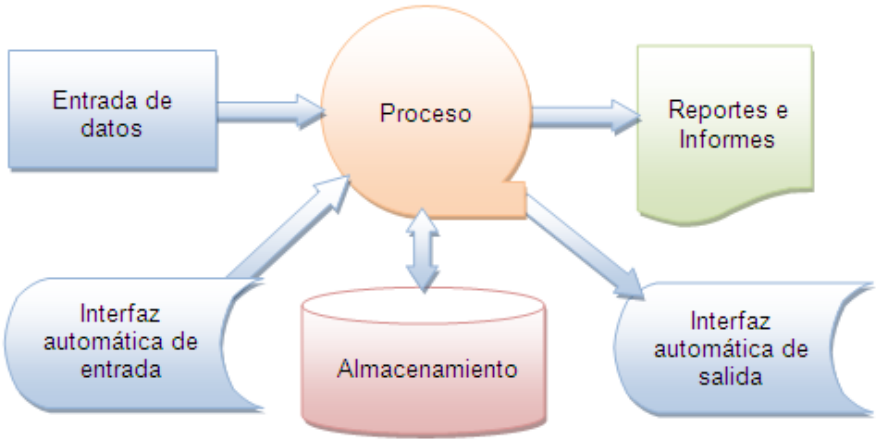


Figura 1: Proceso de un Sistema de Información para la Toma de Decisiones.

1.2.1- Sistema de Información para la Toma de Decisiones (DSS)

Un Sistema de Soporte a la Toma de Decisiones (DSS por sus siglas en inglés *Decision Support System*) es una herramienta de Inteligencia empresarial (*Business Intelligence*) que permite realizar el análisis de las diferentes variables de negocio para apoyar el proceso de toma de decisiones (I.O., y otros, 1991):

1. Permite extraer y manipular información de una manera flexible.
2. Ayuda en decisiones no estructuradas.
3. Permite al usuario definir interactivamente qué información necesita y cómo combinarla.
4. Suele incluir herramientas de simulación, modelización, etc.
5. Puede combinar información de los sistemas transaccionales internos de la empresa con los de otra empresa externa.

Un DSS da soporte a las personas que tienen que tomar decisiones en cualquier nivel de gestión, ya sean individuos o grupos, tanto en situaciones semiestructuradas y no estructuradas, a través de la combinación del juicio humano e información objetiva:

1. Soporta varias decisiones interdependientes o secuenciales.
2. Es adaptable por el usuario en el tiempo para lidiar con condiciones cambiantes.
3. Generalmente utiliza modelos cuantitativos (estándar o hechos a la medida).

4. Los DSS avanzados están equipados con un componente de administración del conocimiento que permite una solución eficaz y eficiente de problemas muy complejos.
5. Puede ser implantado para su uso en Web, en entornos de escritorio o en dispositivos móviles (PDA).
6. Permite la ejecución fácil de los análisis de sensibilidad.

1.2.1.1- DSS existentes

En la actualidad, y por la importancia que llevan estos sistemas dentro de las empresas, existen varios Sistemas de Soporte a la Toma de Decisiones. Los más comunes son los que se especializan dentro de un marco especializado. Estos DSS son en línea y casi todos gratuitos; pero no son DSS que sirvan para implantar en una empresa, porque sus áreas de atención son distintas: DSS para decisiones educacionales, para decisiones de salud, para decisiones sobre Tecnologías de la Información, etcétera. Ninguno de estos es compatible con las exigencias de la empresa por enmarcarse en áreas distintas.

Los sistemas que se necesitan son, por pertenecer a soluciones de *Business Intelligence*, siempre muy caros y de código cerrado. Por tanto, aunque se tienen en cuenta en este caso los dos exponentes más importantes, que servirán de base para dar la solución informática, no se serán tomados como candidatos.

ERGO

Es una solución brindada por la empresa canadiense *Technology Evaluation Centers*, sus principales características son:

- Simplifica y cuantifica las decisiones complejas
- Construye, mantiene y aumenta el capital cognoscitivo de la empresa
- Estandariza los datos primarios y los procesos de evaluación.

Su mayor defecto es que es un programa de escritorio, lo que atenta contra los requisitos funcionales que el sistema debe tener.

SelexSys Decision Assistant

Excelente software de escritorio de la empresa estadounidense *Xerox*. Características técnicas superiores a las de ERGO. Es obviado por la misma razón que el anterior.

1.3- Tecnologías a utilizar

A continuación se abordan temas tecnológicos que necesarios para el desarrollo de la solución. Todas las tecnologías, lenguajes y herramientas que se pretenden tratar son Software Libre (están bajo licencia GNU GPL u otra compatible con esta)[2], por la importancia de este para un desarrollo justo de las tecnologías y por el cual aboga el país.

El software libre es una cuestión de libertad, no de precio. Representa la libertad que tienen los usuarios para ejecutar el programa, para estudiar el funcionamiento del mismo y adaptarlo a sus necesidades, para redistribuir copias que ayuden a otros que las necesiten, así como para mejorar su funcionamiento y luego publicarlo. Su uso implica, entre otras cosas, no tener que pedir permiso ni pagar por ello (Stallman, 2004). Por sus características, significa hoy un paso importante en el desarrollo de las nuevas tecnologías de la información y las comunicaciones. A diferencia del software libre, el software propietario tiene limitadas posibilidades de acceso, modificación y redistribución, debido a que su código fuente no se encuentra disponible para un uso abierto. El software propietario requiere de una licencia que impide a los usuarios que lo utilizan distribuirlo libremente o cambiar sus funcionalidades sin una previa autorización o un pago que respalde las nuevas modificaciones.

Recientemente en La Universidad de las Ciencias Informáticas, en la ECC y en el país en general se ha venido observando una tendencia hacia la utilización de Software Libre por sus marcados beneficios. El presente trabajo parte de esta premisa y propone el análisis, diseño e implementación de un software que haga uso de las tecnologías y herramientas libres.

1.3.1- Arquitectura Cliente-Servidor

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

[2] La Licencia Pública General de GNU (GNU GPL) es la licencia creada por el Proyecto GNU de tipo *copyleft*. Licencias compatibles con esta pueden ser la Licencia BSD Modificada, la Licencia de software del W3C, etc.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma (Wikipedia, Colaboradores de, 2008).

Al estar nuestro sistema basado en la Web, responde a este tipo de arquitectura.

1.3.2- El patrón arquitectónico[3] Modelo Vista Controlador

Siguiendo la filosofía del modelo actual de desarrollo del software, para la realización del sistema se propone organizar los elementos de la aplicación en componentes independientes buscando alcanzar una mayor efectividad a la hora de administrarlos.

El patrón arquitectónico Modelo Vista Controlador (MVC) permite hacer la separación de las capas de interfaz, modelo y lógica de control de esta. La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario (The original MVC reports, 2007).

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, simplemente es necesario conocer la API (Interfaz de Aplicación) que existe entre niveles. La división en capas reduce la complejidad, facilita la reutilización y acelera el proceso de ensamblar o desensamblar alguna capa, o sustituirla por otra distinta (pero con la misma responsabilidad) (Kumbia Framework, 2007).

[3] Ver Capítulo 2 Epígrafe 2.2.1 para conocer más sobre patrones arquitectónicos.

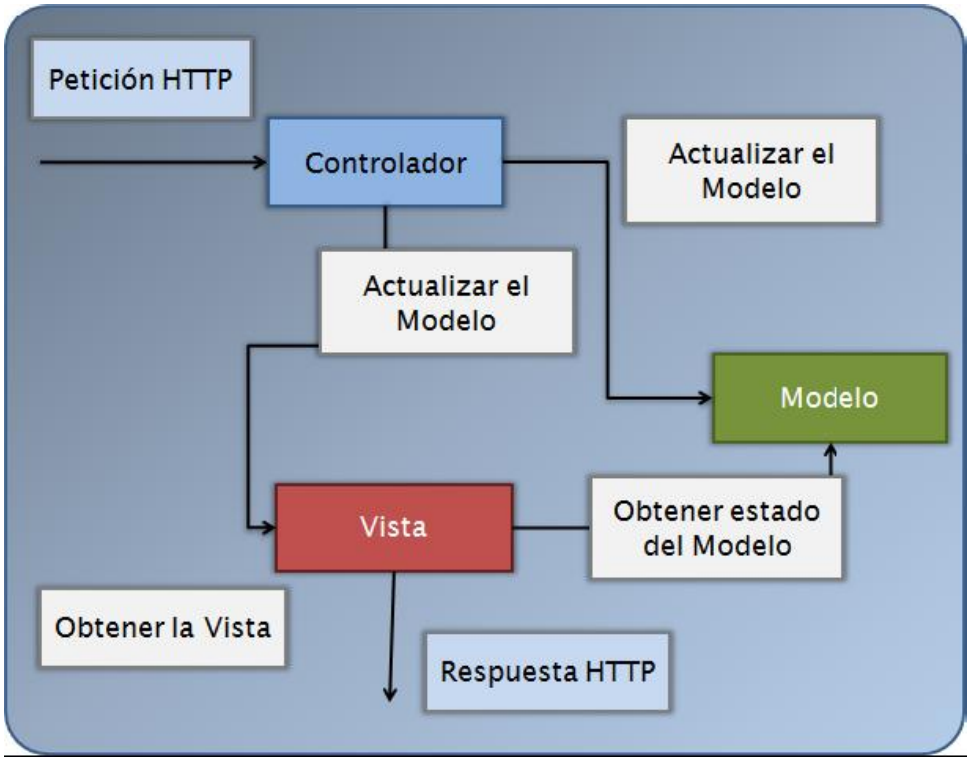


Figura 2: MVC.

Ventajas de esta arquitectura

El objetivo de este patrón es el realizar y mantener la separación entre la lógica de nuestra aplicación, los datos y la presentación. Esta separación tiene algunas ventajas importantes, como:

- Se puede identificar más fácilmente en qué capa se está produciendo un problema con sólo saber su naturaleza.
- Se pueden crear varias presentaciones sin necesidad de escribir varias veces la misma lógica de aplicación.
- Cada parte funciona independiente y cualquier cambio centraliza el efecto sobre las demás, así que se puede asegurar que una modificación en un componente realizará bien las tareas en cualquier parte de la aplicación.

1.3.3- Lenguajes de desarrollo

A continuación se ilustran los lenguajes de programación que se tuvieron en cuenta a la hora de seleccionar los más adecuados para la implementación de la solución.

1.3.3.1- Lenguajes del lado del cliente

Los lenguajes del lado del cliente son los encargados de dibujar, maquetar, estilizar y aportar dinamismo a la aplicación en los navegadores. Los lenguajes del lado del cliente abordados en este epígrafe son precisamente los utilizados, por ser estándares internacionales para el desarrollo Web. Los homólogos por los cuales se hubiera hecho un proceso de selección quedan exentos por su casi nulo desarrollo e incompetitividad frente a estos señalados.

JavaScript

JavaScript es el lenguaje de programación del lado del cliente más utilizado por su compatibilidad con la mayoría de los navegadores modernos (Eguíluz Pérez, 2007). Con él se pueden generar páginas dinámicas en función de las preferencias del usuario, validar los datos introducidos en un formulario o modificar dinámicamente el contenido de la página. Es multiplataforma e interpretado, es decir, no requiere compilación y tiene una sintaxis semejante a los lenguajes Java y C. Este lenguaje es muy fácil de aprender e ideal para agregar ciertas funciones rápidas a una página Web.

XHTML

El lenguaje XHTML (del inglés *eXtensible Hypertext Markup Language,* Lenguaje Extensible de Marcado de Hipertexto) es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, HTML es descendiente directo del lenguaje SGML, mientras que XHTML lo es del XML (que a su vez, también es descendiente de SGML).

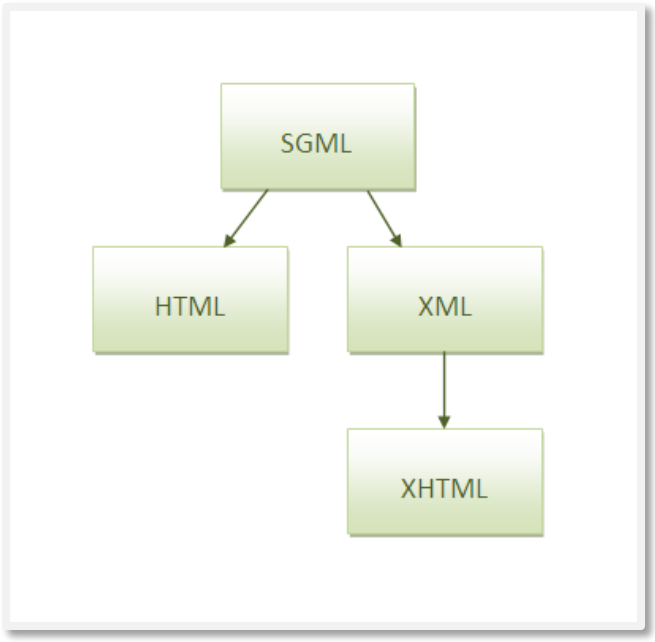


Figura 3: Esquema de la evolución de HTML y XHTML.

Las páginas y documentos creados con XHTML son muy similares a las páginas y documentos HTML. Las discusiones sobre si HTML es mejor que XHTML o viceversa son recurrentes en el ámbito de la creación de contenidos web, aunque no existe una conclusión ampliamente aceptada.

Actualmente, entre HTML 4.01 y XHTML 1.0, los mejores desarrolladores siempre escogen XHTML. En un futuro cercano, si los diseñadores deben escoger entre HTML 5 y XHTML 1.1 o XHTML 2.0, quizás la elección sea diferente.

CSS

CSS (del inglés *Cascading Style Sheets*, Hojas de Estilo en Cascada) es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas.

La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados “documentos

semánticos”). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (Eguíluz Pérez, 2007).

1.3.3.2- Lenguajes del lado del servidor

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor Web y le envía una respuesta al cliente a través de una página. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, tratamientos de la información, entre otras funciones.

Se presentan a continuación tres candidatos que tienen gran aceptación en el desarrollo Web mundial.

Perl

Perl es un lenguaje de programación muy utilizado para construir aplicaciones para la Web. Tiene gran popularidad por ser un intérprete que se distribuye de forma gratuita. Es fácil de usar, soporta tanto la programación orientada a objetos como la programación estructurada, posee también una enorme colección de módulos disponibles. Aunque inicialmente fue diseñado para un entorno UNIX, en la actualidad hay versiones para casi todos los sistemas operativos: DOS, WINDOWS NT, MacOS y otros (Perl, 2008).

PHP

PHP (del inglés *Hypertext Preprocessor*) es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor (Achour, y otros, 2007). Permite la conexión a gran cantidad de servidores de bases de datos lo que constituye una de sus características más destacadas y potentes. Tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, es rápido, posee una gran cantidad de funciones y mucha documentación. Fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores. Actualmente PHP se encuentra en su versión 5.2.

Ventajas de PHP:

- Es multiplataforma.

- Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros muchos. Siempre se podrá disponer de objetos de colección de bases de datos para situaciones que lo requieran. Destaca su conectividad con My SQL.
- Permite leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- PHP es completamente expandible. Está compuesto de un sistema principal (escrito por Zend), un conjunto de módulos y una variedad de extensiones de código.
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es software libre.
- Rapidez. PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz. Está completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.
- Muchas interfaces distintas para cada tipo de servidor. PHP actualmente se puede ejecutar bajo Apache, IIS, AOLServer, Roxen y THTTPD. Otra alternativa es configurarlo como módulo CGI.
- Una gran variedad de módulos cuando un programador PHP necesite una interface para una librería en particular, fácilmente podrá crear una API para esta. Algunas de las que ya vienen implementadas permiten manejo de gráficos, archivos PDF, Flash, Cybercash, calendarios, XML, IMAP, POP, etc.
- Permite las técnicas de Programación Orientada a Objetos.
- Posee una biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Java

Java es un lenguaje de desarrollo de propósito general y como tal es válido para realizar todo tipo de aplicaciones profesionales. Incluye una combinación de características que lo hacen único y está siendo adoptado por multitud de fabricantes como herramienta básica para el desarrollo de aplicaciones comerciales de gran repercusión. Puede considerarse como una evolución de C++, su sintaxis es parecida a la de este y al lenguaje C, aunque no acarrea los inconvenientes de los mismos ya que fue diseñado “partiendo desde cero”, es decir, no necesita ser compatible con versiones anteriores de ningún lenguaje y elimina herramientas de bajo nivel como punteros. Es un lenguaje orientado a objetos puro, creado para

trabajar con ellos desde cero. Es compilado e interpretado. Del código fuente se pasa a una representación denominada *bytecode* (compilación) lo que posteriormente es interpretado por la Máquina Virtual Java (JVM) que es la encargada de ejecutar el programa. El *bytecode* hace que Java sea independiente de la plataforma hardware y del sistema operativo, o sea, cualquier código fuente de Java puede ser compilado en cualquier máquina y sistema operativo que tenga un compilador Java, y que cualquier fichero de *bytecode* resultado de compilar un fichero fuente puede ser ejecutado en cualquier máquina y sistema operativo que tenga una Máquina Virtual de Java (Molina, 2006).

1.3.3.3- Fundamentación de la selección del lenguaje a utilizar

Analizadas las características principales de los lenguajes de programación del lado del servidor descritos en el epígrafe anterior se selecciona a continuación el más indicado para realizar la aplicación.

Todos los lenguajes analizados están soportados por múltiples plataformas, a nivel mundial los más utilizados son PHP y JAVA, sobre los que se pueden encontrar una gran cantidad de información, tutoriales, artículos y códigos de ejemplo, etcétera. Recordar que sólo se están teniendo en cuenta las tecnologías y/o lenguajes libres. El lenguaje PHP es recomendado para aplicaciones que necesiten ser implementadas con rapidez, por la sencillez con que se programa del lado del servidor. En la Universidad de las Ciencias Informáticas el lenguaje PHP es utilizado por gran cantidad de desarrolladores y precisamente los proyectos de producción vinculados al polo postal poseen una gran experiencia sobre este lenguaje. Por lo que se propone el leguaje PHP para realizar el software. Se propone además el uso de JavaScript, XHTML y CSS, teniendo en cuenta las ventajas que ofrecen para la programación del lado del cliente.

1.3.4- AJAX

AJAX (del inglés *Asynchronous JavaScript And XML*, JavaScript y XML asíncronos, donde XML es un acrónimo de *eXtensible Markup Language*), es una tecnología de desarrollo Web para crear aplicaciones interactivas sin la necesidad de cargar la página completamente entre una solicitud y otra.

Estas se ejecutan en el navegador, y mantiene comunicación asíncrona con el servidor en segundo plano mediante un canal de conexión. Debido a que la página Web no se refresca completamente, si no solamente se acceden a elementos de la misma mediante JavaScript u otro lenguaje del lado del cliente, se gana en rapidez e interactividad (Eguíluz Pérez, 2007).

Gracias a esta tecnología la Web se ha ido revolucionando, habiendo cada vez más aplicaciones potentes en este ámbito. Se utilizará por tanto AJAX en el sistema de información para hacer llamadas recurrentes al servidor que permitan al usuario seguir trabajando normalmente mientras se hacen las peticiones.

1.3.5- Servidores Web

Se presentan a continuación dos de los servidores HTTP que se pueden utilizar, los más acreditados.

***Internet Information Server* (IIS)**

Es el principal servidor de aplicaciones Web de Microsoft. Sus principales funcionalidades son la publicación de sitios y aplicaciones Web, sitios FTP (File Transfer Protocol), SMTP (Simple Mail Transport Protocol) y Servicios de noticias. Dispone de soporte necesario para crear páginas en ASP.

Su principal problema está en su pobre seguridad, el cual, fue resuelto posteriormente en la versión 6.0. Es un sistema destinado a la utilización de los servicios de Internet basado en la plataforma Windows (Wikipedia, Colaboradores de, 2007).

Apache HTTP Server

El servidor HTTP[4] Apache es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Apache tiene amplia aceptación en la red: desde 1996 se mantiene como el servidor Web más usado del mundo empleado por el 50.93% de los sitios web en el mundo, seguido del IIS con un 35.56%[5].

[4] Protocolo de Transferencia de Hipertexto (HTTP, *HyperText Transfer Protocol*).

[5] Según el portal netcraft.com, especializado en estadísticas de Internet: http://news.netcraft.com/. Dato obtenido el 8 de marzo de 2008.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente **(Apache, 2007)**.

Se montará el Sistema de Información sobre el servidor Apache, por sus anteriores características y además de que IIS es software no libre.

1.3.6- Navegadores

Un navegador Web es una aplicación que le permite a un usuario recuperar y visualizar documentos de hipertexto, que pueden estar escritos en HTML desde servidores Web de todo el mundo a través de Internet. Actualmente los navegadores actuales permiten mostrar o ejecutar gráficos, secuencias de videos, sonido, animaciones así como otros programas además de texto e hipervínculos. La comunicación entre un servidor Web y el navegador se realizan mediante el protocolo HTTP, aunque también soportan FTP[6] y HTTPS[7] (Network-Press.Org, 2003).

Existen disímiles navegadores entre los que se destacan: Internet Explorer (IE), Mozilla Firefox, Opera, Safari, Koqueror, Nestcape Navigator, entre otros muchos. En Cuba y más específicamente dentro de la universidad los más usados son: Internet Explorer y el Mozilla Firefox, este último poco a poco se va imponiendo sobre el conocido Internet Explorer debido a las grandes ventajas que presenta. Estudiar los navegadores Web a la hora de crear una aplicación Web es muy necesario porque la aplicación debe ser compatible con la mayor cantidad de navegadores disponibles para pueda ser accesada por cualquier tipo de usuario.

El sistema de información cumplirá con los estándares Web de desarrollo definidos por la W3C[8], por tanto todos los navegadores que sean capaces de cumplirlos podrán utilizar la aplicación sin ninguna dificultad. Para Internet Explorer, que a pesar de que no cumple algunos estándares es muy utilizado, la aplicación tiene reglas especiales que verifican si se está ejecutando en este, y adaptarla consecuentemente a este navegador.

---

[6] *File Transfer Protocol*, Protocolo de Transferencia de Archivos.

[7] El protocolo de red HTTPS es la versión segura del protocolo HTTP.

[8] *World Wide Web Consortium*, consorcio internacional que produce estándares para la World Wide Web.

1.4- Herramientas para el desarrollo de la solución informática

A continuación se tratarán una serie de herramientas necesarias para la implementación de la aplicación. Para una mejor compresión, cada uno de los sub-epígrafes que lo componen trata de una herramienta y de las diversas opciones a escoger, y se define entonces un sub-epígrafe para la opción seleccionada.

1.4.1- Entornos integrados de desarrollo

Entre los entornos de desarrollo libres más acreditados se encuentran NetBeans, MonoDevelop, Eclipse, entre otros. Se trabajó con Eclipse por ser el único con soporte para PHP, además de ser muy usado en la universidad, tener un ambiente amigable y ser muy potente.

1.4.1.1- Eclipse

Eclipse es una plataforma que proporciona un marco de trabajo o framework para desarrollar aplicaciones. Esto significa que distintos fabricantes pueden desarrollar e integrar sus herramientas con el WorkBench existente. El WorkBench es un conjunto de frameworks de Java y herramientas de desarrollo equipadas para constructores de herramientas. Eclipse se considera como uno de los IDE más poderosos para el desarrollo de aplicaciones en Java, provee soporte mediante plugins[9] para trabajar con entornos J2EE[10] y J2SE[11] utilizados en el desarrollo de este trabajo (Eclipse, 2000).

La filosofía de plugins permite a Eclipse que muchas personas o empresas creen dichos complementos para desarrollar en esta plataforma en diferentes lenguajes, en este caso se utilizará el plugin PHPEclipse.

[9] Un plugin o componente enchufable (o plug-in -en inglés "enchufar"-, también conocido como *addin*, *add-in*, *addon* o *add-on*) es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, para hacer así funcionar un dispositivo en otro programa.

[10] Java Platform, Enterprise Edition o Java EE (anteriormente Java 2 Platform, Enterprise Edition).

[11] Java Platform, Standard Edition o Java SE (anteriormente Java 2 Platform, Standard Edition).

Dentro de las funcionalidades del Eclipse, la adición de plugins puede facilitar la automatización de diferentes tareas, es el caso de un plugin que permite la administrar el control de versiones de un proyecto, seguidamente se abordará más sobre el tema.

1.4.2- Control de versiones

Es el proceso de gestión de las versiones (revisiones) de todos los elementos de configuración que forman la línea base de un producto o una configuración del mismo. Para la automatización de este proceso se emplea básicamente SVN, herramienta libre que provee una serie funcionalidades necesarias para el control de versiones.

Otra opción a tener en cuenta es CVS, pero esta tiene algunas limitaciones:

- Los archivos en el repositorio sobre la plataforma CVS no pueden ser renombrados.
- El protocolo CVS no provee una manera de que los directorios puedan ser eliminados o renombrados.
- Soporte limitado para archivos Unicode[12] con nombres de archivo no ASCII[13].

Por tanto, se trabajó con SVN.

1.4.2.1- SVN

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Subversion ofrece muchas funcionalidades como:

[12] Unicode es un estándar industrial cuyo objetivo es proporcionar el medio por el cual un texto en cualquier forma e idioma pueda ser codificado para el uso informático.

[13] El código ASCII (acrónimo inglés de *American Standard Code for Information Interchange* — Código Estadounidense Estándar para el Intercambio de Información), es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales. Fue creado en 1963 por el Comité Estadounidense de Estándares como una refundición o evolución de los conjuntos de códigos utilizados entonces en telegrafía.

integración con Eclipse, JDeveloper y Netbeans, presenta la mayoría de las facilidades de CVS y manejo eficiente de ficheros binarios, entre otras (Projects Subversion, 2001).

La experiencia con que cuenta el equipo de desarrollo con SVN fue otro motivo para la selección de esta herramienta.

1.4.2.2- SubEclipse

Esta herramienta no es más que un Plug-in que integra el Eclipse con el SVN. Haciendo uso de este software se podrá facilitar el control de versiones desde el IDE Eclipse de forma tal que se pueden hacer operaciones de *commit*, *update*, *delete*, entre otras. Esta herramienta es libre y es ampliamente utilizada por los desarrolladores de software que utilizan el Eclipse como IDE a nivel mundial.

1.4.3- Base de Datos

Un Sistema de Gestión de Bases de Datos (SGBD) consiste en un conjunto de programas, procedimientos y lenguajes que proporcionan las herramientas necesarias para trabajar con una base de datos, incorporar una serie de funciones que permitan definir los registros, sus campos, sus relaciones, insertar, suprimir, modificar y consultar los datos (Wikipedia, Colaborades de , 2008).

En la actualidad existen numerosos sistemas gestores de bases de datos, de ellos nos interesa estudiar las características de MySQL y PostgreSQL que además de sus potencialidades, cumplen con la condición de ser software libres.

MySQL posee un conjunto de utilidades y ventajas que lo hacen ser el más utilizado por los desarrolladores. Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL (MySQL AB, 2008).

De la misma forma se puede hacer referencia al PostgreSQL como otro de los sistemas gestores de bases de datos que más votos han obtenido por parte de los desarrolladores. PostgreSQL ofrece muchas ventajas para su compañía o negocio respecto a otros sistemas de bases de datos:

- Instalación Ilimitada: Es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la

principal fuente de incumplimiento de licencia. Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software. Esto tiene varias ventajas adicionales:

- ✓ Modelos de negocios más rentables con instalaciones a gran escala.
- ✓ No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- ✓ Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.
- ✓ Mejor soporte que los proveedores comerciales.
- ✓ Ahorros considerables en costos de operación.
- ✓ El software ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales, conservando todas las características, estabilidad y rendimiento.
- ✓ Además de esto, sus programas de entrenamiento son reconocidamente mucho más costo-efectivos, manejables y prácticos en el mundo real que aquellos de los principales proveedores comerciales.

- ❖ Estabilidad y confiabilidad legendarias: En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.
- ❖ Extensible: El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.
- ❖ Multiplataforma: PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows está actualmente en estado beta de pruebas.
- ❖ Diseñado para ambientes de alto volumen: PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes. Los principales proveedores de sistemas de bases de datos comerciales usan también esta tecnología, por las mismas razones.

- Herramientas gráficas de diseño y administración de bases de datos: Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos.

Al haberse estudiado ambos sistemas de gestión de bases de datos, se apuesta por MySQL, por su gran integración con PHP, su ligereza y tener además todas las ventajas al igual que PostgreSQL de ser libre. Además, MySQL es muy utilizado en la universidad, donde existen muchos proyectos productivos que lo utilizan y gran cantidad de documentación.

1.4.3.1- MySQL

Es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones (MySQL AB, 2008).

Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

Las siguientes características son implementadas únicamente por MySQL:

- Múltiples motores de almacenamiento (MyISAM, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example en 5.x), permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.

- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

1.4.4- Marco de trabajo

Un framework o marco de trabajo es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Típicamente incluye soporte de programas, bibliotecas, lenguajes de script y otros software utilizados que ayudan a unir y desarrollar los diferentes componentes de un proyecto. Provee una estructura y una metodología de trabajo que extiende o utiliza las aplicaciones del dominio. Los objetivos principales que se persiguen con su uso son: reutilizar código ya existente, promover buenas prácticas de desarrollo y acelerar el proceso de construcción de un software (Kumbia Framework, 2007).

A continuación se describe el framework seleccionado para la propuesta de solución.

1.4.4.1- Kumbia Framework

Kumbia es un Web *framework* libre escrito en PHP5. Basado en las mejores prácticas de desarrollo web, usado en software comercial y educativo, Kumbia fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones web, reemplazando tareas de codificación repetitivas por poder, control y placer.

Kumbia es una alternativa para proyectos en PHP con características como:

- Sistema de Plantillas sencillo.
- Administración de Cache.
- Scaffolding[14] Avanzado.
- Modelo de Objetos y Separación MVC.
- Soporte para AJAX.
- Generación de Formularios.

[14] *Scaffolding* es un método de meta-programación para construir aplicaciones basadas en bases de datos. Es una técnica soportada por algunos *frameworks* que implementan MVC, con la cual el programador puede escribir una especificación que describa cuál base de datos va a ser usada.

- Componentes Gráficos.
- Seguridad.

El número de prerrequisitos para instalar y configurar es muy pequeño, apenas Unix o Windows con un servidor Web y PHP5 instalado. Kumbia es compatible con motores de base de datos como MySQL, PostgreSQL y Oracle.

¿Qué es Kumbia?

Kumbia es un esfuerzo por producir un framework que ayude a reducir el tiempo de desarrollo de una aplicación web sin producir efectos sobre los programadores. Está basado en los siguientes conceptos:

- Compatible con muchas plataformas.
- Fácil de instalar y configurar.
- Fácil de aprender.
- Listo para aplicaciones comerciales.
- Convención sobre Configuración.
- Simple en la mayor parte de casos pero flexible para adaptarse a casos más complejos.
- Soporta muchas características de Aplicaciones Web Actuales.
- Soporta las prácticas y patrones de programación más productivos y eficientes.
- Produce aplicaciones fáciles de mantener.
- Está basado en Software Libre.

¿Por qué Kumbia y no otro framework?

En un mercado inundado de *frameworks* MVC que prometen ser la solución de desarrollo a cualquier tipo de proyecto, Kumbia pretende ser una solución a cualquier tipo de persona desde el principiante, pasando por el desarrollador que no tiene tiempo para aprender un nuevo framework hasta la empresa de desarrollo de software. Lo importante es que exista una necesidad y que Kumbia pueda ayudar a hacerla realidad.

Kumbia es innovador y su principal enfoque es desarrollar herramientas y escribir cada componente del framework pensando en que sea fácil de usar para cualquiera que lea su documentación.

Cualquier framework para la Web respetable tiene una aplicación del patrón MVC, un ORM (Mapeo objeto relacional), generación de logs, enrutamiento, plantillas, facilidades JavaScript, uso de AJAX y otras cosas más. Entonces, ¿cuál es la diferencia? La diferencia está en el tiempo dedicado a leer su documentación, las veces que se recurre a ella, las veces que se debe recordar sintaxis compleja, y lo más importante: ¿Después de cuánto tiempo se obtienen resultados?

Se le invita a comparar a Kumbia con otros *frameworks* y darse cuenta cómo, usando otras herramientas, escribes x, y, z código, haces 1, 2, 3, etc. pasos para hacer una simple tarea y como en Kumbia esto está reducido a su más mínima unidad con lo que el trabajo se hace más productivo, se aprende más rápido y se dan mejores soluciones, anticipándose a la competencia.

1.5- Conclusiones parciales del capítulo

Se ha hecho mención de los principales conceptos que se abordaran a lo largo de todo este trabajo, introduciendo al lector en la terminología utilizada en el campo de los sistemas de información. Se profundizó además en las tecnologías y las herramientas que se utilizarán en la solución.

2

— ¡Querida, realmente tengo que conseguir un lápiz más fino! No puedo en absoluto manejar éste: escribe todo tipo de cosas, sin que yo se las dicte.

LEWIS CARROLL, Alicia en el país de las maravillas.

CAPÍTULO 2.

IMPLEMENTACIÓN DEL SISTEMA DE INFORMACIÓN.

2.1- Introducción del capítulo

En este capítulo se presenta la concepción de la arquitectura base del sistema. Se abordan los temas necesarios para dar la solución, como los estándares de codificación, los principios para la interfaz de usuario, el diseño y los diagramas de la base de datos; así como las clases usadas y el tratamiento de errores. Por último se tratan los beneficios obtenidos con el producto en la empresa Correos de Cuba: gracias a los despliegues de versiones beta y de la primera versión estable se han recogido estadísticas del funcionamiento del sistema y del volumen de información manejado.

2.2- Arquitectura Base

En el campo del software, la arquitectura nos identifica los elementos más importantes de un sistema así como sus relaciones. Nos da una visión global del sistema.

¿Por qué es esto importante? Porque necesitamos arquitectura para entender el sistema, organizar su desarrollo, plantear la reutilización del software y hacerlo evolucionar.

Cómo determinar los elementos que definen una arquitectura es difícil y muy importante.

Generalmente las metodologías de desarrollo indican principios para identificar y diseñar una arquitectura, aunque por ahora la ayuda real que ofrecen es muy limitada al basarse en principios muy genéricos.

A continuación se ilustra la arquitectura base definida para el Sistema de Información.

2.2.1- Patrones arquitectónicos utilizados

Patrones arquitectónicos sobre aspectos fundamentales de la estructura de un sistema software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes (Buschamann, 1996).

Como se ha abordado en el Capítulo 1, se hizo uso de los patrones arquitectónicos Cliente-Servidor y MVC.

2.3- Estándares de codificación utilizados

Se definen a continuación los estándares de codificación usados para el desarrollo de la solución. Gran parte de estas reglas están basadas en las guías de estilo de grandes proyectos libres, como phpBB[15].

El uso de estándares tiene innumerables ventajas, entre ellas:

- Asegurar la legibilidad del código entre distintos programadores, facilitando la depuración del mismo.
- Proveer una guía para el encargado de mantenimiento/actualización del sistema, con código claro y bien documentado.
- Facilitar la portabilidad entre plataformas y aplicaciones.
- Permite una mejor organización y productividad en la programación de proyectos en equipos.

[15] phpBB es un conjunto gratuito de paquetes de código basados en el popular lenguaje de programación web PHP y lanzado bajo la Licencia pública general de GNU, cuya intención es la de proporcionar un sistema de foros.

2.3.1- Variables locales

- Los nombres de algunas variables locales, como los iteradores o los contadores, pueden especificarse en minúscula y de forma abreviada, siempre que su contexto sea específicamente local y su lectura sea intuitiva. `Ejemplos: $cont, $i, $j.`
- Al hacer asignaciones, debe existir un espacio a ambos lados del signo igual (=), esto funciona tanto para asignar un valor fijo, de otra variable o del resultado de una función.
- En el caso de un bloque de asignaciones relacionadas entre sí (por ejemplo, al inicializar un script), se pueden alinear los signos (=) agregando espacios extra, para mejorar la legibilidad.

2.3.2- Indentación y largo de líneas

- Indentar con 4 espacios, sin tabulador, para que cualquier editor de texto reconozca correctamente la indentación. Por otro lado, si bien existen editores que realizan corte automático de línea, es recomendable hacerlo en forma manual a los 75-80 caracteres.

2.3.3- Estructuras de control

- Incluye **`if`**, **`for`**, **`while`**, **`switch`**, etc. Deben tener un espacio entre la palabra clave y el paréntesis de apertura, para diferenciarlos de las llamadas a funciones. Se recomienda, aunque no sea necesario, la utilización de llaves. Esto mejora la legibilidad y disminuye la posibilidad de errores lógicos al agregar nuevas líneas de código.

2.3.4- Llamadas a funciones

- Las funciones deben ser llamadas sin espacio entre el nombre de la función, el paréntesis de apertura y el primer parámetro. En caso de varios parámetros, separar con espacios entre la coma y cada parámetro, y sin espacios entre el último parámetro, el paréntesis de cierre y el punto y coma.

2.3.5- Definición de funciones

- Todas las funciones deben tener un comentario, antes de su declaración, explicando que hacen.

- Las definiciones de funciones utilizan el estilo *BSD/Allman*. Las características más importantes se resaltan a continuación:
  - ✓ El nombre debe ser lo más descriptivo posible.
  - ✓ Se debe evitar el uso de abreviaturas.
  - ✓ Se debe utilizar la convención *lowerCamelCase*. Para más información, (http://en.wikipedia.org/wiki/CamelCase).
- Colocar los argumentos con valores por defecto, al final de la lista.
- Siempre intentar retornar un valor significativo.
- La llave de inicio de la función se coloca en la misma línea de declaración, luego del paréntesis de cierre de los parámetros.

2.3.6- Definición de clases

- El nombre debe ser descriptivo, evitando abreviaturas, usando la convención *UpperCamelCase* (http://en.wikipedia.org/wiki/CamelCase).
- La llave de inicio de la clase se coloca en la línea siguiente, indentada correctamente.
- Todos los miembros de la clase deben ser privados, es decir, únicamente accesibles a través de métodos de la misma.

2.3.7- Comentarios

- Se aconseja el uso de comentarios en línea para facilitar la comprensión del código, sobre todo en procedimientos complejos. Los comentarios pueden ser con fin documental o bien como 'ayuda-memoria'.
- Se recomienda utilizar los estilos de C (`/* */`) y C++ (`//`), no tanto así el signo numeral o sharp (`#`).

2.3.8- Inclusión de código

- Salvo casos específicos y puntuales, utilizar `require_once` para incluir código incondicionalmente, e `include_once` para los casos condicionales (o sus equivalentes en otros lenguajes).

2.3.9- Etiquetas de bloque PHP

- Siempre utilizar `<?php` y `?>` para iniciar y terminar un bloque de código PHP, no las variantes `<?` y `?>` o `<%` y `%>`. Esto asegura compatibilidad entre diversas configuraciones de equipos.

2.3.10- URLs[16] de ejemplo

- Para denotar URLs de ejemplo y direcciones de e-mail en la documentación y comentarios, utilizar las especificadas en la RFC2606 (http://www.faqs.org/rfcs/rfc2606). Éstas pueden ser: *example.com, example.org, example.net*.

2.3.11- Variables globales (Constantes)

- Los nombres de variables globales deben ser siempre en MAYÚSCULAS, separando las palabras con guiones bajos (_).
- Existen tres excepciones al punto anterior, las cuales deben escribirse siempre en minúscula: **`true`**, **`false`** y **`null`**.

2.3.12- Formato para guardar archivos con extensión ".php"

- Los archivos con código PHP, deben de ser guardados en formato ASCII utilizando la codificación ISO-8859-1. El formato ASCII con codificación ISO-8859-1 es el formato en que se guardan los archivos de texto plano (".txt"). La razón de este estándar es que determinados editores HTML agregan códigos de carácter extraño de salto de línea (como si se tratara de un archivo binario) y esto puede ocasionar que el intérprete de PHP encuentre problemas a la hora de leer el script.

[16] **URL** significa ***U****niform* ***R****esource* ***L****ocator*, es decir, localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

2.4- Principios de diseño para la Interfaz de Usuario

El aspecto de la interfaz de usuario de aplicaciones interactivas (en particular las aplicaciones web) es un punto crítico en el desarrollo que las modernas metodologías tienden a descuidar (Pérez Subirats, 2003).

Ante el cúmulo de información en diversos formatos que introduce el WWW[17] se hace necesario atender al diseño informacional de cualquier página o sitio. A menudo, se menosprecian normas simples para la realización de interfaces como el correcto del lenguaje; la necesaria correspondencia semántica entre titulares y contenidos, así entre íconos y textos. Estas y otras transgresiones a un diseño correcto producen ambigüedades en la información que ofrece un sitio y generalmente obedecen a la carencia de un estudio previo y unos criterios apropiados para una organización y presentación ajustada a las necesidades de información y comunicación de sus usuarios potenciales.

Para la arquitectura de la información de la solución se tuvieron en cuenta la Pautas de Usabilidad y Accesibilidad según WAI-W3C[18]:

- Proporcionamos alternativas equivalentes para el contenido visual y auditivo.
- Utilizamos marcadores y hojas de estilo apropiadamente.
- Identificamos el idioma usado.
- Las tablas se transforman correctamente.
- Las páginas que incorporan nuevas tecnologías se transforman correctamente.
- Se asegura al usuario el control sobre los cambios de los contenidos tempo-dependientes.
- El usuario tiene accesibilidad directa de las interfaces incrustadas.
- El diseño es independiente del dispositivo.
- Se utilizan las tecnologías, estándares y pautas W3C.
- Se proporciona información de contexto y orientación.
- Se proporcionan mecanismos claros de navegación.
- Los documentos mostrados son claros y simples.

[17] *World Wide Web* o Telaraña Mundial.

[18] La *Web Accessibility Initiative* (WAI) o Iniciativa para la Accesibilidad Web es una rama del *World Wide Web Consortium* (W3C) que vela por la accesibilidad de la Web.

Asimismo, el diseño gráfico del producto web está basado en el Manual de Identidad de la Dirección de Publicidad de Correos de Cuba, el cual recoge el diseño de la Identidad Corporativa, la Imagen Corporativa y la Marca de la empresa Correos de Cuba. Diseño, paleta de colores, tipografías, imágenes e ilustraciones fueron tomadas de dicho manual.

A continuación algunos detalles de la marca, la tipografía utilizada, la paleta de colores, etc.

Figura 4: Paleta de colores.

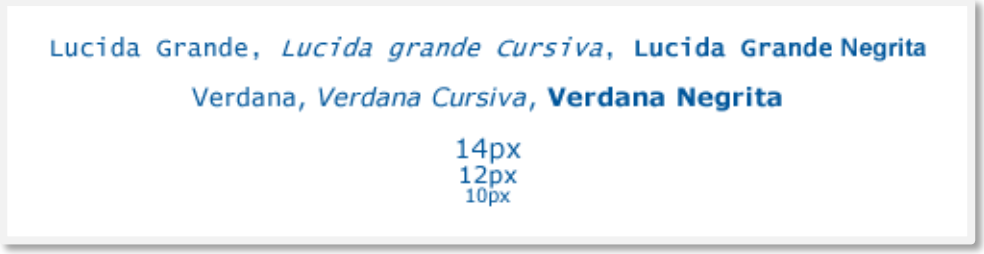


Figura 5: Tipografía.

Figura 6: Marca.

Se presentan a continuación las principales áreas en el sitio, en la siguiente ilustración:

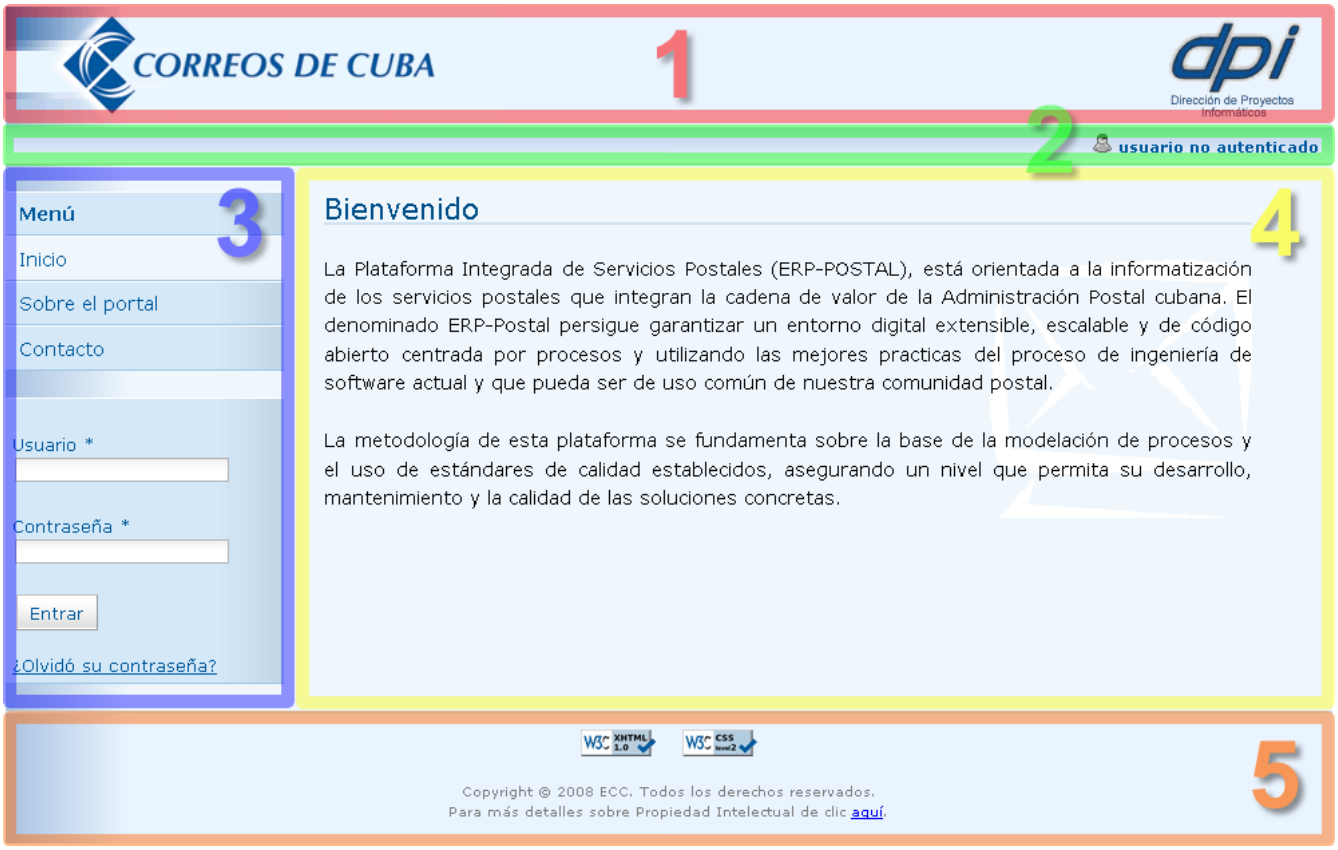


Figura 7: Interfaz gráfica del portal Web.

Descripción de cada una de las áreas:

1. **El área de encabezado:** se puede encontrar en ella el logotipo de Correos de Cuba como empresa al que pertenece el producto, además del logotipo de la Dirección de Proyectos Informáticos. Como aconsejan las buenas pautas de accesibilidad el logo es un vínculo que lleva a la página de inicio del programa.
2. **Barra de información al usuario:** Está sección se compone de dos partes fundamentales:
    a. **Parte izquierda:** Sirve como *breadcrumbs*[19], mostrando el nombre de la página a la que se está accediendo y la provincia o territorio desde el que se está accediendo.

[19] Palabra inglesa que significa migaja de pan. En el ámbito Web se le llama “breadcrumbs” a la sucesión de vínculos que nos hace saber de forma rápida en qué lugar estamos del sitio Web.

   b. **Parte derecha:** Muestra el nombre del usuario que está logueado, si se hace clic sobre este nombre nos llevará a una ventana donde el usuario podrá cambiar los datos de su cuenta. Muestra el botón para cerrar la sesión.
3. **Menú vertical:** En el menú se mostrarán las opciones de trabajo a las que tenga acceso el usuario. Es la principal vía traslado desde una página a otra.
4. **Bloque contenedor principal:** Es aquí donde el usuario llevará a cabo las acciones, en este bloque se mostrarán los formularios, resultados, gráficos, etc., con los que el usuario interactúa.
5. **Pie de página o *footer*:** contiene los Derechos de Autor sobre el contenido de la aplicación y los íconos que hacen válida a la solución en cuando a CSS y XHTML.

El diseño de los sitios Web es un tema objeto de tratamiento frecuente en la literatura especializada, como resultado del crecimiento espectacular experimentado por estos durante la última década. Esgrimiendo con rigor las múltiples normas, las recomendaciones y los requerimientos existentes con estos fines, se puede afirmar que la solución propuesta cumple con los siguientes aspectos:

- El diseño y la programación de la Web se subordinan al control del usuario, a sus requerimientos organizativos y sus niveles cognoscitivos.
- Se logra armonía entre el fondo de las páginas y el contenido que se muestra. Cuando hay textos extensos se usa un tamaño de fuente pequeño, y se contrasta con el fondo para aumentar su legibilidad.
- El espaciado, el tamaño de fuente y el interlineado son los adecuados para la comprensión de los textos y la estética de la página. Los títulos, subtítulos y encabezados se utilizan para enfatizar que se está en una sección específica.
- Las ilustraciones e imágenes se corresponden con el contenido de la página.
- Se utilizan correctamente los colores dentro del cuadro cromático escogido.
- Se Emplean sólo los gráficos animados necesarios para no abarrotar la página Web, cansar a los navegantes o demorar el acceso.

2.5- Diseño de la Base de Datos. Diagrama de Clases Persistentes

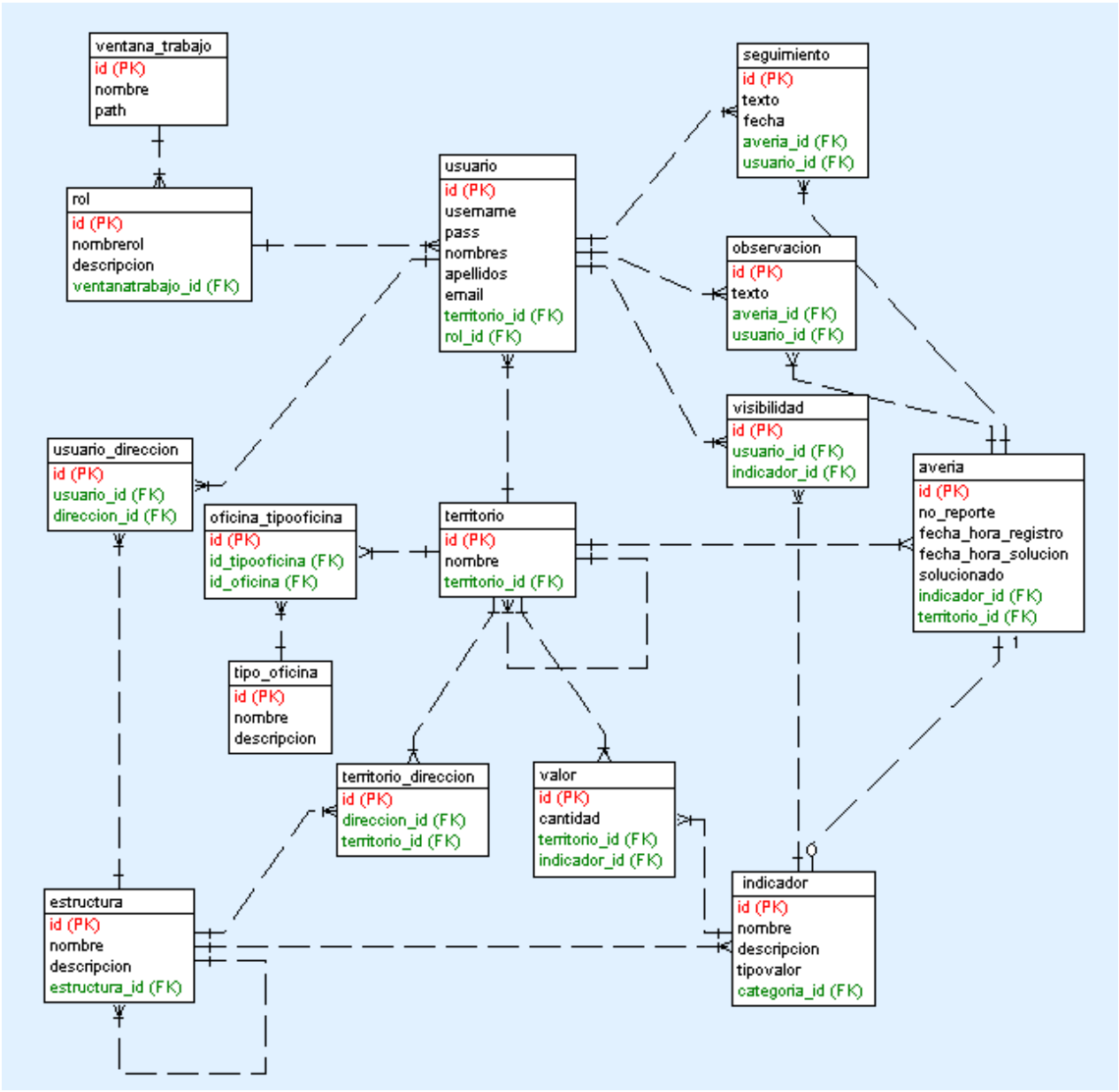


Figura 8: Diagrama de la Base de Datos.

2.6- Descripción de las principales Clases usadas

En esta sección se hará una descripción de aquellas clases usadas en el desarrollo de la aplicación: las clases Entidad, las Controladoras y las clases Interfaz.

2.6.1- Clases Entidad

Nombre: Rol	
Tipo clase: Entidad	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ActiveRecord. Hereda la capacidad de dar altas, bajas, modificaciones al modelo de datos. Encapsula la lógica del negocio y del modelo de datos de un rol.

Nombre: Usuario	
Tipo clase: Entidad	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ActiveRecord. Hereda la capacidad de dar altas, bajas, modificaciones al modelo de datos. Encapsula la lógica del negocio y del modelo de datos de un usuario.

Nombre: Estructura	
Tipo clase: Entidad	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ActiveRecord. Hereda la capacidad de dar altas, bajas, modificaciones al modelo de datos. Encapsula la lógica del negocio y del modelo de datos de una estructura.

Nombre: Categoria	
Tipo clase: Entidad	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ActiveRecord. Hereda la capacidad de dar altas, bajas, modificaciones al modelo de datos. Encapsula la lógica del negocio y del modelo de datos de una categoría.

Nombre: Indicador	
Tipo clase: Entidad	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ActiveRecord. Hereda la capacidad de dar altas, bajas, modificaciones al modelo de datos. Encapsula la lógica del negocio y del modelo de datos de un indicador.

Nombre: Territorio	
Tipo clase: Entidad	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ActiveRecord. Hereda la capacidad de dar altas, bajas, modificaciones al modelo de datos. Encapsula la lógica del negocio y del modelo de datos de un territorio.

Nombre: Oficina	
Tipo clase: Entidad	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ActiveRecord. Hereda la capacidad de dar altas, bajas, modificaciones al modelo de datos. Encapsula la lógica del negocio y del modelo de datos de una oficina.

Nombre: Despacho	
Tipo clase: Entidad	
Atributo	**Tipo**
-	-
Responsabilidad	

Descripción:	Subclase de ActiveRecord. Hereda la capacidad de dar altas, bajas, modificaciones al modelo de datos. Encapsula la lógica del negocio y del modelo de datos de despacho.

2.6.2- Clases Controladoras

Nombre: VicepresidenciaController	
Tipo clase: Controladora	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ApplicationController. Recibe, controla, dirige peticiones desde y hacia el usuario. Controla la seguridad de las acciones que se realizan en el controlador de vicepresidencia.

Nombre: AdministracionController	
Tipo clase: Controladora	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ApplicationController. Recibe, controla, dirige peticiones desde y hacia el usuario. Controla la seguridad de las acciones de administración.

Nombre: InicioController	
Tipo clase: Controladora	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ApplicationController. Recibe, controla, dirige peticiones desde y hacia el usuario. Controla el acceso y salida de la aplicación.

Nombre: CClasificacionController	
Tipo clase: Controladora	
Atributo	**Tipo**
-	-

Responsabilidad	
Descripción:	Subclase de ApplicationController. Recibe, controla, dirige peticiones desde y hacia el usuario. Controla la seguridad de las acciones del controlador de centro de clasificación.

Nombre: ReporteController	
Tipo clase: Controladora	
Atributo	**Tipo**
-	-
Responsabilidad	
Descripción:	Subclase de ApplicationController. Recibe, controla, dirige peticiones desde y hacia el usuario. Invoca a las acciones pertinentes para la generación de los reportes.

2.6.3- Clases Interfaz

En epígrafes anteriores se explica como es el funcionamiento del marco de trabajo. En esta explicación se ve que la forma de representar la información es a través de vistas HTML con código PHP embebido. Por esto en vez de clases interfaz se usan vistas.

Nombre: GestionarRol	
Tipo clase: Vista	
Atributo	**Tipo**
-	-
Responsabilidad	**Descripción**
tablaListarRoles	Lista todos los roles en base de datos y brinda los enlaces a las acciones baja y modificación de un rol.
adicionarRolBoton	Muestra la vista de la acción que da alta a un rol.

Nombre: GestionarUsuario	
Tipo clase: Vista	
Atributo	**Tipo**
-	-
Responsabilidad	**Descripción**
tablaListarUsuarios	Lista todos los usuarios en base de datos y brinda los enlaces a las acciones baja y modificación de un usuario.
adicionarUsuarioBoton	Muestra la vista de la acción que da alta a un usuario.

Nombre: GestionarEstructura	
Tipo clase: Vista	
Atributo	**Tipo**

-	-
Responsabilidad	**Descripción**
tablaListarEstrucutras	Lista todas las estructuras en base de datos y brinda los enlaces a las acciones baja y modificación de una estructura.
adicionarEstructuraBoton	Muestra la vista de la acción que da alta a una estructura.

Nombre: GestionarCategoria	
Tipo clase: Vista	
Atributo	**Tipo**
-	-
Responsabilidad	**Descripción**
tablaListarCategorias	Lista todos las categorías en base de datos y brinda los enlaces a las acciones baja y modificación de un rol
adicionarCategoriaBoton	Muestra la vista de la acción que da alta a una categoría.

Nombre: GestionarTerritorio	
Tipo clase: Vista	
Atributo	**Tipo**
-	-
Responsabilidad	**Descripción**
tablaListarTerritorios	Lista todos los territorios en base de datos y brinda los enlaces a las acciones baja y modificación de un territorio.
adicionarTerritorioBoton	Muestra la vista de la acción que da alta a un territorio.

Nombre: GestionarIndicador	
Tipo clase: Vista	
Atributo	**Tipo**
-	-
Responsabilidad	**Descripción**
tablaListarIndicadores	Lista todos los indicadores en base de datos y brinda los enlaces a las acciones baja y modificación de un indicador
adicionarRolBoton	Muestra la vista de la acción que da alta a un indicador.

Nombre: GestionarOficinas	
Tipo clase: Vista	
Atributo	**Tipo**
-	-

Responsabilidad	Descripción
tablaListarOficinas	Lista todas las oficinas en base de datos y brinda los enlaces a las acciones baja y modificación de una oficina.
adicionarOficinaBoton	Muestra la vista de la acción que da alta a una oficina.

Nombre: EmitirDespacho	
Tipo clase: Vista	
Atributo	**Tipo**
-	-
Responsabilidad	**Descripción**
destinoSelect	Muestra todos centros de clasificación y unidades a las que se puede enviar el despacho.
via_eSelect	Muestra las vías en las que se puede encaminar un despacho.
cant_sTextBox	Recoge la cantidad de sacas que viajan en el despacho.
tiposProductos	Es un componente HTML que agrupa los tipos de productos que puede llevar los despachos y sus cantidades.
emitirDespachoBoton	Botón para emitir el despacho

2.7- Tratamiento de errores

En la aplicación los errores se tratan de dos maneras diferentes. Primeramente, utilizando el lenguaje JavaScript y el framework Dojo Toolkit incluido en la solución. Mediante este se chequean las posibles situaciones excepcionales en el navegador, lo que es utilizado principalmente para validar las entradas del usuario, para evitar que viajen al servidor datos que ya en el cliente se sabía que eran incorrectos.

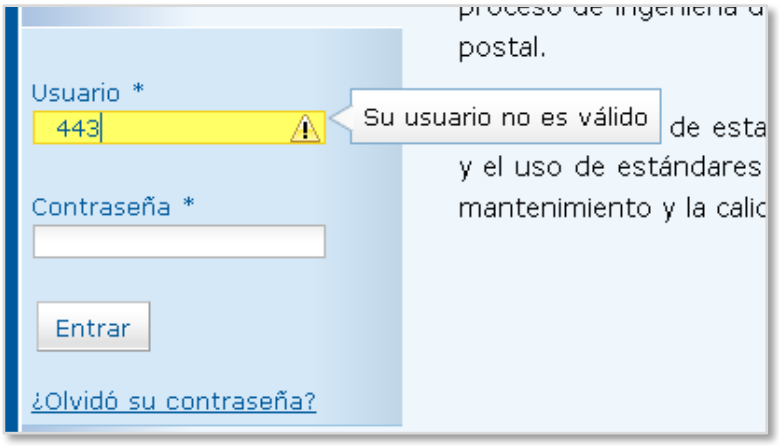


Figura 9: tratamiento de errores del lado del cliente.

El segundo caso se utiliza cuando no es posible detectar el error en el cliente y se ejecuta la comprobación en el servidor.

2.8- Concepción general de la ayuda

La solución propuesta, al cumplir las pautas de accesibilidad y usabilidad, se convierte en una herramienta de trabajo bastante fácil e intuitiva. Pero, como la aplicación va dirigida a una amplia gama de usuarios, algunos por su trabajo dentro de la empresa tienen conocimientos de informática pero otros tienen pocos conocimientos en la rama, y estos además llevan un ritmo de trabajo muy agitado; se hace necesario que el sistema incluya una ayuda para socorrer al usuario que no entienda el flujo de su trabajo.

Por lo anterior, la ayuda se inserta dentro del flujo básico de trabajo, de modo que los usuarios accedan a ella sin hacerlo explícitamente. Esto se materializa por ejemplo en los formularios donde deben entrar datos cuando cada vez que el usuario se coloque sobre un campo se le mostrará un globo de ayuda con un ejemplo de cómo llenar el campo. Se consideró que una ayuda formal, independiente de las acciones habituales del usuario sería poco funcional.

2.9- Diagrama de despliegue

En la siguiente figura se muestra el diagrama de despliegue correspondiente al producto de software construido:

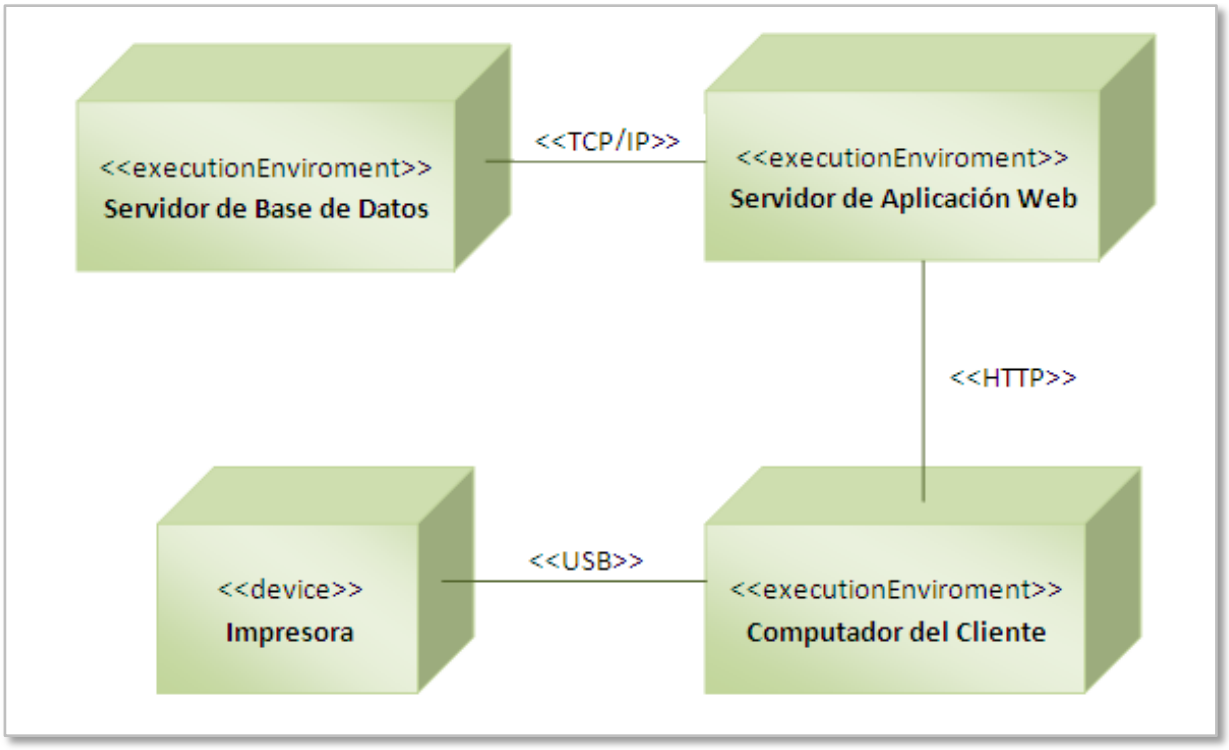


Figura 10: Diagrama de despliegue.

2.10- Beneficios obtenidos con el producto

El sistema de información será en un futuro una parte importante de la Plataforma Integrada de Servicios Postales (ERP-POSTAL) que se encuentra en desarrollo. Esta pieza de software se comunicará mediante interfaces automáticas de entrada y salida con los demás módulos de esta plataforma, interfaces implementadas en forma de servicios web; que permitirán el desarrollo y la gestión de todos los procesos productivos de la entidad.

El sistema propuesto desprenderá, aparte del anterior, varios beneficios en el momento:

1. Los trabajadores de la empresa Correos de Cuba tendrán un sistema que les permitirá realizar su trabajo de forma sencilla y dinámica, por tanto la eficiencia en el trabajo se elevará significativamente.

2. A través de la interfaz del sistema, los usuarios llenarán los datos que antes copiaban en documentos como hojas de cálculo, archivos de texto plano o correos electrónicos. Esto reduce el esfuerzo y el margen de error al mínimo pues se registrarán datos válidos y sólo los necesarios.
3. El que recibe la información ya no tendrá que agruparla, sólo tendrá que observar un informe o un gráfico para ver el comportamiento de los indicadores. Hay entonces un ahorro por concepto de papel, tinta y tiempo de impresión.
4. Se elimina también la duplicación de la información por lo antes mencionado.
5. El sistema presenta un costo, por concepto de salarios, que es prácticamente imperceptible dado que el equipo de desarrollo es no profesional.
6. El sistema ha sido desarrollado totalmente con Software Libre, por lo que no es necesaria la adquisición de licencias para el uso de este producto en la entidad.

2.11- Conclusiones parciales del capítulo

Se presentó la concepción de la arquitectura base del sistema. Se abordaron los temas necesarios para construir la aplicación. Por último, se abordó el tema de los beneficios obtenidos con el producto en la empresa Correos de Cuba.

—Faltan todavía muchas pruebas, con la venia de Su Majestad —dijo el Conejo Blanco, poniéndose apresuradamente de pie—. Acaba de encontrarse este papel.

LEWIS CARROLL, Alicia en el país de las maravillas.

CAPÍTULO 3.

VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

3.1- Introducción del capítulo

En el desarrollo del software, las posibilidades de error son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso indebido de las estructuras de datos, errores al enlazar módulos, etc. El desarrollo del software ha de ir acompañado de alguna actividad que garantice la calidad; la prueba es un elemento crítico para la garantía de calidad del *software*. Por ello, en este tercer capítulo se tocarán sobre las pruebas y las validaciones de los resultados de diferentes flujos de negocio de la solución informática.

Las Pruebas de caja Negra serán el primer punto a abordar: su objetivo, alcance y finalmente casos de prueba en los que se validará si el software funciona correctamente. Luego, entrando en más detalles, se hablará sobre las Pruebas de Caja Blanca, utilizando dentro de la misma la métrica de complejidad ciclomática en cada elemento o módulo para validar la solución propuesta.

Luego, teniendo en cuenta la importancia de la velocidad de descarga de los sitios web para la interacción con el usuario, se expondrán las pruebas de velocidad de descarga en el cliente.

3.2- Pruebas de Caja Negra

3.2.1 - Objetivo

El objetivo de realizar este tipo de prueba al sistema es para detectar el incorrecto o incompleto funcionamiento de este, así como los errores de interfaces, rendimiento y errores de inicialización y terminación.

3.2.2 - Alcance

El proceso de pruebas de caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar el comportamiento de la interfaz gráfica, su interacción con el usuario y la calidad funcional.

3.2.3 - Descripción

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Dentro de las técnicas de Prueba de Caja Negra, se utilizó la Partición de Equivalencia. Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones de software. Mediante la aplicación de esta se examinó los valores válidos e inválidos de las entradas existentes en el software, descubriendo de forma inmediata clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

Para definir las clases de equivalencia se tuvieron en cuenta un conjunto de reglas:

- Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica la cantidad de valores, se identifica una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, se identifica una clase válida para cada uno de ellos y una clase inválida.
- Si una condición de entrada especifica una situación de tipo “debe ser”, se identifica una clase válida y una inválida.

Luego de tener las clases válidas e inválidas definidas, se procedió a definir los casos de pruebas, para los casos de uso más significativos (Insertar Anomalía de Indicador, Emitir Resumen Diario, Crear Vista de Impresión). Cada uno de estos casos de pruebas se definió teniendo en cuenta lo siguiente:

a) Escribir un nuevo caso de cubra tantas clases de equivalencia válidas no cubiertas como sea posible hasta que todas las clases de equivalencia hayan sido cubiertas por casos de prueba.
b) Escribir un nuevo caso de prueba que cubra una y solo una clase de equivalencia inválida hasta que todas las clases de equivalencias inválidas hayan sido cubiertas por casos de pruebas.

3.2.4 - Casos de Pruebas

Caso de Uso	Insertar Anomalía de Indicador			
Caso de prueba	Insertar Anomalía de Indicador			
Condiciones				
Clases válidas	**Clases inválidas**	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Se llenan los datos correspondientes. Observación: (no se llena). Se da clic en el botón Guardar.	Anomalía (Observación)	Diálogo de error: “Debe llenar los campos que faltan”.	Satisfactorio	
Se llenan los datos correspondientes. Observación:”proble mas con la electricidad”. Se da clic en el botón Guardar.		Se crea una nueva anomalía en la BD para el indicador.	Satisfactorio	

Caso de Uso	Emitir Resumen Diario			
Caso de prueba	Emitir Resumen Diario			
Condiciones				
Clases válidas	**Clases inválidas**	**Resultado esperado**	**Resultado de la prueba**	**Observaciones**
Se trata de enviar el Resumen Diario sin haber llenado las observaciones finales de las anomalías vigentes.	Anomalía	Diálogo de error: "Usted está a punto de enviar un resumen diario al que le faltan observaciones por llenar, debe rellenar todos los campos resaltados".	Satisfactorio	
Se trata de enviar el Resumen Diario sin haber llenado los valores de los indicadores.	Valor	Diálogo de error: "Usted está a punto de enviar un resumen diario al que le faltan valores debe rellenar todos los campos resaltados".	Satisfactorio	
Se trata de enviar el Resumen Diario sin haber llenado los valores de los indicadores y las observaciones finales de las anomalías vigentes.	Valor Anomalía	Diálogo de error: "Usted está a punto de enviar un resumen diario al que le faltan valores y observaciones por llenar, debe rellenar todos los campos resaltados".	Satisfactorio	
Se trata de enviar el Resumen Diario con todos los campos rellenados.		Dialogo de aprobación: "Usted está a punto de enviar un Resumen Diario, dé clic en aceptar para confirmar su envío". Los datos son guardados en la BD.	Satisfactorio	

3.3- Pruebas de Caja Blanca

3.3.1- Objetivos

El objetivo de realizar este tipo de prueba al sistema es que se garantice que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo o método, todos los bucles en sus límites operacionales así como las estructuras internas de datos para asegurar su validez.

3.3.2- Alcance

El proceso de pruebas de caja blanca se va a concentrar principalmente en validar a través del marco de trabajo de software libre PHPUnit[20], que cada uno de los módulos o segmentos de código funcione apropiadamente.

3.3.3- Descripción

La prueba de Caja Blanca es considerada como uno de los tipos de pruebas más importantes que se le aplican a los programas informáticos, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Para el desarrollo de estas pruebas unitarias se utilizó la herramienta PHPUnit como se abordó anteriormente, ya que ofrece las funcionalidades necesarias para implementar pruebas en un proyecto desarrollado en PHP.

Mediante esta herramienta se analizaron algunos códigos de los procesos o Casos de Uso Más críticos dentro del sistema y se interpretaron las pruebas inmersas en ellos, ya que utiliza atributos personalizados para interpretar las pruebas y provee además métodos para implementarlas. En general, PHPUnit compara valores esperados y valores generados, si estos son diferentes la prueba es fallida, en caso contrario la prueba es exitosa.

Para realización de cada una de las pruebas de caja blanca, se incluyó en el proyecto una clase que heredase de la clase `"PHPUnit_Framework_TestCase"` perteneciente al framework. Al correr dicha clase al prueba sería ejecutada.

PHPUnit nos devuelve como resultado el tiempo que demoró la prueba y si esta falló o exitosa, y sin involucrarse con el código de la aplicación; lo que hace posible que a la hora de desplegar el sistema las pruebas sean eliminadas sin dificultad.

---

[20] **PHPUnit** es un miembro de la familia de *frameworks* de testeo **xUnit**, y provee tanto el marco de trabajo en el que se pueden escribir pruebas con facilidad, como la funcionalidad para correr fácilmente estas pruebas y analizar los resultados.

3.3.4- Métrica de la complejidad ciclomática

Se le realizó el cálculo de la complejidad ciclomática en todos los bloques de código dentro del sistema, lo cual ayudó a reflejar una medida de la complejidad del código escrito, teniendo en cuenta el número de destinos posibles. Además permitió conocer el esfuerzo a realizar en cada una de las pruebas, el cual es exactamente el valor de la complejidad ciclomática en cada elemento o módulo. A partir de esta medida, se diseñaron pruebas que forzaron el recorrido de estos caminos, lo cual garantiza que se ejecute al menos una vez cada sentencia del programa y que cada condición se ejecute en sus variantes verdadera y falsa.

Ejemplo de segmentos de código analizados:

```
function formEditarUsuario ($id=null) {
    $this->set_response("view");
    $this->user = $this->Usuario->find($id);
    $this->roles = $this->Rol->find();
    $this->territorios = $this->Territorio->getProvincias();
    $this->rol_actual = $this->Rol->find_by_id($this->user->rol_id);
    $this->territorio_actual = $this->Territorio->find($this->user->territorio_id);
    $this->direcciones = $this->Estructura->getDirecciones();
    $this->vicepresidencias = $this->Estructura->getVicepresidencias();
    $this->checked = array();
    foreach($this->direcciones as $d){
        if ($this->UsuarioDireccion->exists("direccion_id= $d->id"))
            $this->checked[$d->id] = true;
        else
           $this->checked[$d->id] = false;
    }
}
```

Figura 11: Método de formulario para editar un usuario.

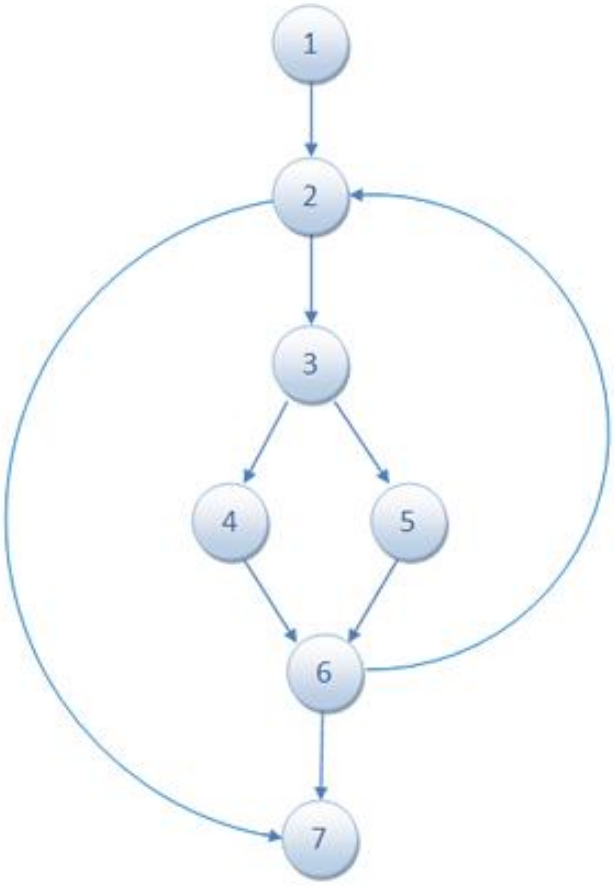


Figura 12: Grafo de flujo para el método de formulario de editar usuarios.

Complejidad ciclomática *V(G)*

V(G) = N. de aristas - N. de nodos + 2

V(G) = 4

Caminos independientes:

- 1-2-7
- 1-2-3-4-6-7
- 1-2-3-5-6-7
- 1-2-3-4-6-2-7
- 1-2-3-5-6-2-7

```
function editarUsuario_Action () {
    $user = new Usuario();
    $user = $this->Usuario->find($this->user->id);
    $user->nombres = $this->post("nombres");
    $user->apellidos = $this->post("apellidos");
    $user->email = $this->post("email");
    $user->username = $this->post("username");
    $user->territorio_id = $this->post("territorio");
    $user->rol_id = $this->post("rol");
    $user->update();
    foreach($this->direcciones as $d){
       if(!$_POST[$d->id]){
          if($this->UsuarioDireccion->exists("direccion_id= $d->id")){
            $this->UsuarioDireccion->find_by_usuario_id($user->id)->delete();
          }
       } elseif($_POST[$d->id]){
        if(!$this->UsuarioDireccion->exists("direccion_id= $d->id")){
            $user_dir = new UsuarioDireccion();
            $user_dir->usuario_id = $user->id;
            $user_dir->direccion_id = $d->id;
            $user_dir->save();
        }
       }
    }

    $this->redirect("administracion/gestionarUsuario");
}
```

Figura 13: Método para editar usuario.

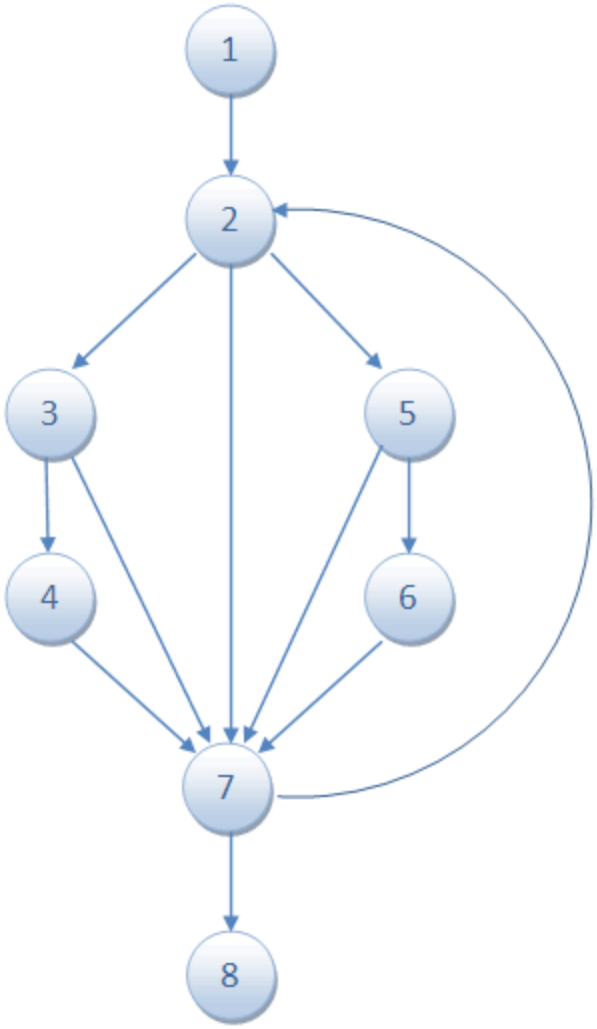


Figura 14: Grafo de flujo para el método de editar usuarios.

Complejidad ciclomática *V(G)*

V(G) = N. de aristas - N. de nodos + 2

V(G) = 6

Caminos independientes:

- 1-2-7-8
- 1-2-3-7-8
- 1-2-3-4-7-8
- 1-2-5-7-8
- 1-2-5-6-7-8
- 1-2-3-7-2-7-8
- 1-2-3-4-7-2-7-8
- 1-2-5-7-2-7-8
- 1-2-5-6-7-2-7-8

3.4- Pruebas de velocidad de descarga

La velocidad de descarga de un portal es algo sumamente significativo: es en la respuesta final al usuario donde más tiempo se emplea en la petición. Cerca de un 80 porciento del tiempo de la solicitud se gasta en esta etapa de la petición (Souders, 2007).

Además, hay varias razones para mejorar el rendimiento de nuestra página web:

1. Mejorar el rendimiento basándose en la interfaz es la mejor forma de que el portal mejore en la velocidad de descarga. Esto reduce el tiempo de respuesta en un 60% o menos; mientras que ocuparse en optimizar la programación del lado del servidor llegará a reducir cuando más un 10% del tiempo de respuesta.
2. Los perfeccionamientos en la interfaz requieren menos tiempo y recursos que los perfeccionamientos en el código del lado del servidor, pues se necesita a veces rediseñar la arquitectura y el código, encontrar y optimizar sentencias o hacer cambios en la base de datos.
3. Los afinamientos en el rendimiento de la interfaz ya han sido comprobados por muchas compañías desarrolladores de sistemas para la Web.

Para mejorar el rendimiento de nuestro sistema, nos basamos en los siguientes puntos:

1. **Minimizar las peticiones HTTP.**

   Las peticiones HTTP se minimizaron haciendo uso de los mapas de imágenes, que son tratados luego con CSS. Además se incluyen todos los scripts en un solo archivo. Esto incrementa la velocidad de descarga ya que se hacen menos peticiones al servidor.

2. **Añadir fecha de expiración para los recursos en la cabecera del HTTP.**

   Al utilizar las fechas de expiración, se hace que los recursos (imágenes, scripts, flash, hojas de estilo, etc.) se carguen desde la caché del navegador del usuario, así no se pedirán de nuevo los recursos al servidor. Esto redunda grandemente en la velocidad de descarga ya que sólo se cargará el contenido dinámico.

3. **Utilizar los componentes Gzip.**

Además, a través del uso de los componentes de compresión se comprimieron los recursos para que pesaran menos, luego el navegador cliente es el encargado de descomprimirlos y mostrarlos correctamente.

4. **Poner las Hojas de Estilo al principio.**

   Para lograr que la página se vaya cargando progresivamente, y a su vez se visualice correctamente se deben poner las Hojas de Estilo en la cabecera del documento HTML.

5. **Poner los Scripts al final.**

   Por otra parte, los scripts deben ir al final, pues casi siempre lo que definen estos es el comportamiento de la página y la interacción con el usuario; con esto se logra que la página se visualice rápidamente y luego se carguen los archivos JavaScripts que casi siempre pesan un poco más.

6. **Evitar las expresiones CSS.**

   Se evitaron las expresiones CSS para no hacer una sobrecarga de ejecución de código en el cliente.

7. **Construir los JavaScript y las Hojas de Estilo externas.**

   El código CSS y JavaScript es cargado independientemente de la página HTML; logrando que se carguen sólo una vez tratándolos con la expiración de cabeceras.

8. **Minimizar el JavaScript.**

   El JavaScript fue comprimido con ShrinkSafe de Dojo Toolkit.

9. **Remover el JavaScript duplicado.**

   Se creó una función en PHP para que no se cargasen JavaScripts que ya están cargados en el cliente.

Después de ejecutados los puntos anteriores, se obtuvieron notables resultados en cuando a rendimiento de la aplicación, consiguiéndose un 37% de mejoras en tiempo de respuesta y un 60% de mejoras en peso de la descarga total, repercutiendo directamente en la calidad de la solución informática.

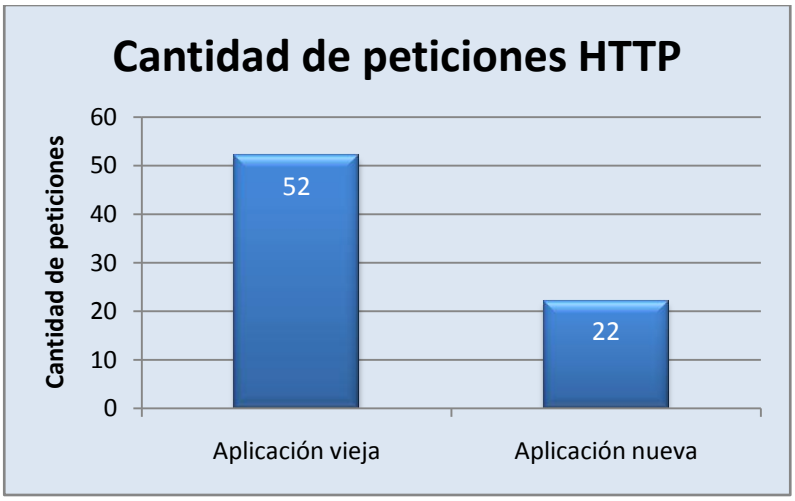


Figura 15: Cantidad de peticiones HTTP.

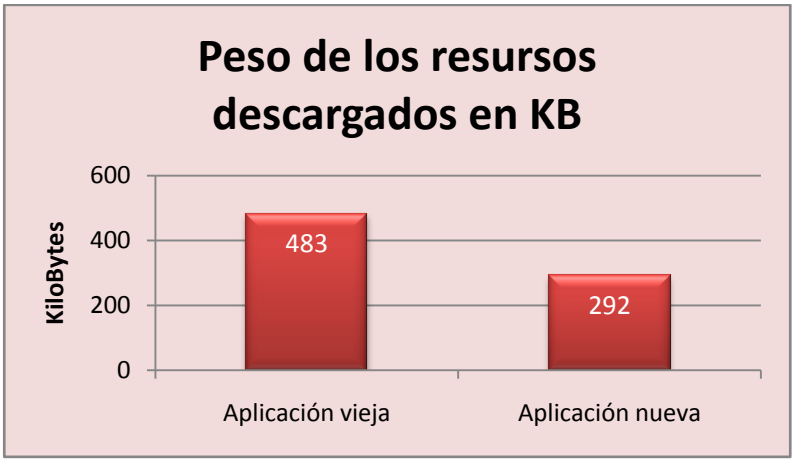


Figura 16: Peso de los recursos descargados en KB.

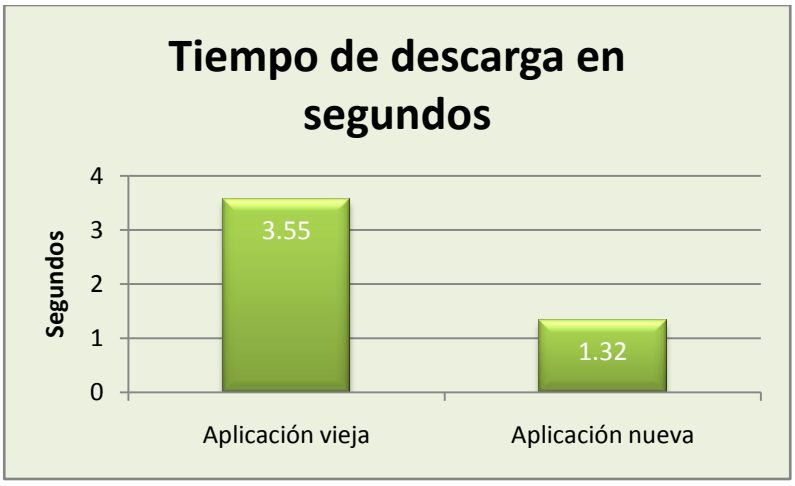


Figura 17: Tiempo de descarga en segundos.

3.5- Conclusiones parciales del capítulo

Se abordaron en este capítulo las validaciones hechas al sistema de información para comprobar su correcto funcionamiento, tanto las pruebas de caja negra, las pruebas de caja blanca, y pruebas de velocidad de descarga en el cliente; por la importancia que lleva el rendimiento de la fachada de la aplicación.

CONCLUSIONES GENERALES

Con la realización de este trabajo se implementó un sistema de información que logra mayor eficiencia en el proceso de toma de decisiones del cierre de operaciones en las áreas postal y de tecnologías de la ECC. Como complemento y añadidura a este objetivo principal, se dio cumplimiento a otros objetivos más específicos:

1. Se profundizó en los conceptos de manipulación de la información.
2. Se demostró que los sistemas que existen en el ámbito internacional no son los más óptimos para modelar el proceso de cierre de operaciones dentro de la empresa Correos de Cuba.
3. La investigación además arrojó como resultado que los sistemas con los que cuenta la ECC no son lo suficientemente potentes como para facilitar una toma de decisiones los más real posible.
4. Se obtuvo un sistema capaz de recibir, procesar y mostrar información. El mismo permite una mejor manipulación de la información y una agilización del proceso de cierre.
5. Los trabajadores de la empresa Correos de Cuba tienen ahora un nuevo sistema que les permite realizar su trabajo de forma sencilla y dinámica, por tanto la eficiencia en el trabajo se elevará significativamente.
6. Se eliminó la duplicación de la información en le proceso de Cierre de Operaciones.
7. El sistema ha sido desarrollado totalmente con Software Libre, por lo que no es necesaria la adquisición de licencias para el uso de este producto en la entidad.

RECOMENDACIONES

1. Extender el sistema para que domine todas las áreas de la ECC.
2. El sistema, por su calidad de programa basado sobre la plataforma Web, puede ser incluido sin ningún inconveniente dentro del ERP Postal, plataforma en desarrollo actualmente.
3. Especializar el producto en un sistema de captura de datos. Por sus facilidades a la hora de capturar lo datos, recomendamos que el sistema se especialice en esta rama, y dejar los reportes, gráfico, tablas, etc. para que sean tratados a través de un reporteador u otra herramienta parecida que cree informes y reportes con mayor flexibilidad y potencia.
4. Implementar Servicios Web para la comunicación automática con otros sistemas de la empresa. Los sistemas de la ECC deberán implementar un conjunto de servicios web para la comunicación óptima con el sistema. Este es además un punto clave para la creación del ERP postal.

BIBLIOGRAFÍA

Achour, Mehdi, y otros. 2007. *PHP Manual.* Los Ángeles : PHP Documentation Group, 2007.

Apache, Software Foundation. 2007. The Apache Software Foundation. [En línea] 2007. [Citado el: 8 de Abril de 2008.] http://www.apache.org/.

Barret, N. 1999. *El Estado del la Cibernación, Consecuencias Culturales, Políticas y Económicas de Internet.* Madrid : s.n., 1999.

Buschamann, F. 1996. *Pattern-Oriented Software Architecture.* s.l. : John Wiley & Sons, 1996. ISBN 0471958697.

Eclipse. 2000. Eclipse.org. [En línea] 2000. [Citado el: 2008 de Abril de 9.] http://www.eclipse.org/.

Eguíluz Pérez, Javier. 2007. *Introducción a CSS.* Madrid : s.n., 2007.

—. 2007. *Introducción a JavaScript.* Madrid : s.n., 2007.

Haag, Cummings y McCubbrey Pinsonneault, Donovan. 2000. *Management Information Systems: For The Information Age.* Ryerson : McGraw-Hill, 2000. ISBN: 0-072-81947-2.

I.O., Angell y S., Smithson. 1991. *Information Systems Management: Opportunities and Risks.* 1991.

Kaplan, Robert S. y Norton, David P. 2000. *The Strategy-focused organization.* Boston : Business School Press Harvard, 2000.

Kumbia Framework. 2007. Kumbia PHP Framework. [En línea] 12 de Noviembre de 2007. [Citado el: 7 de Abril de 2008.] http://www.assembla.com/wiki/show/kumbia/15-Apartir-dun-modelo-MVC_.

Langefors, Börje. 1973. *Theoretical Analysis of Information Systems.* s.l. : Auerbach, 1973.

León, Rolando Alfredo Hernández y González, Sayda Coello. 2002. *El paradigma cuantitativo de la investigación científica.* Ciudad de la Habana : EDUNIV, 2002. ISBN: 959-16-0343-6.

Microsoft. 2003. Microsoft .NET Framework. [En línea] 22 de 4 de 2003. [Citado el: 8 de Abril de 2008.] http://www.microsoft.com/net/Overview.aspx.

Molina, Francisco Javier Pavón. 2006. *Principles of Programming.* Honolulu : Atlantic International University, 2006.

MySQL AB. 2008. *MySQL 5.0 Reference Manual.* 2008.

Network-Press.Org. 2003. Network-Press.Org. [En línea] 2003. [Citado el: 9 de Abril de 2008.] http://www.network-press.org/?navegador_concepto.

Pérez Subirats, Jorge Luis. 2003. *Diseño informacional de los sitios web.* Ciudad de la Habana, Ciudad de la Habana, Cuba : s.n., 21 de Septiembre de 2003.

Perl. 2008. Perl.org. [En línea] 22 de Febrero de 2008. [Citado el: 8 de Abril de 2008.] http://perldoc.perl.org/perlintro.html.

Power, D. J. 2002. *Decision support systems: concepts and resources for managers.* Westport, Conneticut : Quorum Books, 2002.

Projects Subversion. 2001. Tigris.org. [En línea] 2001. [Citado el: 9 de Abril de 2008.] http://subversion.tigris.org/.

Souders, Steve. 2007. *High Performance Web Sites.* Palo Alto : O'Reilly, 2007.

Sprague, R. H. y Carlson, E. D. 1982. *Building effective decision support systems.* Englewood Cliffs, N.J. : Prentice-Hall, 1982. ISBN: 0-130-86215-0.

Stallman, Richard. 2004. *Software Libre para una sociedad libre.* Madrid : Traficantes de Sueños, 2004.

The original MVC reports. **Reenskaug, Trygve. 2007.** Oslo : s.n., 2007.

Turban y Aronson. 2001. *Decision Support Systems and Intelligent Systems.* Upper Saddle River, N. J. : Prentice-Hall, 2001.

Turban, E. 1995. *Decision support and expert systems: management support systems.* Englewood Cliffs, N.J. : Prentice Hall, 1995. ISBN: 0-024-21702-6.

Turban, McLean y Wetherbe. 2002. *Information Technology for Management.* Massachusetts : Wiley, 2002.

Wikipedia, Colaborades de . 2008. Wikipedia. [En línea] 31 de Marzo de 2008. [Citado el: 9 de Abril de 2008.] http://es.wikipedia.org/wiki/SGBD.

Wikipedia, Colaboradores de. 2008. Wikipedia. [En línea] 16 de Marzo de 2008. [Citado el: 9 de Abril de 2008.] http://es.wikipedia.org/wiki/Cliente-servidor.

—. 2007. Wikipedia. [En línea] 2007. [Citado el: 8 de Abril de 2008.] http://es.wikipedia.org/wiki/Internet_Information_Services.