UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE ENSEÑANZA ASISTIDA POR COMPUTADOR - CENEAC

SISTEMA DE GESTIÓN DE PUNTOS DE COMPROMISO Y PUNTOS DE ACEPTACIÓN PARA LOS BANCOS AFILIADOS A LA CORPORACIÓN SUICHE 7B

Trabajo Especial de Grado presentado ante la Ilustre Universidad Central de Venezuela por el bachiller Gustavo Adolfo Alvarez De Oliveira C.I. 17.478.076 Para optar al título de Licenciado en Computación

Tutora:
Profa. Yusneyi Carballo Barrera

Caracas, 28/10/2010

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO I
ÍNDICE DE FIGURAS III
ÍNDICE DE TABLAS IV
AGRADECIMIENTOS Y DEDICATORIAS 1
RESUMEN 2
INTRODUCCIÓN 3
GLOSARIO 5
1 PROBLEMA DE INVESTIGACIÓN 7
1.1 TÍTULO 7
1.2 SITUACIÓN ACTUAL 7
1.3 SOLUCIÓN PROPUESTA 8
1.4 OBJETIVOS 8
1.4.1 Objetivo General 8
1.4.2 Objetivos Específicos 9
1.5 ALCANCE 10
1.6 IMPORTANCIA Y JUSTIFICACIÓN 11
2 MARCO CONCEPTUAL 12
2.1 APLICACIONES WEB 13
2.1.1 Aplicación Web 14
2.1.2 Cliente Web 14
2.1.3 Servidor 15
2.1.4 Arquitectura Cliente-Servidor 16
2.1.4.1 Características de la Arquitectura Cliente – Servidor 18
2.1.5 Arquitectura de Tres Capas 20
2.1.6 Modelo-Vista-Controlador (MVC) 21
2.1.6.1 Ventajas de MVC 24
2.2 TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES WEB 25
2.2.1 Tecnologías del Lado del Cliente 26
2.2.1.1 HTML (HyperText Markup Language) 26
2.2.1.2 DHTML (Dynamic HyperText Markup Language) 27
2.2.1.3 CSS (Cascading Style Sheets) 28
2.2.1.4 JavaScript 31
2.2.1.5 DOM (Document Object Model) 33
2.2.1.6 AJAX (Asynchronous JavaScript And XML) 33
2.2.2 Tecnologías del Lado del Servidor 37
2.2.2.1 Servlet 38
2.2.2.2 JSP (Java Server Pages) 39
2.2.2.3 Apache Tomcat 6.0 42
2.2.3 Sistemas Manejadores de Bases de Datos (SMBD) 44
2.2.3.1 Funciones de un SMBD 44
2.2.3.2 IBM DB2 UDB 8.0 46

2.2.4 Resumen de Tecnologías Usadas en la Aplicación Desarrollada 49
2.2.5 Patrones de Diseño 50
2.2.5.1 Composite View 50

3 MARCO APLICATIVO 57

3.1 PROCESO DE DESARROLLO XP (PROGRAMACIÓN EXTREMA) 58
3.1.1 Características de XP 58
3.1.2 Actividades de la Programación Extrema 60
3.1.2.1 Planificación 60
3.1.2.2 Diseño 61
3.1.2.3 Codificación 61
3.1.2.4 Pruebas 62
3.2 IMPLEMENTACIÓN 63
3.2.1 Planificación de Iteraciones 63
3.2.1.1 Iteración I 63
3.3 BASE DE DATOS DEL SISTEMA 70
3.3.1 Tablas de la Base de Datos del Sistema 72
3.4 ARQUITECTURA FUNCIONAL DEL SISTEMA 75
3.4.1 Vistas o Interfaces 75
3.4.2 Paquete de Controladores 76
3.4.3 Paquete de Acceso al Modelo 76
3.4.4 Paquete de Estructuras de Datos 77
3.4.5 Paquete de Utilidades del Sistema 77
3.5 PRINCIPALES FUNCIONALIDADES DEL SISTEMA 78
3.5.1 Sub-módulo Registrar Usuario 80
3.5.2 Sub-módulo Gestionar Usuario 81
3.5.3 Sub -módulo Crear PDC 83
3.5.4 Sub-Módulo Gestionar PDC 88
3.5.5 Módulo de Reportes 90
3.5.6 Módulo Bitácora 95
3.5.7 Sub-módulo de Consulta MCC 95
3.5.8 Sub-módulo Publicar Noticia 98

CONCLUSIONES Y RECOMENDACIONES 100

REFERENCIAS BIBLIOGRÁFICAS ..... - 102 -

ÍNDICE DE FIGURAS

Figura 1 Arquitectura Cliente - Servidor 17
Figura 2 Arquitectura de Tres Capas 21
Figura 3 Modelo Vista Controlador 22
Figura 4 Diagrama de secuencia del patrón MVC 23
Figura 5 Modelo Web tradicional 34
Figura 6 Modelo Web Tradicional con AJAX 36
Figura 7 Estructura estática del Composite View 52
Figura 8 Diagrama de secuencia del Composite View 53
Figura 9 Actividades de XP 59
Figura 10 Formato de Pruebas de Aceptación 62
Figura 11 Diagrama WAE - Funcionalidad Crear PDC 66
Figura 12 Interfaz Crear PDC 67
Figura 13 Verificación de existencia del PDC en el sistema y 68
Figura 14 Creación de PDC y registro en el sistema 68
Figura 15 Envío de correo y registro de PDAs asociados al PDC 68
Figura 16 Evaluación de la Iteración I 70
Figura 17 Diagrama Entidad Relación - Base de Datos SIPCA 71
Figura 18 Interfaz de Sub-módulo Registrar Usuario 80
Figura 19 Interfaz de Sub-módulo Gestionar Usuario 81
Figura 20 Interfaz de Funcionalidad de Búsqueda de Usuario 82
Figura 21 Interfaz de Funcionalidad Modificar Usuario 82
Figura 22 Interfaz de Crear PDC 84
Figura 23 Interfaz de Funcionalidad Agregar PDA I 85
Figura 24 Interfaz de Funcionalidad Agregar PDA II 86
Figura 25 PDC preparado para creación 87
Figura 26 Interfaz Sub-módulo Gestionar PDC 88
Figura 27 Interfaz de Funcionalidad Buscar PDC 89
Figura 28 Interfaz de Funcionalidad Modificar PDC 90
Figura 29 Interfaz de Módulo de Reportes 91
Figura 30 Interfaz de Funcionalidad Desplegar Reporte 92
Figura 31 Interfaz de Funcionalidad de Exportar Reporte 92
Figura 32 Interfaz de Funcionalidad Desplegar Gráfico 94
Figura 33 Interfaz del Módulo Bitácora 95
Figura 34 Interfaz de Módulo Consulta MCC 96
Figura 35 Interfaz de Sub-módulo Publicar Noticia 98
Figura 36 Página Principal del Sistema con las noticias publicadas 99

ÍNDICE DE TABLAS

Tabla 1 Diferencias entre Java y JavaScript 32
Tabla 2 Historia de Usuario 60
Tabla 3 Historia de Usuario Nº 1 64
Tabla 4 Historia de Usuario Nº 2 64
Tabla 5 Historia de Usuario Nº 3 65
Tabla 6 Pruebas Unitarias Iteración I – Crear PDC 69
Tabla 7 Tabla Banco – Base de Datos SIPCA 72
Tabla 8 Tabla Estado - Base de Datos SIPCA 72
Tabla 9 Tabla PDA - Base de Datos SIPCA 72
Tabla 10 Tabla PDC - Base de Datos SIPCA 73
Tabla 11 Tabla Usuario - Base de Datos SIPCA 73
Tabla 12 Tabla TipoUsuario - Base de Datos SIPCA 74
Tabla 13 Tabla Log_Evento - Base de Datos SIPCA 74
Tabla 14 Tabla MCC - Base de Datos SIPCA 74
Tabla 15 Tabla ATM - Base de Datos SIPCA 74
Tabla 16 Funcionalidades del Sistema y Permisología de Acceso por tipo de Usuario 79
Tabla 17 Muestra de lista de MCC 96

AGRADECIMIENTOS Y DEDICATORIAS

Antes que nada quiero agradecer a Dios por darme la vida y por brindarme salud todos estos años. A mi madre por representar mi modelo a seguir al ser un ejemplo constante de lucha, perseverancia y sacrificio. Una mujer que ha dado todo por convertir a sus hijos en ciudadanos íntegros para la sociedad y exitosos a nivel profesional, que siempre ha inculcado valores y principios morales sólidos e inquebrantables que me han llevado por el camino correcto y por la senda del éxito. A mi familia por siempre haber creído en mí y por apoyarme cuando lo necesité.

Le estoy agradecido a todos y cada uno de los compañeros con los cuales compartí a lo largo de la carrera, fueron incontables tardes estudiando, discutiendo, charlando, haciendo deporte, etc. En especial a Carlos Izquierdo quién fue mi compañero en numerosos numerosas evaluaciones, siempre mostrando gran seriedad y responsabilidad, así mismo estuvo ahí para ayudarme y apoyarme con su pedagogía y conocimiento en una etapa de la carrera muy difícil para mí.

También agradezco a Marilyn Moreira, brillante estudiante, profesora, amiga y compañera sentimental que durante los últimos 4 años ha servido de inspiración y de modelo académico, ha fungido como esa vocecita que siempre me ha apoyado e impulsado cuando he sentido que las dificultades me agobian o que el mundo se me viene encima, ha aguantado mis rabias, molestias y mi fuerte carácter y con la que he compartido los momentos más felices de mi vida hasta la fecha, momentos maravillosos que recordaré toda la vida .

Quiero agradecer a cada uno de los profesores que me brindaron todos sus múltiples conocimientos, en especial a la profesora y tutora de este trabajo especial de grado Yusneyi Carballo, una persona dedicada y responsable que siempre mostró la mayor disposición y diligencia durante la elaboración del Seminario y de este Trabajo Especial de Grado.

De igual manera agradezco a la Corporación Suiche 7B por creer en mi y en mis habilidades al encomendarme el desarrollo de este sistema de tanta importancia para sus clientes.

Gracias a todos de verdad, este logro también es de ustedes.

Universidad Central de Venezuela.
Facultad de Ciencias
Escuela de Computación

Sistema de Gestión de Puntos de Compromiso y Puntos de Aceptación para los Bancos afiliados a la Corporación Suiche 7B

Autor: Gustavo Adolfo Alvarez De Oliveira
Tutora: Profa. Yusneyi Carballo Barrera.
Fecha: 28 de Octubre de 2010.

RESUMEN

El presente trabajo expone fundamentos conceptuales, metodológicos y de implementación que resumen la investigación del Trabajo Especial de Grado realizado con el objetivo de desarrollar una aplicación Web para la Corporación Suiche 7B C.A. que permita el intercambio entre las diferentes instituciones bancarias de información asociada a posibles fraudes en operaciones con tarjetas de crédito y débito. La aplicación en cuestión permitirá la detección de los Puntos de Compromiso (PDC) y Puntos de Aceptación (PDA) utilizados en acciones de fraude, como parte de una nueva estrategia de cooperación interbancaria que ayude a combatir la copia y uso ilegal de tarjetas de débito y crédito a los usuarios de las instituciones bancarias del país. Para alcanzar estos objetivos se aplicó una metodología basado en la adaptación de XP (Programación Extrema), detallando las fases de Planificación, Diseño, Codificación y Pruebas para cada una de las iteraciones que se dividió el desarrollo de la aplicación Web. Se obtuvo como resultado una aplicación Web para el manejo del personal de las diferentes comisiones de Fraude de las instituciones bancarias así como para el personal de la Gerencia de Operaciones de la Corporación Suiche 7B, convirtiéndose así, en la primera aplicación en el país para tal fin.

Palabras Claves: Sistema de Gestión de Puntos de Compromiso y Puntos de Aceptación (SIPCA), Punto de Compromiso (PDC), Punto de Aceptación (PDA), Clonación, Cajero Automático (ATM), Punto de Venta (POS), Score.

INTRODUCCIÓN

La Corporación Suiche 7B, C.A., se constituye como la primera red de conexión interbancaria del país con la mayor cobertura geográfica y operativa, constituida por 23 bancos del sistema financiero venezolano a los cuales se brinda servicios de interconexión e intercambio de transacciones electrónicas interbancarias y con instituciones relacionadas.

Al inicio de cada año las instituciones bancarias miembros de la Corporación Suiche 7B, solicitan a ésta, a través de comités técnicos, la realización de proyectos enmarcados en la obligación de continua mejora en los servicios, así como suministro de nuevos servicios de valor agregado como lo es el Sistema de Puntos de Compromiso y Aceptación SIPCA.

En el presente trabajo de investigación se enumerarán y analizarán todas las herramientas de desarrollo y planificación necesarias para proveer a las instituciones bancarias de una nueva herramienta que sirva como punta de lanza de la nueva estrategia de cooperación interbancaria en la lucha contra el copiado ilegal de tarjetas de crédito y débito.

El documento que presenta esta propuesta está organizado de la siguiente manera:

Capítulo 1 Problema de Investigación

En éste capítulo se expone la forma en la que se llevan a cabo actualmente los procesos de investigación en las diferentes comisiones de fraude de las instituciones bancarias, se especifica el problema a resolver y la solución más apropiada para el mismo, así como los objetivos del trabajo, su alcance, importancia y justificación del desarrollo, evaluando los beneficios que generará el mismo.

Capítulo 2 Marco Conceptual

En este capítulo se enumeran elementos concernientes al universo de las aplicaciones Web, como trabajan, así mismo diversos aspectos de los clientes Web, servidores y de la arquitectura Cliente – Servidor, también la definición, características generales, entre otros puntos de la arquitectura de tres capas, el patrón de desarrollo MVC y de las tecnologías a usar en el desarrollo del sistema en el lado de cliente y en el lado del servidor.

Por otra parte se describen detalles y características del sistema manejador de base de datos IBM DB2 UDB, con el cual trabajará la aplicación.

Capítulo 3 Marco Aplicativo

Se presenta la adaptación del proceso de desarrollo XP al problema particular de investigación, detallando las fases de Planificación, Diseño, Codificación y Pruebas dentro de cada una de las iteraciones definidas para dicho proceso.

Finalmente se presentan conclusiones respecto al trabajo realizado y se indican las referencias bibliográficas y digitales utilizadas.

GLOSARIO

A continuación se definen una serie de términos claves que facilitarán la comprensión del contenido de este trabajo de investigación.

Corporación Suiche 7B

Es la red interbancaria del país con la mayor cobertura geográfica y operativa, constituida por bancos importantes del sistema financiero venezolano con el objeto de integrarlos en servicios de interconexión e intercambio de transacciones electrónicas e información con otros bancos participantes e instituciones relacionadas, cooperando con ellos en la estandarización, crecimiento de uso de las tarjetas, facilitando el uso compartido de la plataforma de cajeros automáticos instalados en el país, gestionado los procesos operativos interbancarios y contribuyendo en la generación de economías de escalas para las Instituciones Miembros.

Clonación

Copia ilícita y fraudulenta de información sensible y privada de tarjetas de crédito y/o débito sin el consentimiento del tarjeta habiente o de la institución bancaria, empleando para ello dispositivos electrónicos especialmente diseñados.

ATM o Cajero Automático

(*Automated Teller Machine*), Términos en inglés colocar en cursiva máquina expendedora usada para extraer dinero utilizando una tarjeta magnética, sin necesidad de personal del banco.

POS o Punto de Venta

(*Point Of Sale*), sistema informático o electrónico micro computarizado que gestiona el proceso de venta mediante una interfaz accesible para los vendedores y clientes.

MCC o Codigo de Categoría de Comercio

(Merchant Category Code), Código único asignado por Master card a las transacciones realizadas acorde con la naturaleza del comercio donde fue realziada dicha transacción.

PDC o Punto de Compromiso

ATM o POS en el cual se realiza el copiado de la información privada de la tarjeta del tarjeta habiente.

PDA o Punto de Aceptación

ATM o POS en el cual se realizan operaciones con la tarjeta fraudulenta que fue copiada de manera ilícita al tarjeta habiente en un PDC.

SIPCA

(Sistema de Puntos de Compromiso y Aceptación) Herramienta a través de la cual las instituciones bancarias ingresarán y gestionarán información de los PDC y PDA detectados a través del análisis interno de sus transacciones, lo que permite que cada banco se alimente de información procedente de otras instituciones. Entre sus funcionalidades también está el generar reportes y gráficos especializados.

Score

Indicador de Criticidad que se asigna a cada PDC que es ingresado en el sistema SIPCA. Sirve como índice de gravedad, debido a que es constituido tomando en cuenta factores como cantidad de bancos que han reportado el PDC, cantidad de semanas desde que el PDC fue registrado y cantidad de PDAs asociados a dicho PDC, cada uno de ellos con un peso especifico dentro del valor del Score.

1 Problema de Investigación

En éste capítulo se detalla el contexto del problema a solucionar, haciendo énfasis en la descripción de los procesos de investigación de las comisiones de fraude de las instituciones bancarias, y la forma en la que se llevan a cabo actualmente estos procesos. Se especifica el problema particular a resolver y una solución adecuada a la nueva estrategia de cooperación interbancaria en esta materia.

Luego se destaca el objetivo general y los objetivos específicos de este Trabajo Especial de Grado, así como su alcance, la importancia y justificación del mismo, enfocando también los beneficios que genera a la actores involucrados en el problema.

1.1 Título

Sistema de Gestión de Puntos de Compromiso y Puntos de Aceptación para los bancos afiliados a la Corporación Suiche 7B.

1.2 Situación Actual

Mediante el uso de tecnologías basadas en el desarrollo de la electrónica, la informática y de las telecomunicaciones, las tarjetas bancarias, de crédito y de débito han constituido tradicionalmente el objeto central de los ataques a patrimonios de las entidades bancarias y de sus clientes.

Si bien otros fraudes y ataques patrimoniales se han sumado a estas prácticas, el fraude de tarjetas sigue constituyendo una problemática de primer orden.

Las entidades bancarias de Venezuela, así como sus clientes no escapan a esta problemática, si bien es cierto que se han logrado avances en el aspecto legal con la promulgación de la Ley de Tarjetas de Crédito, Débito, Propagandas y demás Tarjetas de Financiamiento y Pago Electrónico del 22 de

Septiembre de 2008. Pero a pesar de ello esta práctica delictiva sigue en aumento debido a una serie de razones de índole tecnológico así como de cooperación interbancaria.

En cuanto al aspecto tecnológico se trata de una carrera constante entre quienes realizan el fraude, los investigadores de las comisiones de fraude de los distintos bancos y los investigadores policiales, donde los primeros constantemente conciben nuevos *modus operandi* y tienen acceso a nuevas tecnologías para la copia ilícita de información de las tarjetas, mientras los investigadores de los bancos intentan reconocer patrones de comportamiento mediante el análisis de información de las transacciones de su propio banco.

Esta forma de trabajo de las distintas comisiones de prevención y detección de fraude en los bancos resulta muy limitada debido a que solo tienen acceso a información propia y casi no existe interacción e intercambio de información entre las instituciones bancarias.

1.3 Solución Propuesta

Por ello, a petición de algunas instituciones bancarias, se le solicita a la Corporación SUICHE 7B, que es la encargada de todas las operaciones de interconexión y enrutamiento interbancario de más de 23 bancos en el país, el desarrollo de una herramienta que permita el ingreso de la información de investigación en un repositorio común de datos, así mismo, la herramienta debe proveer acceso a la información de investigación para consulta y edición , esta herramienta también debe generar reportes especializados que ayuden a las investigaciones.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar una herramienta Web que permita el intercambio entre las diferentes instituciones bancarias de información asociada a posibles fraudes en operaciones con tarjetas de crédito y débito, que permita la detección de los Puntos de Compromiso (PDC) y Puntos de Aceptación (PDA) utilizados en acciones de fraude y que incorpore los lineamientos establecidos por las distintas comisiones de detección y prevención de fraude de las instituciones bancarias pertenecientes a la red SUICHE 7B.

1.4.2 Objetivos Específicos

1. Levantar requerimientos e información correspondiente a la investigación de posibles Puntos de Compromiso y Puntos de Aceptación recopilada en los distintos Comités Técnicos realizados entre los bancos pertenecientes a la red SUICHE 7B.
2. Levantar la información que conformará los diferentes reportes y gráficos de interés para SUICHE 7B.
3. Diseñar e implementación de los diferentes módulos que compondrán la aplicación.
4. Diseñar e implementación de las diferentes estructuras de datos necesarias para el manejo de la información dentro de la aplicación.
5. Diseñar e implementación del repositorio de datos que albergará la información que se ingresara y manejará a través de la aplicación.
6. Construir y aplicar un conjunto de pruebas unitarias y de aceptación que permitan verificar el correcto funcionamiento de la aplicación, adaptándose a los requerimientos previamente identificados.

1.5 Alcance

La aplicación Web tiene el siguiente alcance:

- Desarrollo e Implementación de un Sistema en Línea Disponible para ser utilizado por todos los Bancos Miembros de la Red SUICHE 7B y que cumpla con los estándares de seguridad establecidos.
- Acceso a la aplicación vía Web a través de un USUARIO y CONTRASEÑA en la Intranet de SUICHE 7B y los Bancos.
- Ingreso de los Puntos de Compromisos (PDC) y Puntos de Aceptación (PDA), tanto de ATM como POS, que hayan sido detectados y sean reportados por los bancos miembros para ser compartidos en la Red SUICHE 7B.
- Manejo de indicadores de criticidad de los Puntos de Compromiso (PDC) registrados en el sistema mediante la asignación de un *Score* o puntuación asignado mediante una formula diseñada en conjunto por las comisiones de fraude de los distintos bancos y que está compuesta por tres aspectos de peso a la hora de evaluar cuán crítico puede ser un PDC:

  5 * Cantidad de Bancos que han registrado el PDC en el sistema +

  2 * Cantidad de PDA asociados al PDC +

  3 * Cantidad de Semanas transcurridas desde la fecha de la transacción más antigua de las tarjetas registradas que representan los Puntos de Aceptación hasta la fecha actual.
- Notificación vía correo electrónico de nuevos ingresos o cambios de Puntos de Compromiso (PDC) cuyo Score asociado cumpla ciertos parámetros.
- Generación de reportes, entre los cuales se encuentran:
  - PDC ATM con mayor Score.
  - PDC POS con mayor Score.
- Generación de gráficos de cantidad y proporción, entre los cuales se encuentran:
  - Cantidad de PDC por tipo de comercio.
  - Cantidad de PDA por tipo de comercio.
  - Tipos de tarjeta comprometidas.
  - PDC por estado del país.
  - PDA por estado del país.

  - PDA por horas de transacción.
  - PDC por días de la Semana.
  - PDA por días de la semana de Transacciones.
  - PDC por Banco.
  - PDA por Banco.
  - Transacciones con tarjeta de Crédito por Tipo de Comercio.
  - Transacciones con tarjeta de Débito por Tipo de Comercio.
- Registro en base de datos de las acciones realizadas por los usuarios en los diferentes módulos de la aplicación a los efectos de posibles auditorias futuras.
- Registro de Logs de errores que facilitan la detección y diagnóstico de los mismos en caso de ocurrir.
- Directorio de usuarios para facilitar la comunicación entre personal de diferentes instituciones bancarias.
- Consulta de la lista actualizada y traducida al español de los MCC (Merchant Category Code) asociados a los diferentes tipos de comercios.

1.6 Importancia y Justificación

La Corporación Suiche 7B como líder en el área de operaciones de interconexión y enrutamiento interbancario, promueve el desarrollo de la herramienta estudiada en este trabajo de investigación como parte de una nueva estrategia de cooperación interbancaria que ayude a combatir el flagelo que representa la copia y uso ilegal de tarjetas de débito y crédito a los usuarios de las instituciones bancarias del país.

2 Marco Conceptual

Este capítulo se presentan los fundamentos conceptuales del desarrollo realizado en el presente trabajo, y se encuentra dividido en dos secciones fundamentales. La primera parte corresponde a la definición de aplicaciones Web, características, ventajas y desventajas, entre otros. Luego se explica la estructura de la arquitectura para dichas aplicaciones, basadas en un modelo Cliente-Servidor, haciendo énfasis en lo más importante de la misma; finalmente se especificará la arquitectura Web de tres capas, donde se expone el patrón de arquitectura MVC.

Una vez dado a conocer los aspectos de las aplicaciones Web y su arquitectura. La segunda parte de este capítulo describe las tecnologías del lado del cliente como HTML (*HyperText Markup Language*), JavaScript, CSS (*Cascading Style Sheets*), DOM (*Document Object Model*), XML (*eXtensible Markup Language*) y AJAX (*Asynchronous JavaScript And XML*). ; así como también conoceremos las características y ventajas de las tecnologías del lado del servidor, enfocándonos específicamente en Servlets y JSP (*Java Server Pages)*. como las tecnologías escogidas para el desarrollo. Ademas presentando información acerca del servidor de aplicaciones Apache Tomcat, tomando en cuenta su arquitectura y funcionamiento, para luego pasar a estudiar las características, ventajas y desventajas, entre otros aspectos, del sistema manejador de bases de datos IBM DB2 UDB 8.0.

Finalmente se presenta un resumen de las tecnologías del lado del cliente usadas en este trabajo de investigación.

2.1 Aplicaciones Web

Con la llegada de Internet y de la Web en concreto, se han abierto muchas posibilidades en cuanto al acceso a la información desde casi cualquier sitio. Esto representa un desafío a los desarrolladores de aplicaciones, ya que los avances en tecnología demandan cada vez aplicaciones más rápidas, ligeras y robustas que permitan utilizar la Web [1].

Afortunadamente, han surgido nuevas tecnologías que permiten que el acceso a una base de datos desde la Web sea un proceso bastante común. La disponibilidad de tantas opciones tecnológicas ocasiona precisamente dificultad al momento de decidir entre el conjunto de posibilidades cuál es la correcta para cada situación. [1]

El CGI (Common Gateway Interface) ha cumplido con el propósito de añadir operabilidad y dinamismo a las páginas Web pero sus deficiencias en el desarrollo de aplicaciones (cada petición HTTP requiere que el servidor que inicie un nuevo proceso, entre otras) y en la escalabilidad de las mismas ha conducido al desarrollo de API (*Application Programming Interface*) específicos de servidor como ASP (*Active Server Pages*) y PHP (*Hypertext Pre-Processor*), que ofrecen más facilidad de uso y mayor integración con los IDES que su predecesor CGI [2].

Para aprovechar el potencial de estas tecnologías y ofertar una solución de servidor más extensible y portable, Sun Microsystems desarrollo la tecnología llamada servlet. Los servlets Java son muy eficientes, debido al esquema de threads en el que se basan y al uso de una arquitectura estándar como la JVM (*Java Virtual Machine*). Aunado a esto crearon una tecnología que viene a sumarse a las funcionalidades ya existentes de los servidores Web llamada JSP (*Java Server Pages*). Los JSP permiten juntar HTML con código Java embebido y componentes como las JavaBeans creando una página Web [2].

2.1.1 Aplicación Web

Una aplicación Web es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una Intranet. Las aplicaciones Web son populares, debido a la practicidad del navegador Web como cliente ligero. La habilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad [3].

En los primeros tiempos de la computación, cada aplicación tenía su propio programa cliente y su interfaz de usuario, estos tenían que ser instalados separadamente en cada estación de trabajo de los usuarios. Una mejora al servidor, como parte de la aplicación, requería típicamente una mejora de los clientes instalados en cada una de las estaciones de trabajo, añadiendo un costo de soporte técnico y disminuyendo la eficiencia del personal [3].

En contraste, las aplicaciones Web generan dinámicamente una serie de páginas en un formato estándar, soportado por navegadores Web comunes como HTML o DHTML. Se utilizan lenguajes interpretados del lado del cliente, para añadir elementos dinámicos a la interfaz de usuario. Generalmente cada página Web individual es enviada al cliente como un documento estático, pero la secuencia de páginas provee de una experiencia interactiva [3].

2.1.2 Cliente Web

El cliente Web es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN (Local Area Network) o WAN (Wide Area Network). La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente [2].

Los Clientes interactúan con el usuario, usualmente en forma gráfica a través de lo que se conoce como Navegadores Web. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad [2].

Los clientes realizan generalmente funciones como:

- Iniciar solicitudes o peticiones, tienen por tanto un papel activo en la comunicación.
- Esperar y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Manejo de la interfaz de usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.

2.1.3 Servidor

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, entre otros [2].

Los Servidores suministran servicios y están a la espera de requerimientos de los clientes. Cuando reciben un requerimiento, buscan la información solicitada y le envía la respuesta al cliente; incluso puede enviar varios servicios a la vez En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además, deben manejar el ínterbloqueos, la recuperación ante fallas y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Por esta razón se utilizan PCs poderosas, estaciones de trabajo, mini computadores o sistemas grandes. Además deben manejar servicios como administración de la red, mensajes, control y administración del sistema, auditoria y recuperación. Usualmente en los servidores existe algún tipo de servicio de bases de datos. En ciertas circunstancias, este término designará a una máquina. Este será el caso si dicha máquina está dedicada a un servicio particular, por ejemplo: servidores de impresión, servidor de archivos, servidor de correo electrónico, entre otros [2].

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.

- Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste, le responde proporcionándole lo que este solicitó.
- Existen diversos tipos de servidores, que se clasifican basándose en su funcionalidad; estos son denominados servidores dedicados ya que administran el uso de algún recurso en particular, por ejemplo: [5]
    - **Servidores de Archivo:** Servidor donde se almacena archivos y aplicaciones de productividad como por ejemplo procesadores de texto, hojas de cálculo, etc.
    - **Servidores de Bases de Datos:** Servidor donde se almacenan las bases de datos, tablas, índices. Es uno de los servidores que más carga tienen.
    - **Servidores de Transacciones:** Servidor que cumple o procesa todas las transacciones. Valida primero y recién genera un pedido al servidor de bases de datos.
    - **Servidores Web:** Sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red
    - **Servidores de Correo:** Mueven y almacenan el correo electrónico a través de las redes corporativas
    - **Servidores de Aplicaciones:** Son aquellos que comunican distintas aplicaciones.
    - **Servidores de Impresoras:** Son aquellos que llevan la administración de un conjunto de impresoras.

Una vez entendido estos dos conceptos podemos decir que la "Arquitectura Cliente - Servidor" consiste en un servidor que proporciona uno o más servicios a uno o más clientes que desea acceder a dicho servicio a través de una petición, generada por el cliente.

2.1.4 Arquitectura Cliente-Servidor

"Es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes" [4].

La arquitectura Cliente - Servidor es un modelo para el desarrollo de aplicaciones, en el que las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos [4].

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura de comunicación, la cual proporciona los mecanismos básicos de direccionamiento y transporte. La mayoría de los sistemas Cliente - Servidor actuales, se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo cual implica que las aplicaciones deben hacer las verificaciones. La red debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración [4].

Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en computadoras personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

En el servidor permanecen las aplicaciones que deben ser compartidas por varios usuarios; incluso, un servidor puede fungir como cliente de otros servidores.

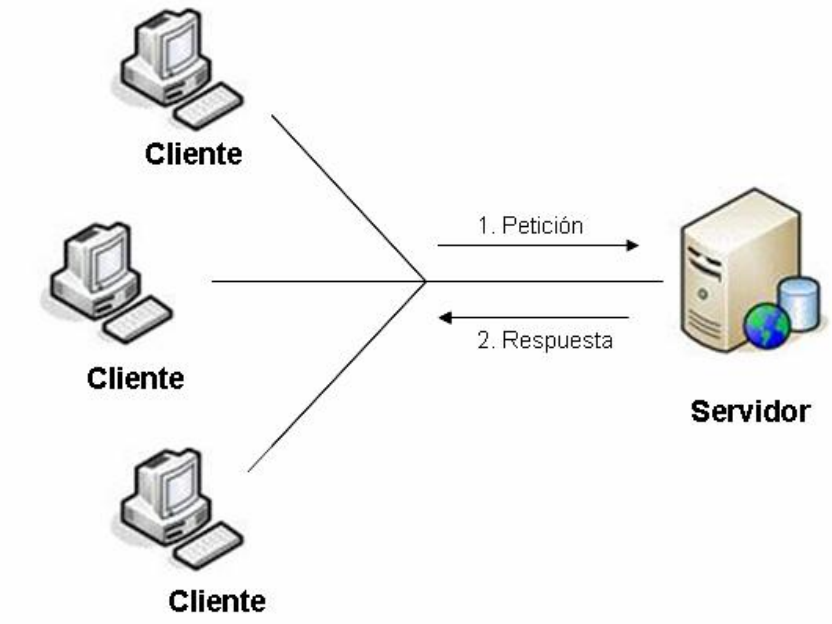

Figura 1 Arquitectura Cliente - Servidor

Como muestra la Figura 1, en esta arquitectura las computadoras de cada uno de los usuarios, llamada Cliente, son las encargadas de activar la comunicación con el servidor, donde inician un proceso

de diálogo; cada uno de estos realiza una acción que produce una petición de alguna información (correo, archivo, imagen, etc.) o solicita recursos (impresora, unidad de disco, etc.). La computadora que recibe la solicitud o petición realizada por los clientes, se conoce como Servidor. Este tiene la potestad de atender dicha petición para dar como resultado algún tipo de respuesta al cliente.

Se puede decir que la arquitectura Cliente - Servidor es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que definidos modularmente en los servidores; estos administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información; estableciendo así un enlace de comunicación transparente entre los elementos que conforman la estructura [4].

2.1.4.1 Características de la Arquitectura Cliente – Servidor

Entre las principales características de la arquitectura cliente – servidor encontramos:

- El cliente y el servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
- Las funciones de cliente y servidor pueden estar en plataformas separadas, o en la misma plataforma.
- Un servidor da servicio a múltiples clientes en forma concurrente.
- Cada plataforma puede ser independientemente escalable. Los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.
- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo necesita su interfaz externa.

- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

2.1.5 Arquitectura de Tres Capas

La arquitectura de tres capas es un diseño que introduce una capa intermedia en el proceso. Cada capa es un proceso separado y bien definido corriendo en plataformas separadas, el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño. Esta arquitectura organiza los elementos de una aplicación Web en: Capa de Presentación, Capa de Negocio y Capa de Datos [6].

Capa de Presentación:

Es la que ve el usuario ("capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio [7].

Capa de negocio:

Es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos [7].

Capa de datos:

Es donde residen los datos. Está formada por uno o más gestor de bases de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio [7].

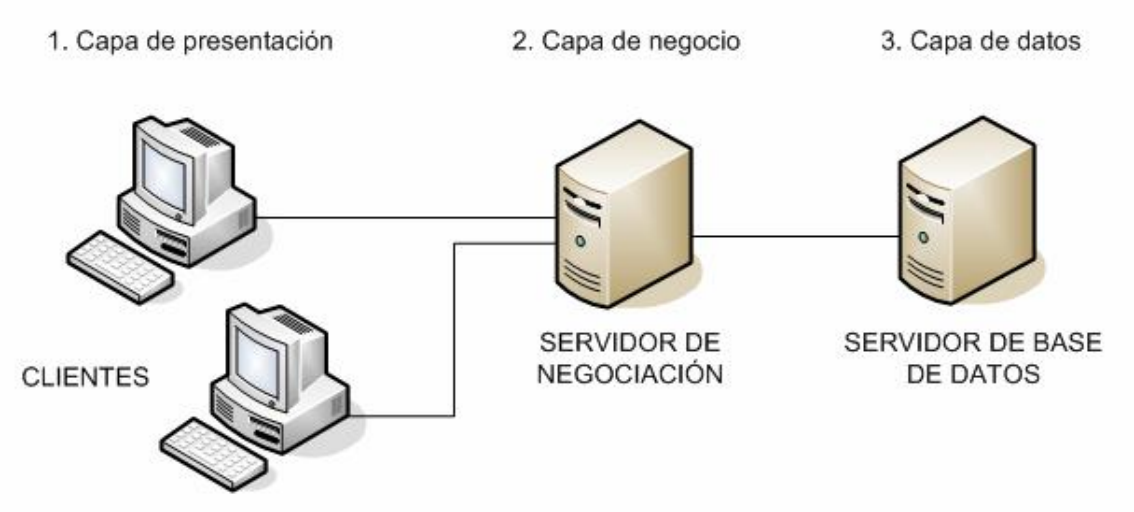

Figura 2 Arquitectura de Tres Capas

En la Figura 2 se muestra cada uno de los componentes de la arquitectura de tres capas, donde la capa de presentación son las estaciones de trabajo (clientes) las cuales son los que inician las peticiones, la capa de negocio (servidor de negociación) es la encargada de procesar la solicitud y la capa de datos (Servidor de Bases de Datos) es la que respalda toda la información o dato de un sistema.

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 que existe entre niveles [8].

En la actualidad el diseño de sistemas informáticos se suele usar la programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten) [8].

2.1.6 Modelo-Vista-Controlador (MVC)

Modelo Vista Controlador (MVC) es un patrón de arquitectura aportado originariamente por el lenguaje SmallTalk a la Ingeniería del Software. El paradigma MVC consiste en dividir las aplicaciones en tres partes: [9]

Modelo

El modelo sería la aplicación que responde a una petición. Es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos [9].

Vista

La vista presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario. Una vez realizadas las operaciones necesarias el flujo vuelve al controlador y este devuelve los resultados a una vista asignada [9].

Controlador

El controlador responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y probablemente en la vista. Es el encargado de redirigir o asignar una aplicación (un modelo) a cada petición; el controlador debe poseer de algún modo, un "mapa" de correspondencias entre peticiones y respuestas (aplicaciones o modelo) que se les asignan [9].

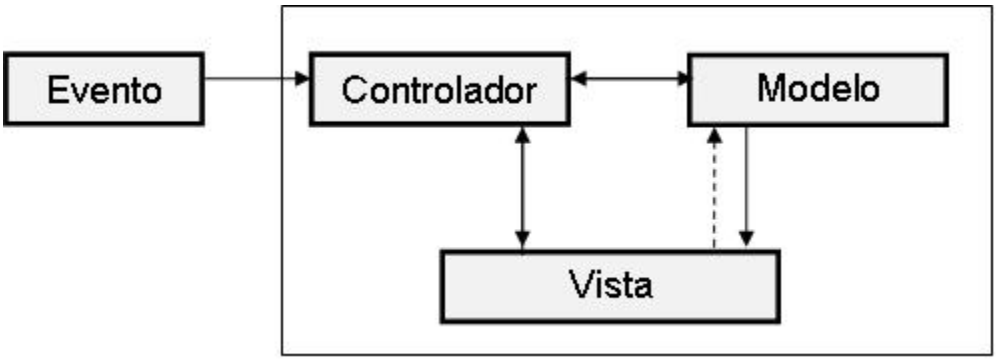

Figura 3 Modelo Vista Controlador

La **¡Error! No se encuentra el origen de la referencia.** muestra como es de manera general la arquitectura MVC. Donde se debe dispara un evento el cual es el controlador que recibe la petición y decide qué hacer, este a su vez invoca al modelo para que procese la solicitud y busque o actualice la información o data; una vez hecho esto el controlador decide a que vista va a redireccionar y el modelo envía la data solicitada a la vista para su representación.

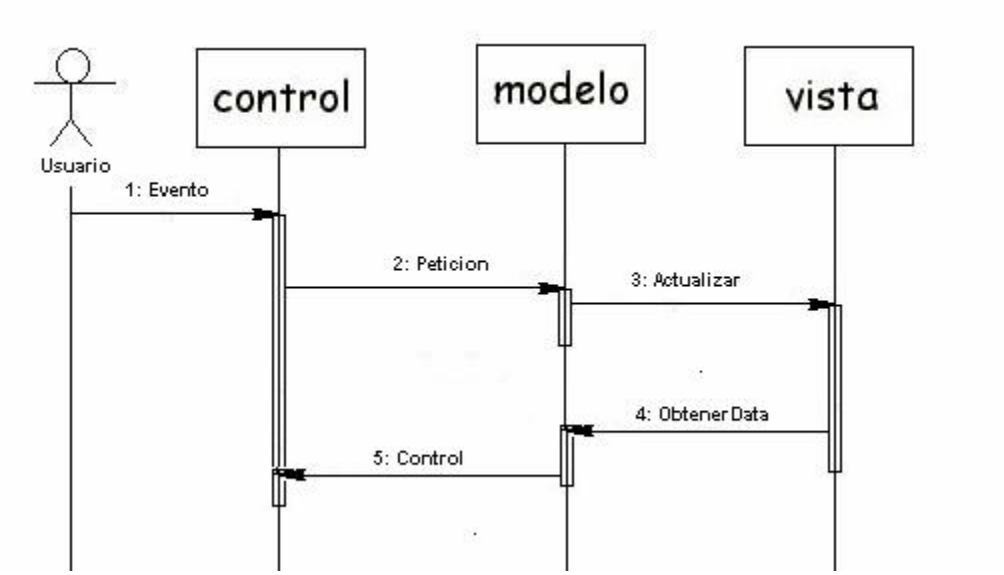

Figura 4 Diagrama de secuencia del patrón MVC
(Fuente: http://sites.google.com/site/flaviodanesse/aprendiendo-java/patrones-de/patron-modelo-vista-controlador)

La Figura 4 representa el diagrama de secuencia de la arquitectura MVC.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control que sigue generalmente es el siguiente: [10]

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo)

a la vista aunque puede dar la orden a la vista para que se actualice. En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

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

2.1.6.1 Ventajas de MVC

Las aplicaciones Web divididas en modelo vista controlador tienen varios beneficios: [10]

1. Menor acoplamiento.
2. Desacopla las vistas de los modelos
3. Desacopla los modelos de la forma en que se muestran e ingresan los datos.
4. Mayor cohesión.
5. Cada elemento del patrón esta altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio).
6. Las vistas proveen mayor flexibilidad y agilidad.
7. Se puede crear múltiples vistas de un modelo
8. Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente
9. Las vistas pueden anidarse
10. Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual
11. Se puede sincronizar las vistas
12. Las vistas pueden concentrarse en diferentes aspectos del modelo.
13. Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales.
14. Una vista para cada dispositivo que puede variar según sus capacidades.
15. Una vista para la Web y otra para aplicaciones de escritorio.
16. Más claridad de diseño.
17. Facilita el mantenimiento.
18. Mayor escalabilidad.

2.2 Tecnologías para el Desarrollo de Aplicaciones Web

El desarrollo de una aplicación Web requiere el uso de un conjunto de tecnologías que permiten diseñar y estructurar el contenido de las páginas, implementar las funcionalidades y el dinamismo de las mismas, alojarlas y ponerlas en funcionamiento para su utilización por parte de los usuarios.

Entre éstas tecnologías tenemos:

- **Tecnologías del Lado del Cliente:** están insertadas en la página HTML del cliente y son interpretadas y ejecutadas por el navegador. Estas tecnologías son utilizadas fundamentalmente para mostrar la información y dar estética al sitio Web. En particular estudiaremos ***HTML, DHTML, CSS, DOM, JavaScript y Ajax.***

- **Tecnologías del Lado del Servidor:** permiten construir código que se ejecuta en el servidor Web, justo antes de que se envíe la página a través de Internet al cliente. Existen diversas tecnologías del lado del servidor, ampliamente utilizadas en el desarrollo de aplicaciones Web, de ésta diversidad se hace énfasis en ***Servlets y JSP***, por ser tecnologías que agregan dinamismo y facilitan la implementación de aplicaciones Web minimizar el tiempo de desarrollo.

- **Servidores de Aplicaciones:** proporcionan servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas. Será descrito el Servidor ***Apache Tomcat*** por ser uno de los servidores de aplicaciones más utilizados.

- **Sistemas Manejadores de Bases de Datos:** consisten en un conjunto de datos relacionados entre sí y un conjunto de herramientas de software (y/o hardware) para tener acceso a dichos datos, y a su vez procesarlos y administrarlos. Se estudiará ***IBM DB2 UDB 8.0*** como sistema manejador de base de datos licenciado por IBM para la Corporación Suiche 7B.

2.2.1 Tecnologías del Lado del Cliente

Las tecnologías del lado del cliente son todas aquellas que son ejecutadas e interpretadas del lado del cliente en una aplicación Web (browser o navegador Web). En muchos casos el buen funcionamiento de estas tecnologías va a depender del tipo de browser y de la versión de cada uno de ellos.

Por lo general estas tecnologías son utilizadas fundamentalmente para mostrar información, para darle formato a esa información, para solicitar datos, etc. Además se pueden generar todo tipo de documento de manera estática como dinámica en un navegador.

2.2.1.1 HTML (HyperText Markup Language)

HTML es el acrónimo de HyperText Markup Language (Lenguaje de Marcado de Hipertexto). Es el lenguaje más utilizado para la presentación de textos estructurados y semi-estructurados en formato hipertexto, estándar de las páginas Web [12].

HTML es utilizado por prácticamente todos los navegadores Web del mercado, con el fin de presentar al visitante de una página Web el contenido de la misma tal como el diseñador quiere que se muestre a su público. HTML es el estándar usado en la World Wide Web, y se ha convertido en uno de los formatos más populares que existen para la construcción de documentos [12].

Este lenguaje nos permite aglutinar textos, sonidos e imágenes y combinarlos a nuestro gusto. Además, y es aquí donde reside su ventaja con respecto a libros o revistas, el HTML nos permite la introducción de referencias a otras páginas por medio de los enlaces hipertexto [12].

HTML es un lenguaje que basa su sintaxis en un elemento de base al que llamamos etiqueta, donde por lo general toda etiqueta representa una instrucción. La etiqueta presenta frecuentemente dos partes: un comienzo y un final, y todo lo incluido en el interior de esa etiqueta sufrirá las modificaciones que caracterizan a esta etiqueta, esto le indica al navegador Web como va a representar esa información, imagen, video, etc., en el browser para ser visualizada por el usuario desde su computador. Por ejemplo: es una etiqueta que indica al navegador que todo lo contenido dentro de esas etiquetas va a estar en negrita y su correspondiente marca de cierre es [13].

Ejemplo:

Codificación HTML: Hola Mundo

Visualización en el browser: Hola Mundo

La estructura básica de un documento HTML es la siguiente:

```
<html>
        <head>
                <title>Titulo de la página</title>
        </head>
        <body>
                Cuerpo de la página
        </body>
</html>
```

Las etiquetas <html> y </html> representan el documento HTML. Las etiquetas <head> y </head> representan el encabezado donde colocaremos etiquetas de índole informativo, como por ejemplo el titulo de la pagina. Las etiquetas <body> y </body> representan el cuerpo de la pagina, que será donde colocaremos nuestro texto, imágenes, etc. delimitados a su vez por otras etiquetas [13].

El lenguaje HTML puede ser creado y editado con cualquier editor de textos básico, como puede ser el Bloc de Notas de Windows (Notepad), o cualquier otro editor que admita texto sin formato como GNU Emacs, Microsoft Wordpad, TextPad, entre otros [13].

2.2.1.2 DHTML (Dynamic HyperText Markup Language)

El HTML Dinámico (DHTML) no es más que, en pocas palabras, una forma que tienen las páginas de aportar interactividad a las mismas. DHTML es una característica de Netscape Communicator 4.0, y Microsoft Explorer 4.0 y posteriores versiones de ambos navegadores, y está orientada al usuario. Es tarea del navegador mostrar y manipular las páginas Web [15].

El DHTML tiene la ventaja de que es una herramienta con la que se pueden crear efectos que requieren poco ancho de banda, a la hora de bajarlos de Internet y, son estos efectos los que aumentan la funcionalidad de la página. Se puede utilizar para crear animaciones, juegos, aplicaciones, para introducir nuevas formas de navegar a través de los sitios Web, y para crear un auténtico entramado de capas que con sólo el HTML sería imposible abordar. Aunque muchas de las características del DHTML se podrían duplicar con otras herramientas como Java o Flash, el DHTML ofrece la ventaja de que no requiere ningún tipo de plug-in para poder utilizarlo [15].

Aunque las tecnologías en las que se basa el DHTML (HTML, CSS y JavaScript) están estandarizadas, la forma en que Netscape y Microsoft las implementan difiere entre sí. Por este motivo, la creación de páginas Web que usen esta tecnología, puede llegar a convertirse en una tarea muy compleja, puesto que hay que conseguir que la página se visualice perfectamente en ambos navegadores [15].

2.2.1.3 CSS (Cascading Style Sheets)

CSS acrónimo de Cascading Style Sheets (hojas de estilo en cascada). Es un mecanismo simple que utiliza una especificación de los estilos aplicables a un documento [10].

Sabiendo que los estilos definen la forma de mostrar los elementos de un documento, CSS define dos tipos de ellos: el estilo lógico y el estilo físico. El estilo lógico se refiere a la composición del documento: cabeceras, párrafos, tablas, entre otros; sin tomar en cuenta la apariencia final, sino la estructura del documento. Por otro lado, el estilo físico si maneja la apariencia final (sin importar su estructura): párrafos con un cierto tipo de letra, tablas con un determinado color de fondo, los enlaces de diferentes colores o subrayados, y más [10].

Esta forma de descripción ofrece a los desarrolladores Web el control total sobre el estilo y formato de sus documentos separando el contenido de la presentación. Inclusive pueden controlar la apariencia de múltiples páginas Web al mismo tiempo, ya que cualquier cambio en el estilo de marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento [10].

La finalidad de las hojas de estilo es crear estilos físicos, separados de las etiquetas HTML (sin colocarlos como atributos de los elementos), y aplicarlos en los bloques de texto en los que se quieran aplicar [14].

Las formas más conocidas de dar estilo a un documento son las siguientes: [14]

- Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento <link>, el cual debe ir situado en la sección <head> del documento HTML.
- Utilizando el elemento <style>, en el interior del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección <head>. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.
- Utilizando estilos directamente sobre aquellos elementos que lo permiten a través del atributo <style> dentro de <body>. Pero este tipo de estilo pierde las ventajas que ofrecen las hojas de estilo al mezclarse el contenido con la presentación.

CSS funciona a base de reglas que consisten en declaraciones que especifican el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas que se aplican a un documento HTML o XML. Una regla tiene dos partes: un selector y la declaración. A su vez la declaración se compone de una propiedad y el valor que se le asigne.

Siguen la siguiente sintaxis:

- Regla: selector {declaración}
- Declaración: propiedad:valor;

Ejemplo:

```
.MiEstilo {  color: #FFFFFF;  }
```

El selector funciona como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración. La declaración es la parte de la regla que establece cuál será el efecto a aplicar sobre el elemento. En el ejemplo anterior, el selector MiEstilo indica que todos los elementos MiEstilo se verán afectados por la declaración donde se establece que la propiedad color va a

tener el valor negro (#FFFFFF) para todos los elementos Estilo del documento o documentos que estén vinculados a esa hoja de estilos.

El conjunto de reglas definidas en un CSS se pueden incluir en el código HTML de la página, escribirse en un archivo externo y hacer referencia a éste o importarlo desde una ubicación remota.

Ventajas de CSS

Las principales ventajas del CSS son: [10]

- Reduce la sobrecarga del servidor y del ancho de banda.
- Reduce tiempo en el diseño y la programación de páginas de Internet.
- Reduce las operaciones de actualización y mantenimiento.
- Los encargados del contenido no tendrán que preocuparse por tablas complejas ni por los contenidos de fuentes ni etiquetas.
- Los diseñadores, programadores y agencias no tendrán que preocuparse por cómo los clientes desplieguen la página.
- Los cambios globales se pueden hacer en cuestión de minutos, con mínimo margen de error.
- Aumenta la compatibilidad mediante el cumplimiento de las recomendaciones *W3C* sobre estándares de Internet.
- Aumenta la accesibilidad ya que elimina algunos de los elementos de presentación del marcado.
- Control centralizado de la presentación de las páginas de un sitio Web completo.

Desventajas de CSS

Algunas desventajas de uso de CSS pueden ser: [10]

- Soporte irregular por parte de los navegadores:
- No todas las propiedades de las CSS son reconocidas igualmente por todos los navegadores, ni siquiera son reconocidas por diferentes versiones del mismo navegador, haciendo que las páginas sean visualizadas con un formato no estándar.
- Algunas propiedades de las CSS, como las que aceptan la posición o visibilidad de los elementos, pueden provocar que una parte del contenido de la página resulte inaccesible desde ciertos navegadores.

2.2.1.4 JavaScript

JavaScript es un lenguaje interpretado, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y C [14].

Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia de manera explícita (puede ser simulada mediante la propiedad *prototype* presente en todo constructor en JavaScript), es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad [14].

Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM [14].

Características de JavaScript

Las principales características de JavaScript son: [16]

- Es un lenguaje orientado a eventos, diseñado específicamente para el desarrollo de aplicaciones dentro del ámbito de Internet.
- Es sencillo y pensado para proveer rapidez y ligereza en las actividades realizadas sobre plataforma Web.
- Los programas en JavaScript pueden estar incluidos en los documentos HMTL y/o tener la referencia a un archivo externo de extensión 'js'; y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos y manejar los elementos de la página.
- Es un lenguaje que permite tanto la programación de pequeños scripts, como de programas más grandes, orientados a objetos, con funciones y estructuras de datos complejas.

Ventajas de JavaScript

Las principales ventajas de JavaScript son: [16]

- Se ejecuta del lado del cliente, por lo tanto no sobrecarga el servidor si existen muchas peticiones.
- Es multiplataforma.

- Es un lenguaje orientado a eventos manteniendo características de programación orientado a objetos.
- Los programas JavaSript tienden a ser pequeños y compactos; no requieren mucha memoria ni tiempo adicional de transmisión. Además, al incluirse dentro de las mismas páginas HTML se reduce el número de accesos independientes a la red

Desventajas de JavaScript

Algunas desventajas de uso de JavaScript son: [16]

- El código debe descargarse completamente antes de ser ejecutado, esto puede en algunos casos aumentar el tiempo de respuesta de la página al momento de la carga inicial.
- Por razones de seguridad las opciones de ejecución y uso de recursos del lado del cliente JavaScript están muy limitadas, sobre todo el acceso al hardware y archivos.
- No es posible ocultar el código fuente y evitar la copia y reutilización de éste (hay algunos esfuerzos encriptando el código fuente con algoritmos RSA, SHA, etc., pero amerita plugins adicionales).
- Los navegadores tienen la opción de evitar la ejecución de códigos JavaScript de las páginas Web, por lo tanto será decisión del cliente si lo permitirá o no.

La tabla 1 muestra las diferencias entre Java y JavaScript.

JAVA	JAVASCRIPT
Compilado (bytecodes). Se descarga del servidor y se ejecuta en el cliente	Interpretado por el cliente
Es necesario definir los tipos de datos de las variables	Los tipos de datos de las variables no se declaran
Se utilizan APPLETS. Se accede a ellos desde documentos.	El código se incrusta en documentos HTML
Orientado a Objeto	Basado en objetos

Tabla 1 Diferencias entre Java y JavaScript

2.2.1.5 DOM (Document Object Model)

DOM acrónimo de **D**ocument **O**bject **M**odel (Modelo de Objetos del Documento). [4] Es una forma de representar los elementos de un documento estructurado (tal como una página Web HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades.

DOM define qué atributos son asociados con cada objeto y cómo los objetos y los atributos pueden ser manipulados. El HTML dinámico (DHTML) se basa en el DOM para cambiar dinámicamente la apariencia de las páginas Web después de que han sido descargadas por un navegador.

Funcionamiento

Cuando un navegador carga una página, crea una jerarquía de su contenido, lo cual representa la estructura del HTML. Esto genera una organización parecida a un árbol de nodos que pueden relacionarse, donde cada nodo representa un elemento, atributo o algún otro objeto.

Cada uno de estos objetos tendrá sus propios y únicos métodos y propiedades más un conjunto común de métodos y propiedades relacionadas con la estructura de árbol del documento para poder manipularlos.

DOM hace posible escribir aplicaciones que funcionen en todos los navegadores, servidores y plataformas; además de crear un modelo de documentos independiente al lenguaje de programación que se utilice para desarrollar.

2.2.1.6 AJAX (Asynchronous JavaScript And XML)

AJAX acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML. [4] Es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de algunas tecnologías ya existentes:

- **XHTML** o **HTML** y **CSS** para el diseño que acompaña a la información.
- **DOM** accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto **XMLHttpRequest** para intercambiar datos asincrónicamente con el servidor Web.
- **XML** es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, entre otros.

El modelo clásico de aplicaciones Web (Figura Nº 2-6) funciona de la forma en que la mayoría de las acciones del usuario en la interfaz disparan un requerimiento *HTTP* al servidor Web. El servidor efectúa un proceso (recopila información, procesa números, habla con varios sistemas propietarios), y le devuelve una pagina HTML al cliente. Este acercamiento tiene mucho sentido a nivel técnico, pero no lo tiene para una gran experiencia de usuario. ¿Mientras el servidor esta haciendo lo suyo, que esta haciendo el usuario? Esperando. Y, en cada paso de la tarea, el usuario espera por más.

Obviamente, si estuviéramos diseñando la Web desde cero para aplicaciones, no querríamos hacer esperar a los usuarios. Una vez que la interfaz esta cargada, ¿porque la interacción del usuario debería detenerse cada vez que la aplicación necesita algo del servidor? De hecho, ¿porque debería el usuario ver la aplicación yendo al servidor?

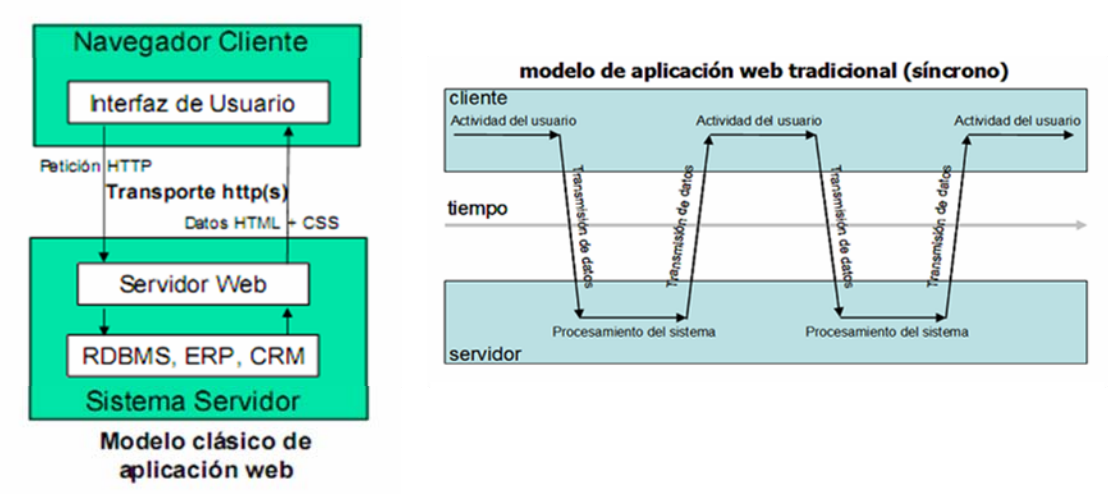

Figura 5 Modelo Web tradicional

Una aplicación AJAX elimina la naturaleza "arrancar-frenar-arrancar-frenar" de la interacción en la Web, esto lo hace introduciendo un intermediario "*motor AJAX*" entre el usuario y el servidor. Parecería que sumar una capa a la aplicación la haría menos reactiva, pero la verdad es lo contrario.

Cada vez que el usuario ejecuta una acción (un clic, la presión de una tecla, el arrastre de un objeto del navegador, etc.) debe solicitar datos al servidor a través de Internet, para luego regenerar la página que el usuario está viendo. Estamos acostumbrados a un modelo de interacción sincrónica basada en clic-petición-presentación.

Con AJAX la interacción pasa a ser asíncrona. Cada vez que se hace clic no necesariamente se establece una conexión con el servidor. Básicamente, la principal virtud de AJAX está en la potencia que se le puede extraer al trabajo asíncrono de peticiones al servidor.

El modelo de AJAX propone descargar un motor especial para cada aplicación, sirviéndose de las tecnologías antes mencionadas y presentes en la gran mayoría de los navegadores. De esta forma, los usuarios pueden acceder de inmediato al contenido sin interrupciones gracias a que el motor de AJAX intercambia datos silenciosamente con el servidor para que estén listos cuando el usuario los requiera y sin recargar la página. Ahí está la magia precisamente, en que ciertos procesos se muestran en la página sin retardo alguno, ya que mientras el usuario miraba unos datos en la pantalla, AJAX le ha estado preparando los siguientes que iba a necesitar.

En el modelo de aplicación Web de AJAX (ilustrado en la Figura 6) en vez de cargar una página Web, al inicio de la sesión, el navegador carga el motor AJAX. Este motor es el responsable de renderizar la interfaz que el usuario ve y de comunicarse con el servidor en nombre del usuario. El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que el servidor haga algo.

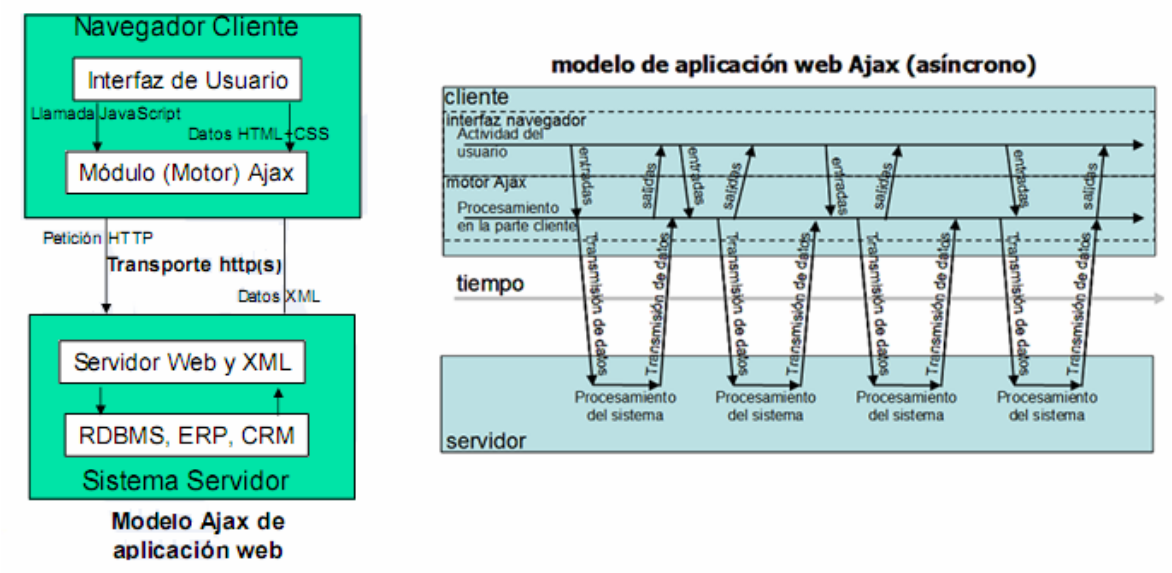

Figura 6 Modelo Web Tradicional con AJAX

Ventajas de AJAX

- Recuperación asíncrona de datos, el usuario no tienen que esperar después de una petición.
- Acercamiento de la metáfora de escritorio.
- No requiere plugins.
- Reduce tiempos de escritura y tiempos de espera.
- Reduce el ancho de banda.
- Reduce la carga al servidor (validación de formularios).

Desventajas de AJAX

- Pueden aumentar las llamadas al servidor.
- Peligro de incompatibilidades en navegadores.
- Deja de existir el botón atrás (los usuarios deben cambiar su manera de entender los sitios Web).
- Uso excesivo de Javascript: seguridad, compatibilidad, accesibilidad y complejidad.
- La percepción de cambio es menor.

2.2.2 Tecnologías del Lado del Servidor

Las tecnologías del lado del servidor son aquellas que no dependen del navegador Web, sino son las que te generan el acceso a una página residente en éste, ya que son interpretadas y ejecutadas por el mismo servidor [10].

Los primeros servidores Web permitían visualizar exclusivamente información estática. Esto representó bien pronto una limitación; sobre todo desde el momento en el que la actividad publicitaria y comercial comenzó a concentrarse también en la red Internet. La primera solución técnica realizada fue la posibilidad que el servidor Web lanzase programas residentes en la máquina de servicio. Esta tecnología, conocida como CGI permite al servidor Web lanzar programas escritos principalmente en C. Si bien la tecnología CGI resolvía el problema de la presentación exclusivamente estática de la información, al mismo tiempo presentaba dos limitaciones importantes: una era el problema de seguridad que podía representar el hecho que mediante la llamada a una página se pueda ejecutar programas indeseados en el servidor, la segunda era de carga del servidor (si una misma página que lanzaba un programa era llamada desde 100 clientes concurrentemente, en el servidor se ejecutaban 100 procesos, uno por cada cliente que solicitaba esa página) [10].

Para resolver estos problemas, se buscó desarrollar una tecnología que permitiera ejecutar, en un único proceso del servidor, todos los pedidos de ejecución de código del servidor Web sin importar la cantidad de clientes que se conectaban concurrentemente. Esta solución, denominada Servlet en tecnología Java o filtro en tecnología Microsoft, permite el poder ejecutar código en un único proceso externo que gestiona todas las llamadas realizadas por el servidor Web, impidiendo al mismo tiempo que el servidor Web pueda llamar a ejecutar programas del sistema operativo [10].

Desde ese momento, se comienzan a desarrollar tecnologías del lado del servidor que controlan los problemas y posibles limitaciones que presentan las aplicaciones Web y que hacen más eficientes el procesamiento, la comunicación con el cliente y la presentación de las páginas finales [10].

Fueron entonces desarrollados lenguajes que pueden ser incluidos al interno de archivos HTML. Estos comandos pueden ser interpretados (como por ejemplo las páginas ASP o PHP) o precompilados (como en las páginas JSP o ASP.NET) [10].

Los Servlets y Java Server Pages (JSPs) son dos métodos de creación de páginas Web dinámicas en el servidor usando el lenguaje Java. En ese aspecto son similares a otros métodos o lenguajes tales como el PHP, los CGI programas que generan páginas Web en el servidor [17].

2.2.2.1 Servlet

Un Servlet es un objeto que se ejecuta en un servidor o contenedor JEE (Java Enterprise Edition), fue especialmente diseñado para ofrecer contenido dinámico desde un servidor Web, generalmente es HTML [18].

Los servlets son programas que funcionan como los CGIs convencionales atendiendo peticiones de un cliente teniendo al servidor como el encargado, pero escritos en Java y con la ventaja de explotar todas las bondades de Java. Por ejemplo, un servlet puede ser responsable de tomar los datos de un formulario HTML y enviarlos a una base de datos para actualización de la misma [18].

Los *Servlets* son un sustituto eficaz de los CGI; proveen la forma de generar documentos dinámicos que son fáciles de escribir y ejecutar. También evitan el problema de desarrollar la programación según la plataforma utilizada. Los servlets son desarrollados con su propia API, una extensión estándar de Java [18].

Características de los Servlets

Entre las principales características de los Servlets encontramos:

- Están escritos en Java y siguen un API bien estandarizado.
- El manejo de las peticiones se hace con hilos y no con procesos, reduciendo la carga para el procesamiento del servidor.
- Pueden comunicarse directamente con el servidor Web.
- Pueden comunicarse entre ellos.
- Se encuentran compilados en el contenedor Web, por lo cual su ejecución es muy rápida.

Ventajas de los Servlets

Entre las principales ventajas de los Servlets encontramos: [19]

- **Desempeño**: Los servlets son más rápidos que los CGI debido a que utilizan threads en lugar de procesos.
- **Portabilidad**: Los servlets son tan portables como cualquier otra aplicación de Java.
- **Seguridad**: Los Servlets no están en riesgo de correr comando de shell no planeados. Los lenguajes compilados como Java (o C) proveen mejor seguridad que los lenguajes que interpretan scripts.
- **Desarrollo**: Las IDEs pueden proveer útiles herramientas fáciles de aprender para nuevos desarrolladores.

2.2.2.2 JSP (Java Server Pages)

JSP es una tecnología que surge por la necesidad de programar páginas Web dinámicas y capaces de procesar información residente en el servidor sin utilizar servlets. Con JSP se utiliza código o funciones Java predefinidas incluidas dentro del documento HTML. En JSP, se escribe el texto que va a ser devuelto en la salida, normalmente código HTML, incluyendo código java dentro de él para poder modificar o generar contenido dinámicamente [14].

Los JSPs son en realidad servlets: un JSP se compila a un programa en Java la primera vez que se invoca, y del programa en Java se crea una clase que se empieza a ejecutar en el servidor como un servlet. La principal diferencia entre los servlets y los JSPs es el enfoque de la programación: un JSP es una página Web con etiquetas especiales y código Java incrustado, mientras que un servlet es un programa que recibe peticiones y genera a partir de ellas una página Web [17].

Características de JSP

Entre las principales características de JSP encontramos: [17]

- **Separación de la presentación y el contenido:** Con JSP se programa dentro de las páginas HTML con código Java, pero encerrando el código en un conjunto de marcas que sólo se interpretan en

el servidor al momento de ejecutar la aplicación. Este mecanismo hace que queden bien delimitados los espacios de procesamiento y diseño de la interfaz de la página Web.

- **Reutilización de componentes:** El modelo de uso de JSP se basa en la reutilización de componentes programados en Java que cumplen tareas bien definidas. El uso de los mismos hace que se optimice considerablemente la utilización de recursos en el servidor.

Ventajas de JSP

Las principales ventajas de los JSP son: [17]

- Cuentan con el soporte y poder de java para implementar sus funcionalidades.
- Como consecuencia del punto anterior, se puede decir que ofrecen independencia de la plataforma.
- Escalabilidad y alto rendimiento.
- Pueden ser enlazados con diferentes bases de datos, como Oracle, Servidores SQL, etc.
- Actualmente las tecnologías JSP y Java Servlets se encuentran lo suficientemente probadas, debido a la amplia utilización de las mismas en el desarrollo de aplicaciones empresariales como parte de la especificación JEE. Además cuenta con una completa documentación disponible para los desarrolladores.

La tabla 2 presenta un resumen de las características de las tecnologías JSP y Servlets, destacando aspectos técnicos de importancia.

Aspectos	JSP / Servlets
Multiplataforma	Sí
Configuración	Regular
Servidores	Tomcat, Resin, Jboss, Glassfish, etc
Despliegue	Sí
Integración de librerías	Manual
Hospedaje	Medio
Costo de Hospedaje	Alto
Rapidez de Desarrollo	Lento/Medio
Tipo de Lenguaje que usa	Java/ JSP/ Servlets
Uso	Regular
Documentación	Mucha

Tabla 2. Características de JSP / Servlets

2.2.2.3 Apache Tomcat 6.0

Tomcat es un servidor Web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor Web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

La historia de Tomcat se remonta a la implementación de referencia de la API Servlet de Sun. Sun no iba a publicar una especificación a menos que tuviera una implementación de referencia. Una implementación de referencia es un software en el que se basan las empresas para crear prototipos de aplicaciones con los que poner a prueba los conceptos y la utilización de una nueva tecnología. Sun desarrolló la implementación de referencia inicial de la API Servlet y la presentó con el nombre de Java Servlet Web Development Kit (JSWDK).

Pero ocurrió que en octubre de 1999, el desarrollador de JSWDK, James Duncan Davidson, convenció a Sun de que donara el código fuente de JSWDK a la Apache Software Foundation. Dos meses más tarde veía la luz Tomcat 3.0. En Apache Software Foundation se habían dado cuenta de las posibilidades de ese código para desarrollar productos de código libre para el servidor con la calidad de productos comerciales y crearon el Proyecto Jakarta. En la actualidad, el Proyecto Jakarta ofrece docenas de productos de código libre escritos en Java, incluyendo Tomcat.

En la actualidad Tomcat es mantenido y desarrollado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software Licence. Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 6.x, que implementan las especificaciones de Servlet 2.5 y de JSP 2.1. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

Características de Tomcat 6.0

- Tomcat contiene una jerarquía de directorios que incluyen:
  - bin - arranque, cierre, y otros scripts y ejecutables.
  - common - clases comunes que pueden utilizar Catalina y las aplicaciones web.
  - conf - ficheros XML y los correspondientes DTD para la configuración de Tomcat.
  - logs - logs de Catalina y de las aplicaciones.
  - server - clases utilizadas solamente por Catalina.
  - shared - clases compartidas por todas las aplicaciones web.
  - webapps - directorio que contiene las aplicaciones web.
  - work - almacenamiento temporal de ficheros y directorios.
- Soporte de Servlet 2.5 y JSP 2.1
- Diseñado para funcionar en Java SE 5.0 y posteriores
- Posibilidad de Comunicación asíncrona con dispositivos I/O

Ventajas de Tomcat 6.0

- Tomcat es gratis, es fácil de instalar, se ejecuta en máquinas más pequeñas y es compatible con las API más recientes de Java.
- Innumerables empresas utilizan Tomcat.
- Tomcat se basa en que miles de desarrolladores contribuyen con código.
- Compatibilidad con J2EE. Aunque Tomcat no es directamente compatible con todas las API de J2EE que soporta WebSphere (como JDBC, JNDI, JavaMail, RMI, JMS, XML y EJB), todas esas API, con la notable excepción de EJB, están disponibles agregando archivos Java (JAR) disponibles gratuitamente.

2.2.3 Sistemas Manejadores de Bases de Datos (SMBD)

El sistema manejador de bases de datos (SMBD) es la porción más importante del software de un sistema de base de datos. Un SMBD es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica [22].

El SMBD es esencial para el adecuado funcionamiento y manipulación de los datos contenidos en la base de datos. Se puede definir como: *"El Conjunto de programas, procedimientos, lenguajes, etc. que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su integridad, confidencialidad y seguridad"* [23].

2.2.3.1 Funciones de un SMBD

El Objetivo primordial de los SMBD es proporcionar un entorno para recuperar, manipular y almacenar la información en la base de datos, para lo cual debe proporcionar a los usuarios una visión abstracta de los datos. Es decir, los detalles de cómo se almacenan y se mantienen los datos, son transparentes para los usuarios.

En el mercado informático están disponibles una gran variedad de SMBD, tanto propietarios como de software libre. Entre los más conocidos de licencia propietaria están: Oracle, IBM DB2, Ms SQLServer, Sybase, etc. y de software libre por su alto rendimiento tenemos: MySQL y PostgreSQL.

Entre las principales funciones de un SMBD encontramos: [22] [23]

- Crear y organizar la Base de datos.
- Establecer y mantener las trayectorias de acceso a la base de datos de tal forma que los datos puedan ser accesados rápidamente.
- Manejar los datos de acuerdo a las peticiones de los usuarios.
- Registrar el uso de las bases de datos.
- Interacción con el manejador de archivos: Los datos en la base se guardan en disco mediante el sistema de archivos, proporcionado comúnmente por el sistema operativo. El manejador de la base, traduce las diferentes proposiciones del manejo de datos en comandos del sistema de

archivos de bajo nivel. De esta forma el manejador se puede encargar del almacenamiento, recuperación y actualización de los datos en la base.

- Implantación de la integridad: Los valores de los datos que se almacenan en la base, deben satisfacer ciertas limitantes de consistencia, estas limitantes deben ser determinadas por el administrador, pero es el manejador el encargado de verificar que las actualizaciones que se hagan a la base cumplan con dichas normas.
- Puesta en práctica de la seguridad: El manejador de la base es quien verifica que los accesos a la base sean realizados por las personas autorizadas.
- Respaldo y recuperación: Entre las labores que debe ejecutar el manejador está la de verificar de forma constante la integridad de la base, y lograr recuperación de datos y/o mejoras en caso que se requieran.
- Control de concurrencia: Se podría entender, esta, como la principal tarea del manejador de la base, o por lo menos la más difícil. Cuando varios usuarios están accesando la base al mismo tiempo, es posible que la consistencia de los datos no se conserve. El manejador debe encargarse de coordinar los accesos de los diferentes usuarios, de forma que los datos en la base no se dañen.

Si bien existen otros SMBD que se utilizan en el desarrollo de aplicaciones Web, entre otros, Oracle, MS SQLServer, Sybase, MySQL, PostgreSQL, la investigación se enfocará en utilizar IBM DB2, debido al acuerdo tecnológico suscrito entre la Corporación Suiche 7B e IBM de Venezuela, a través del cual IBM de Venezuela dispone de un equipo Mainframe IBM/390 con sistema operativo zOS/390 para ambiente de producción y desarrollo.

Esta configuración brinda el ambiente ideal para la ejecución del SMBD IBM DB2, por lo cual a continuación se describen las características, ventajas y desventajas del IBM DB2 Universal Data Base 8.0

2.2.3.2 IBM DB2 UDB 8.0

IBM DB2 Universal Data Base es un sistema gestión de bases de datos relacionales, cuyo origen se remonta a 1970, siendo creado por IBM; sin embargo, es a partir de 1994 cuando el DB2 Universal Database es constituido en base a dos productos incluidos en el DB2 de AIX: el DB2 Common Server, que para propósitos generales incluía funciones avanzadas para el mercado de servidores de bases de datos con soporte de hardware SMP y OLTP; y el DB2 Parallel Edition, que fue desarrollado para soportar aplicaciones de gran escala, como el Data Warehousing y Data Mining y aplicaciones de negocios a nivel mundial como la SAP, People Soft y Baan.

DB2 incluye todo lo necesario para implementar una solución de replicación de datos en cualquier ambiente distribuido o heterogéneo, pues permite enviar los datos a cualquier sitio para cubrir todos los requerimientos de una empresa, desde oficinas centrales a sucursales, usuarios móviles, proveedores, clientes y socios de negocios.

Gracias a su alcance global y de bajo costo, Internet puede ser una solución de negocios muy poderosa para realizar operaciones comerciales garantizando un nivel de seguridad y confiabilidad con sus servicios de autorización y autenticación integrados a redes y sistema operativos, soportando el network−computing utilizando Java y JDBC, incluyendo capacidad nativa de almacenar varios tipos de datos: alfanuméricos, video, imagen, audio y los definidos por el usuario.

En la actualidad la tecnología de gestión de datos de IBM es utilizada por más de 40 millones de usuarios de 300.000 empresas en todo el mundo. Mientras que la evolución del DB2, Universal Data Base dispone de más de 6 millones de usuarios y 1.300.000 licencias instaladas.

Así mismo IBM DB2 UDB es multiplataforma, al funcionar en más de 16 plataformas diferentes (10 no IBM).

Características de IBM DB2 UDB 8.0

- Permite el manejo de objetos grandes (hasta 2 GB), la definición de datos y funciones por parte del usuario, el chequeo de integridad referencial, SQL recursivo, soporte multimedia: texto, imágenes, video, audio; queries paralelos, commit de dos fases, backup/recuperación on−line y offline.
- Además cuenta con un monitor gráfico de performance el cual posibilita observar el tiempo de ejecución de una sentencia SQL y corregir detalles para aumentar el rendimiento.
- Mediante los extensores se realiza el manejo de los datos no tradicionales, por ejemplo si tengo un donde tengo almacenados los curriculums de varias personas, mediante este puedo realizar búsquedas documentos con los datos que me interesen sin tener que ver los CV uno por uno.
- Esta capacidad se utiliza en sistemas de búsqueda de personas por huellas digitales, en sistemas información geográfica, etc.
- Con DB2 es posible acceder a los datos usando JDBC, Java y SQL.
- Plataformas host:
    - OS/390(MVS)
    - VM & VSE
    - OS/400
- Plataformas de servidor:
    - OS/2 Warp Server
    - Sinix
    - SCO Openserver
    - Windows NT / Server 2003 / Server 2008
    - Solaris.
    - Plataformas Cliente:
    - OS/2
    - DOS
    - Sinix
    - SCO OpenServer
    - Windows 3.1 / 95 / 98 / NT / XP / Vista
    - Macintosh System 7
    - Aix
    - HP Ux.
    - Solaris

Ventajas de IBM DB2 UDB 8.0

- Permite agilizar el tiempo de respuestas de esta consulta.
- Permite realizar respaldo es línea con diferentes grados de granularidad, sin afectar la disponibilidad de acceso a los datos por parte de los usuarios.
- Permite almacenar información en un amplio rango de equipos, desde una PC portátil hasta un completo ambiente de mainframes procesando en paralelo
- Recuperación utilizando accesos de sólo índices.
- Implementa Predicados correlacionados, Tablas de resumen, Tablas replicadas y Uniones hash.
- DB2 utiliza una combinación de seguridad externa y control interno de acceso a proteger datos.
- DB2 proporciona un juego de datos de acceso de las interfaces para los diferentes tipos de usuarios y aplicaciones.
- DB2 guarda sus datos contra la pérdida, acceso desautorizado, o entradas inválidas.
- La tecnología de replicación heterogénea (heterogeneous replication) en SQL Server permite la publicación automática de los datos en otros sistemas que no sean SQL Server, entre los que se incluyen DB2.
- La mayoría de los equipos IBM utilizan DB2 porque es confiable y tiene soporte técnico los 365 días del año.
- DB2 se basa en dos ejes que lo hacen fuerte en su rendimiento: utiliza un sistema multiprocesador (SMP) simétrico y un sistema de procesador paralelo masivo.
- DB2 distribuye y recuerda la ubicación de cada pista donde se encuentra la información. En el contexto de una larga base de datos, este sistema de partición hace que la administración sea mucho más fácil de manejar que una base de datos de la misma medida no particionada.
- La base de datos se puede programar para tener una exacta cantidad de particiones que contienen la información del usuario, índice, clave de transacción y archivos de configuración. De esta forma, los administradores definen grupos de nodos, que son una serie de particiones de la base, lo que posteriormente facilita cualquier búsqueda.

Desventajas de IBM DB2 UDB 8.0

- DB2 consume gran cantidad de recursos del computador a la hora de realizar consultas o ejecuciones de gran tamaño.
- Influye en las aplicaciones que se tienen desarrolladas y en las que se van a implementar, debido a que DB2 no limita su consumo de recursos del computador, por lo cual la interacción de la aplicación con la base de datos debe realizarse de manera muy eficiente y teniendo cuidado de no cometer omisiones como dejar conexiones sin cerrar o mal manejo de índices
- Influye en la elección el hardware utilizado debido a que la configuración de hardware del equipo donde se ejecutará el DB2 dependerá de la complejidad de las operaciones que realizarán, así como del volumen de datos que se manejará.
- Requiere una entonación o configuración inicial para obtener alto rendimiento que puede llegar a ser muy compleja y usualmente requiere personal especializado en DB2.
- La migración de repositorios en DB2 a otros SMBD, puede ser engorrosa debido a que DB2 maneja varios tipos de datos únicos, que no tienen equivalente directo en los otros SMBD.

2.2.4 Resumen de Tecnologías Usadas en la Aplicación Desarrollada

- **Tecnologías del lado del cliente**
  - HTML
  - DHTML
  - CSS
  - JavaScript
  - DOM
  - AJAX

- **Tecnologías del lado del servidor**
  - JSP
  - Servlets

- **Servidor Web**
    - Apache Tomcat 6.0

- **Sistema Manejador de Bases de Datos**
    - IBM DB2 UDB 8.0. (Licenciado por IBM para la Corporación Suiche 7B)

2.2.5 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces**.** Es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias). [14]

En la programación orientada a objetos resulta complicado descomponer el sistema en objetos (encapsulación, granularidad, dependencias, flexibilidad, reusabilidad, etc.), los patrones de diseño nos permiten identificar a los objetos apropiados de una manera mucho más sencilla. También nos permiten determinar la granularidad de los objetos. Además, los patrones de diseño, también nos ayudan a especificar las interfaces, identificando los elementos claves en las interfaces y las relaciones existentes entre distintas interfaces.

2.2.5.1 Composite View

El patrón Composite sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol.

Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

Plantilla GoF (Gang of Four) del Patron de Diseño Composite View

- **Nombre:**

  Composite View (Vista Compuesta)

- **Propósito**

Crear Vistas Compuestas de varias sub-vistas de forma modular, flexible y extensible para construir vistas de páginas JSP para aplicaciones J2EE.

- **Motivación**

Cuando un usuario navega a través de aplicaciones gráficas los datos y el contenido entre las diferentes páginas varía, pero muchos elementos, como una cabecera común o una barra lateral permanecen intactos en todas las vistas. La estructura y disposición de cada página puede ser la misma y algunos elementos o secciones de la página pueden aparecer en varias páginas diferentes. Cuando estos elementos y grupos se codifican directamente en la aplicación se vuelve muy difícil la tarea de modificar las vistas y se puede incurrir en inconsistencias.

Las páginas Web sofisticadas presentan contenido de varias fuentes de datos, utilizando una Vista Compuesta (Composite View) formada por otras sub-vistas. Cualquier cambio realizado en una sub-vista es reflejado automáticamente en cada Vista Compuesta que la utilice.

La Vista Compuesta también maneja la disposición de sus sub-vistas y proporciona una plantilla, dando una apariencia consistente y facilidades a la hora de modificarla y mantenerla a lo largo de toda la aplicación.

- **Aplicabilidad**

  Se usará el patrón Composite View cuando:

  - Varias vistas compuestas utilizan sub-vistas similares.
  - Las porciones atómicas del contenido de una vista cambian con frecuencia.

- **Estructura**

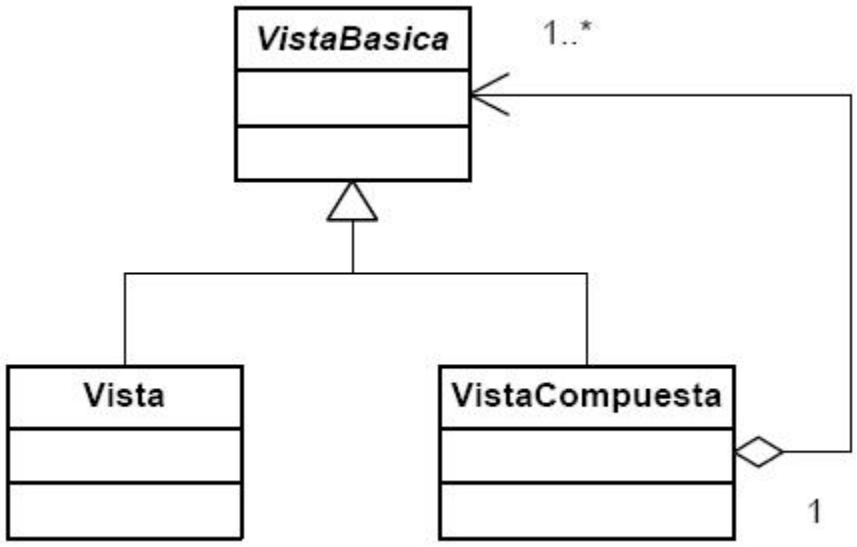

Figura 7 Estructura estática del Composite View

- **Participantes y Colaboraciones**

Vistas Compuestas

Una Vista Compuesta es una vista a la que se le han agregado varias sub-vistas.

***View Manager* (Controlador de Vistas)**

El Controlador de Vista maneja la inclusión de fragmentos de una plantilla en la Vista Compuesta. El Controlador de Vista podría ser parte de un motor de ejecución estándar de páginas JSP, en la forma de la etiqueta <jsp:include> de JSP (véase figura), o podría encapsularse dentro de un *helper* JavaBean (JSP 1.0+) o una etiqueta personalizada (JSP 1.1+) para proporcionar una funcionalidad más robusta.

Vistas Incluidas

Una Vista Incluida es una sub-vista, una pieza atómica de una vista mayor. Esta vista incluida también podría ser una vista compuesta, incluyendo varias sub-vistas.

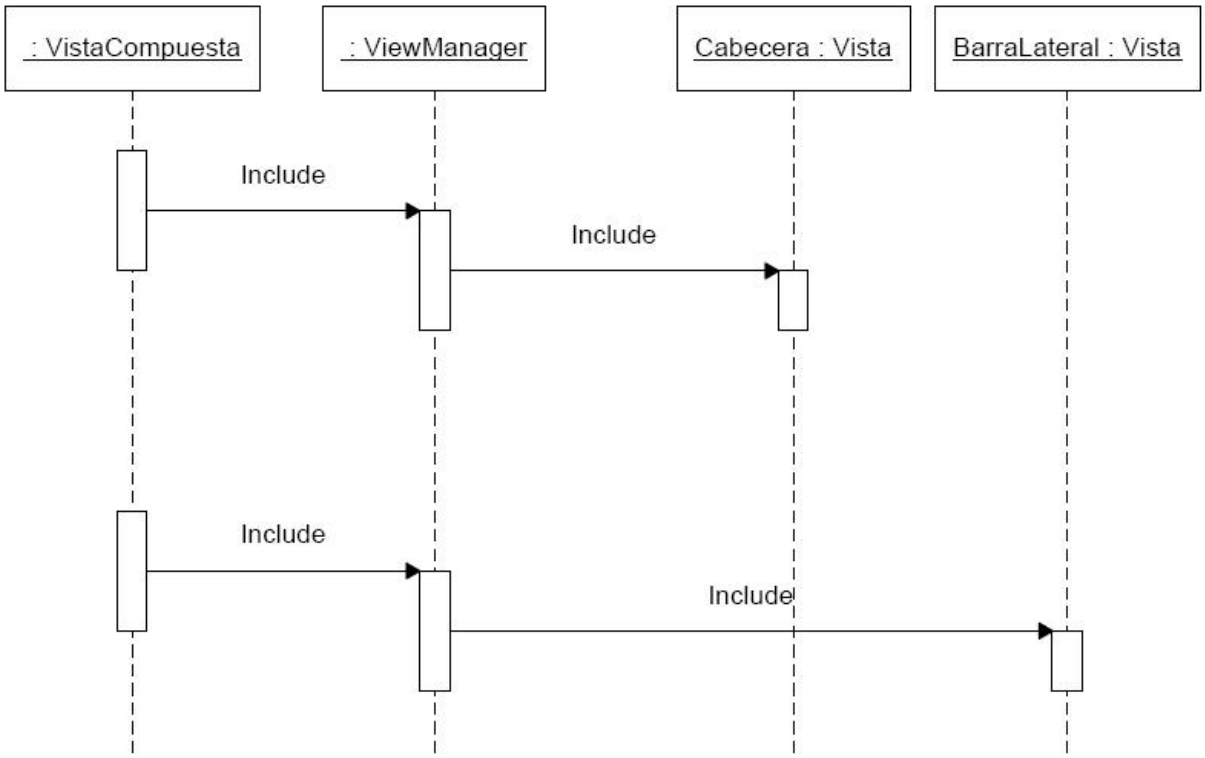

Figura 8 Diagrama de secuencia del Composite View

- **Consecuencias**
  - Cada componente de la plantilla puede incluirse dinámicamente dentro del todo y la distribución de la página puede manejarse de forma independiente del contenido.
  - Esta solución es útil para la creación de vistas basadas en la inclusión y sustitución de fragmentos de plantilla modulares, tanto estáticos como dinámicos. Los diseñadores pueden hacer un prototipo de la distribución de la web, poniendo contenido estático en cada región de la plantilla. Según va progresando el desarrollo, este contenido se puede ir sustituyendo por el contenido dinámico.
  - Promueve la reutilización de porciones atómicas de la vista, animando a realizar diseños modulares.
  - Es apropiado utilizar este patrón para generar páginas que muestren componentes que podrían combinarse de diferentes maneras. Esto ocurre, por ejemplo, con portales que incluyan numerosas sub-vistas independientes, como noticias, información del tiempo, etc. en una sola página. La distribución de la página se maneja y modifica de forma independiente al contenido de las sub-vistas.

- Provoca una sobrecarga en tiempo de ejecución, un precio que hay que pagar por el incremento de flexibilidad que proporciona.

- **Implementación**
  - Un Composite View se puede implementar siguiendo la estrategia JSP page View o bien la estrategia Servlet View.
  - El control de la vista se puede implementar de diferentes formas: utilizando etiquetas jsp estándar, como <jsp:include>, utilizando componentes JavaBeans, y también mediante etiquetas personalizadas (JSP 1.1+).

- **Código de ejemplo**

A continuación se implementa el control de una vista con etiquetas estándar `<jsp:include>`. Utilizar etiquetas estándar para manejar la distribución y composición de las vistas es una estrategia fácil de implementar, pero no proporciona el poder y la flexibilidad de otras estrategias, ya que la distribución de las páginas individuales permanece embebida dentro de esas páginas. Así, aunque esta estrategia permite variar dinámicamente el contenido subyacente, cualquier cambio general en la distribución requeriría modificaciones individuales en muchas páginas JSP. Puede verse en el siguiente ejemplo:

```
<html>
        <body>
                <jsp:include page="/tope/banner.html" flush="true"/>
                <table width="100%">
                        <tr align="left" valign="middle">
                                <td width="20%">
                                        <jsp:include page="/encabezado/encAdmin.jsp" flush="true"/>
                                </td>
                                <td width="70%" align="center">
                                        <jsp:incluye page="/menu/menuAdmin.jsp" flush="true"/>
                                </td>
                        </tr>
                </table>
                <jsp:include page="/pie/footer.html" flush="true"/>
        </body>
</html>
```

Cuando se crea una vista compuesta utilizando etiquetas estándar, se puede incluir tanto contenido estático, ficheros HTML, como contenido dinámico, páginas JSP. Además, el contenido se puede incluir en el momento de la traducción o durante la ejecución. Si el contenido se incluye durante la traducción, la vista permanecerá sin modificada hasta que se recompile la página JSP, momento en el

que serán visibles las modificaciones incluidas en el contenido. En otras palabras, la página se distribuye y genera una sola vez, cada vez que la página JSP se recompila. El siguiente ejemplo, muestra una excepción de una página JSP que genera una vista compuesta de esta manera, utilizando la directiva `<%@ include %>` estándar de JSP que incluye el contenido en tiempo de traducción.

```
<table>
        <tr>
                <td><%@ file="noticias/nacionales.html" %> </td>
        </tr>
        <tr>
                <td><%@ file=" noticias /internacionales.html" %> </td>
        </tr>
        <tr>
                <td><%@ file=" noticias /deportes.html" %> </td>
        </tr>
        <tr>
                <td><%@ file=" noticias /farandula.html" %> </td>
        </tr>
</table>
```

La inclusión de contenido en tiempo de ejecución significa que los cambios en las sub-vistas son visibles en la página compuesta cada vez que el cliente accede a ella. Esto es mucho más dinámico y se puede conseguir utilizando la etiqueta `<jsp:include>` estándar de JSP, como se muestra en el siguiente ejemplo. Por supuesto que hay una sobrecarga del entorno de ejecución asociada con este tipo de generación de vistas, pero este es el inconveniente de mejorar la flexibilidad de las modificaciones de contenidos al-vuelo.

```
<table>
        <tr>
                <td><jsp:include page=" noticias / nacionales.jsp"  flush="true"/> </td>
        </tr>
        <tr>
                <td><jsp:include page=" noticias / internacionales.jsp"  flush="true"/> </td>
        </tr>
        <tr>
                <td><jsp:include page=" noticias / deportes.jsp"  flush="true"/> </td>
        </tr>
        <tr>
                <td><jsp:include page=" noticias / farandula.jsp"  flush="true"/> </td>
        </tr>
</table>
```

Existen otras estrategias que permiten realizar el control de la vista como pueden ser las etiquetas personalizadas o con JavaBeans, pero se ha escogido esta estrategia por su sencillez a la hora de implementarla y su claridad.

3 Marco Aplicativo

El Marco Aplicativo abarca la adaptación del proceso de desarrollo XP al caso particular de la aplicación Web a desarrollar, especificando cada una de las tareas a realizar dentro de cada una de las fases de dicho proceso.

Inicialmente se desarrollarán algunos puntos del Proceso de Desarrollo XP (Programación eXtrema), específicamente su definición y fundamentos, sus principales características y las actividades que contempla este proceso para el desarrollo de aplicaciones Web.

Seguidamente se presenta la planificación de las iteraciones que se llevarán a cabo durante la implementación del sistema Web. Se detallará la iteración realizada en el desarrollo del sub-módulo *Crear PDC* correspondiente al módulo de *Administrar PDC*, y en ella se proveerá información específica de tiempo exacto de duración, historias de usuario contempladas, además de las actividades efectuadas durante las etapas de Planificación, Diseño, Codificación y Pruebas, para llevar correctamente el proceso de desarrollo XP a la práctica.

Finalmente, se exponen las principales funcionalidades del sistema, a través de una descripción de cada una acompañadas con las pantallas o interfaces asociadas a dicha funcionalidad.

3.1 Proceso de Desarrollo XP (Programación Extrema)

XP es acrónimo de eXtreme Programming (Programación Extrema). Es un nuevo enfoque al desarrollo de software. La programación extrema es una metodología de desarrollo ligera basada en una serie de valores y una docena de prácticas. Ya que de alguna manera proporcionan un aumento en la productividad a la hora de generar software [20].

La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son aspectos naturales, inevitables e incluso deseables del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software [20].

3.1.1 Características de XP

Entre las principales características de XP se encuentran: [21]

- **Desarrollo iterativo e incremental:** pequeñas mejoras, unas tras otras.
- **Pruebas unitarias continuas:** frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- **Programación en parejas:** se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.
- **Interacción del equipo de programación con el cliente o usuario:** Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- **Corrección de todos los errores:** antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- **Refactorización del código:** rescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.

- **Propiedad del código compartida:** en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, esta metodología promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto.
- **Simplicidad en el código:** es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario.

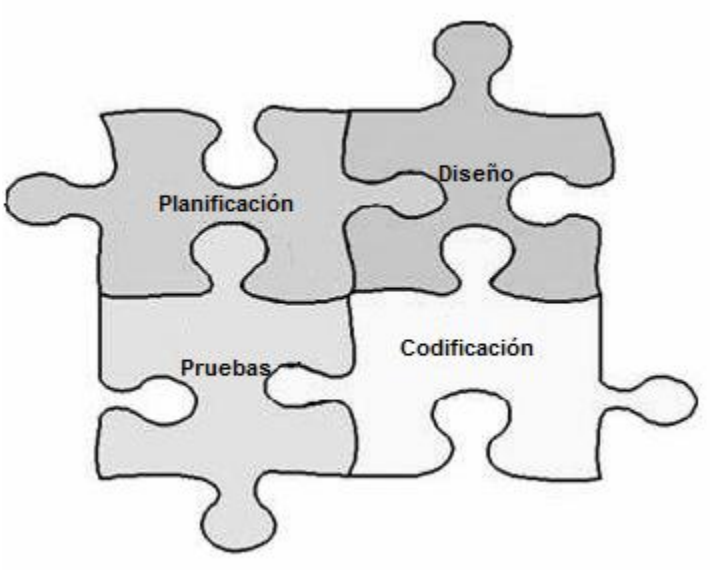

Figura 9 Actividades de XP

El proceso de desarrollo XP se ve representada en 4 actividades (Figura 9). Las actividades de este proceso no necesariamente se tienen que realizar en algún orden en específico, como también pueden trabajar en paralelo.

3.1.2 Actividades de la Programación Extrema

La Programación Extrema comprende las siguientes actividades: [21]

3.1.2.1 Planificación

- Se escriben historias de usuario, cuya idea principal es describir un caso de uso en dos o tres líneas con terminología del cliente, de tal manera que se creen pruebas de aceptación para el Historial y permita hacer una estimación de tiempo de desarrollo del mismo.
- Se crea un plan de lanzamiento, que debe servir para crear un calendario que todos puedan cumplir y en el desarrollo hay n personas involucradas en el proyecto. Para este desarrollo se usaron como base las historias de usuario, participando el cliente en la elección de los que se desarrollarán, y según las estimaciones de tiempo de los mismos se crearán las iteraciones del proyecto.
- Se hacen pequeños lanzamientos con mucha frecuencia.
- El desarrollo se divide en iteraciones, cada una de las cuales comienza con un plan de iteración para el que se eligen las historias de usuario a desarrollar y las tareas de desarrollo.
- Las personas cambian de área para evitar cuellos de botella y fomentar la propiedad colectiva del código.
- Se cambia el proceso lo que sea necesario para adaptarlo a tu proyecto.

Debido a que la Corporación Suiche 7B no cuenta con un formato para el llenado de historias de usuarios se propuso la siguiente tabla para poder registrarlas.

Usuario	
Banco	
Cargo	
Descripción	

Tabla 2 Historia de Usuario

3.1.2.2 Diseño

- Se eligen los diseños más simples que funcionen.
- Se elige una metáfora del sistema para que el nombrado de clases, etcétera, siga una misma línea, facilitando la reutilización y la comprensión del código.
- Son realizados el diagrama de Clases del sistema y el Diagrama Entidad-Relación
- Se refactoriza. consiste en no tener miedo de cambiar un diseño o eliminar un código que ya no sirve, o al menos que ya no es claramente la mejor solución.

3.1.2.3 Codificación

- El cliente está siempre disponible, a ser posible cara a cara. La idea es que forme parte del equipo de desarrollo, y esté presente en todas las fases de XP (escribe las historia de usuario con la ayuda de los desarrolladores, participar en la elección de los que entrarán en el plan de lanzamientos, prueban pequeños lanzamientos, participar en las pruebas de funcionalidad, etc.). La idea es usar el tiempo del cliente para estas tareas en vez de para que cree una detalladísima especificación de requisitos, y evitar la entrega de un producto peor que le hará perder tiempo.
- El código se ajustará a unos estándares de codificación, asegurando la consistencia y facilitando la comprensión y refactorización del código.
- Las pruebas unitarias se codifican antes que el código en sí, haciendo que la codificación de este último sea más rápida, y que cuando se afronte la misma se tenga más claro qué objetivos tiene que cumplir lo que se va a codificar.
- La programación del código se realizará en parejas, para aumentar la calidad del mismo. En cada momento, sólo habrá una pareja de programadores integrando código.
- Se integra código y se lanza dicha integración de manera frecuente, evitando divergencias en el desarrollo y permitiendo que todo el mundo trabaje con la última versión del desarrollo. De esta manera, se evitará pasar grandes periodos de tiempo integrando el código al final del desarrollo, ya que las incompatibilidades habrán sido detectadas enseguida.
- Se usa la propiedad colectiva del código, lo que se traduce en que cualquier programador puede cambiar cualquier parte del código. El objetivo es fomentar la contribución de ideas por parte de todo el equipo de desarrollo.
- Se deja la optimización para el final.

- No se hacen horas extra de trabajo.

3.1.2.4 Pruebas

- Todo el código debe tener pruebas construidas por el programador y debe aprobarlas antes de ser lanzado.
- Cuando se encuentra un error de codificación, se desarrollan pruebas para evitar volver a caer en el mismo.
- Se realizan pruebas de aceptación, publicando los resultados de las mismas. Estas pruebas son generadas a partir de las historia de usuario elegidas para la iteración, y son "pruebas de caja negra", en las que el cliente verifica el correcto funcionamiento de lo que se está probando. Cuando se pasa la prueba de aceptación, se considera que la correspondiente historia de usuario se ha completado.

Para efectos de las pruebas de aceptación, se construyó un pequeño formato de evaluación, el cual será suministrado a un usuario de la aplicación para que lo complete y deje sentada y registrada su aprobación, en cuanto a los productos de software obtenidos luego de la implementación de cada historia de usuario.

Figura 10 Formato de Pruebas de Aceptación

3.2 Implementación

El desarrollo del SIPCA se dividió en módulos principales que en algunos casos agrupaban sub-módulos específicos. El objetivo de cada iteración es obtener una versión del sistema que incluya la implementación de las historias de usuario planteadas en la planificación del proceso de desarrollo XP. Los resultados de las iteraciones son evaluados por el cliente hasta que sean aprobados por el mismo.

Por cada iteración se aplicarán las 4 actividades definidas en la adaptación de XP, mostrando la planificación de las historias de usuario que se implementarán, el diseño con diagramas de clases, la codificación y finalmente las pruebas realizadas.

Para efectos de este trabajo de grado se presentará el sub-módulo *Crear PDC* que está contenido dentro del módulo Administrar PDC. Dicho módulo fue desarrollado a través de la metodología de desarrollo XP.

3.2.1 Planificación de Iteraciones

Para la realización del sub-módulo *Crear PDC* se realizó 1 iteración que a grandes rasgos incluye lo siguiente:

Iteración I: Implementación de la creación de PDC.

3.2.1.1 Iteración I

Fecha Inicio: 05/12/2007 **Fecha Culminación:** 11/12/2007

La iteración I consistió en la creación de una interfaz a través de la cual los usuarios puedan ingresar la información necesaria para la creación de un PDC en el sistema.

Dicha interfaz fue construida en base los requerimientos del cliente plasmados en la siguiente historia de usuario:

Usuario	U1
Banco	Exterior
Cargo	Gerente de Prevención y Control de Fraude
Descripción	Como responsable del área de fraude, me gustaría que el sistema me permitiera ingresar la información de los Puntos de Compromiso que investigo y la lista de Puntos de Aceptación por los cuales se usaron las tarjetas que fueron copiadas en ese punto de compromiso. Para el Punto de Compromiso la información a ingresar debe ser: • Identificador del Terminal que compromete las tarjetas • Banco propietario del Terminal • Tipo de Terminal, pudiendo ser ATM o POS • Ubicación geográfica del Terminal Si el tipo de Terminal es POS se debe ingresar MCC, nombre y código de afiliado del comercio propietario del POS.

Tabla 3 Historia de Usuario Nº 1

A su vez fue requerida otra funcionalidad en el módulo *Crear PDC*, la cual se especifica en la siguiente historia de usuario:

Usuario	U2
Banco	De Venezuela
Cargo	Gerente de Prevención de Fraude
Descripción	Cuando ingreso un PDC el sistema debe indicarme si ese mismo PDC ya fue ingresado por otro Banco, en ese caso, mi información debería ser anexada a la información ya existente ingresada por los otros bancos

Tabla 4 Historia de Usuario Nº 2

Por otra parte también fue requerido ingresar información de los PDAs asociados al PDC que esté creando en ese momento, este requerimiento quedó plasmado en la siguiente historia de usuario:

Usuario	U2
Banco	Occidental de Descuento
Cargo	Coordinador de Comisión de Fraude
Descripción	Cada PDC debe tener asociados PDAs representados por al menos 3 tarjetas distintas, en caso contrario no se cuenta con información relevante y concluyente que nos indique que efectivamente se está ante un PDC. Un PDA debe tener al menos la siguiente información: • Identificador del Terminal que compromete las tarjetas • Banco propietario del Terminal • Ubicación geográfica del Terminal • Numero de tarjeta comprometida • Fecha y hora de la Transacción • MCC de la transacción A su vez podría incluirse los campos de Código de Afiliado y Nombre de Comercio en caso de que apliquen.

Tabla 5 Historia de Usuario Nº 3

Diseño

Para realizar la inserción de un nuevo PDC en el sistema se realizan una serie de pasos en los cuales interactúan conjuntamente realizando pequeñas tareas que preparan la información obtenida desde la interfaz. Esta interacción se ve reflejada el siguiente diagrama WAE del sub-módulo Crear PDC.

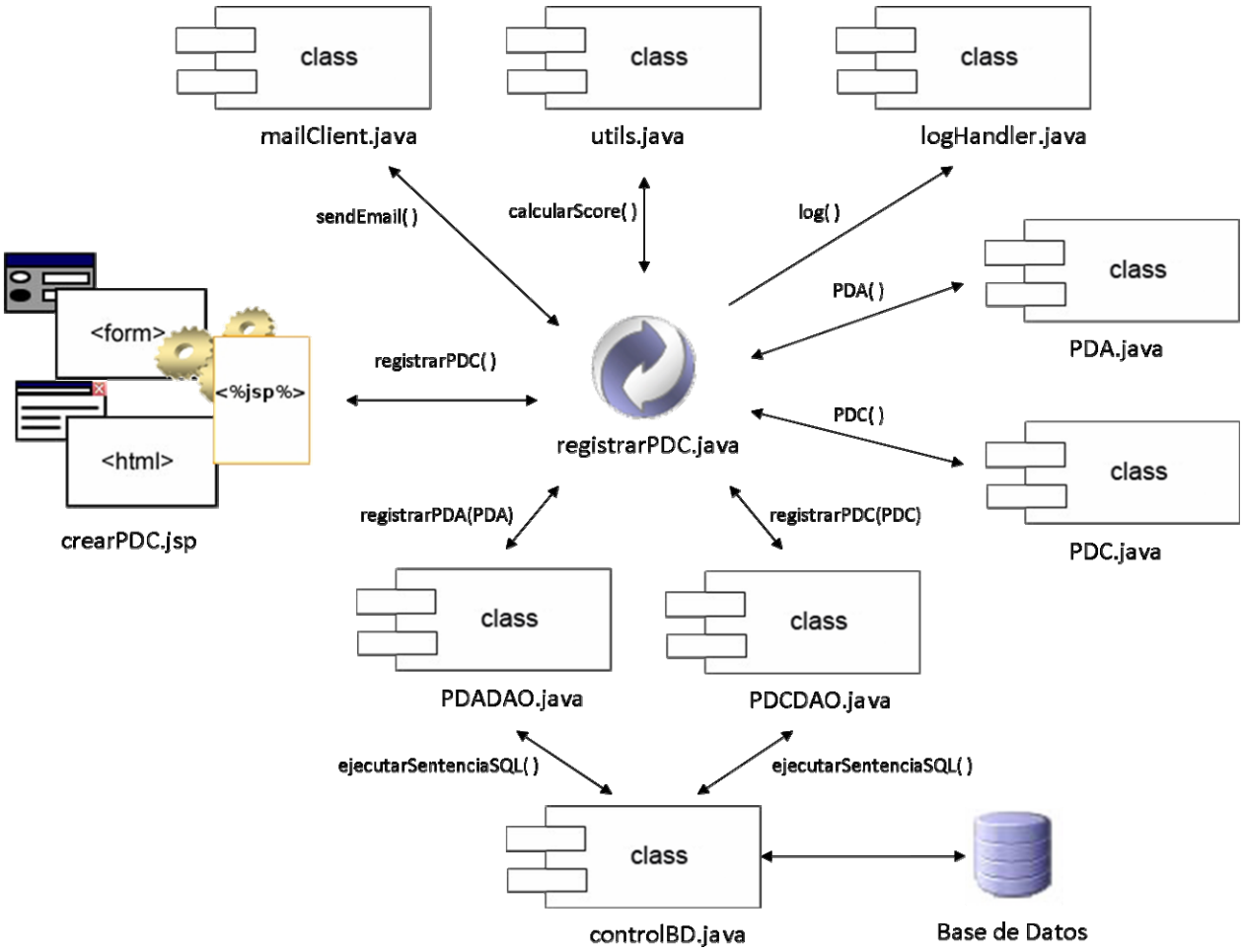

Figura 11 Diagrama WAE - Funcionalidad Crear PDC

Una vez obtenidos los datos necesarios para crear el PDC se verifica si ya se encuentra registrado en el sistema, en caso de que se encuentre registrado únicamente se agregaran los nuevos PDAs a dicho PDC ya existente, así como también se actualizará su Score.

Luego de esta verificación se realiza el procesamiento de la información de los PDAs del PDC, los cuales son almacenados como objetos en una lista de PDAs; dichas operaciones de procesamiento se realizan a través de métodos de la clase *utils*.

Asimismo se verifica si el Store del PDC superó los 50 puntos, en caso de que esto ocurra se envía un correo electrónico a los usuarios del sistema informando dicho evento, el envío de correo electrónico se realiza a través de la clase *mailClient*.

Una vez que toda la información está lista para ser guardada se invoca a los métodos *registrarPDC* y *registrarPDA* de las clases *PDCDAO* y *PDADAO* respectivamente. Dichos métodos se encargan de realizar la inserción de la base de datos, esto lo hacen invocando al método *ejecutarSentenciaSQL* de la clase *controlBD*.

Finalmente se le indica al usuario a través de un mensaje el resultado del proceso de registro del PDC. En caso de ser exitoso se le indica el número de identificación único del PDC en el sistema, el cual le servirá para realiza búsquedas directas de dicho PDC cuando así lo desee. A su vez, si el PDC ya estaba registrado en el sistema se le indica a través del mensaje de resultado del proceso de registro.

Codificación

Las funcionalidades correspondientes a la historia de usuario I forman parte del sub-módulo Crear PDC del sistema SIPCA cuya interfaz para la historia de usuario Nº 1 es la que se muestra en la Figura Nº 3.4.

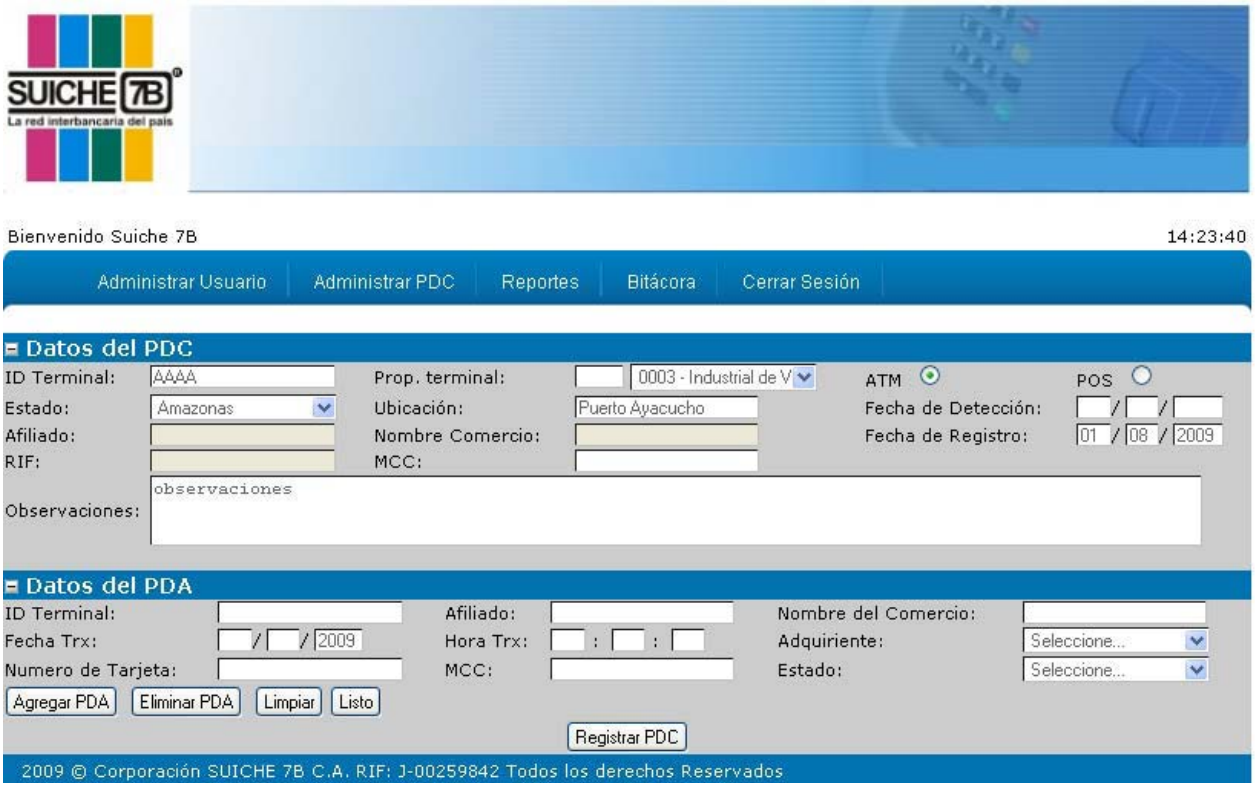

Figura 12 Interfaz Crear PDC

Extractos del algoritmo implementado para este sub-módulo, se muestra en las Figuras Nº 13, 14 y 15 respectivamente.

```
existePDC = PDCDAO.existePDC(idTerminal,terminalBanco);
if(existePDC==0){
    //PDC no registrado anteriormente
    confirmadoPDC = 0;
    bancosAsociados = 1;
    score = utils.calcularScore(bancosAsociados,PDAList,rangoRiesgo);

    if(score > 50){
        emailNotificacion=1;
    }
```

Figura 13 Verificación de existencia del PDC en el sistema y Cálculo de Score en caso de tratarse de un nuevo PDC

```
pdcBusq = new PDC(  idPDC, nombrePDC, estado, ubicacion, fechaDeteccion, emisorReporta, cantidadSemanas,
                    idTerminal, terminalBanco, tipoTerminal, mccPDC, afiliado, nombreComercio, rif,
                    bancosAsociados, PDAList.size(), PDAList, score, cantidadSemanas, observaciones,
                    primeraTarjeta, fechaRegistro, emailNotificacion );

rc = PDCDAO.registrarPDC(pdcBusq);
```

Figura 14 Creación de PDC y registro en el sistema

```
if(pdcResult.getScore() > 50 && pdcResult.getEmailNotificacion()==1){
    try {
        MailClient.sendMail(pdcResult,usuarioDAO.getListaEmail());
    }
    catch(Exception e){
        e.printStackTrace();
    }
}

pdaError = PDADAO.registrarPDA(PDAList,referenciaIdPDC);
```

Figura 15 Envío de correo y registro de PDAs asociados al PDC

Pruebas

Los resultados de las pruebas unitarias correspondientes a la iteración I fueron los siguientes (Ver Tabla Nº 6).

Nombre de la Prueba	Objetivos	Resultados
Validar campos Obligatorios	Verificar que los campos obligatorios sean llenados antes de enviar el formulario.	Es mostrado un mensaje de notificación cuando se intenta crear un PDC o PDA con datos obligatorios sin suministrar
Validar fechas	Verificar que las fechas de las transacciones de los PDAs no sean más antiguas que la fecha de detección del PDC.	Es mostrado un mensaje de notificación indicando que la fecha de los PDA no puede ser más antigua que la fecha de detección del PDC.
Verificar cantidad de tarjetas en PDAs	Constatar que los PDAs suministrados corresponden a al menos tres (3) tarjetas distintas.	Es mostrado un mensaje de notificación indicando que los PDAs deben estar representados por al menos tres (3) tarjetas distintas.
Validar existencia de PDAs	Verificar que los PDAs no hayan sido registrados en el sistema previamente.	Es mostrado un mensaje de notificación indicando que dicho PDA ya fue registrado y que no puede ser re-ingresado.
Resultado de Operación	Verificar que sea presentado al usuario un mensaje de respuesta	Si la creación del PDC es exitosa es indicado el éxito del registro y el número de identificación del PDC en el sistema. En caso contrario es indicado un código de falla

Tabla 6 Pruebas Unitarias Iteración I – Crear PDC

A continuación la Figura Nº 16 se presenta la evaluación realizada para la historia de usuario correspondiente a este Sub Módulo.

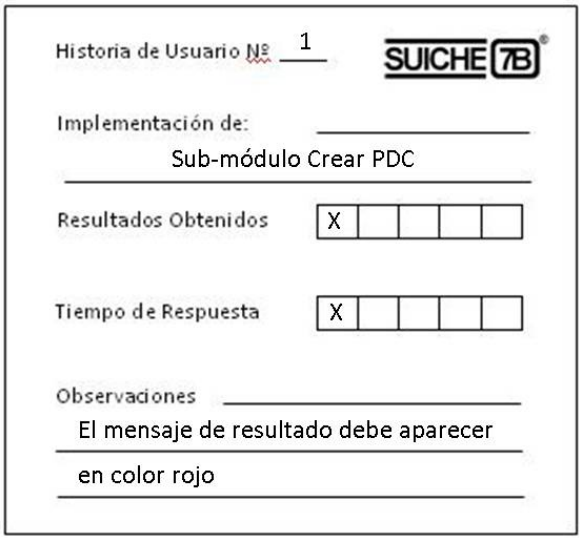

Figura 16 Evaluación de la Iteración I

3.3 Base de Datos del Sistema

La base de datos del sistema SIPCA se encuentra desarrollada en el SMBD IBM DB2 UDB 8.0 licenciado por IBM para la Corporación Suiche 7B.

La base de datos de la aplicación esta contenida en un *tablespace* único para ella denominado TSSIPCA y la información en la tiene una vida útil de 180 días continuos.

La base de datos del sistema es representada en el siguiente diagrama Entidad Relación:

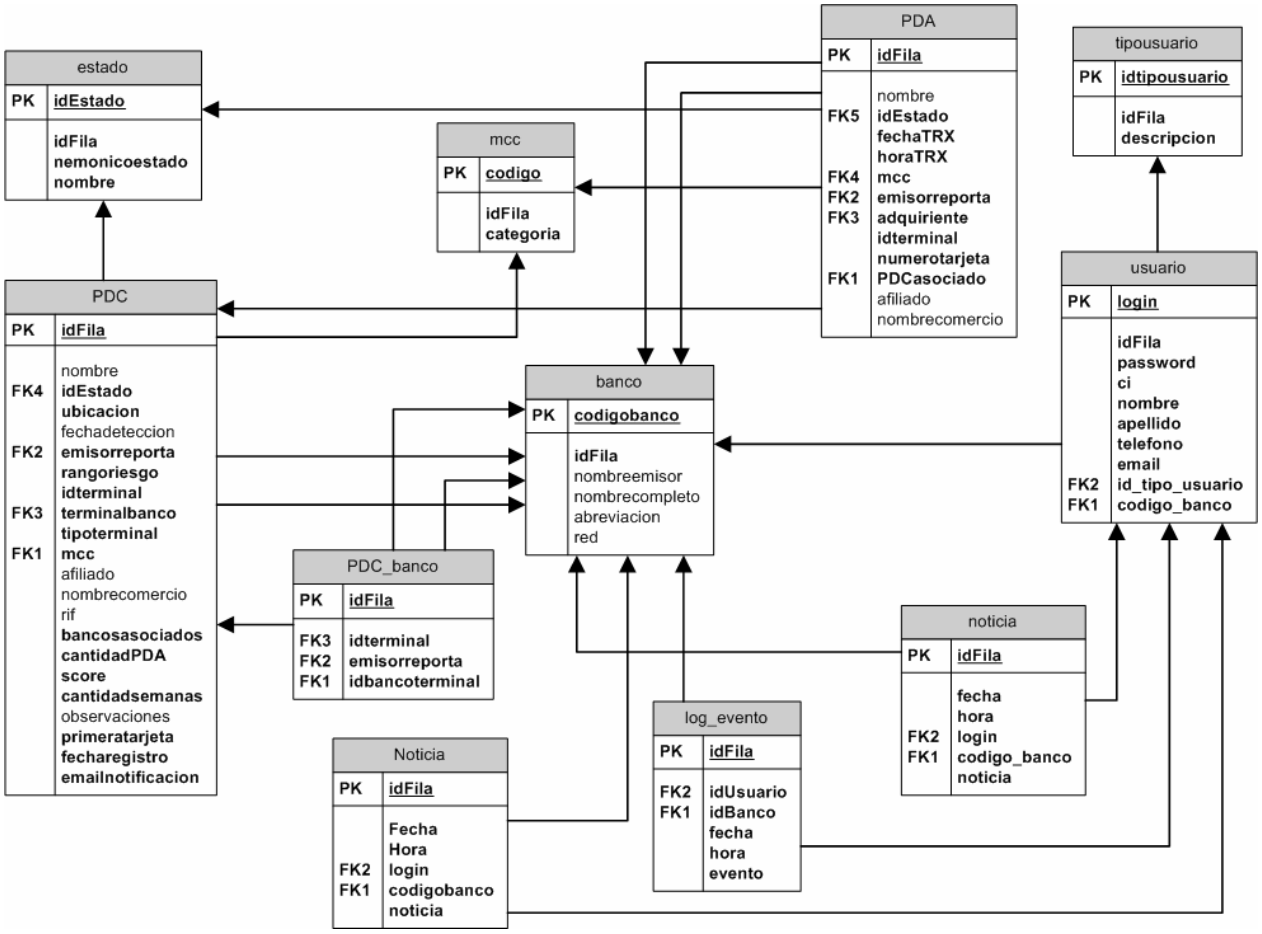

Figura 17 Diagrama Entidad Relación - Base de Datos SIPCA

3.3.1 Tablas de la Base de Datos del Sistema

- **Tabla BANCO:** Esta tabla contiene la información respectiva de cada uno de los bancos del país.

Columna	Tipo	Longitud	Nulo	Caracteristicas
idFila	integer		NO	Identificador único de cada entrada de la tabla
codigobanco	varchar	4	NO	Código ABA del banco
nombreemisor	varchar	100	NO	Nombre corto del banco
nombrecompleto	varchar	100	NO	Nombre Completo del banco
abreviación	varchar	25	NO	Siglas del banco
red	varchar	3	NO	Red a la que pertenece el Banco

Tabla 7 Tabla Banco – Base de Datos SIPCA

- Tabla ESTADO: Esta tabla contiene la información respectiva de cada uno de los Estados del país.

Columna	Tipo	Longitud	Nulo	Caracteristicas
idFila	integer		NO	Identificador único de cada entrada de la tabla
idEstado	integer		NO	Identificador único de estado
nemonicoEstado	varchar	2	NO	Nombre nemónico del estado
Nombre	varchar	30	NO	Nombre del estado

Tabla 8 Tabla Estado - Base de Datos SIPCA

- Tabla PDA: Esta tabla contiene la información respectiva de cada uno de los PDAs que han sido registrados en el sistema.

Columna	Tipo	Longitud	Nulo	Características
idFila	integer		NO	Identificador único de cada entrada de la tabla
nombre	varchar	255	NO	Nombre del PDA
idEstado	integer		NO	Estado donde se encuentra el PDA
fechaTRX	date		NO	Fecha en que fue detectado el PDA
horaTRX	Time		NO	Hora en que fue detectado el PDA
Mcc	varchar	4	NO	Merchant Category Code
emisorReporta	varchar	4	NO	Banco que reporta el PDA
Adquiriente	varchar	4	NO	Banco que recibe la operación
Idterminal	varchar	8	NO	Código único de identificación del terminal
numeroTarjeta	varchar	19	NO	Número de tarjeta
PDCasociado	Integer		NO	Identificador del PDC al cual está asociado
Afiliado	varchar	10	NO	Código del afiliado dueño del POS
nombreComercio	varchar	40	NO	Nombre del comercio dueño del POS

Tabla 9 Tabla PDA - Base de Datos SIPCA

- Tabla PDC: Esta tabla contiene la información respectiva de cada uno de los PDAs que han sido registrados en el sistema.

Columna	Tipo	Longitud	Nulo	Características
idFila	integer		NO	Identificador único de cada entrada de la tabla
nombre	integer		NO	Nombre asignado al PDC
idEstado	varchar	2	NO	Estado donde se encuentra el PDC
ubicación	varchar	30	NO	Dirección donde se encuentra el PDC
fechaDeteccion	varchar	10	NO	Fecha de detección del PDC
emisorReporta	varchar	4	NO	Código ABA del banco que reporta el PDC
rangoRiesgo	integer		NO	Nro. de semanas del periodo de exposición
Idterminal	varchar	8	NO	Identificador del terminal
terminalBanco	varchar	4	NO	Código ABA del Propietario del Terminal
tipoTerminal		3	NO	Tipo de terminal ATM o POS
Mcc	varchar	4	NO	Merchant Category Code
Afiliado	varchar	10	NO	Código de Afiliado del Comercio
nombreComercio	varchar	40	NO	Nombre del Comercio
Rif	varchar	11	NO	Rif del Comercio
bancosAsociados	integer		NO	Nro. de bancos que han reportado el PDC
cantidadPDA	integer		NO	Nro. de PDA's asociados al PDC
Score	integer		NO	Score asignado al PDC
cantidadSemanas	integer		NO	Nro. de semanas del periodo de exposición
Observaciones	varchar	255	NO	Observaciones del PDC
primeraTarjeta	date		NO	Fecha de la tarjeta mas antigua asociada al PDC
fechaRegistro	date		NO	Fecha de Registro del PDC en el sistema
emailNotificacion	integer		NO	Email de notificación enviado

Tabla 10 Tabla PDC - Base de Datos SIPCA

- Tabla USUARIO: Esta tabla contiene la información respectiva de cada uno de los usuarios que han sido registrados en el sistema.

Columna	Tipo	Longitud	Nulo	Caracteristicas
idFila	integer		NO	Identificador único de cada entrada de la tabla
Login	varchar	15	NO	Nombre del usuario único dentro del sistema
Password	varchar	8	NO	Contraseña del usuario para el ingreso al sistema
CI	Integer		NO	Cedula de identidad del usuario
Nombre	varchar	25	NO	Nombre del usuario
Apellido	varchar	25	NO	Apellido del usuario
Teléfono	varchar	11	NO	Teléfono del usuario
Email	varchar	50	NO	Correo electrónico del usuario
Id_Tipo_Usuario	varchar	6	NO	Identificador del tipo de usuario
Codigo_Banco	varchar	4	NO	Código del banco al que pertenece el usuario

Tabla 11 Tabla Usuario - Base de Datos SIPCA

- Tabla TIPOUSUARIO: Esta tabla contiene la información respectiva de cada uno de los usuarios que han sido registrados en el sistema.

Columna	Tipo	Longitud	Nulo	Caracteristicas
idFila	integer		NO	Identificador único de cada entrada de la tabla
idTipoUsuario	varchar	6	NO	Categoría de usuario dentro del sistema
descripción	varchar	30	NO	Descripción del tipo de usuario

Tabla 12 Tabla TipoUsuario - Base de Datos SIPCA

- Tabla LOG_EVENTO: Esta tabla contiene la información de todos los movimientos y operaciones que han realizado cada uno de los usuarios sobre el sistema, así como las notificaciones enviadas por el mismo

Columna	Tipo	Longitud	Nulo	Caracteristicas
idFila	integer		NO	Identificador único de cada entrada de la tabla
idUsuario	varchar	15	NO	usuario que desencadeno el evento u operación
idBanco	varchar	4	NO	Entidad que desencadeno el evento u operación
Fecha	date		NO	Fecha del evento u operación
Hora	time		NO	Hora del evento u operación
evento	varchar	255	NO	Descripción del evento u operación realizada

Tabla 13 Tabla Log_Evento - Base de Datos SIPCA

- Tabla MCC: Esta tabla contiene la lista de Merchant Category Codes o MCC

Columna	Tipo	Longitud	Nulo	Caracteristicas
idFila	integer		NO	Identificador único de cada entrada de la tabla
código	varchar	4	NO	Código de tipo de comercio
categoría	varchar	255	NO	Categoría del comercio

Tabla 14 Tabla MCC - Base de Datos SIPCA

- Tabla NOTICIA: Esta tabla contiene la información de todos los terminales de la red Suiche 7B

Columna	Tipo	Longitud	Nulo	Caracteristicas
idFila	integer		NO	Identificador único de cada entrada de la tabla
Fecha	date		NO	Fecha de publicación de la Noticia
Hora	time		NO	Hora de publicación de la Noticia
Login	integer	15	NO	Login del usuario que publicó la noticia
Codigo_banco	varchar	4	NO	Banco del usuario que publicó la noticia
noticia	varchar	3500	NO	Texto o contenido de la noticia

Tabla 15 Tabla ATM - Base de Datos SIPCA

3.4 Arquitectura Funcional del Sistema

La aplicación Web fue desarrollada siguiendo el patrón Modelo Vista Controlador, a continuación se enumeran los diferentes componentes.

3.4.1 Vistas o Interfaces

- Index.jsp: Página de ingreso al sistema.
- Home.jsp: Página principal.
- adminbBarra.jsp: Menú de opciones del usuario de tipo Supervisor Banco
- adminpBarra.jsp: Menú de opciones del usuario de tipo Administrador Suiche 7B
- user1Barra.jsp: Menú de opciones del usuario de tipo Analista Banco
- crearUsuario.jsp: Página del Sub-módulo Crear de Usuarios
- gestionarUsuario.jsp: Página de la funcionalidad Buscar Usuario del Sub-módulo Gestionar Usuarios
- detalleUsuario.jsp: Página de la funcionalidad Modificar Usuario del Sub-módulo Gestionar Usuarios
- crearPDC.jsp: Página del Sub-módulo Crear PDC
- gestionarPDC.jsp: Página de la funcionalidad Buscar PDC del Sub-módulo Gestionar PDC.
- detallePDC.jsp: Página de la funcionalidad Modificar PDC del Sub-módulo Gestionar PDC.
- reportes.jsp: Página del Módulo de Reportes.
- log.jsp: Página del Módulo de Bitácora.
- publicarNoticia.jsp: Página del sub-módulo Publicar Noticia.
- mcc.jsp: Página del sub-módulo Consulta MCC.

3.4.2 Paquete de Controladores

- buscarPDC.java: Procesa la información recibida desde la vista gestionarPDC.jsp.
- cambiarPassword.java: Procesa el cambio de la contraseña por defecto asignada a los nuevos usuarios del sistema.
- detallePDC.java: Procesa la información recibida desde la vista gestionarPDC.jsp.
- detalleUsuario.java: Procesa la información recibida desde la vista gestionarUsuario.jsp.
- gestionarMCC.java: Procesa la información recibida desde la vista mcc.jsp.
- gestionarUsuario.java: Procesa la información recibida desde la vista buscarUsuario.jsp.
- procesarLog.java: Procesa la información recibida desde la vista log.jsp.
- procesarReporte.java: Procesa la información recibida desde la vista reportes.jsp.
- publicarNoticia.java: Procesa la información recibida desde la vista publicarNoticia.jsp.
- registrarPPDC.java: Procesa la información recibida desde la vista crearPDC.jsp.
- registrarUsuario.java: Procesa la información recibida desde la vista crearUsuario.jsp.
- updatePDC.java: Procesa la información recibida desde la vista detallePDC.jsp.
- updateUsuario.java: Procesa la información recibida desde la vista detalleUsuario.jsp.
- verificarAcceso.java: Procesa la información recibida desde la vista index.jsp.

3.4.3 Paquete de Acceso al Modelo

- controlBD.java: Clase donde están definidos los métodos de comunicación con la Base de Datos.
- MCCDAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto MCC.
- PDADAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto PDA.
- PDCDAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto PDC.
- bancoDAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto banco.
- estadoDAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto estado.
- eventoDAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto evento.
- noticiaDAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto noticia.
- tipoUsuarioDAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto tipoUsuario.
- usuarioDAO.java: Define los métodos de acceso a Base de Datos inherentes al objeto usuario.

3.4.4 Paquete de Estructuras de Datos

- MCC.java: Objeto que representa a un MCC en el sistema.
- PDA.java: Objeto que representa a un PDA en el sistema.
- PDC.java: Objeto que representa a un PDC en el sistema.
- banco.java: Objeto que representa a un banco en el sistema.
- estado.java: Objeto que representa a un estado en el sistema.
- evento.java: Objeto que representa a un evento en el sistema.
- noticia.java: Objeto que representa a una noticia en el sistema.
- tipoUsuario.java: Objeto que representa a un tipoUsuario en el sistema.
- usuario.java: Objeto que representa a un usuario en el sistema.

3.4.5 Paquete de Utilidades del Sistema

- utils.java: Agrupa métodos de utilidad que son invocados en diferentes controladores del sistema.
- logHandler.java: Clase encargada del registro de posibles errores del sistema en logs de texto.
- mailClient.java: Clase encargada del envío de correos electrónicos a los usuarios del sistema
- smtpAuthenticator.java: Clase encargada de realizar la autenticación contra el servidor de correos de la Corporación Suiche 7B.

3.5 Principales Funcionalidades del Sistema

La aplicación Web está divida en una serie de módulos a los que accederán los diferentes usuarios de acuerdo a su jerarquía. Algunos módulos están compuestos por sub-módulos con el fin de no concentrar todas las funcionalidades en un sólo componente, de esta forma se logra mayor simplicidad en la presentación del sistema al usuario.

La estructura general del sistema es la siguiente:

- Módulo Administrar Usuario
    - Sub-módulo Registrar Usuario
    - Sub-módulo Gestionar Usuario
- Módulo
    - Sub-módulo Crear PDC
    - Sub-módulo Gestionar PDC
- Módulo de Reportes
- Módulo de Bitácora
- Módulo Otros
    - Sub-módulo Publicar Noticia
    - Sub-módulo Consulta MCC

Como se indicó anteriormente la disponibilidad de los diferentes módulos varía de acuerdo a la jerarquía del usuario en el sistema, en la siguiente tabla se resume la disponibilidad de los diferentes módulos.

Módulo	Sub-módulo	Funcionalidad	Usuario		
			Admin. Suiche 7b	Supervisor Banco	Analista Banco
Administrar Usuario	Registrar Usuario	Creación de Usuario	SI	NO	NO
	Gestionar Usuario	Consultar Usuario	SI	NO	NO
		Modificar Usuario	SI	NO	NO
		Eliminar Usuario	SI	NO	NO
Administrar PDC	Crear PDC	Creación de PDC	SI	SI	SI
	Gestionar PDC	Consultar PDC	SI	SI	SI
		Modificar PDC	SI	SI	SI
		Eliminar PDC	SI	SI	NO
Reportes	No Aplica	Generar Reportes	SI	SI	SI
		Exportar Reportes	SI	SI	SI
		Generar Gráficos	SI	SI	SI
		Exportar Gráficos	SI	SI	SI
Bitácora	No Aplica	Consultar Bitácora	SI	NO	NO
Otros	Publicar Noticia	Publicar Noticia	SI	SI	NO
	Consulta MCC	Consultar MCC	SI	SI	SI

Tabla 16 Funcionalidades del Sistema y Permisología de Acceso por tipo de Usuario

3.5.1 Sub-módulo Registrar Usuario

3.5.1.1 Descripción

En este módulo se realiza la creación de nuevos Usuarios del Sistema, suministrando para ello una serie de datos (todos ellos obligatorios), como lo son:

- **Login:** Nombre del usuario para acceder al sistema.
- **Password:** Contraseña del usuario para acceder al sistema, sólo puede contener letras, números y guiones bajos.
- **Tipo de Usuario:** Identificador de la jerarquía del usuario dentro del Sistema, entre los tipos de usuario se encuentran:
    - Administrador SUICHE 7B.
    - Supervisor Banco.
    - Analista Banco.
- **Nombre:** Nombre Real del Usuario.
- **Apellido:** Apellido del Usuario.
- **Cédula:** Cédula de Identidad del Usuario.
- **Teléfono:** Número de Teléfono del Usuario.
- **Email:** Correo electrónico del Usuario (En caso de ser administrador SUICHE 7B o Supervisor Banco los correos electrónicos de alertas serán enviados a esta dirección de correo.
- **Banco:** Entidad Bancaria u organización a la cual pertenece el Usuario.

Figura 18 Interfaz de Sub-módulo Registrar Usuario

3.5.2 Sub-módulo Gestionar Usuario

3.5.2.1 Descripción

En este módulo se podrán realizar las operaciones de consulta, modificación y eliminación de usuarios en el sistema.

En primer lugar se presenta la pantalla de búsqueda de usuario, la cual suministra distintos criterios de búsqueda que pueden ser utilizados solos o combinados para dicho proceso.

Figura 19 Interfaz de Sub-módulo Gestionar Usuario

Una vez suministrados los criterios de búsqueda, se muestran en la parte inferior de la zona de datos la lista de resultados, la cual contiene los datos principales de los usuarios, así como un botón para proceder a su eliminación, si así se requiere.

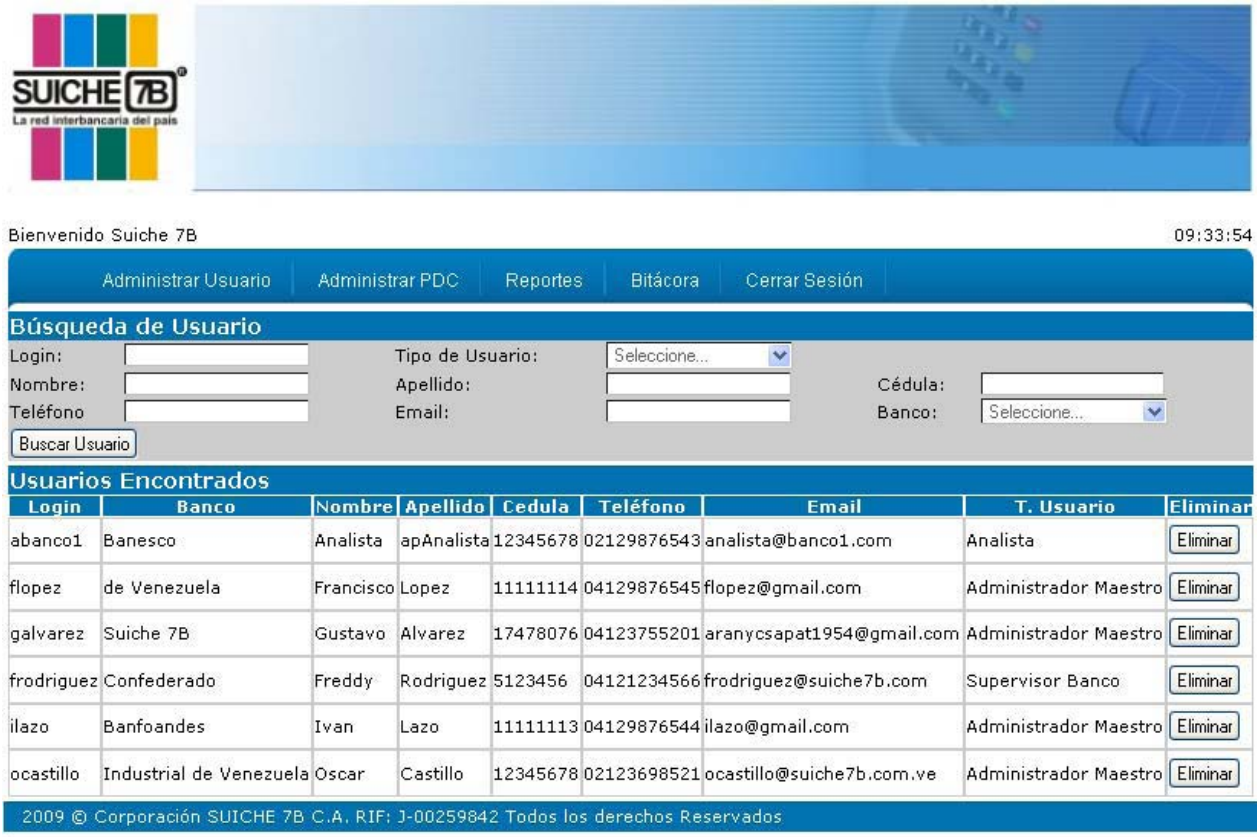

Figura 20 Interfaz de Funcionalidad de Búsqueda de Usuario

Haciendo Clic en una de las filas se accede a la pantalla de detalle del Usuario que fue seleccionado, en la misma se cargan los valores del Usuario guardados en la base de datos para consulta o modificación de los mismos, por ejemplo, cambio de contraseña.

Figura 21 Interfaz de Funcionalidad Modificar Usuario

Para proceder a la modificación o actualización de los datos de usuario una vez suministrados los mismos debe presionarse el botón “Modificar Usuario”.

3.5.3 Sub -módulo Crear PDC

3.5.3.1 Descripción

En este sub-módulo se realiza la inserción de nuevos PDC al sistema o ingreso y asociación de nuevos PDA a un PDC ya existente.

Para la creación de un nuevo PDC se debe realizar el siguiente proceso:

Suministro de datos generales del PDC

Los datos generales del PDC deben ser colocados en los campos de texto de la tabla Datos del PDC y se comprenden en:

- **ID Terminal:** Código único de identificación del Terminal.
- **Propietario del Terminal o Adquiriente:** Banco propietario del terminal que esta comprometiendo información.
- **Tipo de Terminal:** Cajero Automático (ATM) o Punto de Venta (POS).
- **Estado:** Estado de Venezuela donde se ubica el terminal (Internamente codificado en 2 caracteres definidos por SUICHE 7B para la trama ISO 8583).
- **Ubicación:** Ubicación geográfica detallada del Terminal.
- **Fecha de Detección:** Fecha de Detección del Terminal como PDC.
- **Fecha de Registro:** Fecha de Registro del PDC en el sistema.

Si el Tipo de Terminal es un Punto de Venta (POS) deben suministrarse los campos adicionales:

- **Afiliado:** Código de afiliado del Comercio Propietario del Punto de venta detectado como PDC.
- **Nombre Comercio:** Nombre del Comercio Propietario del Punto de venta detectado como PDC.
- **RIF:** Número de RIF del Comercio Propietario del Punto de venta detectado como PDC.
- **MCC:** Merchant Category Code ó Código de Categoría del Comercio, código asignado a cada comercio de acuerdo al tipo de productos o mercancía que vende.

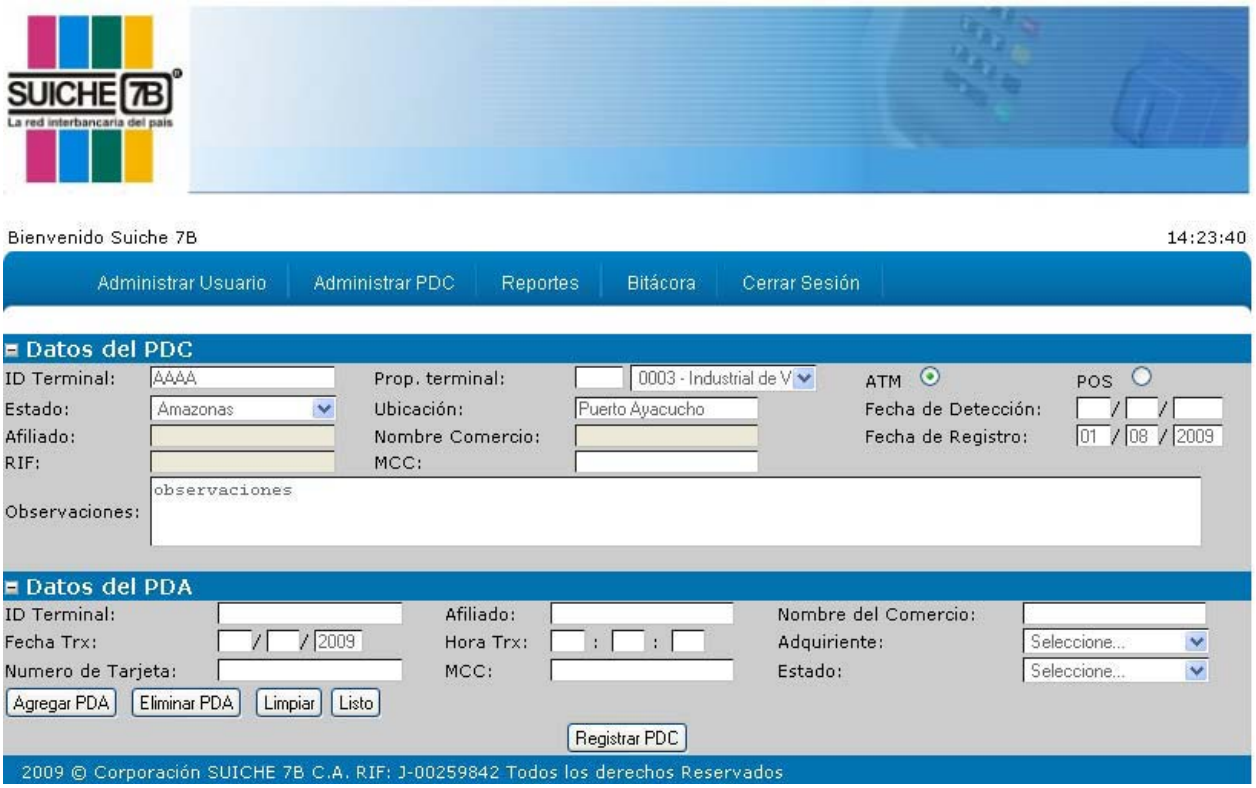

Figura 22 Interfaz de Crear PDC

Asimismo, para registrar un PDC es necesario registrar los Puntos de Aceptación (PDA) asociados a dicho PDC. Como mínimo deben registrarse transacciones de tres (03) tarjetas distintas, de esta manera se suministra información de relevante asociada al PDC.

Para registrar y asignar un PDA a un PDC se deben suministrar una serie de datos que son colocados en los campos de texto de la tabla Datos del PDA:

- **ID Terminal:** Código único de identificación del Terminal.
- **Afiliado:** Código de afiliado del comercio donde se realizó la Transacción.
- **Nombre Comercio:** Nombre del comercio donde se realizó la Transacción.
- **Fecha Trx**: Fecha de la Transacción.
- **Hora Trx:** Hora de la Transacción.
- **ID. Adquiriente:** Código del Banco Adquiriente de la Transacción.
- **MCC:** Merchant Category Code ó Código de Categoría del Comercio, código asignado a cada comercio de acuerdo al tipo de productos o mercancía que vende.
- **Número de Tarjeta**: Número de la Tarjeta sobre la cual se realizó la transacción.

- **Estado:** Estado de Venezuela donde se realizó la Transacción (Internamente codificado en 2 caracteres definidos por SUICHE 7B para la trama ISO 8583).

A medida que se agregan PDA, los mismos se van mostrando en la parte inferior de la zona de ingreso de datos de PDA.

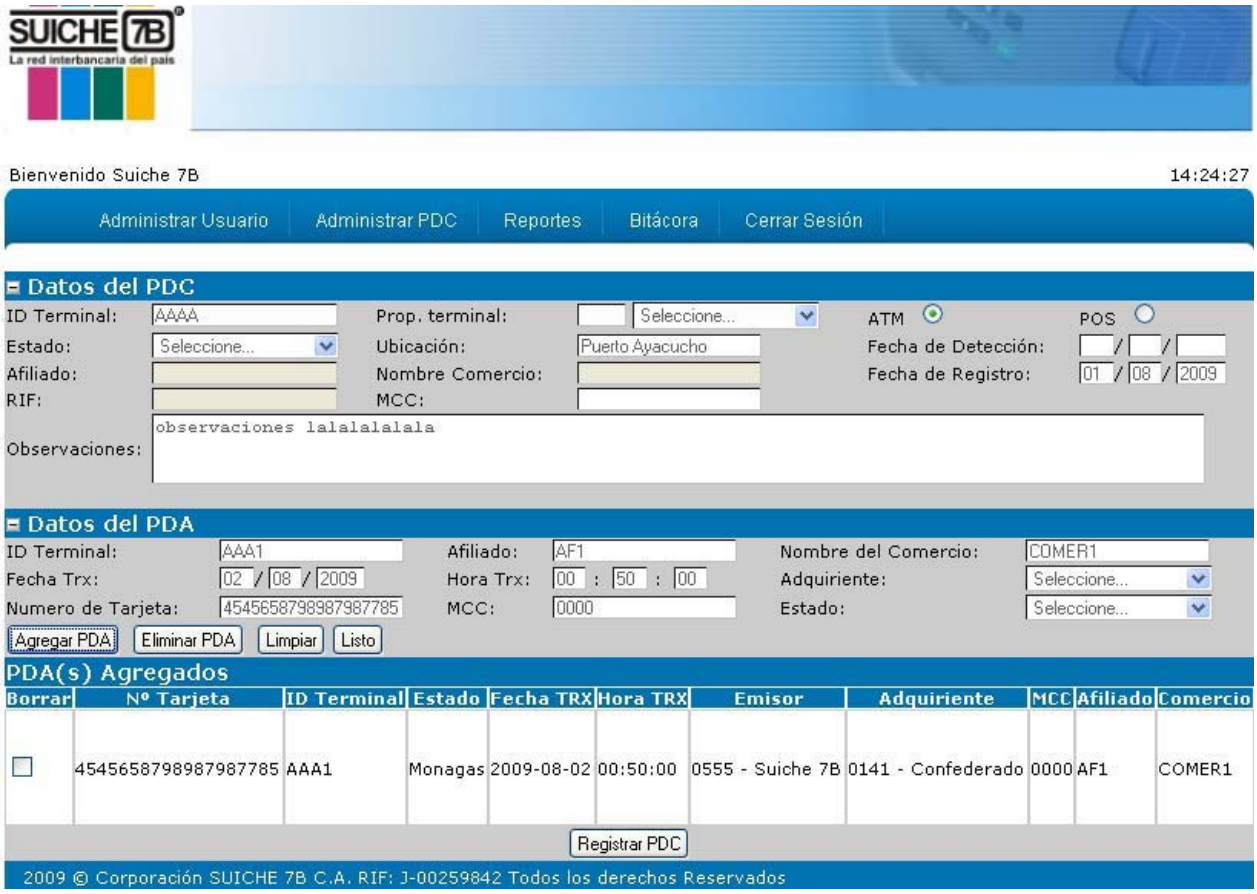

Figura 23 Interfaz de Funcionalidad Agregar PDA I

Si durante el proceso de agregar nuevos PDA al PDC, el usuario se da cuenta que cometió un error en algún o algunos de los datos de uno o varios de los PDA ya agregados puede borrarlos antes de realizar el registro, para ello selecciona el "checkbox" que se encuentra al principio de cada PDA agregado y oprime el botón Eliminar PDA, esto eliminara de la lista de PDAs agregados y todos aquellos que tengan el "checkbox" marcado.

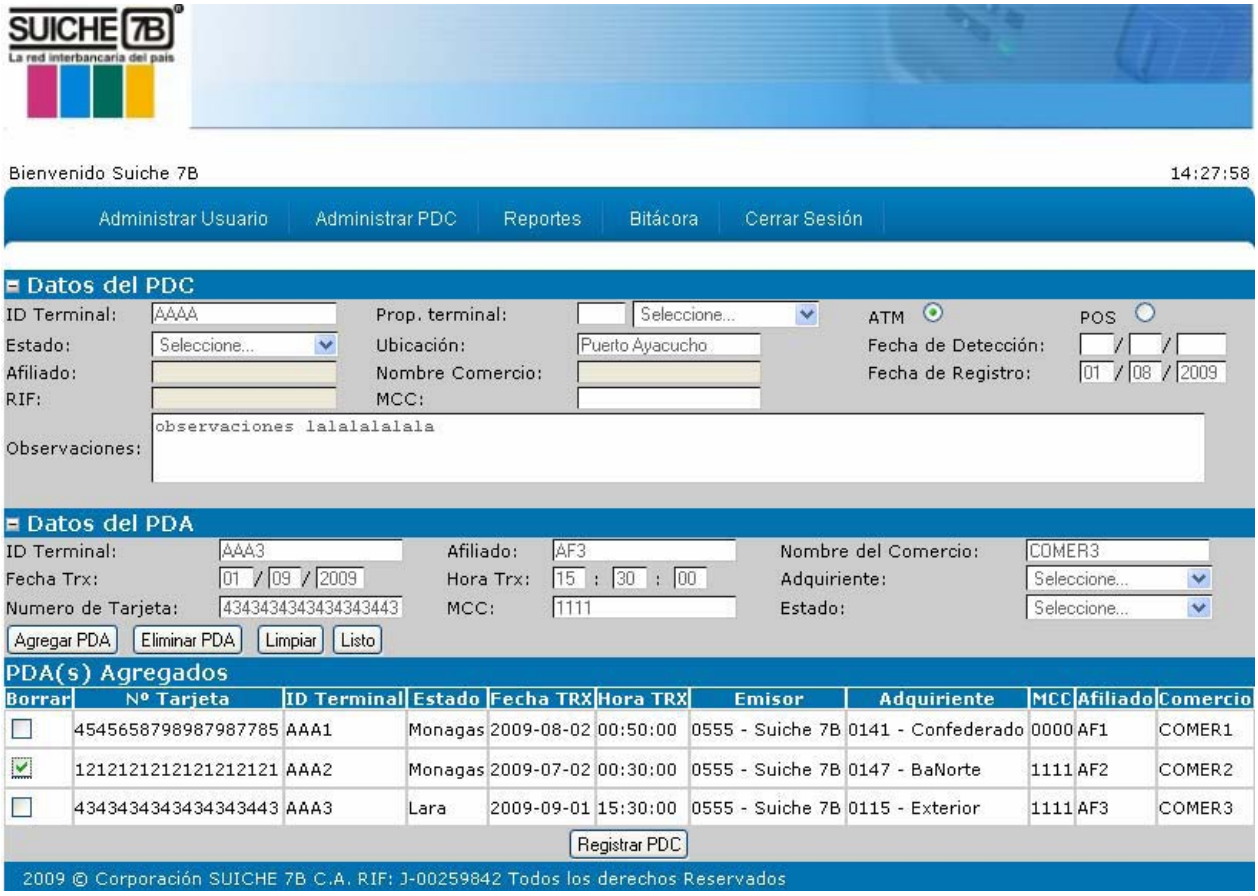

Figura 24 Interfaz de Funcionalidad Agregar PDA II

Igualmente y para comodidad del usuario existe la posibilidad de plegar y desplegar a gusto la zona de ingreso de datos del PDC y la zona de ingreso de datos de PDA, esto se realiza haciendo clic en la imagen "−" para plegar el contenido de dicha zona y haciendo clic en la imagen "+" para desplegar el contenido.

Figura 25 PDC preparado para creación

Una vez completado el proceso de suministro de datos de PDC y sus PDA se procede a registrar el PDC en el sistema, pulsando, para ello el botón Registrar PDC.

Seguidamente se mostrara un mensaje indicando el resultado del proceso de registro y de ser exitoso se mostrar el identificador asignado al PDC dentro del sistema. Este identificador es único dentro del sistema y es de ayuda para la búsqueda del PDC a la hora de realiza consultas.

3.5.4 Sub-Módulo Gestionar PDC

En este módulo se podrán realizar las operaciones de consulta, modificación y eliminación de PDC, en primer lugar se presenta la pantalla de búsqueda de PDC, la cual suministra distintos criterios de búsqueda que pueden ser utilizados solos o combinados para dicho proceso.

La búsqueda de un PDC puede realizarse de dos (2) maneras:

- Suministrando como único criterio de búsqueda su **Número de Identificación** dentro del sistema el cual es mostrado al momento de su creación o en el detalle luego de realizar la consulta.
- Suministrar una serie de criterios de que pueden ser utilizados solos o combinados para dicho proceso.

Figura 26 Interfaz Sub-módulo Gestionar PDC

Una vez suministrados los criterios de búsqueda, se muestran en la parte inferior de la zona de datos la lista de resultados que contiene los datos principales de los PDC, así como un botón para proceder a su eliminación, si así se requiere. El Botón de eliminación sólo estará disponible para los Usuarios de tipo Supervisor Banco y Administradores SUICHE 7B.

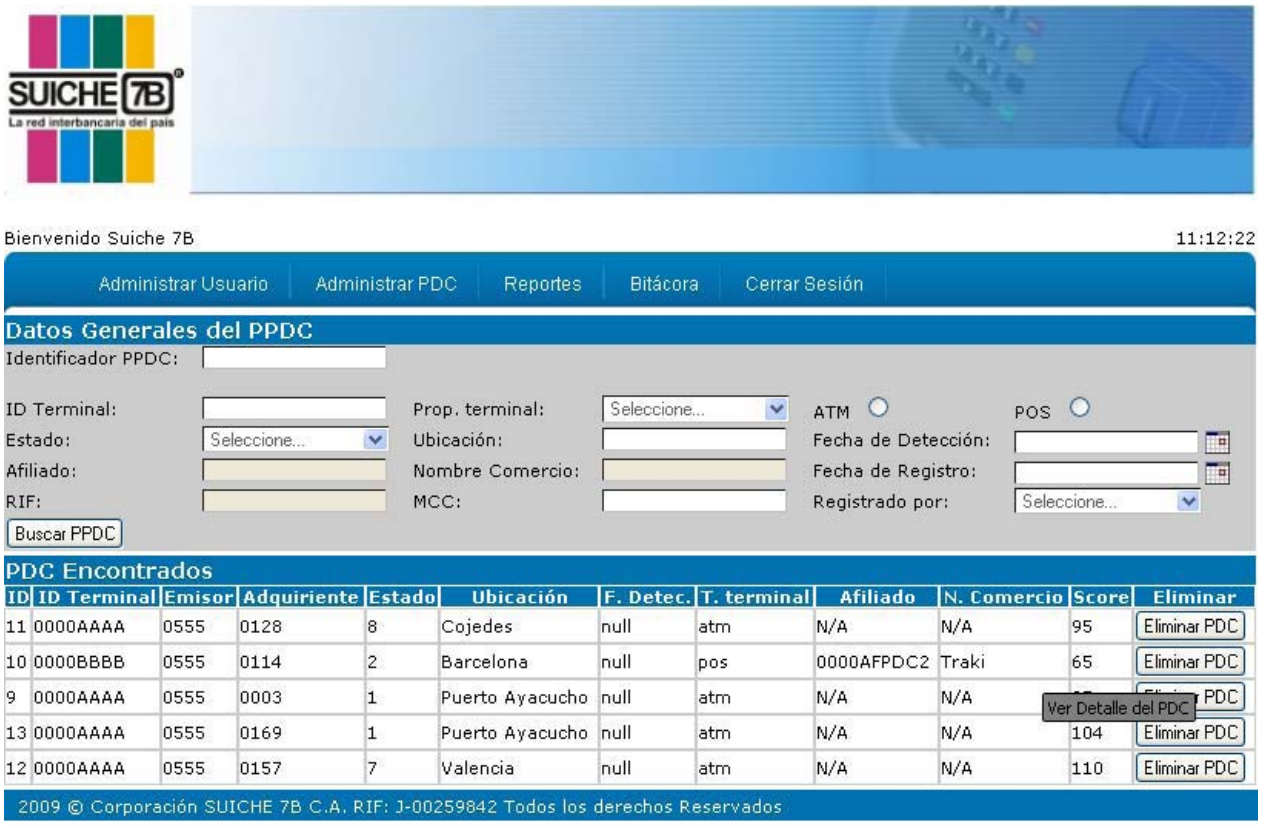

Figura 27 Interfaz de Funcionalidad Buscar PDC

Haciendo Clic en una de las filas se accede a la pantalla de detalle del PDC que se le hizo Clic, en la misma se pre-cargan los valores del PDC guardados en la base de datos para consulta o modificación de los mismos; asimismo, se cargan los PDA asociados a dicho PDC.

Para proceder a la modificación o actualización de los datos de Usuario una vez suministrados los mismos debe presionarse el Botón **Modificar PDC**.

Figura 28 Interfaz de Funcionalidad Modificar PDC

3.5.5 Módulo de Reportes

3.5.5.1 Descripción

En esta sección los usuarios podrán obtener Reportes y Gráficos basados en la información ingresada al sistema, estos se generan mediante la selección del reporte o gráfico solicitado y el suministro de un período de tiempo sobre el cual se realizará la búsqueda, en caso de no suministrar el período de tiempo se realizará el reporte o grafico solicitado en base al histórico de los datos almacenados.

Este módulo está disponible para usuarios de tipo Administrador SUICHE 7B, Supervisor Banco, Analista Banco

El período de tiempo puede ser seleccionado mediante un calendario que se despliega al hacer clic sobre la imagen junto a los campos de texto.

Entre los reportes que genera la aplicación se encuentran: PDC ATM con mayor Score.

- PDCs POS por mayor Score.
- PDCs ATM por mayor Score

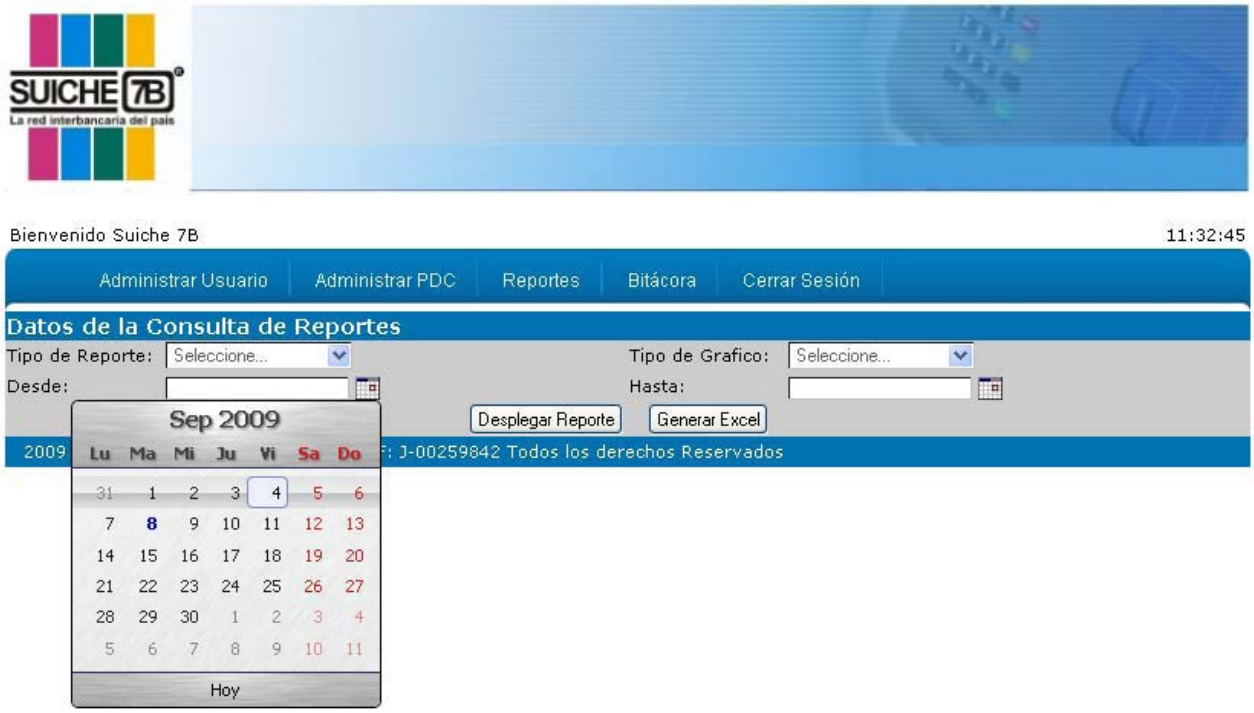

Figura 29 Interfaz de Módulo de Reportes

Los Reportes pueden obtenerse de dos (02) maneras, pueden ser desplegados en la pantalla del navegador mediante el botón Desplegar Reporte o Exportarse en un archivo .xls (Microsoft Excel), el cual puede abrirse o guardarse en su equipo.

Al presionar el botón Desplegar Reporte es mostrada la información del mismo en la pantalla del sistema.

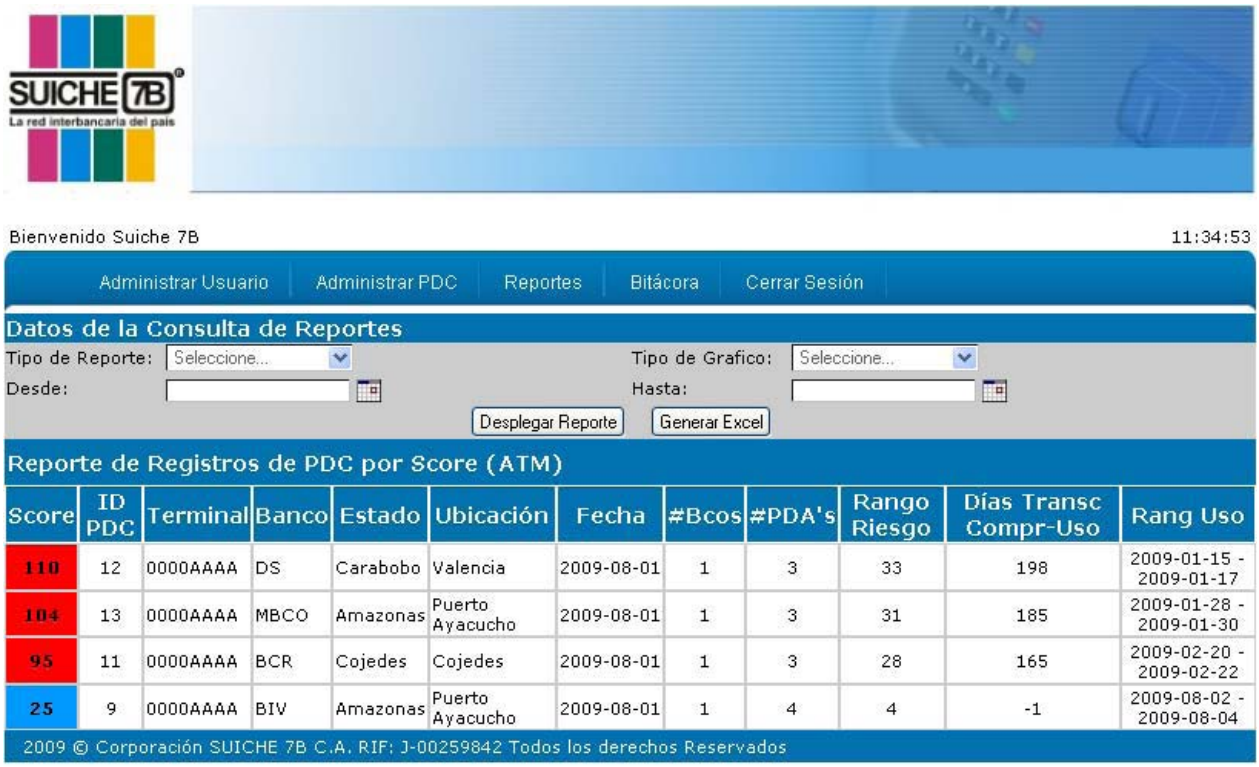

Figura 30 Interfaz de Funcionalidad Desplegar Reporte

Al Presionar el botón Generar Excel, se presenta al usuario una ventana con las opciones de abrir directamente el reporte con MS Excel o guardarlo en el computador.

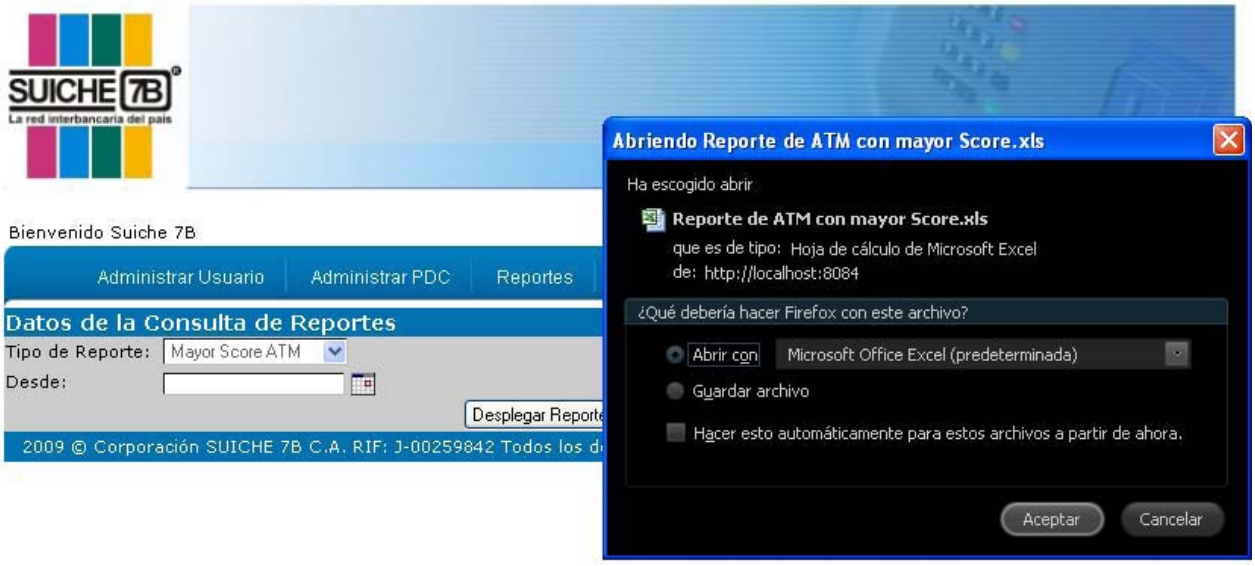

Figura 31 Interfaz de Funcionalidad de Exportar Reporte

Asimismo, la aplicación Web genera gráficos de Cantidad y Proporción para facilitar el análisis de la información registrada por parte de los usuarios.

Entre los gráficos disponibles se encuentran:

- Cantidad de PDC por tipo de comercio.
- Cantidad de PDA por tipo de comercio.
- Tipos de tarjeta comprometidas.
- PDC por estado del país.
- PDA por estado del país.
- PDA por horas de transacción.
- PDC por días de la Semana.
- PDA por días de la semana de Transacciones.
- PDC por Banco.
- PDA por Banco.
- Transacciones con tarjeta de Crédito por Tipo de Comercio.
- Transacciones con tarjeta de Débito por Tipo de Comercio.

Dichos gráficos pueden ser desplegados en la interfaz del navegador a través del botón “Desplegar Reporte” o exportados a PDF al pulsar el botón “Generar PDF”. Para cada uno de los diferentes gráficos se genera un Gráfico de Barras seguido de un Gráfico de Torta, con los diversos resultados representados en colores diferentes para mayor comprensión y con su correspondiente leyenda.

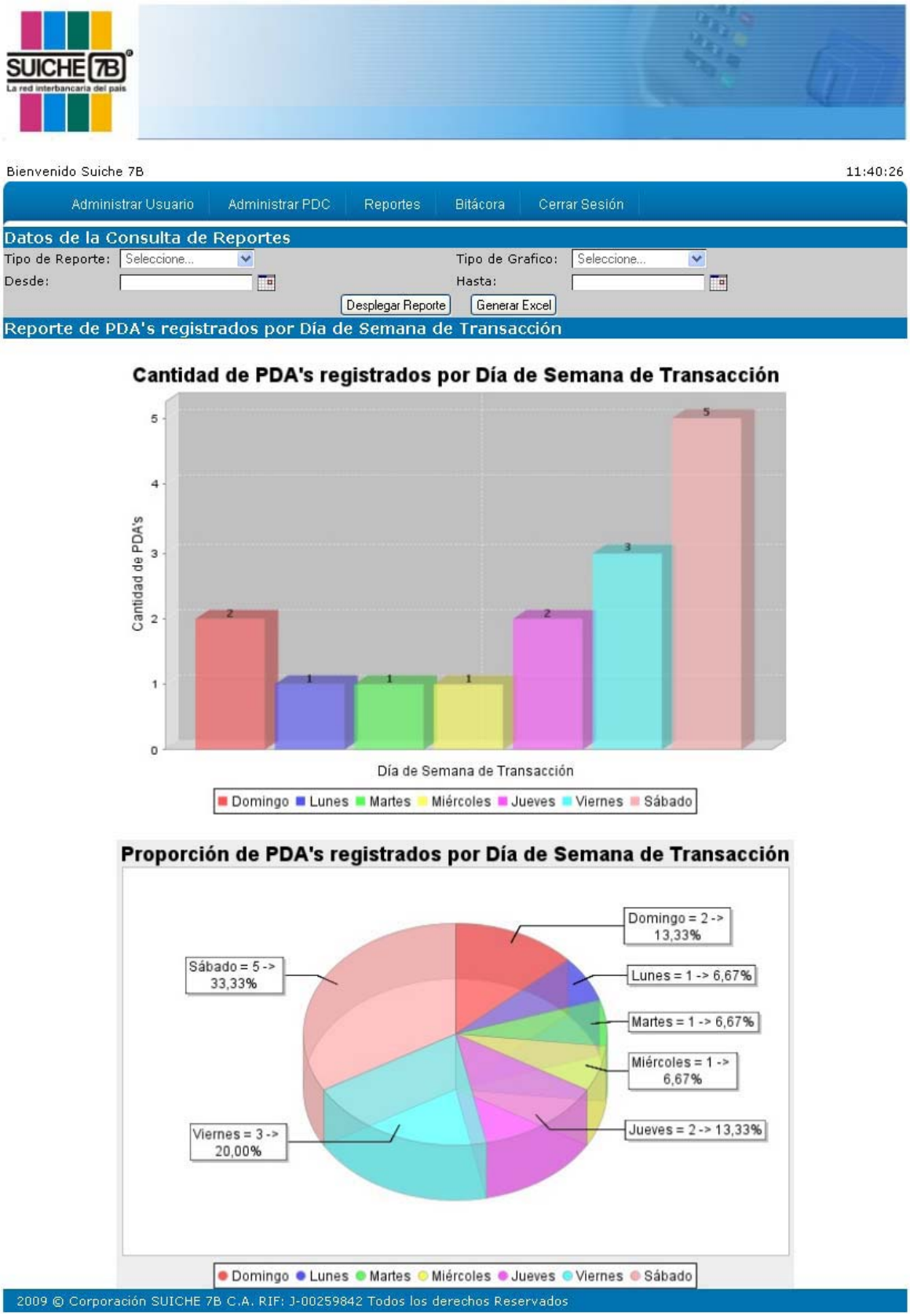

Figura 32 Interfaz de Funcionalidad Desplegar Gráfico

3.5.6 Módulo Bitácora

3.5.6.1 Descripción

En este módulo el Administrador Suiche 7B podrá consultar todas las acciones que los distintos usuarios han realizado sobre el sistema, para ello se provee de una búsqueda con criterio como período de tiempo de las acciones y bancos.

Se realizó este desarrollo debido a que el sistema debe ser auditable y por ello son registrados en base de datos ciertas acciones que realizan los usuarios en el sistema.

Figura 33 Interfaz del Módulo Bitácora

3.5.7 Sub-módulo de Consulta MCC

3.5.7.1 Descripción

A través de este sub-módulo los usuarios podrán consultar los diferentes MCC (Merchant Category Code) actualizados, los cuales son asignados a los diferentes tipos de negocio por Master Card.

Un MCC se compone de un código único de 4 dígitos el cual es asignado a las transacciones realizadas en los comercios. Dicho código viene se asigna de acuerdo a la naturaleza del comercio, como se observa en la tabla 8.

Código	Categoría del Comercio
0742	Servicios Veterinarios
0763	Cooperativa Agrícola
0780	Servicios de Jardinería
1520	Contratistas Generales
1711	Calefacción, Fontanería, Aire Acondicionado
1731	Contratistas Eléctricos
1740	Albañilería, cantería, y yeso
1750	Contratistas de Carpintería
1771	Contratistas de Concreto
3000	Aerolíneas

Tabla 17 Muestra de lista de MCC

Al momento del desarrollo dicha tabla sólo se encontraba disponible en inglés por lo cual fue traducida a fin de brindar mayor comodidad a los usuarios del sistema.

La consulta puede realizarse suministrando el código o parte de el o por la categoría.

Figura 34 Interfaz de Módulo Consulta MCC

3.5.8

Sub-módulo Publicar Noticia

3.5.8.1 Descripción

A través de este sub-módulo los usuarios podrán publicar noticias breves relacionadas con Puntos de Compromiso o Puntos de Aceptación no confirmados, con lo cual los usuarios de otras instituciones bancarias estarán alertas.

Figura 35 Interfaz de Sub-módulo Publicar Noticia

Las noticias publicadas serán mostradas en el home o página principal del sistema mediante la rotación ascendente (a modo de marquesina) de las últimas 10 noticias publicadas.

Figura 36 Página Principal del Sistema con las noticias publicadas

CONCLUSIONES Y RECOMENDACIONES

Finalmente podemos decir que se alcanzó el principal objetivo de esta investigación, al desarrollar una aplicación Web para la gestión de Puntos de Compromiso y Puntos de Aceptación para los bancos afiliados a la Corporación Suiche 7B, con todas las funcionalidades definidas por el cliente en los diferentes Comités Técnicos realizados.

Gracias a la metodología XP el proceso de desarrollo de la aplicación Web fue lo suficientemente flexible en sus diversas fases para permitir cubrir e implementar cada uno de los requerimientos en su totalidad, garantizando la oportuna y constante participación del cliente en el proceso de desarrollo como principal evaluador de cada tarea realizada. Por consiguiente, se alcanzó un alto grado de satisfacción por parte de los miembros de la Corporación Suiche 7B, quienes manifestaron su conformidad con el desempeño de la aplicación y con los resultados obtenidos. Se trata de un producto de alta calidad a pesar de la alta complejidad de información que maneja y por el alto dinamismo que presenta la aplicación, siendo capaz de manejar múltiple información relacionada con las investigaciones de Fraude y proveyendo reportes y gráficos especializados basados en dicha información.

También en el desarrollo la arquitectura planteada fue altamente modular haciendo de ésta, una aplicación de fácil mantenimiento y escalabilidad, distribuyendo en algoritmos más pequeñas las partes más complejas del sistema evitando así cargas adicionales a la base de datos y a las tecnologías del lado del cliente. Asimismo se sobrepasaron las expectativas del cliente, al incorporar herramientas de gran utilidad para las tareas de investigación de las Comisiones de Fraude de las diferentes instituciones bancarias como lo son el módulo de consulta de MCC, la respectiva traducción de dicha tabla y el módulo de publicación de noticias.

Podemos indicar con orgullo profesional, que la importancia de este desarrollo ha sido reconocido al otorgarle el grado de aplicación de uso mandatario, para las instituciones financieras pertenecientes a la Red Suiche 7B. La Superintendencia de Bancos en la actualidad estudia extender su uso a las demás instituciones bancarias del país.

En cuanto a las recomendaciones, sugiero que en un futuro sea adoptado un framework como Struts o Spring para el desarrollo de aplicaciones futuras en la Corporación Suiche 7B, ya que estos permiten un mejor desarrollo y aumentan la productividad del equipo de desarrollo.

También sugiero que sean propuestas continuas mejoras como nuevos reportes o nuevos módulos con el fin de proveer a la aplicación de mayor alcance.

En un futuro esta aplicación podría alimentarse de la información arrojada por las aplicaciones de monitoreo de transacciones en tiempo real propias de cada banco. Ello suministraría una fuente continua de información de vital importancia para las investigaciones.

REFERENCIAS BIBLIOGRÁFICAS

[1] Vegas, Jesús. (2002). Desarrollo de aplicaciones Web.
Consultado el día 04 de Septiembre del 2009 de la World Wide Web: Http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node17.html

[2] Mateu, Carles. (2004). Desarrollo de aplicaciones Web. Software Libre. 378 Pag.
Consultado el día 04 de Septiembre del 2009 de la World Wide Web: http://www.uoc.edu/masters/softwarelibre/esp/materials/Desarrollo_web.pdf

[3] Futurhot. (2007). Aplicación Web de gestión y administración hotelera.
Consultado el día 04 de Septiembre del 2009 de la World Wide Web: http://www.futurhot.com.ar/futurweb1.htm

[4] Instituto Tecnológico de Colima. (2006). Arquitectura Cliente – Servidor.
Consultado el día 04 de Septiembre del 2009 de la World Wide Web: http://www.itcolima.edu.mx/profesores/tutoriales/fundamentosbd/sd_u1_6.htm

[5] Consejo Superior de Administración Electrónica. (2007). Arquitectura Cliente – Servidor.
Consultado el día 04 de Septiembre del 2009 de la World Wide Web: http://www.csi.map.es/csi/silice/Global71.html

[6] Janium Technology. (2007). Aplicaciones Basadas en Web.
Consultado el día 05 de Septiembre del 2009 de la World Wide Web: http://www.janium.com/page2/page1/page6/page7/page7.html

[7] Santa Cruz, de la Sierra. (2004). Diseño de Aplicaciones en tres Capas.
Consultado el día 05 de Septiembre del 2009 de la World Wide Web: http://www.geocities.com/trescapas

[8] González, Carlos. (2004). ONess: un proyecto open source para el negocio textil mayorista desarrollado con tecnologías open source innovadoras.

Consultado el día 05 de Septiembre del 2009 de la World Wide Web: http://oness.sourceforge.net/proyecto/html/ch03s02.html

[9] Altadill, Pello. (2002). Struts – Implementación del patrón MVC en la Web.
Consultado el día 05 de Septiembre del 2009 de la World Wide Web: http://www.emagister.com/public/pdf/comunidad_emagister/STRUTS.pdf

[10] Modelo MVC. Blog de Carlos Serrano Sánchez
Consultado el día 05 de Septiembre del 2009 de la World Wide Web:
http://carlos-serrano-sanchez.blogspot.com/2006/10/modelo-mvc.html

[12] Sitio Oficial de la W3C. (2001).
Consultado el día 18 de Octubre del 2009 de la World Wide Web: http://html.conclase.net/w3c/html401-es/cover.html

[13] Ortiz, Manuel. (2004). Aplicaciones de bases de datos Cliente Servidor.
55 Pág. Consultado el día 20 de Octubre del 2009 de la World Wide Web: http://www.cs.buap.mx/~mmartin/notas/BD_CS2004_v3.pdf

[14] Portillo, Aurelio. (2005). Tecnología informática aplicada a medios audiovisuales y sistemas multimedia. 103 Pág.
Consultado el día 20 de Octubre del 2009 de la World Wide Web: www.despazio.net/activos/textos/tecno_info_medios_audiovisuales.rtf

[15] Viney, Armas. (2007). Tutorial Sobre DHTML.
Consultado el día 20 de Octubre del 2009 de la World Wide Web: http://www.ulpgc.es/otros/tutoriales/tutorial_DHTML/

[16] Sitio Web HTMLPOINT.com (2006).
Consultado el día 21 de Agosto del 2009 de la World Wide Web: http://www.htmlpoint.com/JavaScript/corso/js_02.htm

[17] Merelo, Juan. (2004). Programando con JSPs.
Consultado el día 01 de Septiembre del 2009 de la World Wide Web: http://geneura.ugr.es/~jmerelo/JSP/

[18] Artículo de Java Servlets
Consultado el día 01 de Septiembre del 2009 de la World Wide Web:. http://es.wikipedia.org/wiki/Java_Servlet

[19] García, Javier; Rodríguez, José y Aitor, Imaz. (1999). Aprenda Servlets como si estuviera en segundo. 69 Pág.
Consultado el día 01 de Septiembre del 2009 de la World Wide Web: http://www.scribd.com/doc/2984604/Aprenda-servlets-de-java-como-si-estuviera-en-segundo

[20] Robles Gregorio y Ferrer Jorge. (2002). Programación eXtrema y Software Libre.
23 Pág. Consultado el día 01 de Septiembre del 2009 de la World Wide Web:
http://softwarelibre.unsa.edu.ar/programacion/XPySL-HLV/html/slide_1.html

[21] Molpeceres, A. (2002). Procesos de Desarrollo: RUP, XP y FDD.
Consultado el día 10 de Noviembre del 2009 de la World Wide Web: http://www.javahispano.org/download/articulos/metodos_desarrollo.pdf

[22] Sitio Web del departamento de sistemas y computación del instituto tecnológico de la paz. (2002).
Consultado el día 01 de Noviembre del 2009 de la World Wide Web: http://sistemas.itlp.edu.mx/tutoriales/basedat1/tema1_9.htm

[23] Silva, Hernán. (2006). Procesamiento de datos con ORACLE.
Consultado el día 09 de Noviembre del 2009 de la World Wide Web: http://www.mailxmail.com/curso/informatica/datosoracle/capitulo8.htm