

UNIVERSIDAD CATOLICA BOLIVIANA SAN PABLO
UNIDAD ACADEMICA COCHABAMBA
INGENIERIA DE SISTEMAS



**Sistema de consultas financieras a través de Internet
para una entidad financiera**

Tesis de Licenciatura en Ingeniería de Sistemas

Ivan Paniagua Monroy

Cochabamba – Bolivia
Octubre 2003

TRIBUNAL EXAMINADOR

.....
M.Sc. Ing. Guido Rosales
Profesor Guía

.....
Dr. Reynaldo Vargas
Profesor Relator

.....
Mgr. Consuelo Puente
Directora de Ingeniería de Sistemas

.....
Dr. René Santa Cruz
Vice - Rector Regional

*A Dios, que me hizo la persona que soy,
a mi madre y hermana que me aguantaron todo este tiempo,
a mi padre, aunque ya no esta aquí, estuvo siempre conmigo,
y a todos los docentes, familiares y amigos que me apoyaron.*

ÍNDICE

1. INTRODUCCIÓN.....	1
1.1. Definición del problema	1
1.2. Objetivos	2
1.2.1. Objetivo general.....	2
1.2.2. Objetivos específicos.....	2
1.3. Alcance.....	2
1.4. Estado actual.....	3
1.5. Justificación	5
2. ARQUITECTURA CLIENTE SERVIDOR.....	6
2.1. Arquitectura de dos capas (2 Tier)	6
2.1.1. Centrada en el cliente	7
2.1.2. Centrada en el servidor	8
2.2. Arquitectura de tres capas (3 Tier).....	10
2.3. Aplicación Web	12
2.3.1. Connection pool	12
2.3.2. Sesiones.....	13
2.3.3. La cache	15
2.3.4. Validación de datos.....	16
3. SEGURIDAD.....	17
3.1. Definición de seguridad.....	17
3.2. Algoritmos criptográficos	18
3.2.1. Algoritmos simétricos	19
3.2.2. Algoritmos asimétricos	20
3.3. Resúmenes de mensajes o huella digital.....	22
3.4. Firma digital.....	23
3.5. Certificados digitales	25
3.6. Protocolo SSL	26

4. METODOLOGÍA.....	29
4.1. Elección del método.....	29
4.2. Notación	30
4.3. Proceso de desarrollo	36
5. ANÁLISIS	38
5.1. Conceptualización.....	38
5.1.1. Requerimientos	38
5.2. Comportamiento general del sistema.....	40
5.2.1. Descripción de los actores	40
5.2.2. Diagramas de casos de uso	40
5.2.3. Descripción de los casos de uso	43
5.3. Diagramas de interacción.....	46
5.3.1. Para el cliente	46
5.3.2. Para el administrador del sistema.....	56
6. DISEÑO	64
6.1. Arquitectura lógica	64
6.2. Arquitectura física	75
7. IMPLEMENTACIÓN	78
7.1. Herramientas de software	78
7.2. Arquitectura del software	82
7.3. Interfaz	84
7.3.1. Interfaz sistema de consultas.....	84
7.3.2. Interfaz sistema de administración.....	88
7.4. Pruebas	92
8. CONCLUSIÓN Y EXTENSIÓN FUTURA.....	97
8.1. Conclusión.....	97
8.2. Recomendación y extensión futura.....	98
BIBLIOGRAFÍA.....	100

ANEXOS

1. ESTEREOTIPOS	1
2. ESPECIFICACION DE CLASES (DICCIONARIO DE DATOS).....	4
3. COMPONENTES.....	12
4. RENDIMIENTO DE ALGORITMOS DE ENCRIPCIÓN.....	13
5. RESULTADOS PRUEBAS.....	14

Resumen

Este trabajo comienza, primeramente, mostrando la situación actual de la banca electrónica en nuestro país, revelando, cómo pocas entidades son concientes de los beneficios que otorga Internet, tanto para sus clientes como para la misma institución. Pero al mismo tiempo representa una serie de desafíos que a lo largo de este trabajo han sido manejados.

Luego se presentan las diferentes arquitecturas cliente servidor que existen, describiendo las falencias que poseen. Se hace especial énfasis, principalmente, en la arquitectura de tres capas, que es la más adecuada para el desarrollo del sistema. Sin embargo es necesario considerar aspectos importantes como el rendimiento de la aplicación en el servidor hasta el almacenamiento de páginas en la cache del cliente.

La incorporación de la notación UML para el modelamiento de todo el proceso de desarrollo, juega un papel sumamente importante. En el presente trabajo UML, no esta limitado solamente al desarrollo Orientado a Objetos, sino además, se lo ha extendido para representar aspectos importantes dentro el sistema, como son las páginas web y la base de datos.

En general el aspecto más importante que se maneja a lo largo de esta tesis es la seguridad, en particular los algoritmos criptográficos son la herramienta fundamental que ha sido estudiada y aplicada en la protección de la información, donde se hace uso de herramientas estándares para la protección en la comunicación entre el cliente y el servidor, así como también en la protección de los datos en el servidor.

Sin embargo la seguridad va más allá de la aplicación de la criptografía y se ha tomado en cuenta también un buen análisis y diseño que contemple la validación de los datos, el tiempo de duración de un usuario en el sistema así como también la verificación, división e integración de los diferentes componentes involucrados en el sistema.

En la parte de la implementación se han utilizado las mejores “tecnologías abiertas”, libres de costo, que además, al incluir su código fuente, su desempeño y confiabilidad ha sido verificado ampliamente alrededor del mundo.

1. Introducción

Estamos inmersos en un mundo que cada día cambia, pero estos cambios se están produciendo cada vez con una mayor aceleración. El mundo de los negocios no es ajeno a ello. Internet esta cambiando la manera, como las personas y empresas están realizando sus negocios.

El Comercio Electrónico es uno de los más recientes servicios de los muchos que están haciendo de Internet un nuevo paradigma. Lo más importante que caracteriza a este nuevo paradigma es que deja a un lado la brecha divisoria entre clientes y empresa, consiguiéndose beneficios para unos y otros.

Es un medio que permite a las empresas ser más eficientes con sus clientes, otorgándoles una respuesta más rápida a sus necesidades y expectativas.

No solo se otorga un mejor servicio al cliente, sino también permite que la misma empresa salga beneficiada, extendiendo su alcance a una región, ciudad o país. Gracias a la escalabilidad de este medio el alcance es mundial.

1.1. DEFINICIÓN DEL PROBLEMA

Consultas de saldos, obtener extractos de cuentas, etc. son tareas que cientos de personas realizan cada día en instituciones financieras, pero muchas veces al cliente se le obliga a personarse a sus oficinas a realizarlas, esto resulta incomodo para él, ya que involucra pérdida de tiempo y dinero.

Abrir sucursales en puntos estratégicos de la ciudad podría en cierta manera solucionar el problema, pero esto resulta costoso para una institución, ya que se deben instalar y mantener líneas dedicadas costosas para unir sus sucursales, además de contar con recursos de personal, equipo e infraestructura, para realizarlo.

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Desarrollar un sistema de consultas financieras a través de Internet para una entidad financiera.

1.2.2. OBJETIVOS ESPECÍFICOS

- Permitir a los usuarios realizar consultas de cuentas financieras a través de Internet.
- Preservar la confidencialidad de la información a través de mecanismos de seguridad que incluyen integridad y niveles de encriptación.
- Proteger el acceso al sistema mediante la autenticación de los usuarios.
- Monitorear y controlar las acciones del usuario en la utilización del sistema de consultas.
- Desarrollar un subsistema de administración que permita un manejo adecuado de los usuarios y sus cuentas.

1.3. ALCANCE Y LIMITACIONES

- Se permitirá a los clientes realizar consultas en :
 - créditos - consultas de saldos, extractos de pago, próximas fechas de vencimiento.
 - cajas de ahorro - extractos de movimientos.
 - depósitos a plazo fijo - información de montos y fechas.

- Se utilizarán las herramientas y algoritmos estándares, existentes en el campo de la criptografía.
- El sistema mantendrá la estructura y funcionamiento interno, bajo el cual funciona el sistema ya existente en la institución.
- Se mostraran las ventajas de utilizar las herramientas actuales, para el desarrollo del sistema.

1.4. ESTADO ACTUAL

Hoy en día las entidades financieras desarrollan nuevos canales que les permitan enfrentar a los cambios que demandan las necesidades de sus clientes. En los últimos años ha habido una evolución enorme en la relación cliente-empresa ante la cual surgieron los servicios interactivos considerada para muchos como una nueva forma de negocios.

Este fenómeno no es nuevo en nuestro país y actualmente entre las opciones que existen podemos mencionar la Telebanca, que permite realizar consultas y transferencias de cuentas vía teléfono así como también a los cajeros automáticos.

Pero sin duda alguna una de las tendencias que siguen no sólo entidades financieras si no cualquier empresa en general es Internet.

El mercado en Internet en Bolivia ha estado experimentando un rápido crecimiento gracias a la apertura del mercado de las Telecomunicaciones en Bolivia en noviembre del 2001, terminando de esta manera con el periodo de exclusividad que se le dio a ENTEL para proporcionar conectividad internacional de Internet. Un ejemplo de este crecimiento es la llegada a nuestro país de tecnologías como ADSL (Asymmetric Digital Subscriber Line) que permite conexiones mucho más rápidas.

En Bolivia existen más de 49000 abonados de Internet¹ registrados en proveedores de servicios, sin embargo esta cifra no refleja la cantidad de usuarios real existente ya que no se considera el hecho de que una gran mayoría en nuestro país se conectan a Internet por medio de locales públicos como los cibercafés, al no contar con una computadora y conexión propias.

Gracias a este crecimiento muchas instituciones pero principalmente la banca, ha invertido en la construcción de páginas web en las que ofrecen información de la misma institución y los servicios que otorgan a sus clientes.

Banco	Telebanca	Internet	Banca en Linea
Banco Nacional de Bolivia	Instaltel	www.bnb.com.bo	BNB Net
Banco Union	MarqueUnion	www.bancounion.com.bo	UniNet
Banco Santa Cruz	Hola Bancruz	www.bsc.com.bo	Bancruz Net
Banco de Credito	No	www.bancodecredito.com.bo	Si
Citibank	CitiPhone Banking	www.citibank.com	Paylink
Banco Central de Bolivia	No	www.bcb.gob.bo	No
Banco Ganadero	Ganatel	www.bancoganadero.com.bo	No
BISA	TeleBisa	www.grupobisa.com	No
Banco Mercantil	Superconsulta	www.bancomercantil.com.bo	No
Banco Sol	No	www.bancosol.com.bo	No

Fig. 1.1
Banca electrónica en Bolivia

El sector bancario es, con diferencia, el que más ha avanzado en este campo, por encima de las mutuales, cooperativas, fondos financieros, etc. donde el crecimiento ha sido prácticamente nulo.

Sin embargo, como lo refleja el cuadro 1.1, solo un número muy limitado de sitios podrían clasificarse como verdaderamente bancos en línea en las que se ofrezcan

¹ Datos reportados a la Superintendencia de Telecomunicaciones para el año 2002 - www.sittel.gov.bo

servicios de consulta de saldos o transferencia de cuentas. La mayoría de estas páginas ofrecen información estática y con contactos únicamente de correo electrónico.

1.5. JUSTIFICACIÓN

Como lo reflejan los datos anteriores, hasta ahora son muy pocas las entidades que son conscientes de los beneficios que proporciona el trabajar con Internet. Precisamente una de las grandes ventajas de este medio, es que permite mostrarse a grandes y chicos en igualdad de condiciones. Aparte del indudable valor añadido que aporta a sus clientes, el hecho de posibilitar este tipo de operaciones va a permitir a la entidad financiera, a la larga poder disponer de menos infraestructura alcanzando a su vez nuevos mercados sobre la red.

2. Arquitectura cliente servidor

La arquitectura cliente-servidor consiste en la división y distribución del sistema entre dos o más computadoras conectadas en red. Esta divide al sistema en dos o más tareas menos complejas las cuales deben estar cuidadosamente diseñadas y bien definidas. La división resulta generalmente en tres partes que son:

Presentación.- Maneja como el usuario interactúa con el sistema. Es responsable de la validación de los datos, manejo de eventos y de la presentación de la información, generalmente (pero no necesariamente) diseñando una Interface de Usuario Gráfica (GUI - Graphical User Interface) basada en Windows.

Aplicación.- También conocida como lógica de negocios, maneja los mecanismos del sistema, es decir se ocupa de la ejecución de toda la lógica para procesar una operación, aplicando las reglas o políticas de negocios que se siguen en el sistema.

Manejo de Datos.- Maneja el almacenado y recuperación de los datos. En esta parte, se cuenta comúnmente con un sistema de administración de base de datos (DBMS - Data Base Management System) el cual permite que varios clientes puedan acceder y actualizar un mismo conjunto de datos a la vez.

La arquitectura consiste también en la distribución de estas partes de la aplicación entre el cliente y el servidor, lo que causa un gran impacto en el diseño de la aplicación. Existen diversas formas de acomodar las partes y a continuación se describen las más importantes.

2.1. ARQUITECTURA DE DOS CAPAS (2 TIER)

Existen dos tipos de esta arquitectura, una centrada en el cliente y la otra centrada en servidor, aquí se describen detalladamente cada uno de estos tipos.

2.1.1. Centrada en el Cliente

La implementación de este tipo de arquitectura de dos capas se la realiza colocando la parte de *Presentación* como la de *Aplicación* en el cliente, haciendo lo que se conoce como "cliente pesado" (thick client) [KVA]. En el servidor se encarga del *Manejo de Datos* a través del DBMS. La comunicación se la realiza generalmente a través de una Intranet o red privada de una compañía, usando protocolos de transporte remoto de la base de datos. Estos protocolos son específicos de una base de datos y es necesaria una previa configuración en el cliente antes de poner en marcha el sistema.

La primera capa consiste entonces del "cliente pesado" y la segunda capa de un servidor de base de datos como se ve en la fig. 2.1.

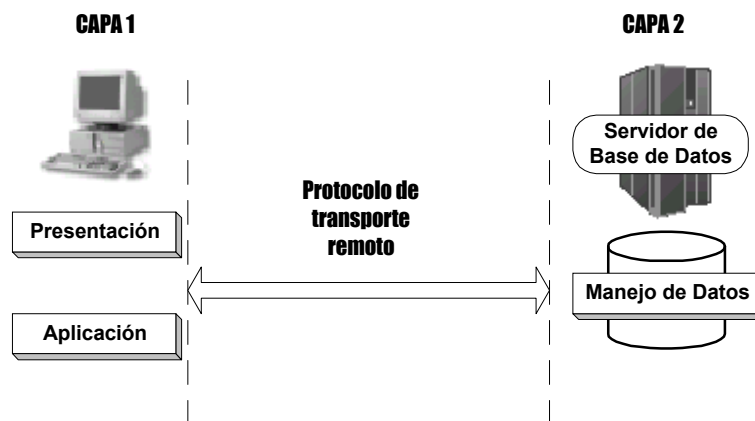


Fig. 2.1

Arquitectura centrada en el cliente

Típicamente la mayoría de las aplicaciones usan esta arquitectura debido a que se implementan rápidamente con herramientas como Visual Basic, Delphi, etc. Sin embargo esta posee una serie de problemas que se deben tomar en cuenta²:

- La necesidad de que toda la información de acceso como ser nombres de cuentas de bases de datos, contraseñas, identificadores, estén en el cliente es inseguro ya que una persona con malas intenciones podría espiar esta información dentro el sistema.
- El mantenimiento y administración del sistema es costoso. Por ejemplo si una empresa quiere realizar cambios en la política de negocios, o en la *Presentación* del sistema, necesita de procedimientos de distribución para entregar la nueva versión del sistema a todos sus clientes. Estos procedimientos de distribución incluyen la distribución sobre redes o la producción de adecuados medios como diskettes, o CDs. Esto puede ser muy caro, propenso a errores y puede consumir mucho tiempo efectuarlo.
- Debido a que el procesamiento de los datos toma lugar en el cliente, los datos sin procesar tienen que ser transportados sobre la red, lo cual causa un gran agotamiento en el ancho de banda de la red.

2.1.2. Centrada en el Servidor

La arquitectura centrada en el servidor distribuye el sistema de una manera más eficiente a la anterior, llevando la *Aplicación* al servidor y quedando en el cliente la parte de *Presentación*. En este tipo de arquitectura la *Aplicación* queda reducida a lo que se conocen como "procedimientos almacenados" (stored procedures) y "activadores" (triggers) [GON].

² Revisar la siguiente bibliografía [BRI98:26] , [DUA], [KVA] y [LLO]

Un "procedimiento almacenado" como su nombre lo dice, es un procedimiento que es programado para cumplir con la parte de la *Aplicación*, que es ejecutado en el servidor junto a la base de datos. El cliente envía una solicitud al servidor de base de datos que incluye el nombre del procedimiento y algunos parámetros. Como en la anterior arquitectura, las solicitudes son transmitidas a través de la Intranet, con protocolos de transporte remoto de la base de datos.

Los "activadores" son eventos que son activados automáticamente cuando ciertos estados son detectados en la base de datos. Esta función permite la implementación de reglas permanentes de la empresa.

Entonces la primera capa solo esta constituida de la *Presentación* del sistema en el cliente, en tanto que los procedimientos almacenados y los activadores residen en el DBMS en el servidor, conformando la segunda capa como se muestra en la fig. 2.2.

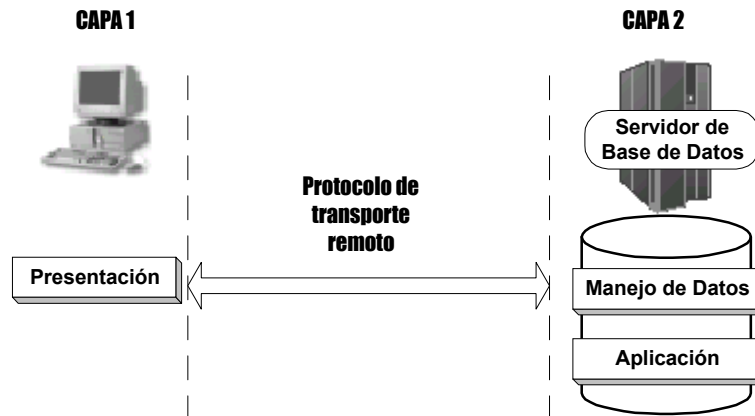


Fig. 2.2

Arquitectura centrada en el servidor

Este esquema representa una importante mejora en la distribución del sistema, ya que esta diseñado para una administración centralizada. Sin embargo esta arquitectura trae un conjunto de características no deseadas³:

- Los procedimientos almacenados y activadores son una solución totalmente propietaria, lo que reduce las posibilidades de flexibilidad de la Aplicación, debido a que existen diferentes lenguajes de procedimientos para cada DBMS. Por ejemplo, Oracle usa PL/SQL, Microsoft Transact SQL, Informix tiene su propio Informix 4GL, etc.
- Persiste el problema de que toda la información de acceso como ser nombres de cuentas de bases de datos, contraseñas, identificadores, estén en el cliente.
- A medida que se incrementa el número de clientes, el sistema va resultando más ineficiente ya que el servidor llega a saturarse.

2.2. ARQUITECTURA DE TRES CAPAS (3 TIER)

Debido a los problemas existentes en una arquitectura de dos capas, para aumentar el rendimiento y el alcance del sistema se añade una capa intermedia (middle tier) entre el cliente y la base de datos, conocida como "servidor de aplicaciones" donde, como su nombre lo indica, se lleva a cabo la *Aplicación*. Esta capa se comunica con el servidor de base de datos (DBMS), usando protocolos específicos de la base de datos. Pero el cliente, ahora conocido como cliente ligero (thin client) se comunica con el servidor de aplicaciones usando protocolos de comunicación estándares basados en TCP/IP como ser HTTP, FTP, etc.

³ Revisar la siguiente bibliografía [GON] ,[KVA] y [BRI98]

Con esta arquitectura se tiene completamente distribuida las tres partes del sistema para las tres capas, mostrado en la fig. 2.3.

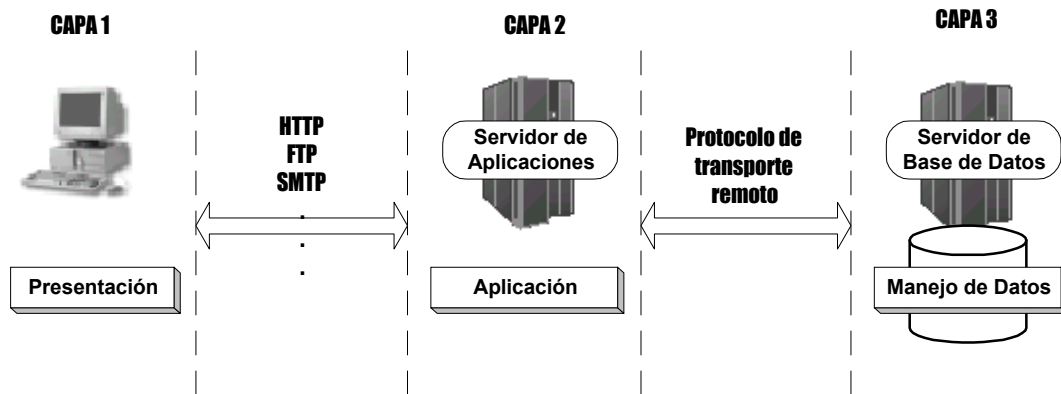


Fig. 2.3

Arquitectura de tres capas

Existen muchas ventajas de esta arquitectura sobre la anterior pero las principales son:

- Cualquiera de las partes divididas del sistema puede ser cambiada o mejorada sin afectar a otras tareas (flexibilidad). Si se realizan, por ejemplo, cambios en la *Presentación* (interfaz de usuario), la *Aplicación* y el *Acceso a datos* permanecen iguales, o cambios en la *Aplicación* no afectan a la interfaz de usuario o al DBMS. Esta administración centralizada permite mantener más fácilmente al sistema lo que a su vez permite ahorrar tiempo y costos.
- El cliente no necesita ningún sistema de software específico (excepto un navegador, por ejemplo) lo cual permite al sistema soportar una amplia variedad de clientes.
- Esta diseñada para trabajar en un ambiente dinámico como Internet.

Es posible tener tanto el servidor de aplicaciones como el servidor de base de datos en la misma plataforma y en muchos casos es la solución más óptima. Pero para que sea reconocida como una arquitectura de tres capas el sistema debe estar separado y comunicado de la forma descrita para aprovechar estas ventajas.

2.3. APLICACIÓN WEB

Una aplicación web es cualquier aplicación que utiliza tecnologías web, incluyendo base de datos, navegadores, servidores web, etc. Una aplicación web esta basada en una arquitectura de tres capas. El servidor de aplicaciones ensambla páginas dinámicas y entrega su contenido a los clientes a través del protocolo HTTP. Existen varias tecnologías para generar páginas dinámicas como ser JSP (JavaServer Pages), Servlets, ASP (Active Server Pages), etc. Para el contexto de este trabajo se las llamara simplemente “páginas servidor”.

A continuación se describe aspectos importantes que deben ser considerados, cuando se trabaja con aplicaciones web.

2.3.1. Connection Pool

Una Página Servidor genera paginas HTML a partir de información almacenada en la base de datos por lo que hacen uso extensivo de la misma. Conectarse a la base de datos es una actividad que consume tiempo, ya que la base de datos debe localizar recursos de memoria y comunicación (en el caso que el servidor de aplicaciones y de BD estén separados físicamente) como también autenticar al usuario que se esta conectando.

Si la cantidad de usuarios va a ser poca, o si solo admite una petición de datos a la vez no hay mucho problema en este aspecto, sin embargo a medida que el número de usuarios crece, y aumenta la cantidad de conexiones simultaneas, lo que es común en

aplicaciones web, la base de datos llega rápidamente a saturarse y puede incluso llegar a fallar.

Una técnica para evitar este tipo de situaciones es el denominado "Connection Pool" o almacén de conexiones.

Un Connection Pool es un intermediario entre paginas servidor y la base de datos. Cuando se inicializa el Connection Pool, se abre un número predeterminado de conexiones por única vez. Cuando una página necesita operar con la base de datos, no solicita una conexión al DBMS directamente, sino que realiza la solicitud al Pool. Este realiza una búsqueda entre sus conexiones y la marca como "ocupada". El solicitante debe realizar sus operaciones con la base de datos lo más rápidamente posible y devolver la conexión al Pool, sin cerrarla.

Esto constituye una mejora importante en el rendimiento de la aplicación,⁴ ya que se reutilizan conexiones ya abiertas. En lugar de abrir una y otra vez una conexión, solo se establece la conexión una sola vez y luego se utiliza la misma conexión para subsiguientes solicitudes a la base de datos.

2.3.2. Sesiones

A menudo es necesario retener datos específicos para que un usuario se mueva de página en página a través de un sitio. HTTP es un protocolo "stateless", lo que significa que este no es capaz de mantener un estado a través de solicitudes de los clientes. Este protocolo crea una nueva conexión por cada solicitud y no hay manera que el servidor reconozca que una serie de solicitudes han llegado del mismo navegador. Cuando un navegador solicita una página en el servidor, este responde a la solicitud y entonces se "olvida" de él.

⁴ Para ver un cuadro comparativo de la mejora del rendimiento ver Anexo 4, Connection Pool

Existen varias formas de añadir estado a HTTP como ser “campos ocultos” (hidden fields), “galletas” (cookies) y sesiones (sessions).

Los “hidden fields” o “campos ocultos”, son campos dentro de una página HTML, que no son desplegados en navegador del cliente. Estos mantienen el estado o la información colocando generalmente el identificador de usuario (User ID) en estos campos, cada vez que una respuesta es enviada al cliente. Esta forma es fácil de implementar y es soportada por todos los navegadores sin embargo como se describe mas adelante, debido a su inseguridad no es aconsejable utilizarlos⁵.

Las “cookies” o “galletas” son piezas de información que son pasadas entre el servidor y el navegador. El servidor envía una “galleta” que contiene datos como por ejemplo el ID del usuario, cuando el navegador recibe la “galleta” este lo almacena y lo envía de vuelta al servidor la próxima vez que el navegador accede al servidor. Sin embargo cuando se almacena información confidencial como el ID del usuario ya sea en los “campos ocultos” o las “galletas”, representa un elevado riesgo de seguridad, ya que con solo ver el código fuente de la página o conocer donde se almacenan las galletas en el cliente, es posible que cualquier otro usuario con acceso a la misma maquina pueda ver estos datos confidenciales.

Afortunadamente las páginas servidor como ASP o JSP, pueden resolver este problema a través de las sesiones. Cada vez que un usuario accede al sitio, una sesión genera un identificador de sesión o sessionId único por cada usuario. Este identificador es usado para mantener el estado a través de las páginas que visite el usuario. Los datos almacenados en las sesiones son *almacenados en el servidor* y lo único que se envía y guarda en el cliente es el identificador de sesión. Generalmente en una sesión se guarda el ID de usuario pero este nunca es enviado a ningún lado y es almacenado temporalmente en el servidor.

⁵ Para ver los problemas en los “hidden fields” ver [ROP]

Debido a que no se envían datos confidenciales al cliente que puedan ser utilizados luego para acceder al sistema, esta forma es más segura que las anteriores, para mantener el estado en el protocolo HTTP, debido a que cada vez se crean nuevos identificadores de sesiones y no pueden volverse a utilizar aquellos generados anteriormente.

Otra medida de seguridad de las sesiones, es la de contar con un tiempo de duración predeterminado por el desarrollador o por defecto de unos 10 a 20 minutos, dependiendo de la versión y de la tecnología que se esté utilizando. Una vez terminado este tiempo, el identificador de sesión es eliminado junto con los datos almacenados en la sesión. De esta manera se evita que los datos queden indefinidamente y no represente un riesgo de seguridad.

2.3.3. La Cache

En aplicaciones web, la cache es el lugar donde se almacenan temporalmente los datos, de tal manera que no tengan que ser recibidos cada vez de la fuente original. Los navegadores tienen una cache donde se almacenan las páginas que ya se han accedido. También los servidores proxy pueden almacenar páginas, con el objetivo de hacer más rápido el acceso.

La cache puede hacer que la aplicación web sea más eficiente, sin embargo cuando se trabaja con información dinámica o páginas dinámicas, estas solo son válidas para el tiempo cuando son creadas originalmente, con un tiempo de vida definido, por lo tanto la página almacenada en la cache podría ya no ser válida.

Para evitar que los navegadores y los servidores proxy almacenen páginas en la cache, las páginas servidor deben enviar cabeceras especiales a través del protocolo HTTP, las cuales son:

“Pragma”, “no-cache”

“Cache-Control”, “no-cache”

De esta manera se evita que guarden en la cache las respuestas del sistema.

2.3.4. Validación de Datos

Algunos datos externos que llegan al servidor son de personas no confiables, por lo que los datos deben ser validados o filtrados antes de ser usados.

El problema surge debido a que muchos de los datos que llegan de afuera son provistos de manera codificada, denominada “URL – encoded”, por lo que los valores son escritos en el formato “%HH” donde HH es el valor hexadecimal de ese byte. El servidor debe manejar estos valores, decodificarlos y verificar que no contengan “valores ilegales”, debido que permiten a los datos, sean interpretados como comandos y así de esta manera, ejecutar comandos arbitrarios e malintencionados en el servidor.

Un conjunto de valores ilegales o peligrosos es el siguiente [STE:Q37]:

“& ; ` ' | * ? ~ < > ^ () [] { } \$ % + \n \r \”.

La validación de estos datos podría efectuarse en el navegador del usuario antes de ser enviados al servidor, a través de lenguajes como JavaScript,⁶ o VBScript, sin embargo aunque esta validación puede ser útil para el usuario, ya que valida inmediatamente, sin el acceso a la red, no cuenta al dentro el aspecto de la seguridad, ya que el atacante podría enviar valores ilegales directamente al servidor, sobrepasando las validaciones del navegador. Por lo que esta validación se la debe realizar en el servidor, más específicamente en la “página servidor”.

⁶ El cual no tiene relación con el lenguaje de programación Java, el cual se tocara más adelante.

3. Seguridad

Uno de los aspectos que más preocupan a usuarios y empresas que desean utilizar y trabajar sobre Internet es la seguridad. Este problema surge porque Internet fue creada para un libre acceso a la información, regida principalmente por el buen uso de la red. Los protocolos desarrollados (especialmente TCP/IP) no fueron concebidos teniendo en cuenta aspectos de seguridad.

Cuando se implantaron redes privadas, se asumían ambientes seguros, con usuarios autorizados y limitados, lo que otorgaba cierto control sobre la seguridad, pero como no se vislumbraba el incremento acelerado de número de usuarios, ni tampoco la conexión a redes externas como Internet, el problema en la seguridad se vuelve crucial. Para una empresa, un ambiente abierto como Internet, representa exponer lo más importante que tiene: su información. Información que en la mayoría de los casos es confidencial y puede ser hurtada, aprovechada por la competencia, o por cualquier otra persona que pueda encontrarla valiosa.

El hecho de que la información tenga que pasar a través de muchas computadoras hace posible para terceras partes, interferir con la comunicación de muchas maneras:

Escuchando la Comunicación (Eavesdropping).- La información permanece intacta, sin embargo la privacidad esta comprometida. Por ejemplo alguien podría ver números de cuenta, o grabar una conversación o interceptar información confidencial.

Modificación del Mensaje (Tampering).- La información en tránsito es cambiada o reemplazada. Por ejemplo alguien podría cambiar el destino de una transferencia de fondos a la suya.

Suplantación (Impersonation).- La información pasa por una persona que imita o suplanta al verdadero destino. Por ejemplo se pretende ser la dirección www.ucbcba.edu.bo cuando en realidad no lo es, esta técnica es conocida como spoofing.

3.1. DEFINICIÓN DE SEGURIDAD

La seguridad se la puede definir de muchas maneras, ya que involucra muchas áreas, lo que lo hace difícil de definir sucintamente, aunque dentro del contexto de este trabajo, la definición más adecuada sea:

“La seguridad de la información concierne un conjunto de tecnologías y procesos para la protección de la información”⁷

El objetivo principal de la seguridad es el, tratar de minimizar la vulnerabilidad de la información a través de técnicas bien establecidas y las podemos clasificar principalmente en cuatro grupos:

Confidencialidad de los datos.- Evitar que se revelen deliberadamente o accidentalmente, los datos privados. Esto no concierne la integridad o el origen de la información.

Integridad de los datos.- Verificar que los datos no sean alterados, esto es, que los datos recibidos por el receptor coincidan por los enviados por el emisor.

Autenticación.- Verificar la fuente de los datos antes de establecer un acuerdo o trato. La autenticación puede ser sólo de la entidad origen, de la entidad destino o de ambas a la vez.

⁷ Definición extraída de [RAZ:2], What exactly do you mean by “Information Security”?

Irrenunciabilidad (no repudio).- Proporciona la prueba, de que alguna de las partes ha participado efectivamente en cierta actividad.

3.2. ALGORITMOS CRIPTOGRÁFICOS

Los algoritmos criptográficos protegen la confidencialidad de la información. Estos utilizan operaciones matemáticas complejas para transformar un mensaje (texto en claro) a una forma ilegible o encriptada. El texto en claro es encriptado con una clave o llave, compuesto generalmente de una cadena de bits aleatorio. Una vez que el mensaje se ha hecho ilegible, este solo podría llegar a ser entendible o descriptado a través de la misma clave si se usan algoritmos simétricos u otra clave si se usa algoritmos asimétricos. Sin la clave el mensaje es inútil, debido a que tomaría mucho tiempo y trabajo descriptarlo.

En general la robustez de un algoritmo esta relacionado en descubrir esta clave, lo cual depende del algoritmo utilizado y de la longitud de la clave. Por ejemplo existen algoritmos, que se basan en la factorización de números primos grandes para hallar la clave. La factorización de estos números es dificultosa para cualquier maquina, debido a que toma mucho trabajo, hallar números primos grandes.

Pero no solo depende del algoritmo sino también de la longitud de la llave. En general llaves de longitud más largas otorgan mayor robustez. Por ejemplo si se tiene una llave de 128 bits de longitud, para un algoritmo en particular, esta llave otorga una protección significativamente mejor a una llave de 40 bits con el mismo algoritmo.

3.2.1. Algoritmos simétricos

La familia de los algoritmos clasificados como simétricos son los más comunes o tradicionales. Requieren que todas las partes que intervienen, compartan la misma clave.

El emisor encripta los datos con la clave privada y el receptor desencripta los datos con la misma clave como se ve en la fig. 3.1.

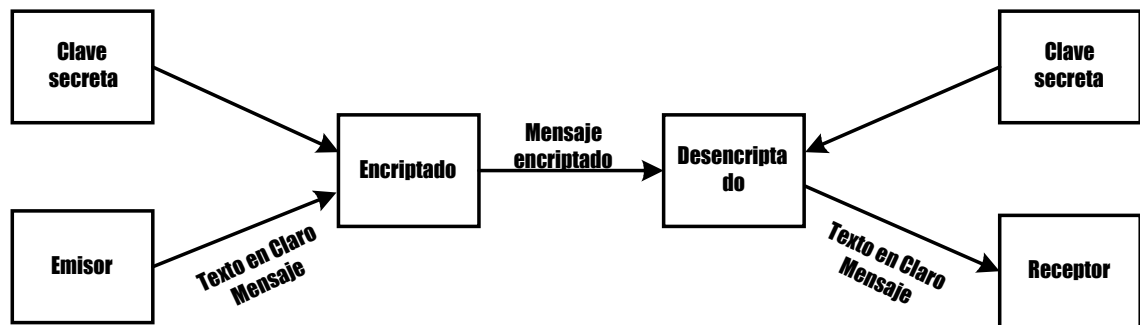


Fig. 3.1

Clave simétrica

Actualmente existen algoritmos de este tipo, muy rápidos y robustos. Si la clave tiene un largo de n bits existen 2^n claves posibles. Si esta es lo suficientemente larga (típicamente se usan valores de 56 a 128 bits) es imposible "reventarlas" (romperlas) usando la fuerza bruta, esto es probando todas las combinaciones posibles de la clave.

El principal inconveniente estriba en la necesidad de que todas las partes conozcan la clave, lo que lleva a problemas de distribución de las claves, es decir como transportar la llave secreta del origen al destino en forma segura y a prueba de violaciones. Si existiera una forma de enviar la llave secreta de forma segura, no habría la necesidad de encriptar / desencriptar un mensaje ya que se usaría esta misma forma segura para enviar mensajes.

3.2.2. Algoritmos asimétricos

La criptografía de clave pública resuelve el problema de distribución de claves definiendo un algoritmo que use dos claves asimétricas. Una es llamada clave pública la

cual esta destinada a difundirse libremente por toda la red y la otra es llamada clave privada la cual nunca es distribuida y es mantenida siempre en secreto.

La primera propiedad que tienen esta clase de algoritmos es que, si se encriptan los datos con la llave publica solo pueden ser descryptados con la llave privada.

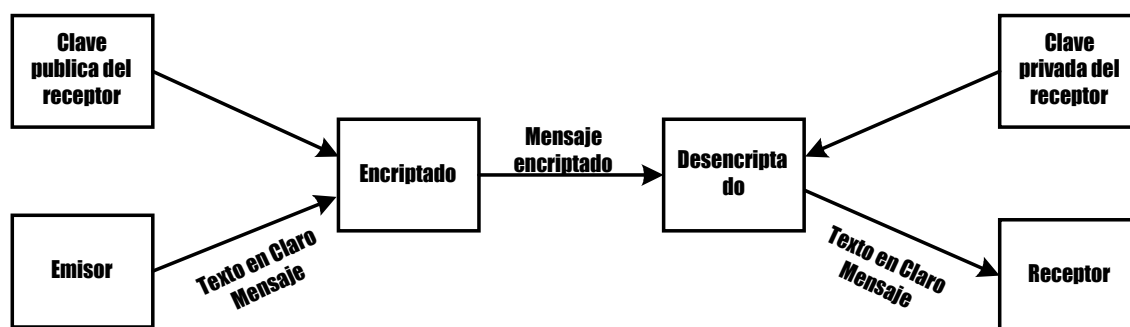


Fig. 3.2

Clave asimétrica – 1ra propiedad

La clave privada es guardada por el receptor, y es usada para descryptar datos, la otra clave - la publica puede ser usada por el emisor para encriptar la información confidencial que tenga que enviar al receptor, como se ve en la figura 3.2. Debido a que solo la llave privada que posee el receptor puede descryptar información que ha sido encriptada por su correspondiente clave publica, nadie que intercepte el mensaje encriptado podrá descryptarlo y el receptor no necesita enviar su clave privada a nadie para que cualquiera pueda enviarle mensajes encriptados.

La segunda propiedad que tienen estos algoritmos es que, si se encriptan los datos con la llave privada estos solo puede descryptarse usando la llave publica mostrado en la figura 3.3.

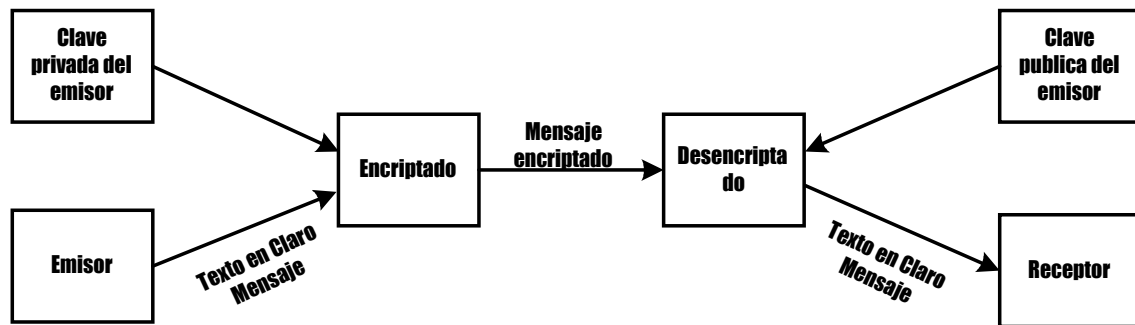


Fig. 3.3
Clave asimétrica – 2da propiedad

Evidentemente de esta forma, cualquiera podría ver los mensajes que son encriptados con la llave privada del emisor pero la verdadera utilidad de esta propiedad, no radica en la confidencialidad de la información sino mas bien, en la utilidad de generar firmas digitales.

Estas propiedades asimétricas hacen que este tipo de algoritmos sean muy útiles. Sin embargo poseen el inconveniente de ser más lentos que sus equivalentes de clave simétrica.

Diferentes algoritmos criptográficos pueden necesitar diferentes longitudes de llaves para tener el mismo nivel de robustez. Por ejemplo, un algoritmo asimétrico que usa la factorización de números primos, puede usar solo un subconjunto de todos los posibles valores de una llave de una longitud dada. En cambio un algoritmo simétrico, puede usar todos los posibles valores de una llave de una longitud dada. Por ello si se usa una llave cuya longitud es de 128 bits con algoritmos simétricos, otorga más protección que una llave de 128 bits con algoritmos asimétricos.

3.3. RESUMENES DE MENSAJES O HUELLA DIGITAL

Si se quiere garantizar la integridad de los mensajes, esto es, que los datos recibidos por el receptor coincidan con los enviados del emisor, se usan los resúmenes de mensajes (message digest).

Una función de resumen toma como entrada al mensaje y produce como salida un resumen del mensaje o huella digital de longitud fija (típicamente de 128 bits y 160 bits) como se ve en la figura 3.4. Esta clase de funciones tienen la propiedad de generar una única huella digital a partir de un mensaje, aunque teóricamente, es posible encontrar dos mensajes con idéntica huella la probabilidad de que esto ocurra es ínfima. Si se cambian los datos del mensaje el resumen o huella cambia. También modificar los datos de forma tan sabia como para obtener la misma huella es algo computacionalmente inabordable [MON].

Otra aplicación de los resúmenes de mensajes es la protección de contraseñas en mecanismos de autenticación. El mensaje seria en realidad la contraseña y una vez aplicado la función de resumen nadie más seria capaz de obtener la contraseña que se tiene en la huella digital. Solo exactamente el mismo mensaje o contraseña generaría la misma huella digital.

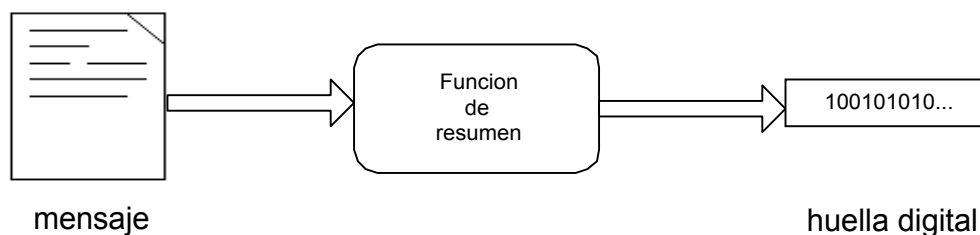


Fig. 3.4
Resúmenes de mensajes

3.4. FIRMA DIGITAL

La firma digital es un conjunto de caracteres que ligam a mensaje con una clave de cifrado. Cuando se desarrollaron los algoritmos asimétricos se descubrió que no solo sirven para encriptar o desencriptar, sino también sirven para ligar mensajes con claves privadas, gracias a la segunda propiedad de los algoritmos asimétricos.

Una firma digital primero aplica al mensaje del emisor una función de resumen obteniendo su correspondiente huella digital o resumen, luego este es encriptado con la llave privada del emisor y se añade al propio mensaje que el emisor quiere enviar, como se muestra en la figura 3.5. El receptor utiliza la clave pública del emisor para desencriptarlo y obtener la huella digital, por otro lado el receptor genera la huella digital del mensaje enviado por el emisor y compara la que genero el emisor y la que genero el. Si coincide con este, se puede asegurar que el mensaje no ha sido alterado gracias a la aplicación de funciones de resumen, además y lo más importante, que lo ha enviado el propietario de dichas claves, garantizando que el mensaje solo ha podido ser enviado por el emisor ya que es el único que conoce su clave privada. Con esto es imposible para el emisor negar haber firmado el mensaje (no repudio) y virtualmente imposible para terceras partes alterar el contenido de cualquier mensaje sin detección.

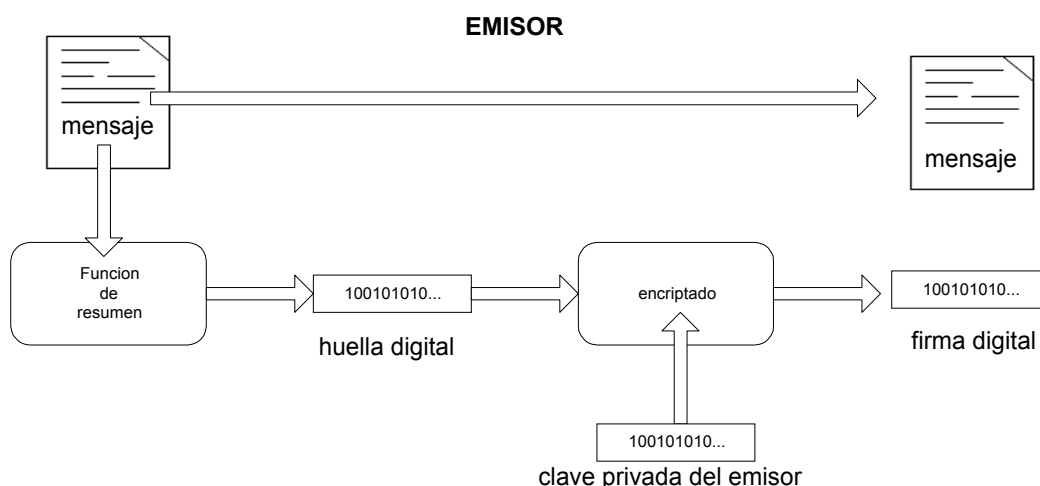


Fig. 3.5
Firma digital

3.5. CERTIFICADOS DIGITALES

Hasta ahora se pueden enviar mensajes confidencialmente y firmarlos digitalmente con la seguridad que los datos no han sido modificados, pero aun persiste un problema, que radica en las llaves públicas. Todo lo que se ha visto hasta ahora, asume que se conoce al dueño de las llaves publicas, pero como se garantiza que las llaves publicas del emisor o receptor son efectivamente de quien dice ser. El emisor puede por ejemplo realizar alguna transacción con un falso receptor (suplantación) incluso encriptado y firmando los mensajes enviados entre sí.

Para este solucionar este problema existen los certificados digitales. Un certificado digital es un documento electrónico usado para identificar a una compañía, persona o alguna otra entidad conocida como *sujeto* y asociarla con sus llaves públicas [ENG]. Este certificado es emitido por una Autoridad Certificadora (AC) en la que confían todas las partes, tanto emisor como receptor y que cumple la función notarial en donde se constata la identidad y solvencia del sujeto.

El certificado digital incluye además de la llave publica del sujeto, información propia del mismo, fecha de expiración, el nombre de la AC que emito el certificado así como también otra información administrativa de la AC, como un número de serie, etc. Pero lo más importante, un certificado incluye la firma digital de la AC. La AC encripta (firma) la llave publica del sujeto con su llave privada para generar el certificado digital. Esta firma permite a los clientes o usuarios que conocen y confían en la AC (en realidad no es el usuario, si no el navegador del usuario que confía en la AC debido a que tiene almacenada la llave publica de la AC que genero la firma digital) pero no conocen la identidad del sujeto, confiar en este.

Un riesgo muy peligroso para una AC es tener su llave privada comprometida, debido a que cualquiera que tuviese esta llave podría generar certificados falsos. Una buena AC utiliza sistemas que ocultan la llave a todos, incluso a los mismos empleados de la AC. En estos casos la firma de la AC es generada por una maquina que no puede revelar la llave privada, incluso puede destruirla si esta es comprometida. Algunas de las mas conocidas AC son VeriSign y Thawte.

En resumen los certificados ayudan a prevenir el uso de llaves publicas falsas, de esta forma los destinatarios se aseguran que el mensaje es realmente de quien dice ser, evitando la suplantación.

3.6. PROTOCOLO SSL

El protocolo SSL (Secure Sockets Layer) es una tecnología ampliamente usada en el mundo en el ámbito del comercio electrónico, para proteger la seguridad en la comunicación. Este opera como una capa adicional entre TCP/IP y la capa de aplicación. De esta forma permite que el protocolo SSL sea independiente de la aplicación, siendo posible usar FTP, SMTP y otras aplicaciones además de HTTP.

Para aplicaciones web la presencia de https:// en la dirección del servidor indica que se trata de un servidor que soporta SSL y que debe utilizarse este protocolo en la comunicación entre dicho servidor y cliente o navegador.

SSL permite una comunicación segura entre el cliente y el servidor, preservando la privacidad a través de la encriptación de los mensajes, autenticación, integridad y no repudio a través de las firmas digitales y certificados digitales. El proceso de autenticación usa algoritmos asimétricos y firmas digitales para confirmar que el servidor es quien dice ser. Una vez que el servidor ha sido autenticado, el cliente y el servidor usan algoritmos simétricos los cuales son más rápidos para encriptar toda la información que intercambian.

Para establecer una comunicación segura utilizando SSL se tienen que seguir una serie de pasos (lo que en la terminología de este protocolo suele denominarse “handshake” o “apreton de manos”) que se describen a continuación.

1.- El proceso comienza por la solicitud del cliente de un URL, a un servidor que soporte SSL. El cliente o navegador informa al servidor que algoritmos de criptografía puede utilizar, la robustez de estos algoritmos, la versión de SSL que soporta y otra información que el servidor necesita para comunicarse con el cliente usando SSL. Además el cliente solicita el certificado del servidor para autenticarlo. Hasta ahora no se ha intercambiado información secreta solo una lista de opciones.

2.- El servidor presenta al cliente su certificado digital, el cual contiene su llave pública, los algoritmos criptográficos que se van a utilizar, la versión de SSL que tiene y otra información que el cliente necesita para comunicarse con el cliente usando SSL.

3.- El cliente autentifica al servidor, verificando la validez del certificado digital enviado por el servidor en el anterior paso. Esto se lleva a cabo descriptando el certificado con la llave pública del servidor y determinando si proviene de una autoridad de certificación de confianza. Después se hace una serie de verificaciones sobre el certificado tales como fecha, URL del servidor, etc.

4.- Una vez que se ha autenticado al servidor el cliente genera un número secreto aleatorio denominado “premaster secret” y lo encripta usando (el algoritmo concertado) la llave pública del servidor y se la transmite al servidor.

5.- Luego que se ha enviado el “premaster secret” al servidor, el cliente genera a partir de este número lo que se denomina “master secret” el cual no es enviado a ningún lado, en ningún momento y no es más que la llave simétrica que se utilizara para enviar mensajes entre el cliente y el servidor.

6.- El servidor recupera el número aleatorio o “premaster secret”, descriptando con su llave privada y a partir de este número el servidor genera la llave maestra “master secret”. El handshake termina en este paso.

Ahora tanto el cliente como el servidor usan la llave maestra para encriptar y descriptar los mensajes que se envían entre sí. Una vez se haya establecido una comunicación segura, se deben hacer verificaciones periódicas para garantizar que la comunicación sigue siendo segura a medida que se transmiten datos. Luego que la transacción ha sido completada, se abandona una sesión SSL presentando un mensaje advirtiéndole que la comunicación no es segura. El handshake se realiza una sola vez y por sesión.

El sistema es tan robusto como lo sea el menos seguro de los algoritmos que se utilicen. Lo general es que los algoritmos a utilizar sean los más altos que puedan soportar ambos, siendo lo normal: 40 bits y mejor de 128 bits.

4. Metodología

El presente capítulo pretende dar a conocer de forma general el método elegido para trabajar en el desarrollo del trabajo. Primeramente se mostrará la conveniencia de utilización del enfoque elegido para el problema que se pretende resolver en este trabajo, además se describirán las características más importantes del lenguaje UML para representar al sistema y por último se describirán los pasos a seguir durante el transcurso del trabajo.

4.1. ELECCION DEL MÉTODO

Cuando se desarrollan aplicaciones web, estas tienen el inconveniente de cambiar rápidamente⁸, adaptándose a nuevos requerimientos, debido a que precisamente en Internet las cosas cambian a un ritmo acelerado. Conjuntamente a esto, la seguridad es un aspecto que siempre estará en constante cambio, ya que siempre surgirán nuevas vulnerabilidades y el sistema debe ser capaz de adaptarse y evolucionar con estos cambios. Por estas razones, estos sistemas pueden llegar a ser bastante complejos rápidamente.

El enfoque orientado a objetos y el método de análisis y diseño orientado a objetos propuesto por Grady Booch, ayudan a manejar estos problemas debido a que una de las características más importantes de este método, es la de seguir un desarrollo evolutivo ya que los sistemas orientados a objetos son más flexibles al cambio y por lo tanto están preparados para evolucionar en el tiempo, reduciendo en gran medida el riesgo que presenta construir sistemas que trabajan en Internet.

Además al ser un método orientado a objetos permite aprovechar las ventajas del enfoque orientado como ser:

⁸ Ver Capítulo 2, sección [2.2]

- Simula el mundo real, haciéndose más fácil de comprender el desarrollo del sistema.
- Permite la evolución del sistema en el tiempo y facilita la corrección de errores.
- Permite aprovechar las ventajas de un lenguaje orientado a objetos.

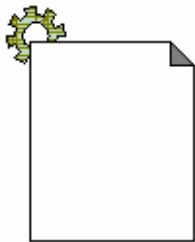
Un método viene acompañado de una notación para representar o visualizar el proceso de desarrollo y Booch tiene su propia notación para representar las fases de análisis y diseño orientado a objetos, sin embargo se ha optado por utilizar otra notación más conveniente para nuestros propósitos, la cual es UML (Unified Modeling Language). En la siguiente sección se explica las características más sobresalientes de esta notación.

4.2. NOTACIÓN

Un buen modelo de un sistema incluye todos los componentes importantes del sistema. En el sistema que se desarrolla en este trabajo, las páginas web, páginas servidor⁹, formularios, etc. son los componentes fundamentales dentro del sistema. Resulta indispensable que los modelos de análisis y diseño representen estos elementos.

La notación UML permite extender y representar estos componentes de manera única [CON99]. Usando esta notación, se puede representar estos componentes como objetos y considerar sus propiedades como atributos y operaciones de una clase. Por ejemplo una página servidor ASP o JSP puede ser considerado como un objeto cuyos atributos representan las variables de la página servidor y las funciones que realiza son considerados como métodos de la misma. Sin embargo para el caso particular de las páginas servidor se debe tener una especial consideración. Estas páginas implementan parte de la lógica de negocios del sistema en el servidor así como también preparan la parte de interfaz de usuario, una página HTML para el navegador. Sin embargo colocar juntos y considerarlos como un objeto, puede ser muy confuso y no ayuda a comprender su rol en el modelo.

Este inconveniente es resuelto dividiendo lo que es la página servidor como tal y lo que <<construye>> o genera (<<builds>>), una página HTML que es cargada en el navegador del cliente, asignando a cada uno de ellos su propio estereotipo. Un estereotipo es un nuevo elemento de modelado y es representado por iconos especiales o texto entre un par de símbolos “<< >>” siendo parte del mecanismo de extensión de UML. En la siguiente tabla se muestran algunos de los más importantes estereotipos para clases que se utilizaran mas adelante en el trabajo. La lista completa se muestra en el Anexo 1.

<p><<server page>></p> 	<p>Representa una página que contiene código que se ejecuta en el servidor. Las funciones de la página representan los métodos de la clase y las variables accesibles por todas las funciones de la página representan los atributos de la clase. Estas páginas están relacionadas con recursos en el lado del servidor como lógica de negocios y bases de datos.</p>
---	---

⁹ Ver capítulo 2, sección [2.3]

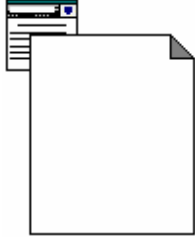
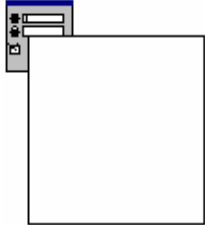
<p><<client page>></p> 	<p>Representa una página HTML, que se “ejecuta” o se carga en el navegador. Los métodos y atributos de esta clase corresponden a las funciones y variables implementadas con JavaScript o VB Script, las que son llevadas a cabo por el navegador. Estas páginas representan la parte de Interfaz de Usuario y tienen asociaciones con otras páginas cliente o servidor.</p>
<p><<form>></p> 	<p>Es un conjunto de campos de entrada (input fields) que forman parte de una página cliente.</p> <p>Sus atributos corresponden a campos de entrada del formulario como ser <<input boxes>>, <<text areas>>, <<radio buttons>>, <<check boxes>>, etc. Un formulario no tiene operaciones ya que una operación no puede ser contenida en un formulario. Cualquier operación que interactúa con el formulario puede ser propiedad de la página que contiene el formulario.</p>

Tabla 4.1
Estereotipos de clases

Además de los estereotipos de clases, la extensión de UML define estereotipos para asociaciones. Las páginas servidor están relacionadas con páginas cliente, a través de la asociación <<builds>>, que indica que una página servidor es responsable de producir como salida una página cliente. Los estereotipos de asociaciones principales son:

<<builds>>	Las páginas servidor <<construyen>> páginas cliente. Esta asociación es siempre direccional de una página servidor a una página cliente
<<link>>	Links son enlaces de una página cliente a otra página cliente o servidor.
<<submit>>	Un formulario envía (<<submit>>) todos los datos que contiene a una página servidor la cual procesa los datos.
<<redirect>>	Una página servidor puede redireccionar el procesamiento a otra página.

Tabla 4.2
Estereotipos de asociaciones

Durante el desarrollo se elaboran los diagramas de componentes (en la notación de Booch conocida como diagramas de módulos) para mostrar la asignación de clases y objetos a componentes en el diseño físico del sistema.

Un diagrama de componentes presenta las piezas físicas del sistema (archivos). En el caso de aplicaciones web, el vínculo fundamental entre las páginas cliente y las páginas servidor se encuentra en el diagrama de componentes.

Un componente servidor, un archivo JSP por ejemplo, es a la vez una página servidor y una página cliente, pero es representado como un solo componente en el diagrama de componentes, aunque lógicamente represente a dos páginas esto se ve en la figura 4.1 Componente Servidor.

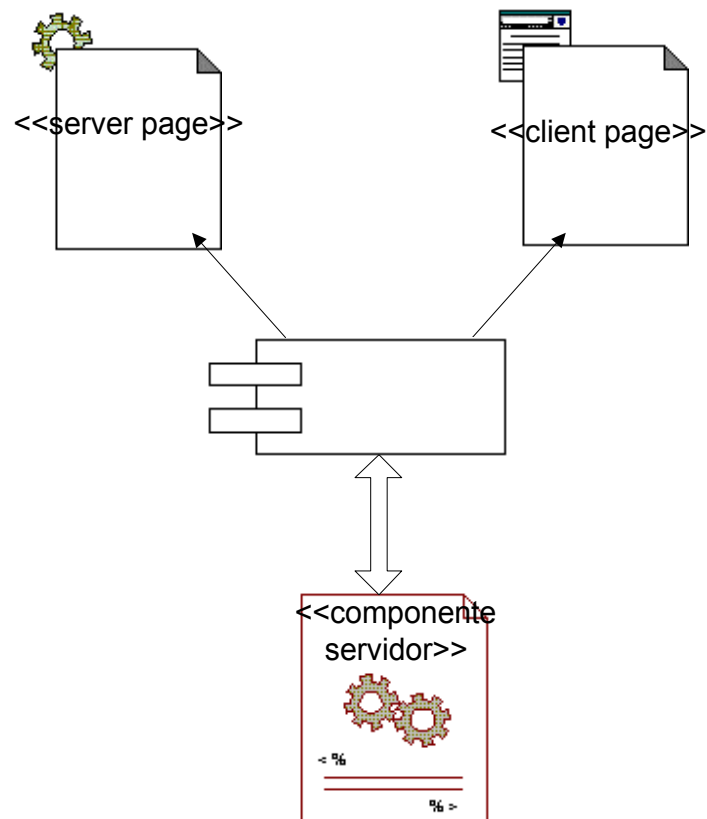


Fig. 4.1
Componente servidor

Se debe aclarar que no todas las páginas cliente son generadas por páginas servidor. Estas páginas cliente son estáticas y están representadas directamente en un componente HTML en el diagrama de componentes como se ve en la figura 4.2 de Componente html. En el Anexo 1 se describen los estereotipos de estos componentes.

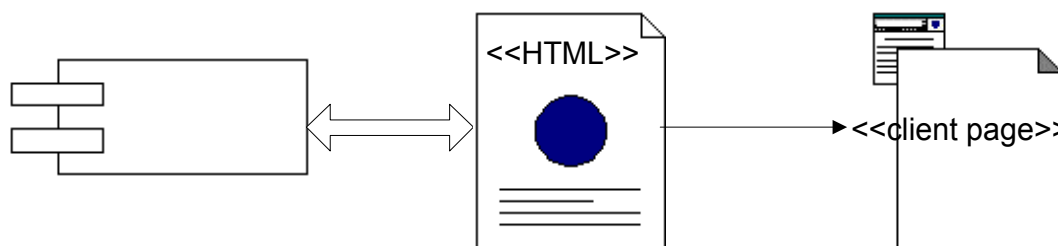


Fig. 4.2

Componente html

El poder de UML no solo se limita al modelamiento de software orientado a objetos, también como se ha visto hasta ahora se puede extender hacia el modelamiento de aplicaciones web. Pero aún UML puede ser aplicado en otra área muy importante del desarrollo de software, como es el modelamiento de datos.

UML soporta las necesidades de modelamiento de la base de datos¹⁰, donde una tabla puede ser representada a través de un nuevo estereotipo y conjuntamente con sus relaciones refleje la estructura interna de la base de datos. Esta tabla se muestra como una clase, donde los atributos son los campos de la tabla y además puede mostrar si algunos de estos campos son claves primarias (PK) o claves foráneas (FK), mostrado en un ejemplo, en la figura 4.3.

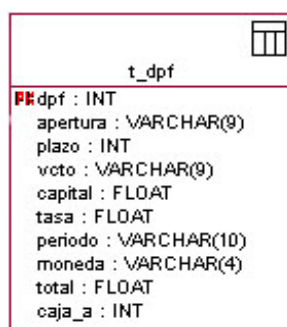


Fig. 4.3

Estereotipo de tabla

¹⁰ Para una introducción completa del tema, se tiene [DOR99]

4.3. PROCESO DE DESARROLLO

En esta parte se describen sucintamente los pasos a seguir en el desarrollo del presente trabajo¹¹.

Análisis.- Comienza con la conceptualización del sistema donde se establecen los requerimientos del sistema. Una manera eficaz para determinar los requerimientos del sistema es la utilización de los *casos de uso*. Los casos de uso son una vista externa del sistema que nos dice, que es lo que el sistema podría hacer, describiendo la interacción del sistema y el usuario. La funcionalidad de los casos de uso es capturada a través de los escenarios o diagramas de interacción. Estos diagramas muestran de manera clara y sencilla los objetos y las relaciones que tienen con otros objetos dentro el sistema¹². Con esto se conforma el comportamiento general del sistema.

Diseño.- Sobre la base de los resultados en la etapa de análisis, el diseño tiene como objetivo dar una arquitectura al sistema. Para lograr dicho objetivo se crea la arquitectura lógica y arquitectura física. Examinando los diagramas de interacción y las relaciones entre objetos se puede definir la arquitectura lógica, conformando los diagramas de clases para expresar la estructura y comportamiento de las clases, además de la dirección y cardinalidad de sus asociaciones. En la física se define donde se situaran los componentes lógicos identificados en la arquitectura lógica, para lo que se crean los diagramas módulos o componentes.

Implementación.- Para la implementación del sistema se deben escoger las herramientas de software necesarias. El sistema es implementado sobre la base de los requerimientos definidos, identificando y corrigiendo deficiencias de forma continua.

¹¹ Para una introducción completa en el tema, se tiene el libro de Booch [BOO94].

¹² Ver pag. 70 del libro Quatrani Ferry [QUA:25].

Pruebas.- Para verificar si el sistema cumple con los requerimientos planteados en la etapa de análisis se realizan diferentes tipos de pruebas. El desarrollo de aplicaciones web complejas, enfatiza la necesidad de un testeo efectivo y además flexible, específicamente adaptado a estos nuevos paradigmas tecnológicos.

5. Análisis

5.1. CONCEPTUALIZACIÓN

El objetivo principal del sistema es permitir a los clientes realizar consultas de cuentas financieras a través de Internet, por lo que se ha decidido dividir el sistema en dos partes o subsistemas:

- Subsistema de consultas.- Brinda al cliente de la institución, la posibilidad de informarse sobre el estado de sus cuentas financieras vía Internet.
- Subsistema de administración.- Lo utiliza el administrador del sistema y se encarga de administrar y controlar las cuentas de usuarios en el servidor.

A continuación se listan los requerimientos de cada uno de estos subsistemas.

5.1.1. REQUERIMIENTOS

Sobre la base de los objetivos que se pretenden, se pueden establecer los siguientes requerimientos para el subsistema de consultas:

- Autenticar a los usuarios del sistema, evitando el ingreso no autorizado.
- Permitir a los usuarios registrados en el sistema, listar todas las cuentas financieras que posee en la entidad.
- Permitir a los usuarios realizar consultas de movimientos tanto en caja de ahorro, depósitos a plazo fijo y créditos.

- Permitir a los usuarios de la entidad obtener un plan de pagos, tanto en caja de ahorro como de crédito.
- Cambiar la contraseña de acceso al sistema.

Los clientes deberán realizar estas operaciones de manera segura, protegiendo la confidencialidad de la información, encriptando esta bajo el protocolo SSL. También se debe contar con un control que permita a un usuario una vez autenticado permanecer en el sistema por determinado tiempo. Las operaciones que realiza este usuario deben quedar registradas en la base de datos, conjuntamente con el lugar, la fecha y hora que se las realizo.

Para el subsistema de administración se identifican los siguientes requerimientos:

- Administrar cuentas de usuarios, lo que involucra funciones como listar, añadir, eliminar o actualizar usuarios.
- Listar las cuentas financieras de uno o todos los usuarios del sistema.
- Administrar el registro o logs de operaciones que fueron realizadas por los usuarios del sistema. Esto comprende operaciones como el listado o vaciado de logs. El vaciado implica encriptar los datos que se están vaciando y exportarlos hacia un archivo externo.

Se podrán realizar también, búsquedas de un subconjunto de usuarios, logs o cuentas financieras, bajo diferentes criterios, desde la que se podrá realizar una exportación de datos a un archivo externo.

El administrador realizara la administración de este subsistema únicamente de forma local en el servidor, esto por razones de seguridad. Además el administrador, debe

también ser autenticado antes de realizar cualquier operación. De la misma forma que en el subsistema de consultas, las operaciones que realiza el administrador serán registradas.

5.2. COMPORTAMIENTO GENERAL DEL SISTEMA

Para describir el comportamiento general del sistema se han identificado dos actores o usuarios en el sistema, los cuales son detallados más específicamente en la siguiente sección. Se exponen los diagramas de casos de uso para estos actores, se describen los casos de uso y se elaboran los escenarios correspondientes.

5.2.1. Descripción de los Actores

“Cliente”

Representa a la persona natural o jurídica que posee al menos una cuenta en la entidad financiera. Cada cliente debe tener un identificador de cuenta o login y su respectiva contraseña de acceso para utilizar el sistema de consultas a través de Internet. El cliente debe tener al menos una cuenta vigente o activa en la entidad y además de contar con un navegador que soporte el protocolo SSL.

“Administrador del sistema”

Es la persona autorizada para utilizar el subsistema de administración en el servidor. Los privilegios otorgados a este actor le permitirán controlar y administrar cuentas de todos los usuarios que se tienen en el sistema.

5.2.2. Diagramas de Casos de Uso

Los diagramas de casos de uso aquí expuestos ayudan a determinar las fronteras del sistema y a describir la funcionalidad desde el punto de vista del usuario.

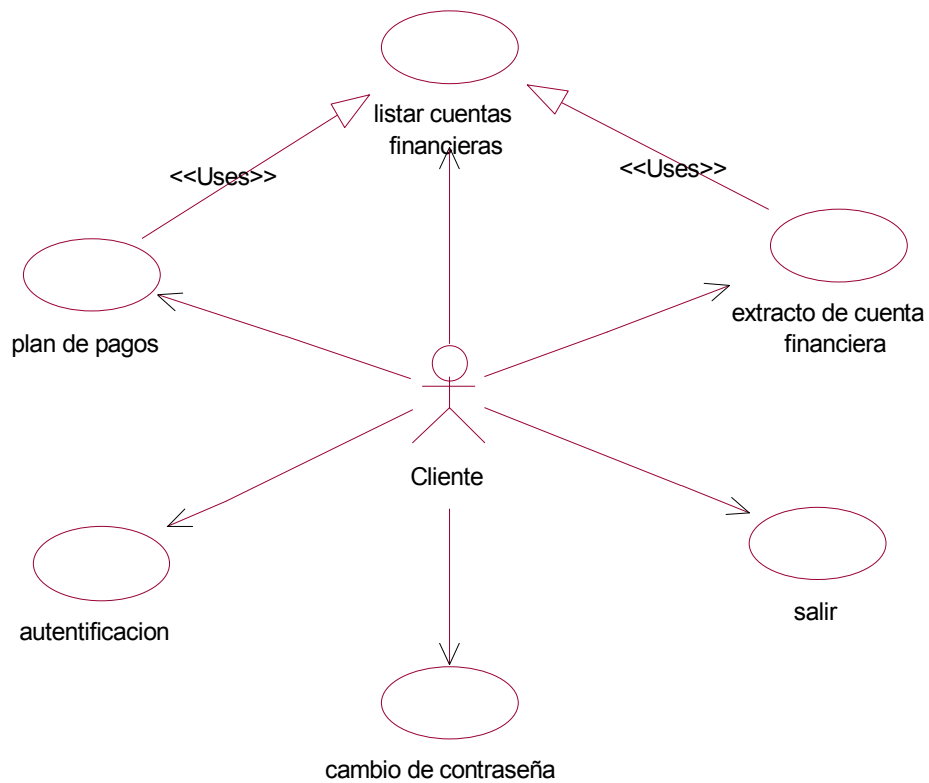


Fig. 5.1
Diagrama de casos de uso:
Subsistema de consultas

Los diagramas de casos de uso, de la figuras 5.1 y 5.2 son un buen punto de comienzo para identificar las páginas y ventanas que podrían existir en el sistema tanto en el subsistema de consultas como en el de administración respectivamente.

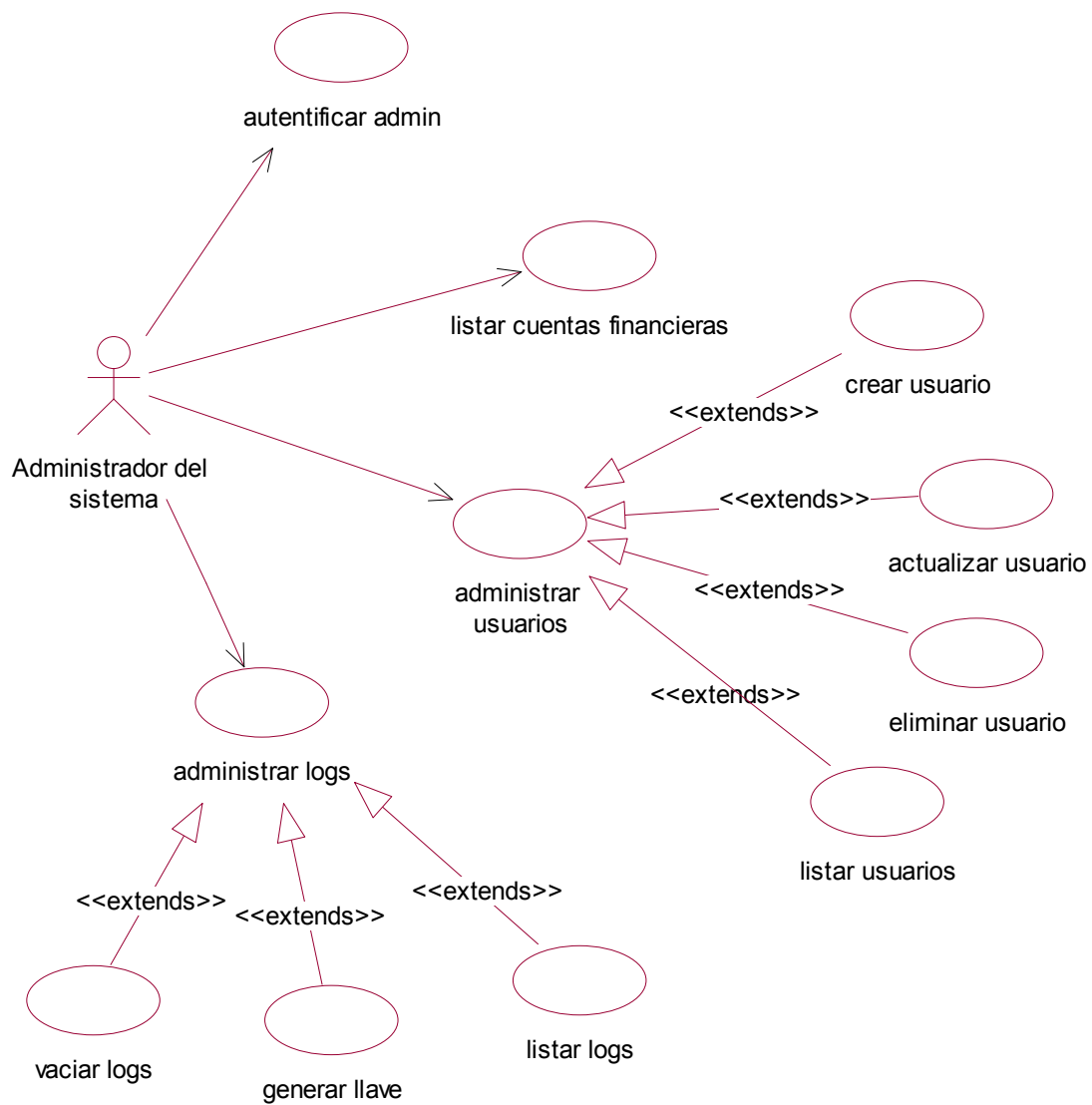


Fig. 5.2
Diagrama de casos de uso:
Subsistema de administración

A continuación se describen cada uno de los casos de uso involucrados en los diagramas ya expuestos.

5.2.3. Descripción de los Casos de Uso

Para el Cliente:

- a) Autenticación.- Tiene como objetivo proteger el ingreso no autorizado al sistema a través de Internet. El cliente debe ingresar su identificador o login y su contraseña de acceso, luego el sistema debe verificar los datos introducidos sean correctos para permitir el ingreso al sistema y a sus diferentes opciones. El control se lo realiza tanto en el lado del cliente como del servidor.
- b) Listado de cuentas financieras.- Una de las opciones que tiene el cliente es listar todas las cuentas financieras que posee en la entidad. La información obtenida variará de acuerdo al tipo de cuenta financiera esto es caja de ahorro, depósito a plazo fijo o crédito que se tenga. Este caso de uso está relacionado con los casos de uso “extracto de cuenta financieras” y “plan de pagos”.
- c) Plan de pagos.- El cliente puede obtener información detallada de sus próximos pagos de las cuentas en depósito a plazo fijo como de crédito que posea. La relación de <<Uses>> entre el caso de uso de “listar cuentas financieras” y “plan de pagos”, significa que si el cliente desea realizar una consulta de una cuenta específica el sistema debería generar previamente el listado de cuentas financieras.
- d) Extracto de cuenta financiera.- Una vez que se tiene la lista de cuentas financieras de un cliente, este puede obtener un historial de movimientos de una cuenta. El sistema debe responder de acuerdo al número de cuenta seleccionado y al CI o RUC del cliente, con información detallada de todos los movimientos que se han efectuado sobre dicha cuenta.

- e) Cambio de contraseña.- Cuando el cliente selecciona esta opción, debe introducir la contraseña que desea cambiar, la nueva contraseña y confirmar la nueva contraseña. De la misma forma que la autenticación, la validación de los datos se lo realiza en el lado del cliente como del servidor. Se deberá encriptar la nueva contraseña introducida antes de ser almacenada en la base de datos.
- f) Salir.- Este caso de uso es necesario debido a que, con esta opción se elimina la sesión iniciada por el cliente al ingresar al sistema y para que otros usuarios no puedan acceder a ver su información sin antes haberse autenticado en el sistema. Esta opción es lanzada por el cliente o activada automáticamente después de un cierto tiempo.

Para el administrador del sistema:

- g) Autenticar administrador.- Tiene como objetivo proteger el ingreso no autorizado al sistema de administración. Esta autenticación es similar al del cliente pero difiere en el hecho de no ser una página de autenticación, debido a que el sistema de administración solo es ejecutado únicamente desde el servidor, en su lugar la autenticación del administrador es realizada a través de una ventana.
- h) Administrar Usuarios.- Este caso de uso involucra varias opciones como listar, crear, eliminar y modificar usuarios. Estas opciones se ven reflejadas con las relaciones <<extends>> que muestran un comportamiento opcional a partir de este caso de uso.
- i) Listar usuarios.- Una vez autenticado al administrador, este recibirá como primera ventana la lista de todos usuarios que se tienen en la entidad, los mismos que pueden ser ordenados por diferentes criterios.

- j) Crear usuario.- El administrador puede crear una cuenta para un cliente de la entidad, para lo cual debe introducir primeramente el CI o RUC según se trate si es una persona natural o jurídica. En base a este valor se relacionara al cliente con o una o varias cuentas financieras, luego se completan los datos restantes del usuario, como por ejemplo la contraseña de acceso o cantidad de intentos continuos. Antes de guardar los datos en la base de datos se validan todos los datos introducidos, después se encripta la contraseña introducida y se lo guarda conjuntamente con los demás datos.
- k) Eliminar usuario.- El administrador puede retirar de servicio a un usuario, para lo que debe seleccionar de la lista de usuarios al cliente que desea eliminar, luego el sistema solicitara una confirmación esta eliminación. El sistema, en realidad no eliminara completamente a un usuario, en su lugar, se pasara a este usuario, de la lista de usuarios activos a otra de usuarios eliminados.
- l) Actualizar usuario.- Para actualizar una cuenta, el sistema solicita al administrador que seleccione de la lista de usuarios el cliente que desea actualizar. El sistema responde con información detallada del mismo y el administrador puede por ejemplo bloquear dicha cuenta o realizar alguna otra actualización. El sistema deberá controlar si se ha actualizado la contraseña de acceso y de ser así se deberá encriptar la nueva contraseña introducida.
- m) Listar cuentas financieras.- Este caso de uso tienen como función listar todas las cuentas financieras que existen en el sistema. En este mismo se contara con una opción de búsquedas de cuentas. También se listarán todas las cuentas financieras pertenecientes a un determinado cliente de la entidad.
- n) Administrar logs.- Este caso de uso involucra varias opciones como listar o vaciar logs detalladas a continuación. Aquí también se hará la utilización o generación de llaves para el encriptado.

- o) Listar logs.- Tiene como función listar todas las operaciones que ha realizado un determinado usuario de la entidad, incluyendo también al administrador. Se muestra información detallada como por ejemplo la hora o el lugar desde donde fue realizada alguna consulta o la creación de un nuevo usuario por parte del administrador.
- p) Vaciar logs.- Se podrá realizar un vaciado de todas las operaciones registradas en la base de datos, esto debido a que puede llegar a saturarse. Este vaciado se lo realizara hacia un archivo externo, sin embargo esta información se la deberá proteger de tal manera que nadie mas pueda verla o modificarla.
- q) Generar llave.- Cuando se realiza el vaciado de logs, se utiliza una llave para el encriptado de la información, por lo ello se deberá contar con una opción que permita la creación de nuevas llaves ya que no siempre se utilizará la misma llave para todos los vaciados. Pero antes de generar la nueva llave se tiene que permitir almacenar la llave anterior en un archivo externo.

5.3. DIAGRAMAS DE INTERACCIÓN

Los diagramas de interacción expuestos en esta sección, son usados para describir como los casos de uso son realizados, a través de la interacción de un conjunto de objetos en el tiempo. Estos siguen un flujo normal de eventos, es decir siguen la ruta mas comúnmente seguida y sirven como base para identificar las abstracciones principales que forman el dominio del problema.

5.3.1) PARA EL CLIENTE

AUTENTIFICACIÓN.-

El usuario debe ingresar su identificador o login y su contraseña de acceso en la página de ingreso al sistema. Antes que sean enviados al servidor, el navegador puede validar los datos introducidos, a través de JavaScript, llamando al método “val”. El sistema verifica los datos introducidos en el servidor a través de la página servidor “autentifica”. Lo primero que se realiza es verificar que los datos que provienen del usuario sean válidos, que no sean nulos y que no contengan caracteres considerados peligrosos. Esto lo realiza el método “valido” del objeto “util”. Luego desde la página “autentifica” y en base del login del usuario, se obtiene el objeto “usuario” desde el objeto “Bd”. Aquí, en realidad el objeto “Bd” se comunica con la base de datos a través del objeto “ConnectionPool” cuyo propósito es descrito en la sección [2.3.1]. Una vez obtenido el objeto usuario se verifica la contraseña introducida con el método “contraseña”, lo que realiza a su vez la encriptación, por medio del método “digest” del objeto “encriptacion” y se la compara con la que se tiene en el objeto “usuario” devuelto. Luego se obtienen la cantidad de ingresos que ha realizado este usuario, para comprobar si el usuario no esta bloqueado. Es desde el objeto usuario, que se registra el ingreso al sistema mediante el método “setLog” del objeto “Bd”. Después se introduce el identificador del usuario en la clase “sesión” y se redirecciona a la página servidor “menu”. Desde esta página se genera la lista de opciones que se tiene en el sistema de consultas.

El diagrama de interacción de la figura 5.3 muestra los pasos aquí descritos para la parte de autenticación.

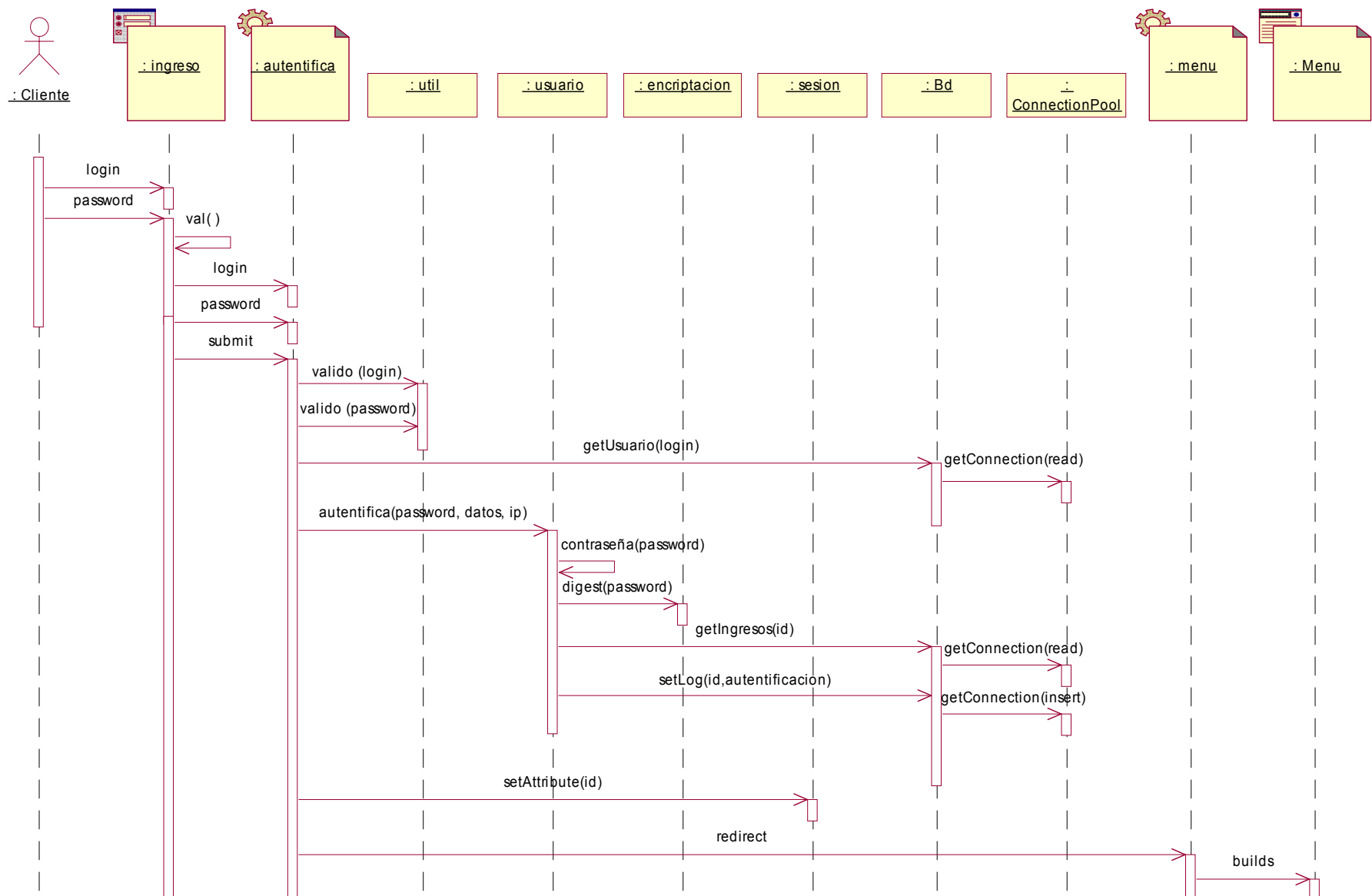


Fig. 5.3
Autenticación

DEVOLVER USUARIO.-

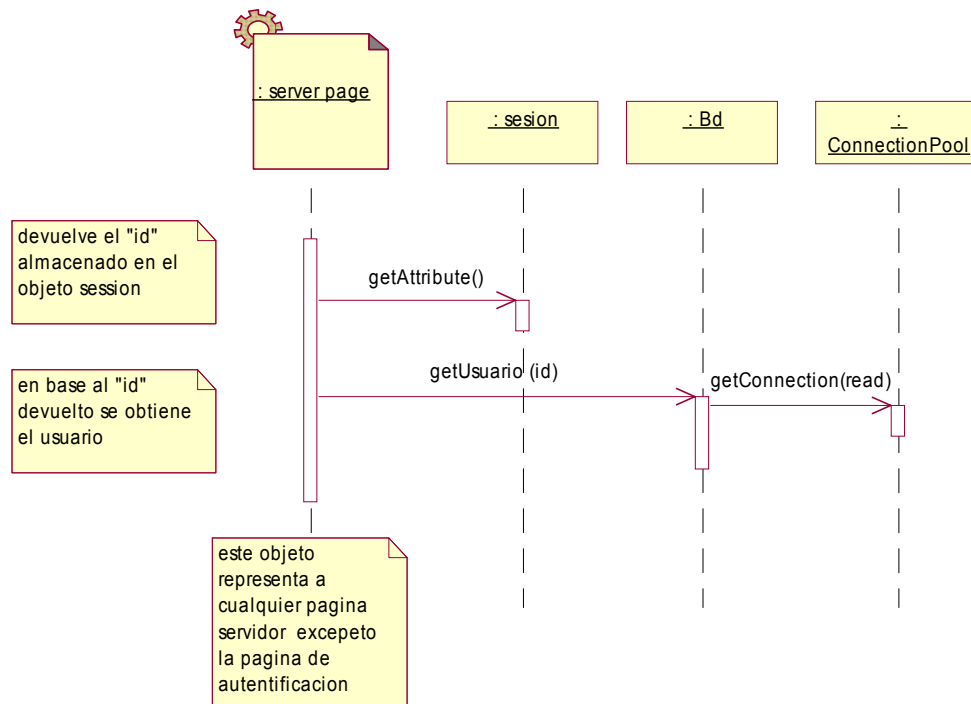


Fig. 5.4

Devolver usuario

El diagrama de interacción, de la figura 5.4, tiene como función devolver un “usuario” que ya ha pasado la autentificación respectiva, estos pasos son utilizados por los siguientes casos de uso pertenecientes al subsistema de consultas, debido a que es necesario reconocer si un usuario se ha autentificado. Los mismos son realizados antes de cualquier operación desde cualquier página servidor excepto por la página de autentificación. Incluso por la página de opciones “menu” del anterior caso. Primeramente se obtiene el valor o “id” almacenado en el objeto sesión y con este valor se obtiene el usuario almacenado en la base de datos, a través del método “getUsuario” del objeto “Bd”, el que a su vez, lo hace a través del objeto “ConnectionPool”.

LISTADO DE CUENTAS FINANCIERAS.-

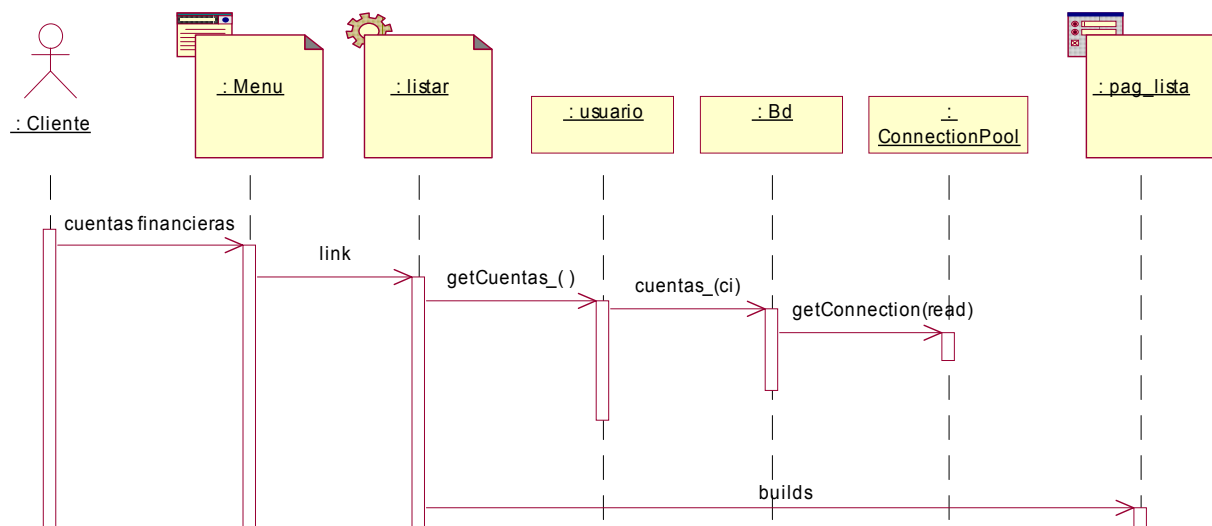


Fig. 5.5

Listado de cuentas financieras

El cliente ha seleccionado la opción de listar cuentas desde el menú de opciones. El objeto “listar” llama a la función `getCuentas()` del objeto “usuario”, que devuelve las cuentas financieras que posee el cliente y sobre estos datos se construye la lista de cuentas. Los datos de las cuentas son obtenidas mediante el método “cuentas” del objeto “Bd” en base al “CI” del cliente. Es preciso hacer notar que este valor, puede representar en caso de personas jurídicas o institucionales, un valor como el RUC de la empresa que represente unívocamente al cliente de la institución. El diagrama 5.5 muestra los pasos aquí descritos.

EXTRACTO DE CUENTA FINANCIERA.-

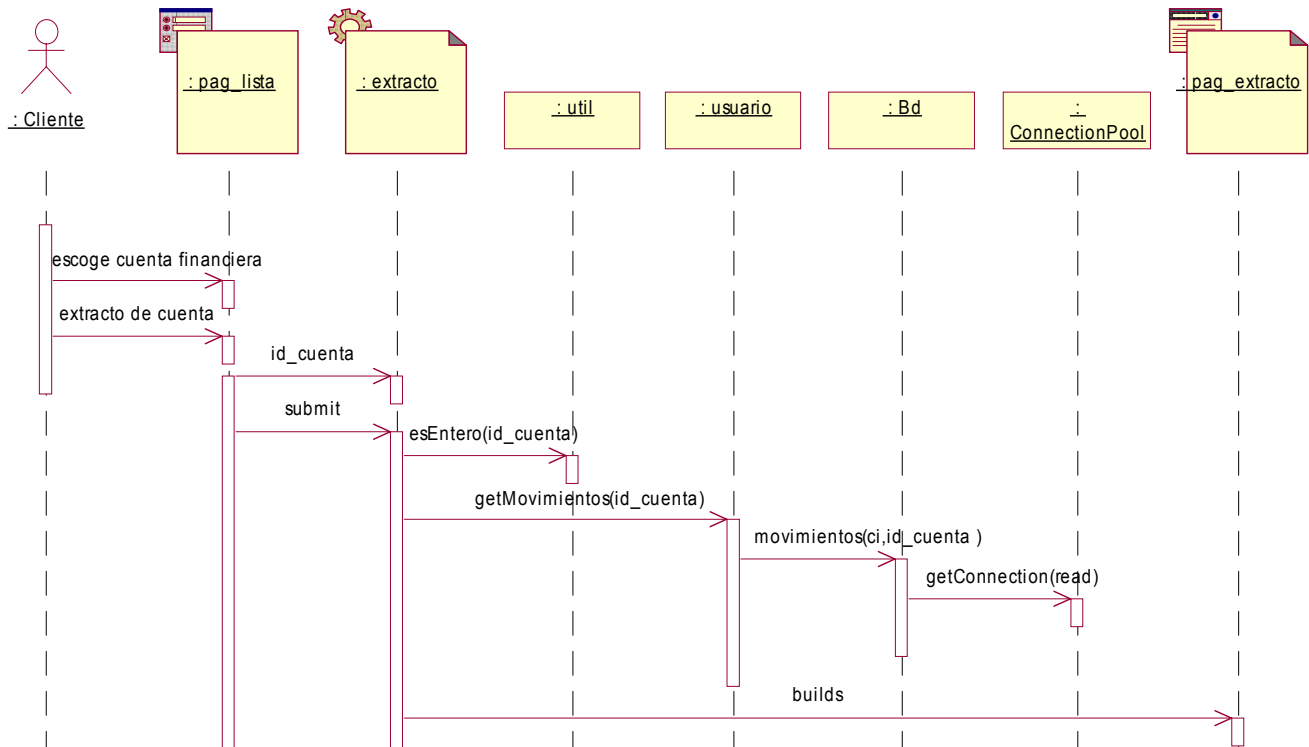


Fig. 5.6

Extracto de cuenta financiera

Una vez que se tiene la lista de cuentas financieras de un determinado cliente, se especifica la cuenta y se selecciona la opción de “extracto de cuenta”, como se ve en el diagrama 5.6. En el servidor antes de realizar cualquier operación, se valida cualquier dato que llega al mismo, por lo que también se validará el identificador de cuenta pasado por el cliente. Esto, a través del método “esEntero” del objeto “util”. Basándose en el “id_cuenta”, se invoca a la operación “getMovimientos” del objeto usuario, el que a su vez lo obtiene de la base de datos mediante el método “movimientos” del objeto “Bd”. Los argumentos pasados a este método son, necesariamente el mismo identificador de cuenta pero además el CI, que representa el CI o RUC del cliente, esto debido a que un determinado cliente solo puede

obtener los movimientos de las cuentas propias y no ajenas. Finalmente se construye la página “pag_extracto”.

PLAN DE PAGOS.-

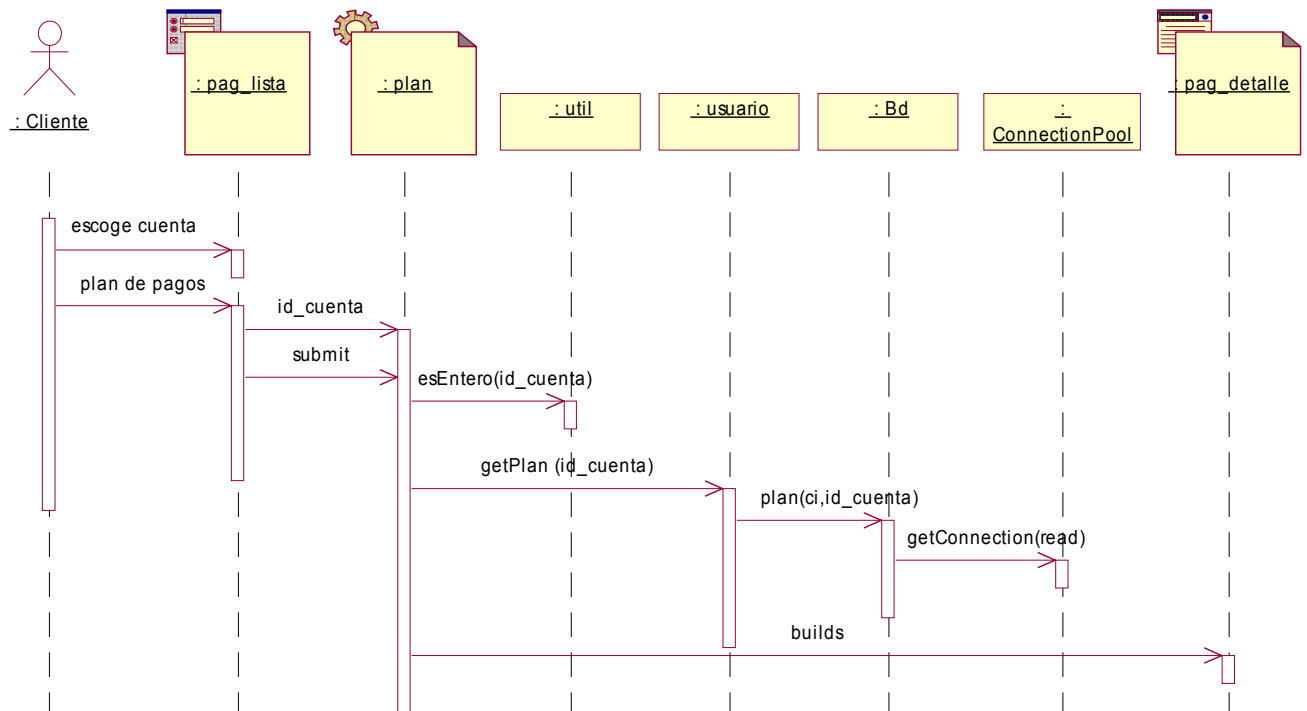


Fig. 5.7
Plan de pagos

De la lista de cuentas financieras se puede obtener el plan de pagos de una determinada cuenta tanto de créditos como de depósitos a plazo fijo. Al igual que en el anterior caso, cualquier dato que llega al servidor es validado. Basándose en el “id_cuenta”, se llama a la operación “getPlan” del objeto usuario, el que a su vez lo obtiene de la base de datos mediante el método “plan” del objeto “Bd”. Para finalizar la página “plan” construye la página “pag_extracto”. El diagrama de la figura 5.7 muestra los pasos aquí descritos.

CAMBIO DE CONTRASEÑA.-

El usuario debe introducir la contraseña con que ingreso al sistema, conjuntamente con la nueva contraseña y confirmar la misma. La pagina cliente "pag_cambio" realizara la validación antes que los datos sean enviados al servidor, a través del método "val". Una vez que los datos llegan al servidor, se los valida nuevamente mediante el método "valido" del objeto "util". La contraseña que se tenia anteriormente, así como la nueva contraseña son pasadas como argumentos al método "cambiar" del objeto usuario. Desde este método se comprueba si la contraseña anterior introducida coincide con la que se tiene registrada y encriptada en el sistema. Antes de guardar los cambios en la base de datos a través de "setUpdate" del método Bd, la nueva contraseña es encriptada. Por último se registra la operación efectuada y se despliega el resultado del cambio. El diagrama 5.8 muestra estos pasos.

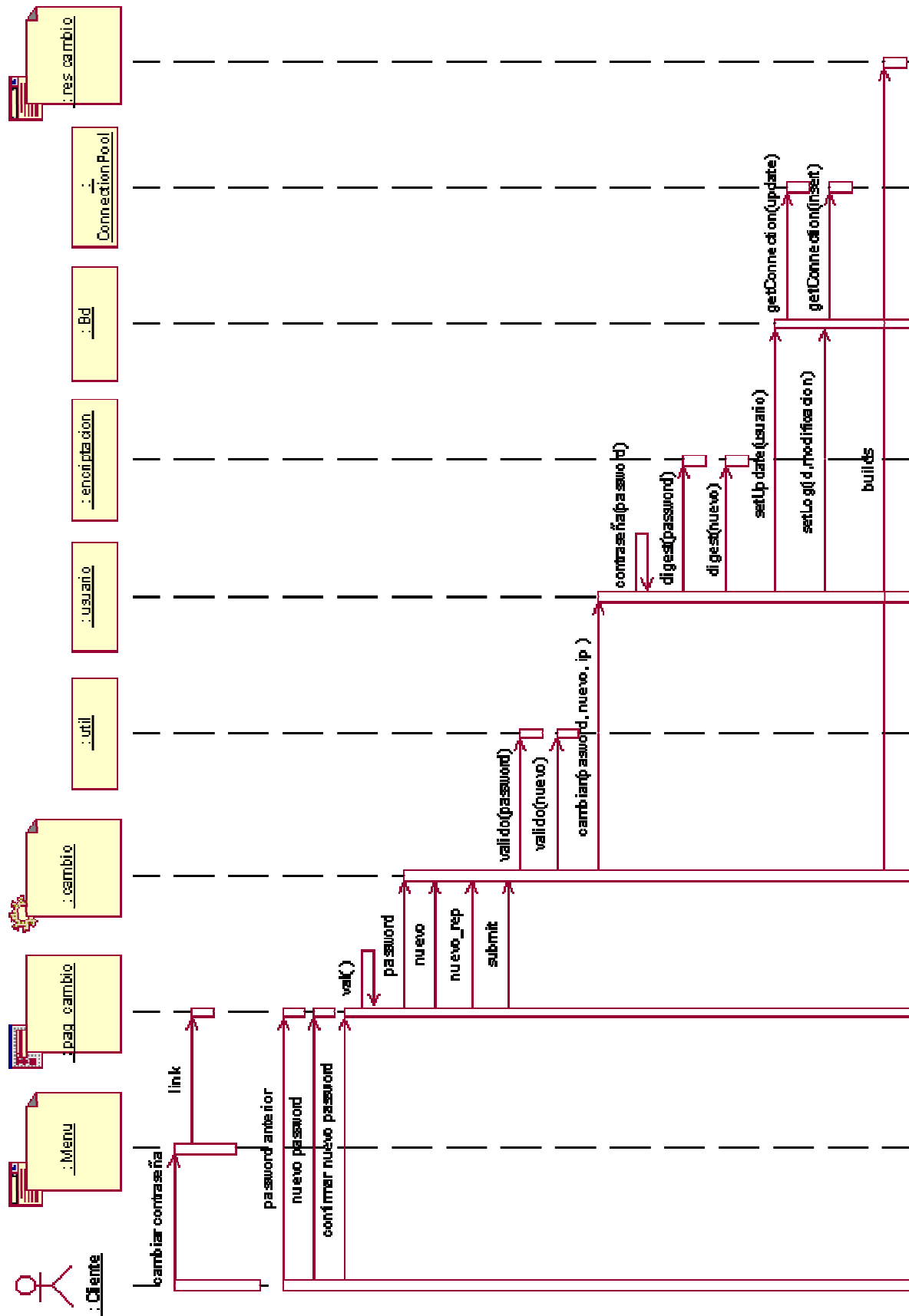


Fig. 5.8

Cambio de
contraseña

SALIR.-

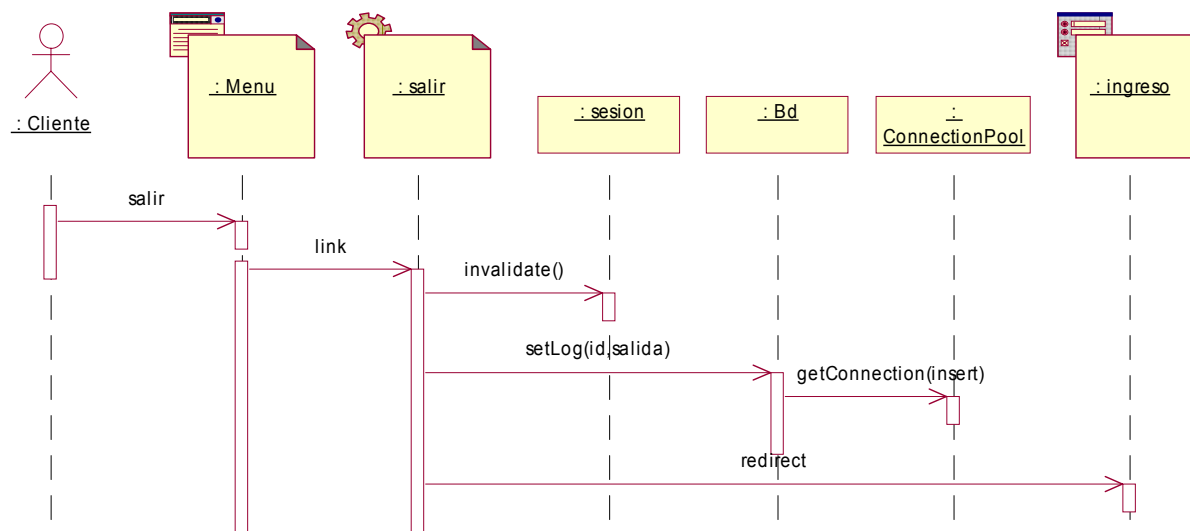


Fig. 5.9

Salir

El usuario ha seleccionado la opción “salir” del sistema, como se ve en el diagrama 5.9, por lo que se abandona la sesión creada al ingresar al sistema, luego se registra esta operación y por último se redirecciona a la página de ingreso.

5.3.2) PARA EL ADMINISTRADOR DEL SISTEMA

AUTENTIFICACIÓN Y LISTADO DE USUARIOS.-

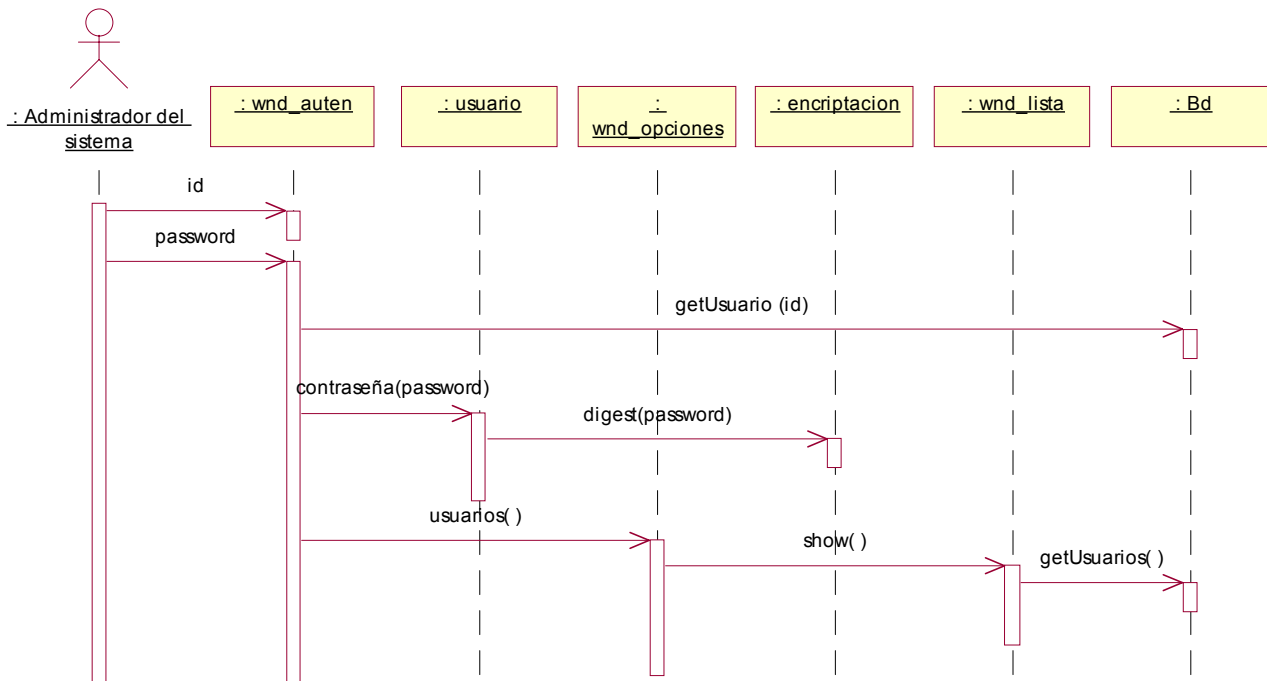


Fig. 5.10

Autenticación y listado de usuarios

El diagrama de interacción, de la figura 5.10, muestra como son realizados los casos de uso “autenticar administrador” y “listado de usuarios”. Las funciones involucradas en la autenticación son similares al subsistema de consultas pero en este no se hace uso de páginas web, debido a que este subsistema es ejecutado en el servidor, en su lugar se hace uso de diálogos o ventanas. Una vez autenticado al administrador, se registrara este ingreso y el sistema mostrara las opciones que se pueden realizar en este sistema a través del objeto “wnd_opciones”.

La primera opción que se desplegara al administrador será el listado de usuarios, a través del objeto `wnd_lista`, que a su vez lo obtiene de la base de datos mediante el método “getUsuarios”. Es necesario aclarar que no se registra esta operación de listado.

CREAR USUARIO.-

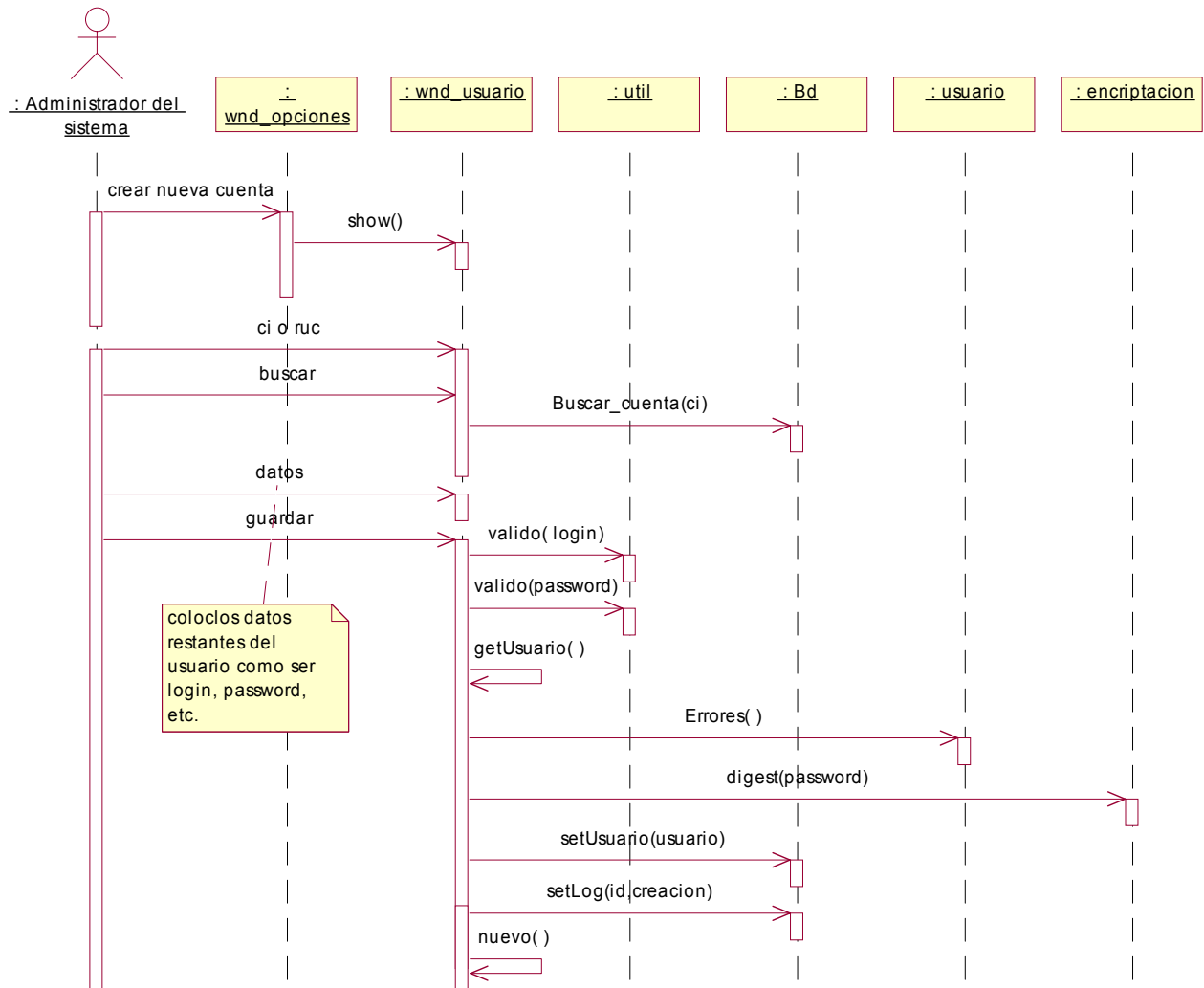


Fig. 5.11
Crear usuario

Desde la ventana de opciones, se selecciona la opción de “usuarios”, que a su vez llama al objeto “wnd_usuario” que muestra la ventana donde se introducirán los datos. El administrador coloca el primeramente el CI o RUC del nuevo cliente que se esta creando, para que el sistema busque las cuentas financieras que están relacionadas con el mismo. Luego se introducen los datos restantes como ser el login y contraseña. Estos datos merecen

una verificación adicional ya que los mismos llegarán del usuario a través de Internet, por lo que se deberá controlar que no posean caracteres considerados peligrosos. Si algún dato introducido no es válido, se lo verificará a través del método “Errores” del objeto “usuario”. Si no existe ningún error, la contraseña es encriptada mediante el método “digest” de “encriptacion”. Completado todo esto se procede a la introducción del nuevo usuario en la base de datos por medio del método “setUsuario” de “Bd”. Por último se registra la operación efectuada y se limpian los valores de la ventana. Los pasos aquí descritos son visualizados en el diagrama 5.11.

ELIMINAR USUARIO.-

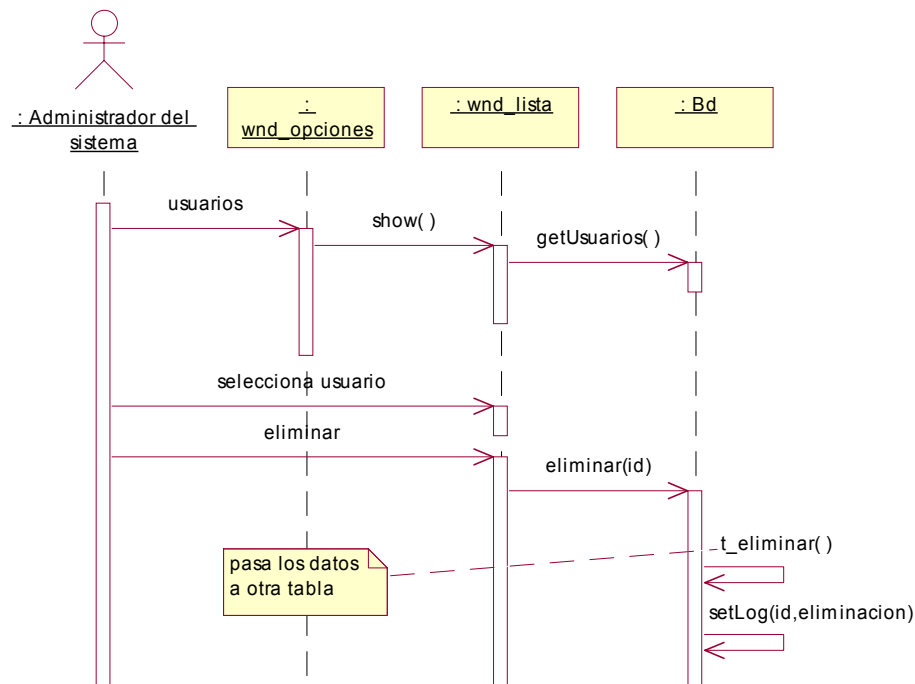


Fig. 5.12
Eliminar usuario

Para retirar un cliente del sistema, el administrador debe generar previamente la lista de usuarios total que existe en el sistema. De la misma, deberá seleccionar el usuario y optar por la opción de eliminación. El id del usuario es pasado como argumento al método "eliminar" del objeto "Bd", el que internamente llama al método "t_eliminar" que realiza el

traspaso de los datos del usuario a eliminar hacia una tabla de usuarios eliminados. Después dentro del método eliminar se procede a la eliminación del usuario y el registro de esta operación. El diagrama 5.12 muestra los pasos aquí descritos.

ACTUALIZAR USUARIO.-

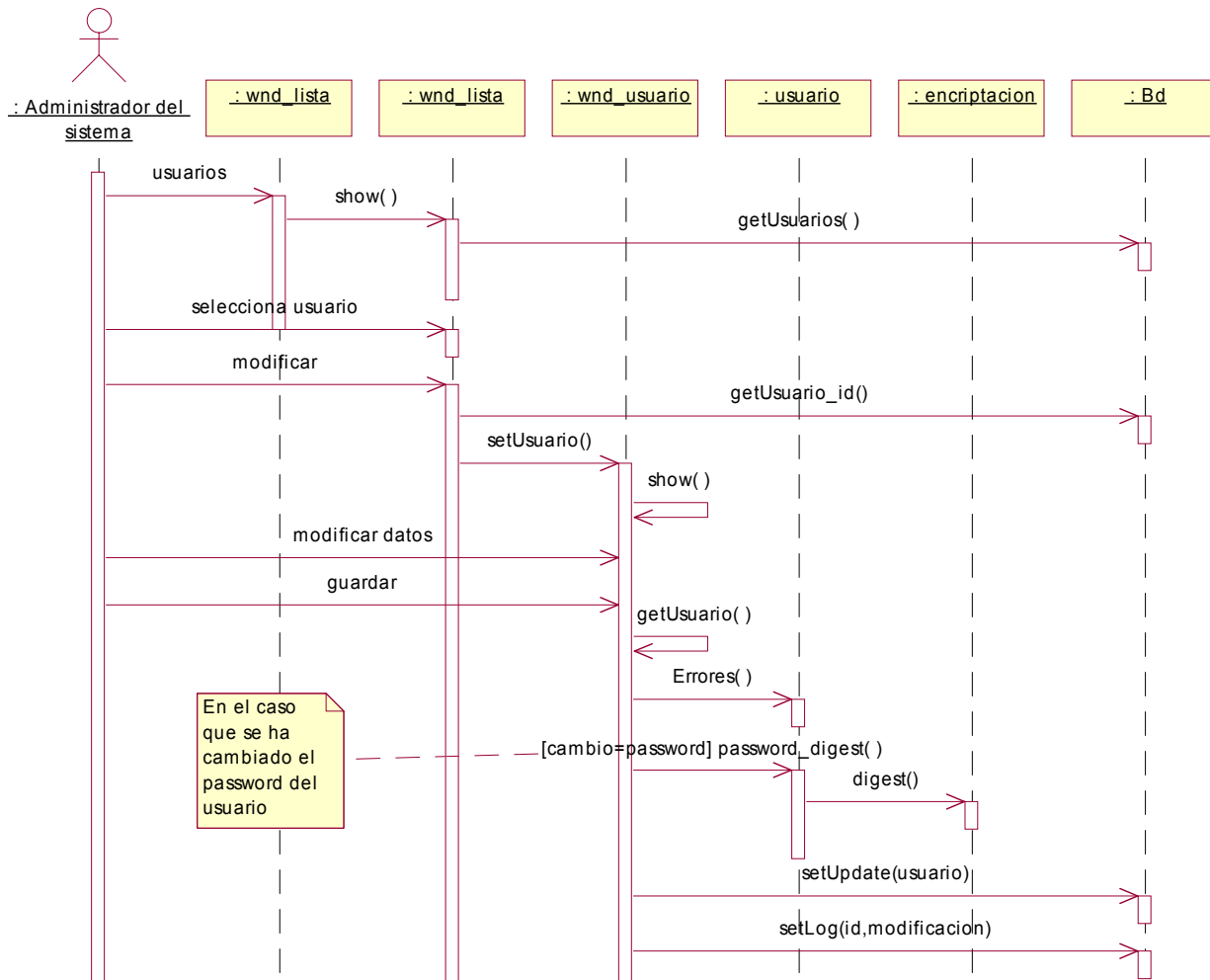


Fig. 5.13
Actualizar usuario

Para actualizar los datos de un usuario primeramente se debe mostrar la lista de usuarios que existen en el sistema, luego se debe seleccionar al usuario a actualizar. Sobre la ventana “wnd_usuario”, la misma que se tiene para la creación de un nuevo usuario, se despliegan

los datos actuales del usuario. El administrador entonces modifica cualquier valor mostrado y antes de guardar los cambios se validan nuevamente todos los nuevos datos, a través del método “Errores” del objeto Usuario. Se debe controlar si se ha cambiado la contraseña que se tenía y si es el caso se debe encriptar la misma. La descripción corresponde al diagrama de interacción 5.13.

LISTAR CUENTAS FINANCIERAS.-

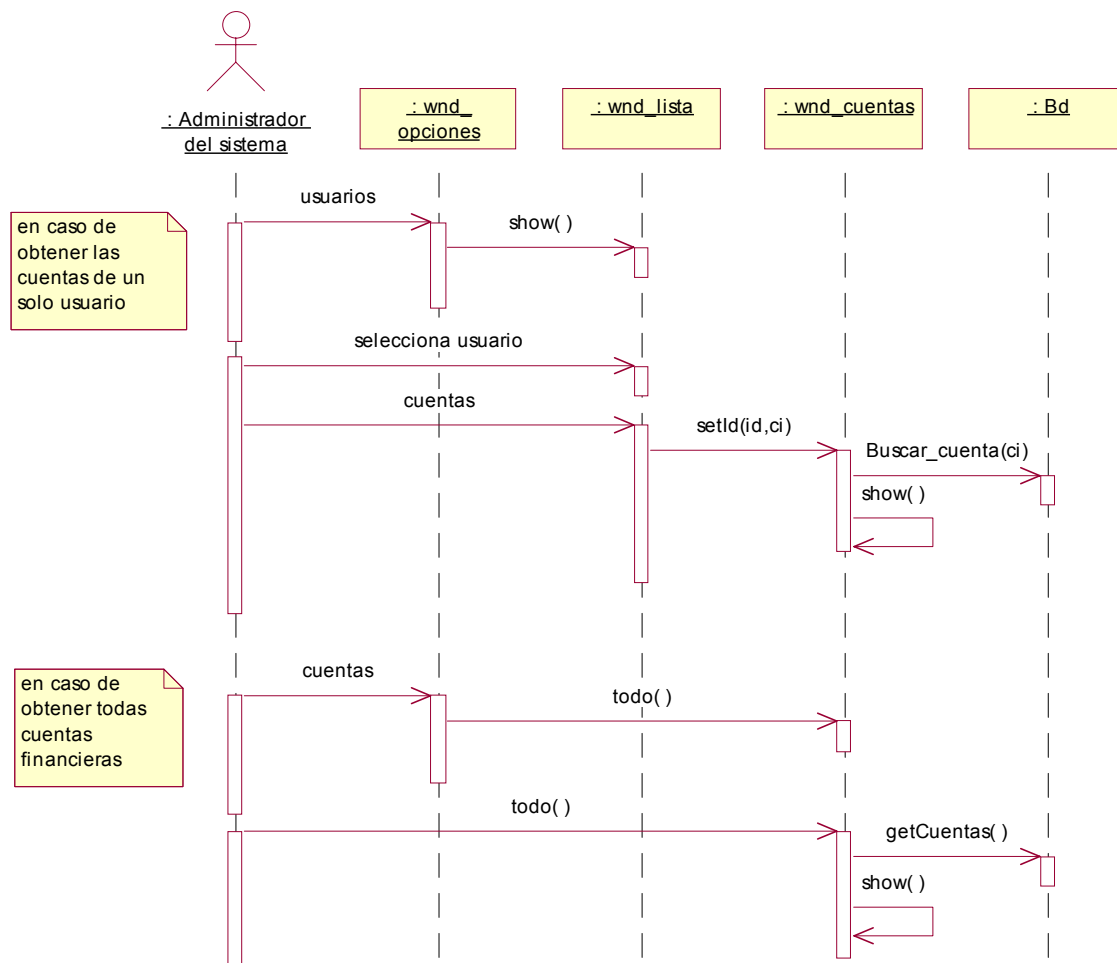


Fig. 5.14

Listar cuentas financieras

Se pueden listar las cuentas financieras de un determinado cliente o todas las cuentas financieras registradas en el sistema. Para el primer caso, para desplegar las cuentas de un cliente se debe desplegar la lista de usuarios, desde la que se cuenta con una opción de

“cuentas” la que a su vez llama al objeto “wnd_cuentas”, pasando como argumento el id de usuario, así como también el CI del mismo. Con este valor se busca en la base de datos las cuentas asignadas a este CI. Después se despliega todas las cuentas en la ventana “wnd_cuentas”.

Para el segundo caso solo se debe optar por la opción todo, en la ventana de “wnd_cuentas”. Ambas acciones son mostradas en el diagrama de interacción 5.14

LISTAR LOGS.-

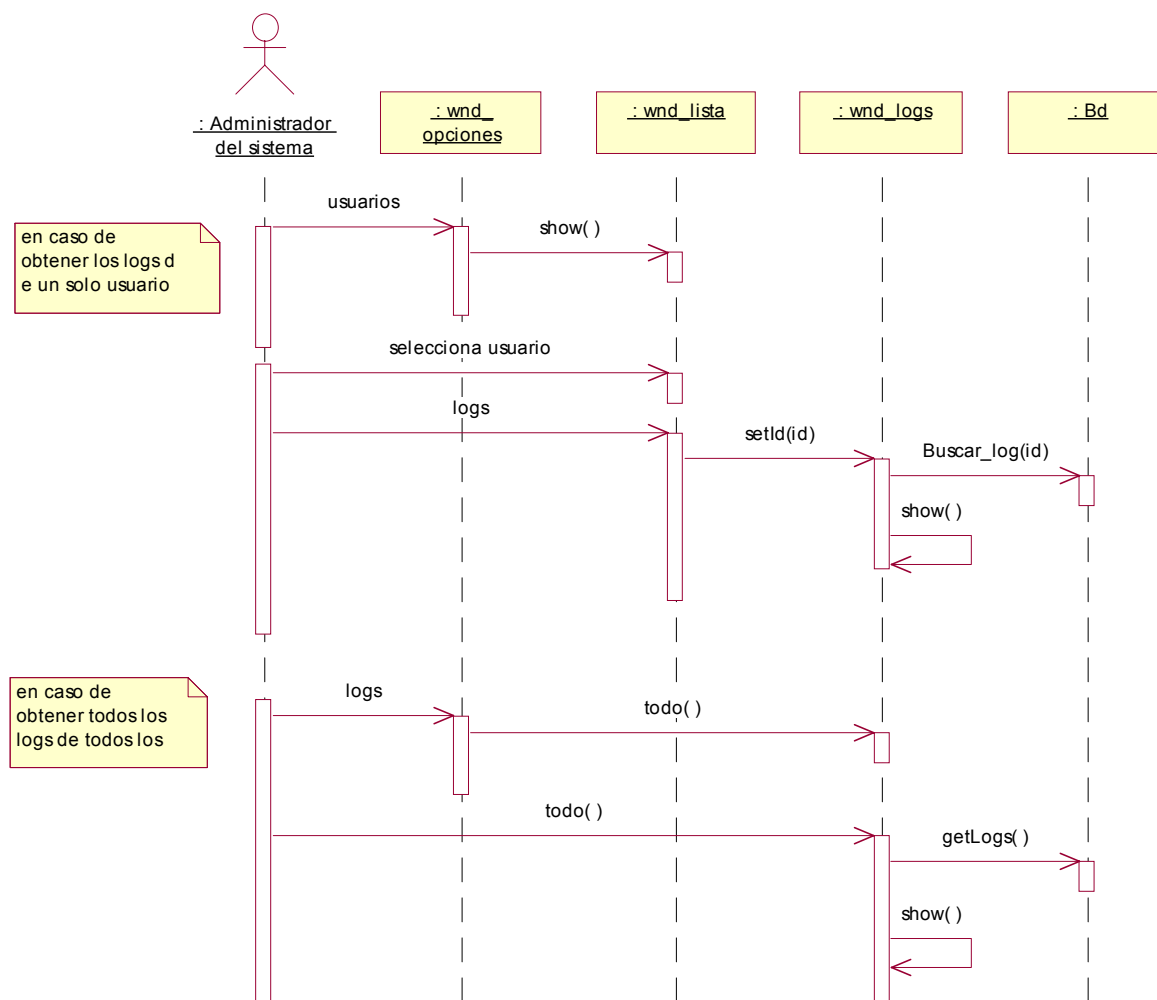


Fig. 5.15
Listar logs

De la misma manera que el listado de cuentas financieras, descrito en el anterior caso, en el listado de logs, se puede optar por un listado completo de todas las operaciones efectuadas en el sistema o solo aquellas realizadas por un determinado usuario. Ambos listados no constituyen en si operaciones que sean necesarias registrarlas. La descripción corresponde al diagrama de interacción 5.15.

VACIAR LOGS Y GENERAR LLAVE.-

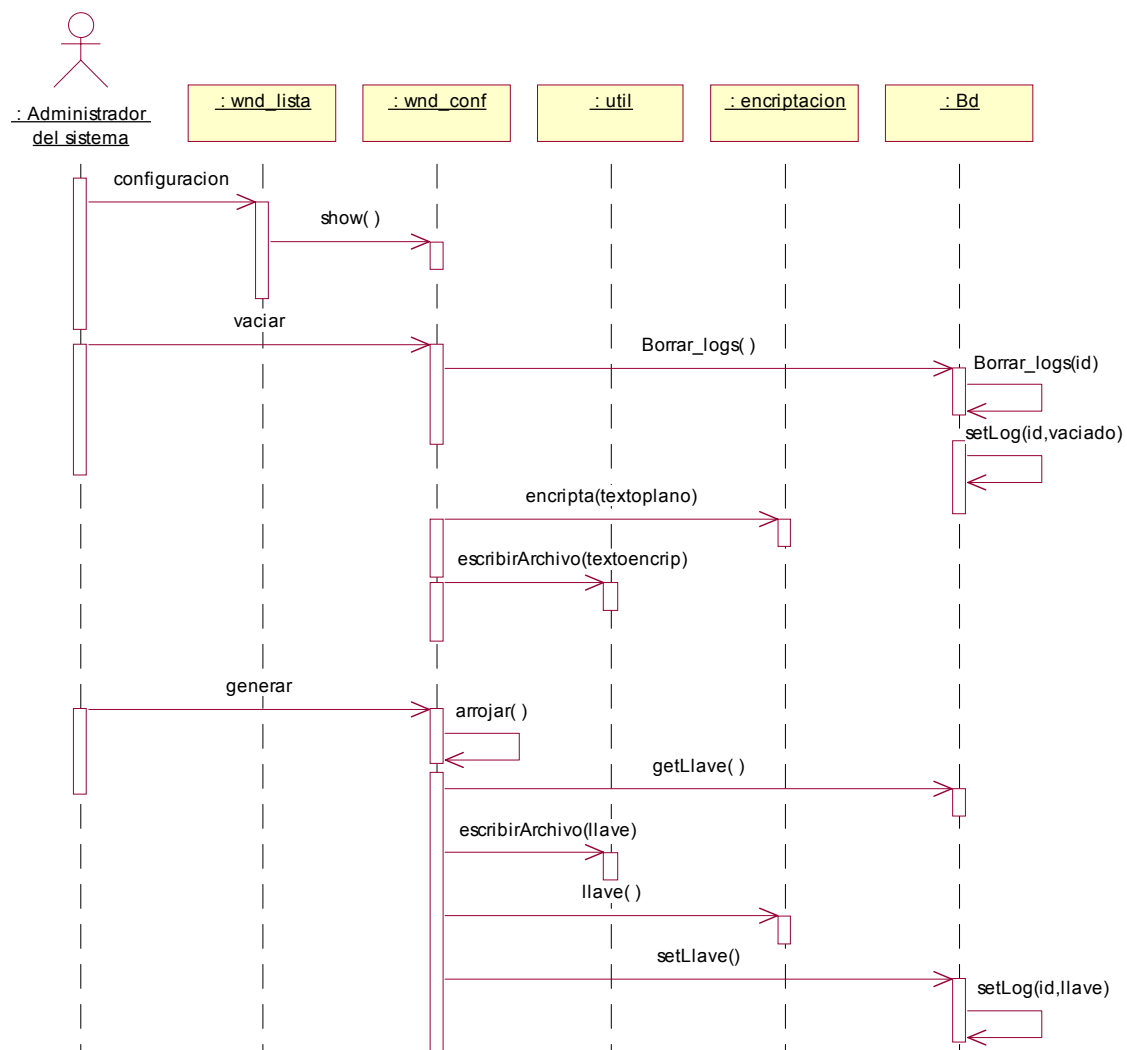


Fig. 5.16

Vaciar logs y generar llave

El diagrama 5.16, muestra como son realizados los casos de uso “vaciado de logs” y “generación de llave”. El sistema realiza un vaciado de todas las operaciones registradas en la base de datos a través del método “Borrar_logs”. Los datos a ser vaciados son encriptados y se los arroja a un archivo externo a través del método “escribirArchivo” del objeto “util”. Sin embargo una vez realizado el vaciado, se registra la misma operación de vaciado. Para la generación de una nueva llave el sistema necesariamente arrojara la que se tenía anteriormente hacia un archivo externo.

6. Diseño

En base a los resultados obtenidos en la etapa de análisis se puede definir la arquitectura del sistema.

6.1.- ARQUITECTURA LÓGICA

La arquitectura lógica del sistema es descrita mediante diversos diagramas aquí expuestos. Se presenta la arquitectura lógica a través del diagrama de clases a nivel superior, donde se encuentran agrupamientos de clases en categorías o paquetes de clases y las dependencias entre estos. Un paquete depende de otro, si se comunica con alguna clase de este.

Se presenta en el diagrama 6.1, las categorías de clases dividida en tres agrupaciones, basado en la arquitectura cliente/servidor. Las clases encargadas de la interfaz de usuario están contenidas en la categoría de clases “User Services”. Las clases encargadas de la aplicación o lógica de negocios están en la categoría intermedia “Business Services” y en la categoría “Data Services” se encuentran todo el manejo de datos.

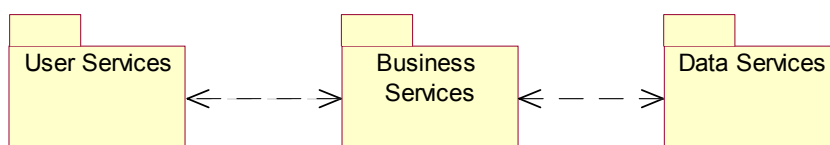


Fig. 6.1
Categoría de clases principal

A su vez la categoría de clases Business Services esta dividida en otras subcategorías y lo conforman los paquetes de “server pages”, “business” y “encriptacion”.

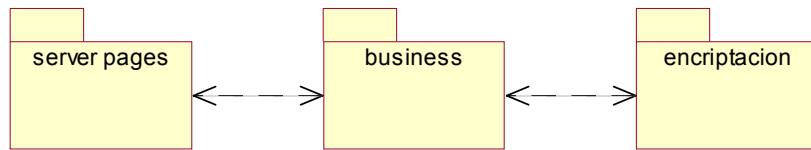


Fig. 6.2
Subcategorías de business services

A continuación se detallan las clases y relaciones contenidas en estas categorías.

Categoría User Services

Para hallar las clases contenidas en esta categoría se examinan los pares actor - diagrama de interacción, ya que manejan la comunicación entre un actor y el sistema.

En el diagrama 6.3 se muestra las clases encargadas de la interfaz del subsistema de administración. Estas están relacionadas por asociación, ya que existe un enlace o conexión entre los objetos de las clases asociadas, observado en los diagramas de interacción del subsistema de administración. La especificación de estas clases se encuentra en el Anexo 2 de Especificación de Clases.

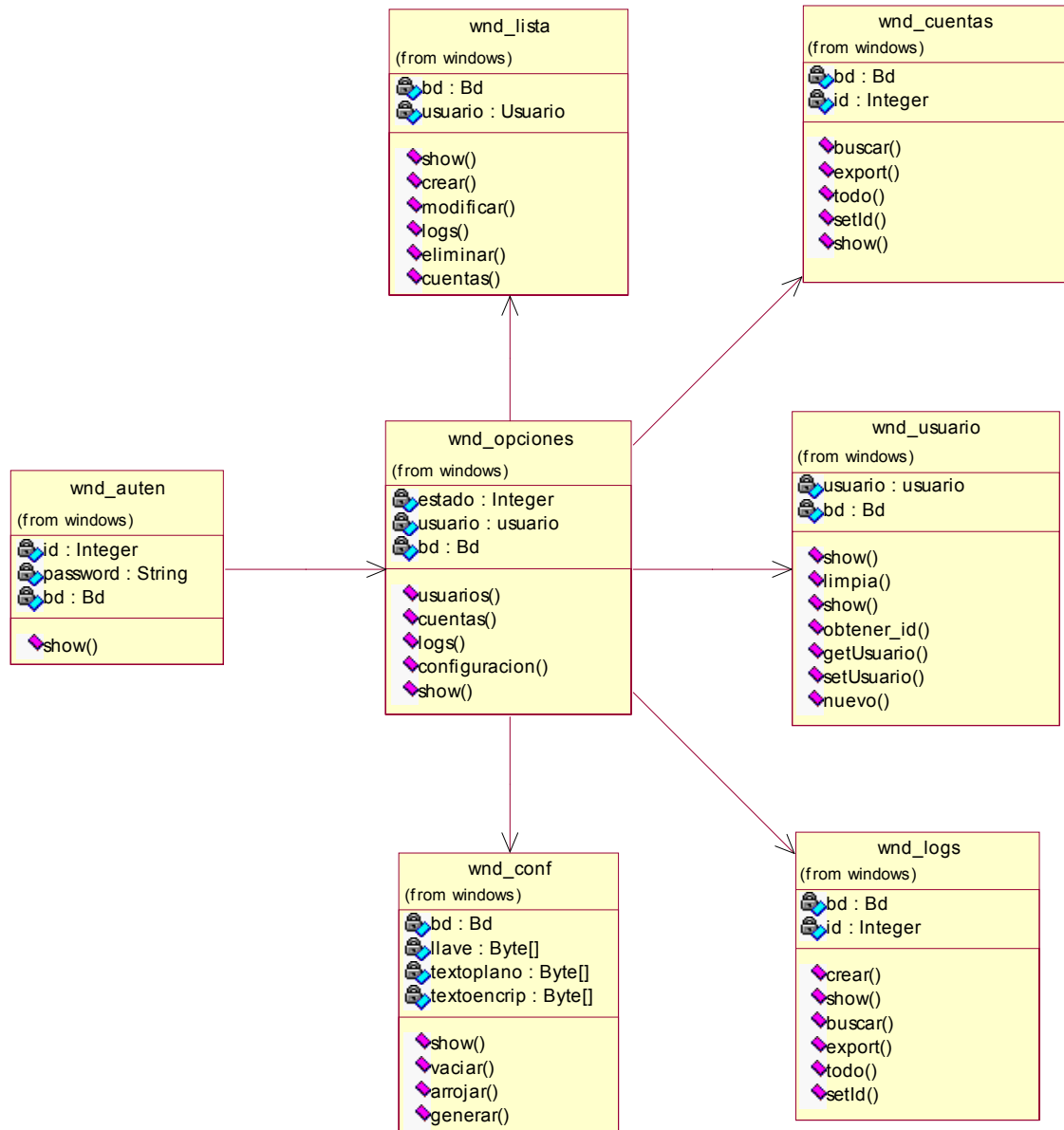


Fig. 6.3
Categoría user services

Para las clases correspondientes al subsistema de consultas, se ha decidido juntar las categorías de User Services y Server Pages subcategoría de Business Services en un solo, debido a las razones expuestas a continuación.

Categorías User Services y Server Pages (Business Services)

En aplicaciones web, el camino de navegación a través del sistema es sumamente importante para entender la aplicación (concepto de mapa del sitio) [CON99], por lo tanto es necesario mostrar un diagrama navegacional que además muestre la relación que existe entre las clases contenidas en los paquetes “User Services” y “Server Pages”. En este diagrama se utilizan los estereotipos correspondientes a cada clase para una mejor comprensión, como se muestra en la figura 6.4.

Posteriormente, sobre las clases identificadas y sus relaciones se definen los atributos y métodos de las mismas, mostrado en el diagrama detallado de la figura 6.5.

Es importante aclarar que los atributos de algunas clases correspondientes a la categoría User Services, son estereotipos, como por ejemplo en la clase ingreso, se tienen los atributos de “id”, “password” y “ok”. Estos no poseen un tipo de variable conocido como podría ser un String o cadena, sin embargo estos son “input fields” como ser <<text>>, <<password>>, <<submit>>, <<radio>> etc. correspondiente al estereotipo <<form>>¹³. Para estas y otras clases se tiene la especificación en el Anexo 2 de Especificación de clases.

¹³ Ver Anexo 1, Estereotipos

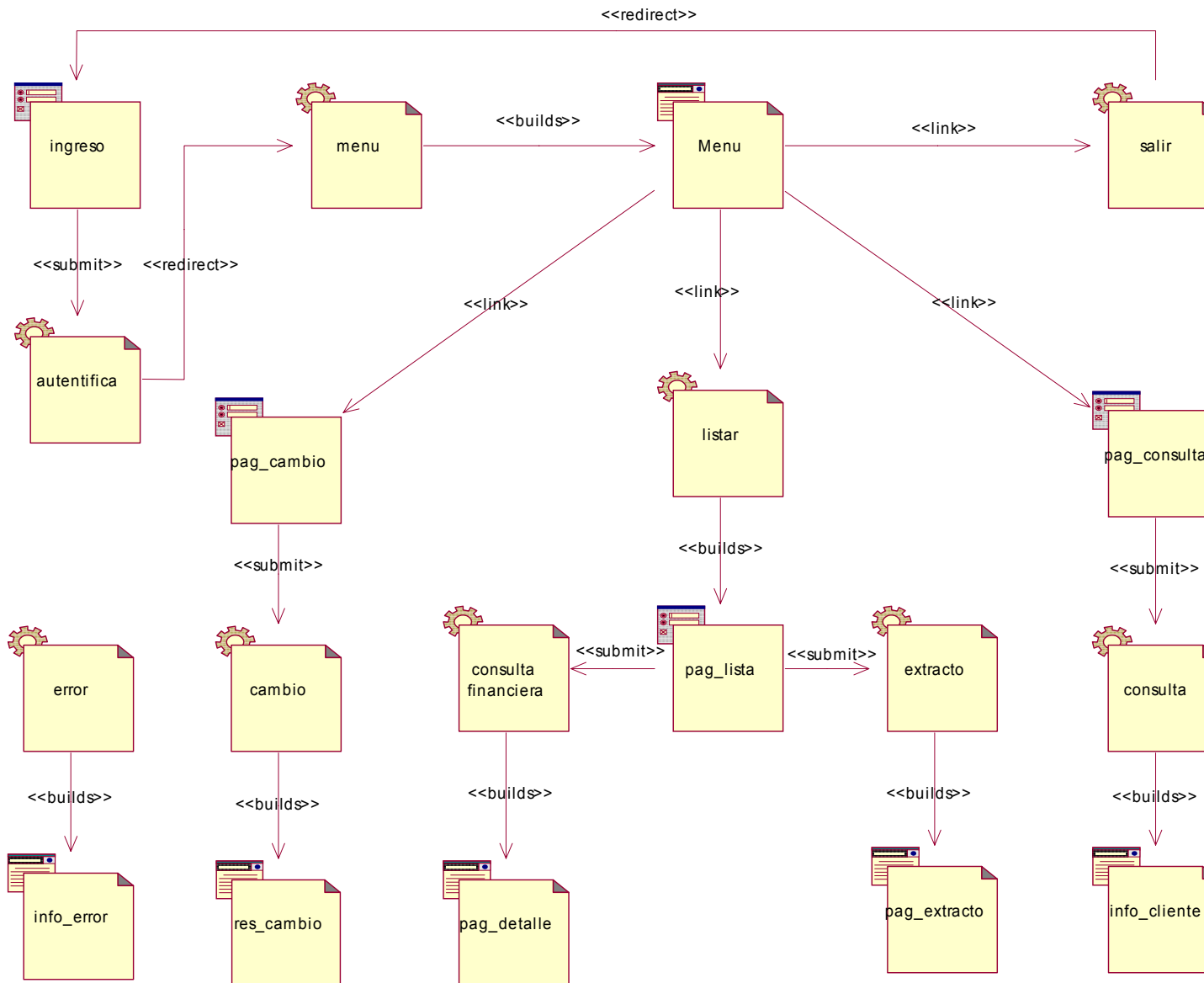


Fig. 6.4
Diagrama
navegacional

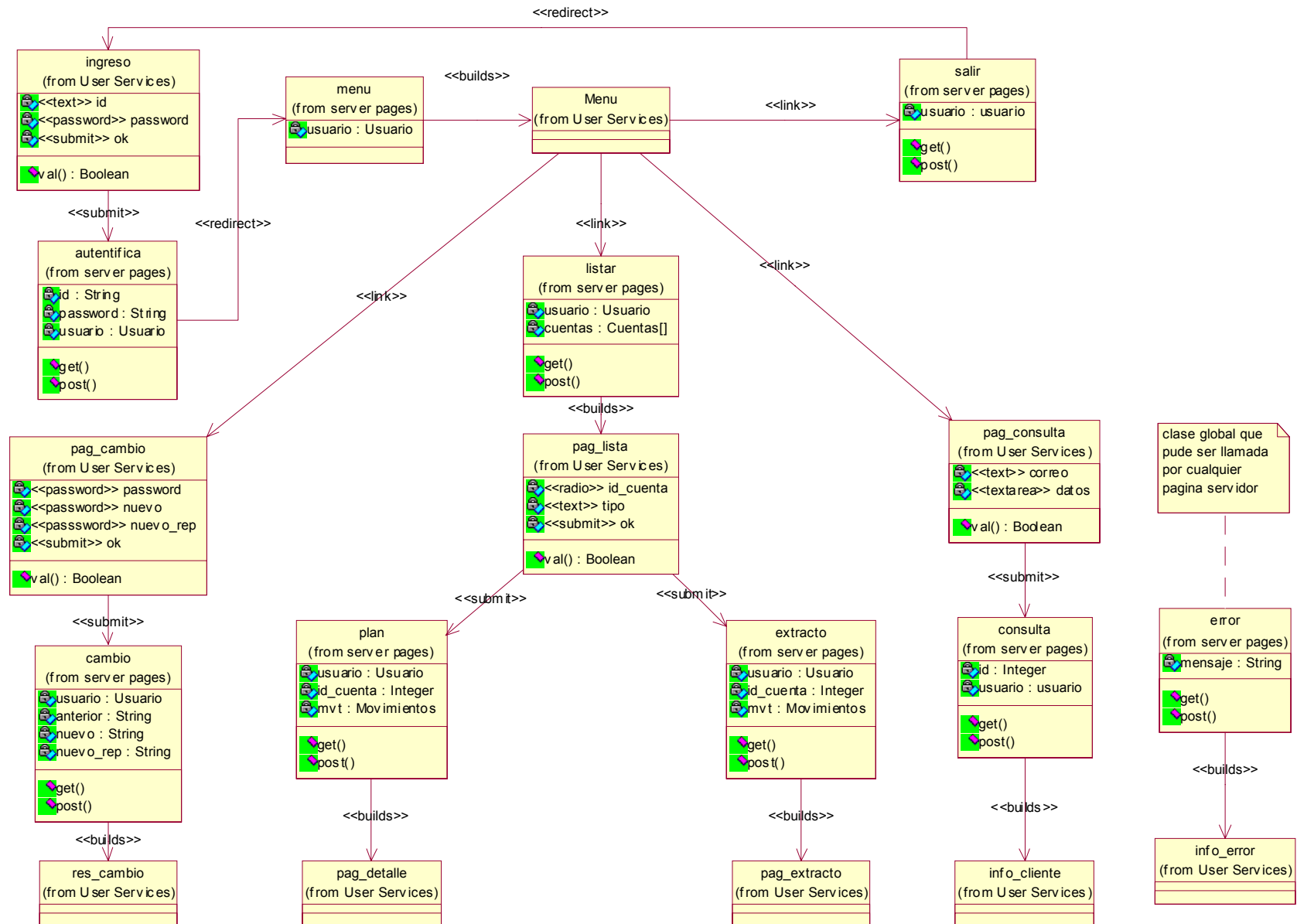


Fig. 6.5
Diagrama detallado

Categorías Business y Encriptación (Business Services)

Las clases contenidas en estas categorías son usadas tanto en el subsistema de consultas como el de administración.

La categoría encriptación esta constituida únicamente de la clase “encriptación” pero esta relacionada con otras clases dentro la categoría “Business”. En esta se hace uso de algoritmos de encriptación simétricos y de funciones de resumen, contrariamente de SSL donde se hace uso tanto de algoritmos asimétricos como simétricos. El protocolo SSL protege la comunicación, pero una vez que los datos llegan al servidor también es necesario protegerlos, pero se los encripta o desencripta usando solamente algoritmos simétricos puesto que la llave no va a ningún lado, además que la velocidad de estos es más rápida que sus similares asimétricos.

Por conveniencia se usa cadenas en la manipulación de los datos encriptados en lugar de solamente bytes. Esto por que facilita tanto al recuperar como al almacenar datos en la base de datos.

A continuación se muestra las funciones miembro y los datos miembro más importantes de la clase Encriptación.

Class Encriptación

```
{
    private String algoritmo = "Blowfish";
    private int longitud = 448;
    private byte[] llave ;

    public byte[] llave() throws GeneralSecurityException
    public byte[] encrip(byte textoplano[], byte llave []) throws GeneralSecurityException
    public byte[] desencrip(byte textoencrip[], byte llave []) throws GeneralSecurityException
}
```

En esta clase, se define el algoritmo a utilizar y la longitud de la llave. Puesto que estamos limitados a las librerías de encriptación¹⁴ se tienen en la tabla 6.1 los algoritmos simétricos soportados:

Nombre	KeySize
DESEngine	64 bits
DESedeEngine (TripleDES)	128,168 bits
IDEAEngine	128 bits
RC6Engine	0..256 bits
RijndaelEngine	0..256 bits
SerpentEngine	128,192,256 bits
BlowFishEngine	0..448 bits

Tabla 6.1
Algoritmos de encriptación

Se tiene en el anexo 4, un cuadro comparando el rendimiento de diferentes algoritmos, donde se observa que el algoritmo “RC6” posee el mejor desempeño. Sin embargo, ya que, en general la robustez de un algoritmo depende de la longitud de la llave¹⁵, es mejor elegir aquel que nos otorgue una mayor longitud y al mismo tiempo una buena velocidad de encriptación, por lo que se ha optado por el algoritmo “Blowfish”, con una longitud de 448 bits.

La clase “util”, es usada por diversas clases pero sirve principalmente para la validación de datos así como también tanto para la lectura como escritura de archivos.

Dentro la categoría Business se encuentran la clase Bd, relacionada principalmente con la clase ConnectionPool, que realiza en sí, las conexiones a la base de datos, por lo que posee en sus atributos, el login y contraseña de acceso a la misma.

¹⁴ Ver la sección [7.1] en las herramientas de software, las librerías de encriptación que se utilizan son JCE.

¹⁵ Ver la sección [3.3] Algoritmos criptográficos.

El usuario posee cero o más cuentas de crédito, depósito a plazo fijo o caja de ahorro. Los atributos de estas clases son explicados en detalle en el Anexo 2.

Todas estas clases y relaciones se ven reflejadas en el diagrama 6.6 mostrado a continuación.

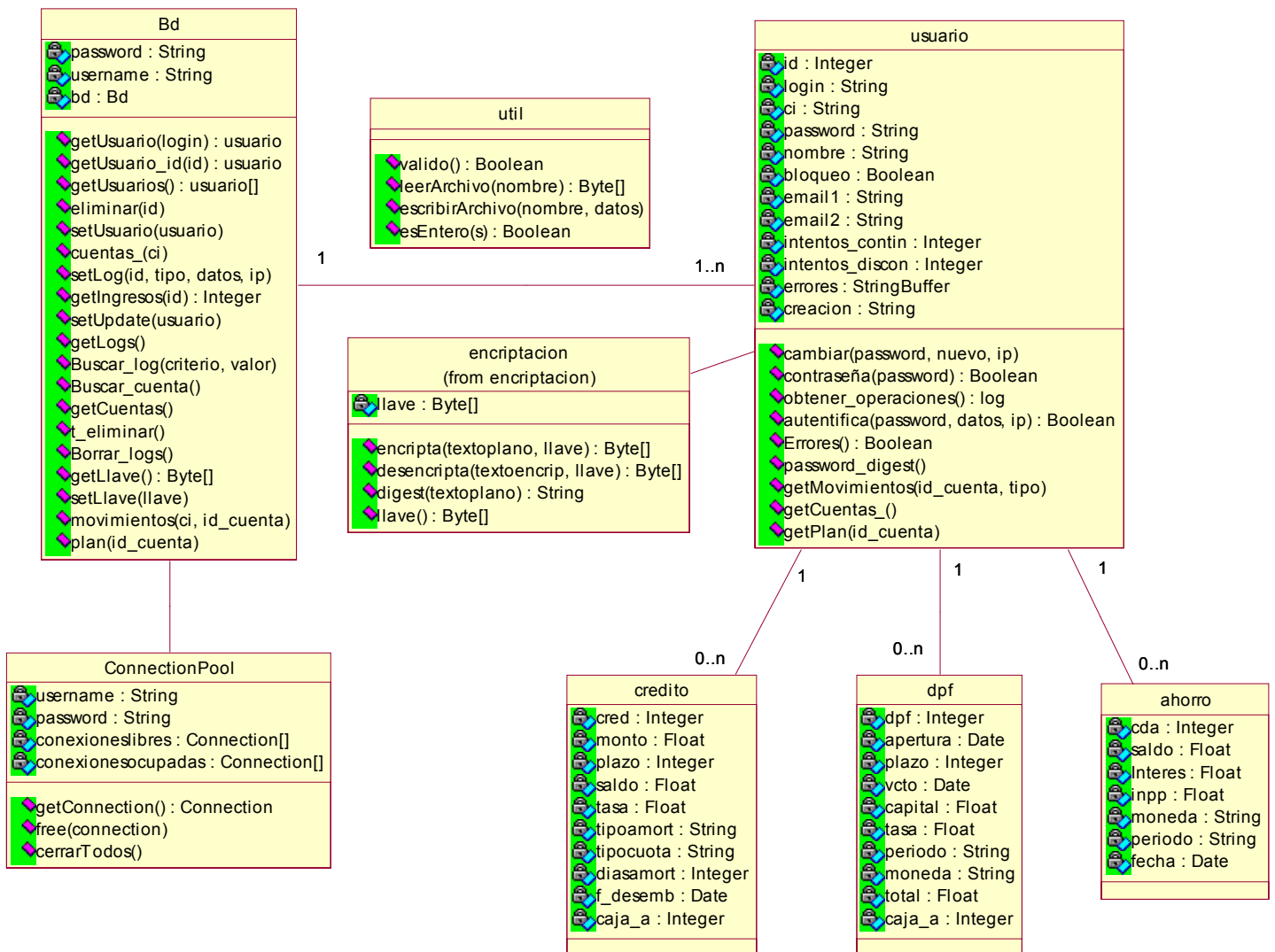


Fig. 6.6
Categoría business

Categoría Data Services

En esta categoría, cuyo diagrama de clases se muestra en el diagrama 6.7, se muestran las tablas que conforman la base de datos del sistema. Las relaciones que existen entre las tablas aparecen cuando una clave primaria constituye una clave externa en otra tabla. Por ejemplo la relación entre las tablas “t_tipo” y “t_registro” existe porque la clave primaria “tipo” de la tabla “t_tipo”, es clave externa en la tabla “t_registro”.

Otro tipo de relación existe entre los usuarios y sus cuentas. Un usuario puede tener de 0 a muchas cuentas en depósito a plazo fijo, créditos o caja de ahorro y esta relacionado a través de las tablas “t_dpfr” para las cuentas en depósito a plazo fijo, “t_credr” en cuentas de créditos y “t_cdar” para cuentas de caja de ahorro.

Estas tablas están relacionadas con sus tablas maestro: “t_dpfr”, “t_credr” y “t_cdar” donde se almacena información de la cuenta, como ser fecha de apertura, tasa, interés, etc. la que varia de acuerdo al tipo de cuenta financiera. Estas tablas maestro están relacionadas con otras tablas a través de su clave primaria, que constituye una clave externa en las otras.

Es necesario señalar que una sola cuenta financiera, cualquiera sea su tipo puede poseer uno más usuarios, esto en el caso de personas jurídicas o instituciones. Por eso las relaciones de las tablas t_credr, t_dpfr y t_cdar hacia la tabla usuario, es de uno a muchos.

Una cuenta financiera puede tener de uno a muchos movimientos registrados. Cada movimiento esta contenido en la tablas “t_dpfrq”, “t_credq” y “t_cdaq”. Las tablas “t_dpfa” y “t_creda” poseen información del plan de pagos para cada cuenta en depósitos a plazo fijo y créditos respectivamente.

Por último, se cuenta con una tabla que no tiene relación con ninguna tabla y contiene los datos de los usuarios que han sido eliminados, “t_eliminados”, la estructura de la tabla es la misma que se tiene en la tabla “t_usuario”.

6.2. ARQUITECTURA FÍSICA

En el diseño de la arquitectura física, las clases son asignadas a componentes. Un componente representa un fichero de software que puede contener código fuente, binario o ser una librería, ejecutable, etc. Generalmente cada clase se asigna a un componente, sin embargo como se verá mas adelante puede que más de una clase esté asignada a un componente.

Los componentes mostrados en el diagrama 6.8, mostrado a continuación, corresponden al diagrama de clases User Services de la figura 6.3. Esta correspondencia es uno a uno, por cada clase a cada componente.

En este diagrama se muestra además el componente “aplicacion”, que representa al archivo principal o ejecutable conteniendo la operación “main” y lo único que hace es llamar al método “show” de la clase “wnd_aunten”.

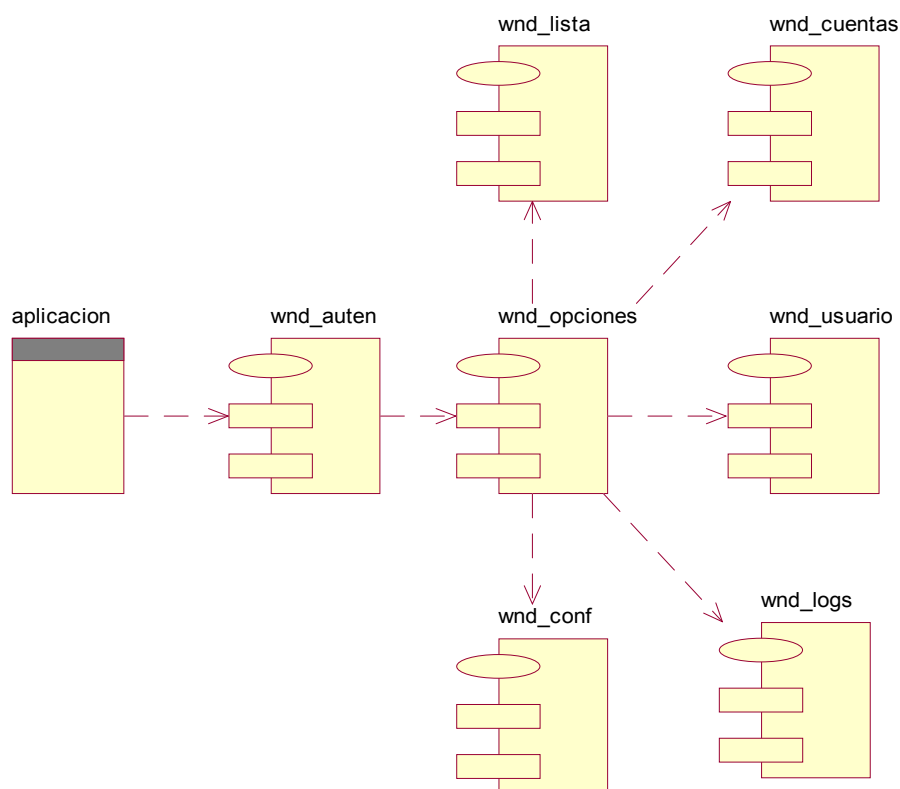


Fig. 6.8

Diagrama de componentes: user services

Algunos componentes en el servidor representan a la vez clases servidor y clases cliente, como se explico en la sección 4.2 de Notación. Por ejemplo el componente “listar” realiza las clases de “listar”, “pag_lista” y “pag_detalle”. La correspondencia completa de cada componente se encuentra en el Anexo 3 de Componentes.

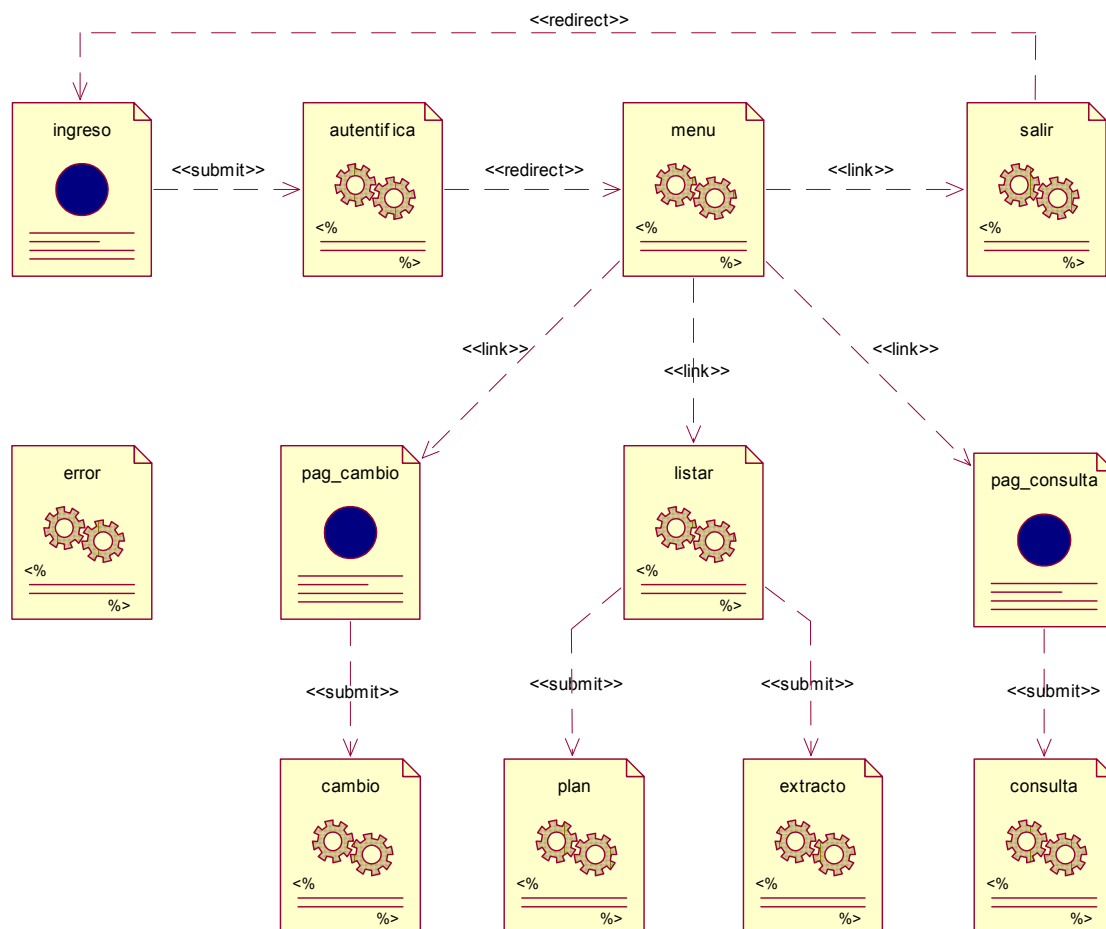


Fig. 6.9

Diagrama de componentes: user services y server pages

Examinando el diagrama de clases de la categoría “Business Service” se puede conformar el diagrama de componentes Business de la figura 6.10.

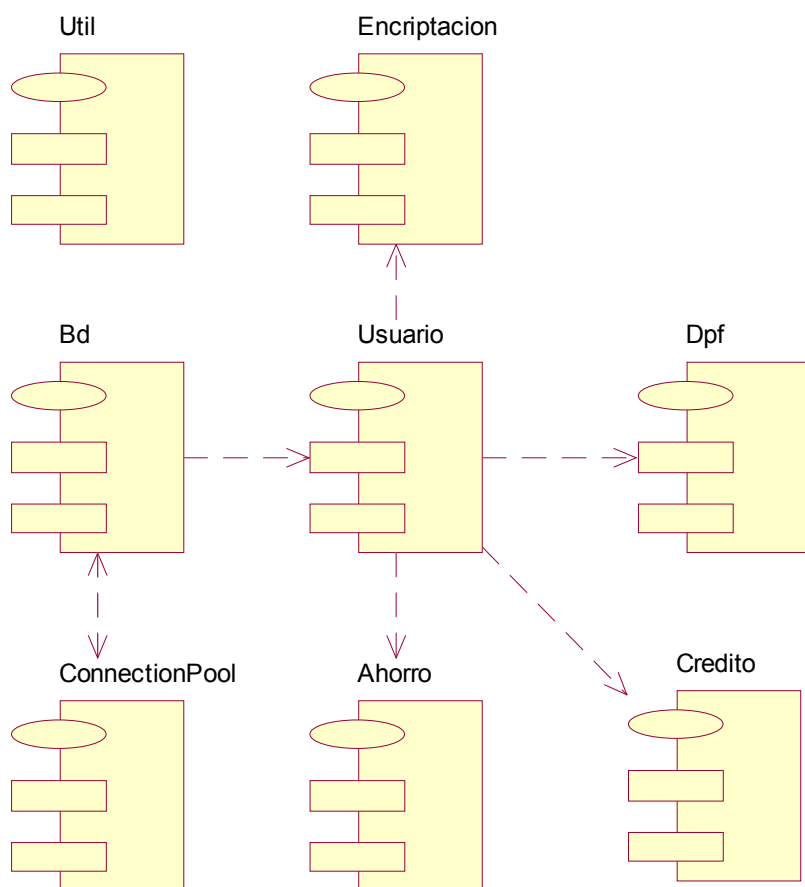


Fig. 6.10
Diagrama de componentes: business

7. Implementación

Después de concluida la etapa de diseño, comienza la fase de implementación que comprende toda la programación del sistema. Durante esta fase se presentaran las herramientas de software necesarias para la construcción del sistema conjuntamente con su arquitectura. También se mostraran las páginas o ventanas que lo conforman el sistema. Por último se describirán las pruebas que se han realizado.

7.1. HERRAMIENTAS DE SOFTWARE

Las herramientas utilizadas para la implementación del sistema son "tecnologías abiertas", con esto se quiere decir que estas herramientas están libres de costo, incluyen su código fuente y además pueden ser utilizadas a través de múltiples plataformas. Su desempeño y confiabilidad ha sido corroborado ampliamente al rededor del mundo.

Java (<http://java.sun.com>) .-

Se ha elegido Java como lenguaje de programación por las ventajas que este nos otorga en el desarrollo del sistema.

Java es un lenguaje que desde su aparición ha ido incorporando elementos destinados a proporcionar un mayor control sobre la seguridad. Una estricta verificación del código antes y durante su ejecución asegura que el código no trate de sobrepasar las protecciones impuestas por el lenguaje. Por ejemplo programas escritos en Java, no experimentan violaciones de memoria que permitan bajar a un servidor. El control es llevado a cabo de manera intrínseca por la maquina virtual java o JVM (Java Virtual Machine) [MAG].

Otra característica importante es su amplio conjunto de componentes y librerías que pueden incorporarse y ser utilizados dentro nuestro sistema en la forma de paquetes. Entre estas tenemos por ejemplo las librerías de criptografía JCE (Java Cryptography Extension)

imprescindibles en el sistema. JDBC (Java Data Base Connectivity) permite conectar nuestro sistema con la Base de Datos, Swing es un conjunto de componentes involucrados con la interfaz de usuario utilizado en el sistema de administración del sistema. Esta es ampliamente soportada por entornos de desarrollo o RAD (Rapid Application Development). JSP (Java Server Pages) y Servlets son la solución para la creación de contenido dinámico en aplicaciones web.

Con todo esto se puede decir que es posible crear una solución completa desde un solo lenguaje de programación.

Estos son algunos de los atributos clave que hacen de Java un entorno de programación superior.

Tomcat (<http://jakarta.apache.org>) .-

Tomcat es el entorno de ejecución o contenedor de la arquitectura Servlets y JSPs. El contenedor es ejecutado conjuntamente con la maquina virtual java, lo que implica que el contenedor este completamente escrito en Java.

Diseñado principalmente para generar contenido dinámico, puede procesar también cualquier tipo de documento estático, aunque se recomienda que esto lo haga mejor un servidor web como Apache, esto debido a performance y estabilidad ya que si cae la maquina virtual el servidor web seguirá ejecutándose lo que permite añadir una protección al sistema a nivel de procesos. Para unir el servidor web apache con Tomcat es necesario un plugin o conector (mod_jk) que maneje la comunicación entre estos dos módulos.

Tomcat proporciona otras facilidades en cuanto a seguridad y estabilidad pudiendo lanzarse varias instancias (redundancia) de este contenedor en una o diferentes maquinas lo que permite evitar un único punto de falla.

OpenSSL - Mod_ssl (<http://www.openssl.org> - <http://www.modssl.org>) .-

Ambos módulos trabajan conjuntamente para proporcionar la seguridad a nuestro sistema a través de SSL.

OpenSSL constituye en si el núcleo de SSL, mientras que Mod_ssl otorga la interfase entre Apache y OpenSSL. Estos módulos soportan los protocolos SSLv2, SSLv3 así como también el protocolo TLSv1 (Transport Layer Security). Además soporta llaves de 128 bits de longitud para cada sesión SSL lo que permite una mayor seguridad (a la convencional de 40 bits).

Otra característica importante es que autoridades certificadoras reconocidas como son Verisign o Thawte pueden emitir certificados a OpenSSL - Mod_ssl.

PostgreSQL (<http://www.postgresql.org>) .-

PostgreSQL es un poderoso manejador de base de datos relacional. Este repositorio de datos posee muchas características encontradas en otros DBMSs pero no posee ninguna restricción en cuanto a licencias y costos. Entre las más importantes que se pueden mencionar están el soporte de transacciones, subconsultas, claves externas (foreign keys), tipo de datos definidos por el usuario, funciones y procedimientos almacenados.

PostgreSQL puede manejar una gran cantidad de datos de manera confiable. Posee un sistema de verificación que permite controlar quienes y de donde se están conectando a la Base de Datos.

Estas características son las que hacen de PostgreSQL un poderoso DBMS.

Mysql (<http://www.mysql.com>).-

Es quizás una de las bases de datos más conocidas y utilizadas en el mundo de código abierto, gracias principalmente a su rapidez, objetivo con el que fue, fundamentalmente diseñado.

Mysql, posee también características importantes encontradas en otros DBMS, como ser el hecho de soportar transacciones, claves externas y poseer un conjunto amplio de funciones, aunque aun le faltan algunas otras características como ser, subconsultas y procedimientos almacenados, que están actualmente, siendo implementadas.

Algo que es importante señalar, es el hecho que Mysql es capaz de manejar una gran cantidad de datos de manera segura, además de contar con diferentes tipos de autenticación para la conexión a la base de datos.

Apache (<http://www.apache.org>).-

Es el servidor web más utilizado en el mundo¹⁶ debido en gran parte a su poder y flexibilidad. Apache trabaja conjuntamente con mod_ssl para la encriptación a través de SSL y a su vez con Tomcat (mod_jk) para la ejecución de páginas dinámicas. Antes que Apache procese una solicitud, mod_ssl encripta-desencripta los datos de y hacia el servidor. Luego se verifica si la solicitud demanda una página servidor (un servlet o JSP) y en caso de ser así Apache redirecciona la solicitud al servidor Tomcat el que lo procesa. De otro modo Apache directamente procesa la solicitud.

JMeter - JTest (<http://www.jakarta.apache.org/jmeter>).-

Ambas herramientas han sido utilizadas para realizar las pruebas en el sistema, la primera para medir el rendimiento de la aplicación y la segunda para realizar pruebas de unidad dentro el sistema.

Aunque existen muchas herramientas para realizar este tipo de pruebas muchas de ellas bastante caras, estas herramientas son bastante poderosas e incluyen muchas opciones útiles, como ser en el caso de JMeter la posibilidad de medir los tiempos de respuesta del servidor web, con y sin SSL, así como también el rendimiento de la base de datos a través del Connection Pool.

En el caso de JTest, esta herramienta puede realizar de manera automática pruebas de caja blanca, donde se examina interiormente lo que realiza cada componente y las de caja negra donde se examina la funcionalidad de cada componente sin saber lo que sucede interiormente.

7.2. ARQUITECTURA DEL SOFTWARE

La construcción del sistema esta constituido de dos partes o subsistemas mostrado en la figura 7.1 : el subsistema de consultas (parte izquierda) y el subsistema de administración (parte derecha).

En el sistema de consultas, el usuario se conecta a través de su navegador y el servidor web es el que se encarga de manejar esta solicitud. Todos los módulos involucrados se muestran en el nodo "Servidor Web". Luego la solicitud es manejada por la maquina virtual java donde se ejecuta en sí todo el sistema de consultas (nodo Java 1). Este trabaja conjuntamente con JCE para la seguridad y se conecta a PostgreSQL o Mysql a través de JDBC.

El subsistema de administración se ejecuta en otra maquina virtual java (nodo Java 2). Este subsistema involucra los módulos de Swing para la interfaz gráfica, JCE y JDBC.

¹⁶ Según información obtenida de <http://www.netcraft.com/survey>

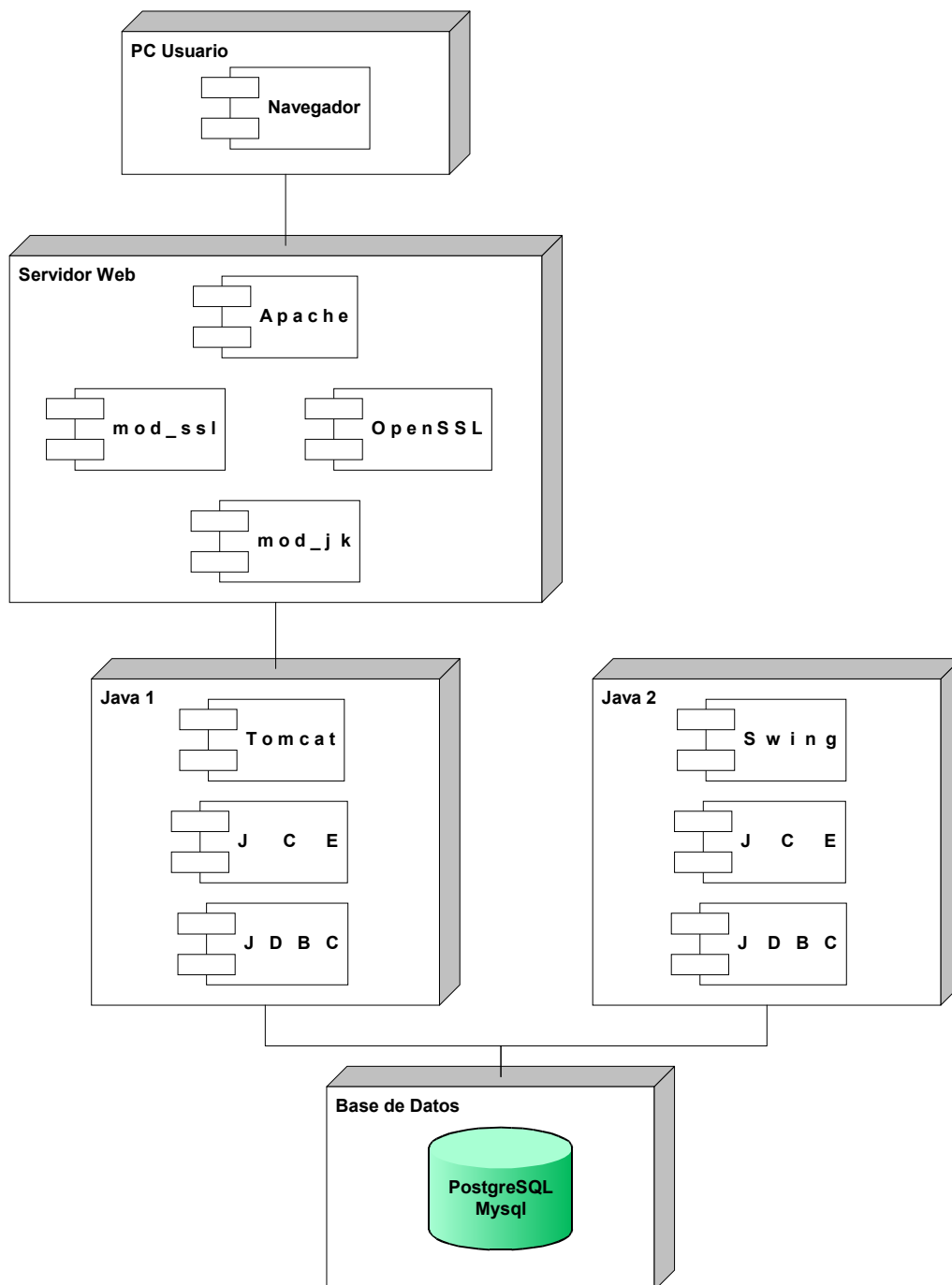


Fig. 7.1
Arquitectura del software

7.3. INTERFAZ

7.3.1. INTERFAZ DEL SISTEMA DE CONSULTAS

La primera página que se le presenta al usuario es la de ingreso al sistema, mostrado en la figura 7.2, que no es generado por ninguna pagina dinámica. En esta página existen dos campos para la introducción de su login y su password, esta ultima, como corresponde, no es mostrada al usuario.

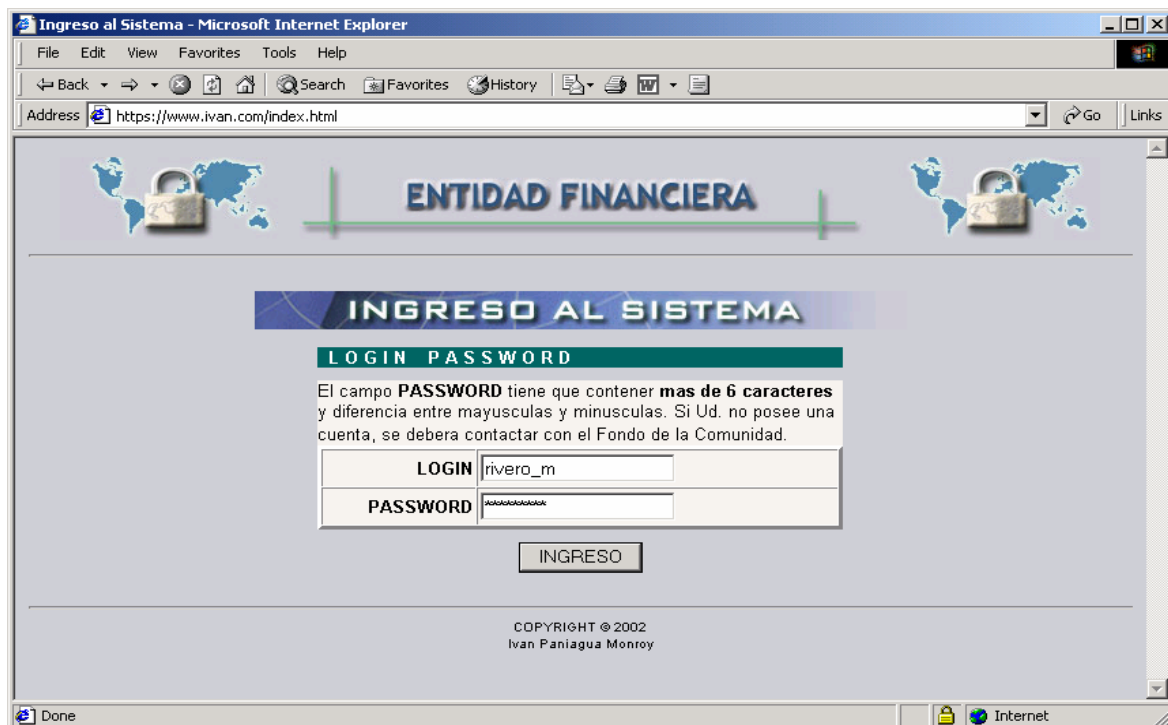


Fig. 7.2
Ingreso al sistema

Una vez ingresado al sistema se le presenta al usuario el menú de opciones como se muestra en la figura 7.3. Conjuntamente con las opciones: "listado de cuentas financieras", "cambio de contraseña", "contacto" y "salida" se despliega el nombre y la fecha de ingreso.

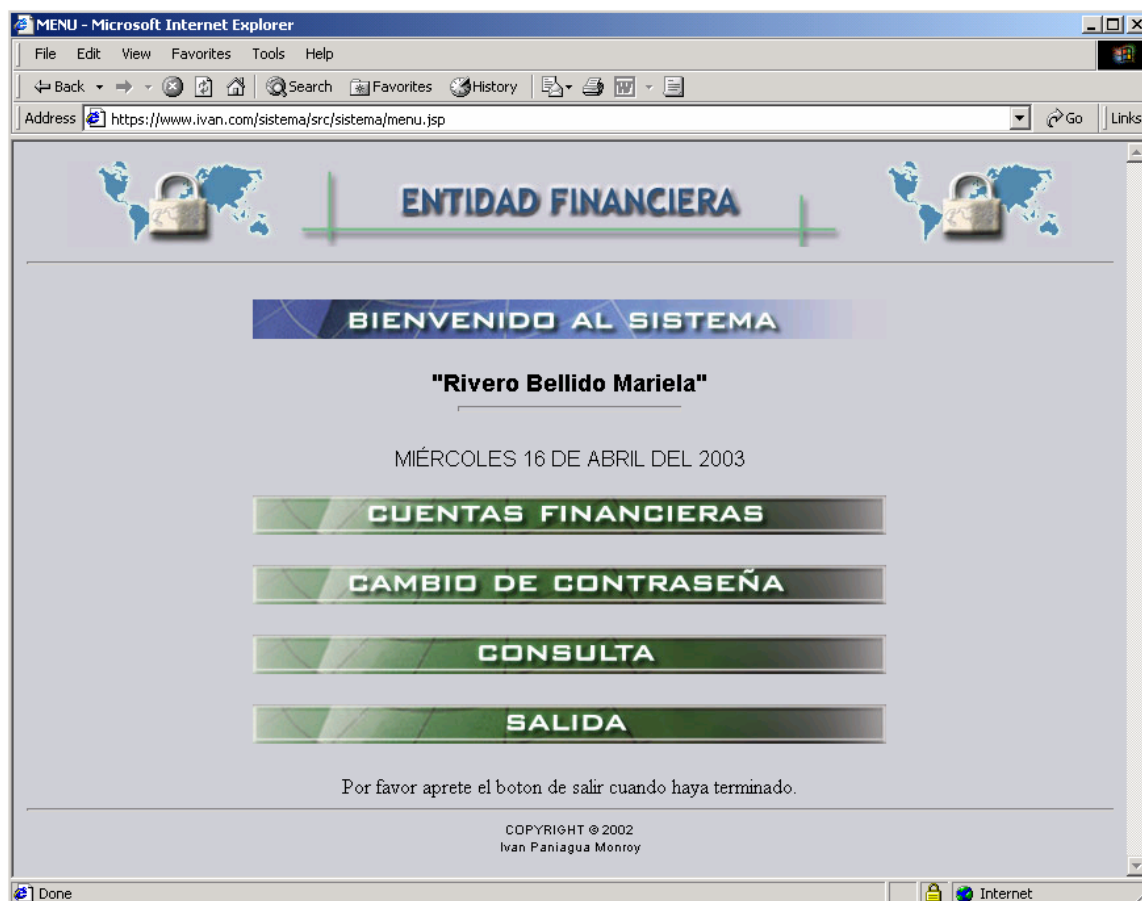


Fig. 7.3
Menú de opciones

Si el usuario selecciona el listado de "cuentas financieras" este obtiene la página "listar.jsp" mostrada en la figura 7.4. En esta se despliegan todas las cuentas que posee el usuario en dpf, crédito y caja de ahorro. Cada uno de estos tipos de cuentas están contenidas en un formulario ya que el usuario debe seleccionar una cuenta de dicha lista (Columna de "Sel" de Selección) y oprimir el botón de extracto de una determinado tipo de cuenta ya sea de dpf, credito o caja de ahorro, para así obtener su correspondiente extracto. Otra opción que

también se le da al usuario que posee cuentas de depósitos a plazo fijo o crédito es la de obtener el plan de pagos de dicha cuenta.

DEPOSITOS A PLAZO FIJO

Dpf	Apertura	Plazo	Vencimiento	Capital	Tasa	Periodo Intereses	Moneda	Total	Sel
1	11-Sep-96	360	06-Sep-97	793,00	10,50	Unico	\$us.	8.326,50	C
4	19-Sep-96	180	18-Mar-97	20.089,60	9,50	Mensual	\$us.	190.851,20	C
5	26-Sep-96	360	21-Sep-97	2.274,55	10,50	Unico	\$us.	23.882,78	C

Extracto de Dpf Plan de Pagos

CREDITOS

Credito	Monto	Tasa	Plazo	Tipo de Cuota	Tipo de Amortizacion	Dias de Amortizacion	Fecha de Inicio	Saldo	Sel
12	9.627,45	16,21	4830	C.FIJA	F.FIJA	30	12/09/1996	6.267,17	C
17	32.193,00	15,98	5190	C.FIJA	DIAS-F.	30	12/09/1996	29.807,50	C

Extracto de Credito Plan de Pagos

CAJA DE AHORROS

Caja de Ahorro	Moneda	Interes	Periodo Capitalizable	Saldo	Interes Pendiente	Fecha	Sel
67	\$us.	11,00	Mensual	14.000,00	11,00	15-Abr-98	C

Fig. 7.4
Lista de cuentas financieras

Por ejemplo si el usuario desea obtener el extracto de crédito de una determinada cuenta obtendría la siguiente página, mostrado en la figura 7.5.

EXTRACTO DE CREDITO - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites History




Address <https://www.ivan.com/sistema/src/sistema/credito.jsp> Go Links

ENTIDAD FINANCIERA

EXTRACTO DE CREDITO

Cuenta : "103"

Fecha	Operacion	Monto	Amortizacion	Interes	Otros Pagos	Monto Pagado	Saldo Capital
20/01/2001	Apertura de Prestamo	3.000,00					3.000,00
19/02/2001	Amortizacion		60,00	30,00	10,00	100,00	2.940,00
21/03/2001	Amortizacion		50,00	35,00	5,00	90,00	2.890,00
20/04/2001	Amortizacion		50,00	35,00	5,00	90,00	2.840,00
20/05/2001	Amortizacion		50,00	35,00	5,00	90,00	2.790,00
19/06/2001	Amortizacion		50,00	35,00	5,00	90,00	2.740,00

 INICIO
  LISTA
  CONTRASEÑA
  CONTACTO
  SALIR

Internet

Fig. 7.5
Extracto de cuenta

Cuando el usuario selecciona la opción de cambio de contraseña, el sistema redirecciona a una pagina estática que consta de tres campos, uno para la introducción de la contraseña con la que ingreso al sistema, la nueva contraseña y su confirmación como es mostrado en la figura 7.6.

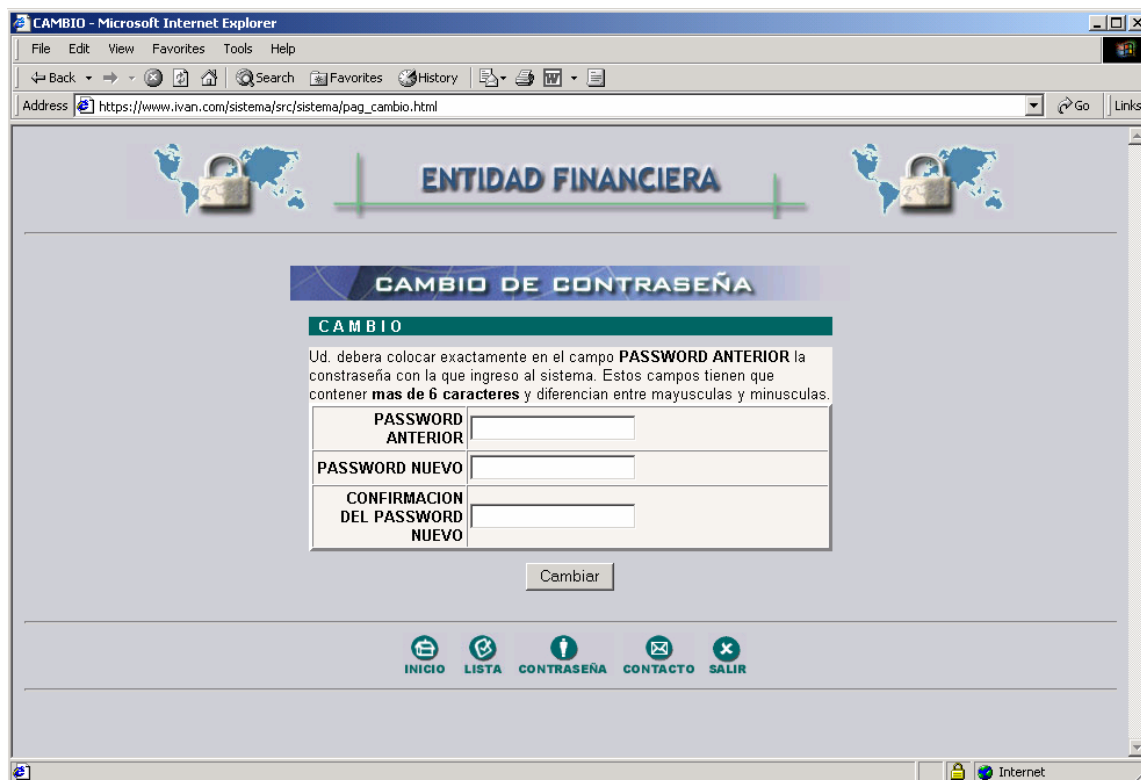


Fig. 7.6
Cambio de contraseña

7.3.2. INTERFAZ DEL SISTEMA DE ADMINISTRACIÓN

Una vez que el administrador se ha autenticado las opciones que tiene dentro del sistema están conformadas por: "Usuarios", "Cuentas", "Logs", "Configuracion" y "Salida".

Cuando el administrador selecciona la opción de "Usuarios" este obtiene otro subconjunto de opciones entre las que se tienen "Lista", "Buscar" y "Usuario".

En "Lista" se tienen todos los usuarios registrados conjuntamente con las diferentes opciones que se pueden realizar. Desde esta ventana, reflejada en la figura 7.7, el administrador puede modificar, eliminar, obtener el log y las cuentas financieras de un determinado usuario, previa selección de la lista desplegada.

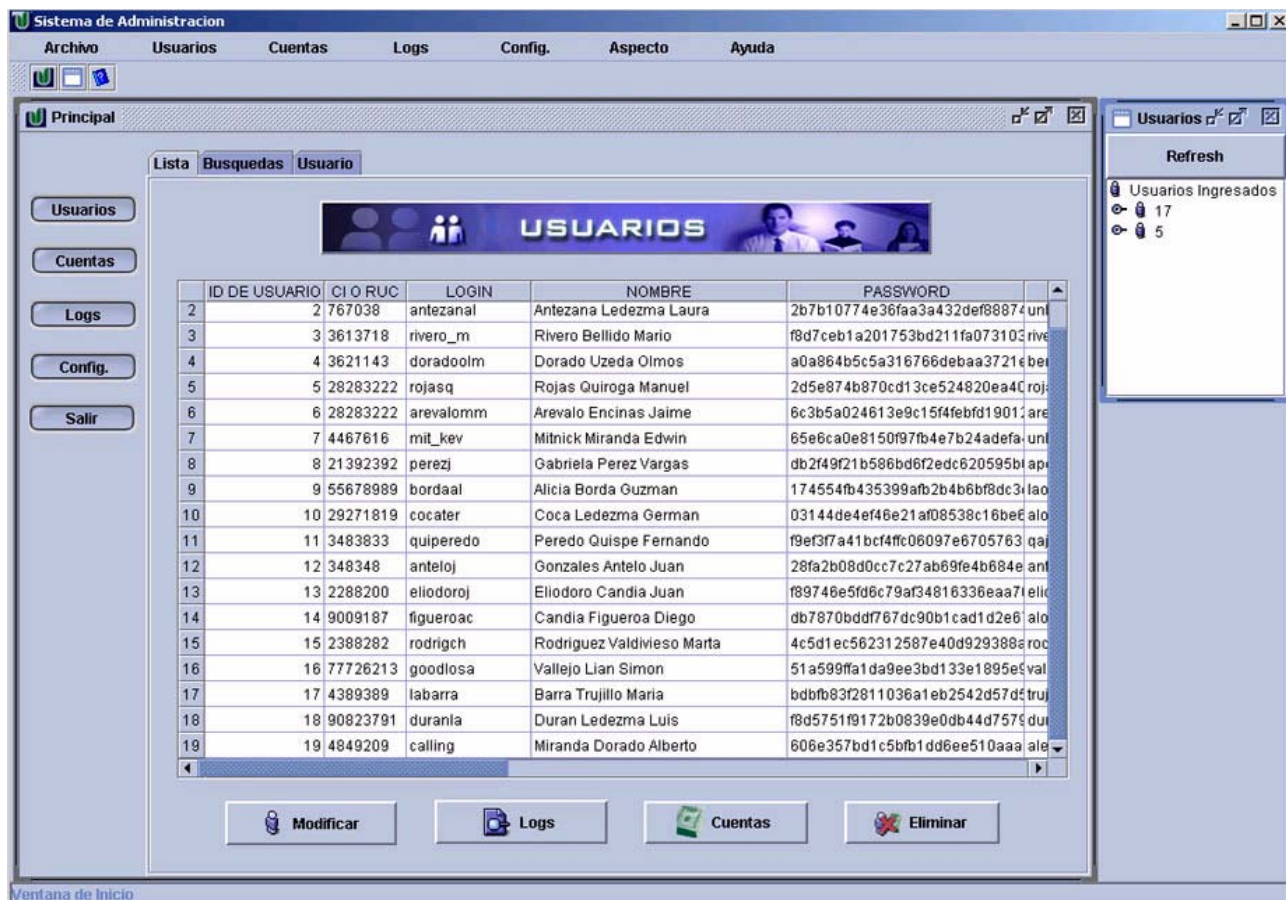


Fig. 7.7
Pantalla principal
del sistema de administración

La ventana de "Usuario" sirve a la vez para la creación y modificación de un usuario. El propósito de la opción "Nuevo" mostrada en esta ventana, de la figura 7.8, es el vaciar los datos de la misma para la creación de un nuevo usuario. Cualquier cambio o creación se vera reflejado en la Base de Datos una vez seleccionado el botón de "Guardar". También se cuenta con la opción de "Buscar", que verifica si el C.I. o R.U.C. introducido tiene asociada una cuenta financiera.

Sistema de Administración

Archivo Usuarios Cuentas Logs Config. Aspecto Ayuda

Principal

Lista Busquedas **Usuario**

Usuarios

Cuentas

Logs

Config.

Salir

USUARIO

ID 0

CI o RUC

LOGIN

PASSWORD

CONFIRMA PASSWORD

NOMBRE

E-MAIL 1

E-MAIL 2 (Opcional)

BLOQUEADO ☐ INTENTOS CONTINUOS

INTENTOS DISCONTINUOS

CREACION 2002-05-09 07:48:53 ▼

Creacion y Modificación de un Usuario

Fig. 7.8
Pantalla: usuario

La pantalla de "Cuentas" muestra las cuentas financieras de un determinado usuario, o todas las cuentas financieras registradas en el sistema en una tabla. En esta ventana se despliega en un campo el nombre del usuario que posee las cuentas mostradas. Se tienen también las opciones de búsqueda de una o varias cuentas y la de exportación de los datos desplegados por la tabla a un archivo externo.

Por último la opción de Configuración que permite al administrador cambiar su contraseña de acceso al sistema y también las opciones de "Mantenimiento de Logs", que comprende las opciones de "Vaciar Logs" que desplegara a su vez un dialogo que permite escoger el

archivo al cual vaciar, "Generar Nueva Llave" que permite cambiar la llave anterior por una nueva y la opción de "Arrojar Llave Actual" que despliega el dialogo donde se especificara el archivo al cual arrojarlo. En esta parte también se muestra información del último vaciado efectuado así como también el número de logs actual. La ventana de la figura 7.9 muestra estas opciones.

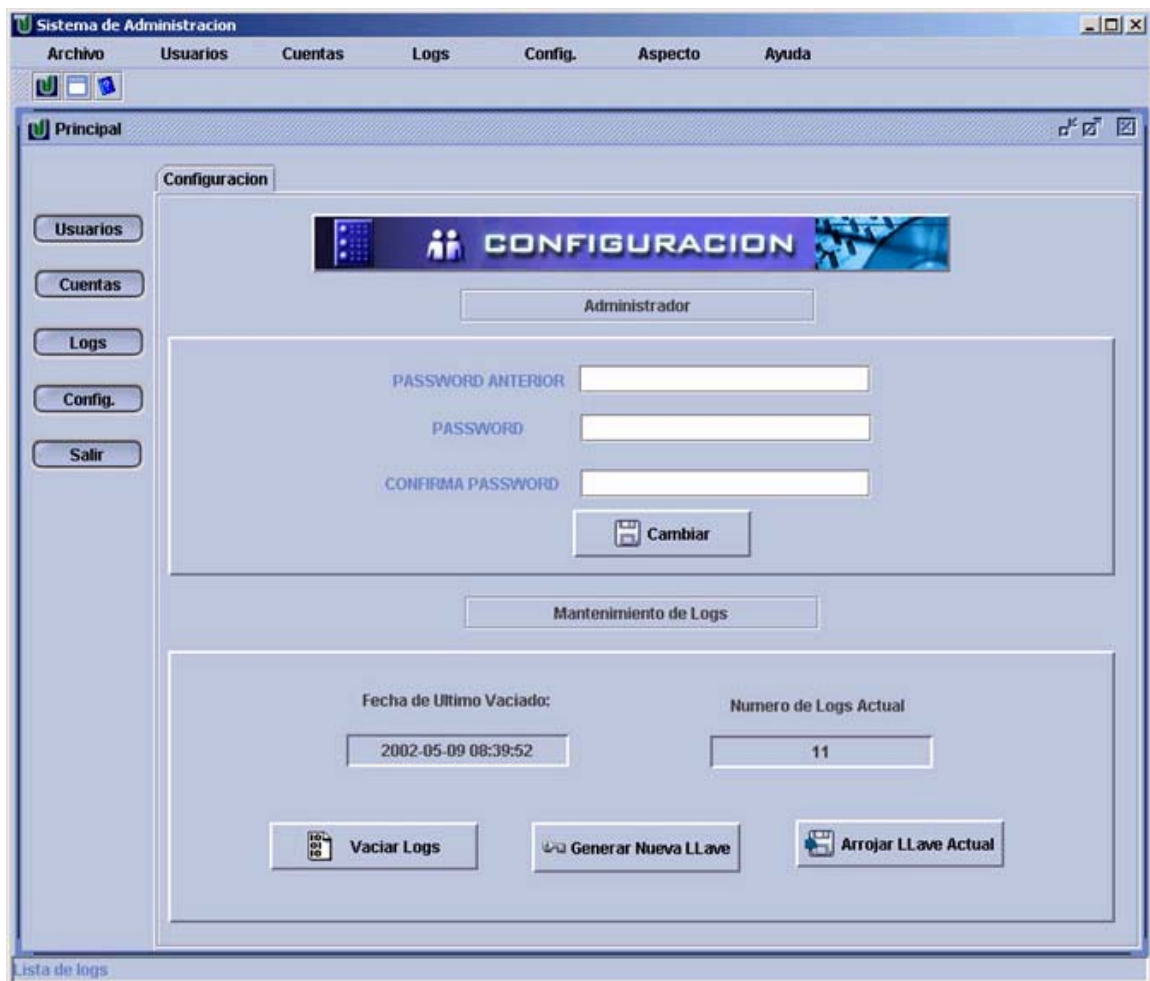


Fig. 7.9
Pantalla: configuración

7.4. PRUEBAS

El objetivo de las pruebas es el de verificar si el sistema cumple con los requerimientos planteados. Se realizaron diferentes pruebas para validar características importantes del sistema siguiendo (principalmente cuatro fases) diferentes tipos de pruebas.

Las especificaciones del equipo en el que se realizó estas pruebas, se encuentran detalladas en el Anexo 4.

Pruebas de Unidad.- Durante esta fase se verificaron los distintos componentes de manera aislada a medida que fueron implementados, asegurando que cada módulo cumpla de manera correcta con el propósito por el que fue diseñado. Es necesario señalar la utilización de la herramienta JTest, para la verificación de cada componente, no solo que este cumpla con aspectos estándares de diseño, sino además que el componente sea consistente y completo. Como resultado de estas pruebas se realizaron cambios en varias clases dentro la categoría business, mas específicamente en el manejo de valores nulos a métodos de estas clases.

Pruebas de Integración.- Como resultado de estas pruebas, los errores encontrados en la etapa de integración fueron en la base de datos. La relación de los usuarios con sus cuentas financieras no coincidía con la forma operacional que se trataba dentro de la institución en la que fue probada, se tuvo que añadir una tabla para cada tipo de cuenta financiera que relacionara un usuario con una cuenta financiera.

Para la integración del cliente con el servidor se han utilizado distintos de navegadores con soporte de cookies y de SSL, requisitos indispensables para la utilización del sistema. Se tuvo que crear un certificado de prueba con la herramienta OpenSSL, sin embargo este certificado debe ser obtenido de una autoridad certificadora como VERSIGN o TAWTHE. Para propósitos de este proyecto se ha creado un certificado propio, emulando ser una autoridad certificadora. La seguridad en la comunicación es asegurada con SSL sin

embargo los navegadores emitirán un mensaje de alerta que el certificado no esta reconocido por ningún organismo oficial, pero que nadie vera los datos transmitidos entre el cliente y el servidor (Fig 7.10).



Fig. 7.10

Alerta de seguridad

Se puede examinar el certificado, cuya información es mostrada en la figura 7.11:

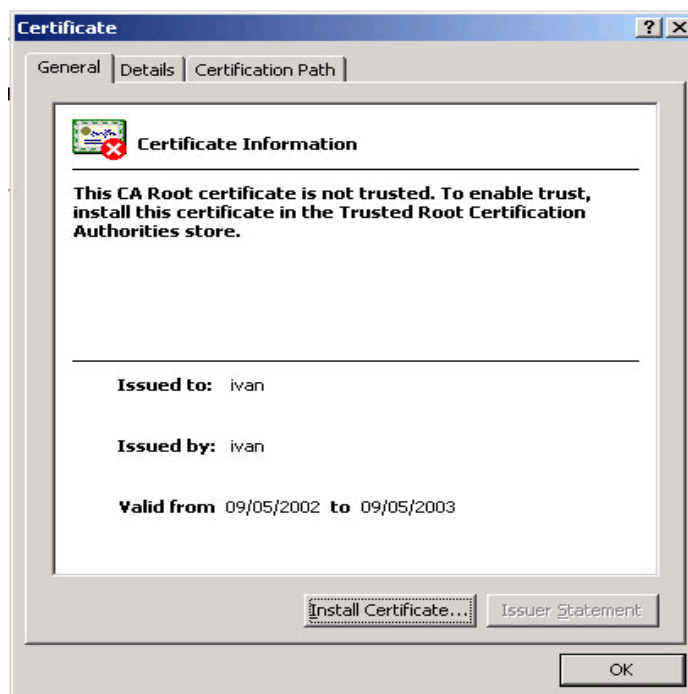


Fig. 7.11
Información del certificado

Puebas de sistema y aceptación.- Este tipo de prueba involucra a los usuarios potenciales del sistema. El sistema se presentó a personas tanto del área técnica como administrativa, que trabajan dentro del área de instituciones no bancarias, sector donde se concentra el presente trabajo.

Ellos examinaron toda la funcionalidad del sistema y reportaron grietas entre la funcionalidad entregada y los requerimientos originales dentro del sistema. También evaluaron aspectos como facilidad de uso, seguridad del sistema, así como también la velocidad de despliegue del sistema.

Como resultado de estas pruebas se tuvo que modificar algunos parámetros, en la duración de las sesiones, en el sistema, ya que se tenía un tiempo muy corto de permanencia en el sistema. También se modificaron y añadieron algunas páginas en las opciones del usuario.

Pruebas de desempeño.- El propósito de este tipo de prueba es asegurar que el sistema pueda manejar un volumen elevado de datos de entrada en un entorno controlado. Además que tenga el tiempo de respuesta adecuado. Esto involucra simular una gran cantidad de usuarios para determinar la escalabilidad y descubrir cuellos de botella.

Uno de los factores críticos en la aplicación es la conexión a la base de datos y gracias a la aplicación del Connection Pool, mejora notoriamente el manejo de un gran volumen de solicitudes.

El gráfico 7.12, muestra la diferencia que existe entre la utilización del Connection Pool y sin ella. La medición está basada en la cantidad de respuestas que puede manejar en un segundo.

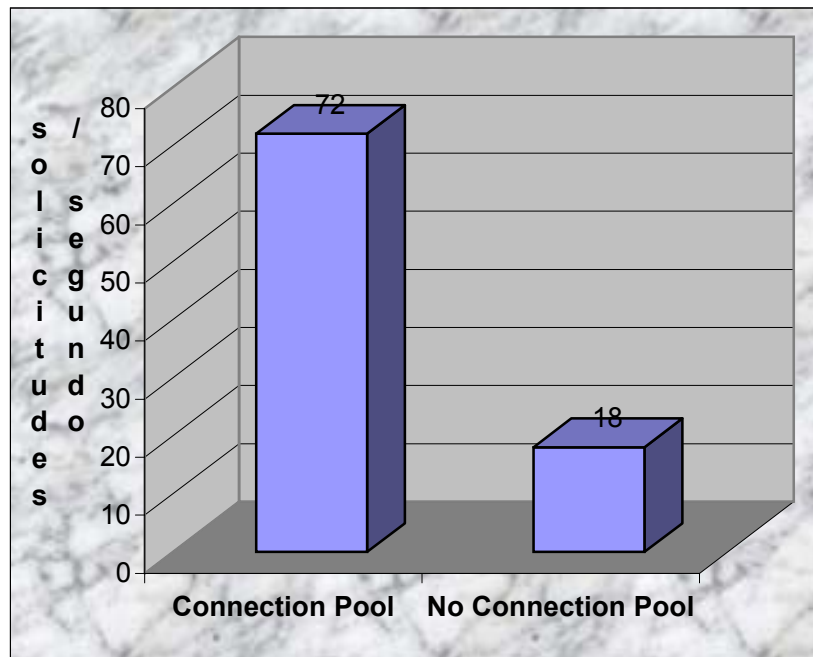


Fig. 7.12
Comparación de la aplicación
de Connection Pool

Otro aspecto muy importante a medir en el sistema es la aplicación de SSL, ya que se realizan muchas operaciones criptográficas entre el cliente y el servidor, lo que hace decrecer enormemente el tiempo de respuesta.

Para ver una comparación de respuesta de la primera página al ingresar al sistema, a través de SSL y sin ella, se puede observar en el gráfico 7.13, la diferencia que existe entre ambos.

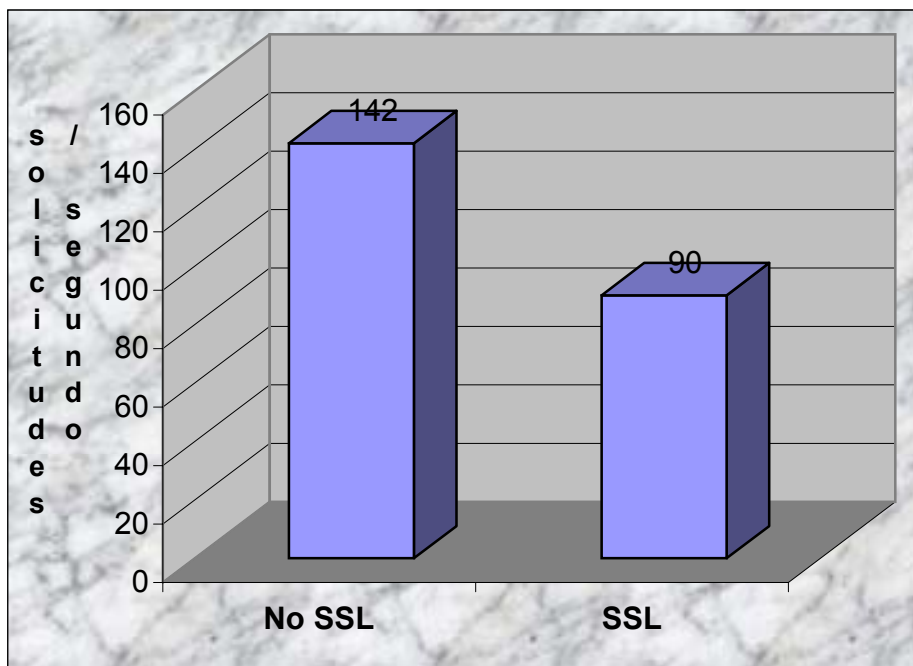


Fig. 7.13
Comparación de la aplicación de
SSL

8. Conclusión y extensión futura

Luego de haber completado el desarrollo del sistema propuesto, en este capítulo se presenta las conclusiones finales de este trabajo así como también algunas recomendaciones y extensiones futuras que pueden ser llevadas a cabo.

8.1 CONCLUSION

Teniendo como objetivo principal el de, desarrollar un sistema de consultas financieras, se ha analizado, diseñado e implementado el sistema, separándolo en dos sistemas o subsistemas. Uno que permita a los usuarios realizar sus consultas a través de Internet, de sus cuentas tanto de crédito, caja de ahorro y depósito a plazo fijo. El segundo, el subsistema de administración que permite, al administrador del sistema un manejo y control adecuado de los usuarios y sus cuentas. Al mismo tiempo que realiza un monitoreo y registro de las acciones que realizan los usuarios dentro del sistema.

En general el aspecto más importante que se maneja a lo largo de esta tesis es la seguridad, de esta, en particular los algoritmos criptográficos son la herramienta fundamental que ha sido estudiada y aplicada para preservar la confidencialidad de la información. Haciendo uso de herramientas estándares para la protección de la información entre el cliente y el servidor así como también en la protección de los datos en el servidor.

Sin embargo la seguridad va más allá de la aplicación de la criptografía y se ha tomado en cuenta también un buen diseño que contemple la autenticación de los usuarios, la verificación de los datos enviados así como también el control y monitoreando las acciones que realiza dentro del sistema. Aquí, la incorporación de la notación UML para el modelamiento de todo el proceso de desarrollo, juega un papel sumamente importante, ya que se lo ha extendido para representar aspectos importantes dentro del sistema, como son las páginas web y la base de datos.

Se han utilizado “tecnologías abiertas”, libres de costo lo que permite un ahorro significativo para la empresa, aspecto muy importante en nuestro medio, donde pequeñas instituciones no pueden realizar grandes inversiones, pero sobre todo al incluir su código fuente su desempeño y confiabilidad ha sido verificado ampliamente.

Como lo reflejan los datos arrojados en la etapa de pruebas, gracias a la aplicación de técnicas aplicadas tanto a la base de datos como a las sesiones se puede mejorar el rendimiento no solo del sistema, sino de cualquier aplicación web.

8.2 RECOMENDACIONES Y EXTENSION FUTURA

Son varias las recomendaciones y extensiones que surgen de la realización de este trabajo y en esta sección se presenta las más importantes y que deberían ser tomadas en cuenta en futuros trabajos.

Un aspecto muy importante al momento de trabajar con aplicaciones web, es el entorno donde se ejecuta el sistema, que contempla el sistema operativo, los servicios que otorga, los puertos activos. Se deben tener la menor cantidad de servicios y puertos activos, ya que abren más vulnerabilidades al sistema. También se deben tener en cuenta una adecuada autorización de permisos, restringiendo el acceso a los archivos del sistema. Utilizar un “firewall”¹⁷ o cortafuegos, que limite el acceso externo a los servicios o puertos que son necesarios y no se los puede quitar. Revisar constantemente las nuevas vulnerabilidades que surgen de los sistemas involucrados en el trabajo, realizando continuamente las actualizaciones necesarias.

Este proyecto de tesis no se queda aquí y pretende crecer hasta convertirse en un sistema de transacciones financieras donde le permita al cliente de la institución realizar traspasos entre cuentas, por lo que se deberá mejorar la seguridad del sistema contemplando por ejemplo una doble autenticación por un lado, así como también en cuanto a caídas de

¹⁷ Generalmente es un aparato (hardware), aunque también existen implementaciones en software. Para una introducción completa en el tema ver [ZW100]

servidores, reduciendo puntos únicos de falla (single point of failure) u otros aspectos que permitan un correcto funcionamiento del sistema.

Finalmente es necesario decir que con todas las herramientas que podamos tener, con los últimos algoritmos que existan, con todas las técnicas de protección que se puedan emplear, la seguridad es un tema que no tiene una solución definitiva y comprende también un conjunto de leyes que deben acompañarla así como un buen uso y conocimiento por parte del usuario.

Bibliografía

1. Bibliografía con soporte escrito:

[BOO94] Booch Grady. 1994, *Análisis y Diseño Orientado a Objetos con aplicaciones*, Estados Unidos, 2ª edición, Addison-Wesley.

[BRI98] Brian Francis, Alex Homer Et.Al. 1998, *Profesional Active Server Page 2.0*, Estados Unidos, 2ª edición, Wrox Press Inc.

[CON99] Conallen Jim, 1999, *Building Web Applications with Unified Modeling Language*, Estados Unidos, 1ª edición, Addison- Wesley.

[DOR99] Dorsey Paul, Hudicka R. Joseph, 1999, “*Oracle8 Diseño de bases de datos con UML*”, España, 1ª edición, McGraw-Hill.

[GAR01] Garms Jess, Somerfield Daniel, 2001, *Profesional Java Security*, Estados Unidos, 1ª edición, Wrox Press Inc.

[KNU98] Knudsen B. Jonathan, 1998, *Java Cryptography*, Estados Unidos, 1ª edición, O'Reilly.

[QUA98] Quatrani Terry, 1998, *Visual Modeling with Rational Rose and Unified Modeling Language*, Estados Unidos, 1ª edición, Addison-Wesley.

[HAL98] Hall Marty, 1998, *Core Servlets and Java Server Pages*, Estados Unidos, 1ª edición, Prentice Hall and Sun Microsystem.

[ZWI00] Zwicky D. Elizabeth, Cooper Simon, & D. Brent Chapman, 2000, *Building Internet Firewalls*, 2ª edición, O'Reilly.

2. Referencia: Internet

[RAZ] Razvan Peteanu. “*Best Practices for Secure Web Development*”.

<http://www.dat.etsit.upm.es/~mmonjas/pago/index.html>

[STE] Stein y Stewart. “*WWW Security Frequently Asked Questions*”.

<http://www.w3.org/Security/Faq/>

[PAL] Pallos S. Michael. “*Designing WepSphere Application Server for Better Performance using Best Practices*”.

http://www.websphere-users.ca/presentations/03202003/Candle_WAS_Best_Practices.pdf

[MON] Monjas Llorte Miguel Angel. “*Uso de criptografía en mecanismos de pago, Departamento de Ingeniería de Sistemas Telemáticos*”, Universidad Politécnica de Madrid.

<http://www.dat.etsit.upm.es/~mmonjas/pago/index.html>

[KVA] Kvartin Hana. “*Modern Client/Server Architecture – Industrial Aspects*”, LIAM Software Systems LTD.

<http://www.cs.huji.ac.il/course/oodist/classes/lecture2/sld001.htm>

[COS] Costa Romero Manuel. “*Localización de Datos para Aplicaciones Distribuidas*”, Universidad Politécnica de Cataluña.

<http://www.lsi.upc.es/~cquer/esd210/locDades/loc.html>

[GON] Breogan Gonda y Juan Nicolas Jodal. “*Revising the Client/Server Architecture: The Present and Trends*”.

<http://www.genexus.com>

[DUA] Duane K. Fields.”*Communicaction for Enterprise Applications (Architectural Options)*”.

http://developer.netscape.com/viewsource/fields_servlet/fields_servlet.html

[ENG] Engelschall, Ralf S.. “*User Manual Mod_ssl 2.7*”.

<http://www.modssl.org/>

[LLO] Lloyd Taylor “*Cliente/Server Frequently Asked Questions*”.

<http://www.faqs.org/faqs/client-server-faq/preamble.html>

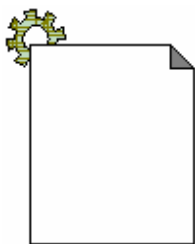
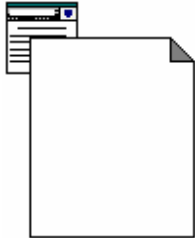
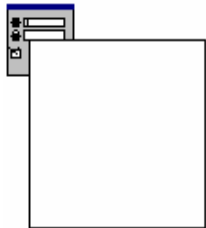
[ROP] Ropero Antonio. “*Vulnerabilidades en multiples sistemas shopping cart – Hispasec*”. <http://www.hispasec.com/unaaldia.asp?id=460>

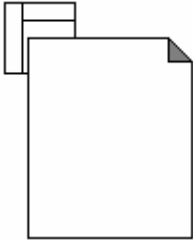
[MAG] MageLang Institute. “*Fundamentals of Java Security*”.

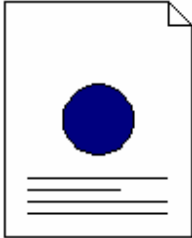
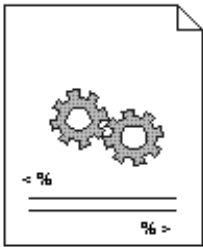
<http://developer.java.sun.com/developer/onlineTraining/Security/Fundamentals/Security.html>

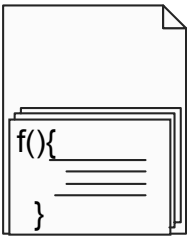
Anexos

ANEXO 1 – ESTEREOTIPOS

ESTEREOTIPOS DE CLASES	
<p><<server page>></p> 	<p>Representa una página que contiene código que se ejecuta en el servidor. Las funciones de la página representan los métodos de la clase y las variables accesibles por todas las funciones de la página representan los atributos de la clase. Estas páginas están relacionadas con recursos en el lado del servidor como lógica de negocios y bases de datos.</p>
<p><<client page>></p> 	<p>Representa una página HTML, que se “ejecuta” o se carga en el navegador. Los métodos y atributos de esta clase corresponden a las funciones y variables implementados con JavaScript o VB Script, las cuales son llevadas a cabo por el navegador.</p> <p>Estas páginas representan la parte de Interfaz de Usuario y tienen asociaciones con otras páginas cliente o servidor.</p>
<p><<form>></p> 	<p>Es un conjunto de campos de entrada (input fields) que forman parte de una página cliente.</p> <p>Sus atributos corresponden a campos de entrada del formulario como ser <<input boxes>>, <<text areas>>, <<radio buttons>>, <<check boxes>>, etc. Un</p>

	<p>formulario no tiene operaciones ya que una operación no puede ser contenida en un formulario. Cualquier operación que interactúa con el formulario puede ser propiedad de la página que contiene el formulario.</p>
<p><<frame set>></p> 	<p>Un frame set es un contenedor de varias páginas web. Nos permite presentar múltiples páginas cliente al mismo tiempo. Estas páginas concurrentes están relacionadas para representar una única interfaz de usuario. El área rectangular de vista de la página es dividida en pequeños frames rectangulares. El contenido de un frame puede ser una página web u otro frame set. Debido a que un frame set es una <<página cliente>> este también puede tener operaciones y atributos.</p>
<p><<scriptlet>></p>	<p>Un scriptlet es una página cliente pero almacenada temporalmente en el cliente (cache). Esta generalmente contiene controles que son reusados por subsecuentes páginas clientes. Su ventaja es la reutilización de páginas cliente lo que disminuye el ancho de banda. Sin embargo esta propiedad solo está presente exclusivamente en navegadores Internet Explorer 4.0 o posterior.</p>
ESTEREOTIPOS DE ASOCIACIONES	
<p><<builds>></p>	<p>Las páginas servidor <<construyen>> páginas cliente. Esta asociación es siempre</p>

	direccional de una página servidor a una página cliente
<<link>>	Links son enlaces de una página cliente a otra página cliente o servidor.
<<submit>>	Un formulario envía (<<submit>>) todos los datos que contiene a una página servidor la cual procesa los datos.
<<redirect>>	Una página servidor puede redireccionar el procesamiento a otra página web.
<<frame content>>	Una asociación de este tipo es una asociación de agregación que expresa que un frame contiene otra página u otro frame set indicando frames anidados.
ESTEREOTIPOS DE COMPONENTES	
<<HTML>> 	Un componente cliente es una página HTML. Este puede ser demandada por su nombre por el navegador. Este componente esta relacionado con un componente en la vista de componentes en el modelo del sistema. Además puede contener scripts cliente como JavaScript.
<<componente servidor>> 	Este componente representa a una página servidor así como también a las páginas cliente que este genera debido a que este componente servidor representa una vista física del sistema. Es implementado generalmente con tecnologías como ASP o JSP.
<<componente scripts>>	Este componente representa a un archivo que contiene scripts que sirven como complemento a una página cliente. Este

	<p>generalmente esta implementado en código JavaScript y su extensión es jc.</p>
---	--

ANEXO 2 - ESPECIFICACIÓN DE CLASES (DICCIONARIO DE DATOS)

wnd_autenticacion Ventana inicial para ingresar al subsistema de administración, donde se pide que se introduzca el login y contraseña para utilizarlo.

Private Attributes:

id : String El id del Administrador

password : String La contraseña del Administrador

Bd : Bd El objeto que se encarga de la conexión y el manejo de todas las operaciones de la base de datos.

Public Operations:

show () : Al tratarse de un objeto ventana, se tiene este método que sirve para desplegar en pantalla esta ventana.

Wnd_opciones Ventana principal, que muestra las diferentes opciones que se tienen, como por ejemplo añadir un nuevo cliente, listar logs de usuarios, etc.

Private Attributes:

estado : Integer Variable que se utiliza para almacenar el estado u opción en el que se encuentra el administrador. Por ejemplo en estado=0 significa que nos encontramos en las opciones del usuario, en estado=1 en las cuentas del usuario, en estado=2 en sus logs y por ultimo en estado= 3 en la opción configuración.

W_usuario : Wnd_usuario Objeto de la clase Wnd_usuario que maneja parte de las opciones del usuario.

W_lista : Wnd_lista Objeto de la clase Wnd_lista que maneja el listado de todos los usuarios en el sistemas.

W_cuentas : Wnd_cuentas Objeto de la clase Wnd_cuentas que representa a las cuentas de usuario en particular tanto de deposito a plazo fijo, caja de ahorro y de crédito, además al mismo tiempo estos objetos muestran todas las cuentas de todos los usuarios en la entidad.

W_logs : Wnd_logs Objeto de la clase Wnd_logs que maneja el listado de todas las operaciones realizadas en el sistema.

W_conf : Wnd_conf Objeto de la clase Wnd_conf que maneja las diferentes opciones administrativas como ser el vaciado de logs, el cambio de contraseña.

Public Operations:

usuarios(Sel : Integer) : Método que despliega las opciones de los usuarios, pero como las opciones de los usuarios comprende otras subopciones como Listar,

Búsqueda y Usuarios el parámetro Sel de este método, selecciona e indica a cual de las opciones de usuario se esta refiriendo.

Cuentas(Sel :Integer) : Método que despliega todas cuentas registradas en la entidad, pero como las cuentas pueden ser en dpf, caja de ahorro o crédito, el parámetro pasado en esta función, señala de que tipo de cuenta se quiere obtener sus opciones.

Logs(): Método que muestra el objeto w_logs, que contiene todas las operaciones realizadas.

Configuración(): Método que muestra el objeto w_config, que contiene todas las opciones de configuración del sistema.

Salir(): Método que termina con el sistema de administración, desplegando un dialogo de confirmación de salida, luego se cierra la base de datos.

Wnd_usuario Ventana que sirve para manipular los datos de un usuario. Esta es utilizada indistintamente para modificar los datos de u usuario ya existente o también para la creación de uno nuevo.

Private Attributes:

usuario : *Usuario* El objeto que guarda los datos del usuario temporalmente.

Bd : *Bd* El objeto que se encarga de la conexión y el manejo de todas las operaciones de la base de datos.

Public Operations:

nuevo () : Se crea un nuevo objeto usuario con datos en blanco y se lo pasa al método setusuario().

Get_usuario () : *usuario* Método que devuelve el objeto usuario que se despliega en la ventana.

Set_usuario(usuario:Usuario): Método que recibe como parámetro un usuario, y establece sus valores en esta ventana

guardar() : Método que sirve para guardar los cambios realizados en un usuario, o en la creación de un usuario nuevo.

Buscar(): Método utilizado para la búsqueda de cuentas financieras en la entidad, de acuerdo al número de CI o RUC del nuevo cliente que se esta creando.

Wnd_lista Representa la ventana que despliega la lista completa de usuarios del sistema en una tabla. Desde la misma se pueden realizar operaciones directamente como la eliminación de un determinado usuario o a través de otros objetos, la modificación o la búsqueda de cuentas financieras de un usuario cualquiera.

Private Attributes:

bd : *Bd* El objeto que se encarga de la conexión y el manejo de todas las operaciones de la base de datos.

Public Operations:

modificar() Este método primeramente obtiene de la tabla, el identificador de que usuario se desea alterar sus datos y con este valor se obtiene a través del atributo bd, el usuario con todos sus datos. Por ultimo este objeto “usuario”, creado se lo pasa al objeto wnd_usuario con el método set_usuario.

Logs() : Este método es llamado cuando el administrador desea obtener las acciones que ha realizado algún usuario. Al igual que el anterior método se obtiene el id del usuario, de la tabla mostrada y este valor se pasa al objeto “wnd_log” a través del

método “set_id”, donde a su vez llama al método “Buscar_log” del objeto “bd”, que obtiene todos los datos de logs requeridos.

Eliminar(): Este método, obtiene el “id” del usuario en la tabla mostrada y luego antes de mandar al método “eliminar” del objeto “bd” muestra una ventana de confirmación de la acción a realizar.

Cuentas(): Este método, obtiene todas las cuentas de un determinado usuario, a través del metodo “Buscar_cuenta” del objeto “bd”.

Wnd_logs Sirve para desplegar la lista de todas las operaciones que realizo un determinado usuario así como también todas las operaciones realizadas por todos los usuarios.

Private Attributes:

id : usuario El id del usuario del cual se obtiene sus operaciones

Bd : Bd El objeto que se encarga de la conexión y el manejo de todas las operaciones de la base de datos

Public Operations:

buscar() : En este objeto, se tiene la posibilidad de buscar alguna operación realizada por cualquier usuario, bajo diferentes criterios, como por ejemplo la fecha de operación, el tipo de operación, etc.

Todo() : Método que muestra todas las operaciones realizadas en el sistema.

Export(): Método que sirve para la exportación de los datos mostrado en la tabla hacia un archivo plano.

Wnd_conf Clase que sirve al administrador para la configuración del sistema. Desde aquí puede vaciar todos los logs realizados, ver cuando ha sido el último vaciado o generar una nueva llave de encriptación.

Private Attributes:

llave :Byte[] Representa la llave que se tiene para la encriptación de datos.

Textoplano :Byte[] Representa todos los datos que van a ser ecnriptados y en este caso todos los logs pero representados como bytes, ya que de esta forma es mas fácil encriptarlos.

Textoencrip :Byte[] Representa los datos ya encriptados de todos los logs del sistema.

Public Operations:

vaciar(): Método que realiza la eliminación de los logs del sistema a través del método “borrar_logs” del objeto “bd”, al mismo tiempo este método devuelve en un array de bytes, con todos los logs que han sido eliminados de la base de datos. Con estos datos en el array se procede a encriptarlos a través del método “encripta” del objeto encriptación. Por último se muestra una ventana de dialogo que sirve para elegir donde arrojar los datos encriptados. La escritura de archivos se lo realiza a través del método “escribirArchivo” del objeto “util”.

Arrojar(): ☐ enú ☐ n Método que sirve para guardar la llave actual que se maneja, en un archivo.

Generar() Se utiliza este método para generar una nueva llave, pero antes se debe botar en un archivo la llave que se tenia antes.

Sesion Clase que maneja las sesiones de usuario.

Public Operations:

setAttribute(id) Introduce el id en la sesión

getAttribute():Object Obtiene el valor almacenado en la sesión

autentifica Clase que controla el acceso al sistema, validando los datos introducidos por un usuario

Private Attributes:

id : Integer El id del usuario que intenta ingresar al sistema de transacciones

password : String La contraseña del usuario que intenta ingresar al sistema

usuario : usuario Representa al objeto usuario que se obtiene a partir del id introducido

Bd : Bd El objeto que se encarga de la conexión y el manejo de todas las operaciones de la base de datos.

Public Operations:

doGet () Procesa una petición “Get” y esta redirecciona esta petición al método Post.

doPost() Procesa una petición “Post”. En este método se recupera el login, password y otros datos importantes donde se los valida y autentifica. También en este método se mandan los Headers o Cabeceras de control de la cache así como también se crea la sesión a ser utilizada por las siguientes clases. Por último se redirecciona a la página “menú”

Menú Despliega las diferentes opciones que tiene el usuario autenticado, para lo cual antes debe verificar que no exista ningún error y se ha autenticado al usuario correctamente.

Private Attributes:

sesion:Session representa la sesión en la que se recuperara los datos almacenados anteriormente en el objeto autentifica

id : Integer El id del usuario que se ha recuperado de la sesión.

Usuario : usuario Representa al objeto usuario que se creara a partir del id obtenido de la sesión.

Listar Lista todas las cuentas financieras que posee un determinado cliente, para lo cual antes se debe verificar la autenticación del usuario y solo después de esto se generara la página “pag_lista”

Private Attributes:

usuario : Usuario Representa al objeto cliente utilizado para obtener los datos y así generar la lista de cuentas

cuentas: Cuentas[] El arreglo o vector contiene todas los objetos de tipo “credito”, “dpf” y “ahorro”

plan Maneja información detallada de una cuenta específica tanto de crédito, dpf, o caja de ahorro. Esta contiene información como fecha de creación de la cuenta, monto, etc.

Private Attributes:

id_cuenta_f : Integer El id de la cuenta financiera que se quiere consultar

Public Operations:

escribir_consulta() : Genera la pagina con el detalle de una cuenta específica.

Extracto Genera el extracto de una cuenta, ya sea de crédito o de caja de ahorro.

Private Attributes:

cliente : *cliente*

id_cuenta_f : *Integer* El id de la cuenta financiera que se quiere consultar

Public Operations:

escribir_extracto () : Genera la lista de movimientos que se han efectuado en una cuenta específica. En esta se detalla información como la fecha, tipo de movimiento, cantidad, total.

Cambio Pagina servidor que permite validar y cambiar la contraseña tanto de acceso como la contraseña de transacción.

Private Attributes:

usuario : *usuario*

password : *String* La contraseña anterior

nuevo : *String* La nueva contraseña

nuevo_rep : *String* La confirmación de la nueva contraseña

Public Operations:

escribe_resultado () : Devuelve el resultado del cambio en la pagina que genera esta clase.

Salir Pagina desde la cual se eliminan la sesión creada al ingresar al sistema

Private Attributes:

usuario : *usuario*

usuario, t_usuario Clase que representa a un usuario en general, en esta se encuentran atributos y operaciones comunes a los usuario consulta, administrador y cliente.

Private Attributes:

password : *String* La contraseña de acceso al sistema

id : *Integer* El id de un usuario.

Nombre : *String* El nombre del usuario

bloqueo : *Boolean* Indica si una cuenta de usuario esta bloqueada o no.

Intentos_contin : *Integer* = 3 Representa el número de intentos continuos que puede tratar un usuario. Este tiene un valor inicial de 3.

Intentos_discon : *Integer* = 7 Representa el número de intentos discontinuos que puede tratar un usuario. Este tiene un valor inicial de 7.

Errores : *String []* Contiene todos los errores que se han efectuado al manejar esta clase.

Email1, Email2 Son los emails, el primero obligatorio y el segundo opcional.

Creación: *TimeStamp* representa la fecha de creación del usuario.

Public Operations:

cambiar (password, nuevo, ip) : Cambia la contraseña de acceso, dentro de este método también se almacena en el registro de operaciones el ip del usuario, que es parámetro ip de este método.

Contraseña (password :) : *boolean* Verifica que no contenga caracteres repetidos, que no sea secuencial. También compara la contraseña que se tiene en el objeto usuario con la que se pasa como argumento.

Get_Logs () : *String []* Devuelve todas las operaciones que ha realizado un usuario, en un arreglo de objetos registro_de_operacion.

Errores () : String[] Devuelve todos los errores encontrados en este usuario.
Autentifica(password, datos, ip) Método que es utilizado para validar un usuario cuya contraseña de acceso es pasada como argumento, junto con algunos datos importantes como ser el ip desde donde se esta queriendo autenticar así como también otros datos como ser el nombre del host y el navegador que utiliza.
Password_digest () Este método encripta la contraseña de acceso.
Getcuentas_() : Devuelve las cuentas tanto en dpf, caja de ahorro y crédito que tiene el usuario.

Dpf, t_dpf Clase y tabla que representa a una cuenta de depósito a plazo fijo.

Private Attributes:

Dpf: Integer El numero de cuenta de la cuenta de depósito a plazo fijo.
Apertura :String Representa la fecha de apertura de dicha cuenta.
Plazo : Integer Representa los días de plazo de la cuenta.
Vcto:String Representa la fecha de vencimiento de la cuenta.
Capital :Float El capital de la cuenta de dpf.
Tasa : Float Representa la tasa de interés que se aplica a dicha cuenta.
Periodo: String Representa el periodo capitalizable, este puede ser mensualmente trimestralmente, etc.
Moneda: String Representa la moneda de la cuenta.

T_dpfp Clase y tabla que representa los movimientos de una cuenta de depósito a plazo fijo.

Dpf: Integer El numero de cuenta de la cuenta de depósito a plazo fijo.
Fecha: String La fecha de operación efectuada sobre dicha cuenta.
Operación:String El Tipo de operación efectuada sobre la cuenta, como ser pago efectivo, apertura de la cuenta, etc.
D_h : String Representa si el movimiento es una operación de debe o haber.
Monto : Float Representa la cantidad de dinero que se maneja en este movimiento.

T_dpfa Clase y tabla que representa el plan de pagos de de una cuenta de depósito a plazo fijo.

Dpf: Integer El numero de cuenta de la cuenta de depósito a plazo fijo.
Fecha: String La fecha de próximo pago de la cuenta de deposito a plazo fijo.
Monto_int : Float Representa el Interés que se pagara en la fecha del plan de pagos.

Caja_ahorro,t_cda Clase que representa a una cuenta de Caja de Ahorro.

Private Attributes:

cda: Integer El numero de cuenta de la cuenta de la caja de ahorro.
Saldo : Float Representa el saldo o dinero que se tiene en la cuenta de caja de ahorro.
Interes : Float Representa la tasa de interés que se aplicara , a la cuenta.
Periodo: String Representa el periodo capitalizable, este puede ser mensualmente trimestralmente, etc.
Moneda: String Representa la moneda de la cuenta.

T_cdaq Clase y tabla que representa los movimientos de una cuenta de caja de ahorro.

Cda: Integer El numero de cuenta de la cuenta de la caja de ahorro.
Fecha: String La fecha de operación efectuada sobre dicha cuenta.
Operación:String El Tipo de operación efectuada sobre la cuenta, como ser pago efectivo, apertura de la cuenta, etc.
D_h : String Representa si el movimiento es una operación de debe o haber.
Monto : Float Representa la cantidad de dinero que se maneja en este movimiento.

Credito, t_cred Clase y tabla que representa a una cuenta de crédito.

Private Attributes:

cda: Integer El numero de cuenta de la cuenta de crédito.
Saldo : Float Representa el saldo del préstamo
monto : Float Representa la cantidad de dinero prestado para este crédito.
Tasa : Float Representa la tasa de interés que se aplica a dicha cuenta.
Plazo : Integer Representa el plazo que se asigno al crédito para su cancelación en días.
f-desemb:String Representa la fecha de desembolso del préstamo.
Tipocuota :String Este campo representa el tipo de cuota del la cuenta de crédito como ser sobre saldos o cuota fija.
Tipoamort : String Representa el tipo de amortización como ser mensual, trimestral, etc.
Dias_amort : Integer Representa la cantidad de días de la amortización, como ser 30, 90 etc. Días.

T_credq Clase y tabla que representa los movimientos de una cuenta de crédito.

Cda: Integer El numero de cuenta de la cuenta de crédito.
Fecha: String La fecha de operación efectuada sobre dicha cuenta.
Operación:String El Tipo de operación efectuada sobre la cuenta, como ser pago efectivo, apertura de la cuenta, etc.
D_h : String Representa si el movimiento es una operación de debe o haber.
Interes:Float La tasa de interés que se aplico a este movimiento.
Otros.Float Otro tipo de pagos que se aplicaron a este pago de crédito.
monto : Float Representa la cantidad de dinero que se maneja en este movimiento.
Saldo_cap : Float Representa el saldo capital a pagar.

Encriptación Clase encargada de la Encriptación-desencriptacion de los datos que se cambian u obtienen de la Bd.

Private Attributes:

clave : String La clave del algoritmo simétrico.

Public Operations:

encripta (textoplano, llave):byte[] Encripta toda información que es pasada en el argumento, y devuelve los datos encriptados en un arreglo de bytes.
Desencripta(textoencriptado, llave):byte[] Desencripta toda información que es pasada en el argumento y devuelve los datos encriptados en un arreglo de bytes.
Llave():byte[] Crea una nueva llave de encriptación.

Bd Representa la clase que interactúa con la Base de Datos.

Private Attributes:

login : String El login de acceso a al Bd.

Password : String La contraseña de acceso a la Bd.

Public Operations:

getUsuario(login) : Usuario Obtiene un usuario a partir de login pasado como argumento.

getUsuario_id(id) : Usuario Obtiene un usuario a partir de identificador de usuario pasado como argumento.

getUsuarios() : Usuarios[] Obtiene todos los usuarios del sistema.

Eliminar (id) : Elimina el usuario especificado por el id pasado como argumento.

setUsuario (Usuario) : Actualiza el usuario pasado como argumento

Cuentas (ci) : Obtiene las cuentas que posee la cuenta pasada como argumento.

setLog(id,tipo,datos,ip) Registra las operaciones realizadas, registrando como el ip desde donde se efectuó la operación que se desea registrar o si la operación se realizó localmente por el administrador. Conjuntamente con esto se registra también el tipo de operación que se está efectuando como ser la eliminación de algún usuario, o la autenticación fallida de otro.

getLogs() : Devuelve todas las operaciones que se han registrado en la base de datos.

BuscarLog(criterio,valor) : Método que realiza la búsqueda de un determinado log, bajo un criterio.

Borrar_logs() : Método que elimina todos los logs de la base de datos, excepto las que son del tipo autenticación que señala que un usuario está dentro del sistema.

Movimientos(ci, id_cuenta) : Método que devuelve todos los movimientos que se han efectuado por el usuario identificado por el ci, y la cuenta por el id_cuenta.

Util Clase que realiza distintas operaciones como ser la lectura de archivos y la validación de algunos datos.

Public Operations:

valido (String cad) : Método que valida la cadena introducida, puesto que algunas cadenas deben ser filtradas antes de ser usadas.

LeerArchivo (nombre) : byte[] Lee un archivo con el nombre pasado como parámetro, y devuelve como un arreglo de bytes.

EscribirArchivo (nombre, datos) : Método que escribe un archivo cuyo nombre junto con sus datos.

EsEntero(String s) : boolean Valida si la cadena pasada como argumento es un entero.

ConnectionPool Clase utilizada para el manejo de conexiones a la base de datos.

Private Attributes:

conexioneslibres : Connection[] Representa un arreglo de conexiones libres

conexionesocupadas : Connection[] Representa un arreglo de conexiones ocupadas, esto cuando una conexión libre pasa a ser una conexión ocupada.

Public Operations:

getConnection () : Connection Método que devuelve del arreglo de conexiones libres una conexión, y la pasa al arreglo de conexiones ocupadas

free (connection) Método que libera una conexión que ya no es utilizada realizando la operación inversa al getConnection.

CerrarTodos() Método que libera todas las conexiones que se han realizado.

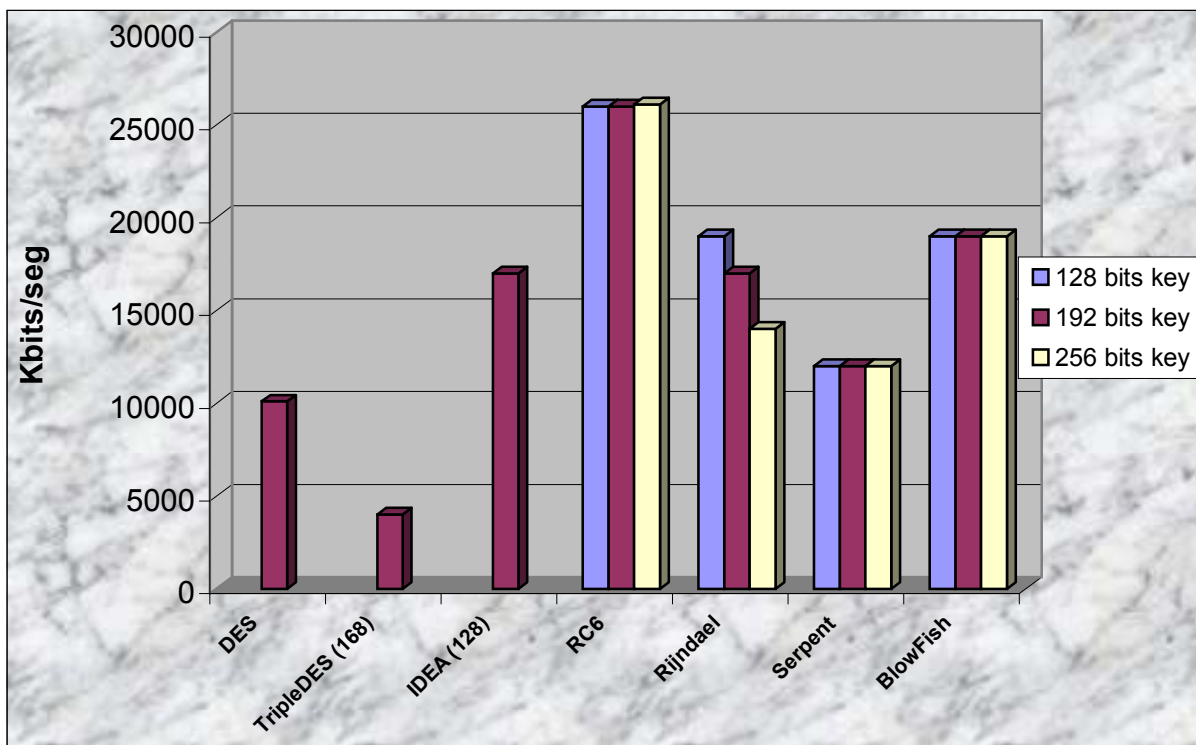
ANEXO 3 – COMPONENTES

<i>Componentes</i>	<i>Clases</i>
Ingreso	Ingreso.
Autentifica	Autentifica.
Menu	Menu, pag_menu.
Listar	Listar, pag_lista.
Extracto	Extracto, pag_extracto.
Pag_cambio	Pag_cambio.
Cambio	Cambio, resultado.
Cuentas_financieras	dpf, caja_ahorro, credito.
Extracto	Extracto, pag_extracto.
Plan	Plan, pag_detalle.
Error	Error, info_error.
Salir	Salir.

ANEXO 4 – RENDIMIENTO DE ALGORITMOS DE ENCRIPCIÓN

En esta sección se muestra el rendimiento de los algoritmos de encriptación simétricos mas importantes, que son implementados por las librería de encriptación de java, JCE (Java Cryptography Extension).

Los datos obtenidos han sido obtenidos de [MON], donde se observa que el algoritmo RC6 tiene el mejor desempeño en comparación con los otros algoritmos, con llaves de longitud hasta de 256 bits. Aquí es necesario aclarar que el único algoritmo que soporta una mayor cantidad de bits es el Blowfish, soportando hasta 448 bits de longitud, no mostrado en el cuadro.

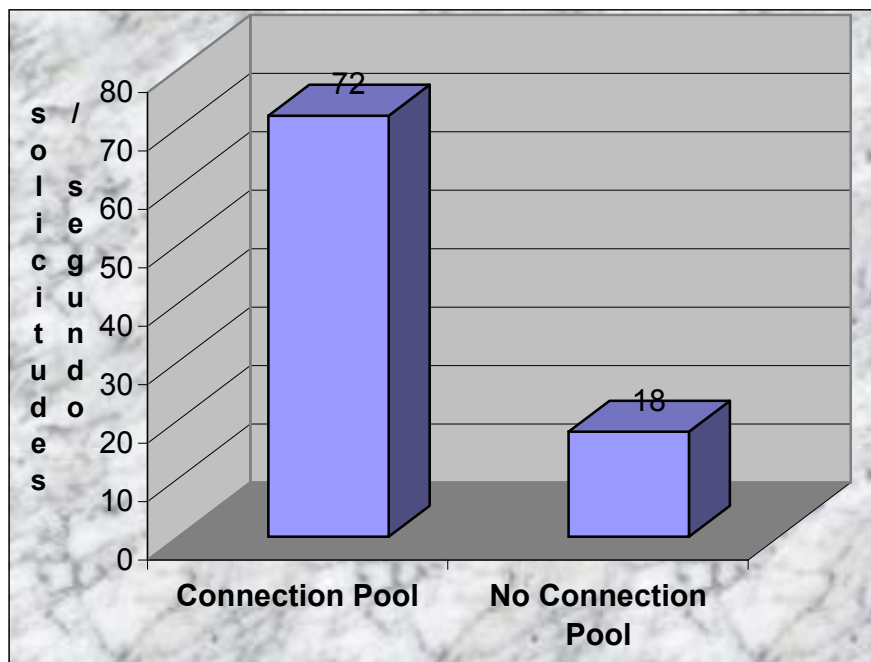


ANEXO 5 – RESULTADO DE PRUEBAS

Las pruebas y mediciones efectuadas a continuación fueron realizadas en un equipo Pentium III, con 800 Mhz de CPU, bajo la plataforma Windows 2000 Advanced Server. Con todas las herramientas especificadas en la parte de herramientas de software del capítulo de implementación.

CONNECTION POOL.-

El siguiente gráfico muestra la diferencia que existe entre la utilización de un “almacén de conexiones” o Connection Pool y sin ella. La medición esta basada en la cantidad de respuestas (en cientos) que puede manejar en un segundo.



A medida que se incrementaba la cantidad de conexiones simultáneas que podía manejar el Connection Pool, la cantidad de solicitudes por segundo también aumentaba, sin embargo con la base de datos PostgreSQL, a diferencia de Mysql, las dos bases de datos con que se efectuó las pruebas, la cantidad de conexiones simultáneas era menor y comenzaba a fallar.

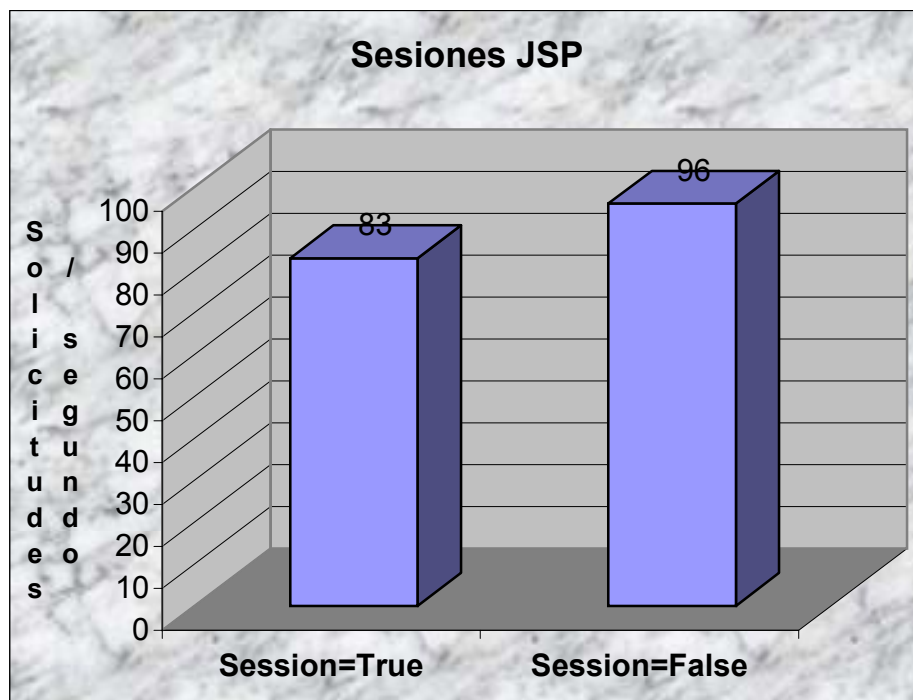
Mysql soporta una cantidad mayor de conexiones, hasta 25 más precisamente, antes que comience a decrecer su rendimiento.

Sin embargo, se pueden ajustar el máximo de conexiones que puede manejar cada base de datos, ajustando parámetros de configuración dentro los DBMS respectivos.

SESIONES.-

Al momento de manejar sesiones dentro de aplicaciones web, se aconseja almacenar objetos pequeños, haciendo menos uso de memoria, así como también desactivar la creación de sesiones por defecto de los archivos JSP, ya que siempre se crean, aunque no se los necesite.

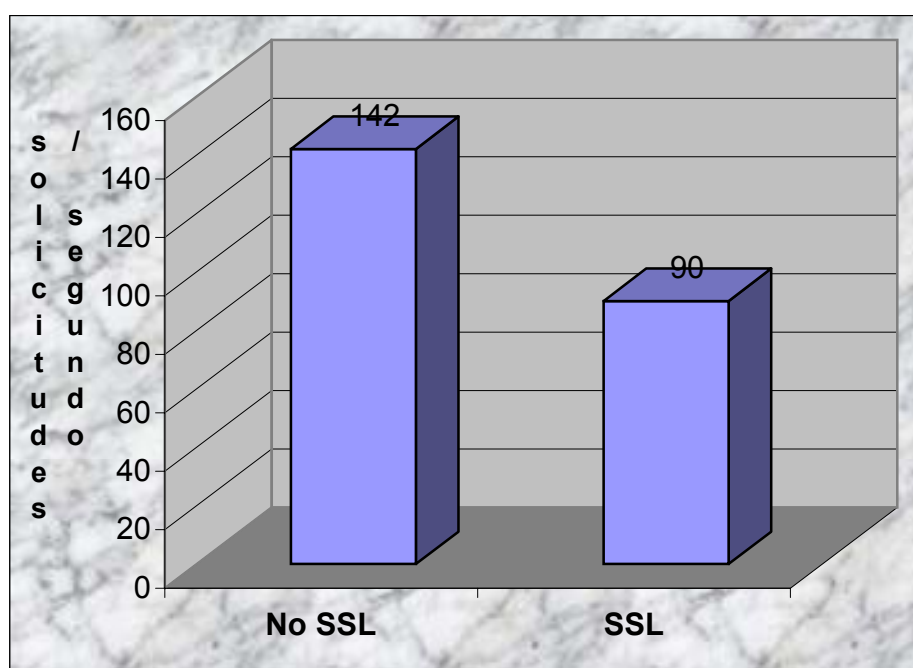
La forma de deshabilitar la creación de sesiones es: `<%@ page session="false" %>`



Otro aspecto que es importante definir, es la duración de cada sesión, cambiando el valor establecido por defecto, por el contenedor o entorno de ejecución en nuestro caso Tomcat, de 30 minutos, a uno menor, ya que este valor es muy elevado para el tipo de aplicación aquí expuesto. Este valor tampoco debe ser muy pequeño ya que podría perjudicar directamente a los usuarios. De acuerdo a las pruebas efectuadas, el valor adecuado para la sesión es de 10 minutos, asegurando que el usuario tiene la cantidad de tiempo suficiente para realizar todas sus operaciones.

SSL.-

Cuando se trabaja con SSL, se incrementa la utilización de recursos del servidor, ya que se tienen que realizar muchas operaciones criptográficas asociadas al HandShake, como ser una fuente de datos impredecible para que trabaje correctamente, la elección del diversos algoritmos criptográficos, la longitud de las llaves, etc. En el siguiente gráfico se compara la diferencia que existe entre el servidor web con soporte de SSL y sin el.



Existen algunas formas de mejorar el volumen de manejo de SSL, sin incurrir a por ejemplo la compra de aparatos criptográficos que se manejen la parte de SSL.

Una de ellas es mantener las conexiones abiertas “Keep Alive”, similar al de la base de datos pero en el servidor web. Esto repercute en la ventaja de no solicitar una nueva sesión SSL, para recibir una página servidor y así de esta manera evitar las operaciones criptográficas costosas asociadas con el “HandShake”.

El servidor web, Apache viene configurado, de una manera que no es la optima para el tipo de aplicación como la nuestra, por lo que se tuvo que modificar el tiempo que se mantiene abierta la conexión a unos 10 minutos, valor similar al de la sesión.

Otro factor que es importante, es el hecho de contar con páginas sencillas y precisas, sin muchos gráficos, ya que también se las encripta, y consume muchos mas recursos.