

005.269  
Ort  
S  
TD-0154-06

TD-0154-06



Universidad de las Ciencias Informáticas.



Instituto Superior Politécnico "José Antonio Echeverría"

Facultad de Ingeniería Industrial  
Centro de Estudios de Ingeniería y Sistemas

## **SISTEMA DE AUTENTICACIÓN DE APLICACIONES EN LA INTRANET.**

**Trabajo de diploma para optar por el título de Ingeniería en Informática**

**Autores:** Idelvis Ortega Ruiz  
Lester Omar Bello Batista

**Tutor:** Ing. Manuel Alejandro Gil Martín

Dirección de Informatización  
Universidad de las Ciencias Informáticas

Ciudad de La Habana, Cuba<sup>o</sup>  
Junio, 2006

## **Agradecimientos**

*A nuestros padres y familiares, por todo su amor, por guiarnos y apoyarnos durante tantos años. Especialmente a nuestros abuelos, vivos y fallecidos. A nuestros hermanos y novias*

*A nuestros compañeros de estudio y esfuerzo, por su amistad, y por compartir tantas cosas buenas y malas, en Camagüey y en La Habana, que durarán en nuestra memoria para siempre.*

*A nuestros compañeros de años superiores, por ser nuestros faros, por su amistad y su ayuda.*

*A nuestro tutor y a las Universidades que nos han dado la preparación para alcanzar este momento: Universidad de Camagüey, CUJAE y UCI.*

*A nuestros amigos...*

## Resumen

La autenticación de aplicaciones en la intranet es un proceso de suma importancia en el contexto de la seguridad en la Intranet universitaria. Actualmente, las aplicaciones manejan la autenticación de manera independiente, lo que trae como consecuencia demoras en el desarrollo de los proyectos y en el tiempo de mantenimiento. En la Universidad hay un gran flujo de accesos a aplicaciones de la intranet, tanto de aplicaciones, como de trabajadores y estudiantes, esto debe estar sometido a un control eficaz para garantizar la seguridad de los sistemas.

Así surge el sistema, que tiene como objetivo concreto servir de interfaz de autenticación (5), brindando servicios al que tendrán acceso otras aplicaciones para manipular el proceso de autenticación (3), la autorización (4) es responsabilidad a cada aplicación, ya que la manera de hacerlo depende de la arquitectura y tecnología de cada una.

El Sistema es extensible, permite una separación entre la manera en que se guardan las credenciales y la lógica que brinda el mismo para las aplicaciones que lo utilizan. Principalmente, el sistema es capaz de soportar diferentes mecanismos para obtener información de usuarios y roles. Además, tiene acceso a reportes para ver patrones de uso de las aplicaciones que lo utilicen, esto tiene un valor agregado considerable y su uso puede ser muy diverso. (Ver Anexo 1).

En este documento se plasman los resultados del estudio realizado en Universidad de las Ciencias Informáticas (UCI) para la construcción del sistema, se explican los conceptos relacionados con el mismo, se hace un análisis e implementación de la propuesta del sistema, y se dejan algunas recomendaciones para el mejoramiento futuro del mismo.

# Índice

<b>INTRODUCCIÓN.....</b>	<b>1</b>
<b>CAPÍTULO 1 FUNDAMENTACIÓN.....</b>	<b>4</b>
1.1 INTRODUCCIÓN .....	4
1.2 OBJETO DE ESTUDIO.....	4
1.2.1 <i>Objetivos estratégicos de la organización</i> .....	4
1.2.2 <i>Flujo actual de los procesos</i> .....	4
1.2.3 <i>Análisis crítico de la ejecución de los procesos</i> .....	5
1.3 PROCESOS OBJETO DE AUTOMATIZACIÓN.....	5
1.4 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN. ....	5
1.5 OBJETIVOS .....	6
1.6 TENDENCIAS Y TECNOLOGÍAS ACTUALES.....	7
1.6.1 <i>Java 2 Enterprise Edition</i> .....	7
1.6.2 <i>Java</i> .....	8
1.6.3 <i>LDAP</i> .....	11
1.6.4 <i>Fundamentación del Gestor de Bases de Datos utilizado</i> .....	12
1.6.5 <i>Web Services</i> .....	13
1.6.6 <i>Metodología RUP (Rational Unified Process)</i> .....	13
1.6.7 <i>UML (Unified Modeling Language)</i> .....	14
1.6.8 <i>Rational Rose</i> .....	15
1.7 CONCLUSIONES.....	16
<b>CAPÍTULO 2 MODELO DEL DOMINIO.....</b>	<b>17</b>
2.1 INTRODUCCIÓN .....	17
2.2 DEFINICIÓN DE LAS ENTIDADES Y LOS CONCEPTOS PRINCIPALES .....	17
2.3 REPRESENTACIÓN DEL MODELO DEL DOMINIO .....	18
2.4 CONCLUSIONES.....	19
<b>CAPÍTULO 3 REQUISITOS.....</b>	<b>20</b>
3.1 INTRODUCCIÓN .....	20
3.2 DEFINICIÓN DE LOS REQUISITOS FUNCIONALES.....	20
3.3 DEFINICIÓN DE LOS REQUISITOS NO FUNCIONALES .....	21
3.4 ACTORES DEL SISTEMA A AUTOMATIZAR.....	24
3.5 DIAGRAMA DE CASOS DE USO DEL SISTEMA A AUTOMATIZAR.....	25
3.6 DESCRIPCIÓN DE LOS CASOS DE USO .....	26
3.7 CONCLUSIONES.....	30

<b>CAPÍTULO 4 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.....</b>	<b>31</b>
4.1 INTRODUCCIÓN .....	31
4.2 DIAGRAMA DE CLASES DEL DISEÑO .....	31
4.2.1 Estructura general de paquetes.....	33
4.3 PRINCIPIOS DE DISEÑO .....	42
4.3.1 Interfaz de usuario.....	42
4.3.2 Formato de salida de los reportes.....	43
4.3.3 Ayuda.....	44
4.4 TRATAMIENTO DE ERRORES .....	44
4.5 DISEÑO DE LA BASE DE DATOS .....	46
4.5.1 Modelo lógico de datos .....	46
4.5.2 Modelo físico de datos.....	47
4.6 DIAGRAMA DE DESPLIEGUE .....	47
4.7 CONCLUSIONES .....	48
<b>CAPÍTULO 5 ESTUDIO DE FACTIBILIDAD .....</b>	<b>49</b>
5.1 INTRODUCCIÓN .....	49
5.2 PLANIFICACIÓN BASADA EN CASOS DE USO.....	49
5.2.1 Paso 1. Cálculo de los Puntos de casos de uso Desajustados.....	49
5.2.2 Estimación de esfuerzo a través de los puntos de casos de uso.....	52
5.3 BENEFICIOS TANGIBLES E INTANGIBLES.....	53
5.4 ANÁLISIS DE COSTOS Y BENEFICIOS .....	54
5.5 CONCLUSIONES .....	54
<b>CONCLUSIONES.....</b>	<b>55</b>
<b>RECOMENDACIONES.....</b>	<b>56</b>
<b>REFERENCIAS BIBLIOGRÁFICAS .....</b>	<b>57</b>
<b>BIBLIOGRAFÍA.....</b>	<b>58</b>
<b>GLOSARIO DE TÉRMINOS .....</b>	<b>59</b>
<b>ANEXO 1 .....</b>	<b>60</b>
<b>ANEXO 2 .....</b>	<b>61</b>

## Índice de tablas

Tabla 1. Definición de actores del sistema a automatizar.....	24
Tabla 2. Descripción del caso de uso < Autenticar usuario > .....	26
Tabla 3. Descripción del caso de uso < Registrar aplicación> .....	27
Tabla 4. Descripción del caso de uso < Configurar aplicación >.....	28
Tabla 5. Descripción del caso de uso < Obtener datos usuario >.....	29
Tabla 6. Descripción del caso de uso < Obtener información del Servicio de Directorio > .....	30
Tabla 7. Descripción del caso de uso < Listar reportes de uso de aplicaciones > .....	30
Tabla 8. Factor de Peso de los Actores sin Ajustar. ....	49
Tabla 9. Factor de Peso de los Casos de Uso sin Ajustar. ....	50
Tabla 10. Factores de Complejidad Técnica. ....	51
Tabla 11. Factores de Ambiente.....	52
Tabla 12. Distribucion del Esfuerzo por Flujo de Trabajo. ....	53

## Índice de figuras

Figura 1 Modelo del dominio.....	19
Figura 2. Diagrama de casos de uso .....	25
Figura 3. Estructura general de paquetes.....	33
Figura 4. Diagrama de clases del paquete Acceso a datos (Subpaquete Interfaces) .....	34
Figura 5. Diagrama de clases del paquete Acceso a datos (Subpaquete Hibernate) .....	35
Figura 6. Diagrama de clases del paquete Acceso a datos (Subpaquete XML).....	36
Figura 7. Diagrama de clases del paquete Acceso a datos (Subpaquete Directorio).....	37
Figura 8. Diagrama de clases del paquete Entidades.....	38
Figura 9. Diagrama de clases del paquete Procesamiento. ....	39
Figura 10. Diagrama de clases del paquete Presentación (Subpaquete RMI).....	40
Figura 11. Diagrama de clases del paquete Presentación (Subpaquete Web Services).....	40
Figura 12. Diagrama de clases del paquete Presentación (Subpaquete Configuración Web).....	42
Figura 13. Interfaz de usuario. ....	42
Figura 14. Uso del color rojo para mensajes de error.....	43
Figura 15. Ejemplo de reporte.....	44
Figura 16. Utilización de mensajes de confirmación .....	45
Figura 17. Modelo lógico de datos.....	46
Figura 18. Modelo físico de datos .....	47
Figura 19. Diagrama de despliegue.....	48

## Introducción

La UCI actualmente cuenta con una gran cantidad de aplicaciones en uso, número que aumenta cada día. Las aplicaciones manejan un gran volumen de información, que necesita en muchos casos de acceso limitado.

Los módulos de seguridad han sido implementados independientemente en proyectos. Algunas aplicaciones utilizan para la autorización, autenticación con el servidor de dominio uci.cu, otros mantienen sus usuarios en base de datos relacionales propias. En la UCI el número de usuarios sobrepasa los 10 000, por lo que el tiempo de búsqueda y los recursos utilizados por cada una de las aplicaciones para el proceso es considerable.

La UCI se encuentra inmersa en un proceso de migración hacia software libre, para ello son necesarios cambios, de forma que, los módulos de seguridad que fueron implementados para una plataforma o sistema de bases de datos determinado, necesitan un gran mantenimiento para migrar. Cada proyecto tiene que desarrollar un modulo de seguridad que se adapte a un lenguaje y métodos de autenticación determinados, provocando una perdida de tiempo en el estudio e implementación de los métodos autenticación que son comunes para muchos sistemas. Algunas aplicaciones que por sus características no necesitan de un soporte de almacenamiento, se ven obligadas a emplearlo para guardar las credenciales de los usuarios.

Por todo esto, se hizo necesario desarrollar un sistema automatizado que controle la autenticación de aplicaciones de manera centralizada en la UCI, reduciendo el tiempo de desarrollo y mantenimiento de los proyectos.

El Sistema de autenticación de aplicaciones de la intranet puede ser capaz de aplicarse a cualquier empresa o institución. Su principal funcionalidad es servir de interfaz de autenticación (5), para el desarrollo de aplicaciones en la intranet. Entre sus principales beneficios se encuentra: el ahorro en el tiempo de desarrollo y mantenimiento de aplicaciones que lo utilicen, debido a que todo lo referente a la autenticación se separaría de los proyectos y si cambia el soporte de las credenciales será transparente para estas aplicaciones, además del valor agregado de los reportes que brinda.



Por tanto el **objeto de estudio** de este trabajo es todo lo referente a la seguridad de aplicaciones en la (UCI).

De aquí, se deriva que el campo de acción queda enmarcado específicamente en el proceso de autenticación de aplicaciones en la intranet

Como **Hipótesis** se parte de la idea de que si se desarrolla un sistema automatizado que controle la autenticación de aplicaciones de manera centralizada en la UCI, entonces es posible reducir el tiempo de desarrollo y mantenimiento de los proyectos.

Para darle respuesta a la situación problemática planteada, se nos hemos trazado como **objetivo general**: elaborar un sistema automatizado que controle la seguridad de aplicaciones de manera centralizada para evitar que los proyectos inviertan tiempo implementando módulos de autenticación, de aquí, se derivan los siguientes **objetivos específicos**: realizar un estudio detallado de la seguridad de aplicaciones en la UCI, estudiar la tecnología necesaria y factible para la construcción del sistema que se propone, implementar una primera versión del sistema hasta la puesta en marcha.

Para cumplir los objetivos trazados, se desarrollaron las siguientes tareas:

- ✓ Realizar un estudio del entorno de trabajo.
- ✓ Identificar las necesidades de la institución.
- ✓ Declarar los requisitos que debe cumplir el sistema.
- ✓ Descripción de los procesos que se van a implementar en el sistema.
- ✓ Declaración de los ciclos de desarrollo.
- ✓ Especificación de los procesos que se van a implementar en el primer ciclo de desarrollo.
- ✓ Modelar conceptualmente las clases que están implicadas en el sistema.
- ✓ Desarrollar los diagramas de actividad.
- ✓ Desarrollar los diagramas que describen el diseño Web del sistema.
- ✓ Descripción de las clases del diseño.
- ✓ Diseño de la Base de Datos.
- ✓ Diseño de la interfaz.
- ✓ Implementación de la aplicación.

Se utiliza la metodología RUP para el desarrollo del sistema y UML como lenguaje de modelación para describir su estructura.

Este trabajo ha sido organizado de la siguiente manera:

En el primer capítulo se recoge los conceptos que se necesitan dominar para explicar en detalles cómo surge y de qué se encarga el Sistema de autenticación de aplicaciones, así como las entidades que la forman y con las que se relaciona. También incluye como aspectos de actualidad una descripción de los lenguajes a utilizar para la implementación del sistema. Trata la situación de las tecnologías a utilizar en el desarrollo de la aplicación, se comparan y seleccionan las mejores propuestas para el trabajo, y se explican los conceptos principales que se van a tratar.

El siguiente capítulo describe el Modelo de Negocio, se hace el análisis del sistema a desarrollar. Se definen las funcionalidades del sistema y se describen detalladamente, utilizando herramientas de modelación, los principales procesos del mismo.

Se empleará un tercer capítulo que mostrará lo referente a los requisitos del sistema propuesto: los actores, los casos de uso, su estructuración por paquetes y las relaciones entre ellos. El capítulo cuarto aborda la elaboración de la solución mediante diagramas de clases, se plantean los principios para el diseño y la implementación. Aquí se desglosan y explican las funcionalidades que se definieron en el capítulo anterior.

*El capítulo final* es un estudio de factibilidad de la construcción del sistema, observando los beneficios tangibles e intangibles y analizando los costos del desarrollo de la propuesta.

# Capítulo 1 Fundamentación

## 1.1 Introducción

El contenido de este capítulo constituye la base teórica del presente trabajo. En él se describen los principales conceptos y lineamientos, como resultado de la investigación realizada para la concepción del Sistema de Autenticación de Aplicaciones.

Se abordan temas relacionados con la seguridad que sirven de base para entender el funcionamiento de la autenticación. Además se realiza una comparación de las herramientas existentes y se determina cuáles van a ser las utilizadas en el sistema.

## 1.2 Objeto de estudio

### 1.2.1 Objetivos estratégicos de la organización

El objeto de estudio de este trabajo es la seguridad de aplicaciones en la UCI.

La UCI es un centro de estudios de nuevo tipo. Su principal reto es la formación de miles de jóvenes de todo el país como futuros profesionales en la rama de la informática.

En ella se garantiza una preparación académica de gran calidad, así como el entrenamiento profesional de los estudiantes, por medio de la participación de forma directa y sistemática en la producción. Se brinda una gran flexibilidad en los diseños curriculares, como nueva alternativa para la formación y capacitación del capital humano vinculado a la informática.

Las perspectivas apuntan al papel de esta Universidad como decisivo en el desarrollo de la industria nacional del software y del programa de informatización. Se están dando grandes pasos en los sectores de la salud y educación, con la realización de proyectos productivos en los mismos. Para ello hacen uso de una moderna infraestructura tecnológica y método novedoso de organización, todo ello tributando a una calidad como la requerida en las normas internacionales.

### 1.2.2 Flujo actual de los procesos

El proceso de autenticación se lleva de manera independiente en cada una de las aplicaciones que requieren de seguridad, siendo similar en cada una de ellas, la manera en que se realiza. Actualmente los procesos fluyen de la siguiente manera:

1. **Registrar usuario:** Las aplicaciones necesitan tener definido en una base de datos o un fichero, quiénes pueden acceder a sus recursos y cuáles son sus privilegios dentro de la misma, para manejar la autorización.
2. **Autenticar:** Cuando un usuario intenta acceder a recursos de una aplicación, esta necesita verificar sus credenciales (6), para verificar que el usuario es quien dice ser y luego denegar o conceder el acceso.

### **1.2.3 Análisis crítico de la ejecución de los procesos**

Para registrar un usuario se necesita previamente del análisis y diseño de un medio de almacenamiento en el que se almacenaran los usuarios y grupos de usuarios, a los que se asigna un privilegio o permiso dentro de la aplicación. Esto es un proceso que es común en las aplicaciones cuando se implementa la seguridad, no obstante es un inconveniente en muchas ocasiones, por ejemplo en el desarrollo de una aplicación Web; para manejar su seguridad eficientemente, esta tendrá que hacerse dinámica, vincularse a una base de datos y el gestor de base de datos seleccionado restaría flexibilidad al sistema. Este es un proceso que aumenta el nivel de complejidad de los sistemas, al que hay que dedicarle un tiempo importante por ser la base de la seguridad de la aplicación, el mal diseño de base de datos dedicada a estos fines, haría vulnerable al sistema. La autenticación puede ser implementada de diferentes formas y está sujeta a protocolos de seguridad que deben seguir los desarrolladores: garantizar la comunicación del sistema con los recursos donde se guardan las credenciales y la contraseña se debe proteger cuando viaja por la red, para que en caso de ser capturada no pueda descifrarse.

### **1.3 Procesos objeto de automatización**

Se desea automatizar el proceso de autenticación, de forma tal que se englobe en un sistema central al que puedan acceder todas las aplicaciones que se registren en la intranet independizando a las aplicaciones de implementar la lógica de este proceso y que permita la recogida de la información necesaria para estudiar patrones de utilización las aplicaciones

### **1.4 Sistemas automatizados existentes vinculados al campo de acción.**

- **Servicio Central de Autenticación (Central Authentication Service(CAS))**

El CAS es un sistema de autenticación creado originalmente por la universidad de Yale, para proporcionar un medio confiable a las aplicaciones para autenticar un usuario. El CAS se convirtió en un proyecto del Grupo de Interés Especial de las Arquitecturas de Java ("JA-SIG", por sus siglas en inglés) en diciembre del 2004. Ofrece Single Sign-On(SSO) para la Web. Los usuarios se autentican al servidor CAS y solo necesitan hacerlo una sola vez por sesión del navegador. Brinda la posibilidad a las aplicaciones de autenticación por Proxy a terceras capas de abastecedores de servicios("back-end"), que eligen aceptar sus credenciales de Proxy.[1]

Este sistema no cumple con todos los requerimientos para satisfacer las necesidades de la UCI, para ello hay que emplear tiempo en estudio y modificación de sus elementos. El núcleo central del proyecto no ha variado desde sus inicios, no ha aprovechado el gran número de soluciones que han surgido desde entonces. Sus últimas versiones se encuentran ligadas a un sistema gestor de base de datos específico.

- **Cams**

El Cams es un SSO software para Web que centraliza la autenticación de usuarios, el control de acceso y la administración. Provee seguridad para recursos que están hospedados en todos los líderes Web y aplicaciones J2EE servidor, están incluidos Apache, Microsoft IIS, BEA WebLogic, IBM WebSphere, JBoss, Oracle 9iAs, Pramati y Tomcat. Los recursos protegidos por Cams pueden residir en una intranet corporativa, extranet o Internet, y pueden ser documentos estáticos y JSP (13)/servlet, ASP.Net, PHP, Cold Fusion, y aplicaciones Web CGI.

Este software satisface muchas de las necesidades de la UCI, pero presenta un elevado costo, y cobran los servicios de soporte que ofrecen.

## **1.5 Objetivos**

### **1.5.1 Objetivo General.**

El objetivo es realizar un sistema que controle de forma centralizada la seguridad de las aplicaciones en la intranet.

### **1.5.2 Objetivos Específicos:**

- ✓ Realizar un estudio detallado sobre el tratamiento a la seguridad de aplicaciones en la UCI como base conceptual

- ✓ Realizar un estudio sobre la tecnología necesaria y factible para la construcción de sistemas que controlen de forma centralizada el proceso de autenticación
- ✓ Realizar todo el proceso de desarrollo del sistema hasta la puesta en marcha

## **1.6 Tendencias y tecnologías actuales**

Una variada gama de tecnologías posibilita brindar una solución efectiva a sistemas que requieren máxima seguridad, robustez y flexibilidad de programación.

### **1.6.1 Java 2 Enterprise Edition**

J2EE (Java 2, Enterprise Edition), versión empresarial de la plataforma de desarrollo de aplicaciones Java 2, de Sun Microsystems, que aporta estándares tecnológicos para la creación de aplicaciones web (servlets, JSP), el acceso a bases de datos (JDBC), el tratamiento de XML (JAXP), servicios de directorio (JNDI), etc.

La utilización de Servidores de Aplicaciones y la ejecución de aplicaciones basadas en tecnología Java 2 Enterprise Edition (J2EE) impactan en mayor o menor medida en distintas disciplinas dentro de una organización, y analizaremos el impacto sobre aquellas que consideramos algunas de las más relevantes:

Modelado de Datos.

Análisis de Requerimientos.

Arquitectura y Diseño.

Codificación.

Testing.

Entorno y Operación.

Administración de Proyectos.

Gerencia de la Configuración Del Software.

#### **Arquitectura y Diseño**

Durante el proceso de diseño de aplicaciones J2EE, la cantidad de alternativas de solución a un mismo problema es extensa. La elección de la más adecuada dependerá de varios factores, entre los que cabe destacar las necesidades de performance, la experiencia del grupo en desarrollo de aplicaciones, la disponibilidad de herramientas correctas y las posibilidades de escalabilidad.

Es en este sentido que la arquitectura física gana relevancia y deberá acompañar en todo momento la toma de decisiones de diseño: la utilización o no de EJBs (Enterprise Java Beans). La delegación o no de cuestiones como persistencia, seguridad y transaccionabilidad en sus

contenedores, la separación de la solución (lógica y eventualmente física) en tres capas, la componentización (packaging) de funcionalidad con la utilización de la novedoso entorno de trabajo Hibernate (11) y su integración con Spring (12), con el objetivo de facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos, entre otras.

### **Codificación**

El estándar J2EE sugiere una arquitectura multicapas.

En una arquitectura de estas características aparece la necesidad de contar con distintos roles de desarrollo. Los componentes correspondientes a la primera capa, el contenedor de clientes y profile de las aplicaciones, serán las encargadas de definir las características de interfase o “look and feel” de las aplicaciones.

### **Administración de Proyectos**

J2EE define los distintos roles de un grupo de desarrollo que utiliza para esta tecnología: Tool Provider, Enterprise Bean Developer, Developer, Web Developer, Application Assembler, Hibernate, Spring, Etc.

## **1.6.2 Java**

Lenguaje desarrollado por Sun Microsystems para la elaboración de aplicaciones exportables a la red y capaces de operar sobre cualquier plataforma.

### **Clases y Objetos**

Java esta totalmente orientado a objetos. Todo en Java (aparte de algunos tipos primitivos) es un objeto. Contrariamente a lenguajes híbridos como C++ o el popular Visual Basic, en Java no se permite programar fuera de un objeto o clase. No hay módulos ni funciones globales.

### **Multiplataforma**

Una característica todavía mas distintiva de Java es su capacidad multiplataforma. Lenguajes como C o COBOL se han implementado en múltiples plataformas, pero siempre han necesitado recopilaciones o adaptaciones al pasar de una a otra. En cambio, Java desde el principio ha sido pensado para adaptarse a varios entornos. Esto lo consigue no sólo a nivel de código fuente, sino también a nivel de código compilado. El programa que escribimos se puede compilar en

Windows o en Linux. Además, el programa compilado puede ejecutarse sin más preparación, en distintas máquinas. Eso lo consigue porque Java se compila y ejecuta, no en un procesador o entorno en particular, sino en lo que se llama “virtual machine”, una máquina virtual. Un programa Java podrá ejecutarse en cualquier sistema operativo que tenga una máquina virtual Java compatible.

### **Paquetes**

Java no es solamente un lenguaje, es una tecnología. Al estar basado en clases y objetos, viene acompañado de un conjunto de estos, que nos sirven como base para la programación de aplicaciones, de texto, gráficas o que se ejecuten en una página Web como “applets”, esta clase se agrupa en paquetes. Podemos nombrar los paquetes AWT (Abstract Window Toolkit) para el manejo de gráficos en cualquier entorno, el paquete Swing con nuevos componentes gráficos, y el JDBC (Java Database Connectivity), para acceso a base de datos, y podemos construir nuestros propios paquetes, o comprar paquetes ya desarrollados.

### **Java en el browser**

Al ser multiplataforma, Java fue adoptado por Netscape como la tecnología con la que se desarrollaron applets, pequeñas aplicaciones que corren dentro de una página Web. Cada navegador implementa una máquina virtual Java, que permite ejecutar las applets en el computador del cliente. Esto ha sido implementado por los navegadores de Netscape y de Microsoft, y en el caso del primero, en varias plataformas. Esto ha permitido que las applets se puedan ejecutar tanto en Windows, como en Unix o Linux.

### **Java en el Servidor**

Si bien Java alcanzó popularidad en la Web, hoy la corriente principal del desarrollo se concentra en el servidor, donde Sun ha desarrollado una plataforma denominada “Java Enterprise”, Java empresarial. El foco se ha puesto en la creación de las aplicaciones de negocio, que corren en servidores preparados para albergarlas.

### **JSP**

Es un acrónimo de Java Server Pages, que en castellano vendría a decir algo como Páginas de Servidor Java. Es pues, una tecnología orientada a crear páginas Web con programación en Java.

Con JSP podemos crear aplicaciones Web que se ejecuten en variados servidores Web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP



están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Por tanto, las JSP podremos escribirlas con nuestro editor HTML/XML habitual. [2]

### **Acceso de base de datos**

Para controlar el acceso a base de datos emplearemos Hibernate que es un potente mapeador objeto/relacional y servicio de consultas para Java. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java.

Hibernate permite desarrollar clases persistentes a partir de clases comunes, incluyendo asociación, herencia, polimorfismo, composición y colecciones de objetos. El lenguaje de consultas de Hibernate HQL (Hibernate Query Language), diseñado como una mínima extensión orientada a objetos de SQL, proporciona un puente elegante entre los mundos objetual y relacional. Hibernate también permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.

Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web, y por supuesto también puede utilizarse en aplicaciones standalone.

#### **Características clave:**

- ***Persistencia transparente.***

Hibernate puede operar proporcionando persistencia de una manera transparente para el desarrollador.

- ***Modelo de programación natural.***

Hibernate soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.

- ***Soporte para modelos de objetos con una granularidad muy fina.***

Permite una gran variedad de mapeos para colecciones y objetos dependientes.

- ***Sin necesidad de mejorar el código compilado (bytecode)***

No es necesaria la generación de código ni el procesamiento del bytecode en el proceso de compilación.

- ***Escalabilidad extrema***

Hibernate posee un alto rendimiento, tiene una caché de dos niveles y puede ser usado en un cluster. Permite inicialización perezosa (lazy) de objetos y colecciones.

- ***Lenguaje de consultas HQL***

Este lenguaje proporciona una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.

- ***Soporte para transacciones de aplicación***

Hibernate soporta transacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución) y gestiona la política optimistic locking automáticamente.

- ***Generación automática de claves primarias.***

Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos (secuencias, columnas autoincrementales,...) así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.

### **1.6.3 LDAP**

Protocolo Ligero de Acceso a Directorios (LDAP) es un conjunto de protocolos abiertos usados para acceder a información guardada centralmente a través de la red. Generalmente, almacena la información de login (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otro tipo de información (datos de contacto del usuario, ubicación de diversos recursos de la red, permisos, certificados...).

Es un sistema cliente/servidor. El servidor puede usar una variedad de bases de datos para guardar un directorio, cada uno optimizado para operaciones de lectura rápida y en gran volumen

#### **¿Cuales son las ventajas de los directorios LDAP?**

LDAP es utilizable por distintas plataformas y basado en estándares, de ese modo las aplicaciones no necesitan preocuparse por el tipo de servidor en que se hospeda el directorio. De hecho, LDAP tiene mucha más aceptación a causa de ese estatus como estándar de Internet. LDAP, no requiere de un servidor específico, puesto que interactuar con cualquier servidor LDAP verdadero acarrea el mismo protocolo, paquete de conexión cliente y comandos de consulta.

No hay que pagar por cada conexión de software cliente o por licencia LDAP, te permite delegar con seguridad la lectura y modificación basada en autorizaciones según las necesidades utilizando Lista de Control de Accesos (ACL). Las ACL pueden controlar el acceso dependiendo de quien está solicitando los datos, que datos están siendo solicitados, dónde están los datos almacenados, y otros aspectos del registro que está siendo modificado.

## 1.6.4 Fundamentación del Gestor de Bases de Datos utilizado

Los principales objetivos de un Gestor de Base de Datos (SGBD) son: evitar la redundancia de los datos, eliminando así la inconsistencia de los mismos, mejorar los mecanismos de seguridad de los datos y la privacidad. Podemos distinguir cuatro tipos de contextos para usar mecanismos de seguridad: seguridad contra accesos indebidos a los datos, seguridad contra accesos no autorizados a la bases de datos, seguridad contra destrucción causada por el entorno seguridad contra fallos del propio sistema (fallos del hardware, del software, etc.).

### 1.6.4.1 MySQL

MySQL es un sistema de administración de Base de Datos. Utiliza la arquitectura cliente/servidor. Es el sistema gestor de bases de datos "Open Source" más popular, o sea que puede ser bajado de Internet y usarlo sin tener que pagar, además que cualquiera puede estudiar su código y adecuarlo a las necesidades que requiera. [3]

"MySQL es muy rápido, fiable y fácil de usar, surge para manipular bases de datos muy grandes. Es un sistema multiplataforma de base de datos relacionales, lo que da velocidad y flexibilidad, cuenta con un sistema de privilegios contraseñas muy seguro que permite la autenticación básica para el acceso al servidor".

### 1.6.4.2 SQL Server

Microsoft **SQL Server** pertenece a la familia de los sistemas de administración de base de datos, operando en una arquitectura cliente - servidor de gran rendimiento. Su desarrollo fue orientado para hacer posible manejar grandes volúmenes de información, y un elevado número de transacciones. Es una aplicación completa que realiza toda la gestión relacionada con los datos. La aplicación cliente sólo tiene que enviarle una cadena de caracteres (la sentencia SQL) y esperar a que le devuelvan los datos.

El motor de SQL Server admite aplicaciones exigentes, entre las cuales se pueden encontrar aplicaciones de ayuda en toma de decisiones y procesamiento de transacciones en línea, mediante la versión de Microsoft del SQL (Transact-SQL).

Permite la creación de procedimientos almacenados, los cuales consisten en instrucciones SQL que se almacenan dentro de una base de datos de SQL Server, realizados en lenguaje SQL, se trata de procedimientos que se guardan semicompilados en el servidor y que pueden ser invocados desde el cliente. Se ejecutan más rápido que instrucciones SQL independientes.

El SQL Server puede manejar perfectamente bases de datos de TeraBytes con millones de registros y funciona sin problemas con miles de conexiones simultáneas a los datos, solo depende de la potencia del hardware del equipo en el que esté instalado

Será utilizado en esta primera versión, una vez probado el sistema se realizarán los ajustes necesarios para que pueda utilizar además MySQL, Postgre SQL y otros, esta capacidad se puede lograr gracias al uso del Hibernate. La selección del SGBD dependerá de los intereses de la UCI en el momento de su puesta en marcha.

### **1.6.5 Web Services.**

Los Web services son la revolución informática de la nueva generación de aplicaciones que trabajan en colaboración, en las cuales el software esta distribuido en diferentes servidores.

Los servicios Web XML permiten que las aplicaciones trabajen en conjunto, haciendo uso de de funcionalidades brindadas por otras aplicaciones independientemente de cómo se hayan creado, cuál sea el sistema operativo o la plataforma en que se ejecutan y cuáles los dispositivos utilizados para obtener acceso a ellas. Aunque los servicios Web XML son independientes entre sí, pueden vincularse y formar un grupo de colaboración para realizar una tarea determinada.

Los servicios XML Web Services son los elementos fundamentales en la evolución hacia la computación distribuida a través de Internet. Se están convirtiendo en la plataforma de integración de aplicaciones gracias a los estándares abiertos y al énfasis en la comunicación y colaboración entre personas y aplicaciones. Las aplicaciones se crean utilizando los servicios XML Web Services múltiples de origen distinto que funcionan conjuntamente, sin importar su ubicación o la forma en que se implementaron.

En la UCI muchas las aplicaciones que se utilizan en la intranet ofrecen o “consumen” servicios Web de otras, es decir, existe una colaboración entre los sistemas de la red para lograr la reutilización y la funcionalidad de estas. Nuestro sistema adiciona el servicio de autenticación.

### **1.6.6 Metodología RUP (Rational Unified Process)**

Para desarrollar un software se necesita una forma ordenada de trabajo. Un proceso que integre las múltiples facetas del desarrollo. Se necesita un método común, un proceso que:

Proporcione una guía para ordenar las actividades de un equipo.

Dirija las tareas de cada desarrollador por separado y del equipo como un todo

Especifique los artefactos que deben desarrollarse

Ofrezca criterios para el control y la medición de los productos y actividades de proyectos

“El Proceso Unificado es un proceso de desarrollo de Software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto...El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas.” [4]

“El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) para preparar todos esquemas de un sistema software , De hecho, UML, es una parte esencial del Proceso Unificado – sus desarrollos fueron paralelos”. [4]

No obstante, los verdaderos aspectos definitorios del Proceso Unificado se resumen en tres frases claves: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Esto es lo que hace único al Proceso Unificado.

### **1.6.7 UML (Unified Modeling Language)**

“UML son las siglas de Unified Modeling Language (Lenguaje Unificado de Modelado), notación (esquemática en su mayor parte) con que se construyen sistemas por medio de conceptos orientados a objetos”. [4].

El Lenguaje Unificado de Modelado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML permite una forma de modelar conceptos como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reutilizables.

UML permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria. Tiene como objetivo

brindar un material de apoyo que le permita al lector poder definir diagramas propios como también entender diagramas ya existentes.

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real.

### **1.6.8 Rational Rose**

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

Es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML.

Esta herramienta propone la utilización de cuatro tipos de modelos para realizar el diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

“Rational Rose utiliza un proceso de desarrollo iterativo controlado (controlled iterative process development), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que estos se hagan mínimos” [5]

Cuando la implementación pasa todas pruebas que se determinan en el proceso, esta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Rational Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades.

“Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML. Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.”[5]

## **1.7 Conclusiones**

Con el estudio de los fundamentos teóricos de las herramientas, tecnologías, lenguajes que se han abordado en este capítulo se ha llegado a la conclusión que el sistema se desarrollará utilizando como gestor de base de datos, el SQL Server 2000, la programación se hará con Java por las ventajas que este brinda y por que la UCI va al software libre, para el análisis y desarrollo se utilizará el Proceso Unificado de Desarrollo (RUP) que a su vez hará uso del Lenguaje Unificado de Modelado(UML) utilizando como herramienta el Rational Rose.

## Capítulo 2 Modelo del dominio

### 2.1 Introducción

En este capítulo se presenta el negocio a través de un modelo de dominio debido a que el objetivo primario es la gestión y presentación de información y que el negocio que se está estudiando, tiene poca estructuración en sus procesos; para poder entender el contexto en que se emplaza el sistema necesitamos definir conceptos que podemos agrupar en un Modelo de Dominio, para capturar correctamente los requisitos y poder construir un sistema correcto. Se definen entidades y conceptos principales.

Un Modelo del Dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos[6]. La modelación del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes en el sistema.

### 2.2 Definición de las entidades y los conceptos principales

*“Definición de las entidades y los conceptos principales (cosas que existen y eventos que suceden) del entorno en el que trabajará el sistema.”*

#### **Aplicación:**

Es la entidad que representa al cliente que desea desarrollar el proceso de autenticación, se encarga de gestionar lo referente a sus usuarios y grupos de usuarios.

#### **Usuario:**

Es una cuenta creada para representar un individuo (persona o aplicación) que tiene derechos dentro de la aplicación, para acceder a sus recursos.

**Servicios de Directorio:** Esta entidad se encarga de la gestión de los usuarios de las aplicaciones que se encuentran en un directorio de servicios. Implementando la funcionalidad necesaria para brindar los servicios de autenticación. La comunicación se implementa mediante LDAP v3 ("Lightweight Directory Access Protocol").



**Grupo:**

La característica que va a diferenciar a un usuario o grupo de usuarios dentro de una aplicación, por el cual se determina los derechos a acceder a determinados recursos. Un grupo es un contenedor de usuarios y grupos.

Ejemplo: administrador, empleado, cliente.

**Sistema:**

Es la clase del sistema encargada de brindar todas las funciones, manipula las aplicaciones y su persistencia.

**Fichero XML:**

Es la entidad que se encarga de almacenar los datos (aplicaciones, usuarios y grupos) en ficheros.

**Interfaz Servicio Web (Web Service):**

Esta entidad es una interfaz que exporta los métodos que serán necesarios a las aplicaciones clientes, para desarrollar el proceso de autenticación.

**Interfaz de Invocación Remota de Métodos (RMI):**

Esta entidad es una interfaz que exporta los objetos que serán necesarios a las aplicaciones clientes, para desarrollar el proceso de autenticación. Esta interfaz se crea los objetos remotos de las clases y hace accesibles unas referencias a dichos objetos remotos, y espera a que los clientes llamen a estos métodos u objetos remotos.

**Interfaz Web de Configuración:**

Es una aplicación Web que brinda los servicios de configuración de cada aplicación, gestión de usuarios y grupos, etc. y los reportes para ver patrones de uso de las aplicaciones. Permitirá probar de forma visual las funciones del sistema.

## ***2.3 Representación del modelo del dominio***

El modelo del dominio se describe mediante diagramas UML, específicamente con un diagrama clases conceptuales significativas en el dominio del problema.

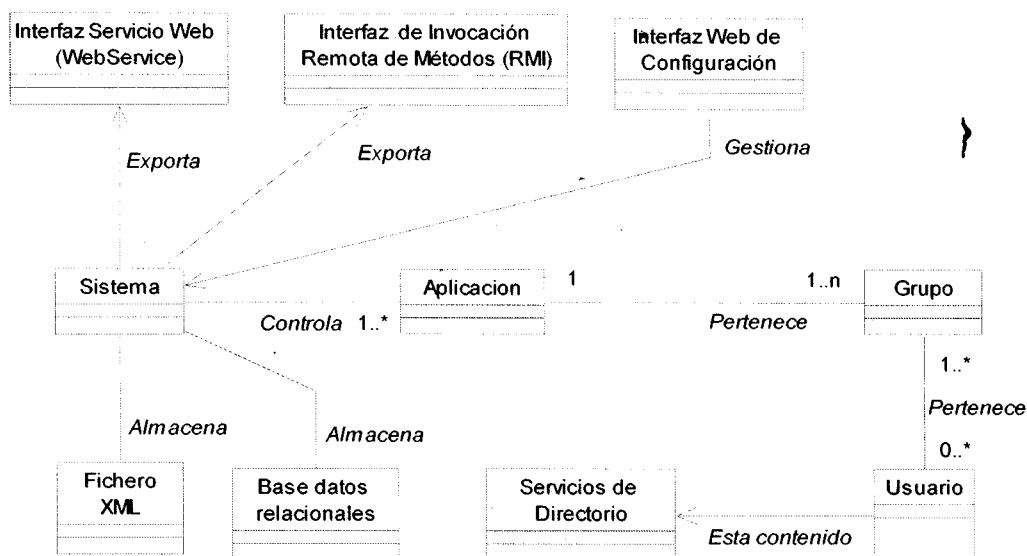


Figura 1 Modelo del dominio

## 2.4 Conclusiones

Se llega a la conclusión de que el negocio que se está estudiando, tiene muy bajo nivel de estructuración, con soluciones similares que llevan el mismo propósito, satisfacer una necesidad, “seguridad de la información”, que tiene como base el proceso de autenticación. Por lo que en este capítulo se definieron las entidades y conceptos agrupándolas en un, modelo de dominio y se logro una mejor comprensión del contexto donde trabaja el sistema.

## Capítulo 3 Requisitos

### 3.1 Introducción

En el presente capítulo se enumeran los requisitos funcionales y no funcionales que debe tener el sistema que proponemos, lo que permite hacer una concepción general del sistema, e identificar mediante un Diagrama de Caso de Uso, las relaciones de los actores que interactúan con el sistema, y las secuencias de acciones con las que interactúan.

### 3.2 Definición de los requisitos funcionales

#### R 1. Administrar Sitio

- 1.1 Pedir nombre de usuario y contraseña para entrar a los recursos que se están solicitando.
- 1.2 Enviar directamente al usuario registrado a las opciones principales con que cuenta su nivel registrado.
- 1.3 Permitir que el usuario registrado cambie sus datos o modifique su contraseña.
- 1.4 Permitir que el usuario registrado cierre su sesión de trabajo desde cualquier lugar del sistema.

#### R 2. Obtener datos de Usuario, en solicitud de aplicaciones clientes.

- 2.1 Verificar las credenciales de usuarios (usuario y contraseña) para la autenticación.
- 2.2 Obtener información básica del usuario, nombre, descripción, etc.
- 2.3 Obtener grupos a los que pertenezca el usuario, así como descripción de los mismos.
- 2.4 Verificar la pertenencia de un usuario a un grupo, para tomar decisiones de autorización.

#### R 3. Registrar aplicación

- 3.1 Insertar nueva aplicación, llenar datos iniciales.
- 3.2 Crear usuario administrador de aplicación.

#### R 4. Configurar aplicación.

- 4.1 Permitir cambiar datos iniciales de la aplicación.
- 4.2 Administrar Grupos.
  - 4.2.1 Insertar grupo.
  - 4.2.2 Modificar datos del grupo.

4.2.3 Eliminar grupo.

4.3 Administrar Usuarios.

4.3.1 Agregar nuevo usuario, especificando los datos y Grupo

4.3.2 Modificar datos usuario y cambiar contraseña si es local.

4.3.3 Eliminar usuario.

**R 5. Obtener información del Servicio de Directorio.**

5.1 Listar usuarios filtrado por nombre y usuario.

5.2 Autenticar dado usuario y contraseña.

**R 6. Listar reportes de uso de aplicaciones,** para obtener patrones de uso por periodo de tiempo.

6.1 Listar reportes para todas las aplicaciones:

6.1.1 Mostrar ordenadas por cantidad de accesos.

6.1.2 Mostrar ordenadas por cantidad de usuarios que accedieron.

6.2 Listar reportes por aplicaciones.

6.2.1 Mostrar cantidad de accesos por usuarios.

6.2.2 Mostrar todos los accesos de un usuario.

6.2.3 Mostrar todos los accesos con usuario.

### **3.3 Definición de los requisitos no funcionales**

#### **Apariencia o interfaz externa:**

- Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.
- Paginación de reportes de búsqueda, y listados.

#### **Usabilidad:**

- El sistema está concebido para ser usado por los desarrolladores de las aplicaciones de la intranet, por lo que es necesario que cuente con un diseño de interfaz profesional de fácil uso teniendo en cuenta que muchos de ellos no tienen gran experiencia en el manejo de Web Services y RMI, y porque muchos de los desarrolladores de la universidad son estudiantes, que no tienen experiencia en el manejo de la tecnología Web.
- Documentar bien el sistema y proporcionar materiales de ayuda para hacer mejor uso de todos los servicios que este ofrece.

- La interfaz Web de configuración del sistema podrá ser usada por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.
- Deberá visualizarse en los principales navegadores (Internet Explorer, Mozilla Firefox, Opera, Maxthon).
- El sistema debe ser independiente de la base de datos.

**Rendimiento:**

- La aplicación debe ser eficiente, precisa y con tiempos de respuestas rápidos al igual que la velocidad de procesamiento de la información. Ya que de ella dependerán otros sistemas para su funcionamiento.

**Soporte:**

- El sistema debe ser de fácil instalación, adaptable a numerosas plataformas y de fácil mantenimiento.

**Portabilidad:**

- Necesidad de que el sistema sea multiplataformas.

**Seguridad:** (Revisión bibliográfica del tema, especificar para el sistema la propuesta de: controles de seguridad y privacidad, seguridad física, controles administrativos y requerimientos funcionales que genera). En caso de que la empresa objeto de estudio cuente con políticas de seguridad bien establecidas y/o sistemas informáticos de seguridad, especificar cómo encaja en ellos el sistema propuesto.

- El sistema debe tener confidencialidad e integridad de los datos.
- Garantizar que las funcionalidades del sistema se ejecuten de acuerdo al privilegio del usuario.
- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- Verificación sobre acciones irreversibles (eliminaciones).
- Utilizar contraseñas de usuario que cumplan con políticas de complejidad y longitud.
- Guardar un hash(8) de la contraseña de los usuarios, utilizando un algoritmo de hashing(9) unidireccional

- Utilizar HTTPS(10) y RMI, para la comunicación entre las aplicaciones y el sistema

**Legales:**

- Reglas de seguridad informática del centro.

**Confiabilidad:**

- El sistema deberá tener un 100% de disponibilidad, porque podrá ser usado las 24 horas del día.
- El tiempo medio de reparación debe ser menor de 1 día.
- Todas las salidas del sistema tienen que tener el 100% de veracidad y precisión.
- Capaz de recuperarse rápidamente a las fallas del sistema.

**Interfaz interna:**

- Se define una Portada Estándar para Usuarios Anónimos la cual será el punto de entrada de todos los usuarios del sistema.
- Se define una Portada Personalizada para cada usuario del Sistema, la cual se visualiza a los administradores que se autentican, dándole la posibilidad de interactuar con un conjunto de servicios en dependencia de su rol.
- Las interfaces que soporta la aplicación son: cliente Web con protocolo http, cliente Web Service con protocolo HTTPS y cliente RMI
- Interfaz de usuario: Para el diseño de la interfaz de usuario se utilizará un diseño personalizado.
- Interfaz de hardware: Debe estar soportado por una red.
- Interfaz de software: La aplicación se realizará en ambiente Web.
- Interfaz de comunicación: La aplicación se comunica con los usuarios del sistema a través de la red de área local.

**Ayuda y documentación en línea:**

- La aplicación debe tener una ayuda en línea soportada por páginas Web, que estará disponible al usuario en todo momento.

**Software:**

- Para el cliente Navegador compatible o superior con Internet Explorer 4, o Mozilla Firefox.
- Java 2 Platform, Standard Edition, v 1.4 o superior.
- Servidor Web Apache Tomcat 5.0 o superior
- Un servidor de bases de datos (SQL Server, MySql Server, Postgre Sql u otro ).
- **Hardware:**
- Un servidor.
- Una red de computadoras.

**Restricciones en el diseño y la implementación:**

- Para el análisis y el diseño del sistema debe ser utilizada la metodología RUP, usando el lenguaje de modelación UML y como herramienta para llevarlo a cabo el Rational Rose.
- El diseño de la base de datos debe ser estándar, utilizar características que sean soportadas por todos los gestores de bases de datos.

### **3.4 Actores del sistema a automatizar**

<b>Nombre del actor</b>	<b>Descripción</b>
<b>Aplicación</b>	Representa un sistema informático (aplicación) que utiliza el sistema y tiene la capacidad de obtener datos del usuario
<b>Usuario</b>	Representa a una persona que puede autenticarse.
<b>Administrador de Aplicación</b>	Generaliza al administrador del sistema. Puede configurar una aplicación, obtener información del Servicio de Directorio y ver reportes de uso de aplicaciones. Encapsula además el comportamiento de usuario
<b>Administrador del Sistema</b>	Encapsula el comportamiento de Administrador de Aplicación y puede además: registrar una nueva aplicación
<b>Directorio</b>	Actor pasivo que brinda al sistema la información de los usuarios que se encuentran en un Servicio de Directorio

**Tabla 1. Definición de actores del sistema a automatizar**

### 3.5 Diagrama de casos de uso del sistema a automatizar

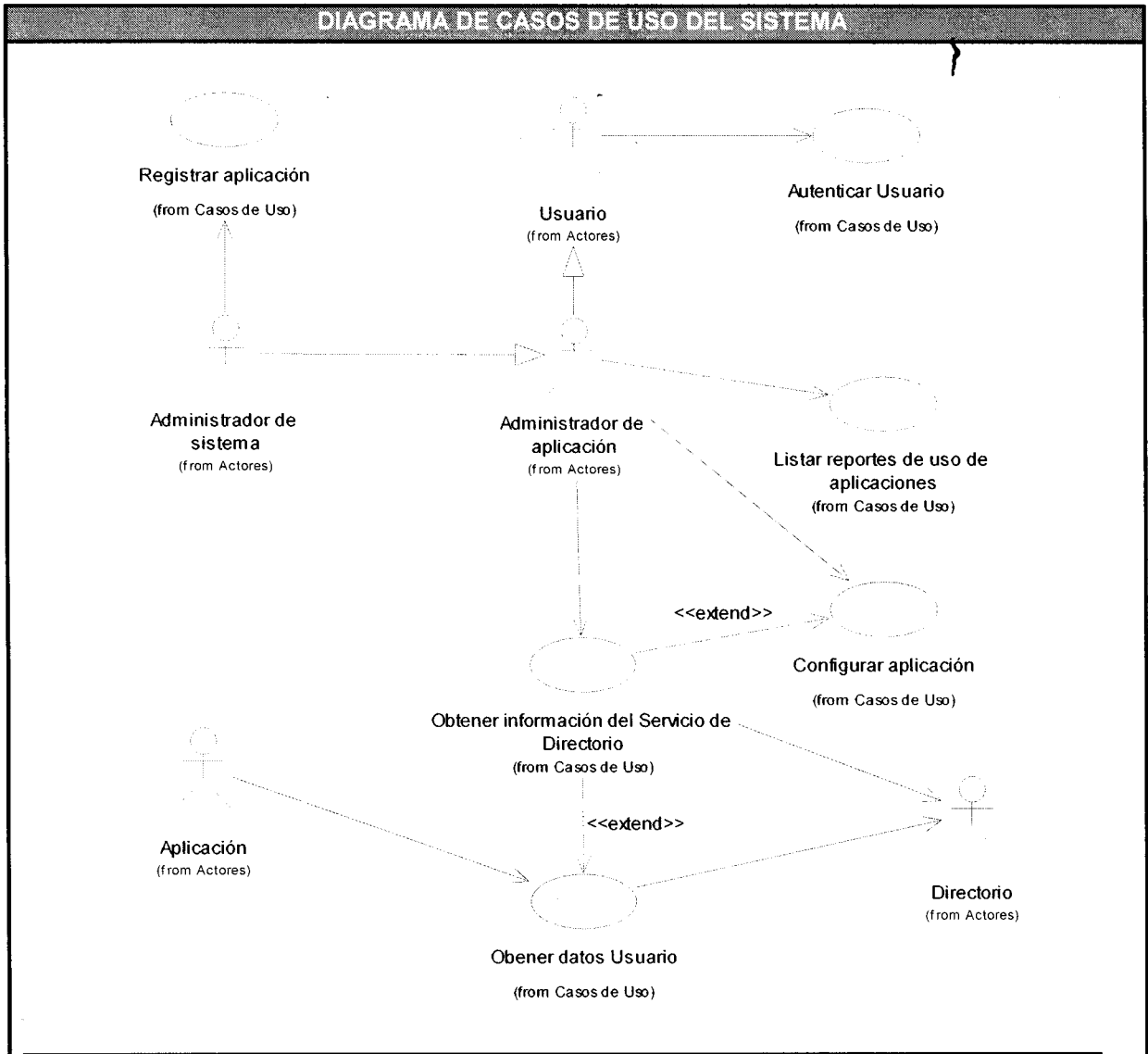


Figura 2. Diagrama de casos de uso



### 3.6 Descripción de los casos de uso

<b>Nombre del caso de uso</b>	<b>Autenticar usuario</b>
<b>Actores</b>	Usuario (inicia)
<b>Propósito</b>	Posibilita a los usuarios identificarse contra el sistema para acceder a las opciones disponibles para su nivel de acceso.
<b>Resumen</b>	El caso de uso se inicia cuando el Usuario intenta acceder a recursos que requieren de privilegios, el sistema solicita mediante una página de identificación que introduzca sus credenciales, las verifica y en caso de ser correctas le permite acceder a los recursos a los cuales tiene acceso, en caso de no ser válidas, el sistema deniega el acceso e informa al usuario para que este introduzca las correctas. El caso de uso concluye cuando las credenciales son verificadas como correctas o el usuario cierre la página de identificación.
<b>Precondiciones</b>	
<b>Poscondiciones</b>	El usuario puede acceder a los recursos del sistema de acuerdo a sus privilegios y no tiene que volver a autenticarse mientras no cierre la sesión.
<b>Requisitos especiales</b>	

Tabla 2. Descripción del caso de uso < Autenticar usuario >

<b>Nombre del caso de uso</b>	Registrar aplicación
<b>Actores</b>	Administrador de Sistema (inicia)
<b>Propósito</b>	Permitir al Administrador de Sistema crear una nueva aplicación con el usuario Administrador de aplicación, para delegar el resto de la configuración a este.
<b>Resumen</b>	El caso de uso se inicia cuando el actor accede a la página de configuración del sistema para realizar el registro de una nueva aplicación, necesitando datos como nombre y descripción que debe introducir, y un identificador que es generado por el sistema, El registro no concluye hasta que no se haya creado el usuario administrador de la aplicación, dentro del grupo administradores que el sistema tiene por defecto para cada aplicación que se registre.

<b>Precondiciones</b>	Tiene que estar autenticado el usuario administrador del sistema, solo este tiene privilegios para realizar la operación.
<b>Poscondiciones</b>	La aplicación ya esta habilitada para su uso y puede ser configurada por el administrador de la aplicación.
<b>Requisitos especiales</b>	

**Tabla 3. Descripción del caso de uso < Registrar aplicación >**

<b>Nombre del caso de uso</b>	<b>Configurar aplicación</b>
<b>Actores</b>	Administrador de Aplicación (Inicia).
<b>Propósito</b>	Permitir modificar la configuración de la aplicación, definiendo: usuarios y grupos del sistema e información inicial de la aplicación.
<b>Resumen</b>	<p>El caso de uso se inicia cuando el Administrador de Aplicación accede la página de configuración de aplicaciones donde puede cambiar datos iniciales de la Aplicación (<b>A</b>), administrar Usuarios (<b>B</b>) y administrar Grupos (<b>C</b>).</p> <p><b>A-</b> Puede cambiar los datos de la aplicación como nombre, descripción y ver el número de identificación.</p> <p><b>B-</b> Puede agregar un nuevo usuario introduciendo el nombre, la descripción y en caso que lo desee el grupo en el que estará contenido. En los usuarios que existen se puede cambiar el nombre, la descripción y los grupos en el que está contenido. Puede eliminar un usuario seleccionado</p> <p><b>C-</b> Puede agregar un nuevo usuario para lo que debe introducir el nombre de usuario, nombre completo y la contraseña, si el usuario que desea adicionar se encuentra en un Servicio de Directorio entonces puede buscarlo(a través del caso de uso obtener información del Servicio de Directorio) filtrando por nombre de usuario y/o nombre completo, en este caso se introducen automáticamente los datos, excepto la clave, se le deja la responsabilidad de almacenarla al controlador de dominio. Para modificar los datos del usuario este debe ser local de la aplicación, pues los usuarios del Servicio de Directorio s no se les pueden modificar sus datos. Los datos a modificar son los introducidos en el proceso de inserción.</p> <p>El caso de uso se termina cuando se salvan los cambios efectuados y se sale</p>

	de la página.
<b>Precondiciones</b>	El Usuario está autenticado en el sistema como Administrador y seleccionó la opción de administrar aplicación.
<b>Poscondiciones</b>	La aplicación queda actualizada con los cambios efectuados.
<b>Requisitos especiales</b>	

**Tabla 4. Descripción del caso de uso < Configurar aplicación >**

<b>Nombre del caso de uso</b>	<b>Obtener datos Usuario</b>
<b>Actores</b>	Aplicación (inicia), Directorio.
<b>Propósito</b>	Brindar las funciones necesarias a las aplicaciones clientes, para desarrollar los procesos de autenticación y autorización logrando la seguridad de las aplicaciones.
<b>Resumen</b>	<p>El caso de uso se inicia cuando una aplicación registrada solicita al sistema la realización de una o varias de las siguientes funciones (Pueden ser accedidas a través del Web Service o por RMI):</p> <ul style="list-style-type: none"> <li>A. Verificar las credenciales de usuarios (usuario y contraseña) para la autenticación. La aplicación en la solicitud envía al sistema su identificador, con el usuario y contraseña a comprobar. Si el usuario es del directorio, entonces hay que verificar a través del caso de uso Obtener información del Servicio de Directorio. Si el usuario es local entonces se comprueban en los datos de la aplicación en el sistema.</li> <li>B. Obtener información básica del usuario, nombre, descripción. Para obtener estos datos, la aplicación en la solicitud envía el su identificador y usuario del que se requiere la información. Esta información se encuentra en el sistema para los usuarios de la aplicación.</li> <li>C. Obtener grupos a los que pertenece el usuario, así como descripción de los mismos. En la solicitud la aplicación envía su identificador y el usuario, se obtiene como respuesta del sistema, una lista con los grupos en los que está contenido el usuario.</li> </ul>

	<p>D. Verificar la pertenencia de un usuario a un grupo, para tomar decisiones de autorización. En la solicitud se envía el identificador de la aplicación, el usuario y el grupo, el sistema busca en el grupo enviado el usuario, si no se encuentra, entonces busca en todos los grupos hijos, de pertenecer a alguno entonces se considera verdadera la pertenencia al grupo.</p> <p>El caso de uso culmina cuando es respondida la solicitud de la aplicación cliente.</p>
<b>Precondiciones</b>	Tienen que estar disponibles las funciones en el Web Services y por RMI. Además de tener configurado los usuarios y los grupos previamente en la aplicación.
<b>Poscondiciones</b>	
<b>Requisitos especiales</b>	

**Tabla 5. Descripción del caso de uso < Obtener datos usuario >**

<b>Nombre del caso de uso</b>	<b>Obtener información del Servicio de Directorio.</b>
<b>Actores</b>	Administrador de aplicación (inicia), Directorio.
<b>Propósito</b>	Brindar las funciones necesarias para que se pueda obtener información de los usuarios que se encuentran el Directorio
<b>Resumen</b>	<p>El caso de uso se inicia cuando el Administrador de Aplicación, solicita información referente a un usuario que se encuentra el Directorio. Puede obtener una lista de usuarios del dominio, para la búsqueda se filtra por los campos: Nombre completo y/o usuario (login) de forma que se pueda disminuir el número de usuarios de la vista facilitando la búsqueda del deseado. El filtrado se puede realizar avanzado, con las opciones comunes de búsqueda: "comienza con", "está contenido", "es", "termina con". Esta función es necesaria en el momento de agregar un usuario a la aplicación. Además se puede autenticar las credenciales de un usuario del Directorio, ya que el usuario y contraseña se encuentra en el mismo.</p> <p>El caso de uso culmina cuando es dado el resultado de las operaciones solicitadas.</p>

<b>Precondiciones</b>	Tiene que estar configurada la conexión con el Servicio de Directorio con un usuario y contraseña con permisos para realizar las operaciones.
<b>Poscondiciones</b>	
<b>Requisitos especiales</b>	

**Tabla 6. Descripción del caso de uso < Obtener información del Servicio de Directorio >**

<b>Nombre del caso de uso</b>	<b>Listar reportes de uso de aplicaciones</b>
<b>Actores</b>	Administrador de aplicación (inicia).
<b>Propósito</b>	Obtener patrones de uso de las aplicaciones que lo utilizan y de uso del sistema.
<b><u>Resumen</u></b>	<p>El caso de uso se inicia cuando un Usuario accede a la página de Reportes de Uso, en esta, puede obtener los siguientes reportes:</p> <p>Para todas las aplicaciones:</p> <ul style="list-style-type: none"> <li>• Mostrar ordenadas por cantidad de accesos.</li> <li>• Mostrar ordenadas por cantidad de usuarios que accedieron.</li> </ul> <p>Por aplicaciones.</p> <ul style="list-style-type: none"> <li>• Mostrar cantidad de accesos por usuarios.</li> <li>• Mostrar todos los accesos de un usuario.</li> <li>• Mostrar todos los accesos con usuario</li> </ul>
<b>Precondiciones</b>	Esperar un tiempo desde la puesta en marcha para obtener mejores estimaciones en los patrones.
<b>Poscondiciones</b>	
<b>Requisitos especiales</b>	

**Tabla 7. Descripción del caso de uso < Listar reportes de uso de aplicaciones >**

### **3.7 Conclusiones**

En este capítulo se desarrolló un listado con las funciones que debe tener el sistema, que se representaron mediante un Diagrama de Casos de Uso, y finalmente se describieron paso a paso todas las acciones de los actores del sistema con los casos de uso con los que interactúan. Gracias a esto ahora se puede empezar a construir el sistema, tratando de que se cumplan todos los requerimientos y las funciones que han sido consideradas necesarias en este capítulo.

## Capítulo 4 Descripción de la solución propuesta

### 4.1 Introducción

En el presente capítulo se modelan los artefactos de diseño del sistema, utilizando para su modelado el Lenguaje Unificado de Modelado (UML). Se define el diagrama de clases, se tratan los principios del diseño del sistema, se explica el tratamiento de errores. En los últimos epígrafes, se diseña de la Base de Datos y se realizan los diagramas de despliegue.

### 4.2 Diagrama de clases del diseño

Para una mejor comprensión del diagrama de clases del presente trabajo, se han separado las mismas en 4 paquetes atendiendo a su funcionalidad, dichos paquetes se han dividido a su vez en otros subpaquetes para una mayor organización y legibilidad.

El paquete *AccesoDatos* contiene las clases para hacer posible la persistencia y recuperación de objetos. Está dividido en cuatro subpaquetes: *Interfaces*, en el que se encuentran las interfaces con los métodos que tienen que implementar todos los proveedores de datos, para garantizar que un cambio proveedor de datos, sea transparente para el resto de la aplicación. *XML*, que abarca las clases para manipular la persistencia de las entidades utilizando para ello ficheros XML. El subpaquete *Hibernate*, contiene las clases encargadas de acceder a la base de datos para manipular la persistencia de las entidades. El Subpaquete *Directorio* que abarca las clases para acceder a Servicios de Directorio de la Intranet, necesarios para obtener información sobre personas de la UCI. El Paquete *AccesoDatos* en general permite a la aplicación abstraerse del origen de los datos y de la lógica de su persistencia, logrando un bajo acoplamiento entre sus componentes.

El paquete *Procesamiento* contiene la lógica de negocio de la aplicación. Contiene las clases controladoras de todas las entidades. Implementa las funciones necesarias para brindar y administrar los servicios de autenticación, así como la validación de datos y tratamiento de excepciones.

El paquete *Entidades* contiene clases que no tienen comportamiento, sólo propiedades y son representaciones de entidades reales del dominio, son clases persistentes que son accedidas por las clases de los paquetes *Procesamiento* y *AccesoDatos*.

Por último, el paquete **Presentación** contiene las clases de presentación del sistema, en este caso, el se cuenta con varias interfaces. Este paquete está dividido en 3 subpaquetes: **Configuración Web**, que contiene las páginas relacionadas con la administración de las aplicaciones clientes; el subpaquete **Web Services**, que contiene una clase que exporta las funciones necesarias para realizar el proceso de autenticación a través de HTTPS. El subpaquete **RMI**, contiene una clase para permitir la invocación remota de métodos (RMI) brinda iguales funciones que el paquete Web Services.

Esta división por paquetes obedece a la arquitectura dividida por capas en que se ha diseñado la aplicación y también a la funcionalidad de las clases, las cuales han sido agrupadas de esta forma para lograr mayor desacoplamiento, reutilización y legibilidad de los diagramas.

#### 4.2.1 Estructura general de paquetes.

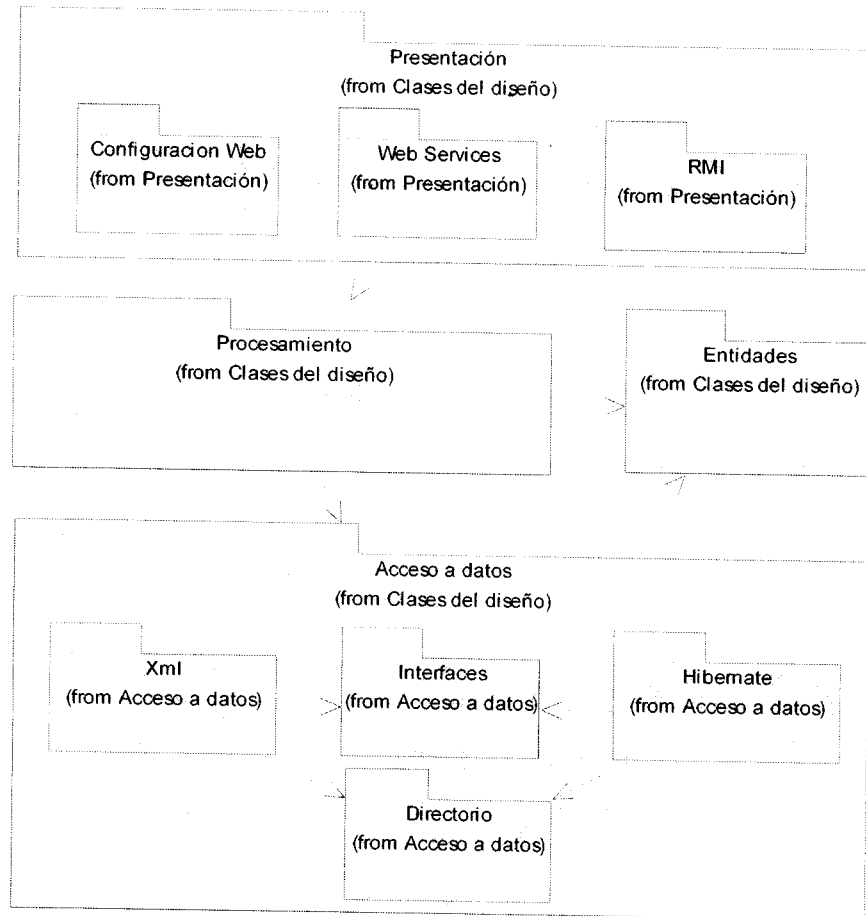


Figura 3. Estructura general de paquetes



### 4.2.1.1 Paquete Acceso a datos (Subpaquete Interfaces)



Figura 4. Diagrama de clases del paquete Acceso a datos (Subpaquete Interfaces)

#### 4.2.1.2 Paquete Acceso a datos (Subpaquete Hibernate)

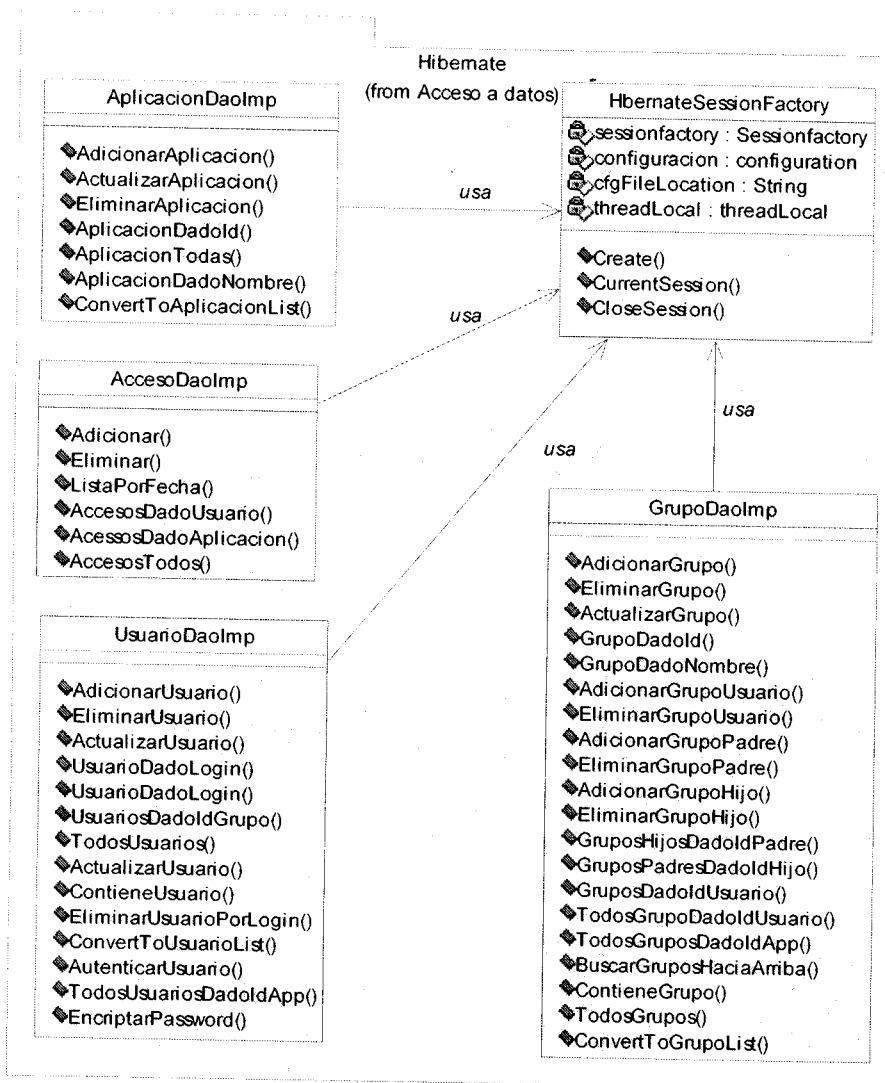


Figura 5. Diagrama de clases del paquete Acceso a datos (Subpaquete Hibernate)

### 4.2.1.3 Paquete Acceso a datos (Subpaquete XML)



Figura 6. Diagrama de clases del paquete Acceso a datos (Subpaquete XML)

#### 4.2.1.4 Paquete Acceso a datos (Subpaquete Directorio)

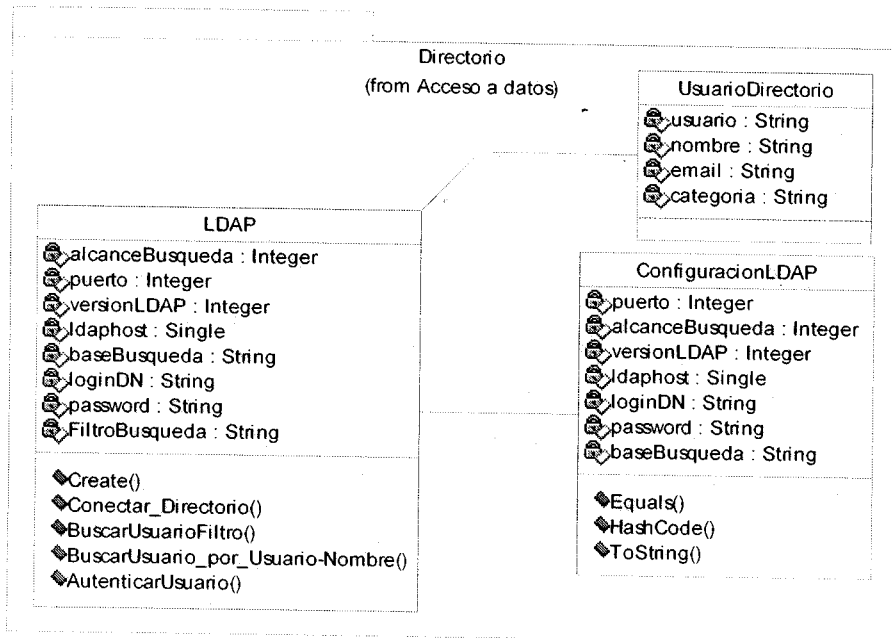


Figura 7. Diagrama de clases del paquete Acceso a datos (Subpaquete Directorio)

### 4.2.1.5 Paquete Entidades.

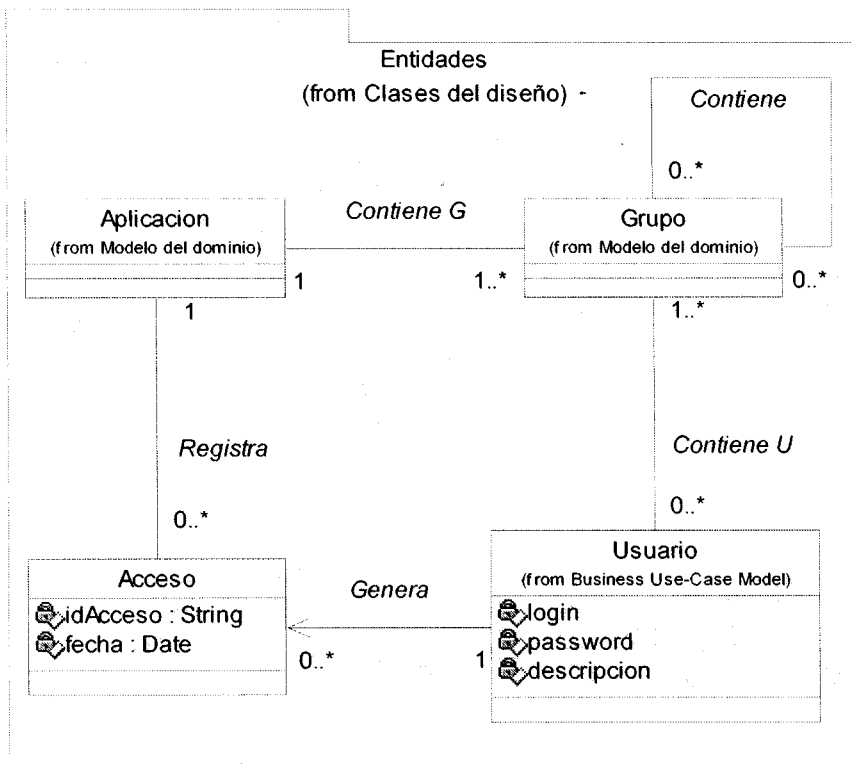


Figura 8. Diagrama de clases del paquete Entidades

#### 4.2.1.6 Paquete Procesamiento.

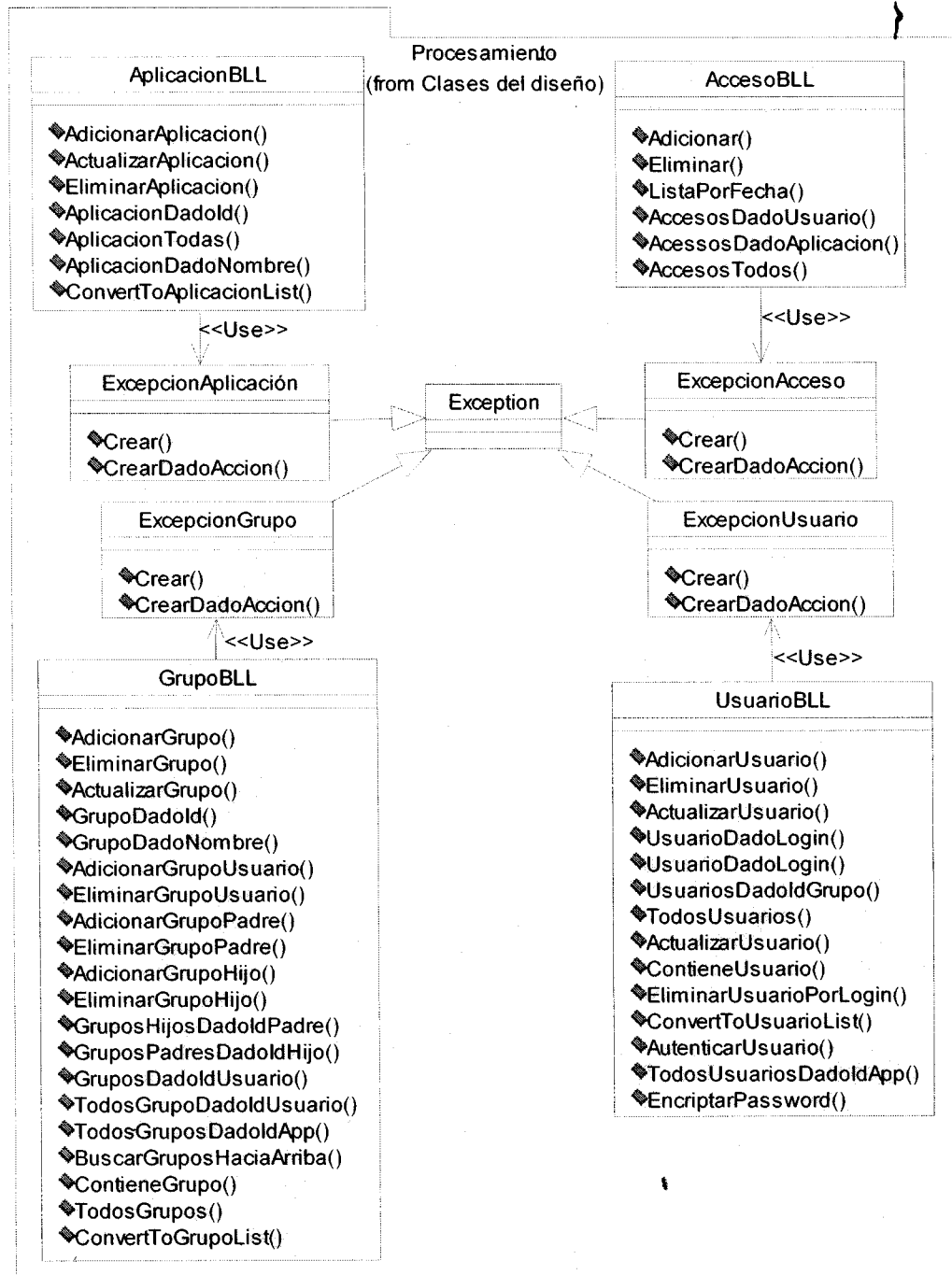


Figura 9. Diagrama de clases del paquete Procesamiento.

#### 4.2.1.7 Paquete Presentación (Subpaquete RMI)

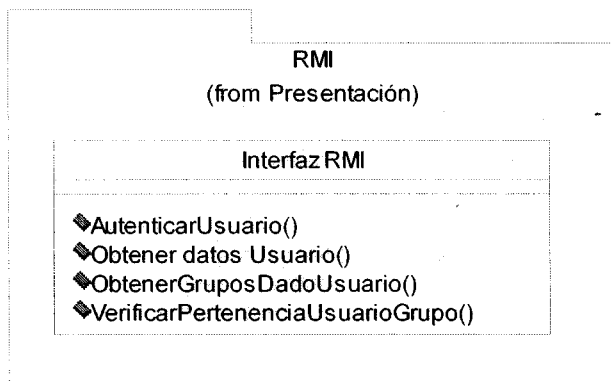


Figura 10. Diagrama de clases del paquete Presentación (Subpaquete RMI)

#### 4.2.1.8 Paquete Presentación (Subpaquete Web Services)

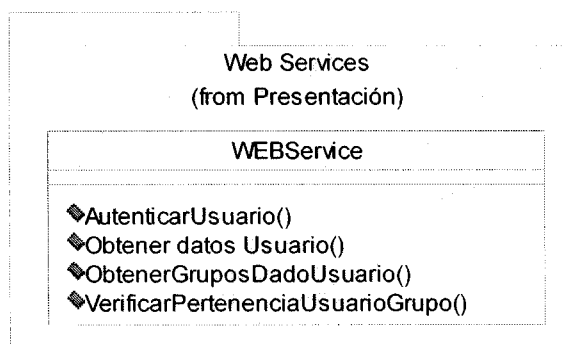
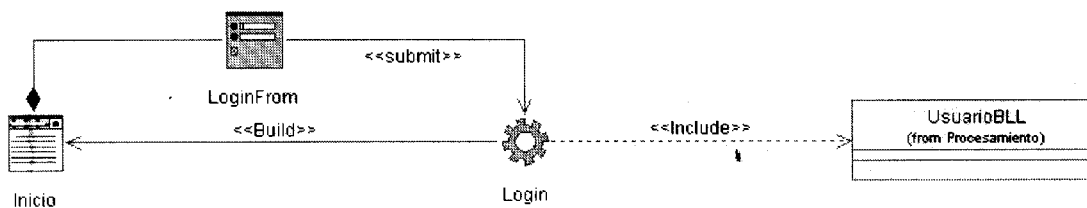
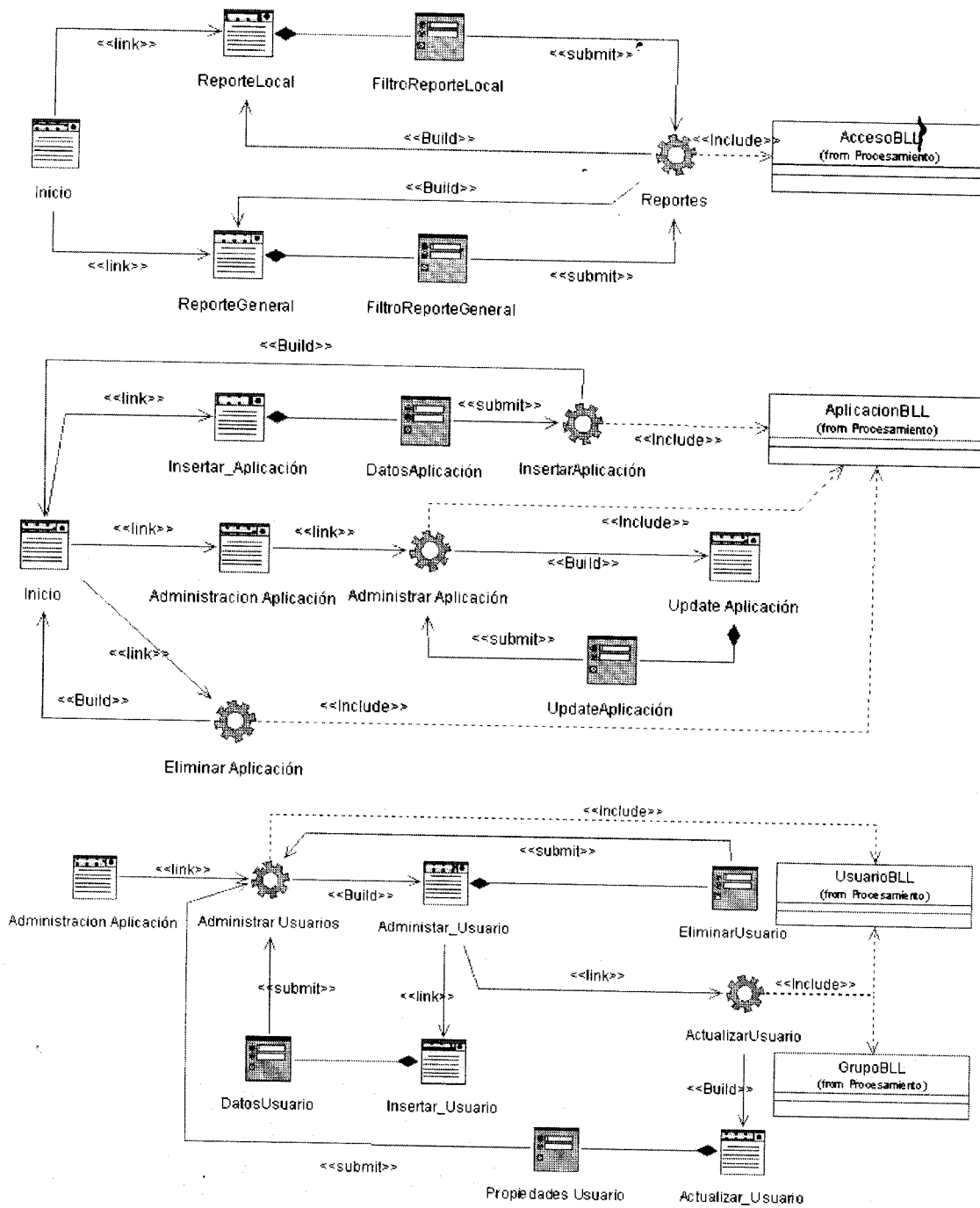


Figura 11. Diagrama de clases del paquete Presentación (Subpaquete Web Services)

#### 4.2.1.9 Paquete Presentación (Subpaquete Configuración Web)







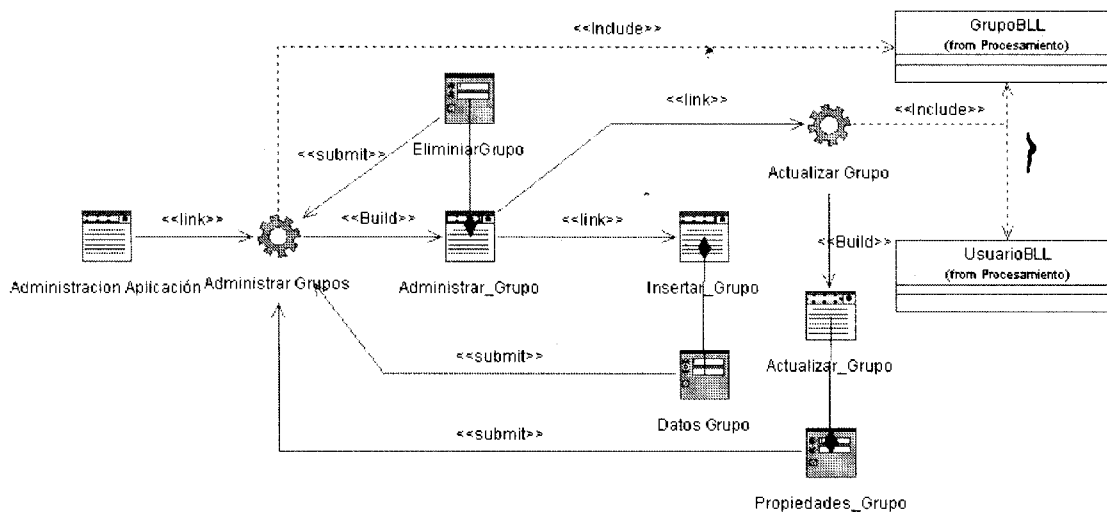


Figura 12. Diagrama de clases del paquete Presentación (Subpaquete Configuración Web)

### 4.3 Principios de diseño

El diseño ha sido elaborado pensando en los usuarios finales, de forma que sea una interfaz amigable e intuitiva. Se ha mantenido un diseño consistente en todas las páginas, para lograr que el usuario se sienta cómodo y logre acostumbrarse rápidamente a la aplicación.

#### 4.3.1 Interfaz de usuario

Todas las páginas Web tienen una estructura similar, contienen una Identificación del sistema, además de opciones comunes a todos los usuarios. (Figura 13, A Estándar de Interfaz de usuario)

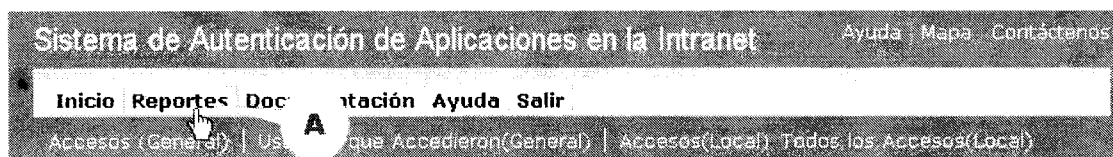


Figura 13. Interfaz de usuario.

Los colores utilizados son en su mayoría los tonos de gris, azul, y negro. Se ha utilizado azul porque es un color relajante. El gris es un color neutral, provoca la sensación de estabilidad y orden, razón por la cual se utiliza como fondo en la mayoría de aplicaciones para Windows y otros sistemas operativos.

Se utiliza el rojo para resaltar errores de campos requeridos (Figura 14, A), con formato incorrecto (Figura 14, B), o mensajes de operaciones no válidas.

Nuevo Usuario

Nombre:  \*

Seleccione el(los) grupo(s) al(a los) cual(es) pertenece:

Administradores

Invitados

Usuarios

Descripción:

Contraseña:  \*

Confirmar Contraseña:  \*

Tipo:  Local  Directorio

ERROR

- Tiene que completar los campos marcados con \* para realizar la operación.
- La contraseña tiene que tener más de 8 caracteres y debe contener números y caracteres especiales.

**Figura 14. Uso del color rojo para mensajes de error.**

En general se realizan múltiples operaciones en cada página, de forma que el usuario no tenga que moverse tanto dentro de la aplicación, para completar una operación. Por ejemplo, se puede hacer la inserción, actualización y eliminación en las páginas donde se muestran listados.







#### 4.3.2 Formato de salida de los reportes

El sistema brinda los reportes en forma de tabla, se da la posibilidad de ordenar por campo (*Figura 15, A*), para facilitar la búsqueda de información, también se da la posibilidad de filtrar los resultados atendiendo a rangos de fecha. Las filas de los reportes son de colores alternos, para facilitar la lectura, se han utilizado colores grises claros, resaltando los nombres de los campos con fondo azul y letras blancas.


Los reportes permiten paginado, de forma que por cada búsqueda sean visibles solamente un número limitado de registros, permitiendo moverse adelante y hacia atrás, gracias a pequeñas flechas en la parte inferior de las tablas (*Figura 15, B*).

**Reporte de Accesos de SGSC**

Desde  Hasta  Reporte

Acceso	Fecha y hora  	Usuario  
11	01/01/2006 08:30:00 	lomar
12	01/01/2006 10:03:54	irotegar
13	01/01/2006 13:43:02	<u>lomar</u> 
14	02/01/2006 08:45:33	abdanys
15	02/01/2006 09:32:40	paulp
16	02/01/2006 14:20:13	chony
17	02/01/2006 17:01:11	chony
18	02/01/2006 18:30:44	vparrado
19	02/01/2006 20:31:00	biasmey
20	02/01/2006 20:31:18	ycabrerago

Mostrando (10) accesos de un total de (94)

 << < > >> Pagina 2 de 3

**Figura 15. Ejemplo de reporte**

### 4.3.3 Ayuda

El sistema cuenta con un manual de usuario que esta disponible desde cualquier página del mismo, en dicho manual, se explica su funcionamiento, requerimientos, y se muestra una lista con preguntas frecuentes y sus respuestas.

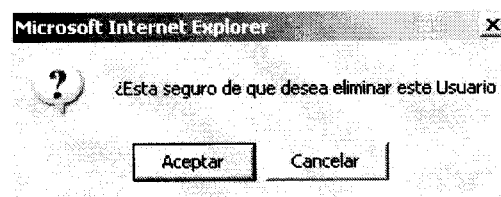
## 4.4 Tratamiento de errores

Para prevenir errores por parte del usuario, sólo se le brindan las opciones mínimas necesarias, a la hora de efectuar cualquier operación, por ejemplo, se deshabilitan ciertos botones si el usuario no tiene que utilizarlos en ese momento.

Mediante una combinación de validación en el lado del cliente y en el lado del servidor, se garantiza que los datos suministrados por los usuarios, se almacenen íntegros y no existan inconsistencias. Se verifican los campos obligatorios y que cumplan con los requisitos necesarios. (Figura 14, A,B)

Se manipulan las excepciones que podrían ocurrir en tiempo de corrida de la aplicación, generándose en un fichero XML un registro de estos, para una eventual revisión por parte del administrador.

Algunos errores serán generados por funciones JavaScript para evitar la ejecución de la página en vano. Este es el caso de los formularios de inserción/actualización, y las eliminaciones. Por último, se utilizan errores en forma de mensajes de texto de color rojo en la misma página donde se ejecutó la acción, de forma que el usuario pueda corregir más fácilmente y continuar. Se utilizan mensajes de confirmación, para acciones que son irreversibles como es el caso de las eliminaciones (*Fig. 16*).



**Figura 16. Utilización de mensajes de confirmación**

## 4.5 Diseño de la base de datos

### 4.5.1 Modelo lógico de datos

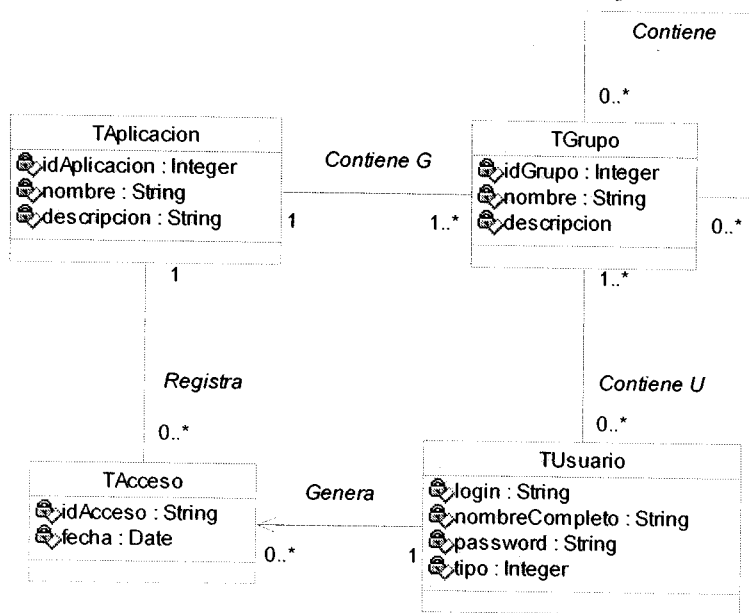


Figura 17. Modelo lógico de datos

#### 4.5.2 Modelo físico de datos

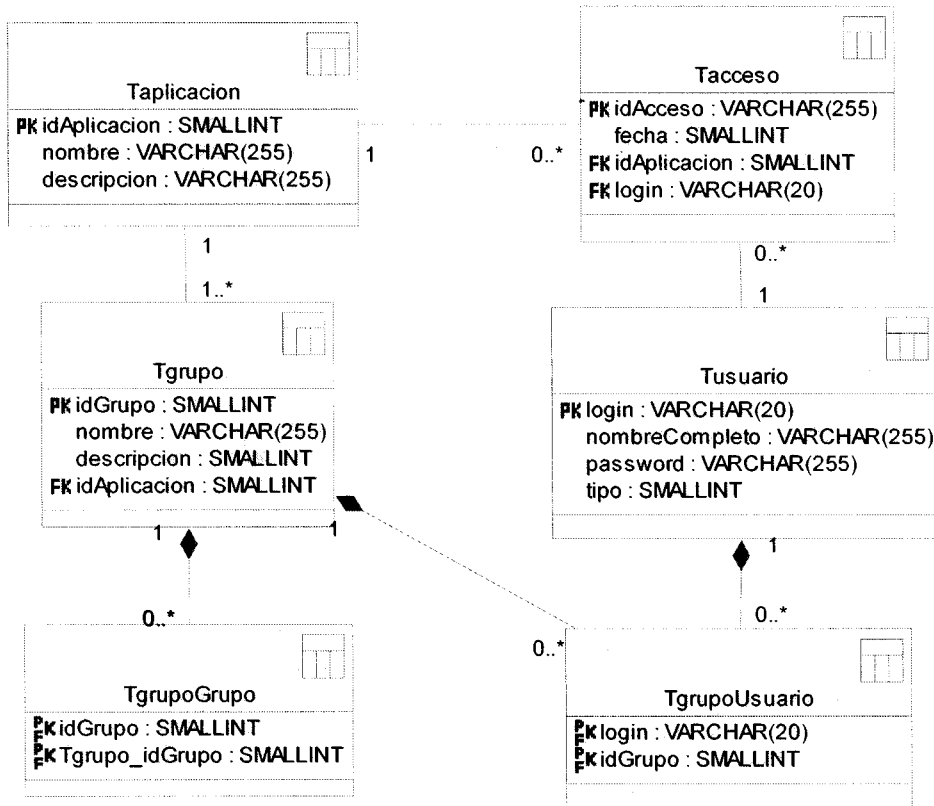


Figura 18. Modelo físico de datos

#### 4.6 Diagrama de despliegue

Un diagrama de despliegue muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en ellos.

El sistema se ha construido siguiendo la arquitectura de tres capas (Ver Anexo 2), o sea, la separación de los componentes en Acceso a Datos, Lógica de Negocio, Cliente. Dentro de la capa de acceso a datos, se han separado físicamente la librería para acceder a la base de datos y la librería para acceder a los Servicios de Directorio, esto es con el fin de lograr una independencia entre ambas y poder reemplazar una u otra si cambia el entorno de la aplicación.

Existe un servidor que contiene la base de datos. El servidor Web reside en un servidor independiente, contiene al Apache Tomcat, tiene instalada la plataforma J2EE. , Los clientes son

los distintos Usuarios desde sus terminales accediendo al sistema a través de HTTP, mediante un navegador Web y las Aplicaciones Clientes que se encuentran en servidores de aplicaciones que acceden a través de HTTPS o RMI al servidor. Además en un Controlador de Dominio al cual accede el Sistema de autenticación de Aplicaciones para obtener información de los usuarios que se encuentran en un Directorio de Servicios.

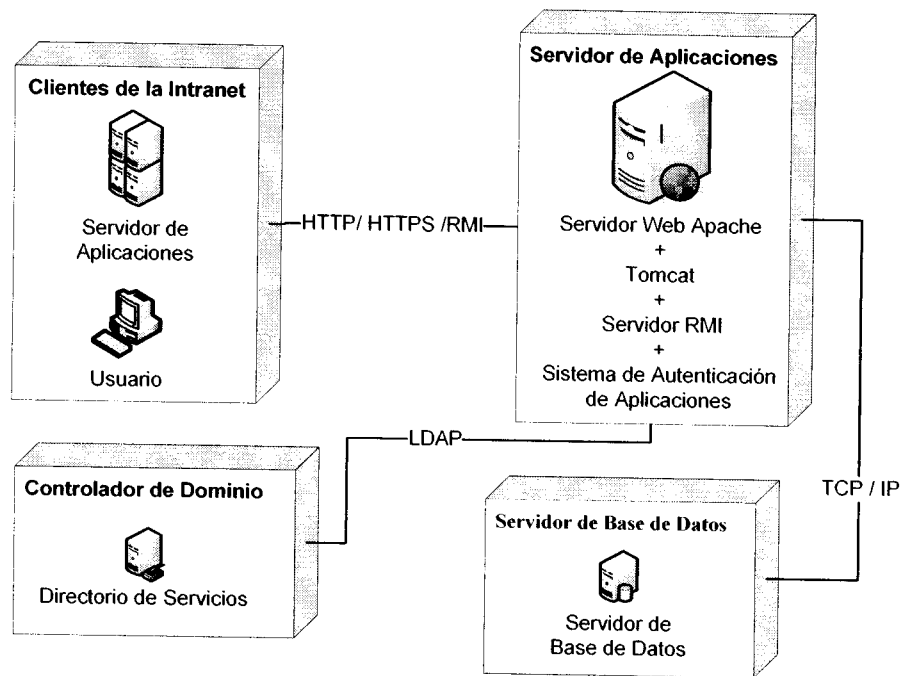


Figura 19. Diagrama de despliegue

## 4.7 Conclusiones

En el presente capítulo se mostraron los resultados de la etapa de diseño del sistema se obtuvo el diagrama de clases donde se representaron las clases y sus asociaciones. Se plantean principios de diseño que ayudan a un mejor diseño del Sistema. Se presentó la concepción del tratamiento de errores, y el sistema de ayuda. Además se realizó el diseño de la base de datos y el diagrama de despliegue. Todos estos elementos obtenidos brindan una idea mucho más clara de la estructura del sistema e influyen en el logro de una mejor herramienta.

## Capítulo 5 Estudio de factibilidad

### 5.1 Introducción

Es importante evaluar la factibilidad de un proyecto antes de su elaboración, para conocer si es conveniente llevarlo a cabo. La viabilidad y el análisis de riesgo están relacionados de muchas maneras. Si el riesgo del proyecto es alto, la viabilidad de producir software de calidad se reduce. En el presente capítulo se hace un estudio de factibilidad, beneficios y costo del sistema propuesto, utilizando estimación basada en puntos de casos de uso que es un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de pesos a determinados factores que lo afectan para contabilizar el tiempo total estimado para ese proyecto a partir de esos factores.

### 5.2 Planificación basada en casos de uso

#### 5.2.1 Paso 1. Cálculo de los Puntos de casos de uso Desajustados.

$$UUCP = UAW + UUCW$$

Donde:

*UUCP* : Puntos de Casos de Uso Sin Ajustar

*UAW* : Factor de Peso de los Actores sin Ajustar

*UUCW* : Factor de Peso de los Casos de Uso sin Ajustar

Tipo de actor	Descripción	Factor de peso	Actores	UAW
Simple	Sistema con sistema a través de interfaz de programación.	1	1	1
Medio	Sistema con sistema mediante protocolo de interfaz basada en texto.	2	1	2
Complejo	Persona que interactúa con el sistema mediante interfaz gráfica.	3	2	6

Tabla 8. Factor de Peso de los Actores sin Ajustar.



$$UAW = \sum cant \text{ actores} * peso$$

$$UAW = 1*1 + 2*1 + 3*2 = 9$$

Tipo de CU	Descripción	Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	4	20
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	2	20
Complejo	El caso de uso tiene más de 8 transacciones.	15	-	-

Tabla 9. Factor de Peso de los Casos de Uso sin Ajustar.

$$UUCW = \sum cant CU * Peso$$

$$UUCW = 4 * 5 + 10 * 2$$

$$UUCW = 20 + 20$$

$$UUCW = 40$$

$$UUCP = UAW + UUCW$$

$$UUCP = 9 + 40$$

$$UUCP = 49$$

**Paso 2. Cálculo de los Puntos de casos de uso ajustados.**

$$UCP = UUCP * TCF * EF$$

Donde:

*UCP* : Factor de Complejidad Técnica

*UUCP* : Puntos de Casos de Uso Sin Ajustar

*TCF* : Factor de Complejidad Técnica

*EF* : Factor de Ambiente

El factor de complejidad técnica (TCF) se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada factor se cuantifica en un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

Factor	Descripción	Peso	Valor	Total
--------	-------------	------	-------	-------

			asignado	
T1	Sistema distribuido	2	1	2
T2	Tiempo de respuesta	1	5	5
T3	Eficiencia del usuario final	1	4	4
T4	Funcionamiento Interno complejo	1	3	3
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0,5	2	1
T7	Facilidad de uso	0,5	4	2
T8	Portabilidad	2	5	10
T9	Facilidad de cambio	1	4	4
T10	Concurrencia	1	5	5
T11	Incluye objetivos especiales de seguridad	1	5	5
T12	Provee acceso directo a terceras partes	1	0	0
T13	Se requieren facilidades especiales de entrenamiento de usuarios	1	3	3

**Tabla 10. Factores de Complejidad Técnica.**

$$TCF = 0.6 + 0.01 * \sum ( peso * valor \text{ asignado } )$$

$$TCF = 0.6 + 0.01 * 48)$$

$$TCF = 1.08$$

El factor de ambiente (EF) está relacionado con las habilidades y entrenamiento del grupo de desarrollo que realiza el sistema. Cada factor se cuantifica con un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

Factor	Descripción	Peso	Valor asignado	Total
E1	Familiaridad con el modelo de proyecto utilizado	1,5	3	4,5
E2	Experiencia en la aplicación	0,5	3	1,5
E3	Experiencia en la orientación a objetivos.	1	4	4
E4	Capacidad del analista líder.	0,5	4	2
E5	Motivación.	1	5	5
E6	Estabilidad de requerimientos	2	4	8
E7	Personal Part-Time	-1	5	-5

E8	Dificultad del lenguaje de programación	-1	4	-4
----	---	----	---	----

**Tabla 11. Factores de Ambiente.**

$$EF = 1.4 - 0.03 * \sum ( \text{peso} * \text{valor asignado} )$$

$$EF = 1.4 - 0.03 * (16)$$

$$EF = 0.92$$

$$UCP = UUCP * TCF * EF$$

$$UCP = 49 * 1.08 * 0.92$$

$$UCP = 48.6864$$

### 5.2.2 Estimación de esfuerzo a través de los puntos de casos de uso.

$$E = UCP * CF$$

Donde:

*E* : Esfuerzo Estimado en Horas – Hombres

*UCP* : Puntos de Casos de Uso Ajustados

*CF* : Factor de Conversión

$$E1 - E6 = 2 \quad E7 - E8 = 2$$

Para obtener el factor de conversión (CF) se cuentan cuantos valores de los que afectan el factor ambiente (E1...E6) están por debajo de la media (3), y los que están por arriba de la media para los restantes (E7, E8). Si el total es 2 o menos se utiliza el factor de conversión 20 Horas-Hombre / Punto de Casos de uso. Si el total es 3 o 4 se utiliza el factor de conversión 28 Horas-Hombre / Punto de Casos de uso. Si el total es mayor o igual que 5 se recomienda efectuar cambios en el proyecto ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

En este caso se puede decir que:

$$CF = 20 \text{ Horas-Hombre / Punto de Casos de uso.}$$

$$E = UCP * CF$$

$$E = 48.6864 * 20$$

$$E = 973.428 \text{ Horas – Hombre}$$

#### Paso 4. Calcular esfuerzo de todo el proyecto.

Actividad	Porcentaje	Horas-Hombres
Análisis	10 %	243,432

Diseño	20 %	486,864
Implementación	40 %	973,728
Pruebas	15 %	365,148
Sobrecarga (otras actividades)	15 %	365,148
Total	100 %	2434,32

**Tabla 12. Distribución del Esfuerzo por Flujo de Trabajo.**

$$E = 2434.32 \text{ Horas} - \text{Hombres}$$

$$\text{Cantidad de Horas} - \text{Hombre por mes (CH)} = 240$$

$$E = 3003 / 240 \text{ Meses} - \text{Hombres}$$

$$E = 10.143 \text{ Meses} - \text{Hombres}$$

Si el esfuerzo total es de **2434.32 horas-hombre** y por cada 240 horas yo tengo 1 mes eso daría un  **$E_T = 10.143$  mes-hombre.**

Esto quiere decir que 1 persona puede realizar el sistema analizado en aproximadamente 10

### **Costo del Proyecto**

Se asume como salario básico mensual (SBM) es de \$225.00

$$\text{CHM} = 2 * \text{SBM}$$

$$\text{CHM} = 450.00 \text{ pesos/mes}$$

$$\text{Costo} = \text{SM} * E_T$$

$$\text{Costo} = 225 * 10.143$$

$$\text{Costo total} = \$ 2282.175$$

### **5.3 Beneficios tangibles e intangibles**

Los beneficios obtenidos con el desarrollo del software son fundamentalmente intangibles, debido a que permite garantizar la seguridad de las aplicaciones en la intranet de forma centralizada, como parte del proceso de informatización de la UCI, además de la ventaja que constituye para los desarrolladores, en la implementación y el manteniendo de las aplicaciones, por lo servicios que este brinda, fundamentalmente el ahorro de tiempo.

## **5.4 Análisis de costos y beneficios**

Al desarrollo de todo producto informático va asociado un costo, el justificarlo depende de los beneficios tangibles e intangibles que produce.

La utilización de este nuevo sistema para brindar servicios de autenticación y de esta manera lograr la seguridad de las aplicaciones parte de la idea de concebir la información como un recurso estratégico que debe ser protegido, de forma tal que solo tengan acceso a esta quienes deben. Este sistema del que dispondrá la UCI, le permitirá lograr este objetivo. Los beneficios de este sistema aumentan con el tiempo ya que entre más aplicaciones usen el sistema mas serán los beneficios que obtendrán los usuarios debido a que se podrán desarrollar nuevas estrategias y servicios, relacionando la seguridad entre las aplicaciones que lo usen.

Además, mejora considerablemente el trabajo de los desarrolladores que, con solo utilizar el sistema, garantizan la seguridad.

Es factible desarrollar esta herramienta ya que los beneficios obtenidos superan los costos y se van incrementando con el tiempo.

## **5.5 Conclusiones**

En este capítulo se efectuó el estudio de factibilidad correspondiente al desarrollo del proyecto. Este permitió llegar a la conclusión que resultará factible implementar la aplicación. La herramienta propuesta trae consigo una serie de beneficios sobre todo intangibles para el centro, pero no menos necesarios e importantes, porque va a contribuir a mejorar su funcionamiento, lo que indica que es factible implementar la herramienta propuesta.

## Conclusiones

Con el desarrollo de este trabajo se obtienen las siguientes conclusiones:

- A través del estudio que se llevo a cabo se detectaron las deficiencias en la manera en que eran desarrollados los módulos de seguridad de los sistemas en la intranet, lo que constituyó el punto de partida de esta investigación.
- Nuestro Sistema da Solución a las deficiencias, permitiendo una autenticación Centralizada y eficiente, apoyado en por su interfaz Web.
- Se cumple con el objetivo general del trabajo lográndose la conclusión los flujos de trabajo especificados en la metodología utilizada.
- Se ha demostrado la eficacia de los lenguajes y tecnologías utilizadas para el desarrollo del sistema fundamentalmente el uso de hibernate.
- La solución propuesta utilizando el Modelo del Dominio ha sido acertada, los requerimientos soportan al sistema y los casos de uso satisfacen las necesidades funcionales.
- El sistema esta basado en una arquitectura de 3 capas (Presentación, Negocio y Datos).
- Se han seguido los principios básicos de diseño descritos para el desarrollo del sistema.
- Se logra una seguridad y protección de los datos consecuente con el nivel de seguridad requerido.
- Es factible implementar el sistema de autenticación de aplicaciones, ya que de acuerdo a los beneficios que reporta, su utilización es más significativa en comparación a los costos de su desarrollo.
- Se obtiene la propuesta de una aplicación que efectivamente reduce el tiempo de desarrollo de la aplicación y su mantenimiento.
- El valor social del sistema se expresa en la contribución a mejorar las condiciones de trabajo, de los especialistas del área, promoviendo la programación distribuida, con la cual se reduce de forma considerable el tiempo de estudio e implementación necesarios.

Por todo lo anterior se concluye que los objetivos propuestos para el presente proyecto han sido cumplidos satisfactoriamente. Se incluyen una serie de recomendaciones que deben tenerse en cuenta para el trabajo futuro.

## Recomendaciones

Los objetivos generales de este trabajo han sido logrados, pero a lo largo de su desarrollo, han ido surgiendo ideas que podrían implementarse en un futuro, de forma que se logre una la aplicación más útil y efectiva, para lo cual se recomienda:

- Integrar más las aplicaciones de la intranet con el sistema, para beneficiarse de la información de estos y obtener ideas de sus necesidades y patrones de uso.
- Extender aún más el sistema de manera que pueda ser utilizado no sólo en la Universidad, sino en cualquier empresa que requiera de un servicio de autenticación, convirtiéndose en una fuente de ingresos.
- Continuar el desarrollo de este sistema, adicionándole nuevas funcionalidades, como: la posibilidad de tenerlo como un módulo incorporable a otros sistemas stand-alone, así como mejoras en la característica SSO, adecuándolo más a las demandas de la creciente y dinámica intranet de la Universidad y haciéndolo más útil y provechoso.

## Referencias bibliográficas

1. JA-SIG. *JA-SIG Java Architectures*. 2006 [cited 2006; Available from: <http://www.ja-sig.org>].
2. DesarrolloWeb.com. *Qué es cada tecnología*. 2006 [cited 2006; Available from: <http://www.desarrolloweb.com/manuales/15/>].
3. AB, M. *Mysql*. 2005 [cited 2005; Available from: <http://www.mysql.com/>].
4. Booch, G., Rumbaugh, J., Jacobson, *El Lenguaje Unificado de Modelado. Manual de usuario*. Addison-Wesley ed. B.J. Rumbaugh. 1999.
5. Martínez, G.M. *Ingeniería de SoftwareUML*. 2005 [cited; Available from: <http://www.monografias.com/trabajos5/insof/insof.shtml>].
6. Jacobson, I.B., G. y Rumbaugh, J, *El Proceso Unificado de Desarrollo de software*. Addison-Wesley., ed. J.B. Rumbaugh. 2000.
7. Cafésoft LLC. *Cams*. 1996-2006 [cited 2006; Sitio oficial CAMS]. Available from: <http://www.cafesoft.com/>.



## Bibliografía

- 1.- Booch, G., Análisis y Diseño Orientado a Objetos. 2da ed. Addison-Wesley 1996.
- 2.- Booch, G., Rumbaugh, J., Jacobson, El Lenguaje Unificado de Modelado. Manual de usuario. Addison-Wesley ed. B.J. Rumbaugh. 1999.
- 3.- Booch, G., Rumbaugh, J., Jacobson, El Lenguaje Unificado de Modelado. Manual de Referencia. Addison -Wesley, ed. B.J. Rumbaugh. 2000.
- 4.- Cafésoft LLC. Cams. 1996-2006. Available from: <http://www.cafesoft.com/>.
- 5.- Deepak Alur, J.C., y Dan Malks., Core J2EE Patterns. Second Edition ed, ed. P. Hall. 2003.
- 6.- González, C.S., ONESS. 2004, UNIVERSIDADE DA CORUÑA. p. 138.
- 7.- Hoeller, R.J.y.J., Expert One-on-One J2EE Development without EJB. Wrox. 2004.
- 8.- IBM. Unified Modeling Language. 2006 [cited; Available from: <http://www-306.ibm.com/software/rational/uml/>].
- 9.- Inderjeet Singh, B.S., y Mark Johnson., Designing Enterprise Applications with the J2EE Platform. Second Edition. ed. Addison Wesley. 2002.
- 10.-Martin, M.A.G., Sistema de Control de Acceso, in Ingenieria en informática. 2005, UCI: La Habana. p. 118.
- 11.-Microsystems, S. Java Code Conventions. 2006 [cited; Available from: <http://java.sun.com/docs/codeconv/index.html>].
- 12.-Nilet María Soto López, Y.S.R., HUBBLE. 2004, CUJAE: La Habana. p. 112.
- 13.-O'Regan, G. Introduction to Aspect-Oriented Programming. 2004 [cited 2006; Available from: <http://www.onjava.com/pub/a/onjava/2004/01/14/aop.html>].
- 14.-Pessoa, J. XV Simpósio Brasileiro de Banco de Dados - SBBD'2000. 2000 [cited; Available from: <http://www.lbd.dcc.ufmg.br/bdbcomp/servlet/Evento?id=21>].
- 15.-University, Y. Central Authentication Service (CAS). 2006 [cited; Available from: <http://www.yale.edu/tp/auth/>].

## Glosario de términos

- (1). **Web Services:** Aplicación que realiza un función y que puede formar parte de otros servicios para formar un servicio más completo. La comunicación hacia y desde el Webservice se realiza con XML. Permite una llamada a una funcionalidad localizada en un servidor remoto.
- (2). **XML:** Es el acrónimo de extensible Markup Language (lenguaje de marcado extensible) desarrollado por el World Wide Web Consortium (W3C).
- (3). **Autenticación:** Se define como la capacidad de identificar al usuario de un sistema de información.
- (4). **Autorización:** (proceso por el cual la red de datos autoriza al usuario identificado a acceder a determinados recursos de la misma).
- (5). **Interfaz de Autenticación:** En el caso de nuestro sistema no se refiere al término interfaz en el sentido de algo visual, sino de un servicio al que tendrán acceso otras aplicaciones para manipular el proceso de autenticación.
- (6). **Credenciales:** Se refiere al par usuario - contraseña. Es utilizado comúnmente en cualquier aplicación que requiera autenticar.
- (7). **Single Sign-On (SSO):** Los usuarios se identifican en el sitio solo una vez, y le es dado acceso a una o varias aplicaciones en un dominio o a través de varios dominios.
- (8). **Hash o algoritmo Hash:** Se refiere a una función o método para generar claves o llaves que representen de manera unívoca a un registro, etc. La función hash, también se utiliza para los procesos de encriptación y decriptación (utilizadas para autenticar).
- (9). **Hashing:** Proceso por el cual se transforma una cadena de caracteres en un valor de un largo pre-establecido o clave de la cadena original.
- (10). **HTTPS:** Versión segura del protocolo HTTP. utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para cualquier tipo de servicio que requiera el envío de datos personales o contraseñas.
- (11). **Hibernate:** Es un mapeador objeto-relacional que proporciona un puente entre la programación orientada a objetos y los sistemas de gestión de bases de datos relacionales.
- (12). **Spring Framework:** Entorno para el desarrollo de aplicaciones fomentando el patrón inversión de control y la integración entre tecnologías.
- (13). **JSP (Java Server Pages)** para el desarrollo del interfaz.

## Anexo 1

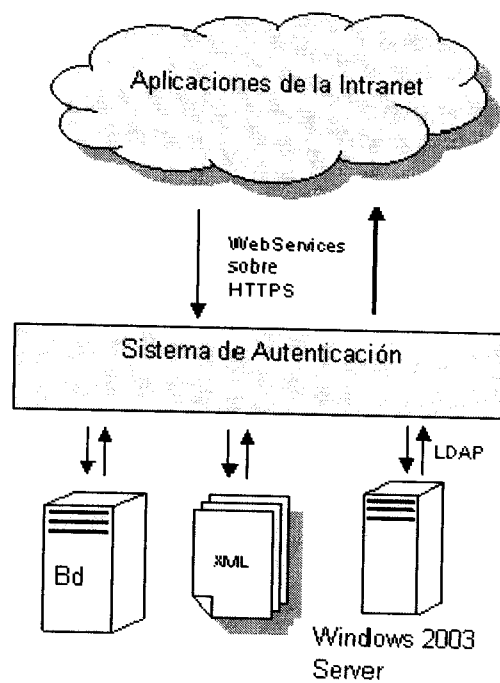


Fig. Esquema general de la aplicación.

## Anexo 2

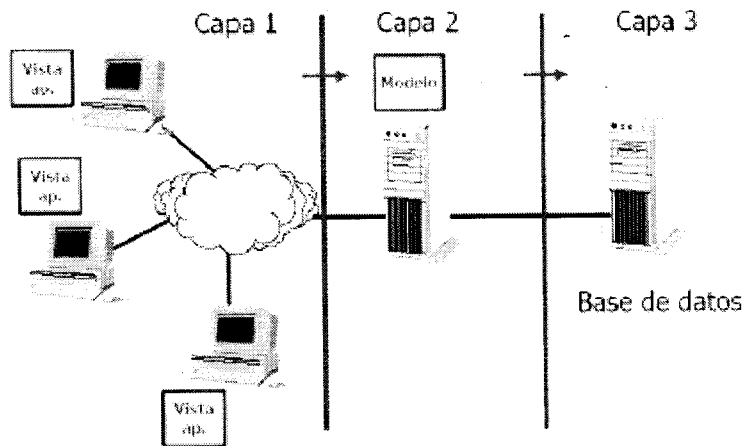


Fig. Arquitectura en tres capas