Universidad de las Ciencias Informáticas

FACULTAD 9

Título: Desarrollo del Componente de Seguridad para el subsistema Mondrian perteneciente al sistema Pentaho.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor

Cinolkis Cobas Hernández

Tutor: Tte. Ing. Alejandro Abiague Nápoles

Co-tutor: Tte. Ing. Dayron Jesús Barrueco Barreto

Ciudad de La Habana, 2010

"Año 52 de la Revolución"

"...el futuro de nuestra Patria, tiene que ser, necesariamente un futuro de hombre de ciencias..."

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

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

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

_____

Cinolkis Cobas Hernández
Autor

_____ _____

Tte. Ing. Alejandro Abiague Nápoles
Tutor

Tte. Ing. Dayron Jesús Barrueco Barreto
Co-Tutor

DATOS DE CONTACTO

Nombre y Apellidos: Alejandro Abiague Nápoles.
Edad: 25 años.
Ciudadanía: cubano.
Institución: Universidad de las Ciencias Informáticas.
Título: Ingeniero en Ciencias Informática.
Categoría Docente:
e-mail: aabiague@uci.cu

Graduado en la Universidad de las Ciencias Informáticas, 2 años de experiencia.

Nombre y Apellidos: Dayron Jesús Barrueco Barreto.
Edad: 25 años.
Ciudadanía: cubano.
Institución: Universidad de las Ciencias Informáticas.
Título: Ingeniero en Ciencias Informática.
Categoría Docente:
e-mail: djbarrueco@uci.cu

Graduado en la Universidad de las Ciencias Informáticas, 2 años de experiencia.

AGRADECIMIENTOS

A mis padres por darme la fuerza y el impulso, por su confianza en todo momento, por acompañarme aún estando lejos, por creer en mi desde que inicié mis estudios.

A mis hermanos por ver en mí un ejemplo digno de seguir, por su cariño y respeto…..

A toda mi familia por siempre estar ahí, especialmente a mi tía Magalis, por ser mi otra madre exigente y preocupada.

A Isabel por su paciencia, por su apoyo, por creer en mí en todo momento.

A la Revolución Cubana y a nuestro Comandante en Jefe por ser el autor intelectual de este gran sueño que es la Universidad de las Ciencias Informáticas (UCI) y hacernos partícipe de su realidad.

A Alejandro Abiague Napoles, a Dayron Jesús Barrueco Barreto y a Velmour Muñoz Casals, mas que tutores, amigos, por su ejemplo, entrega, seriedad y apoyo incondicional, por sus consejos, sus exigencias, su esfuerzo, su responsabilidad.

A todos mis compañeros y compañeras de aula y de la Universidad en estos cinco años de estudio, por compartir tantos momentos buenos y malos, que durarán en nuestras memoria para siempre.

A todos aquellos que de una forma u otra hicieron posible que hoy me encuentre aquí.

A todos, ¡Muchas Gracias!

DEDICATORIA

A toda mi familia, mis padres, hermanos, tías, tíos, primos, abuelos……

A todas mis amistades de la universidad y de toda la vida……

A mi nueva familia santiaguera……

Y a mi primer hijo que hoy es un embrión de cuatro mesecitos……

Resumen

La seguridad de la información es un factor fundamental en cualquier institución, cualquier pérdida, desvío o mala manipulación de los datos ocasionaría daños económicos y sociales irreparables. Se requiere de una buena administración y control de la información, así como una adecuada seguridad acorde a los intereses del país.

En la UCID se personaliza para el análisis de la información, la Plataforma Pentaho BI, por las ventajas que brinda para la toma de decisiones, teniendo en cuenta la soberanía tecnológica abogada por la política de país al ser una herramienta de software libre. La plataforma Pentaho BI está integrada por una serie de módulos. Entre ellos, uno de los principales módulos es el de Análisis, quien suministra a los usuarios un sistema avanzado de análisis de la información, a través del servidor OLAP Mondrian, que gestiona comunicación entre una aplicación de procesamiento analítico en tiempo real y la base de datos. A pesar de las ventajas que brinda este módulo para la toma de decisiones, este no cumple con los requisitos especificados por las FAR respecto a la seguridad y compartimentación de la información, por lo que surge como necesidad, el desarrollo de un componente de seguridad para el subsistema Mondrian.

Palabras Claves: componente, seguridad, Mondrian, Pentaho.

ÍNDICE DE TABLAS Y FIGURAS.

ÍNDICE

INTRODUCCIÓN

La Revolución Cubana con más de 50 años de proceso renovador ha desarrollado toda una serie de estrategias de trabajo para elevar el nivel cultural y profesional de todo el país, que en estos momentos se encuentra inmerso en un proceso revolucionario social, denominado Batallas de Ideas. Este proceso va dirigido a elevar el nivel cultural y científico de esta sociedad a escalas nunca antes pensadas para un país del tercer mundo. Como parte de la Batalla de Ideas se concibió estratégicamente la creación de la Universidad de las Ciencias Informáticas (UCI) con el fin de contribuir a la informatización del país como centro rector de la industria cubana de producción de software. La UCI se ha insertado en este vasto y competitivo mercado, evidenciando como elemento fundamental la formación de profesionales comprometidos con su país y altamente calificados en la rama de la Informática, producción de software y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación.

En la UCI actualmente se desarrollan grandes proyectos, con un gran compromiso con la Revolución. La Unidad de Compatibilización, Integración y Desarrollo de Productos Informáticos para la Defensa (UCID), perteneciente al MINFAR, se encuentra ubicada en la infraestructura productiva de la Universidad, bajo los principios de compatibilizar soluciones informáticas de gran interés para el país y el desarrollo de software con calidad.

En la Unidad de Compatibilización, Integración y Desarrollo de Productos Informáticos para la Defensa actualmente se desarrollan gran parte de los sistemas informáticos existentes hoy en las FAR. En la mayoría de ellos se genera el intercambio y almacenamiento de grandes volúmenes de datos, haciendo que los usuarios de estos sistemas no puedan captar tanta información, dificultando la toma de decisiones, tornándose más difícil cuando se trata de sectores especializados que a menudo se ven perdidos dentro de esta cantidad de información.

El avance tecnológico de la sociedad actual en interés de ayudar a contrarrestar este fenómeno, ha generado necesidades y exigencias que abogan por una nueva etapa de la gestión de la información. Una potente herramienta para la Inteligencia de Negocios, personalizada para la gestión de la información en los sistemas de las FAR y de la UCID, es la plataforma Pentaho, que está integrada por una series de

módulos que hacen que sea una solución muy flexible para cubrir una amplia gama de necesidades en el análisis de datos y de informes empresariales, entre sus principales módulos se pueden encontrar:

- Módulo de Reportes, herramienta de reportes flexibles y con clase empresarial, de escritorios o basados en Web. La herramienta de reportes PENTAHO permite comenzar desde sencillos reportes iníciales hasta formar complejos reportes ajustados a las necesidades de negocio.

- Módulo de Integración de Datos, es la herramienta que permite la integración de ambientes y datos de diferentes sistemas de una empresa en un almacén de datos.

- Módulo de Minería de Datos, provee un completo conjunto de algoritmos que automatizan los procesos de transformación de datos a la forma en que la minería de datos puede explotarlos. Los resultados pueden ser visualizados en modo gráfico ya sea agrupado, segmentado, de árbol de decisión, bosque aleatorio, redes neurales y componentes de análisis. Utiliza filtros para la discreción, normalización, re-muestreo, selección y trasformación de atributos. Maneja clasificadores suministrando modelos para la predicción nominal o cantidades numéricas.(1)

- Módulo de Análisis suministra a los usuarios un sistema avanzado de análisis de información a través del servidor OLAP (On Line Analytical Processing) de software libre Mondrian escrito en Java que permite analizar grandes cantidades de datos almacenados en bases de datos SQL de una forma interactiva sin necesidad de escribir las sentencias que serían necesarias para ello en SQL. (2)

Muchas son las ventajas que brinda el módulo de Análisis de la Plataforma Pentaho para el análisis de datos y para la toma de decisiones, pero no cumple con los requisitos de seguridad especificados por las FAR respecto a la seguridad y la compartimentación de la información, cuenta con una definición de roles, pero está definido de manera estática, lo que provoca que no se puede integrar con otras aplicaciones que tenga implementado su propio sistema de seguridad.

A raíz de las condiciones descritas anteriormente se definió como **problema a resolver**: ¿Cómo garantizar la seguridad en el subsistema de análisis de información Mondrian, perteneciente al sistema Pentaho?

Tomando como ***objeto de estudio*** los procesos para el control de la seguridad en los sistemas informáticos, enmarcando como ***campo de acción*** los procesos de control de la seguridad en los sistemas de análisis de información para las FAR. Derivándose como ***objetivo general,*** desarrollar un componente de seguridad para el subsistema de análisis de información Mondrian.

Para darle cumplimiento al objetivo trazado se determinaron como ***objetivos específicos:***

- Analizar los procesos para el control de la seguridad en los sistemas de análisis multidimensional de la información existente.
- Definir las herramientas a utilizar en el desarrollo del componente de seguridad.
- Diseñar un componente que gestione la seguridad del subsistema de análisis de información Mondrian.
- Desarrollar un componente de seguridad como parte del subsistema Mondrian.
- Validar la solución propuesta.

Idea a defender: Con el desarrollo de un componente de seguridad para el subsistema Mondrian se obtendrá un sistema más confiable y seguro, cumpliendo con los requisitos especificados por las FAR respecto a la seguridad y compartimentación de la información hasta nivel de datos, además de ser capaz de integrarse con el sistema de seguridad de la UCID.

1

Capítulo

Fundamentación Teórica

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

La seguridad en entornos digitales es uno de los factores más importantes de cualquier sistema, la autenticación y autorización son dos factores esenciales para garantizar la seguridad. La implementación de un componente que conjuntamente pueda garantizar que estas dos funcionalidades puedan ser auténticas y seguras, lleva un conjunto de investigaciones con el fin de garantizar la calidad del producto final a obtener.

Durante el análisis del presente capítulo, se explica en detalle el proceso de investigación llevado a cabo para la implementación del componente, se describen los principales conceptos asociados al dominio del problema. También se hace un análisis de las tecnologías, metodologías, lenguajes y herramientas a utilizar en el desarrollo del componente.

1.1 Conceptos básicos asociados al problema

1.1.2 Componente

Un componente es una parte no trivial, casi independiente, y reemplazable de un sistema que llena claramente una funcionalidad dentro de un contexto en una arquitectura bien definida. Un componente se conforma y provee la realización física por medio de un conjunto de interfaces. (3)

1.1.3 Análisis Multidimensional

El Análisis Multidimensional no es más que las diferentes perspectivas mediante las cuales logramos analizar la información de una compañía: tiempo, producto, geografía, vendedor, etc. y, tener la posibilidad de cruzar la información, para ver un producto en una geografía determinada y durante un tiempo específico. (4)

1.2 Antecedentes

Como parte de la investigación se realiza un estudio profundo y detallado de los sistemas informáticos de seguridad existentes que pudieran dar solución al problema planteado. Entre las soluciones existentes a nivel Internacional se puede hacer mención al Servicio Central de Autentificación (Central Authentication Service (CAS)) el cual es un sistema de autentificación creado originalmente por la Universidad de Yale,

con el objetivo de proporcionar un medio confiable a las aplicaciones para la autenticación de usuarios. Actualmente ofrece servicios de Single Sign On (SSO), los usuarios se autentican al servidor CAS y solo necesitan hacerlo una sola vez por sesión del navegador. Trabaja sólo con aplicaciones y recursos accedidos vía Web, los accesos son interceptados con la ayuda de un servidor proxy o de un componente instalado en el servidor Web destino. Brinda la posibilidad a las aplicaciones de autenticación por Proxy a terceras capas de abastecedores de servicios que eligen aceptar sus credenciales de Proxy.

Este sistema aunque simplifica los procedimientos que siguen las aplicaciones para la autenticación, no cumple con todos los requerimientos para satisfacer las necesidades de la UCID, pues no cuenta con mecanismos que gestionen el proceso de autorización y no permite el trabajo con roles.

En nuestro país específicamente en la UCI en el año 2008 se realizó un “Sistema de Gestión de Accesos (SGA)”, el cual brinda principalmente, los servicios de autenticación y autorización de usuarios a las disímiles aplicaciones. Soporta los mecanismos necesarios para, a través de él, obtener información de las aplicaciones, funcionalidades, roles, usuarios y grupos de usuarios de la Universidad.

Esta aplicación pudo haber sido una de las soluciones de software a implantar en la Universidad, pero tiene la desventaja de que no contiene un Single Sign On (SSO), por lo tanto, los usuarios tenían que autenticarse nuevamente siempre que se accediera a otra aplicación.

Otra solución desarrollada igualmente en el año 2008 en la UCI para gestionar la seguridad es el “Sistema de Gestión de Sesiones (SGS)” el cual se convierte en una iniciativa viable de protección y control que permite ofrecer servicios de valor añadido con altos niveles de seguridad y confianza a los usuarios.

La implantación de este sistema podría mejorar la seguridad de la red en la universidad, permitiendo a su vez a los usuarios disminuir su trabajo de autenticación y el tiempo que pierden los mismos en este proceso. En la universidad cada uno de los servicios o aplicaciones cuenta con su propio componente de seguridad, lo cual generalmente compromete la seguridad de todo el sistema, dado que el nivel de seguridad de todo un sistema es igual al nivel de seguridad del componente más inseguro que tenga.

Esta es una de las razones por las que no se usa esta solución de software pues no hace un control de roles ni de usuarios, pero si permitiría a los usuarios autenticarse una única vez y hacer un control de sesiones eficiente.

En el año 2009 se desarrolló el “Sistema de Gestión Integral de Seguridad (SIGIS)”, basado en la necesidad de desarrollar un sistema que gestione la seguridad de forma centralizada en un entorno de varias aplicaciones, el cual brinda principalmente, los servicios de autenticación y autorización de usuarios a las disímiles aplicaciones. SIGIS brinda sus servicios a todos los sistemas que se suscriban a él. Para ello gestiona las conexiones a la base de datos, las funcionalidades asociadas y las acciones que realizan las mismas. Una vez registrada esta información procede a la creación de roles a los cuales se les dan los permisos dentro de cada sistema. Luego crea los usuarios con el perfil seleccionado y se le asigna a uno o muchos roles en una o muchas entidades respectivamente.

Esta es una de las soluciones de software para garantizar la seguridad de forma centralizada más completa de la UCI.

1.3 Fundamentación de las Tecnologías, Herramientas y Metodología Aplicada

Con el objetivo de lograr un componente que no solo que solucione el problema existente, sino que además tenga la calidad requerida y que cumpla con las normas tecnológicas establecidas por la dirección de la UCID; se realiza un estudio minucioso sobre las tecnologías, herramientas, metodologías y lenguajes a utilizar en la confección de la propuesta de solución. A continuación se justifica la tecnología escogida para la confección del componente.

1.3.1 Servidor Web

Un Servidor Web es un programa que se ejecuta continuamente en un ordenador (también se emplea el término para referirse al ordenador que lo ejecuta), manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.

1.3.1.1 Apache Tomcat

Es un servidor web con soporte de servlets y páginas Servidoras de Java (JSPs). Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. Tomcat es del mundo del software libre, fácil de instalar, se ejecuta en máquinas más pequeñas y es compatible con las API más recientes de Java. Tomcat ocupa muy poco espacio, teniendo su código binario un tamaño total de apenas un megabyte, de

modo que no es raro que se ejecute tan deprisa. Otra ventaja de Tomcat es que es muy fiable. Innumerables empresas utilizan Tomcat. Y como dijo Linus Torvalds acerca del código libre, "Dado un número suficiente de ojos, todos los errores son irrelevantes". (5)

1.3.2 Base de datos

Una base de datos es un conjunto de datos que pertenecen al mismo contexto, almacenados sistemáticamente para su uso posterior. Utilizar una base de datos permite globalizar y compartir información, eliminar la redundancia e inconsistencia de datos, mejorar los mecanismos de privacidad y seguridad de los mismos, y mantener la integridad en la información.

Para desarrollar un software que gestione la autenticación y autorización de usuarios, es inevitable controlar de alguna forma toda la información relacionada con los usuarios, los roles que juegan y los permisos que tienen sobre las distintas aplicaciones; por lo que es fácil identificar la necesidad de utilizar una base de datos que almacene toda esta información.

1.3.3 Gestor de bases de datos

Los gestores de base de datos son sistemas formados por un conjunto de datos y un paquete de software para la gestión del mismo. Se controla el almacenamiento de datos redundantes. Estos resultan independientes de los programas que los usan, se guardan las relaciones entre los datos junto con éstos y se puede acceder a ellos de diversas formas. (6)

Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad. Entre los gestores de base de datos más comunes se encuentra el Microsoft SQL Server, el MySQL, Oracle y PostgreSQL.

Ventajas que ofrecen los SGBD:

- Facilidad de manejo de grandes volúmenes de información.
- Gran velocidad en muy poco tiempo.

- Independencia del tratamiento de información.
- Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones y consultas.
- No hay duplicidad de información, comprobación de esta en el momento de introducirse.
- Integridad referencial el terminar los registros.

1.3.3.1 PostgreSQL

Es un servidor de base de datos relacional orientada a objetos y es el más utilizado por todos aquellos programadores que realizan aplicaciones cliente servidor, complejas o críticas. Es una alternativa económica a SQL Server, pues su costo es menor y tiene similares prestaciones. Este se puede utilizar sobre cualquier sistema operativo, característica que lo pone por encima de SQL Server y al parejo con MySQL. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.

1.3.4 Framework Hibernate

Es una herramienta de mapeo objeto-relacional para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la POO. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual

de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

1.3.5 Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas, que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. Permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico. Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa. (7)

1.3.5.1 Java

Nace como un lenguaje de programación fácil de utilizar, lo que lo convierte en uno de los lenguajes más elaborados y utilizados para la creación de software. Este permite a los desarrolladores:

- Desarrollar software en un Sistema Operativo y ejecutarlo en prácticamente cualquier otro.
- Crear programas para que funcionen en un navegador web y en servicios web.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados.
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital.

A continuación se describen algunas de sus principales características:

Orientado a Objetos: Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.

Distribuido: Java se ha construido con extensas capacidades de interconexión TCP/IP13. Existen librerías de rutinas para acceder e interactuar con protocolos como http14 y ftp15. La característica de ser distribuido es debido a que proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que puedan ejecutarse en varias máquinas.

Robusto: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores lo antes posible, en el ciclo de desarrollo.

Interpretado: Se traduce el código fuente a un código intermedio denominado ByteCode, que es interpretado por la Maquina Virtual de Java, lo cual permite que se pueda ejecutar en cualquier sistema operativo.

Seguro: No se permite el acceso ilegal a memoria ya que no se trabaja con punteros. El código Java pasa muchas pruebas antes de ejecutarse en una máquina. El código se pasa a través de un verificador de ByteCodes que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal, que falsean punteros, violan derechos de acceso sobre objetos o intentan cambiar el tipo o clase de un objeto.

Dinámico: No conecta todos los módulos que comprende una aplicación hasta el tiempo de ejecución, ya que las librerías nuevas o actualizadas no paralizarán las aplicaciones actuales siempre que mantengan el API anterior.

1.3.6 Herramienta de desarrollo

Eclipse

Eclipse es un Entorno de Desarrollo Integrado (IDE) de código abierto. Permite crear entornos de desarrollo integrados para distintos lenguajes de programación. Es multiplataforma y permite trabajar con varios proyectos a la vez. Dentro de sus principales características están: editor de texto resaltado de sintaxis, compilación en tiempo real, refactorización. Con esta herramienta se desarrolla lo que el proyecto (Fundación Eclipse) llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

1.3.7 Lenguaje de modelado

Uniendo varios conceptos y teorías, se puede conceptualizar un lenguaje de modelado como una estandarización de notaciones y reglas, que permitan graficar un sistema, o parte de él. La elección de un aceptado lenguaje de modelado tiene una alta importancia, pues un buen modelamiento del software influye determinantemente, en lograr una adecuada comunicación entre los desarrolladores y los clientes.

1.3.7.1 Lenguaje Unificado de Modelado (UML)

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (8)

Como todo lenguaje, UML proporciona un vocabulario y unas reglas para permitir una comunicación. Éste particularmente, está compuesto por elementos que no son más que abstracciones que constituyen los bloques básicos de construcción, los cuales pueden unirse mediante relaciones para conformar los diagramas. Facilita la representación gráfica de un sistema, teniendo como objetivo sustancial, brindar un material de apoyo que le permita al lector poder definir diagramas propios, como también entender diagramas ya existentes. Sus principales funciones son visualizar, especificar, construir y documentar cada una de las partes que comprende el desarrollo de software.

UML tiene tres características que lo hacen ideal:

- Sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten.

- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). (9) Esto permite que el modelo y el

código estén actualizados, por lo que siempre se puede mantener la visión en el diseño, de la estructura del proyecto.

- Es completamente independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML, se pueda implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

Producto de todas estas ventajas, UML no solo es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; sino que además, se ha convertido en el estándar ansiado para describir un "plano" de los sistemas informáticos.

1.3.8 Herramientas de modelado

El hecho de usar la notación UML para el intercambio de información de diseño e ideas, requiere de un software que ofrezca todas las herramientas necesarias, para hacer eficientemente este tipo de trabajo. Esta clase de software es a lo que se denomina herramienta de modelado y son usados para capturar, guardar, rechazar e integrar automáticamente información y diseño de documentación, labores difíciles de lograr con un simple procesador de texto.

1.3.8.1 Visual Paradigm

Existen herramientas CASE como el Analise, el Designe, el Rational Rose y el Visual Paradigm que permiten realizar el modelado del desarrollo de los proyectos. Para la realización de este trabajo se ha decidido utilizar Visual Paradigm ya que a diferencia de otras herramientas de este tipo esta es una herramienta multiplataforma que utiliza “UML”: como lenguaje de modelaje.

Visual Paradigm para UML proporciona un ambiente de modelado visual para satisfacer la tecnología del software y necesidades de comunicación. Es muy fácil de usar y presenta un ambiente gráfico agradable para el usuario.

- Permite generar código en lenguajes tan utilizados como: Java, C++, PHP, C #, Delphi, Perl.

- Permiten incorporar los dibujos de Visio® en cualquier diagrama de UML. Muchas herramientas CASE realizan solo diagramas de UML normales, Visual Paradigm posibilita modelar hardware dominio-específico, el software, conectando una red de computadoras los componentes usando sus iconos, más allá de las anotaciones de UML normales.

- Es multiplataforma, Visual Paradigm está disponible en muchas plataformas incluso Windows, Linux, el Mac OS X, etc.

1.3.9 Metodología de desarrollo

Para desarrollar un software se necesita una forma ordenada de trabajo, un proceso que integre y guíe las múltiples facetas del desarrollo y que además, ofrezca criterios para el control y la medición de los productos. A esta clase de procesos se le conoce como metodología de desarrollo.

1.3.9.1 Proceso de Desarrollo y Gestión de Proyectos de Software

Este modelo de desarrollo de software fue definido en la UCID para el desarrollo de los proyectos que se desarrollan en la misma, el cual describe la secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones de software. Lograda con la combinación entre los modelos basados en Componentes, el Iterativo y el Incremental.

El desarrollado de un software mediante este proceso, tiene un ciclo de vida que se descompone en el tiempo en cinco fases secuenciales (**Figura 1**), que son: Inicio, Modelación, Construcción, Explotación Experimental, Despliegue.

Figura 1: Fases del Proceso de Desarrollo de Software.

Las fases del proceso permiten mejorar el planeamiento, ejecución y control del proyecto. La terminación y consecución exitosa de cada fase es indispensable para poder continuar con el proyecto.

Conclusiones Parciales

En este capítulo se expusieron los conceptos más importantes y se realizó un análisis de los sistemas existentes que gestionan la seguridad para garantizar la seguridad del subsistema de análisis de información Mondrian. También se realizó un estudio de las tecnologías, paradigmas y tendencias actuales que permitió escoger las metodologías, lenguajes y herramientas más adecuados que serán utilizadas a lo largo del proceso de desarrollo.

2

Capítulo

Desarrollo de la Solución

CAPÍTULO 2: DESARROLLO DE LA SOLUCIÓN

En este capítulo se hace un levantamiento de los requisitos funcionales y no funcionales del componente desarrollado. También se muestra el diseño de la solución propuesta así como la descripción de las clases del diseño, para lograr una mejor comprensión de la funcionalidad e integralidad del sistema.

2.1 Problemática

Actualmente en la Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa se desarrollan gran parte de los sistemas informáticos existentes hoy en las FAR, dichos sistemas generan el intercambio y almacenamiento de grandes volúmenes de datos, dificultando el análisis de los mismos y la toma de decisiones de los usuarios, tornándose más difícil cuando se trata de sectores especializados. En la actualidad existen potentes herramientas como Oracle BI, Pentaho BI, que facilitan el manejo y análisis de la información. En la UCID se personaliza la Plataforma Pentaho BI, por las ventajas que brinda para la toma de decisiones, teniendo en cuenta la soberanía tecnológica abogada por la política de país al ser una herramienta de software libre. La plataforma Pentaho BI está integrada por una serie de módulos. Entre ellos, uno de los principales módulos es el de Análisis, quien suministra a los usuarios un sistema avanzado de análisis de la información, a través del servidor OLAP(On Line Analytical Processing) escrito en Java Mondrian, que gestiona comunicación entre una aplicación de procesamiento analítico en tiempo real y la base de datos. A pesar de las ventajas que brinda este módulo para la toma de decisiones, este no cumple con los requisitos especificados por las FAR respecto a la seguridad y compartimentación de la información.

2.2 Propuesta de sistema

Teniendo en cuenta la problemática existente y con la tarea de cumplir los objetivos trazados para dar solución al mismo, el componente que se propone debe brindar principalmente los servicios de autenticación y autorización a los usuarios para el acceso al subsistema de análisis de información Mondrian, haciendo uso de los servicios web publicados por el Sistema de Gestión Integral de Seguridad (SIGIS), así como garantizar la compartimentación de la información según el acceso a la misma.

El tratamiento de requisitos es el proceso mediante el cual se especifican y validan los servicios que debe proporcionar el sistema, así como las restricciones sobre las que se deberá operar. Consiste en un proceso iterativo y cooperativo de análisis del problema, documentando los resultados en una variedad de formatos y probando la exactitud del conocimiento adquirido. La importancia de esta fase es esencial puesto que los errores más comunes y más costosos de reparar, así como los que más tiempo consumen se deben a una inadecuada captura de requisitos

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema

2.3 Requerimientos funcionales del Sistema

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

RF1 Autenticar usuario

R1.1 Entrar datos de la autenticación.

R1.1.1 Estos datos son nombre de usuario (obligatorio) y contraseña (obligatoria).

R1.2 Validar datos introducidos usando los servicios del sistema de seguridad SIGIS.

R1.3 Mostrar al usuario la información a la que tiene acceso según el rol o permiso asignado.

R1.4 Cerrar la sesión de un usuario registrado en cualquier momento.

RF2 Compartimentar Información

RF 2.1 Cargar rol.

2.4 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido, confiable.

Usabilidad (USB)

Para usar el sistema no se necesitará ser especialista en la informática sólo es necesario poseer conocimientos mínimos sobre manejo de aplicaciones Web.

Rendimiento (REN)

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos.

Seguridad (SEG)

Autenticación (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Portabilidad (POR)

El sistema debe ser multiplataforma, haciendo énfasis en Linux y Windows.

Políticos culturales (CUL)

El componente solo podrá ser utilizado en territorio cubano y por las entidades autorizadas por el Ministerio de las FAR. El producto no debe contener palabras en otros idiomas. El producto debe respetar los términos empleados normalmente por los especialistas en el tema de la esfera que se automatiza.

Software (SFT)

Para el cliente:

- Navegador Mozilla Firefox.
- Sistema operativo Windows 98 o superior o Linux.

Para el servidor:

- Sistema operativo Windows Advancer Server (2000 o superior) o Linux en cualquiera de sus distribuciones.
- Servidor Apache Tomcat 5.5 o superior.
- Un servidor de base de datos PostgreSQL 8.3 o superior.

Hardware (HDW)

Para el servidor:

- Requerimientos mínimos: Procesador Pentium III a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.

Para el cliente:

- Requerimientos mínimos: Procesador Pentium II a 133Mhz con 128 Mb de memoria RAM.

2.5 Descripción de los Requisitos Funcionales

2.5.1 RF1 Autenticar Usuario.

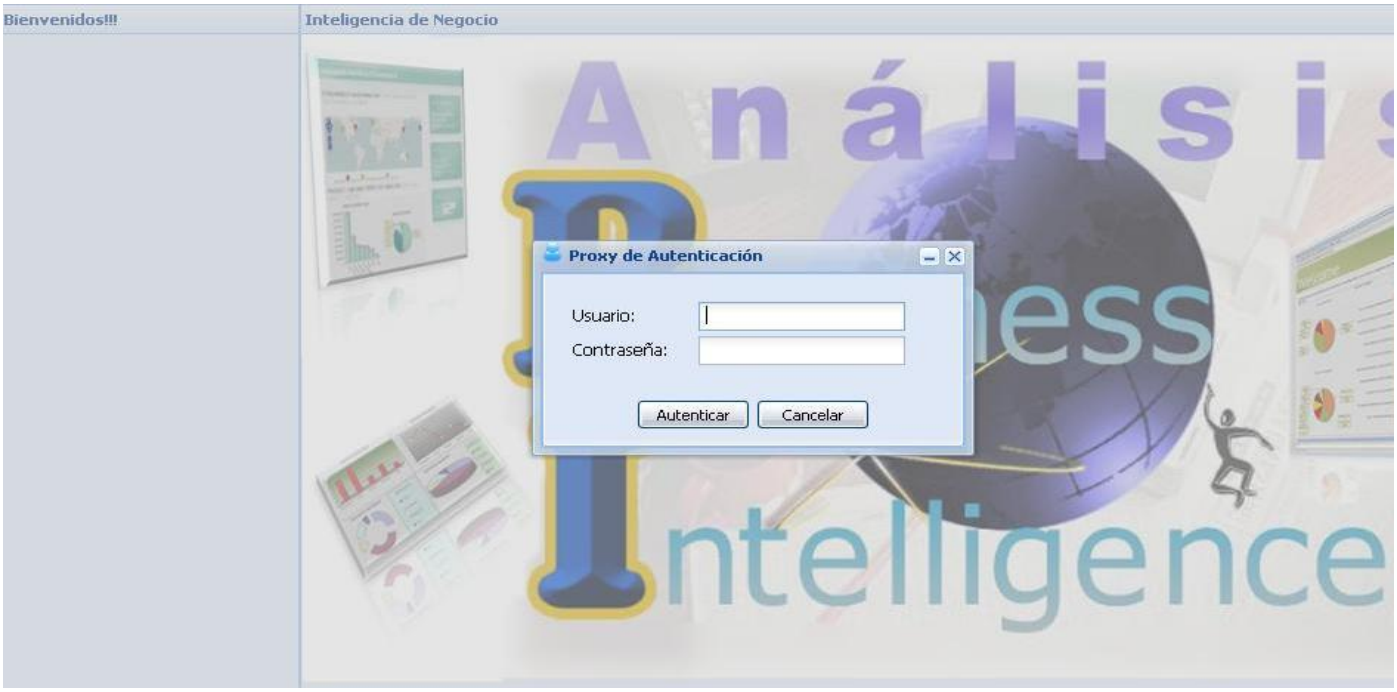

Figura 2: Prototipo interfaz RF1 Autenticar Usuario.

Conceptos tratados	**Conceptos**	**Atributos**
	Seguridad	usuario contraseña
Precondiciones	**Precondiciones**	**Pre-requisito**
	El usuario se encuentra en la portada principal del subsistema.	*no procede*

Descripción	El usuario accede a la interfaz de autenticación 1. El requisito inicia cuando el usuario introduce al sistema sus datos (nombre de usuario y contraseña). 2. El sistema busca el usuario y compara la contraseña haciendo uso de los servicios web publicados por sistema de seguridad SIGIS. 3. En caso de ser correctos los datos introducidos se le habilita la entrada al sistema al usuario. 4. En caso de no ser correctos se envía un mensaje de error.
Validaciones	
Post-condiciones	Usuario autenticado y se habilitan las funcionalidades al usuario según los privilegios.
Post-requisito	Compartimentar información

Tabla 1: Descripción del Requisito Autenticar Usuario.

2.5.2 RF2 Compartimentar información

Conceptos tratados	**Conceptos**	**Atributos**
	Información	
Precondiciones	**Precondiciones**	**Pre-requisito**
	El usuario esta logueado en la aplicación	*RF1 Autenticar Usuario*

Descripción	*Una vez autenticado el usuario* *1. El sistema carga los roles* *2. Según el usuario autenticado que tiene un rol asignado en la Base de Datos.* *3. Se le indica al sistema que rol está autenticado.* *4. El sistema muestra la información al usuario según el rol autenticado.*
Validaciones	
Post-condiciones	*no procede*
Post-requisito	*no procede*

Tabla 2: Descripción del Requisito Compartimentar información.

2.6 Diseño de Software

El diseño es parte fundamental en la construcción del sistema que se pretende crear, pues es una abstracción de cómo quedará el producto final. A partir del diseño se obtendrá un modelo cercano a la versión final del software que se construye. El diseño del software permite traducir los requisitos analizados de un sistema, tanto funcionales como no funcionales, en una representación del software, que inicialmente da una visión del mismo y tras posteriores refinamientos sirve como esquema para la implementación del sistema y es, de ese modo, utilizado como una entrada fundamental de las actividades de implementación. Los objetivos del diseño detallado son los siguientes:

- Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables.
- Visualizar y reflexionar sobre el diseño utilizando una notación común.

2.6.1 Diseño de clases

Mediante los diagramas de clases de diseño es posible obtener una abstracción de cómo quedará la implementación del sistema, los diagramas especifican la relación entre los distintos elementos lo que hace que sea más fácil la implementación, facilitan y proveen una abstracción de lo que será el producto final. En este epígrafe presentaremos el diseño de las clases del sistema, este diagrama se mostrará divido primeramente por las clases correspondientes a cada paquete, y luego se mostrará el diagrama general. El estudio de este diagrama ayuda a un mejor entendimiento de cómo fue diseñada la solución.

Clases pertenecientes al paquete mondrian.seguridad.

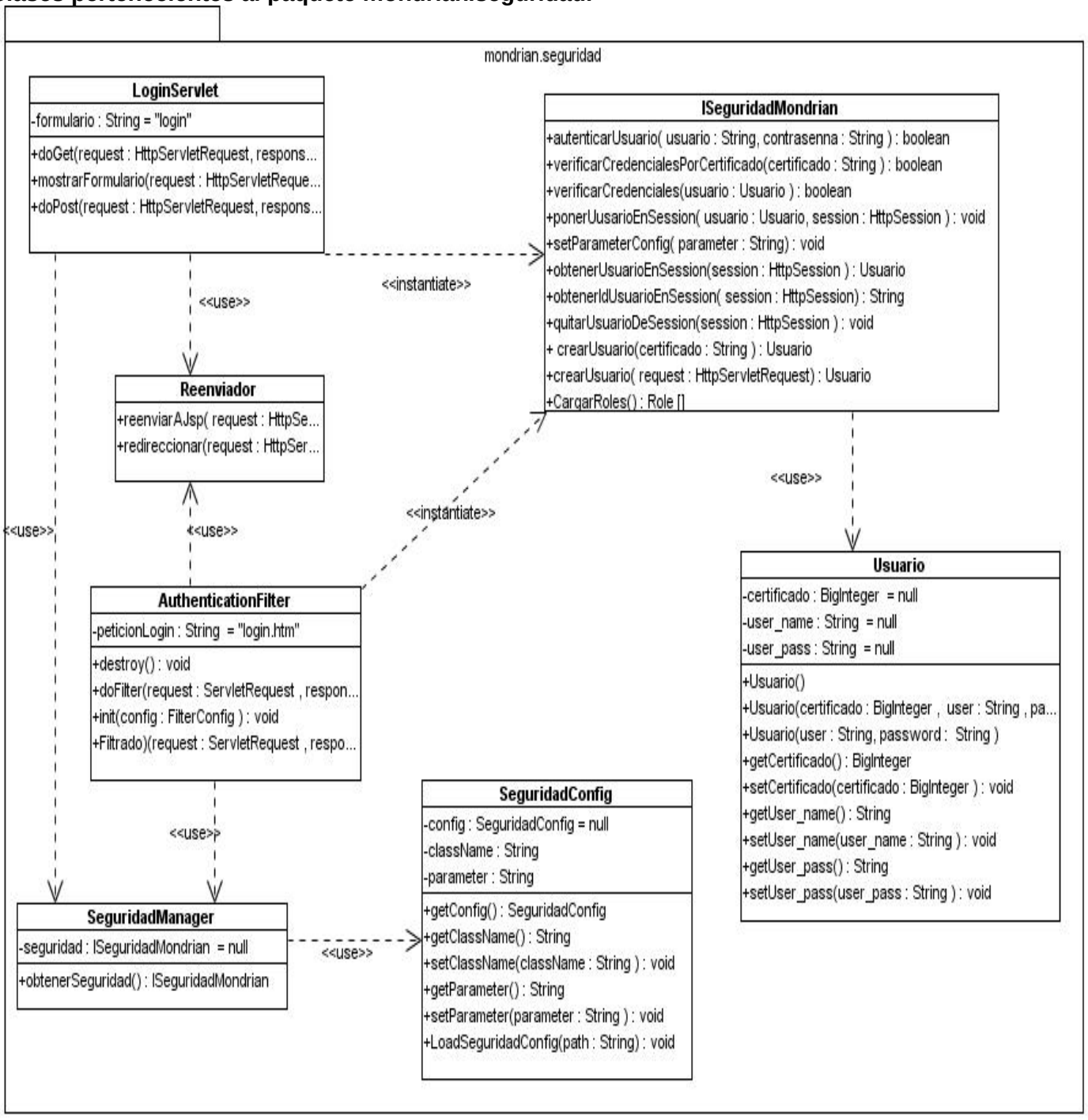

Figura 3: Clases pertenecientes al paquete mondrian.seguridad

Clases pertenecientes al paquete plugins_seguridad_mondrian

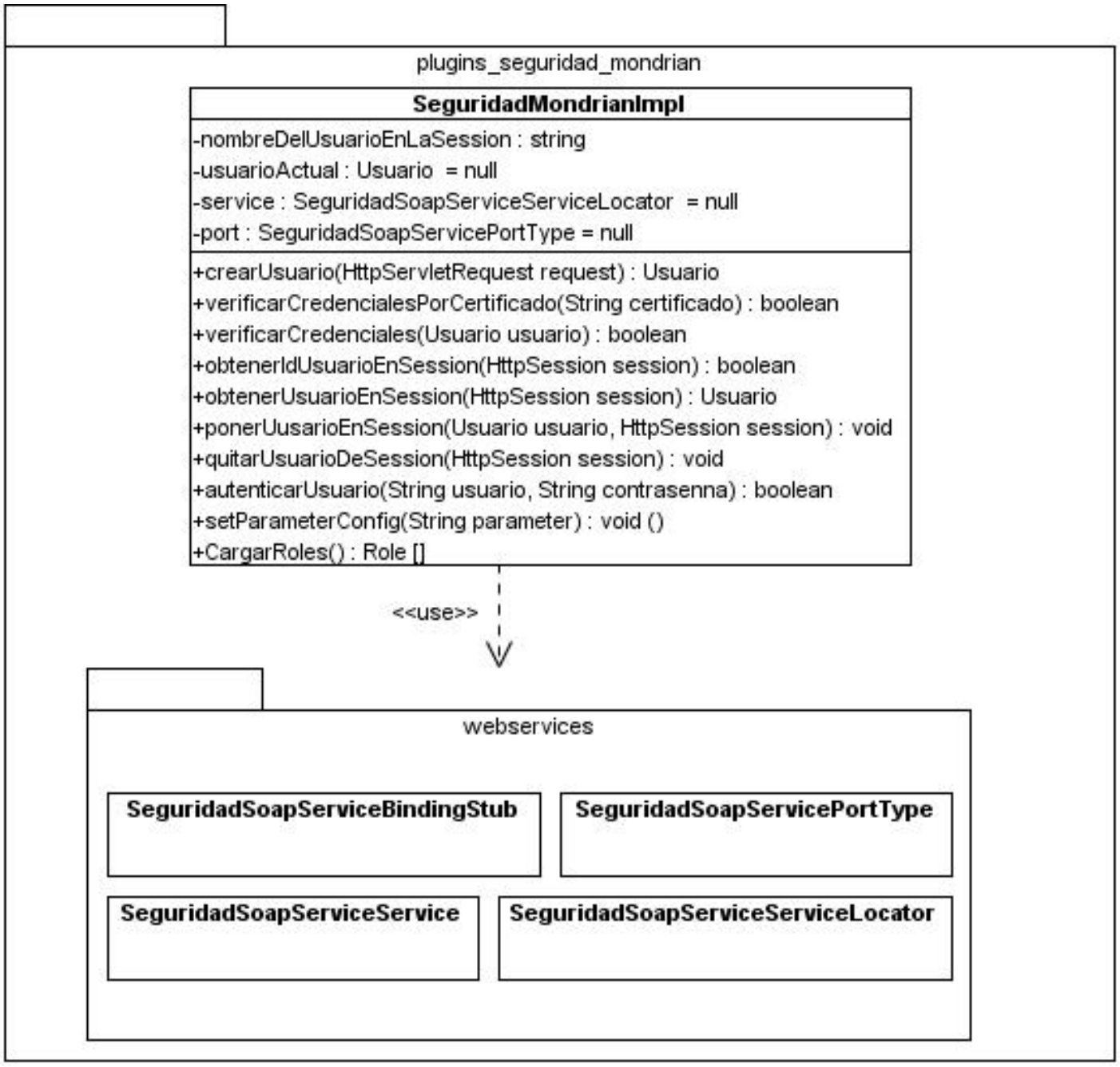

Figura 4: Clases pertenecientes al paquete plugins_seguridad_mondrian.

Clases pertenecientes al paquete Roles.

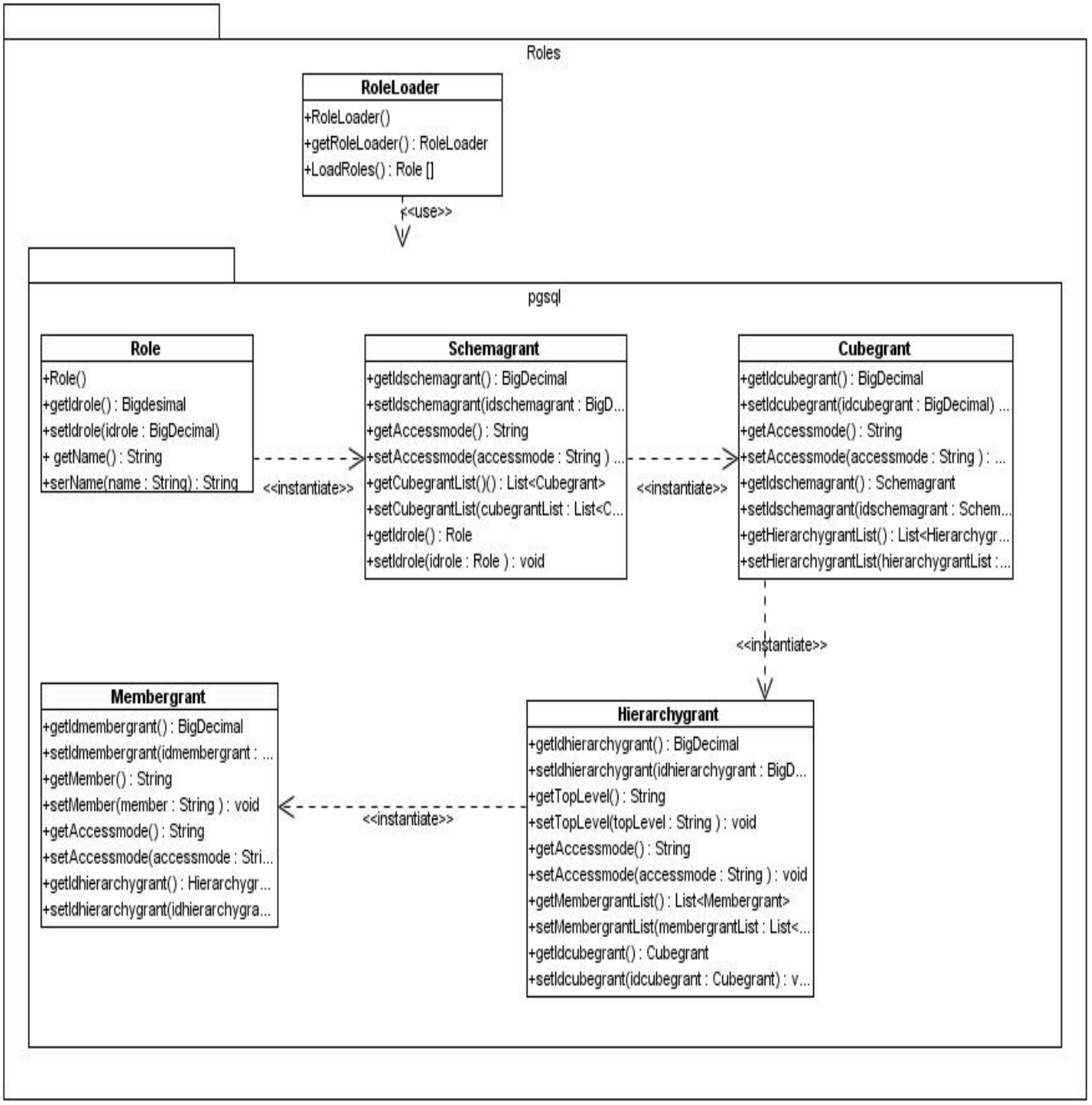

Figura 5: Clases pertenecientes al paquete Roles.

Diagrama de Clase General

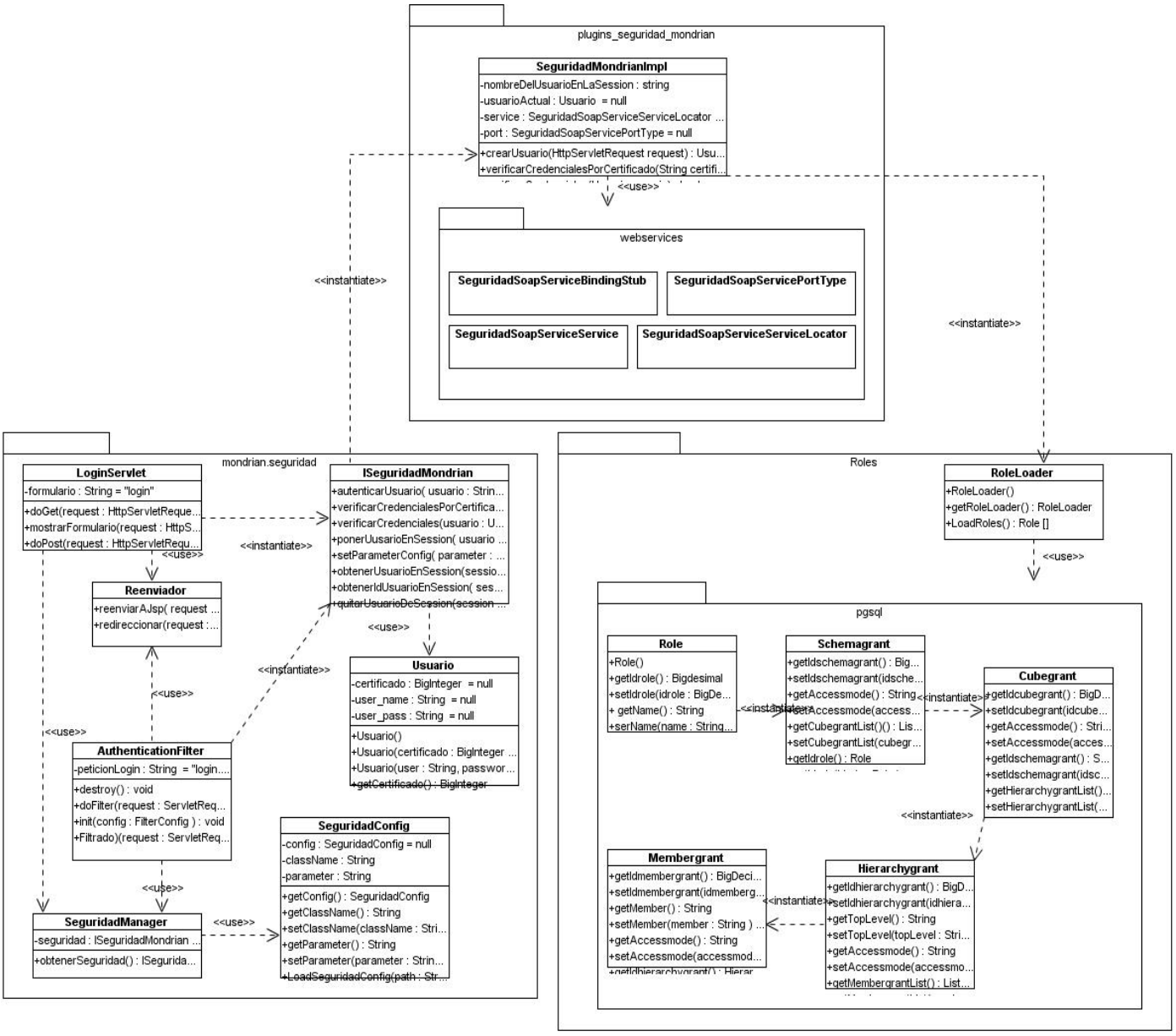

Figura 6: Diagrama de Clase General.

Descripción de las clases

Nombre de la Clase: ISeguridadMondrian	
Tipo de Clase: interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	autenticarUsuario(usuario : String, contrasenna : String) verificarCredencialesPorCertificado(certificado : String) verificarCredenciales(usuario : Usuario) ponerUusarioEnSession(usuario: Usuario, session : HttpSession) setParameterConfig(parameter : String) obtenerUsuarioEnSession(session : HttpSession) obtenerIdUsuarioEnSession(session : HttpSession) quitarUsuarioDeSession(session : HttpSession) crearUsuario(certificado : String) crearUsuario(request : HttpServletRequest) CargarRoles()
Descripción	Clase interfaz que brinda las funcionalidades que gestionan la seguridad dentro del subsistema Mondrian.

Tabla 3: Descripción de la clase IseguridadMondrian.

Nombre de la Clase: Usuario	
Tipo de Clase: entidad	
Atributo	**Tipo**
certificado user_name user_pass	BigInteger String String
Para cada responsabilidad:	
Nombre:	Usuario(certificado : BigInteger , user : String , password : String) getCertificado() setCertificado(certificado : BigInteger) getUser_name() setUser_name(user_name : String) getUser_pass() setUser_pass(user_pass : String)
Descripción:	

Tabla 4: Descripción de la clase Usuario.

Nombre de la Clase: LoginServlet	
Tipo de clase :controladora	
Atributo	**Tipo**
formulario	String
Para cada responsabilidad:	
Nombre:	doGet(request : HttpServletRequest, response : HttpServletResponse) mostrarFormulario(request :HttpServletRequest, response : HttpServletResponse) doPost(request : HttpServletRequest, response : HttpServletResponse)
Descripción:	Clase que se encarga de autenticar el usuario en el subsistema.

Tabla 5: Descripción de la clase LoginServlet.

Nombre de la Clase : AuthenticationFilter	
Tipo de clase: controladora	
Atributo	**Tipo**
peticionLogin	String
Para cada responsabilidad:	
Nombre:	destroy() doFilter(request : ServletRequest , response : ServletResponse , filterChain : FilterChain) init(config : FilterConfig) Filtrado(request : ServletRequest , response : ServletResponse , filterChain : FilterChain)
Descripción:	Clase que controla si en cada petición que se le haga al subsistema Mondrian, el usuario este autenticado.

Tabla 6: Descripción de la clase AuthenticationFilter.

Nombre de la Clase : SeguridadMondrianImpl	
Tipo de Clase: controladora	
Atributo	**Tipo**
nombreUsuarioEnSession	String
usuarioActual	Usuario
service	SeguridadSoapServiceServiceLocator
port	webservices.SeguridadSoapServicePortType
Para cada responsabilidad:	

Nombre:	crearUsuario(HttpServletRequest request) verificarCredencialesPorCertificado(String certificado) verificarCredenciales(Usuario usuario) obtenerIdUsuarioEnSession(HttpSession session) obtenerUsuarioEnSession(HttpSession session) ponerUusarioEnSession(Usuariousuario, HttpSession session) quitarUsuarioDeSession(HttpSession session) autenticarUsuario(String usuario, String contrasenna) setParameterConfig(String parameter) CargarRoles()
Descripción:	Esta implementa los métodos para la seguridad de la clase interfaz **IseguridadMondrian** haciendo uso de los servicios web publicados por el sistema de seguridad de la UCID, SIGIS.

Tabla 7: Descripción de la claseSeguridadMondrianImpl.

Nombre de la Clase: RoleLoader	
Tipo de clase controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	RoleLoader() getRoleLoader() LoadRoles()
Descripción:	**Cargar roles de BD**

Tabla 8: Descripción de la claseRoleLoader.

Nombre de la Clase : Role	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Role() getIdrole() setIdrole(idrole : BigDecimal) getName() serName(name : String)
Descripción:	

Tabla 9: Descripción de la clase Role.

Nombre de la Clase : Schemagrant	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	getIdschemagrant() setIdschemagrant (idschemagrant : BigDecimal) getAccessmode() setAccessmode(accessmode : String) getCubegrantList() setCubegrantList(cubegrantList : List<Cubegrant>) getIdrole() setIdrole(idrole : BigDecimal)
Descripción:	

Tabla 10: Descripción de la clase Schemagrant.

Nombre de la Clase : Cubegrant	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	getIdcubegrant() setIdcubegrant (idcubegrant : BigDecimal) getAccessmode() setAccessmode(accessmode : String) getIdschemagrant() setIdschemagrant (idschemagrant : BigDecimal) getHierarchygrantList() setHierarchygrantList(hierarchygrantList : List<Hierarchygrant>)
Descripción:	

Tabla 11: Descripción de la clase Cubegrant.

Nombre de la Clase : Hierarchygrant	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	getIdhierarchygrant() setIdhierarchygrant(idhierarchygrant : BigDecimal) getTopLevel() setTopLevel(topLevel : String) getAccessmode() setAccessmode(accessmode : String)

	getIdcubegrant() setIdcubegrant (idcubegrant : BigDecimal) getIdschemagrant() setIdschemagrant (idschemagrant : BigDecimal) setMembergrantList() setMembergrantList(membergrantList : List<Membergrant>)
Descripción:	

Tabla 12: Descripción de la clase Hierarchygrant.

Nombre de la Clase : Membergrant	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	getIdmembergrant() setIdmembergrant(idmembergrant : BigDecimal) setMember() setMember (member : String) getIdhierarchygrant() setIdhierarchygrant(idhierarchygrant : BigDecimal) getAccessmode() setAccessmode(accessmode : String)
Descripción:	

Tabla 13: Descripción de la clase Membergrant.

2.6.2 Diagramas de interacción.

Los diagramas de interacción se dividen en dos tipos, los diagramas de secuencia y los diagramas de colaboración, a continuación se presenta el diagrama de secuencia correspondiente al requisito Autenticar Usuario.

Escenario Autenticar Usuario

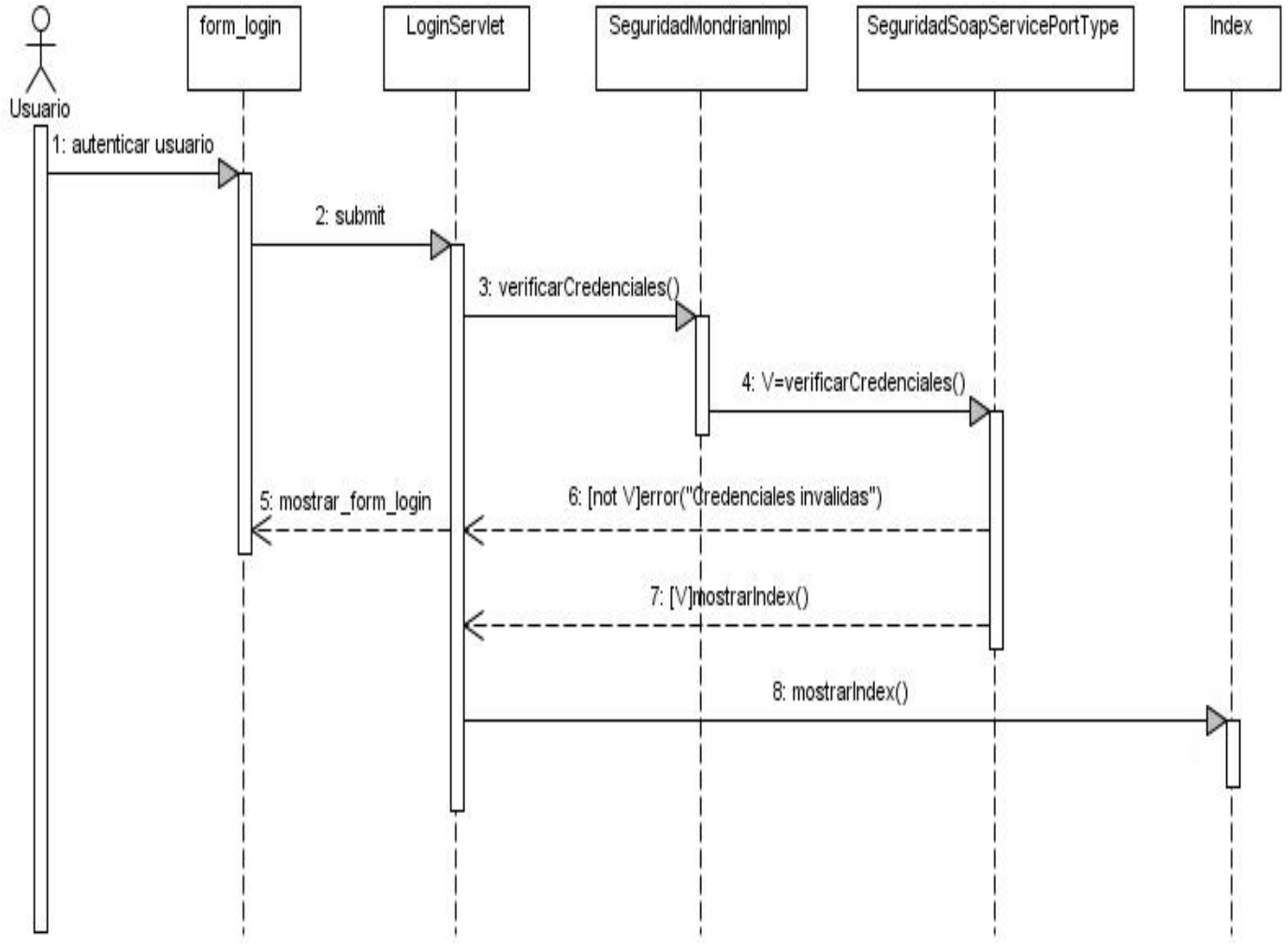

Figura 7: Escenario Autenticar Usuario.

2.6.3 Diagrama de clases persistentes

Los diagramas de clases persistentes contienen las clases que tendrá la base de datos del sistema, los atributos de cada clase y sus relaciones así como la cardinalidad que existe entre ellas.

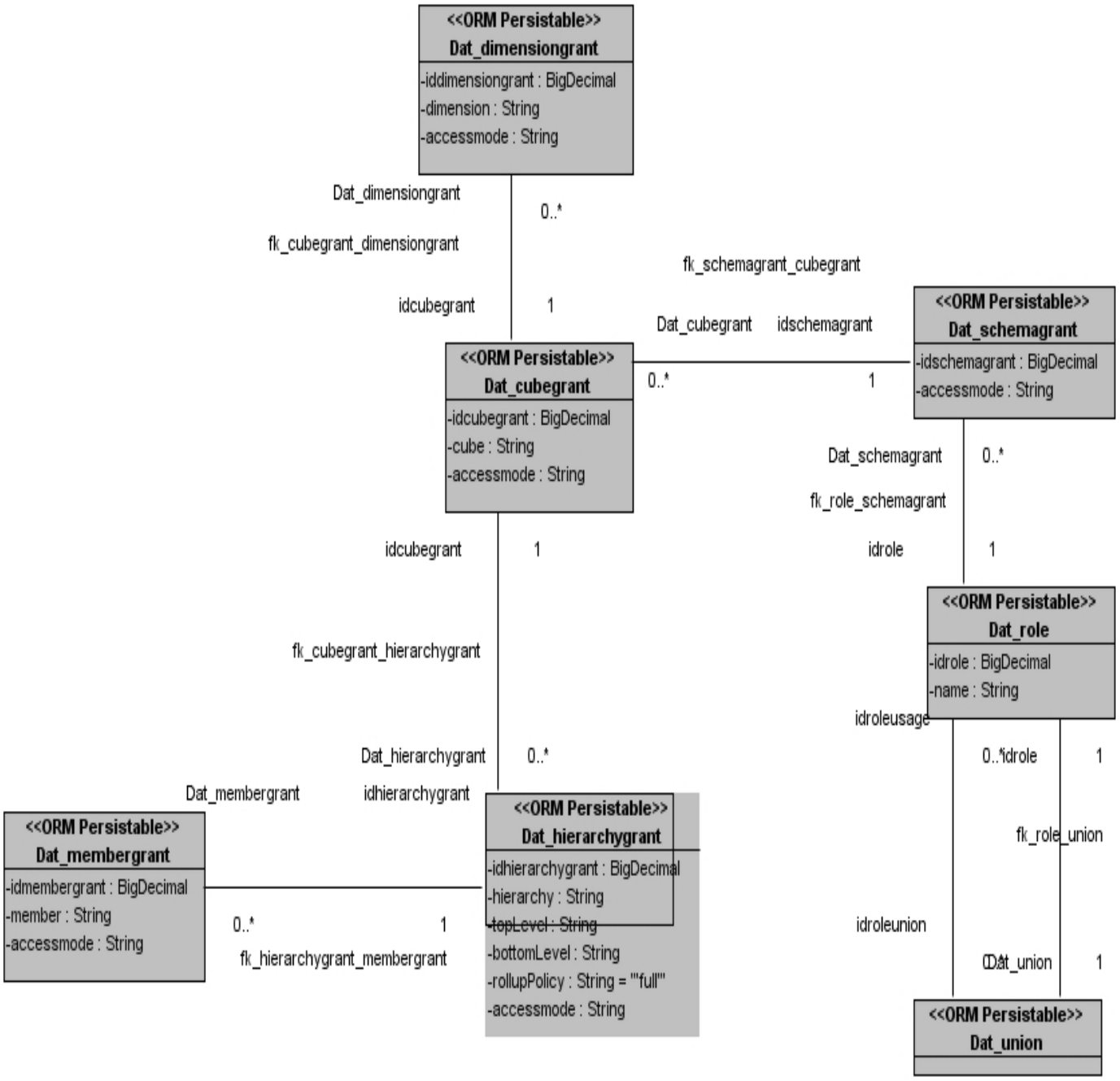

Figura 8: Diagrama de clases persistentes.

Conclusiones Parciales

Durante el transcurso del capítulo se argumentaron los fundamentales aspectos que se llevan a cabo durante el proceso de análisis de la solución propuesta, comenzando por el planteamiento de los requisitos funcionales y de los requisitos no funcionales para garantizar una ejecución eficaz del componente. También se plasmó el diseño de clases para una mejor comprensión de cómo está estructurada la solución del problema.

3

Capítulo

Implementación y Prueba.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Después de haber realizado un buen Diseño se da paso a la Implementación del componente por parte de los desarrolladores, con el objetivo de ir conformando el proyecto como un sistema completo y con un acabado a la altura de los requerimientos previamente definidos.

Los objetivos de este capítulo son, representar el Modelo de datos y el diagrama de componente. Se abordará sobre el tipo de prueba que se realizarán al componente.

3.1 Diseño de la base de datos.

Las bases de datos necesitan una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usarán.

La puesta en práctica de la base de datos es el paso final en el desarrollo de aplicaciones de soporte del negocio. Se conforman con los requisitos del proceso del negocio, que es la primera abstracción de la vista de la base de datos.

En este epígrafe se muestra el diseño de la base de datos del sistema propuesto a través del modelo lógico de datos y el modelo físico de datos.

3.1.1 Modelo físico de datos (modelo de datos).

El modelo físico de los datos describe la representación lógica y física de datos persistentes en el sistema, dicho modelo se muestra a continuación.

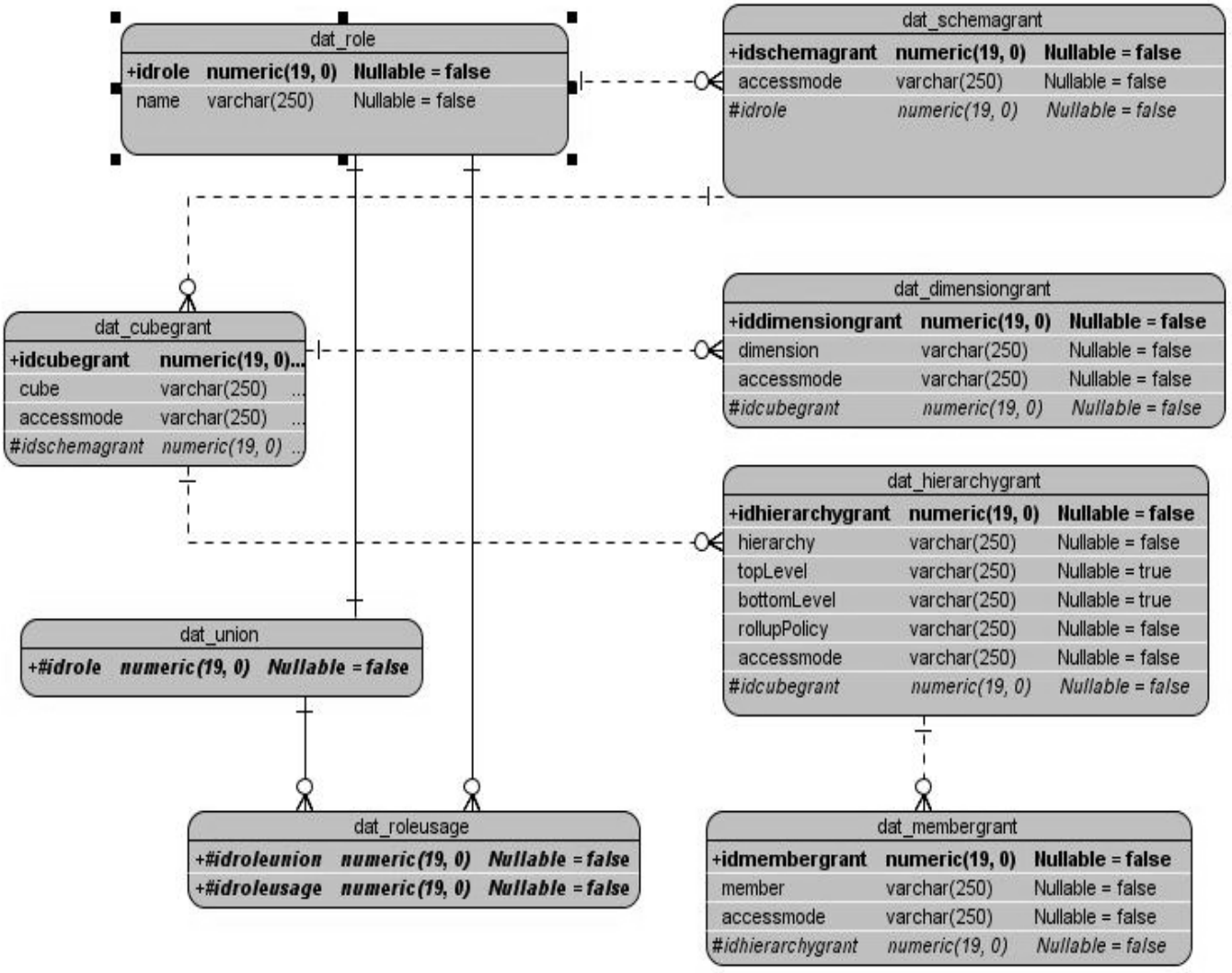

Figura 9: Modelo físico de datos.

3.1.2 Descripción de las tablas de la base de datos.

Nombre: dat_role		
Descripción: tabla que almacena roles que tendrán una serie de concesiones o rechazos por elementos del esquema de la BD.		
Atributo	**Tipo**	**Descripción**
idrole	decimal	Identificador del rol
name	varchar	Nombre del rol

Tabla 14: Descripción de la tabla dat_role.

Nombre: dat_schemagrant		
Descripción: Permite o niega al rol el acceso al esquema, en el acceso estaría todo, todas las dimensiones o nada. Si el acceso es todas a las dimensiones, el rol tiene acceso a todo pero todavía requiere acceso explícito al cubo.		
Atributo	**Tipo**	**Descripción**
idschemagrant	decimal	Identificador del esquema de la BD
accessmode	varchar	Modo de acceso al esquema
idrole	decimal	Identificador del rol que tiene acceso a este esquema

Tabla 15: Descripción de la tabla dat_scemagrant.

Nombre: dat_cubegrant		
Descripción: Permite o niega al rol el accede a un cubo, el acceso seria a todo o nada.		
Atributo	**Tipo**	**Descripción**
idcubegrant	decimal	Identificador del cubo
cube	varchar	Nombre del cubo dentro del esquema
accessmode	varchar	Modo de acceso al cubo
idschemagrant	decimal	Identificador del esquema donde está contenido el cubo dentro de la BD

Tabla 16: Descripción de la tabla dat_cubegrant.

Nombre: dat_dimensiongrant		
Descripción: Permite o niega al rol el acceso a una dimensión, este sería a todo o nada. El rol tiene acceso implícito a una dimensión cuando se le da acceso a un cubo.		
Atributo	**Tipo**	**Descripción**
iddimensiongrant	decimal	Identificador de la dimensión
accessmode	varchar	Modo de acceso a la dimensión
idcubegrant	decimal	Identificador del cubo donde esta contenido la dimensión

Tabla 17: Descripción de la tabla dat_dimensiongrant.

Nombre: dat_hierarchygant		
Descripción:		
Atributo	**Tipo**	**Descripción**
idhierarchygant	decimal	Identificador de la jerarquía
hierarchy	varchar	Nombre de la jerarquía
topLevel	varchar	
bottomLevel	varchar	
rollupPolicy	varchar	
accessmode	varchar	Modo de acceso a la jerarquía
idcubegrant	decimal	Identificador del cubo donde esta contenido la jerarquía

Tabla 18: Descripción de la tabla dat_hierarchygrant.

Nombre: dat_membergrant		
Descripción:		
Atributo	**Tipo**	**Descripción**
Id_membergrant	decimal	Identificador del elemento
member	varchar	Nombre del elemento
accessmode	varchar	Modo de acceso a los elementos
idhierarchygant	decimal	Identificador de la jerarquía donde está contenido el elemento

Tabla 19: Descripción de la tabla dat_membergrant.

3.2 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean éstos de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

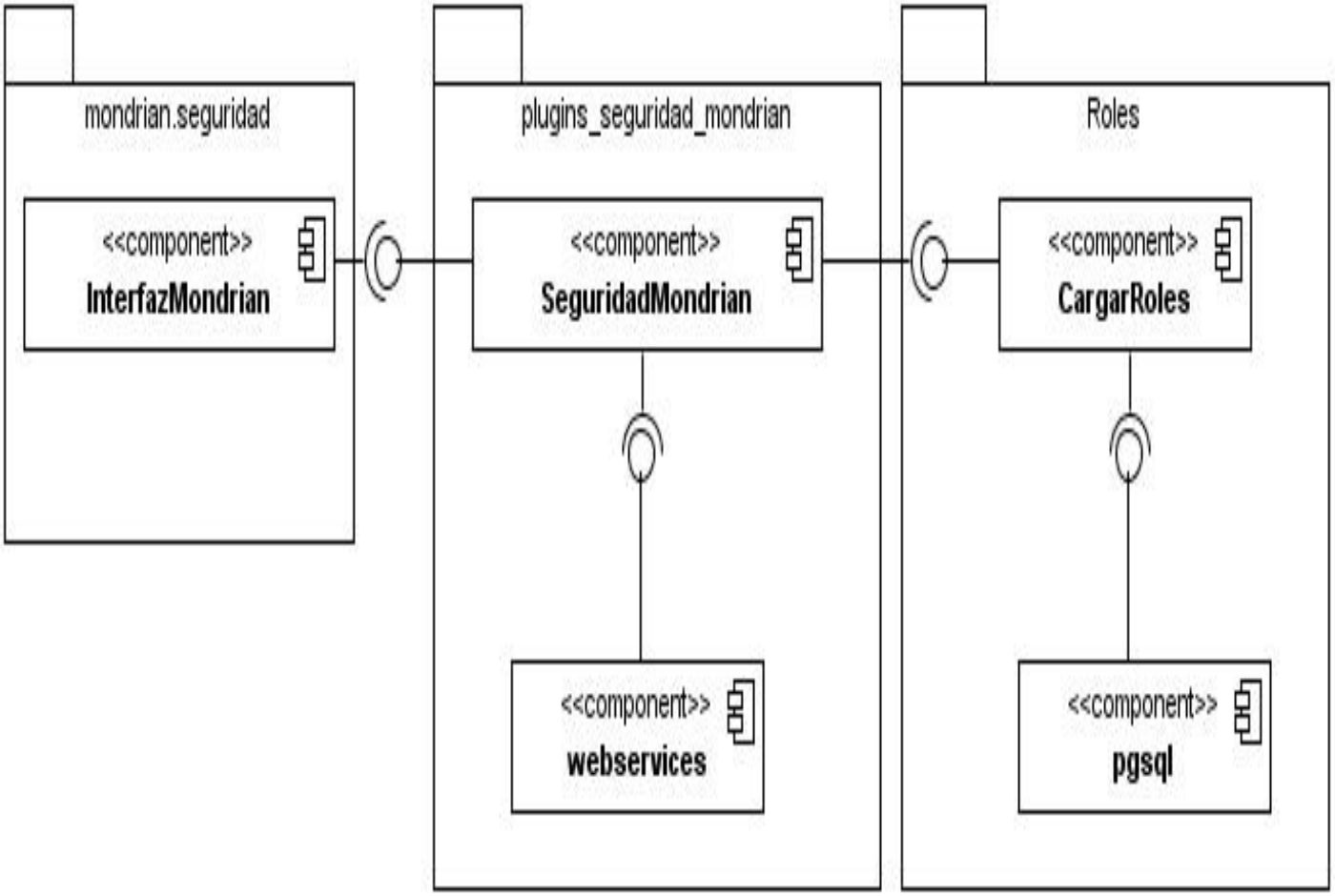

Figura 10: Diagrama de componentes

	Componentes Internos		
Componentes Internos	**ISeguridadMondrian**	**RoleLoader**	**Webservices**
SeguridadMondrianImpl	autenticarUsuario() verificarCredencialesPorCertificado() verificarCredenciales() ponerUusarioEnSession() setParameterConfig() obtenerUsuarioEnSession() obtenerIdUsuarioEnSession() quitarUsuarioDeSession() crearUsuario() CargarRoles()	LoadRoles()	verificateCertificate (BigInteger certificate) authenticateUser (String usuario, String password) setSeguridadSoapServicePortEndpointAddress (String address) getSeguridadSoapServicePort()

Tabla 20: Matriz de Integración de Componentes Interna.

	Componentes externos	
Componentes Internos	**SIGIS**	
Webservices	verificateCertificate (BigInteger certificate) authenticateUser (String usuario, String password) setSeguridadSoapServicePortEndpointAddress (String address) getSeguridadSoapServicePort()	

Tabla 21: Matriz de Integración de Componentes Externa.

3.3 Prueba

Existen en la actualidad varios tipos de pruebas de software, todas están destinadas a encontrar la mayor cantidad de defectos para ofrecer un producto de óptima calidad, las pruebas no pueden probar la ausencia de defectos, sino detectar la existencia de estos. Entre los principales métodos de prueba se encuentran, pruebas de **Caja Blanca** y de **Caja Negra**, las primeras se centran en la revisión del código fuente del software y requieren conocimiento del funcionamiento interno del sistema. Las últimas por su parte se refieren a las pruebas que se realizan a la interfaz del sistema, son conocidas también como pruebas de comportamiento y se basan en los requerimientos funcionales del sistema.

3.3.1. Diseños de caso de prueba

Los diseños de casos de prueba son una técnica para probar el sistema mediante el método de Caja Negra, se pueden elaborar por CU o por requisitos y su objetivo fundamental es ofrecer al usuario una guía detallada de cómo realizar la prueba, mostrando los pasos a seguir y los datos a introducir con la finalidad de obtener el resultado esperado.

Condiciones de ejecución

- El usuario se encuentra en la portada principal del subsistema

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Autenticar Usuario	El requisito brinda los servicios de autenticación y autorización al subsistema Mondrian.	Autenticar Usuario correctamente	Se le muestra al usuario la interfaz para autenticarse en el subsistema. El usuario llena los datos correctamente y presiona el botón Autenticar. El sistema entra al subsistema Mondrian, indicándole al usuario que la operación se realizó de manera correcta.

		Autenticar Usuario con datos incorrectos.	Se le muestra al usuario la interfaz para autenticarse en el subsistema. El usuario llena los campos con datos incorrectos y/o deja campos obligatorios vacíos y presiona el botón Autenticar. El sistema muestra un mensaje de error indicando que la operación no se realizó.

Tabla 22: Descripción del caso de prueba

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	user	Textbox	NO	
2	password	Textbox	NO	

Tabla 23: Descripción de variable.

Id del escenario	Escenario	Variable 1 user	Variable 2 password	Respuesta del sistema	Resultado de la prueba
EP 1	Autenticar Usuario correctamente.	V (ciego)	V (ciego)	El sistema muestra la interfaz principal del sistema con la información a la cual tiene acceso el usuario.	Resultado satisfactorio
		V (holguin)	V (holguin)		
EP 2	Autenticar Usuario con datos incorrectos y/o campos vacios.	I(Vacio)	V(ciego)	El sistema emite un mensaje de error.	Resultado satisfactorio
		V(ciego)	I(Vacio)		
		I(admin)	I(admin)	El sistema emite un mensaje de error.	
		I(admin)	V(ciego)		

Tabla 24: Juegos de datos a probar.

Teniendo en cuenta el DCP, se realizaron las pruebas a las funcionalidades implementadas obteniendo resultados satisfactorios, validando así las solución propuesta.

Conclusiones Parciales

En este capítulo se modeló el diagrama de componentes y el modelo de datos, además se describe el proceso de prueba realizado al componente para comprobar si las funcionalidades implementadas cumplen con los requisitos argumentados en el proceso de desarrollo.

CONCLUSIONES

Posterior al estudio que se realizó del estado del arte referente a los sistemas existentes en Cuba y en el mundo para gestionar la seguridad, se llevó a cabo la implementación de un componente de seguridad con el objetivo de garantizar la confidencialidad de la información. Para esto se realizó también:

- Un análisis de las principales tecnologías para el desarrollo que permitió seleccionar las herramientas y metodologías más adecuadas para desarrollar el componente.
- El desarrollo del diseño y la implementación de un componente que permite gestionar la seguridad y la confidencialidad de los datos del subsistema Mondrian.
- La aprobación de la solución propuesta, proceso en el cual surgieron una serie de recomendaciones para propiciar el perfeccionamiento futuro del componente.

De acuerdo con las consideraciones expuestas anteriormente es posible afirmar que se alcanzó de manera satisfactoria el objetivo propuesto: Desarrollar un componente de seguridad para el subsistema de análisis de información Mondrian.

RECOMENDACIONES

Se recomienda realizar futuras iteraciones al componente para mantener una constante actualización con el sistema de seguridad SIGIS, así como proveer implementaciones para otras fuentes de autenticación tales como LDAP[1].

[1] **LDAP** (*Lightweight Directory Access Protocol*), (Protocolo Ligero de Acceso a Directorios) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

Referencias Bibliográficas

1. Introduccion a Pentaho. *Gravitar.* [Online] 2009. http://www.gravitar.biz/index.php/bi/introduccion-pentaho-parte-1/.

2. Pentaho. *Pentaho.* [Online] 2009. [Cited: enero 21, 2010.] http://mondrian.pentaho.org/. ISBN.

3. Maribel Ariza Rojas, Juan Carlos Molina García. [Online] Septiembre 30, 2004. [Cited: Enero 10, 2010.] http://pegasus.javeriana.edu.co/~jcpymes/Docs/DSBC.pdf.

4. [Online] [Cited: Febrero 10, 2010.] http://www.itenlinea.com/noticias.php?tb=n_ebusiness&section=e-Business&id=7.

5. Apache Tomcat. *Apache Tomcat.* [Online] 2009. [Cited: Diciembre 10, 2009.] http://tomcat.apache.org/.

6. Bases de Datos. [Online] 2008. [Cited: Enero 20, 2009.]

7. Lenguajes de Programación*Lenguajes de Programación*http://www.lenguajes-de-programacion.com.

8. **Marzo, Josep Vilalta.** Introducción a UML. [Online] 2008. http:www.vico.org/aRecursosPrivats/TRAD_introUML.pdf.

Bibliografía

Valdéz Péres, Damián. Maestros del Web. Qué son las bases de datos? [En línea] octubre de 2007. *Bases de Datos.* http:www.dspace.espol.edu.ec/bitstream/123456789/1416/1/2751.pdf : s.n., 2009.

Apache Tomcat. *Apache Tomcat.* [Online] 2009. [Cited: Diciembre 10, 2009.] http://tomcat.apache.org/.

Bases de Datos. [Online] 2008. [Cited: Enero 20, 2009.]

[Online] http://wolfkira.wordpress.com/2008/02/28/postgresqlvsmysql.PostGreSQL vs. MySQL. *PostGreSQL vs. MySQL.* [Online] [Cited: Enero 15, 2010.] http://wolfkira.wordpress.com/2008/02/28/postgresqlvsmysql

Dashboard Reporting*Dashboard Reporting*http://www.roseindia.net/enterprise/persistenceframework.shtml

M. Fowler, K. Scott. *Scott. UML gota a gota. s.l.* . s.l. : Prentice Hall, 1997.

Bases de Datos. [Online] 2008. [Cited: Enero 20, 2009.] http://www.dspace.espol.edu.ec/bitstream/123456789/1416/1/2751.pdf

Sistema de Gestion de Base de Datos*Sistema de Gestion de Base de Datos* http://www.monografias.com/trabajos56/sistemasbasesdedatos/

[Online] [Cited: Febrero 10, 2010.] http://www.itenlinea.com/noticias.php?tb=n_ebusiness&section=e-Business&id=7

Lenguajes de Programación *Lenguajes de Programación* http://www.lenguajes-de-programacion.com.

Introduccion a Pentaho. *Gravitar.* [Online] 2009. http://www.gravitar.biz/index.php/bi/introduccion-pentaho-parte-1/

Morate, Diego García. *MANUAL DE WEKA.* Nueva Zelanda : s.n., 1993.

Seifert, Jeffrey W. *Data Mining:An Overview.* Washington : s.n., 2004.

Knowledge Discovery in Databases: An Overview. **Frawley, William J., Piatetsky-Shapiro, Gregory and Matheus, Christopher J.** 1992, AI Magazine, p. 70.

Marzo, Josep Vilalta. Introducción a UML. [Online] 2008. http:www.vico.org/aRecursosPrivats/TRAD_introUML.pdf.

Pentaho. *Pentaho.* [Online] 2009. [Cited: enero 21, 2010.] http://mondrian.pentaho.org/. ISBN.

Pentaho. *Pentaho Data Integration.* [Online] 2005-2010. http://www.pentaho.com/products/data_integration/

Portada sobre la Plataforma Pentaho Open Sourse Businees Intelligence. *Portada sobre la Plataforma Pentaho Open Sourse Businees Intelligence.* [Online] 2006-2008. http://pentaho.almacen-datos.com/.

GLOSARIO

SSO: Single Sing-On. Mecanismo que permite la autenticación única por parte del usuario para acceder a sus recursos, es decir, introducir una sola vez el usuario y contraseña, sin necesidad de volver a poner el login a la hora de acceder a otros recursos en los que aún no se había autenticado.

UCID: Unidad de Compatibilización Integridad y Desarrollo de Software para la Defensa.

LDAP: Lightweight Directory Acces Protocol. Es un protocolo de tipo cliente – servidor para acceder a un servicio de directorio ordenado y distribuido para buscar información en un entorno de red.

Autenticación: Mecanismo del sistema de información para poder identificar a los usuarios que acceden a sus recursos, asegurando la integridad y autenticidad de los datos.

Autorización: Proceso por el cual se autoriza al usuario identificado a acceder a determinados recursos del sistema.