Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite de Pentaho

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

Autores:

Ramsés López Sosa

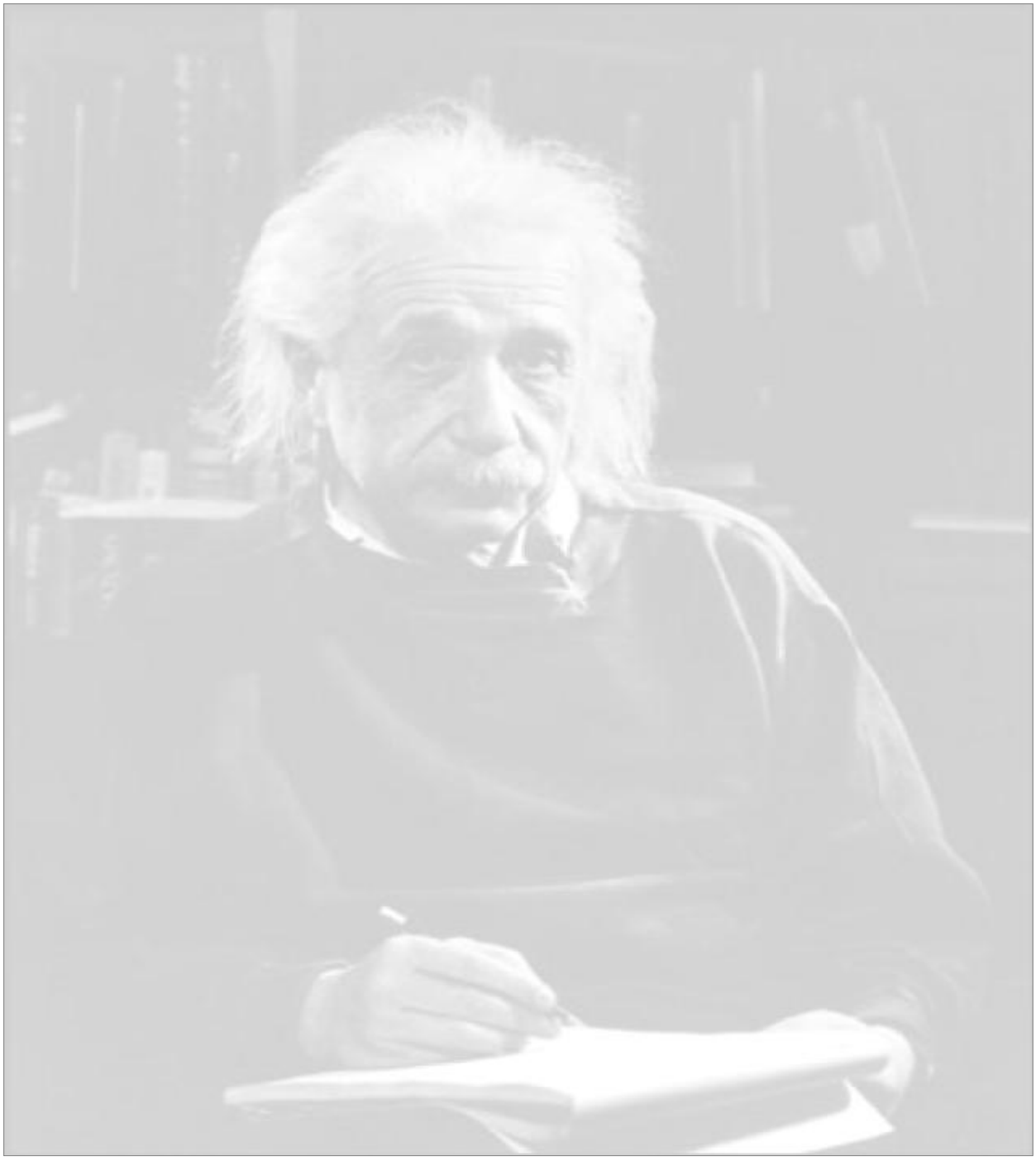
Yuniel Acebal Urtiaga

Tutor:

Ing. Yunier Santana Aldana

La Habana, junio 2012

"Año 54 de la Revolución"



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber"

Albert Einstein

DECLARACIÓN DE AUTORÍA

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

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

_____ _____

Yuniel Acebal Urtiaga
Firma del autor

Ramsés López Sosa
Firma del autor

_____

Ing. Yunier Santana Aldana

Firma del tutor

DATOS DE CONTACTO

Tutor:

Ing. Yunier Santana Aldana
Universidad de las Ciencias Informáticas, La Habana, Cuba
Correo electrónico: ysaldana@uci.cu

AGRADECIMIENTOS

A toda mi familia por darme fuerza y confianza a cumplir mi sueño, a mi padre que siempre he podido contar con él, con sus consejos, por ayudarme a convertirme en el profesional que hoy soy, por su esfuerzo, por estar a mi lado durante todo este tiempo, a mi madre por estar siempre a mi lado, a mi hermano, a mi abuela por darme fuerzas a terminar y decirme siempre "ya horita terminas, falta poco", me lo dijo desde primer año y fue de mucha ayuda, a todos mis primos por confiar en mí, los quiero como hermanos y siempre serán parte de mi vida.

A una persona muy especial en mi vida, Yohana, por ayudarme en estos años de Universidad y por estar a mi lado en todo momento, por quererme tanto y por formar parte de mi vida, nunca te voy a olvidar.

A mis profesores Dennis y Dany por poder contar con ellos y por ser mis amigos, a mis amistades de la universidad y de toda la vida Ale, Tico, Arumis, Yanet, Saray, Yoandy, al Friki (Rene) por estar siempre en mi equipo y estar juntos en todo momento, a mis viejos amigos de siempre Harold, Arturo y Sheyla por compartir buenos momentos con ellos.

A todos los que de una forma u otra se preocupan por saber cómo estoy, si he dejado de mencionar algún nombre quiero que sepa que está en mi corazón, que le agradezco ser parte de mi vida y siempre tendrá mis agradecimientos.

De Ramsés

AGRADECIMIENTOS

A dos mujeres que me han enseñado todo lo que sé y que se encargaron siempre de darme lo mejor y guiarme por el camino correcto, mi mami y mi abue, siempre estaré en deuda con ustedes.

A mi tía Margui, a mis primitas bellas Claudia y Yaniris, a mi tío Osva, a Yanet y a mi hermanita Mary que siempre se está preocupando por mí y que me quiere tanto.

A mis dos hermanitos de la universidad que espero siempre poder contar con ellos, Dayi y Tico, gracias por todo.

A una familia que me abrió sus puertas y me han hecho sentir todo el tiempo un miembro más de ellos. A Idania, a Juan, a Sadan, al abuelo, a la abuela, a Juan Carlos, a Sonia, por su ternura, amor y dedicación gracias.

A una personita que llegó hace poco tiempo pero que se ha encargado de llenar mi vida de mucho amor y demostrarme que está conmigo para llorar tristezas y reír alegrías, mi ST linda gracias por ser esa personita especial que eres.

A Sadiel un amigo que conocí hace poco pero que me ha demostrado que puedo contar con él en cualquier momento.

A todas mis amistades de la universidad con las cuales compartí lindos momentos.

De Yuniel

DEDICATORIA

Este trabajo va dedicado a dos personitas que son mi adoración, mi mami y mi abuelita, las cuales han sido madre y padre a la vez.

De Yuniel

A toda mi familia y mis amistades que de una forma u otra hicieron posible cumplir mi sueño y ser hoy la persona y el profesional que soy, a mi papa por darme tanto apoyo en todo momento, a mi madre, por su amor, los quiero mucho a los dos, a mis primos que son parte fundamental de mi vida, a mis tías y mi tío por inspirarme a lograr mi sueño, a toda mi familia en general los amo. A todos mis amigos los nuevos y los de siempre por preocuparse por saber cómo estoy y preocuparse por mí, a Yohana por ser una persona muy especial y ocupar un lugar en mi corazón además de poder contar con ella en todo momento.

De Ramsés

RESUMEN

La seguridad es un factor fundamental para cualquier entidad o empresa. La pérdida, robo o mala manipulación de los datos conlleva a daños económicos y sociales irreparables. En la Universidad de la Ciencias Informáticas se utilizan una serie de herramientas para el análisis y tratamiento de la información, una de ellas es el Servidor de Inteligencia de Negocios de la Suite de Pentaho. A pesar de las ventajas que brinda esta herramienta para la toma de decisiones, presenta una serie de deficiencias en cuanto a la seguridad. El propósito del presente trabajo consiste en el desarrollo de un módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite de Pentaho. Para guiar el proceso de desarrollo se empleó la metodología XP (Programación Extrema), como herramienta de modelado Visual Paradigm en su versión 8.0, como Sistema Gestor de Base de Datos PostgreSQL en su versión 9.0, como servidor web Apache Tomcat en su versión 6.0, como entorno de desarrollo integrado Eclipse en su versión 3.2 y como lenguaje de programación Java Web. Se tuvo en cuenta además, el uso de patrones y estándares de codificación. Al módulo se le realizaron pruebas de aceptación, funcionalidad y además se probó en un ambiente centralizado de autenticación de usuarios, obteniendo en todos los casos resultados favorables.

Palabras clave: módulo, seguridad, Suite de Pentaho.

ÍNDICE

ÍNDICE DE TABLAS Y FIGURAS

INTRODUCCIÓN

La era de la tecnología ha propiciado que el mundo se encuentre en constante desarrollo tecnológico. Las empresas buscan alternativas que les permitan mantenerse bien ubicadas dentro del mercado, que cada vez se torna más competitivo. Muchas compañías han optado por recopilar parte de la información relevante que manejan en almacenes de datos bien estructurados, a fin de poder analizarla, ya que constituye uno de los soportes fundamentales para el proceso de toma de decisiones.

Cuba no se encuentra aislada de este desarrollo, la necesidad del análisis de la información en los organismos y empresas estatales, se convierte en tarea fundamental para la planificación de la economía del país, enmarcada en profundos cambios. Para ello se han realizado un grupo de acciones con vista a fomentar el desarrollo e informatización de un gran número de sectores.

La Universidad de las Ciencias Informáticas (UCI), forma parte de las alternativas que se han llevado a cabo, con el objetivo de contribuir al desarrollo y uso de las nuevas Tecnologías de la Información y las Comunicaciones (TIC). La misma cuenta con varios centros de desarrollo. Uno de ellos es el Centro de Tecnologías de Gestión de Datos (DATEC), en el cual se desarrollan soluciones de almacenes de datos, así como soluciones integrales de software con componentes de análisis de información. Para la construcción de los almacenes de datos se utiliza un conjunto de herramientas que se encuentran incluidas en la Suite de Pentaho.

La Suite de Pentaho incluye herramientas de administración que reducen los costos de operación al simplificar el despliegue, mejoran la fiabilidad y la facilidad de uso, mejoran el rendimiento, la estabilidad de la solución y muestran perspectivas de análisis de información que apoyan el proceso de toma de decisiones. También contiene otras herramientas para el análisis de información, dentro de las cuales se encuentra el Servidor de Inteligencia de Negocios de la Suite de Pentaho (Pentaho BI Server).

Pentaho BI Server funciona como un sistema basado en administración web de informes, el servidor de integración de aplicaciones y un motor de flujo de trabajo ligero. Está diseñado para integrarse fácilmente en cualquier proceso de negocio, sin embargo presenta algunas deficiencias entre las cuales se encuentra el proceso de gestionar los permisos a los roles. Esta tarea no se realiza de manera interactiva, sino de forma manual, convirtiendo el trabajo en una tarea tediosa, engorrosa y costosa en tiempo y esfuerzo. Además, otra deficiencia en cuanto a seguridad, es que contiene una

base de datos que almacena toda la información de los usuarios, parámetros de conexión y los datos necesarios para el funcionamiento de la plataforma en ficheros de texto plano de fácil acceso, que no poseen ninguna protección, siendo vulnerables al robo, modificación o pérdida de los datos. Otra deficiencia que posee esta plataforma, es en el manejo de las contraseñas de los usuarios, que al no poseer un mínimo de fortaleza, vuelve vulnerable el sistema de seguridad del Pentaho BI Server.

A raíz de las condiciones descritas anteriormente se define como **problema a resolver**: ¿Cómo contribuir a la seguridad del Servidor de Inteligencia de Negocios de la Suite de 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 para el control de la seguridad en la herramienta Pentaho BI Server definiéndose como **objetivo general**: Desarrollar un módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite de Pentaho.

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

1. Realizar una evaluación de la seguridad del Servidor de Inteligencia de Negocios de la Suite de Pentaho.
2. Realizar análisis y diseño del módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite Pentaho.
3. Realizar la implementación y prueba del módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite Pentaho.

Para cumplir con los objetivos específicos, se trazaron las siguientes **tareas de la investigación**:

1. Selección y revisión bibliográfica para actualizar los logros y limitaciones existentes en la evaluación de la seguridad del Servidor de Inteligencia de Negocios.
2. Caracterización de las metodologías, herramientas y tecnologías a utilizar en el desarrollo del módulo de seguridad.
3. Selección de un gestor de bases de datos para el despliegue de la base datos del sistema que trae el servidor.
4. Levantamiento de requisitos.
5. Diseño del modelo de datos.

6. Realización del diseño del subsistema de gestión de permisos a los roles del sistema.
7. Diseño del subsistema de control de seguridad en las contraseñas de usuarios.
8. Integración del Pentaho BI Server con el gestor de base de datos seleccionado.
9. Implementación del subsistema de gestión de permisos a los roles del sistema.
10. Implementación del subsistema de control de seguridad en las contraseñas de usuarios.
11. Probar en un escenario el subsistema de control de seguridad en las contraseñas de usuarios.
12. Realización de pruebas de caja negra.
13. Realización de los casos de pruebas.

Este documento se encuentra organizado en tres capítulos, a continuación se brinda una breve descripción de los mismos:

En el **primer capítulo "Fundamentación teórica",** se definen los principales conceptos investigativos asociados al dominio del problema. También se realiza un análisis completo sobre la estructura de la plataforma de la Suite de Pentaho, principalmente en el Servidor de Inteligencia de Negocios, como un análisis para definir la metodología y herramientas que se utilizarán para el desarrollo del módulo de seguridad.

En el **segundo capítulo "Descripción y análisis de la solución propuesta",** se realiza todo lo vinculado con el flujo de trabajo análisis y diseño, así como el planteamiento de la solución que se quiere desarrollar. Se identifican los requisitos funcionales y no funcionales que el sistema debe cumplir para lograr con esto una mejor comprensión de la funcionalidad e integridad del sistema. Además se describe el diseño del módulo de seguridad.

En el **tercer capítulo "Implementación y pruebas",** se hace un enfoque en la implementación y prueba de la solución propuesta. Se muestra el diagrama de despliegue para dar una visión de cómo debe quedar el despliegue de la solución. Se utilizan las pruebas de caja negra como método para realizar las pruebas de aceptación y se realizan los casos de prueba pertinentes a la implementación para lograr el desarrollo de un software con la calidad requerida.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En entornos digitales la seguridad es uno de los factores más importantes para cualquier sistema. El desarrollo de un módulo que sea capaz de erradicar las deficiencias detectadas en el Servidor de Inteligencia de Negocios de la Suite de Pentaho, lleva un conjunto de investigaciones con el fin de brindar un producto final con una elevada calidad.

En el transcurso de este capítulo, se explicará en detalle el proceso de investigación que se llevará cabo para la implementación del módulo de seguridad y se describirán los principales conceptos investigativos asociados al dominio del problema. Además se hará un análisis de las tecnologías, metodologías, lenguajes y herramientas a utilizar en el desarrollo del mismo.

1.1 Seguridad

La seguridad informática se define como el conjunto de medidas (administrativas, organizativas, físicas, técnicas legales y educativas) dirigidas a prevenir, detectar y responder a las acciones que pongan en riesgo la integridad, confidencialidad y disponibilidad, de la informatización que se procesa, intercambie, reproduzca o conserve a través de las tecnologías de la información (1).

Teniendo en cuenta el creciente uso de la informática en todas las esferas del desarrollo científico-técnico, económico, político y social a nivel mundial; así como el surgimiento de nuevos riesgos asociados principalmente con el uso de las redes de datos de alcance global, que pueden poner en peligro la seguridad de la información, resulta necesario adoptar nuevos mecanismos y tecnologías que permitan mantener un alto nivel de seguridad en los sistemas informáticos. Producto de esto, grandes empresas y compañías invierten cada vez más en el aseguramiento y protección de la información.

Existen innumerables herramientas y sistemas de seguridad orientados a preservar la disponibilidad confidencialidad e integridad de la información. La oferta en este sentido es muy numerosa y toda organización debe dedicar un esfuerzo significativo a su estudio y selección. Entre las técnicas más consolidadas se encuentran las copias de respaldo, los antivirus, los cortafuegos, los mecanismos de autenticación y la criptografía. Las copias de respaldo y en general cualquier forma de redundancia, se encaminan a garantizar la disponibilidad de los sistemas frente a cualquier eventualidad.

Los antivirus pretenden evitar la aparición de lógica maliciosa y en caso de infección tratan de eliminarla de los sistemas (2). Entre los antivirus conviene destacar aquellos que inspeccionan los correos electrónicos evitando la infección de sus destinatarios. Por su parte, los cortafuegos tratan de reducir el número de vías potenciales de acceso a los sistemas corporativos desde el exterior, estableciendo limitaciones al número de equipos y de servicios visibles. Otra de las técnicas imprescindibles en toda organización la forman los mecanismos de autenticación. Estos mecanismos pueden variar desde esquemas simples basados en los pares usuario contraseña, hasta complejos sistemas distribuidos basados en credenciales o sistemas de autenticación biométricos basados en el reconocimiento mecanizado de características físicas de las personas.

En las aplicaciones web, la seguridad es un aspecto importante a tener en cuenta, debido a que es necesario tener almacenada de manera segura toda la información que estas contienen. Estas aplicaciones son programas diseñados para usarse íntegramente en el navegador y pueden realizar las mismas funciones dinámicas que las aplicaciones para ordenador. Existen diversos mecanismos para elevar la seguridad de una aplicación web y uno de ellos es mediante la autenticación centralizada, en la que se puede usar el Servicio Central de Autenticación (CAS). Este servicio se usa en medios donde exista una arquitectura cliente-servidor, basada en el Protocolo Ligero de Acceso a Directorios (LDAP). El servicio de directorio LDAP se basa en una arquitectura cliente-servidor como se mencionaba anteriormente, donde uno o más servidores LDAP contienen los datos que forman el árbol de directorio. Como se observa en la figura 1, un cliente se conecta con un servidor LDAP y le hace una pregunta, el servidor responde con la información o con un apuntador indicando donde el cliente puede conseguir más información.

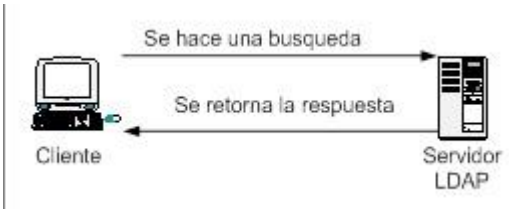


Figura 1. Esquema de consulta en un servidor LDAP

A continuación, para un mayor entendimiento, se explica de manera más amplia en que consiste este sistema de autenticación de usuarios.

1.2 Servicio Central de Autenticación (CAS)

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. Es un servicio que funciona por el protocolo HTTP (Protocolo de Transferencia de Hipertexto), con el propósito de permitir al usuario tener acceso a múltiples aplicaciones, proporcionando un nombre de usuario y contraseña una sola vez (Single SignOn (SSO)). Está implementado como varios Servlets de Java y funciona a través del servidor HTTPS. Se accede a través de tres direcciones URL (Localizador Uniforme de Recursos): la URL de login, la de validación y la de logout, utiliza los tickets (Ticket es un número de caracteres único e irrepetible, generado por el servidor CAS), como medio de autenticación, y no pueden ser fácilmente falsificados ya que únicamente el servidor que los genera puede reconocerlos como válidos.

El proceso Single SignOn (SSO), se realiza cuando el usuario se autentica al servidor CAS y solo necesita 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.

La implantación de este sistema permite obtener todo el mecanismo de seguridad de autenticación de los usuarios implementado en el servidor LDAP de la entidad.

1.3 Suite de Pentaho

La Suite de Pentaho es una plataforma de Inteligencia de Negocios (Business Intelligence(BI)), orientada a la solución y centrada en procesos, que incluye los principales componentes requeridos para implementar soluciones y ha sido concebido desde el principio para estar basada en procesos.

Las soluciones que Pentaho ofrece se componen fundamentalmente de una infraestructura de herramientas de análisis e informes integrados con un motor de flujo de trabajo en procesos de negocio. La plataforma será capaz de ejecutar las reglas de negocio necesarias, expresadas en forma de procesos y actividades, además de presentar y entregar la información adecuada en el momento preciso.

Dicha plataforma cubre las necesidades de análisis de los datos y los informes empresariales. Las soluciones y el ambiente de implementación están basados en Java. Esto convierte a la Suite de Pentaho en una solución flexible para cubrir una amplia gama de necesidades empresariales.

Entre las herramientas que componen la Suite de Pentaho se encuentran:

- ✓ **Pentaho Data Integration**
- ✓ **Pentaho Reporting**
- ✓ **Mondrian**
- ✓ **Pentaho BI Server**

Pentaho Data Integration: es una de las soluciones más utilizadas en el proceso de extracción, transformación y carga de datos (Extract, Transform and Load (ETL)) por la comunidad de software libre. Cuenta con una abundante documentación, solidez y robustez que le hace una herramienta recomendable. Permite realizar transformaciones y trabajos de una forma muy sencilla e intuitiva. Igualmente los proyectos realizados con esta herramienta son fáciles de mantener (3).

Pentaho Reporting: es una solución integrada en la Suite de Pentaho para el desarrollo de informes que incluye tres herramientas con diferentes enfoques y dirigidos a diferentes tipos de usuarios.

1. ***Pentaho Report Designer:*** es un editor basado en eclipse con prestaciones profesionales de calidad y con capacidad de personalización de informes a las necesidades de negocio destinado a desarrolladores. Incluye asistentes para facilitar la configuración de propiedades. Está estructurado de forma que los desarrolladores pueden acceder a sus prestaciones de forma rápida, además incluye un editor de consultas para facilitar la confección de los datos que serán utilizados en un informe.

2. ***Pentaho Report Design Wizard:*** herramienta de diseño de informes, que facilita el trabajo y permite a los usuarios obtener resultados de forma inmediata. Está destinada a usuarios con menos conocimientos técnicos.

A través de pasos sencillos permite:

- ✓ Conectarse a todo tipo de bases relacionales.

- ✓ Integrar el resultado dentro del portal de Pentaho.
- ✓ Posibilidad de montar codificación semafórica.

3. ***Web ad-hoc reporting***: es similar a la herramienta anterior pero vía Web. Extiende la capacidad de los usuarios finales para la creación de informes a partir de plantillas pre configuradas y siguiendo un asistente de creación (4).

Mondrian: es un servidor de código abierto que gestiona la comunicación entre una aplicación que maneja información multidimensional mediante el uso de OLAP (Procesamiento Analítico En Línea) y una base de datos relacional, y además permite a los usuarios analizar grandes cantidades de información en tiempo real (5).

Servidor de Inteligencia de Negocios (**Pentaho BI Server):** es una aplicación implementada en Java que permite realizar análisis de información mediante el uso de técnicas de inteligencia de negocios. Cuenta con una interfaz de usuario donde se encuentran disponibles todos los informes, vistas OLAP, cuadros de mando y accesos. Posee una consola de administración que posibilita gestionar y supervisar la aplicación, los usuarios registrados, los roles, los informes vistos por cada usuario cuando se han consultado y el rendimiento de la aplicación (6).

1.4 Pentaho BI Server versión Comercial

Pentaho BI Server en su versión Comercial agrega funcionalidades mejoradas y una mayor facilidad de uso e implementación. También provee soporte técnico profesional para la herramienta con altos niveles de servicio a un bajo costo de suscripción. A continuación se muestra una comparativa con respecto a la versión comunitaria:

Tabla 1. Comparación entre versión Comunitaria y versión Comercial

Software y Servicios	Edición Comunitaria	Edición Comercial
Reporting	✓	✓
Integración de datos (ETL)	✓	✓

Soporte profesional	✓
Documentación	✓
Single Sign-On	✓
Control de rendimiento	✓
Servicios de repositorio	✓

EL Pentaho BI Server en su versión 3.10 presenta significativas mejoras y ventajas entre ellas: el trabajo con nuevos reportes interactivos, capacidades enriquecidas para la exploración de datos y una interfaz de usuario totalmente rediseñada. La funcionalidad más destacada de esta nueva versión es el nuevo diseño web de reportes interactivos, el cual provee una interfaz intuitiva y potente para usuarios de negocios no técnicos, que les permite satisfacer sus propios requerimientos de información sin tener que recurrir a otras áreas. A esto se suma las fuertes capacidades del reportador operativo de Pentaho, y complementa las capacidades de análisis de datos, integración de datos, todas con el beneficio de hacerlo bajo un esquema de suscripción de bajo costo. Además alberga la información para su funcionamiento en una base de datos SQL Lite, que se carga en tiempo de ejecución y una consola de administración para el manejo de los usuarios y roles que interactúan con el sistema. A continuación se expone una breve explicación de estas herramientas.

Lenguaje de Consulta Estructurado Hyperthreaded (HSQL)

Pentaho BI Server utiliza una base de datos de tipo HSQL en la cual se almacena toda la información necesaria para el funcionamiento de la plataforma, parámetros de conexión y datos de los usuarios. HSQL es un Sistema Gestor de Base de Datos (SGBD) libre y escrito en Java que brinda algunas ventajas tales como permitir ejecutar varias aplicaciones a la vez y sentencias SQL interactivas, así como usar plugins de Eclipse para manipular directamente la base de datos. Toda la información que maneja este SGBD se encuentra almacenada en ficheros de textos planos sin protección, poniendo en riesgo los datos y permitiendo el fácil acceso a usuarios no autorizados (7).

Consola de administración

La consola de administración del Pentaho BI Server proporciona una ubicación central desde la cual permite administrar las implementaciones, los trabajos de programación y gestión de servicios. Además posee una funcionalidad adicional que le permite controlar el desempeño y monitorear remotamente la actividad, verificar las conexiones, la configuración de pruebas, entre otras ventajas.

La funcionalidad de esta consola es limitada, pues no permite simplificar muchas tareas administrativas comunes como es el caso de la gestión de permisos a los roles. Esta función no se realiza de una forma óptima, para esto el usuario otorga los permisos de forma manual y los datos introducidos tienen que ser exactos y descritos textualmente.

1.5 Tecnologías y herramientas aplicadas en el desarrollo del módulo

Para la elaboración de este módulo de seguridad se realizará un estudio profundo acerca de las diferentes metodologías, lenguajes y herramientas existentes en la actualidad, con el objetivo de desarrollar un software que posea la calidad requerida. A continuación se justifican la tecnologías escogidas para la confección de este módulo de seguridad.

1.5.1 Metodologías de desarrollo de software

Las metodologías de desarrollo de software son una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. Tienen como objetivo garantizar la eficiencia en el cumplimiento de los requisitos. Para lograr esto es necesario adoptar un conjunto de pasos y procedimientos que permitan confeccionar y plantear el proceso de desarrollo de software. Entre las metodologías más conocidas se pueden mencionar a Microsoft Solution Framework (MSF), Proceso Racional Unificado (RUP) y Programación Extrema (XP). Esta última es la metodología seleccionada para la planificación y desarrollo del módulo de seguridad.

Se escogió XP en busca de simplificar el desarrollo del software y reducir el costo del proyecto. Al mismo tiempo combina las que han demostrado ser las mejores prácticas de desarrollo de software. Conjuntamente se rediseñará todo el tiempo, dejando el código siempre en el estado más simple posible. Se harán pruebas no sólo de cada nueva clase, sino que también los clientes comprobarán que el proyecto va satisfaciendo sus necesidades. Las iteraciones serán radicalmente más cortas de lo que es usual en otros métodos, esto permite beneficiarse de la retroalimentación tan a menudo como sea posible.

Programación Extrema (XP)

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. XP está pensado para equipos pequeños de diez personas, encargados de desarrollar software en proyectos cuyos requerimientos son ambiguos o muy volátiles.

Esta metodología define cuatro valores: comunicación, retroalimentación, simplicidad y coraje. Construye sobre ellos una docena de prácticas que los proyectos XP deben seguir. Muchas de estas prácticas son técnicas antiguas, tratadas y probadas, aunque a menudo olvidadas por muchos, incluyendo la mayoría de los procesos planeados. Construye un proceso de diseño evolutivo que se basa en construir un sistema simple en cada iteración. Todo el diseño se centra en la iteración actual y no se hace nada anticipadamente para necesidades futuras. El resultado es un proceso de diseño disciplinado, que combina la disciplina con la adaptabilidad de una manera que indiscutiblemente la hace la más desarrollada de entre todas las metodologías adaptables (8).

Las historias de usuarios

Son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuarios es dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Proceso XP

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el

software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

1.5.2 Lenguaje de modelado

El lenguaje de modelado se puede conceptualizar 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.

Lenguaje Unificado de Modelado (UML)

El lenguaje de modelado UML, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un plano del sistema (*modelo*), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables (9).

1.5.3 Herramientas Case

La industria informática diseñó un soporte automatizado para el desarrollo y mantenimiento del software. Este es llamado Ingeniería de Software Asistida por Computadora (Computer Aided Software Engineering (CASE)). Se pueden definir las herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida de desarrollo del software. Entre las más utilizadas se encuentran Rational Rose y Visual Paradigm (10).

Visual Paradigm

Se seleccionó Visual Paradigm por ser una herramienta multiplataforma, además de poseer un entorno de creación de diagramas para UML 2.1. Soporta una gama de lenguajes en la generación de código e ingeniería inversa como son: Java, C++, PHP, Ada, Python, C#, Ruby, Delphi y Perl, facilitando el desarrollo del producto. Soporta el ciclo de vida completo de desarrollo del software, permitiendo crear

todo tipo de diagramas UML. Es de fácil uso y posibilita generar código desde diagramas y documentación, agilizando el trabajo del desarrollador (11).

1.5.4 Sistema Gestor de Bases de Datos (SGBD)

Los Sistemas 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 (12).

Un SGBD brinda diversas ventajas, entre las que se destacan: el manejo de grandes volúmenes de información de una forma óptima y la organización de los datos con un impacto mínimo en el código de los programas. Además ofrecen recursos para definir y garantizar el cumplimiento de las restricciones de integridad, disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos. También poseen mecanismos de protección, asegurando la integridad, confidencialidad y seguridad de la información.

PostgreSQL

Es un SGBD relacional orientado a objetos, utilizado por programadores que realizan aplicaciones cliente servidor complejas o críticas. Es capaz de ajustarse al número de microprocesadores y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole posible soportar una mayor cantidad de peticiones simultáneas de manera correcta.

Este SGBD posee numerosas ventajas entre las cuales se encuentran:

- ✓ Permite un modelado más natural de la realidad.
- ✓ Facilita la reutilización de componentes de software.
- ✓ Ofrece mecanismos de abstracción para mantener controlable la construcción de sistemas complejos (13).

1.5.5 Servidor Web

Se puede definir como servidor web un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet.

El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados.

Apache Tomcat

Es un servidor web con soporte de *Servlets* y *JSPs*. Un contenedor de Servlets es un *Shell* de ejecución que maneja e invoca *Servlets* por cuenta del usuario. Tomcat no es un servidor de aplicaciones, como *JBoss* o *JOnas*. Incluye el compilador *Jasper*, que compila *JSPs* convirtiéndolas en *Servlets*. El motor de *Servlets* de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat 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. Este 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 (14).

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

1.5.6 Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado (IDE) es un programa compuesto por un conjunto de herramientas para un programador utilizarlas, puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. El entorno de desarrollo es imprescindible en la producción de un software. Es donde se definen el conjunto de herramientas, tecnologías y versiones a usar que intervienen en un proceso de desarrollo del software. A continuación se presentan herramientas utilizadas en el desarrollo de este trabajo, se exponen sus características y ventajas para tener un mayor conocimiento de su utilidad.

Eclipse

Eclipse es principalmente una plataforma de programación, sobre la que se puede montar herramientas de desarrollo de cualquier lenguaje mediante la implementación de los *plugins* adecuados. La arquitectura de *plugins* de Eclipse permite además introducir aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo, como herramientas UML, editores visuales

de interfaces y otros. Además permite crear entornos de desarrollos integrados para distintos lenguajes de programación. Eclipse dispone de un editor de texto con resaltado de sintaxis y la compilación es en tiempo real (15).

1.5.7 Lenguajes de programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Una característica relevante de los lenguajes de programación es precisamente que más de un programador pueda usar un conjunto común de instrucciones que sean comprendidas entre ellos para realizar la construcción de un programa de forma colaborativa.

Java Web

Java es un lenguaje de programación orientado a objetos, desarrollado por *Sun Microsystems* a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema, ya que esta es gestionada por el propio lenguaje y no por el programador (16).

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible.

Con la programación en Java, se pueden realizar distintos aplicativos, como son *applets*, que son aplicaciones especiales, que se ejecutan dentro de un navegador al ser cargada una página HTML en un servidor web. Por lo general los applets son programas pequeños y de propósitos específicos.

La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente y de esta forma logra distribuir el trabajo a realizar.

Algunas características de este lenguaje se describen a continuación:

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.[8]

Distribuido: Java se ha construido con extensas capacidades de interconexión *TCP/IP*. Existen librerías de rutinas para acceder e interactuar con protocolos como *HTTP* y *FTP*. La característica de ser distribuido es debido a que proporciona las librerías y herramientas para que los programas 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 máquina 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 *bytecode* 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.

1.5.8 Sistemas de autenticación centralizados

En los sistemas informáticos utilizados por las organizaciones actuales, los usuarios deben dar pruebas de quiénes son, para así verificar si pueden acceder a cierto tipo de información de la compañía. Hay archivos, datos, documentos, gráficos y otros secretos empresariales que sólo ciertos empleados privilegiados pueden conocer. El problema radica, en que la mayoría de los casos, los usuarios deben usar tantos mecanismos de autenticación como sistemas de información tiene la compañía, lo cual es bueno para mantener fuera a los invasores, pero bastante incómodo y laborioso para los autorizados Existen dos modelos de autenticación, centralizado y descentralizado. En el

modelo descentralizado, cada servicio de la red maneja sus claves de forma independiente, por ejemplo, los usuarios de *Oracle*, los usuarios de un firewall, los administradores de un sitio web; cada uno de estas aplicaciones maneja por separado sus claves y las mismas no son compartidas.

En la autenticación centralizada los usuarios y sus claves se almacenan en un repositorio central. Las diferentes aplicaciones que utilizan dicho repositorio se configuran para establecer conexión con este y realizar el proceso de autenticación. Las claves estarán ubicadas dentro de un servidor de directorio *LDAP* (protocolo cliente servidor hecho para acceder a un servicio de directorio), pero en general podrían estar almacenadas en un archivo de texto plano o en una base de datos relacional, entre otros métodos de almacenamiento de información.

1.5.8.1 Sistemas de gestión de seguridad utilizados a nivel mundial

Desde el punto de vista de la seguridad, es recomendable disponer de un servicio centralizado de autenticación. Históricamente en el mundo *Unix* se ha utilizado el *Servicio de Información de Red* (NIS), pero actualmente ya se encuentra en desuso y se recomienda el uso del *Protocolo Ligero de Acceso a Directorios* (LDAP). Otro sistema que se destaca a nivel mundial es el conocido como *Microsoft Forefront*, que ofrece protección y control de la seguridad en la infraestructura informática orientada a empresas. Su completo conjunto de productos de seguridad, que se integran entre sí y con la infraestructura informática de la empresa, puede complementarse y operar con soluciones de terceros. De esta manera, se obtienen soluciones de seguridad muy completa y de defensa en profundidad.

1.5.8.2 Sistemas de gestión de seguridad utilizados en Cuba y en la UCI

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.

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.

En la UCI, existe un subsistema de gestión de la seguridad implantado en el Proyecto Akademos, asegurando la confidencialidad de la información con carácter docente manejada en el proyecto

anteriormente mencionado, reduciendo costes asociados a pérdidas de información por incidentes de seguridad y brindando servicios a los demás módulos pertenecientes al sistema.

El Sistema Central de Autenticación (CAS), es el sistema de gestión de seguridad que se emplea en toda la universidad para el control y el manejo de los usuarios registrados.

Conclusiones

En este capítulo se plantean los principales conceptos relacionados con los procesos para el control de la seguridad en los sistemas informáticos, que permitieron obtener los elementos teóricos necesarios para el desarrollo de la investigación. La sistematización del estudio de las principales tecnologías y herramientas para el desarrollo del módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite de Pentaho, permitió seleccionar las adecuadas para su implementación.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

Introducción

En este capítulo se muestra todo lo relacionado con el flujo de trabajo análisis y diseño. Se realiza además el levantamiento de los requerimientos funcionales y no funcionales que el sistema debe cumplir. 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 del sistema.

2.1 Propuesta de la solución

El desarrollo de este módulo tiene como objetivo simplificar algunas de las deficiencias que presenta el Servidor de Inteligencia de Negocios. Lo primero será aumentar el nivel de fortaleza de las contraseñas en un ambiente no centralizado de autenticación de usuario. También se realizará la migración de la información del sistema, los usuarios, conexiones y datos de configuración almacenada en el fichero de texto plano, al SGBD PostgreSQL, aumentando con esto la integridad de los datos. Además, se realizará el proceso de gestión automática de los permisos a los roles, haciendo menos engorroso el trabajo y menos costoso en tiempo esta tarea. Por último se realizará la integración del Pentaho BI Server con el CAS en un ambiente centralizado de autenticación de usuario.

2.2 Requerimientos

Los requerimientos son una descripción de las necesidades o deseos de un producto. La meta principal en esta etapa es identificar y documentar lo que en realidad se necesita, en una forma en que pueda fácilmente ser transmitido al cliente y al equipo de desarrollo.

2.2.1 Requisitos Funcionales

Los requerimientos funcionales son capacidades o condiciones que un sistema determinado debe cumplir. Seguidamente se enumeran los que se han identificado para el desarrollo de este módulo.

RF 1 Autenticar Usuario.

RF 1.1 Validar datos introducidos.

RF 2 Migrar la información del sistema, los usuarios, conexiones y datos de configuración al Sistema Gestor de Base de Datos PostgreSQL.

RF 3 Cargar fichero XML con la estructura dimensional de un almacén de datos.

RF 3.1 Validar fichero.

RF 4 Administrar rol.

RF 4.1 Adicionar rol

RF 4.2 Modificar rol.

RF 4.3 Definir los permisos de un rol.

RF 4.4 Eliminar rol.

2.2.2 Requisitos no funcionales

Los requerimientos no funcionales detallan las propiedades o cualidades que el producto debe tener, aumentándole funcionalidad al sistema pues hacen al producto atractivo, fácil de usar, rápido y confiable. Los requisitos no funcionales se encuentran separados por categorías.

Apariencia o interfaz externa:

- ✓ Se deben utilizar imágenes y colores identificados con el sistema.
- ✓ La interfaz externa debe estar diseñada para verse en cualquier resolución, pero se recomienda 1024x768 px.

Usabilidad:

- ✓ Utilizar un patrón de navegación que permita estructurar y hacer intuitivo el acceso a las funcionalidades del sistema.

Accesibilidad:

- ✓ La información y las funcionalidades estarán disponibles y el usuario podrá acceder a ellas en todo momento.

Rendimiento:

- ✓ El sistema de seguridad permitirá que múltiples usuarios con diferentes roles estén trabajando a la vez.
- ✓ Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos.

Legales:

- ✓ Las herramientas seleccionadas están respaldadas por licencias libres, bajo las condiciones de software libre.

Seguridad:

- ✓ Solo se podrán autenticar usuarios que tengan un nivel de fortaleza elevado en las contraseñas.
- ✓ Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- ✓ Restringir el acceso a la información mediante los diferentes roles.

Software:

- ✓ Para el cliente:

  Navegador Web

- ✓ Para el servidor

  Linux en cualquiera de sus distribuciones y Windows XP o superior

  Servidor Apache Tomcat 6.0 o superior

  Servidor PostgreSQL 8.4 o superior

  Máquina Virtual de Java (JDK 6)

Hardware:

- ✓ Para el cliente:

  Requerimientos mínimos: Procesador Dual Core a 2.0GHZ con 1G de memoria RAM.

- ✓ Para el servidor:

Requerimientos mínimos: Procesador Dual Core a 2.0GHZ con 1G de memoria RAM.

2.3 Actores del sistema

Un actor es un conjunto coherente de roles que los usuarios de casos de uso desempeñan cuando interaccionan con estos casos de uso. En la siguiente tabla se muestra la definición del actor Usuario, así como su descripción:

Tabla 2. Justificación del actor del sistema

Actor	Descripción
Usuario	Es el encargado de darle permiso a los diferentes roles e interactuar con el sistema.

2.4 Fase de exploración

La exploración es la fase donde se especifican los requisitos no funcionales de la aplicación informática y donde se aplicarán diferentes iteraciones. Para hacerlo será necesaria la existencia de reglas que se han de seguir por las partes implicadas en el proyecto para que todas las partes tengan voz y se sientan realmente partícipes de la decisión tomada. En esta fase se elaboran las historias de usuarios que es uno de sus artefactos más importantes así como el plan de iteraciones y el plan de entregas.

2.4.1 Historias de usuarios (HU)

Las historias de usuarios tienen el mismo propósito que los casos de uso. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema. Las historias de usuarios son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario. También conducirán el proceso de creación de los test de aceptación (empleados para verificar que las historias de usuarios han sido implementadas correctamente). Existen diferencias entre estas y la especificación de requisitos tradicional. La principal diferencia es el nivel de detalle. Las historias de usuarios solamente proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario.

A continuación se muestra la historia de usuario “Autenticar usuario”, el resto de las historias se encuentran expuestas en los anexos:

Tabla 3. HU Autenticar usuario

Historia de Usuario	
Número: 1	**Nombre de la Historia de Usuario:** Autenticar Usuario
Cantidad de modificaciones a la Historia de Usuario: 2	
Usuario: Yuniel Acebal Urtiaga Ramsés López Sosa	**Iteración asignada:**1
Prioridad en negocio: Alta	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Alta	**Puntos reales:** 2 semanas
Descripción: El usuario podrá autenticarse poniendo sus datos en los campos correspondientes. El sistema validará los datos y si son correctos mostrará la interfaz principal de acceso al Servidor de Inteligencia de Negocios.	
Observaciones: El usuario solo logrará autenticarse si los datos introducidos cumplen con lo necesario para lograrlo.	

Para el desarrollo de este módulo de seguridad se realizó una estimación de la duración para cada una de las historias de usuarios identificadas. A continuación se muestran los resultados.

Tabla 4. Estimación de duración de las HU

Historias de usuarios	Estimación
Autenticar usuario	2 semanas
Migrar la información del sistema, los usuarios, conexiones y datos de configuración al sistema gestor de base de datos PostgreSQL.	2 semanas
Cargar fichero XML con la estructura dimensional de un almacén de datos.	1 semana
Gestionar rol	3 semanas

2.4.2 Lista de reserva del producto

Una lista de reserva es una tabla donde se encierran los requisitos funcionales que el sistema a desarrollar debe cumplir ordenados por prioridad de implementación, así como los requisitos no funcionales.

Tabla 5. Lista de reserva del producto

Ítem	Descripción	Estimación	Estimado por
Prioridad: Alta			
1	Autenticar usuario.	2 semanas	Analista
Prioridad: Alta			

2	Migrar la información del sistema, los usuarios, conexiones y datos de configuración al sistema gestor de base de datos PostgreSQL.	2 semanas	Analista
Prioridad: Alta			
3	Cargar fichero XML con la estructura dimensional de un almacén de datos.	1 semana	Analista
Prioridad Alta			
4	Gestionar rol.	3 semanas	Analista
Requisitos no funcionales			
1	**Apariencia o interfaz externa:** se deben utilizar imágenes y colores identificados con el sistema. La interfaz externa debe estar diseñada para verse en cualquier resolución, pero se recomienda 1024x768 px.		Analista
2	**Usabilidad**: Utilizar un patrón de navegación que permita estructurar y hacer intuitivo el acceso a las funcionalidades del sistema.		Analista

3	**Accesibilidad**: La información y las funcionalidades estarán disponibles y el usuario podrá acceder a ellas en todo momento.		Analista
4	**Rendimiento**: El sistema de seguridad permitirá que múltiples usuarios con diferentes roles estén trabajando a la vez. Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos.		Analista
5	**Legales**: Las herramientas seleccionadas están respaldadas por licencias libres, bajo las condiciones de software libre.		Analista
6	**Seguridad**: Solo se podrán autenticar usuarios que tengan un nivel de fortaleza elevado en las contraseñas. Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. Restringir el acceso a la información mediante los diferentes roles.		Analista
7	**Software**: Para el cliente: ✓ Navegador Web.		Analista

	Para el servidor: ✓ Linux en cualquiera de sus distribuciones y Windows XP o superior. ✓ Servidor Apache Tomcat 6.0 o superior. ✓ Servidor PostgreSQL 8.4. ✓ Máquina Virtual de Java (JDK 6).		
8	**Hardware:** Para el cliente: ✓ Requerimientos mínimos: Procesador Dual Core a 2.0GHZ con 1G de memoria RAM. Para el servidor: ✓ Requerimientos mínimos: Procesador Dual Core a 2.0GHZ con 1G de memoria RAM.		Analista

2.4.3 Tareas de ingeniería

Se evalúa cada escenario, entiéndase por escenario historias de usuarios o requisitos y lo divide en tareas donde cada una de estas representa una característica del sistema. La HU número uno fue dividida en dos tareas de ingeniería, la HU número dos en dos tareas de ingeniería y la tercera HU en tres tareas de ingeniería para facilitar la implementación del módulo de seguridad. A continuación se muestra la tarea de ingeniería “Validar datos” que pertenece a la HU número uno, el resto de las tareas se encuentran descritas en los anexos:

Tabla 6. Tarea de ingeniería

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Validar datos introducidos	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 13/2/12	**Fecha Fin:** 17/2/12
Programador Responsable: Yuniel Acebal Urtiaga Ramsés López Sosa	
Descripción: El sistema validará los datos introducidos por el usuario.	

2.5 Fase de planificación

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. El proyecto comienza recopilando “historias de usuarios”, las que sustituyen a los tradicionales “casos de uso”. Una vez obtenidas las “historias de usuarios”, los programadores evalúan rápidamente el tiempo de desarrollo de cada una. Si alguna de ellas tiene riesgos que no permiten establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba (*spikes*), para reducir estos riesgos. Una vez realizadas estas estimaciones, se organiza una reunión de planificación con los diversos actores del proyecto (cliente, desarrolladores, gerentes), a los efectos de establecer un plan o cronograma de entregas (*Release Plan*) en los que todos estén de acuerdo. Una vez acordado este cronograma, comienza una fase de iteraciones en la que en cada iteración se desarrolla, prueba e instala unas pocas “historias de usuarios”.

2.5.1 Plan de iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que siguientes iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección y de esta forma prever que no vuelva a ocurrir.

Una vez definidas las historias de usuarios, es necesario crear un plan de iteraciones con el objetivo de definir cuáles son las tareas con más prioridad y establecer los tiempos de implementación.

A continuación se muestra el plan de iteraciones donde queda definida en qué iteración serán desarrolladas cada una de las historias de usuarios.

Iteración 1: en esta iteración se desarrolla la historia de usuario número uno, la cual permite al usuario entrar sus datos, el sistema valida los datos introducidos y muestra la interfaz de bienvenida al Servidor de Inteligencia de Negocios.

Iteración 2: en esta iteración se desarrolla la historia de usuario número dos, en la cual se realiza la migración de toda la información del sistema, los usuarios, conexiones y datos de configuración al Sistema Gestor de Base de Datos PostgreSQL, asegurando de esta manera que todos los datos se encuentren almacenados de forma segura en una base de datos. También se desarrolla la historia de usuario número tres, en la cual se realiza la validación del fichero XML que el usuario carga, con el objetivo de evitar que se cargue un fichero de otro tipo de extensión o con errores en su estructura interna.

Iteración 3: en esta iteración se desarrolla la historia número cuatro, la cual permite adicionar, modificar, eliminar y definir los permisos de un rol.

A continuación se muestra un plan de duración de las iteraciones con el objetivo de garantizar un desarrollo exitoso y con calidad del módulo de seguridad.

Tabla 7. Plan de duración de las iteraciones

Iteración	Historia de usuario	Duración en semanas
Iteración 1	Autenticar usuario.	2 semanas
Iteración 2	Migrar la información del sistema, los usuarios, conexiones y datos de configuración al sistema gestor de base de datos PostgreSQL. Cargar fichero XML con la estructura dimensional de un almacén de datos	3 semanas
Iteración 3	Gestionar rol.	3 semanas

2.6 Diseño del software

El diseño es parte fundamental en la construcción del sistema que se pretende crear, pues es una abstracción de cómo queda el producto final. A partir del diseño se obtiene un modelo cercano a la versión final del software que se construye. El diseño del software, permite traducir los requisitos de un sistema en una representación que inicialmente da una visión del mismo y tras posteriores refinamientos sirve como esquema para la implementación.

La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo lo menos complicado posible para lograr un diseño fácilmente entendible e implementable que a la larga cueste menos tiempo y esfuerzo desarrollar.

2.6.1 Tarjetas CRC(Clases, Responsabilidades y Colaboradores)

Para el diseño de aplicaciones informáticas XP, no se requiere la presentación del sistema mediante diagramas utilizando UML como lenguaje de modelado. En su lugar se utilizan otras técnicas como las tarjetas CRC (Clases, Responsabilidades y Colaboradores).

Las tarjetas CRC representan objetos. La clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

Las tarjetas CRC están divididas en 3 partes:

- ✓ Una *clase* es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las cosas que conoce y las que realiza, sus atributos y métodos.
- ✓ Los *colaboradores* de una clase son las demás clases con las que trabaja en conjunto.
- ✓ Llevar a cabo sus *responsabilidades*.

El formato físico de las tarjetas CRC facilita la interacción entre clientes y equipo de desarrollo, en sesiones en las que se aplican técnicas de grupos como tormenta de ideas o juego de roles y se ejecutan escenarios a partir de especificación de requisitos o historias de usuarios. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones. Luego en un estado de diseño avanzado o ya en la implementación del sistema, las tarjetas CRC se convierten en clases con métodos, atributos, relaciones de herencia, composición o dependencia.

A continuación se muestra la tarjeta CRC correspondiente a la clase ReaderXML, la cual se encarga de leer el fichero XML y crear el objeto Schema:

Tabla 8. Tarjeta CRC de la clase ReaderXML

Tarjeta CRC	
Clases: ReaderXML	
Responsabilidades	**Colaboraciones**

- Cargar XML - Salvar XML	- Schema - Cube - Dimension

2.6.2 Modelo de datos del subsistema de integración con el Sistema Gestor de Base de Datos PostgreSQL

En las siguientes imágenes se muestra el modelo físico de las base de datos “hibernate” y “quartz”, en las cuales se almacena información del sistema, de usuarios, conexiones y de datos de configuración.

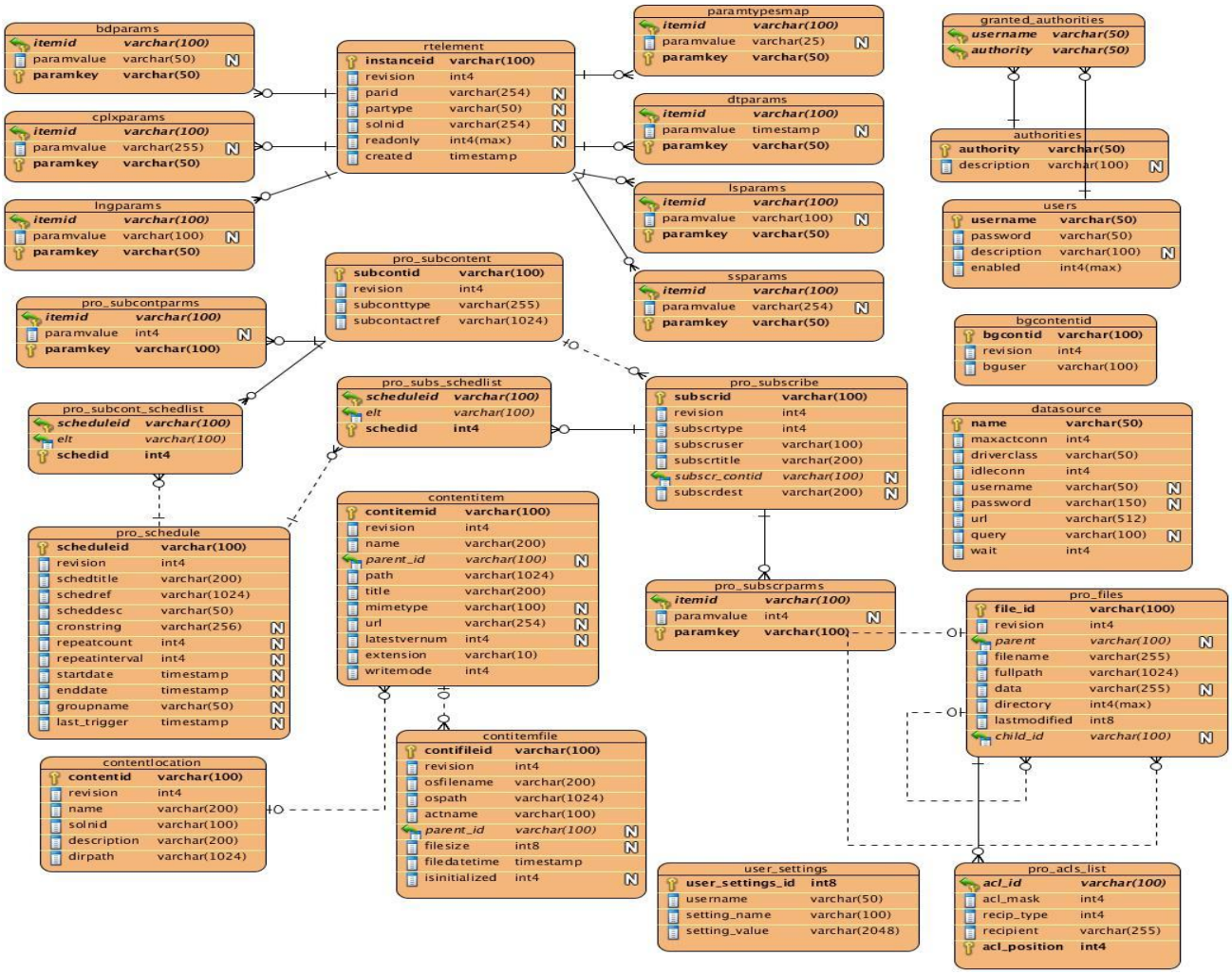


Figura 2. Modelo físico de datos de la base de datos "hibernate"

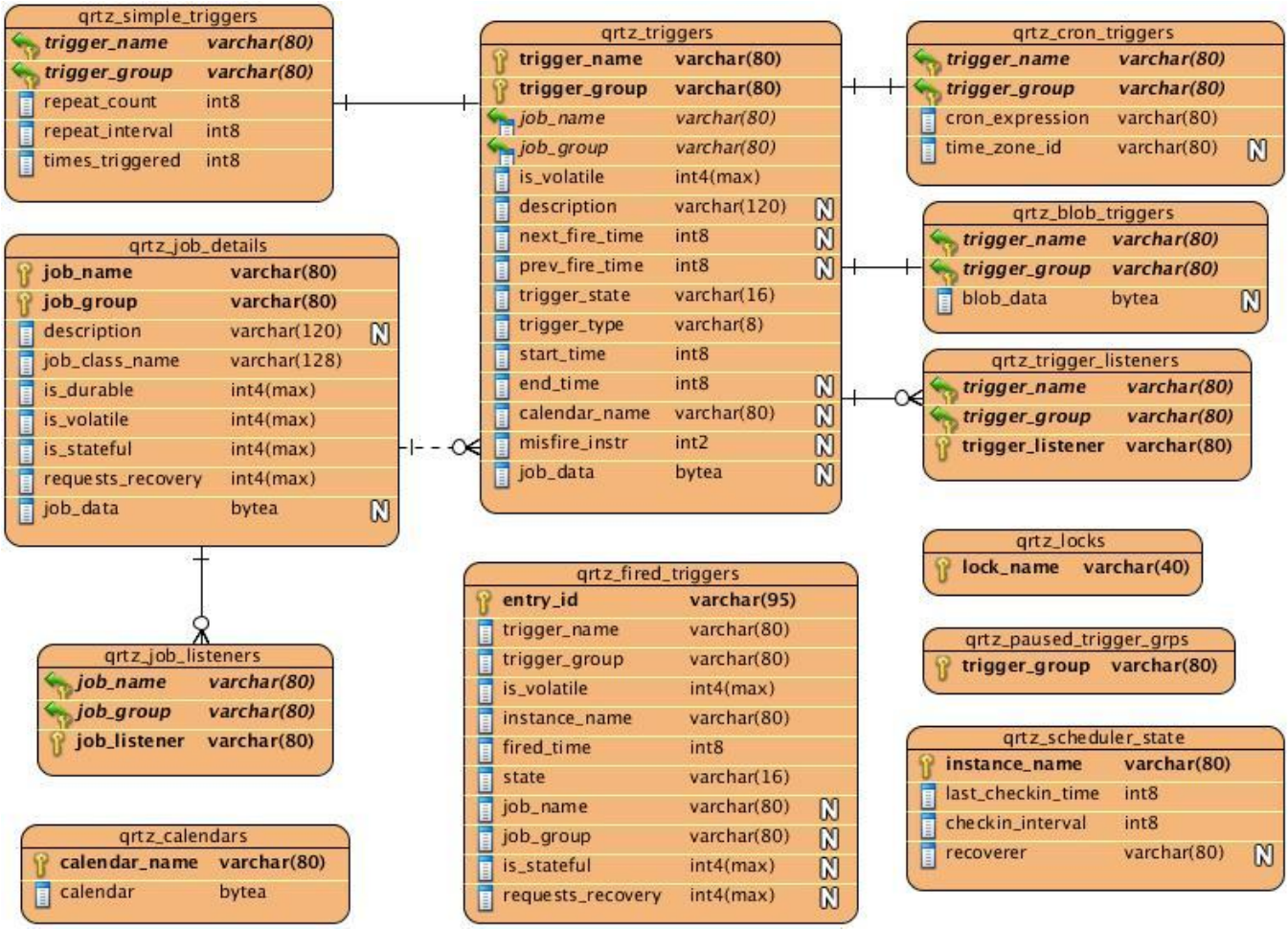


Figura 3. Modelo físico de datos de la base de datos "quartz"

2.6.3 Patrones de Diseño

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en muchas situaciones a la hora de realizar una aplicación orientada a objetos. Los patrones de diseño son relativamente fáciles de comprender, lo que a veces se hace complejo es utilizarlos. Por ello hay que conocerlos bien, especialmente los más importantes, porque realmente facilitan el trabajo y sobre todo, hacen el código más legible.

Los patrones de diseño son independientes del lenguaje que se utilice, siempre y cuando el lenguaje sea orientado a objetos (14).

Patrones generales de software para asignar responsabilidades (GRASP)

Entre los patrones GRASP que se utilizan en el diseño de la solución están:

- ✓ Experto en información
- ✓ Creador
- ✓ Alta cohesión
- ✓ Bajo acoplamiento

Experto en información: la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.

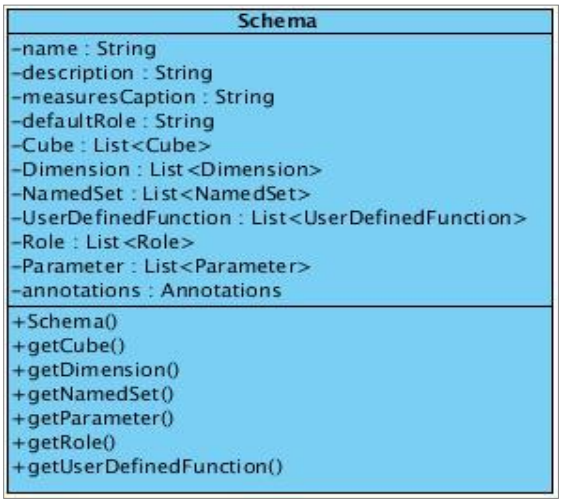


Figura 4. Ejemplo del patrón experto

Creador: este patrón como su nombre lo indica, es el que crea y guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

- ✓ B contiene a A
- ✓ B es una agregación (o composición) de A
- ✓ B almacena a A
- ✓ B tiene los datos de inicialización de A (datos que requiere su constructor)
- ✓ B usa a A

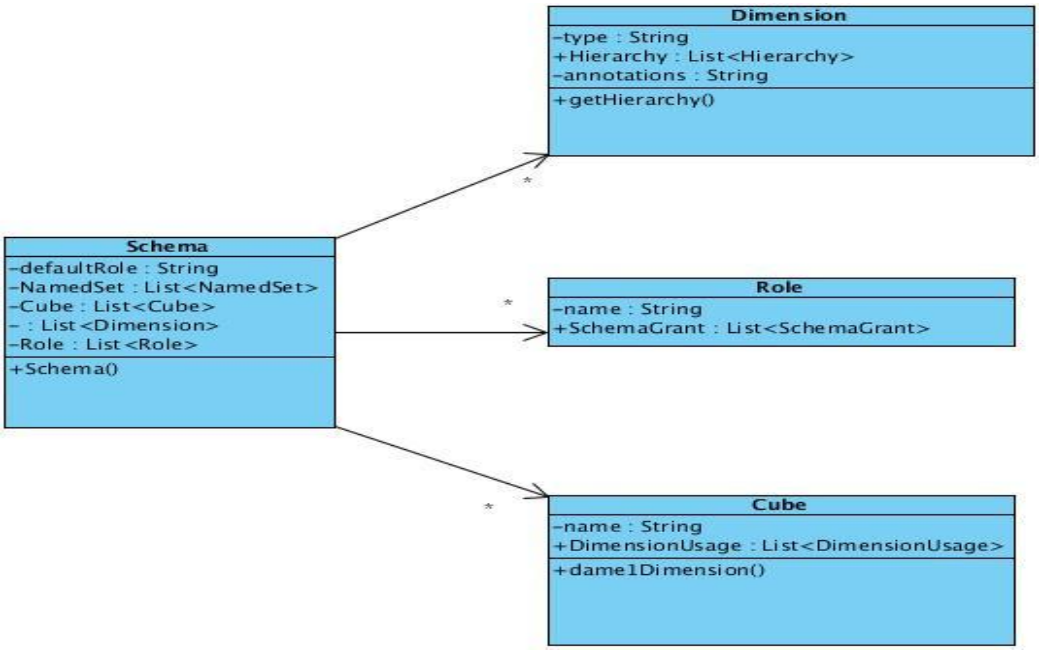


Figura 5. Ejemplo del patrón creador

Alta cohesión: la cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema no desempeñada por el resto de los elementos y auto-identificable. Una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo.

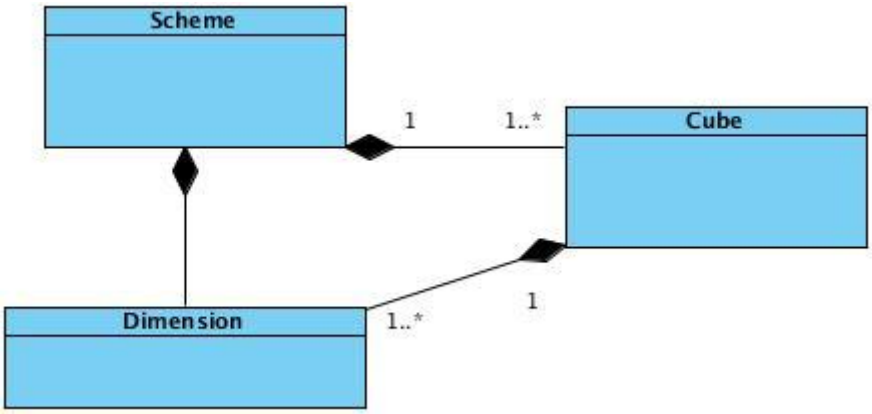


Figura 6 Ejemplo del patrón alta cohesión

Bajo acoplamiento: guía la asignación de responsabilidades relacionadas con el acoplamiento entre las clases, garantizando la menor dependencia entre cada una de ellas, de forma tal, que de producirse alguna modificación en algunas de ellas no se afecten los cambios en otros componentes, potenciando la reutilización.

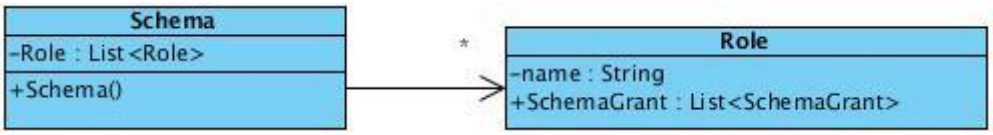


Figura 7. Ejemplo del patrón bajo acoplamiento

2.6.4 Patrones de Arquitectura

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación, se engloban dentro de la llamada Arquitectura de Software o Arquitectura Lógica. Se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para guiar el desarrollo del software dentro de un sistema informático.

Estas Arquitecturas están definidas muchas veces por el tipo de tecnología a la cual se enfrenta un programador o grupo de programadores, por lo cual algunos tipos de arquitectura son más recomendables que otras para ciertas tecnologías.

Un patrón de arquitectura de software describe un problema particular de diseño recurrente y presenta un esquema genérico de sus soluciones. El esquema de solución contiene componentes, sus responsabilidades y relaciones.

Arquitectura en tres capas

La arquitectura de una aplicación es la vista conceptual de la estructura de esta. Toda aplicación contiene código de presentación, de procesamiento y de almacenamiento de datos. La arquitectura de las aplicaciones difiere según como está distribuido este código.

La programación por capas es una forma de programar bajo un objetivo principal: que las distintas lógicas presentes en la aplicación se separen y estas posean estructuras bien planteadas.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

En el diseño de sistemas informáticos actuales se suele usar las arquitecturas multinivel o programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que

permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

Figura 8. Arquitectura en tres capas

La capa de presentación: esta capa se encarga de proveer una interfaz entre el sistema y el usuario. Básicamente, se responsabiliza de que se le comunique información al usuario por parte del sistema y viceversa, manteniendo una comunicación exclusiva con la capa de negocio que veremos a continuación. Además dentro de esta capa entraría aquello que el usuario ve cuando se conecta a la aplicación.

La capa de negocio: es la capa que contiene los procesos a realizar con la información recibida desde la capa de presentación, las peticiones que el usuario ha realizado y responsabilizándose de que se le envíen las respuestas adecuadas a la capa de presentación. Podríamos verla como una capa intermedia, a medio camino entre la capa de presentación y la capa de datos, puesto que se relaciona con ambas y por supuesto, procesa también la información devuelta por la capa de datos.

La capa de acceso a datos: es la capa donde se almacenan los datos. Mediante la capa de negocio, se puede encargar de ofrecer, modificar, almacenar, borrar y recuperar datos, mediante el gestor (o los gestores) de bases de datos que la aplicación requiera.

Una de las ventajas de la arquitectura en tres capas es que separa la interfaz de usuario de la lógica de la aplicación. Esta separación permite tener diferentes presentaciones accediendo a la misma lógica. La redefinición del almacenamiento de información no tiene influencia sobre la presentación.

2.6.5 Modelo despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes de hardware y software en el sistema final. Es un grafo de nodos unidos por conexiones de comunicación donde cada nodo puede contener instancias de componentes. En general un nodo puede ser una unidad de computación de algún tipo, desde un sensor hasta un mainframe y las instancias de componentes de software pueden estar unidas por relaciones de dependencia.

A continuación se muestra la vista de despliegue del módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite de Pentaho.

Figura 9. Modelo de despliegue

La vista de despliegue está conformada por un nodo *PC Cliente* que representa las estaciones de trabajo desde donde se conectan los clientes o usuarios al servidor de inteligencia de negocios usando el protocolo HTTP al nodo servidor web Apache Tomcat. En dicho servidor se encuentra desplegada la aplicación web y se conecta usando el protocolo TCP/IP al Servidor de Base de Datos. En este nodo se encuentran almacenados todos los datos persistentes que necesita la aplicación para su funcionamiento y para dar respuesta a todas las peticiones de los usuarios.

Conclusiones

En este capítulo se definieron las principales características que poseerá el módulo de seguridad que se desea desarrollar. También se realizó el levantamiento de requisitos y se definió un plan de iteraciones con el objetivo de definir cuáles historias de usuarios serán desarrolladas en cada iteración. Se diseñaron las tarjetas CRC para facilitar la interacción entre el cliente y el equipo de desarrollo. Además, se expusieron los patrones tanto arquitectónicos como de diseño que se utilizarán en el desarrollo de la aplicación informática y se diseñó el modelo de despliegue para dar una visión de cómo quedará desplegada la aplicación una vez finalizada su implementación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

Introducción

En este capítulo se expone todo lo relacionado con los flujos de trabajo implementación y pruebas, cerrando con su realización el desarrollo del módulo de gestión de seguridad para el servidor de inteligencia de negocios.

3.1 Implementación

El cliente es una parte más del equipo de desarrollo, su presencia es indispensable en las distintas fases de XP y aún más necesaria a la hora de codificar una historia de usuario. Los clientes son los que crean las historias de usuarios y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario, el cliente debe especificar detalladamente lo que esta hará y también tendrá que estar presente cuando se realicen los *test* que verifiquen que la historia implementada cumple la funcionalidad especificada.

La codificación debe hacerse ateniendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión.

Realizar pruebas que verifiquen el funcionamiento de los distintos códigos implementados, ayudará a desarrollarlos. Crear estas pruebas antes, ayuda a saber qué es exactamente lo que tiene que hacer el código a implementar y se sabrá que una vez implementado pasará dichos *test* sin problemas. Se puede dividir la funcionalidad que debe cumplir una tarea a programar en pequeñas unidades, de esta forma se crearán primero los *test* para cada unidad y a continuación se desarrollará dicha unidad, así poco a poco se conseguirá un desarrollo que cumpla todos los requisitos especificados.

Como ya se comentó anteriormente, XP opta por la programación en pareja ya que permite una implementación eficiente y con gran calidad. La metodología XP sugiere un modelo de trabajo usando repositorios, donde las parejas de programadores publican cada pocas horas sus códigos implementados y corregidos junto a los *test* que deben pasar. De esta forma el resto de programadores que necesiten códigos ajenos, trabajarán siempre con las últimas versiones.

La metodología XP también propone un modelo de desarrollo colectivo en el que todos los programadores están implicados en todas las tareas, cualquiera puede modificar o ampliar una clase o método de otro programador si es necesario y subirla al repositorio de código. El permitir al resto de los

programadores modificar códigos que no son suyos no supone ningún riesgo, ya que para que un código pueda ser publicado en el repositorio tiene que pasar los *test* de funcionamiento definidos para el mismo.

Dicha metodología afirma que la mayoría de los proyectos que necesiten más tiempo extra que el planificado para ser finalizados, no podrán ser terminados a tiempo aunque se añadan más desarrolladores y se incrementen los recursos. La solución que plantea XP es realizar un nuevo "*Ralease plan*" para concretar los nuevos tiempos de publicación y de velocidad del proyecto (6).

3.1.1 Estándares de codificación

Se entiende como estándar de codificación, a un estilo de programación homogéneo en un proyecto que permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.

En general, un estándar de codificación son reglas que se siguen para la escritura del código fuente y con esto facilitar a otros programadores entender el código (cómo identificar las variables, las funciones o métodos y declaraciones de clases).

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador lo hubiera escrito todo de una sola vez. Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada.

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores o mejorar el rendimiento. Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad, es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continua un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas y posteriormente se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

Estas son algunas de las razones que conllevan a la utilización de un estándar de codificación:

- ✓ Eleva la mantenibilidad del código.
- ✓ Sirve como punto de referencia para los programadores.
- ✓ Mantiene un estilo de programación.
- ✓ Ayuda a mejorar el proceso de codificación.

Las convenciones de código mejoran la lectura del software, permitiendo entender código nuevo mucho más rápidamente y más a fondo. Algunas de estas convenciones son:

Comentarios

Existen dos tipos de comentario, los de implementación y los de documentación, estos últimos existen solo en Java y se encuentran limitados por /**…*/.

De implementación:

```
//-------->Atributos para MemberGrant
xs.useAttributeFor(MemberGrant.class,"member"); //Atributo member de la clase MemberGrant
xs.useAttributeFor(MemberGrant.class,"access"); //Atributo access de la clase MemberGrant
```

De documentación:

```
//Conexión a la BD
/*parámetros para la conexión*/
String driver = "org.postgresql.Driver";
String url = "jdbc:postgresql://localhost/"+BaseDatos.getNombrebd();
String usuario = Conexcion.getUsuario();
String clave = Conexcion.getClave();

/*procedimiento de la conexión*/
try{
```

Declaraciones de variables

Las variables son declaradas en una sola línea y el nombre estará siempre escrito en letra minúscula.

Ejemplo:

```
private static String usuario;
private static String clave;
private static Dimension dimension;
private static int posnombrehierar;
private static int posierarchy;
private static int posaux;
private static int memberpos;
private static String xml;
```

Declaraciones de clases

Las clases comienzan con mayúscula, si la clase es compuesta, empiezan con mayúsculas las dos letras iniciales de cada palabra y separadas por un guión bajo.

Al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:

- ✓ No debe existir ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.
- ✓ La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.
- ✓ La llave de cierre "}" empieza una nueva línea para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{".
- ✓ Los métodos se separan tres líneas en blanco.

```
package load;

public class BaseDatos {

    private static String nombrebd;


    public BaseDatos() {
        // TODO Auto-generated constructor stub
    }
    public static String getNombrebd() {
        return nombrebd;
    }

    public static void setNombrebd(String nombrebd) {
        BaseDatos.nombrebd = nombrebd;
    }
}
```

Sentencias "*for*"

Las declaraciones de un ciclo *for* tendrán dos estructuras diferentes:

1-

```
for (Cube cubo : listaCubos) {
    String name = cubo.getName();
    out.print("<option value=" + name + ">" + name + "</option>");
}
```

2-

```
for (int x=1;x<=rs.getMetaData().getColumnCount();x++){
   out.print("<td>"+rs.getMetaData().getColumnName(x)+"</td>");

}
```

3.1.2 Implementación del subsistema de fortalecimiento de las contraseñas en un ambiente no centralizado de autenticación de usuario

Para aumentar el nivel de seguridad de las contraseñas de los usuarios del sistema, se decide crear un fichero "javascript" con funcionalidades que realicen el proceso de verificación de la fortaleza de las contraseñas de cada usuario que se crea, con el objetivo de que cumpla con las siguientes características:

- ✓ Mínimo número de caracteres 8

- ✓ Al menos 1 Letra Mayúscula
- ✓ Al menos 1 letra Minúscula
- ✓ Al menos un Número
- ✓ Al menos un carácter especial como @, $, #, u otro

A continuación se muestra el código correspondiente a la función que implementa el mecanismo para el fortalecimiento de contraseñas:

```
function ValidarNuevoPassword(source, arguments)
{
  strmensajeerror="";
  error=0;
if(document.getElementById('<%=txtPassword.ClientID %>')==null)
{
   return;
}
var Pass = document.getElementById('<%=txtPassword.ClientID %>').value;
NCaracteres = Pass.length;
if(NCaracteres<8)
{
   strmensajeerror += '<%= GetGlobalResourceObject("mensajes","ClaveMasCaracteres") %>' + "<br>";
   error=1;
}
if(NCaracteres>12)
{
  strmensajeerror += 'La clave debe contener maximo 12 caracteres' + "<br>";
  error=1;
}
if(Pass.search(/[0-9]/)== -1)
{
  strmensajeerror += '<%= GetGlobalResourceObject("mensajes","ClaveUnNumero") %>' + "<br>";
  error=1;
}
if(Pass.search(/[A-Z]/)== - 1)
{
  strmensajeerror += '<%= GetGlobalResourceObject("mensajes","ClaveUnaMayuscula") %>' + "<br>";
  error=1;
}
if(Pass.search(/[a-z]/)== - 1)
{
  strmensajeerror += '<%= GetGlobalResourceObject("mensajes","ClaveUnaMinuscula") %>' + "<br>";
  error=1;
```

```
if(Pass.search(/(\s|\\|\/|!|"|ú|\$|%|&|\(|\)|=|\?|¨|\||@|#|ª|€|\^|`|\[|\]|\+|\*|ù|ï|\{|\}|\-|_|\.|:|,|;|>|<)/ )== - 1)
{
  strmensajeerror += '<%= GetGlobalResourceObject("mensajes","ClaveUnCaracterEspecial") %>' + "<br>";
  error=1;
}
document.getElementById('<%=cuvValidaClave.ClientID %>').errormessage = strmensajeerror;
if(error!=0)
{
arguments.IsValid = false;
}
}
```

Con esto se asegura que las contraseñas tengan un nivel de fortalezas apropiado para evitar el fácil acceso al Pentaho BI Server.

3.1.3 Implementación del subsistema de integración con el servidor de base de datos PostgreSQL.

El Pentaho BI Server utiliza como gestor de base de datos HSQL, con los inconvenientes mencionados en la sección 1.4 del capítulo 1. Por cuestiones de seguridad, se decide almacenar estas bases de datos en el gestor de base de datos PostgreSQL. Los pasos para esta migración quedan expuestos a continuación:

1- Crear el usuario pentaho_user, dba, hibuser con la contraseña que el usuario desee siempre y cuando emplee esa contraseña en los ficheros de conexión a la base de datos, para estos usuarios se utilizó la contraseña "*password*".

2- Crear las bases de datos hibernate y quartz, cargando los script de cada base de datos, y asignar privilegios de administración al usuario pentaho_user.

3- Modificar la conexión al gestor de base de datos, para eso modifique el fichero *context.xml*, ubicado en: */Pentaho/bi-server/tomcat/webapps/pentaho/META-INF/,* con los siguientes parámetros:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/pentaho" docbase="webapps/pentaho/">
    <Resource name="jdbc/Hibernate" auth="Container" type="javax.sql.DataSource"
        factory="org.apache.commons.dbcp.BasicDataSourceFactory" maxActive="20" maxIdle="5"
        maxWait="10000" username="hibuser" password="password"
        driverClassName="org.postgresql.Drive" url="jdbc:postgresql://localhost:5432/hibernate"
        validationQuery="select 1" />

    <Resource name="jdbc/Quartz" auth="Container" type="javax.sql.DataSource"
        factory="org.apache.commons.dbcp.BasicDataSourceFactory" maxActive="20" maxIdle="5"
        maxWait="10000" username="pentaho_user" password="password"
        driverClassName="org.postgresql.Drive" url="jdbc:postgresql://localhost:5432/quartz"
        validationQuery="select 1"/>
</Context>
```

4- Re direccionar hibernate a PostgreSQL, para esto es necesario editar el fichero *hibernate-settings.xml*, ubicado en */Pentaho/bi-server/pentaho-solutions/system/hibernate/* con la siguiente información:

```
-->
  <config-file>system/hibernate/postgresql.hibernate.cfg.xml</config-file>

<!--
```

5- Editar en la misma dirección el fichero *postgresql.hibernate.cfg*

```
<!--  Postgres 8 Configuration -->
<property name="connection.driver_class">org.postgresql.Driver</property>
<property name="connection.url">jdbc:postgresql://localhost:5432/hibernate</property>
<property name="dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="connection.username">hibuser</property>
<property name="connection.password">password</property>
<property name="connection.pool_size">10</property>
<property name="show_sql">false</property>
<property name="hibernate.jdbc.use_streams_for_binary">true</property>
```

6- Modificar la seguridad a la base datos de PostgreSQL para edite el *fichero applicationContext-spring-security-hibernate.properties*, ubicado en: */Pentaho/biserver-ce/Pentaho-solutions/system/*

```
jdbc.driver=org.postgresql.Driver
jdbc.url=jdbc:postgresql://localhost:5432/hibernate
jdbc.username=hibuser
jdbc.password=password
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

7- Modificar la conexión con las fuentes de datos, para esto edite el fichero *jdbc.properties* ubicado en: */Pentaho/biserver-ce/pentaho-solutions/system/simple-jndi/.*Cambie la definición y la dirección de los drivers por:

```
Hibernate/type=javax.sql.DataSource
Hibernate/driver=org.postgresql.Driver
Hibernate/url=jdbc:postgresql://localhost:5432/hibernate
Hibernate/user=hibuser
Hibernate/password=password
Quartz/type=javax.sql.DataSource
Quartz/driver=org.postgresql.Driver
Quartz/url=jdbc:postgresql://localhost:5432/quartz
Quartz/user=pentaho_user
Quartz/password=password
```

8- Insertar el driver de conexión de postgreSQL (“postgresql-9.1-703.jdbc4.jar”) en: */Pentaho/biserver-ce/tomcat/lib/*

9- Detener el arranque de la base de datos HSQLDB, editar el fichero web.xml, ubicado en: */Pentaho/biserver-ce/tomcat/webapps/pentaho/WEB-INF/*

```
<!-- [BEGIN HSQLDB STARTER] -->
<!-- <listener>
    <listener-class>org.pentaho.platform.web.http.context.HsqldbStartupListener</listener-class>
</listener> -->
<!-- [END HSQLDB STARTER] -->

<!-- JPivot resources initializer -->
```

10- Editar el fichero ubicado en la dirección: */Pentaho/biserver-ce/tomcat/conf/Catalina/localhost/pentaho*

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/pentaho" docbase="webapps/pentaho/">
    <Resource name="jdbc/Hibernate" auth="Container" type="javax.sql.DataSource"
        factory="org.apache.commons.dbcp.BasicDataSourceFactory" maxActive="20" maxIdle="5"
        maxWait="10000" username="hibuser" password="password"
        driverClassName="org.postgresql.Driver" url="jdbc:postgresql://localhost/hibernate"
        validationQuery="select 1" />

    <Resource name="jdbc/Quartz" auth="Container" type="javax.sql.DataSource"
        factory="org.apache.commons.dbcp.BasicDataSourceFactory" maxActive="20" maxIdle="5"
        maxWait="10000" username="pentaho_user" password="password"
        driverClassName="org.postgresql.Driver" url="jdbc:postgresql://localhost/quartz"
        validationQuery="select 1"/>
</Context>
```

Con esta migración, las base de datos de Pentaho BI server quedan resguardadas con mayor seguridad.

3.1.4 Integración del módulo de seguridad al Servidor de Inteligencia de Negocios

A continuación se muestran los pasos correspondientes para la integración del módulo de seguridad al Servidor de Inteligencia de Negocios de la Suite de Pentaho:

1. Colocar la aplicación en el servidor web ubicado en*: /Pentaho/bi-server/tomcat/webapps/hibernate/.*

Figura 10. Ubicación del Módulo de seguridad

2. Insertar el siguiente código en la página *launch.jsp* ubicado en: *Pentaho/biserver-ce/tomcat/webapps/pentaho/mantle/.*

```
<td align="center" class="smallButton"><a href="http://localhost:8080/Gestrol/index.jsp" target="_blank" ><button class=
"pentaho-button" id="button0" > Manage Rol</button></td>
```

3. Interfaz final con el acceso a la aplicación Gestionar Rol.

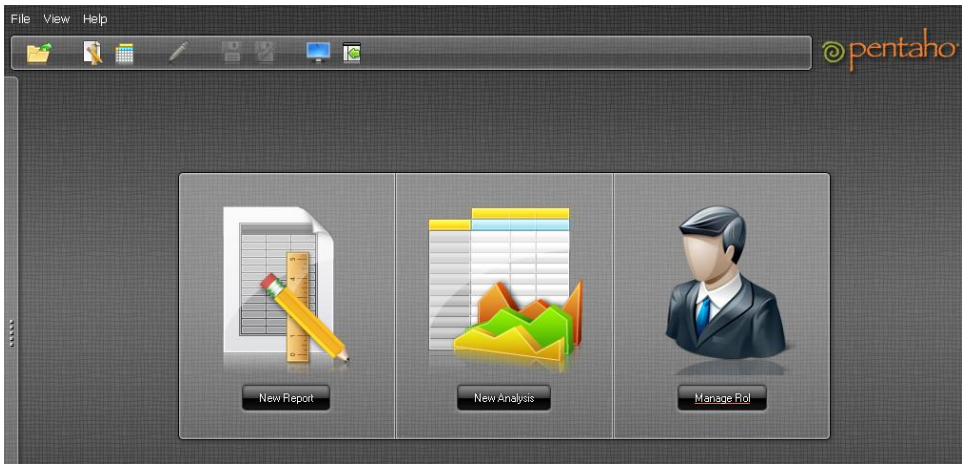


Figura 11. Interfaz principal

3.1.5 Interfaces de la aplicación correspondientes al subsistema “Gestionar rol”

El BI-Server no permite definir en detalle el nivel de acceso que tendrán los roles a la información. Esto se realiza a través de la herramienta Schema Workbench de la Suite de Pentaho, introduciendo los datos de los roles de forma manual. El usuario tiene que realizar consultas a la base de datos para verificarlos y escribirlos exactamente como aparecen en las tablas, posibilitando que se cometan errores al transcribirlos y convirtiéndose en una tarea costosa en tiempo y esfuerzo.

A continuación se muestran algunas de las interfaces correspondientes a la historia de usuario “Gestionar rol”, donde se le da la opción al usuario de que seleccione el XML con la estructura de los cubos multidimensionales de una solución determinada. El resto de las interfaces se encuentran en los anexos:

Figura 12. Cargar fichero XML

Figura 13. Datos de una Jerarquía

Figura 14. Conexión a la base de datos

Figura 15. Selección de datos

3.2 Pruebas a la aplicación

El proceso de prueba es uno de los pilares fundamentales de la metodología XP, este le proporciona la posibilidad al cliente de verificar y concretar las funcionalidades de las historias de usuarios, por lo que

favorece la comunicación entre el cliente y el equipo de desarrollo. Esta filosofía ayuda a identificar y corregir fallos u omisiones cometidas en las mismas, por lo que se reduce el número de errores no detectados así como el tiempo entre la introducción de éste en el sistema y su detección. Permite identificar historias de usuarios adicionales que no fueron obvias para el cliente o en las que este no hubiese pensado de no enfrentarse a dicha situación. Todo esto contribuye a elevar la calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir cambios o modificaciones.

La metodología XP como metodología ágil y enfocada principalmente en las necesidades del cliente, propone para el correcto funcionamiento del producto, la realización de dos grupos de pruebas, las unitarias y las de aceptación.

Las pruebas unitarias tienen como objetivo revisar el código de forma automática y son desarrolladas por los programadores. En cambio las pruebas de aceptación están destinadas a verificar que las historias de usuarios cumplen con las funcionalidades requeridas por el cliente al final de cada iteración. Las pruebas de aceptación tienen mayor peso que las unitarias ya que representan un indicador de la satisfacción del cliente con el producto. Este tipo de pruebas debe realizarse lo más rápido posible para que los programadores puedan hacer los cambios que se necesiten con mayor rapidez.

3.2.1 Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Las pruebas de aceptación son consideradas como “pruebas de caja negra” (*Black box system tests*). Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución.

Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación.

Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información.

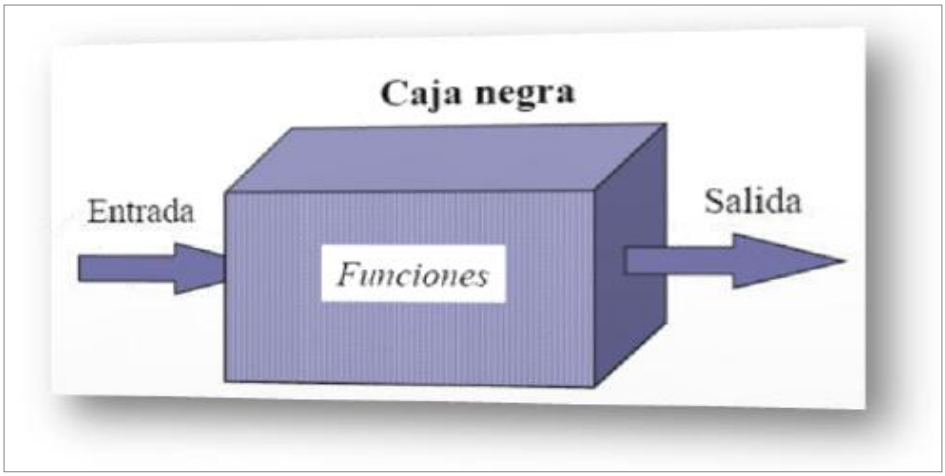


Figura 16. Prueba de caja negra

A continuación se muestran los casos de pruebas correspondientes a la historia de usuario "Autenticar usuario", el resto de los casos de pruebas se encuentran en los anexos:

Tabla 9. Caso de prueba "Autenticar usuario"

Escenario	Descripción	Variable1	Variable 2	Respuesta	Flujo central
EC 1.1 Autenticar usuario correctamente	Permitirá la autenticación de un usuario dado el usuario y la contraseña.	V (yacebal)	V (FormaCorrecta.*2012)	El sistema muestra la interfaz de bienvenida al Servidor de Inteligencia de Negocio de la Suite de Pentaho.	1-El usuario introduce el nombre de usuario. 2- El usuario introduce la contraseña. 3- El usuario presiona el botón "iniciar sesión"
EC 1.2 Autenticar usuario con campos incorrectos.	El sistema mostrará un error de autenticación.	I (yace2bal) V	V (FormaCorrecta.*2012) I	El sistema muestra un mensaje: Usuario o contraseña incorrecta.	

		(yacebal)	(FormaIncorrecta)		
EC 1.3 Autenticar usuario con campos vacíos.	El sistema no permitirá autenticar al usuario.	I ()	I ()	El sistema muestra un error indicando que el usuario o la contraseña son incorrectos.	

Dicho caso de prueba cuenta con tres escenarios: autenticar usuario correctamente, autenticar usuario con campos incorrectos y autenticar usuarios con campos vacíos. Estos escenarios son las operaciones que se realizan al autenticar un usuario, en las cuales pueden ocurrir errores a la hora de la autenticación. Para probar estos errores en la aplicación se definieron dos variables, donde la variable número uno es el nombre de usuario, el cual debe coincidir con el registrado en el servidor de dominio de la universidad y la variable número dos es la contraseña, la cual debe cumplir con las exigencias indicadas en el servidor de dominio.

En las siguientes imágenes se observan otras pruebas de funcionamiento realizadas al módulo de seguridad:

Figura 17. Interfaz de error cuando el usuario intenta cargar un archivo que no es de tipo XML

Figura 18. Interfaz de error cuando el XML tiene errores internos

Figura 19. Interfaz de error cuando el usuario va a terminar de modificar una Hierarchy y no ha definido el acceso

3.2.2 Prueba de integración del Pentaho BI Server con el CAS en un ambiente centralizado de autenticación de usuario

Se toma como mecanismo para autenticar los usuarios del sistema, integrar el Pentaho BI Server con el CAS, utilizando el LDAP de la Universidad de las Ciencias Informáticas. Para la realización de dicha integración se lleva a cabo una serie de pasos que son mostrados a continuación:

Lo primero es poner en el archivo *web.xml* ubicado en */biserver-ce/tomcat/webapps/pentaho/WEB-INF* la dirección donde se encuentra la carpeta *pentaho-solutions:*

```
<context-param>
    <param-name>solution-path</param-name>
    <param-value>biserver-ce/pentaho-solutions</param-value>
</context-param>
```

1- Agregarle a la app de Pentaho los jars *cas-client-core-3.1.10* y *spring-security-cas-client-2.0.4* para lograr la integración. Esto se realiza en la siguiente dirección:

biserver-ce/tomcat/webapps/pentaho/WEB-INF/lib

2- Actualizar el fichero *pentaho-spring-beans.xml* ubicado en *biserver-ce/pentaho-solutions/system* con la siguiente información:

```
<beans>
<import resource="pentahoSystemConfig.xml" />
<import resource="adminPlugins.xml" />
<import resource="systemListeners.xml" />
<import resource="sessionStartupActions.xml" />
<import resource="applicationContext-spring-security.xml" />
<import resource="applicationContext-common-authorization.xml" />
<import resource="pentahoObjects.spring.xml" />
<import resource="applicationContext-spring-security-jdbc.xml"/>
<import resource="applicationContext-pentaho-security-jdbc.xml"/>
<import resource="applicationContext-spring-security-cas.xml"/>
</beans>
```

3- Crear un nuevo archivo con el nombre *applicationContext-spring-security-cas.xml* y ubicarlo en la dirección *biserver-ce/pentaho-solutions/system* con la siguiente información:

```
        <property name="ticketValidator">
            <ref local="ticketValidator"/>
        </property>
        <property name="key" value="my_password_for_this_auth_provider_only"/>
    </bean>
    <bean autowire="default" class="org.jasig.cas.client.validation.Cas20ServiceTicketValidator"
dependency-check="default" id="ticketValidator" lazy-init="default">
        <constructor-arg index="0" value="http://localhost:8080/CAS"/>
    </bean>
<!-- overridden from applicationContext-spring-security.xml to specify logoutSuccessUrl as CAS
logout page -->
    <bean autowire="default" class="org.springframework.security.ui.logout.LogoutFilter" dependency-check="default" id="logoutFilter"
        <constructor-arg value="http://localhost:8080/CAS/logout?url=http://localhost:8080/pentaho/Home"/>
<!-- URL redirected to after logout -->
        <constructor-arg>
            <list>
                <bean autowire="default"
class="org.pentaho.platform.web.http.security.PentahoLogoutHandler" dependency-check="default" lazy-init="default"/>
                <bean autowire="default"
class="org.springframework.security.ui.logout.SecurityContextLogoutHandler" dependency-check="default"
lazy-init="default"/>
            </list>
        </constructor-arg>
        <property name="filterProcessesUrl" value="/Logout"/>
    </bean>
</beans>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--+
| Application context containing FilterChainProxy. This version overrides
| certain beans from applicationContext-spring-security.xml to enable CAS.
+-->
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans default-autowire="no" default-dependency-check="none" default-lazy-init="false">
<!-- ======================== FILTER CHAIN ======================== -->
<!-- overridden from applicationContext-spring-security.xml to enable CAS -->
    <bean autowire="default" class="org.springframework.security.util.FilterChainProxy" dependency-check="default"
    id="filterChainProxy" lazy-init="default">
        <property name="filterInvocationDefinitionSource">
            <value>
<![CDATA[CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
PATTERN_TYPE_APACHE_ANT
/**=securityContextHolderAwareRequestFilter,httpSessionContextIntegrationFilter,logoutFilter,casProcessingFilter,basicProcessingFilter
            </value>
        </property>
    </bean>
<!-- ==================== HTTP REQUEST SECURITY ==================== -->
    <bean autowire="default" class="org.springframework.security.ui.cas.ServiceProperties"
dependency-check="default" id="serviceProperties" lazy-init="default">
        <property name="service"
value="http://localhost:8080/pentaho/j_spring_cas_security_check"/>
        <property name="sendRenew" value="false"/>
    </bean>
<!-- replaces authenticationProcessingFilter in filterChainProxy above -->
    <bean autowire="default" class="org.springframework.security.ui.cas.CasProcessingFilter"
dependency-check="default" id="casProcessingFilter" lazy-init="default">
        <property name="authenticationManager">
            <ref bean="authenticationManager"/>
        </property>
        <!-- <property name="authenticationFailureUrl" value="/public/casFailed"/>-->
        <property name="authenticationFailureUrl" value="/Home"/>
        <property name="defaultTargetUrl" value="/"/>
        <property name="filterProcessesUrl" value="/j_spring_cas_security_check"/>
    </bean>
```

```
    <bean autowire="default" class="org.springframework.security.ui.ExceptionTranslationFilter"
dependency-check="default" id="exceptionTranslationFilter" lazy-init="default">
        <property name="authenticationEntryPoint">
            <ref local="casProcessingFilterEntryPoint"/>
        </property>
        <property name="accessDeniedHandler">
            <bean autowire="default"
class="org.springframework.security.ui.AccessDeniedHandlerImpl" dependency-check="default" lazy-init="default"/>
        </property>
    </bean>
    <bean autowire="default" class="org.springframework.security.ui.cas.CasProcessingFilterEntryPoint"
dependency-check="default" id="casProcessingFilterEntryPoint" lazy-init="default">
        <property name="loginUrl" value="http://localhost:8080/CAS/login"/>
        <property name="serviceProperties">
            <ref local="serviceProperties"/>
        </property>
    </bean>
<!-- overridden from applicationContext-spring-security.xml -->
    <bean autowire="default" class="org.springframework.security.providers.ProviderManager"
dependency-check="default" id="authenticationManager" lazy-init="default">
        <property name="providers">
            <list>
<!--ref bean="daoAuthenticationProvider" /-->
                <ref bean="anonymousAuthenticationProvider"/>
                <ref bean="casAuthenticationProvider"/>
            </list>
        </property>
    </bean>
    <bean autowire="default"
class="org.springframework.security.providers.cas.CasAuthenticationProvider" dependency-check="default"
id="casAuthenticationProvider" lazy-init="default">
        <property name="userDetailsService">
            <ref bean="userDetailsService"/>
        </property>
        <property name="serviceProperties">
            <ref local="serviceProperties"/>
        </property>
```

Con estos pasos queda realizada la integración del Pentaho BI Server con el CAS, y se delega el mecanismo para fortalecer las contraseñas de usuario hacia el implementado por el LDAP de la universidad, que además cumple con los mismos principios mencionados en la sección 3.1.2 del Capítulo 3, excepto que exige un mínimo de tres combinaciones de las propuestas en esa sección, lo cual evidencia que es un mecanismo seguro para la autenticación de usuarios.

Figura 20. Interfaz una vez integrado el módulo con el CAS

Luego de integrado al CAS, para adicionar un usuario al sistema se podrá realizar de la misma manera en la consola de administración, con la salvedad de que el usuario debe coincidir con el registrado en el Servidor de Nombres de Dominio (DNS) de la universidad.

Conclusiones

En este capítulo se describe la implementación de las historias de usuarios definidas en la fase de planificación, las cuales fueron divididas en siete tareas de ingeniería para facilitar la implementación de las mismas. Se definió el estándar de codificación que se utilizó para la implementación del sistema y se dio una breve descripción de las interfaces para dar una visión de cómo quedará la aplicación informática. Además se presentan las pruebas realizadas al módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite de Pentaho para garantizar su buen funcionamiento y calidad.

CONCLUSIONES

Como resultado de la investigación se desarrolló un módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite de Pentaho, dando con esto cumplimiento a los objetivos específicos y tareas trazadas. Se puede concluir con el desarrollo del mismo:

- ✓ El análisis de los principales conceptos relacionados con la seguridad del Servidor de Inteligencia de Negocios proporcionó los elementos teóricos necesarios para guiar el proceso de desarrollo.
- ✓ Los artefactos generados en el flujo de análisis y diseño permitieron implementar de forma satisfactoria el módulo de seguridad.
- ✓ La implementación de los diferentes subsistemas en los que se divide el módulo de seguridad, dio cumplimiento a las necesidades del cliente.
- ✓ La realización de pruebas al módulo de seguridad para comprobar su correcto funcionamiento, arrojaron un conjunto de no conformidades que fueron corregidas satisfactoriamente.

RECOMENDACIONES

Con la culminación del presente trabajo se obtuvo un módulo de seguridad para el Servidor de Inteligencia de Negocios de la Suite de Pentaho. Los desarrolladores de dicho módulo recomiendan:

- ✓ A Los especialistas del departamento utilizar el módulo desarrollado, para cumplimentar las pruebas exploratorias a la aplicación.
- ✓ Integrar el módulo a los proyectos del departamento, para elevar el nivel de seguridad de las soluciones que se desarrollen.

REFERENCIAS BIBLIOGRÁFICAS

1. VN., Vialart. Módulo de Gestión de Información en Salud . *Maestría de Psicología de la Salud.* [En línea] [Citado el: 06 de 01 de 2012.] http://www.aulauvs.sld.cu.

2. *Seguridad Informática.* Galdámez, Pablo. Madrid, España : s.n., 2003.

3. Raju, Prashant. *Pentaho Business Intelligence Server.* 2010.

4. Ordaz, Blvd. Diaz. Gravitar. [En línea] 2012. [Citado el: 25 de 02 de 2012.] http://www.gravitar.biz/index.php/herramientas-bi/pentaho/caracteristicas-pentaho/.

5. Wood, Sherman. Pentaho Powerfull Analytics Made Easy. [En línea] 2007-2009. [Citado el: 02 de 03 de 2012.] http://mondrian.pentaho.com/documentation/workbench.php.

6. Ing.Lilliam Vega Torres, Ing.Luis Rojas Díaz, Lic.Cecilia Placeres Villar. *LA INTELIGENCIA DE NEGOCIO. SU IMPLEMENTACIÓN MEDIANTE LA PLATAFORMA PENTAHO.* Ciudad de la Habana, Cuba : s.n., 2008.

7. Acosta, Rodrigo Martín, Remedi, Rolando Martin, Schumacher, Diego Leonel. BASES DE DATOS LIBRES ¿Qué alternativas existen? [En línea] [Citado el: 02 de 03 de 2012.] http://oooes.org/archivos/Bases_de_datos_libres.pdf.

8. [En línea] http://www.mitecnologico.com/Main/RequerimientosFuncionalesYNoFuncionales.

9. Marzo, Josep Vilalta. Introducción a UML. [En línea] 2008. [Citado el: 04 de 03 de 2012.] Marzo, Josep Vilalta. Introducción a UML. [Online] 2008..

10. Herramientas Case. [En línea] www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf. 875-99-OI-OTDETI-INEI.

11. Visual Paradigm. [En línea] [Citado el: 04 de 03 de 2012.]. http://www.visualparadigm.com

12. Bases de Datos. [Online] 2008. [Citado: Enero 20, 2012.].

13. Boyd, Omar y Kilani, Emily. PostgreSQL. [En línea] 1996-2008. [Citado el: 6 de diciembre de 2011.] http://archives.postgresql.org/pgsql-es-fomento.

14.The Appache Software Foundation Apache Tomcat. [En línea] 1999-2011. [Citado el: 7 de diciembre de 2011.] http://tomcat.apache.org.

15. SCRIBD. [En línea] [Citado el: 07 de 03 de 2012.]. http://es.scribd.com/doc/35236225/Eclipse-Java-en-Espanol.

16. EcuRed [En línea] [Citado el: 07 de 03 de 2012.]. *www.ecured.cu/index.php/Lenguaje_de_programación_Java.*

BIBLIOGRAFÍA

1. http://www.iti.es/media/about/docs/tic/01/2003-07-seguridad.pdf Autor: Pablo Galdámez
2. 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/.
3. http://www.pentaho.com/products/data_integration/
4. Lenguajes de Programación Lenguajes de Programación http://www.lenguajes-de-programacion.com.
5. Introducción a Pentaho. Gravitar. [Online] 2009. http://www.gravitar.biz/index.php/bi/introduccion-pentaho-parte-1/
6. VN., Vialart. Módulo de Gestión de Información en Salud . Maestría de Psicología de la Salud. [En línea] [Citado el: 06 de 01 de 2012.] http://www.aulauvs.sld.cu.
7. Seguridad Informática. Galdámez, Pablo. Madrid, España : s.n., 2003.
8. Raju, Prashant. Pentaho Business Intelligence Server. 2010.
9. Ordaz, Blvd. Diaz. Gravitar. [En línea] 2012. [Citado el: 25 de 02 de 2012.] http://www.gravitar.biz/index.php/herramientas-bi/pentaho/caracteristicas-pentaho/.
10. Wood, Sherman. Pentaho Powerfull Analytics Made Easy. [En línea] 2007-2009. [Citado el: 02 de 03 de 2012.] http://mondrian.pentaho.com/documentation/workbench.php.
11. Ing.Lilliam Vega Torres, Ing.Luis Rojas Díaz, Lic.Cecilia Placeres Villar. LA INTELIGENCIA DE NEGOCIO. SU IMPLEMENTACIÓN MEDIANTE LA PLATAFORMA PENTAHO. Ciudad de la Habana, Cuba : s.n., 2008.
12. Pentaho. Pentaho Data Integration. [Online] 2005-2010.
13. Pentaho. Pentaho. [Online] 2009. [Cited: enero 21, 2010.] http://mondrian.pentaho.org/. ISBN.
14. Matheus, Christopher J. 1992, AI Magazine, p. 70.
15. Marzo, Josep Vilalta. Introducción a UML. [Online] 2008. http:www.vico.org/aRecursosPrivats/TRAD_introUML.pdf.
16. Acosta, Rodrigo Martín, Remedi, Rolando Martin, Schumacher, Diego Leonel. BASES DE DATOS LIBRES ¿Qué alternativas existen? [En línea] [Citado el: 02 de 03 de 2012.] http://oooes.org/archivos/Bases_de_datos_libres.pdf.
17. [En línea] http://www.mitecnologico.com/Main/RequerimientosFuncionalesYNoFuncionales.
18. Marzo, Josep Vilalta. Introducción a UML. [En línea] 2008. [Citado el: 04 de 03 de 2012.] Marzo, Josep Vilalta. Introducción a UML. [Online] 2008.

19. Herramientas Case. [En línea] www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf. 875-99-OI-OTDETI-INEI.
20. Visual Paradigm. [En línea] [Citado el: 04 de 03 de 2012.]. http://www.visualparadigm.com
21. Bases de Datos. [Online] 2008. [Citado: Enero 20, 2012.].
22. Boyd, Omar y Kilani, Emily. PostgreSQL. [En línea] 1996-2008. [Citado el: 6 de diciembre de 2011.] http://archives.postgresql.org/pgsql-es-fomento.
23. The Appache Software Foundation Apache Tomcat. [En línea] 1999-2011. [Citado el: 7 de diciembre de 2011.] http://tomcat.apache.org.
24. SCRIBD. [En línea] [Citado el: 07 de 03 de 2012.]. http://es.scribd.com/doc/35236225/Eclipse-Java-en-Espanol.
25. EcuRed [En línea] [Citado el: 07 de 03 de 2012.]. www.ecured.cu/index.php/Lenguaje_de_programación_Java.

ANEXOS

A continuación se muestran ejemplos de historias de usuarios, tareas de ingeniería, tarjetas CRC e interfaces de la aplicación. El resto se pueden apreciar en las planillas correspondientes a cada una de ellas.

Anexo 1. Historia de usuario “Migrar la información del sistema, los usuarios, conexiones y datos de configuración al sistema gestor de base de datos PostgreSQL”. El resto se pueden observar en la

Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Migrar la información del sistema, los usuarios, conexiones y datos de configuración al sistema gestor de base de datos PostgreSQL.
Cantidad de modificaciones a la Historia de Usuario: 2	
Usuario: Yuniel Acebal Urtiaga Ramsés López Sosa	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Alta	**Puntos reales:** 2 semanas
Descripción: Se realiza la migración de las bases de datos “hibernate” y “quartz” en las cuales se encuentra la información del sistema, los usuarios, conexiones y datos de configuración, al gestor de base de datos PostgreSQL.	

Observaciones:

Anexo 2. Tarea de ingeniería “Diseño de la interfaz para la autenticación de los usuarios” de la HU número 1.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Diseño de la interfaz para la autenticación de los usuarios.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 6/2/12	**Fecha Fin:** 10/2/12
Programador Responsable: Yuniel Acebal Urtiaga Ramsés López Sosa	
Descripción: Se implementa y diseña la interfaz para la autenticación de los usuarios.	

Anexo 3. Tarjeta CRC de la clase Schema.

Tarjeta CRC	
Clases: Schema	
Responsabilidades	**Colaboraciones**

- Llenar las listas de datos - Construir el objeto Schema	- Rol - Cube - Dimension

Anexo 4. Interfaz “Modificar Esquema”.

GLOSARIO DE TÉRMINOS

Almacenes de datos: Colección de datos que recoge información de múltiples sistemas fuentes u operacionales dispersos.

Applets: Programa Java diseñado para ejecutarse en una página web a través de un navegador que soporte código Java.

Autenticación: Verificación de la identidad de una persona, usuario o proceso, para así acceder a determinados recursos.

Bytecode: Archivo binario que contiene un programa ejecutable.

Cortafuegos: Sistema que previene el uso y el acceso desautorizados a tu ordenador.

Encapsulación: Habilidad de una parte de un programa para ocultar sus datos al resto del código, impidiendo así accesos incorrectos o conflictos con los nombres de otras variables.

FTP: File Transfer Protocolo (Protocolo de Transferencia de Ficheros).

HTTP: Protocolo de transferencia de hipertexto.

HTTPS: Protocolo seguro de transferencia de hipertexto.

Módulo: Porción de un programa de computadora hace una de las tareas que dicho programa debe realizar.

Oracle: Es un sistema de gestión de base de datos.

Plugin: Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño.

Proxy: Es un servidor que actúa de intermediario entre una conexión a internet y una red interna.

LDAP: Lightweight Directory Access Protocol (Protocolo Ligero de Acceso a Directorios).

Single SignOn: Es la autentificación única por parte del usuario para acceder a sus recursos.

SQL Lite: Es un sistema de gestión de bases de datos relacional.

TCP/IP: Conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

URL: Uniform Resource Location (Localizador de Recursos Uniformes).