Universidad de las Ciencias Informáticas

Facultad # 1

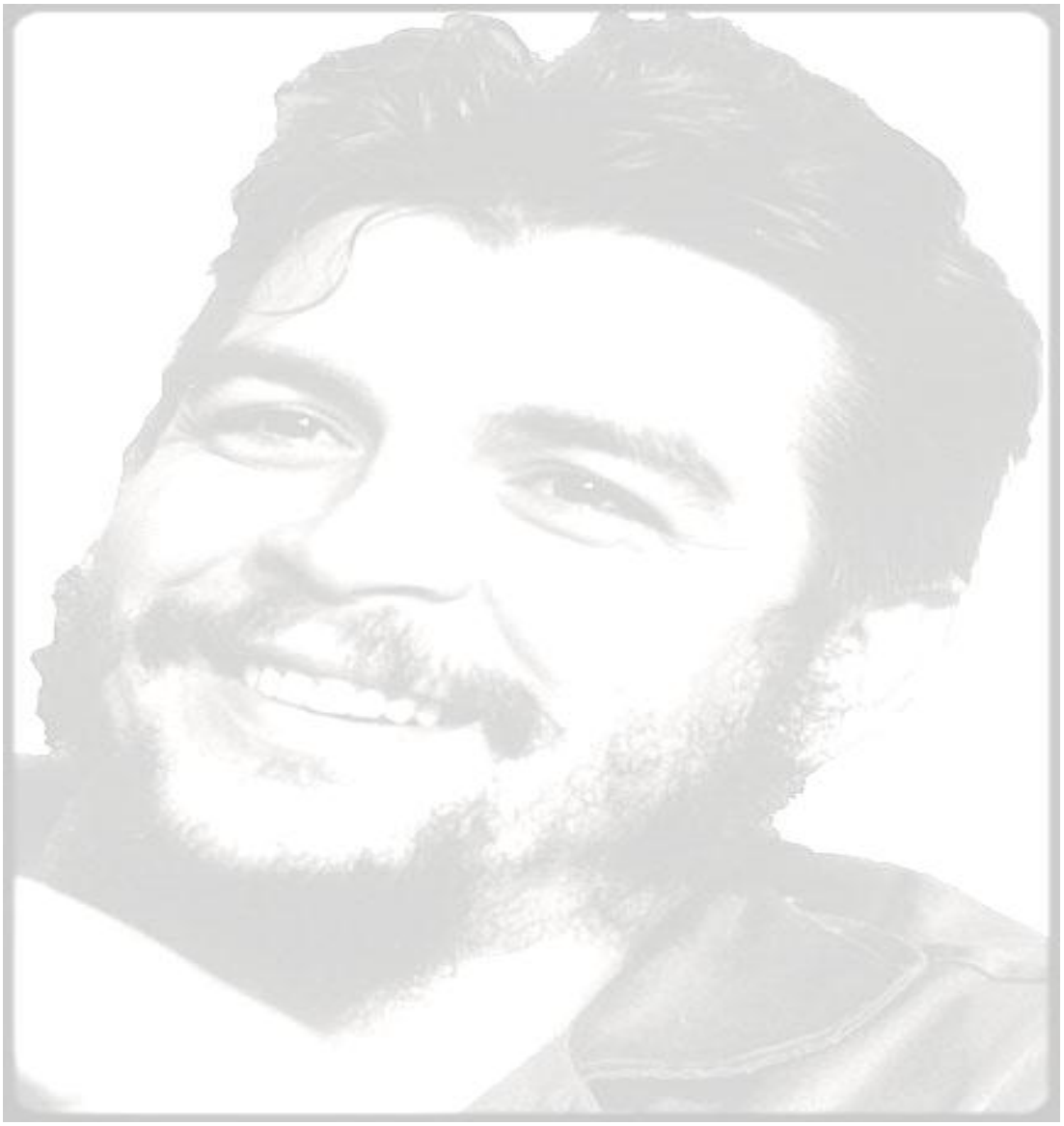
Módulo de Conectores para el Subsistema de Aprovisionamiento de Usuarios del Sistema Administración de Identidades.

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

Autores: Wilhem Manuel Verano Escalona
Ernesto Raúl Martín Suárez

Tutor: Ing. Roberto Quiñones Bondartchuk

Junio, 2011



"La arcilla fundamental de nuestra obra es la juventud; en ella depositamos nuestra esperanza y la reparamos para tomar de nuestras manos la bandera."

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la facultad 1 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Wilhem Manuel Verano Escalona

_____

(Autor)

Ernesto Raúl Martín Suárez

_____

(Autor)

Roberto Quiñones Bondartchuk

_____

(Tutor)

AGRADECIMIENTOS

A nuestro tutor Robertón por ayudarnos siempre en todo, a Maikel por siempre dar sus ideas y ayudar en lo que pudiera, en fin a todos los profesores del proyecto y a aquellas personas que de una forma u otra ayudaron en la realización del trabajo.

<<Wilhem>>

A mi mamá, por siempre hacer hasta lo imposible por hacer realidad mis sueños y demostrarme que siempre se puede, por quererme tanto y haber sido la mejor madre del mundo, no tengo palabras para agradecerte todo lo que has hecho por mí.

A mi hermanita chiquita, que aunque crezcas, para mí siempre vas a ser una niña, por ser tan cariñosa y quererme tanto.

A mi princesita linda que siempre estuvo a mi lado en los buenos y malos momentos dándome todo su apoyo y amor.

A mis suegros por acogerme como un hijo más.

A mi abuelita postiza Evita, jaja, por ser tan buena siempre conmigo.

A mi compañero de tesis Ernestón por siempre brindarme su apoyo.

A todos mis amigos que tanto me ayudaron en mi primer año.

<<Ernesto>>

Son muchas las personas que han hecho de este momento una realidad y que me gustaría agradecerles por su apoyo y confianza.

Quisiera agradecer de una forma muy especial a mi mamá, mi papá y a mi abuelita por ser una guía en todo momento de mi vida, por brindármelo todo, para que yo alcanzara mi sueño.

A mi hermano, mi cuñada y su familia por acogerme cuando lo necesité y por siempre estar buscándome soluciones a mis problemas.

A mis tíos, especialmente a Rolado y Marta por ser como unos padres para mí.

A mi novia Milena con la que he pasado momentos lindos que nunca podré olvidar.

A Reinier que es como mi hermano por todos los trabajos que pasamos y a los cuales siempre les encontrábamos una solución.

A Maidelys (La flaca), mimi y mima por acogerme en su casa como un integrante más de la familia.

A todas las personas que conocí en esta universidad y que convivimos como una familia en estos 5 años.

A todas aquellas amistades que conocí este año, de la antigua facultad 10 con las que compartí momentos especiales.

A mi compañero de tesis Wilhem por todo el trabajo que realizó y siempre estar aconsejándome para que saliera un trabajo con calidad.

DEDICATORIA

<<Wilhem>>

A mi madre, por ayudarme como siempre lo ha hecho y darme todo el amor del mundo, sin ti este sueño no hubiera sido posible. A mi hermanita por ser tan cariñosa y buena. A mi querida novia que tanto me ha apoyado en estos cinco años y me ha brindado todo su amor. A mis buenos suegros y a mi linda abuelita postiza.

<<Ernesto>>

Este trabajo de diploma va dedicado a lo más lindo que he tenido en mi vida, mi mamá y mi papá, porque gracias a su apoyo, dedicación y confianza he cumplido todas mis metas como estudiante y podré ser la persona que tanto ellos desearon que fuera.

RESUMEN

El presente trabajo tiene como objetivo el diseño, implementación y pruebas del Módulo de Conectores del Subsistema de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades, con la intención de dar solución a los problemas existentes que giran en torno al proceso de aprovisionamiento dentro del Sistema.

Para cumplir los objetivos del trabajo se realizó un estudio de diferentes sistemas de aprovisionamiento haciendo énfasis en sus arquitecturas de conectores, con vista a encontrar ventajas y desventajas para luego llegar a una propuesta. Además se analizaron patrones sobre la plataforma .NET que permitieran la extensibilidad del Módulo y que permitiera transacciones. Fueron generados los artefactos definidos por la metodología Microsoft Solution Framework (MSF) para así guiar de una manera correcta el proceso de desarrollo.

Finalmente fue implementado el Módulo de Conectores sobre la plataforma .NET, el mismo permite la conexión a diferentes recursos específicos para realizar tareas de aprovisionamiento de cuentas de usuario, además de brindar a través de una API la posibilidad de continuar su implementación para incluir nuevos conectores para nuevos recursos. Además fue desarrollado un modelo que transacciones que permite ejecutar cada funcionalidad de manera transaccional. Posteriormente fueron realizadas las pruebas pertinentes para verificar su correcto funcionamiento, las cuales arrojaron resultados satisfactorios.

Palabras clave: aprovisionamiento de usuarios, administración de identidades, conectores.

ÍNDICE

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

El avance de la Informática y las Telecomunicaciones ha tenido un gran impacto en el desarrollo de las empresas lo que ha cambiado totalmente el paradigma de la forma en que se almacena la información, pasando del contexto físico al digital.

Las grandes organizaciones de hoy en día han hecho uso de las nuevas tecnologías para automatizar sus procesos, los que suelen ser complejos y con diseños poco satisfactorios para aprovisionar los sistemas con información para los usuarios. Estos últimos en ocasiones tienen que esperar un tiempo prolongado para tener acceso a aquellas aplicaciones que necesitan para realizar su trabajo. Además el conjunto de actividades que realizan de forma manual afines con el aprovisionamiento de identidades agregan gastos generales y retrasan la productividad de los trabajadores.

Uno de los sistemas que se desarrollan en el Centro de Identificación y Seguridad Digital (CISED), perteneciente a la Universidad de las Ciencias Informáticas (UCI) es el Sistema Administración de Identidades. Aún se encuentran en desarrollo los Subsistemas, Autenticación (centraliza el proceso de autenticación y está basado en el estándar SAML), Autorización (permite definir políticas de acceso a los recursos, está basado en el modelo RBAC) y Aprovisionamiento de Usuarios. Este último constituye una necesidad fundamental puesto que debe permitir entre otras funciones las conexiones a los recursos tales como: Directorios de Identidades, Servidores de Mensajería Instantánea, Servidores de Correo, Servidores de Base de Datos y realizar la creación, modificación y eliminación entre otras funciones de manera centralizada de cuentas de usuarios en los mismos. El Subsistema de Aprovisionamiento aún carece de estas funciones.

Para establecer la comunicación con dichos recursos los sistemas de aprovisionamiento de usuarios se valen de *Conectores* predefinidos en los cuales recaen las tareas de aprovisionamiento para cada tipo de recurso y fabricante, siendo estos los encargados de traducir los comandos del sistema de provisión al lenguaje que pueden entender los recursos o sistemas administrados. Algunos de estos conectores se crean con la instalación del sistema de aprovisionamiento y otros son agregados de acuerdo a las necesidades de la organización que haga uso de ellos.

El módulo Conectores proporcionará un funcionamiento eficaz del Subsistema Aprovisionamiento de Usuarios del Sistema Administración de Identidades. Con la implementación del mismo se proveerán múltiples beneficios permitiendo a este Subsistema la gestión y centralización de cuentas de usuarios solicitadas de una manera rápida, segura y sencilla. Este trabajo de diploma centra su análisis precisamente en las funciones de estos conectores.

Problema a resolver:

¿Cómo gestionar las conexiones a los recursos que utiliza el Subsistema de Aprovisionamiento de Usuarios del Sistema Administración de Identidades para poder realizar las tareas de aprovisionamiento?

Objeto de estudio:

Proceso de aprovisionamiento de usuarios.

Campo de acción:

Conectores a recursos de sistemas de aprovisionamiento.

Objetivo general:

Desarrollar el Módulo de Conectores para el Subsistema de Aprovisionamiento de Usuarios del Sistema Administración de Identidades.

Objetivos específicos:

- Diseñar el Módulo Conectores para el Subsistema de Aprovisionamiento de Usuarios.
- Implementar el Módulo Conectores para el Subsistema de Aprovisionamiento de Usuarios.
- Probar el funcionamiento del módulo implementado.

Idea a defender:

La implementación del Módulo Conectores del Subsistema de Aprovisionamiento del Sistema de Administración de Identidades permitirá conectar este Subsistema a todos los recursos disponibles para aprovisionar las cuentas de usuario.

Tareas a Cumplir:

- Elaborar el marco teórico para la justificación del trabajo.
- Analizar el funcionamiento de los Conectores en un Sistema de Aprovisionamiento.
- Estudiar las herramientas y tecnologías para diseñar, implementar y probar el módulo.
- Analizar los distintos sistemas de aprovisionamiento existentes.
- Realizar un estudio de conectores existentes en los distintos sistemas de aprovisionamiento.
- Diseñar el Módulo Conectores.
- Instalar y configurar los servidores necesarios para realizar el trabajo (Servidor de Correo Zimbra, Servidor de Mensajería Instantánea Openfire, Servidor de Base Datos Oracle 11g y Servicio de Directorio OpenLdap).
- Implementar el Módulo Conectores en la plataforma establecida.
- Aplicar Pruebas Unitarias.

Métodos Científicos

Para desarrollar esta investigación y lograr los objetivos planteados se utilizan métodos teóricos y empíricos, mediante los cuales se obtiene una idea más detallada de lo que se quiere lograr. En la investigación se utilizan los siguientes:

Teóricos:

Histórico - Lógico: Se hizo un estudio de los procesos de conexión de sistemas de administración de identidades a diferentes recursos, adquiriendo una visión de las fortalezas y debilidades que presentaban estos sistemas, concluyendo así, la necesidad de implementar un módulo que gestionara las conexiones a dichos recursos.

Analítico - Sintético: Se analizó el modelo Provider y el patrón Command los cuales serán utilizados por sus características en el desarrollo del Módulo Conectores.

Empíricos:

Observación: Se estudió la manera en que los sistemas de administración de identidades se conectaban a recursos, logrando así identificar los elementos que necesitaría el sistema a desarrollar.

Teniendo en cuenta el valor metodológico y práctico de este trabajo se obtienen como aportes: un módulo de conexión que facilitará y gestionará las conexiones a diferentes recursos para realizar las tareas de aprovisionamiento, una API que permitirá la implementación de nuevos conectores dada la inclusión de algún recurso en el sistema y un modelo de transacciones que permitirá que el Módulo de Conectores soporte las transacciones.

Capítulo 1

Fundamentación Teórica

1.1 Introducción

El desarrollo de las tecnologías y las comunicaciones ha propiciado la creación de herramientas de Gestión de Identidades Digitales, las cuales ahorran tiempo y dinero a aquellas organizaciones que las usan. Estos programas incluyen procesos como la Autenticación, Autorización y Aprovisionamiento de Usuarios. Existen varias soluciones de Administración de Identidades con gran prestigio a nivel mundial, estas tienen precios muy altos lejos del alcance de nuestro país. Surge la necesidad de la creación de un sistema de Administración de Identidades que cuente con un componente de Aprovisionamiento de Usuarios, el cual incluya una variedad de *Conectores* predefinidos además de brindar la posibilidad de incluir nuevos. Para dar solución a lo antes planteado se hace necesario estudiar aquellas soluciones existentes así como las tecnologías y herramientas a utilizar.

1.2 Aprovisionamiento de usuarios.

Dentro de los procesos existentes en la Administración de Identidades se encuentra la Autenticación, Autorización, Auditoría y el Aprovisionamiento de Usuarios, este último se encarga de controlar el ciclo de vida de las identidades digitales. Este se refiere a la creación, mantenimiento y a la desactivación de las cuentas de usuario en uno o más sistemas de manera dispersa o centralizada, ya que los usuarios pueden ser representados por objetos múltiples en múltiples sistemas.

Figura 1 Procesos existentes en la Administración de Identidades.

Un sistema de aprovisionamiento es una solución o grupo de soluciones de software para realizar funciones sobre las cuentas de usuario de manera automática y centralizada.

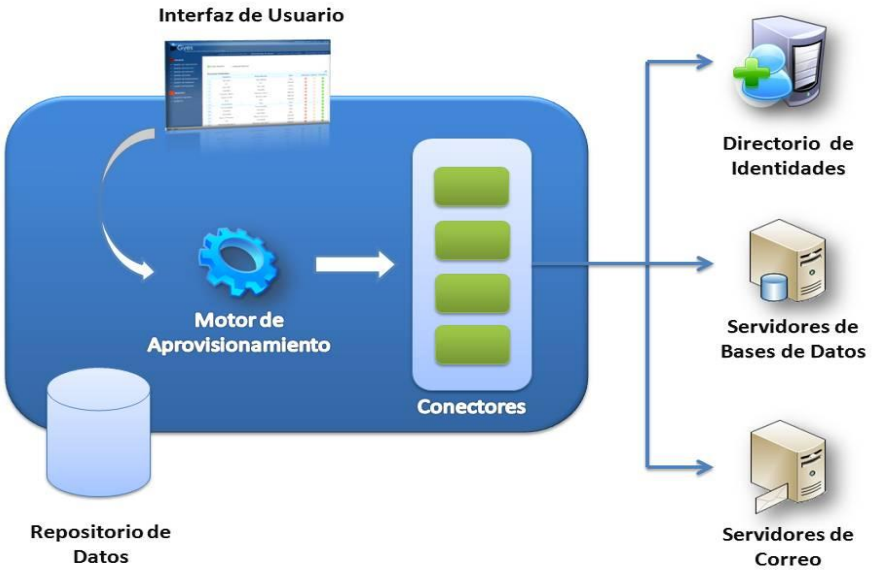


Figura 2 Sistema de Aprovisionamiento de Usuarios.

Los Sistemas de Aprovisionamiento incluyen de manera general el siguiente grupo de componentes:

- **Interfaz de usuario:** donde los usuarios pueden revisar los contenidos de la base de datos interna, realizar solicitudes de cambio, aprobar o rechazar los cambios propuestos, etc.
- **Motor de aprovisionamiento:** interpreta las solicitudes enviadas y las distribuye entre los recursos apropiados (Bhuvan Urgaonkar, 2008).
- **Repositorio de Datos**: encargado de rastrear los objetos de usuarios y otros datos de sistemas integrados y aplicaciones. Estos repositorios generalmente son bases de datos y servicios de directorios basados en protocolo LDAP (IETF, 2006) (ITU, 2008).
- **Conectores**: encargados de leer información acerca de los usuarios de sistemas integrados y aplicaciones, además de realizar actualizaciones (crear, modificar y eliminar información de un usuario).

1.3 Conectores de un sistema de aprovisionamiento de usuarios.

Los conectores de un Sistema de Administración de Identidades son los encargados de garantizar la interacción con los recursos que va a utilizar el sistema. Los grandes sistemas líderes del mercado en aprovisionamiento de usuario suministran algunos conectores predeterminados que responden a recursos específicos más frecuentes los cuales dependen del fabricante y modelo, del mismo modo dan la posibilidad de crear otros nuevos conectores para comunicarse prácticamente con cualquier recurso mediante interfaces de programación de aplicaciones (API).

Un conector está compuesto por un conjunto de clases, en las cuales quedan implementadas las funcionalidades para realizar las tareas de aprovisionamiento. Los conectores para poder realizar sus funciones tienen que conocer algunas credenciales del recurso al cual se van a conectar para de esta forma ejecutar sus tareas. Los parámetros de conexión varían en dependencia del recurso y una vez introducidos por el administrador del sistema son guardados cifrados en una base de datos encargada de contener toda esta información.

En el presente trabajo se da la solución de la implementación de cuatro tipos de conectores específicos, los cuales son: Conector para BD Oracle 11g, Conector para Servicio de Directorio OpenLdap (IETF, 2006) (ITU, 2008), Conector para Servidor de Mensajería Instantánea Openfire y un Conector para Servidor de Correo Zimbra, además de que se brindará una API que permitirá la implementación de nuevos conectores.

1.3.1 Conectores para base de datos.

Los conectores de base de datos dependen de las credenciales del servidor y algunos parámetros necesarios para realizar una conexión. Estos parámetros son variables dependiendo del gestor de base de datos, los más comunes son: Ip del servidor, puerto, usuario y contraseña. Luego de establecida una conexión él conector está disponible para realizar las funciones sobre las cuentas de usuario. Estos procedimientos son llevados a cabo a través de instrucciones enviadas hacia el gestor de base de datos siendo este último el encargado de ejecutarlas.

1.3.2 Conectores para servicio de directorio.

Las credenciales necesarias por este conector para realizar sus funciones generalmente son: el Ip del servidor, el puerto, que generalmente es el 389, además del usuario y la contraseña con privilegios administrativos. La conexión a un servicio de directorio generalmente se puede hacer mediante el protocolo LDAP, el cual además brinda la posibilidad de realizarla a través de un canal seguro (SSL).

La conocida empresa de software Novell y líder en esta rama de la administración de identidades ha desarrollado librerías para el framework .NET para trabajar con el servicio de directorio OpenLdap a través del protocolo LDAP las cuales brindan las funcionalidades necesarias para desarrollar este conector.

1.3.3 Conectores para servidores de correo.

Los conectores para servidores de correo son los encargados de garantizar que la conexión con estos se realice de forma satisfactoria y que se puedan atender todas las solicitudes de creación, modificación y eliminación de cuentas en dichos servidores y además ser ejecutadas.

Los servidores de correo utilizan varios protocolos de conexión para la trasferencia y obtención de correos electrónicos, entre los protocolos más usados se encuentran POP (con su versión más actual POP3 (G. Myers, et al., 1996)), IMAP (Crispin, 1996) y SMTP (Postel, 1982).

El aprovisionamiento de usuarios en un servidor de correo no depende del protocolo que usen para su conexión, sino de las distribuciones que usan dichos protocolos. Entre estas distribuciones encontramos Sendmail, Microsoft Exchange Server, Postfix, Zimbra Server, entre otros, algunas con licencias de código abierto y otros con licencias propietarias.

1.3.4 Conectores para servidores de mensajería instantánea.

La mensajería instantánea en su concepto es una forma de comunicación en tiempo real entre dos o más personas basada en texto. El texto es enviado a través de dispositivos conectados a una red como Internet. Adicionalmente, hay programas de mensajería instantánea que utilizan el protocolo abierto XMPP (Saint-Andre, 2011), con un conjunto descentralizado de servidores (Rodríguez, 2010).

En la actualidad existe una gran variedad de implementaciones de servidores de mensajería instantánea, algunos son libres (Openfire, Ejabberd, Tigase, entre otros) y otros propietarios (Jabber platform, TIMP, Rhombus, entre otros).

Los conectores para servidores de mensajería instantánea serán los encargados asegurar la conexión a cualquiera de estos servidores garantizando al sistema de aprovisionamiento las funciones de creación, modificación, eliminación, entre otras, de cuentas de usuarios. Estos servidores utilizan un gestor de bases de datos (MySQL, PostgresSQL, Oracle) o un directorio de identidades (OpenLdap, Microsoft Active Directory) para la autenticación y el almacenamiento de información de usuario.

1.4 Sistemas de aprovisionamiento líderes del mercado.

Los sistemas de aprovisionamiento de usuarios surgieron con el fin de satisfacer varias necesidades de las organizaciones, tales como: el incremento de la seguridad y el ahorro de recursos. Estos sistemas posibilitan realizar la creación, modificación y eliminación de cuentas de usuario en recursos de manera rápida y centralizada. Además permiten llevar un control detallado del ciclo de vida de cada cuenta de usuario dentro de una organización.

Existen varias empresas con productos de administración de identidades de gran calidad en el mercado, entre estas se pueden mencionar Oracle, IBM, Novell y CA.

A continuación se mencionan productos de estas compañías y son presentados algunas de sus características y conectores.

1.4.1 Oracle Identity and Access Management Suite.

La Suite de Administración de Acceso e Identidad de Oracle permite a las empresas administrar todo el ciclo de vida de los usuarios en todos los recursos empresariales tanto dentro como fuera del firewall. Permite a los clientes cumplir los requisitos normativos con eficacia, proteger sus aplicaciones más importantes y datos confidenciales además de reducir los costes de explotación. Ofrece funcionalidades para la administración de roles e identidades, la administración de aprobaciones y pedidos, la administración de derechos basados en políticas, la integración de tecnología y la automatización para el cumplimiento y las auditorias.

La adquisición de Sun por parte de Oracle refuerza la oferta de gestión de identidades. La estrategia de Oracle es ofrecer un conjunto de productos de gestión de identidades integrados que combinen los mejores productos y prestaciones (Oracle, 2009).

A continuación se muestra la arquitectura de los conectores de Oracle Identity Management (OIM) para realizar una conexión con una base datos del mismo tipo:

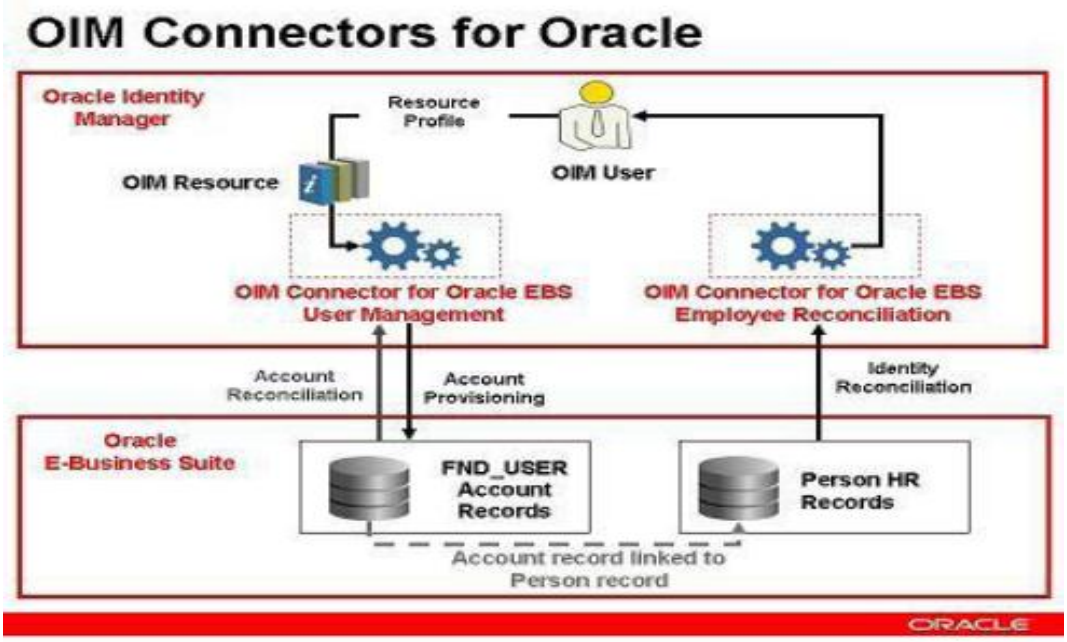


Figura 3 Arquitectura de conectores de OIM para una base datos Oracle.

Algunas de las características clave de la conexión son:

- Capacidad para conciliar atributos adicionales personalizados con un enfoque de consulta para que los clientes puedan dirigir el aprovisionamiento, certificación, información y capacidades de administración delegada con estos atributos.
- Detección de todos los eventos importantes del ciclo de vida de una cuenta de usuario, los cambios de trabajo, traslados, despidos, etc.

Para la realización del aprovisionamiento en la base datos Oracle se realizan las siguientes actividades:

- **Creación de cuenta:** basado en la información proporcionada el conector deberá crear un usuario con todos sus atributos.
- **Actualización de cuenta:** permite la modificación de los atributos relacionados con un usuario específico.
- **Agregar/Eliminar asignación de responsabilidades:** permite la eliminación o adición de nuevas responsabilidades a cumplir por el usuario.
- **Deshabilitar cuenta:** permite deshabilitar una cuenta dada una fecha tope para su funcionamiento.
- **Habilitar cuenta:** al quitar la fecha tope de funcionamiento la cuenta queda habilitada nuevamente.

Por otro lado la conciliación de cuenta es la función que se realiza para la actualización y verificación de todos los cambios realizados en la base de datos.

1.4.2 IBM TIM v.7.1.

El Sistema de Administración de Aprovisionamiento Tivoli de la empresa IBM (IBM Tivoli Provisioning Manager) se encuentra dentro de la suite de Administración de Identidades de Tivoli (Tivoli Identity Management TIM), este software incluye una diversidad de componentes que son los encargados de realizar las tareas de aprovisionamiento. Está dividido fundamentalmente en un servidor de suministro, una consola de operador y administrador, basada en web y en un entorno de desarrollo de paquetes de automatización (IBM, 2009) (IBM, 2009).

A continuación se mencionan los componentes principales de Tivoli Provisioning Manager (IBM, 2009).

- **Servidor de suministro**: En el servidor de suministro está instalado el Administrador de Aprovisionamiento de Tivoli. Este servidor contiene algunos componentes como son: una base de datos de suministro, modelo de datos, conformidad y remediación, informes, descubrimiento y automatización; los cuales realizan funcionalidades especificas dentro del servidor.
- **Infraestructura de despliegue**: Permite reconfigurar y reasignar los recursos del entorno gestionado utilizando dos infraestructuras de despliegue distintas: de distribución de software escalable y de motor de despliegue.
- **Consola de operador y administrador**: La consola de operador y administrador basada en web le permite interactuar con el servidor de suministro de Tivoli. Proporciona una representación gráfica del centro de datos, incluye asistentes para simplificar la configuración y ofrece otras funciones.
- **Entorno de desarrollo de paquetes de automatización:** APDE (Automation Package Developer Environment) es un entorno de plug-in basado en Eclipse que los desarrolladores de paquetes de automatización pueden utilizar para personalizar los paquetes existentes o crear nuevas automatizaciones (IBM, 2009).

A continuación se muestra la arquitectura de IBM Tivoli.

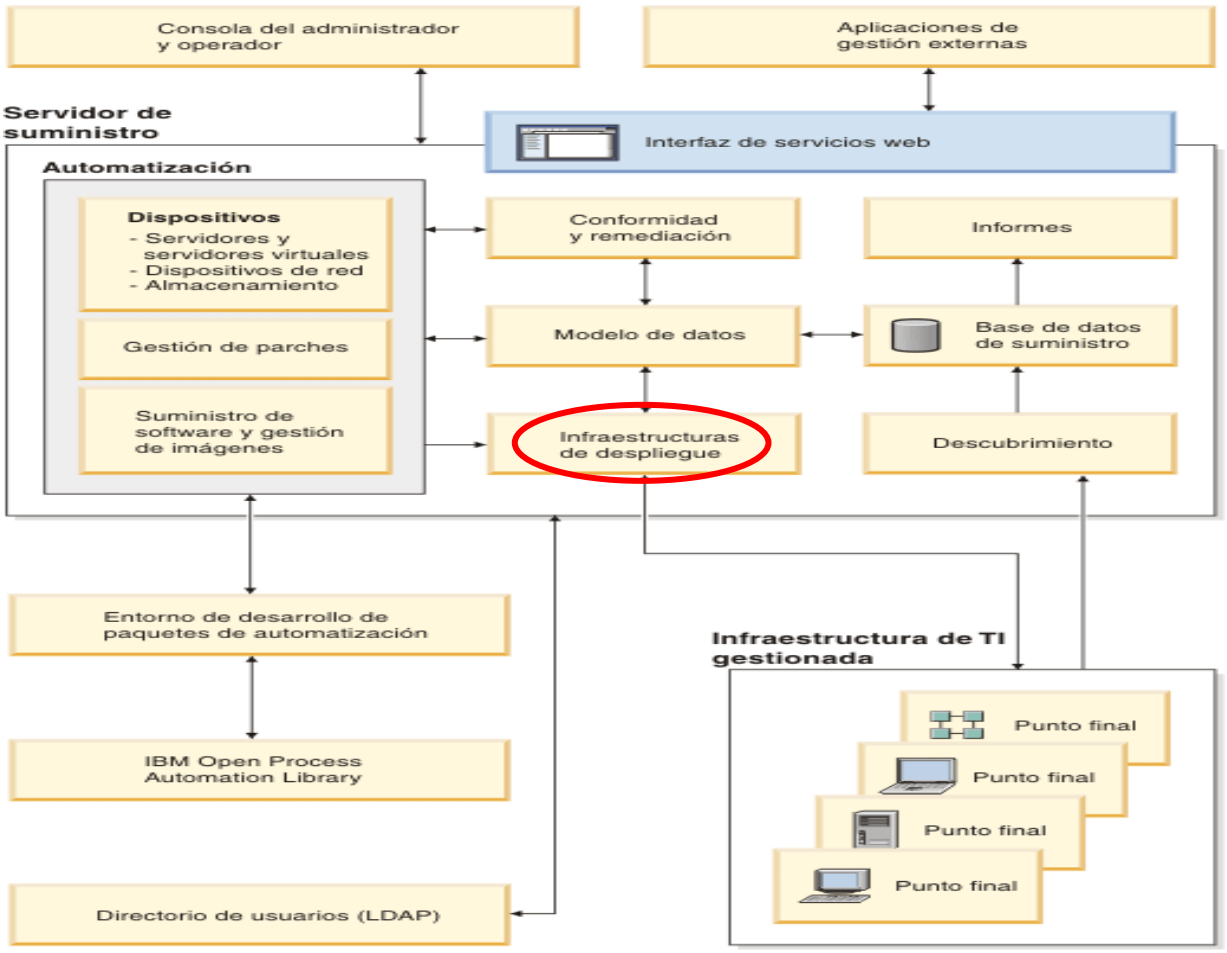


Figura 4 Arquitectura de IBM TIM.

Esta arquitectura cuenta una Consola de administración, en la cual se realizan las solicitudes, también posee un servidor de suministro o servidor de aprovisionamiento, el cual presenta un componente de automatización y tiene una relación con la Infraestuctura de despliegue. En esta última se encuentran los conectores o adaptadores, siendo estos los encargados de realizar las solicitudes de aprovisionamiento específicas dentro de los recursos de la organización.

1.4.3 Microsoft Provisioning System (MPS).

Aunque el Sistema de Aprovisionamiento de Microsoft (MPS) no se encuentre entre los cinco primeros software más potentes de administración de identidades a nivel mundial ha sido estudiado y analizado ya que se está haciendo uso de la plataforma .NET y estudiando sus tecnologías.

MPS es una plataforma extensible establecida en Windows que proporciona un marco basado en XML el cual permite crear soluciones personalizadas de aprovisionamiento para la web, datos y alojamiento de

aplicaciones. Esta solución incluye tareas tales como agregar nuevos usuarios, la actualización de las entradas de la guía, y aprovisionamiento de aplicaciones y servicios.

Contiene varios componentes que son instalados y configurados con las herramientas de despliegue de MPS. También incluye una variedad de componentes que soportan aprovisionamiento en el entorno del proveedor de servicio, dentro de los que se incluyen los siguientes (Microsoft, 2008):

- **Espacios de nombre:** los espacios de nombres son definiciones XML que contienen colecciones de uno o más procedimientos. Estos y sus colecciones de procedimientos son usados para invocar las funcionalidades requeridas procesando transacciones que implementan servicios de aprovisionamiento.
- **Procedimientos:** los procedimientos son también definiciones XML que se utilizan para invocar las funcionalidades de cualquiera de los proveedores o de otros procedimientos de espacios de nombres.
- **Motor de Aprovisionamiento:** el motor de aprovisionamiento es un servicio que recibe peticiones XML y realiza una serie de acciones de aprovisionamiento predefinidas o creadas en el contexto de una identidad segura específica.
- **Proveedores:** los proveedores son bibliotecas de vínculos dinámicos (DLL) que contiene componentes de procedimientos programados que hacen llamadas a la API para atender las solicitudes de aprovisionamiento. MPS cuenta en su instalación con un conjunto básico de los proveedores.
- **Bases de datos:** MPS incluye bases de datos de configuración, registro de transacciones, auditoría y servicios de recuperación, y recursos administrativos (Microsoft, 2008).

MPS presenta una arquitectura como se muestra a continuación:

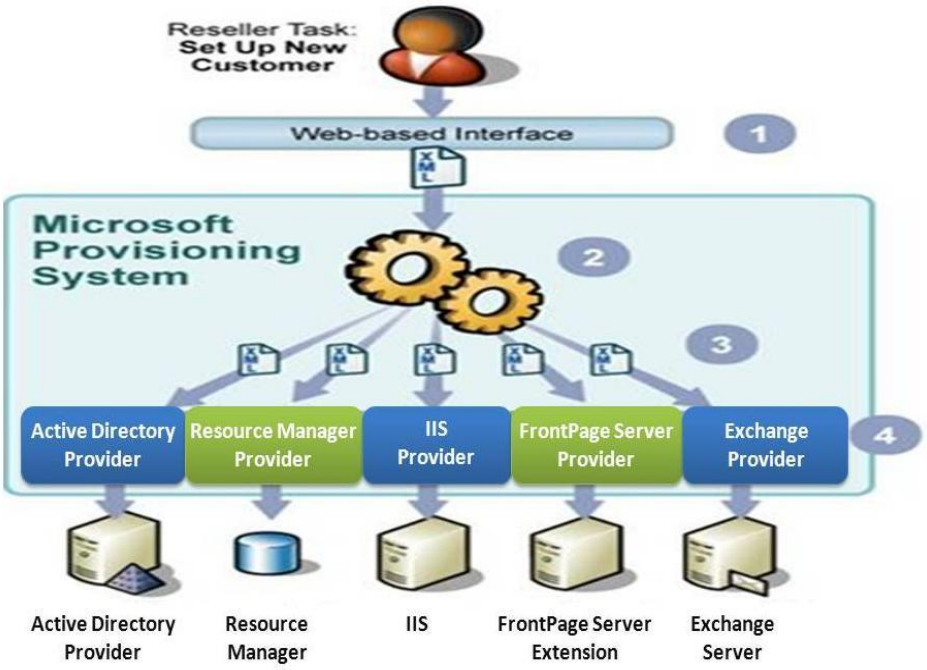


Figura 5 Arquitectura de Microsoft Provisioning System (MPS).

Para iniciar los servicios de aprovisionamiento son recibidas peticiones XML como entrada de MPS, seguidamente se convierte el código XML especificado en llamadas a la API para aplicaciones externas como el Directorio Activo de Microsoft, Servidor de Nombres de Dominio (DNS), Servicios de Información de Internet (IIS) y Servidor SQL de Microsoft (Microsoft SQL Server), donde las tareas adecuadas de aprovisionamiento son realizadas. El motor transaccional de provisión permite que las transacciones que fallen sean revertidas. También se puede registrar las operaciones de aprovisionamiento de referencia histórica y solución de problemas. Además, el sistema proporciona la utilización de una cola para la presentación de solicitudes XML que pueden ser ejecutadas más tarde.

1.5 Tecnologías y Herramientas

1.5.1 Tecnología .NET.

Microsoft .NET no es más que un grupo de tecnologías sobre las cuales Microsoft ha estado trabajando en estos últimos años, las cuales ha querido unir en un solo paquete. El objetivo fundamental de .NET es ofrecer una plataforma potente en la distribución de servicios, que permita la comunicación entre aplicaciones sin importar el lenguaje con el cual se hayan desarrollado y que a la vez sea fácil de usar.

Esta tecnología ofrece un entorno de desarrollo de aplicaciones llamado Visual Studio .NET, el mismo cuenta con varios lenguajes de programación, los cuales son: Visual Basic .NET, Visual C sharp, Visual FoxPro y Visual C++ .NET. Estos últimos han combinado las características de la mayoría de los lenguajes existentes en el mundo para de esta manera lograr un potente sistema de desarrollo (Microsoft, 2008).

1.5.2 Microsoft .NET Framework 4.

Microsoft .NET Framework 4 proporciona varias mejoras y características nuevas, entre las que se destacan adelantos en Lenguaje Común de Rutina (Common Language Runtime CLR) y la biblioteca de clases base (BCL), en el rendimiento, incluida una mayor compatibilidad con equipos multi-núcleo, recolección de elementos no utilizados en segundo plano y asociación del generador de perfiles en el servidor, además de incluir algunos avances en ASP.NET, los cuales se muestran a continuación:

- Más control sobre HTML, identificadores de elemento y hojas CSS personalizadas que facilitan enormemente la creación de formularios Web que admiten optimización del motor de búsqueda y son conformes a los estándares.
- Compatibilidad con formularios Web para nuevas mejoras de librerías de AJAX, incluida la compatibilidad integrada con redes de entrega de contenido (CDN).
- MVC, con nuevos métodos de aplicación auxiliar para las vistas, compatibilidad con aplicaciones MVC subdivididas y controladores asincrónicos (Microsoft, 2011).

1.5.3 Lenguaje de Programación C Sharp.

C Sharp o C#, como usualmente se le conoce, es un lenguaje de programación orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Utiliza el modelo de objetos de la plataforma .NET haciéndolo un lenguaje caracterizado por su sencillez, seguridad y alto nivel de productividad, siguiendo el mismo patrón de los lenguajes de programación modernos. Incluye un amplio soporte de estructuras, componentes, programación orientada a objetos, manipulación de errores que permite mayor flexibilidad en lo que se desea desarrollar. Este potente lenguaje también provee soporte para interfaces, estructuras y características de componentes orientados, como propiedades, eventos y construcciones declaradas (también llamados atributos) (Sierra, 2006).

1.5.4 Plataforma de Programación ASP .Net.

ASP.NET puede ser escrito en cualquier lenguaje soportado por este Framework, es decir: VB.net; C# y JScript.net. Es un lenguaje totalmente orientado a objetos. Cuenta con grandes ventajas entre las cuales se destacan:

- **Rendimiento**: la aplicación se compila en una sola vez al lenguaje nativo, y luego, en cada petición tiene una compilación Just In Time, es decir se compila desde el código nativo, lo que permite mucho mejor rendimiento. También permite el almacenamiento del caché en el servidor.
- **Rapidez en programación:** mediante diversos controles, ya que con unas pocas líneas y en menos de 5 minutos mostrar toda una base de datos y hacer rutinas complejas.

- **Simplicidad:** ASP.NET hace más fácil la realización de las tareas comunes, comenzando desde el envío de formularios y la autenticación del cliente hasta la implementación y la configuración de sitios.
- **Seguridad:** tiene diversas herramientas que garantizan la seguridad de las aplicaciones. Brinda esquemas de autenticación y autorización predeterminados para aplicaciones Web. Los programadores pueden adicionar, quitar o reemplazar fácilmente estos esquemas de acuerdo a los requerimientos de la aplicación (Microsoft, 2009).

1.5.5 IDE de desarrollo Visual Studio 2010 Ultimate.

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Microsoft Visual Studio 2010 Ultimate es la completa suite de herramientas, incluye aplicaciones de gestión de ciclo de vida de los equipos para garantizar resultados de calidad, además de diseño y la implementación.

Este IDE proporciona grandes facilidades en el desarrollo de un software como la inclusión de un editor de código que permite resaltar los errores en el código escrito, así como accesos directos hasta ellos para corregirlos rápidamente, presenta excelentes características de navegación y búsqueda dentro de nuestro código, utiliza IntelliSense para el auto-completamiento de código disminuyendo la complicidad del trabajo y la pérdida de tiempo (Microsoft, 2011).

1.5.6 Metodología de desarrollo de software.

Las metodologías de desarrollo de software abarcan todo el ciclo de vida del software, y se definen como “un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software". Adoptando la misma se obtiene un producto software con características deseables como:

- Existencias de reglas bien definidas.
- Verificaciones en cada etapa.
- Planificación y control.
- Comunicación efectiva.
- Fácil comprensión.
- Herramientas CASE (Computer Aided Software Engineering).
- Soporte de la reutilización y mantenimiento de software.

1.6 Metodología de desarrollo Microsoft Solution Framework (MSF) Ágil.

El desarrollo ágil de software es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. El proceso de desarrollo en MSF consta de 5 fases, las cuales junto a los modelos mencionados anteriormente, complementan el ciclo de desarrollo de software en esta metodología (Jackson, 2006).

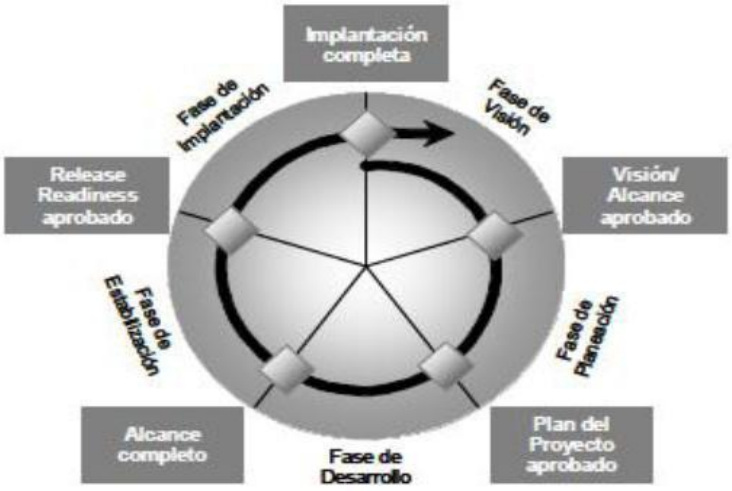


Figura 6 Modelo de Proceso de Desarrollo de Aplicaciones MSF.

- **Fase 1**: Visión. En esta fase el equipo y el cliente definen los requerimientos del negocio y los objetivos generales del proyecto. La fase culmina con el hito Visión y Alcance aprobados.
- **Fase 2:** Planeación. Durante la fase de planeación el equipo crea un borrador del plan maestro del proyecto, además de un cronograma y de la especificación funcional del proyecto. Esta fase culmina con el hito Plan del proyecto aprobado.
- **Fase 3:** Desarrollo. Esta fase involucra una serie de liberaciones internas del producto, desarrollados por partes para medir su progreso y para asegurarse que todos sus módulos o partes están sincronizados y pueden integrarse. La fase culmina con el hito Alcance completo.
- **Fase 4:** Estabilización. Esta fase se centra en probar el producto. El proceso de prueba hace énfasis en el uso y el funcionamiento del producto en las condiciones del ambiente real. La fase culmina con la liberación terminada aprobada.
- **Fase 5:** Implantación: En esta fase el equipo implanta la tecnología y los componentes utilizados por la solución, estabiliza la implantación, apoya el funcionamiento y la transición del proyecto, y obtiene la aprobación final del cliente. La fase termina con el hito Implantación completa.

1.7 Patrones de diseño.

Un patrón de diseño es una solución general a un problema común en el diseño de software, es una descripción o patrón de cómo resolver un problema y que puede ser usado en diferentes situaciones.

Muchas veces son más útiles con algunos lenguajes de programación y en otros casos son aplicables a cualquier lenguaje. Ayudan a los desarrolladores a la reutilización de código haciendo de manera más práctica describir ciertos aspectos de la organización de un programa**.** Estos Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO) (Wesley, 2003).

Los patrones se clasifican según su propósito en:

Patrones de Creación: Tratan la creación de instancias o sobre qué objetos un objeto delegará responsabilidades (Wesley, 2003).

- **Abstract Factory:** proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar su clase concreta.
- **Builder:** permite a un objeto construir un objeto complejo especificando sólo su tipo y contenido.
- **Factory Method:** define una interfaz para crear un objeto dejando a las subclases decidir el tipo específico al que pertenecen.
- **Prototype:** permite a un objeto crear objetos personalizados sin conocer su clase exacta a los detalles de cómo crearlos.
- **Singleton:** garantiza que solamente se crea una instancia de la clase y provee un punto de acceso global a él.

Patrones Estructurales: Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad, describiendo así la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros (Wesley, 2003).

- **Adapter:** convierte la interfaz que ofrece una clase en otra esperada por los clientes.
- **Bridge:** desacopla una abstracción de su implementación y les permite variar independientemente.
- **Composite:** permite gestionar objetos complejos e individuales de forma uniforme.
- **Decorator:** extiende la funcionalidad de un objeto dinamizante de tal modo que es transparente a sus clientes.
- **Facade:** simplifica los accesos a un conjunto de objetos relacionados proporcionando un objeto de comunicación.
- **Flyweight:** usa la compartición para dar soporte a un gran número de objetos de grano fino de forma eficiente.
- **Proxy:** proporciona un objeto con el que se controla el acceso a otro objeto.

Patrones de Comportamiento: Tratan la interacción y cooperación entre clases. Organizan, manejan y combinan comportamientos (Wesley, 2003).

- **Chain of Responsability:** evita el acoplamiento entre quien envía una petición y el receptor de la misma.
- **Command:** encapsula una petición de un comando como un objeto.

- **Interpreter:** dado un lenguaje define una representación para su gramática y permite interpretar sus sentencias.
- **Iterator:** acceso secuencial a los elementos de una colección.
- **Mediator:** define una comunicación simplificada entre clases.
- **Memento:** captura y restaura un estado interno de un objeto.
- **Observer:** una forma de notificar cambios a diferentes clases dependientes.
- **State:** modifica el comportamiento de un objeto cuando su estado interno cambia.
- **Strategy:** define una familia de algoritmos, encapsula cada uno y los hace intercambiables.
- **Template Method:** define un esqueleto de algoritmo y delega partes concretas de un algoritmo a las subclases.
- **Visitor:** representa una operación que será realizada sobre los elementos de una estructura de objetos, permitiendo definir nuevas operaciones sin cambiar las clases de los elementos sobre los que opera.

1.7.1 Modelo Provider.

El modelo Provider es un patrón de diseño formulado por Microsoft para usarlo en ASP.NET Starter Kits y formalizarlo en .NET versión 2.0. Este patrón puede incluirse dentro de los estructurales. Es usado para permitir a una aplicación escoger una de muchas implementaciones en la configuración de la aplicación, por ejemplo, proporciona acceso a diferentes orígenes de datos para recuperar la información del login, o para usar diferentes metodologías de almacenamiento como puede ser una base de datos, binaria a disco, XML, etc (Schackow, 2006).

El modelo Provider de .NET permite a un componente tener múltiples implementaciones usando un enfoque abstracto del patrón Factory. Los proveedores son subclases de la clase base del proveedor y típicamente instanciada usando un método Factory.

Con la ayuda del modelo Provider, ASP.NET 2.0 puede ser configurado para almacenar un estado prácticamente en cualquier lugar. Algunas compañías prefieren adquirir proveedores personalizados de terceros. Otros, sin embargo, prefieren escribir el suyo propio, ya sea porque ningún proveedor adecuado está disponible fuera de la plataforma, o porque desean adaptar ASP.NET para medios de almacenamiento tradicionales (Microsoft, 2005).

1.7.2 Patrón Factory Method.

La idea del patrón Factory Method es separar la creación de ciertas clases de las funciones que consumen esas clases. Mientras las clases implementan una interfaz común, o derivan de una clase común, una

característica puede encapsular la creación de clases con un mecanismo genérico que no requiere ninguna difícil dependencia en tiempo de compilación.

En otras palabras, una característica que hace uso del patrón Factory Method no tiene una fuerte referencia con un tipo en concreto. En lugar de una referencia a clases de entidades a través de interfaces o referencias de la clase base, aplaza la creación real de implementaciones concretas de alguna otra pieza de código (Schackow, 2006).

1.7.3 Patrón Singleton.

El patrón Singleton se utiliza cuando un desarrollador quiere que exista una única instancia de una clase dentro de una aplicación, esto generalmente ocurre cuando la instanciación de un objeto y la destrucción de una clase son muy caros, y por lo tanto es posible que sólo se quiera una instancia para incurrir jamás en la sobrecarga de la construcción de objetos. El patrón Singleton se utiliza cuando se quiere mediar en el acceso a un recurso específico con una simple instancia de objeto que bloquea el acceso al recurso, es posible implementar la sincronización lógica dentro de la instancia de objeto de modo que sólo un subproceso activo puede tener acceso al recurso a la vez (Schackow, 2006).

1.7.4 Patrón Command

El Patrón Command permite que las peticiones a un objeto existan como un objeto. Esto significa que si es enviada una solicitud para alguna función a un objeto, el objeto de comando puede albergar la petición dentro del objeto. Esto es útil en el caso de deshacer o rehacer una acción, o simplemente guardar una acción en una cola de la petición en un objeto (Lasater, 2007).

Este patrón asegura que cada objeto recibe sus propios comandos y proporciona una disociación entre el emisor y el receptor. Un remitente es un objeto que invoca una operación, y un receptor es un objeto que recibe la solicitud y actúa sobre ella (dofactory, 2001).

1.8 Conclusiones.

En el presente capítulo se ofrece una panorámica del aprovisionamiento de usuarios, proceso clave dentro de un Sistema de Administración de Identidades. Además se describen algunas características de sistemas de aprovisionamiento líderes en el mercado revisando algunas de sus arquitecturas de conectores y se explica brevemente el funcionamiento de los conectores predefinidos a implementar. Se muestran tecnologías, herramientas y metodologías de desarrollo pertenecientes a la plataforma .NET. Al finalizar el capítulo se arribaron a las siguientes conclusiones:

- La creación de los conectores para los recursos base de datos, servicio de directorio, servidor de correo y servidor de mensajería instantánea permitirá la realización de tareas de aprovisionamiento en los mismos.

- Las tecnologías y herramientas utilizadas serán aquellas pertenecientes a la plataforma .NET por el conocimiento del equipo de desarrollo además de ser las que han venido siendo utilizadas en anteriores iteraciones del sistema.
- El estudio de diferentes Sistemas de Aprovisionamiento permitió un mayor entendimiento de sus procesos y funcionamiento.
- Con la creación de un correcto diseño de arquitectura de conectores se logrará una mejor compresión del producto para una futura implementación.
- Los patrones de diseño que serán usados en el desarrollo para lograr la extensibilidad de las funcionalidades del Módulo de Conectores son: Provider, Factory Method, Singleton y Facade; y para hacer posible que el módulo soporte las transacciones será utilizado el patrón Command.

Capítulo 2

Visión y Planificación

2.1 Introducción.

En el proceso de desarrollo de un software informático es necesario explicar algunos conceptos y requerimientos de una forma sencilla y descifrable para cualquier persona que esté implicada en el desarrollo. La metodología del marco de trabajo de solución de Microsoft (*Microsoft Solution Framework)* provee una serie de artefactos y actividades para llevar una idea clara del proceso de desarrollo del software en su ciclo de vida.

Esta metodología plantea obtener una visión clara de lo que se desea desarrollar y propone realizar una planificación que guíe al grupo de trabajo hacia la construcción exitosa del sistema.

En este capítulo se generan algunos artefactos siguiendo la metodología de desarrollo de software propuesta. También se hace una descripción de los escenarios y requisitos que conformarán el módulo, además se presenta la línea base de la arquitectura propuesta describiendo sus aspectos más representativos.

2.2 Capturar visión del proyecto.

El desarrollo de un producto informático requiere en un inicio del establecimiento claro de la visión del proyecto, visión que puede sufrir cambios a medida que se va desarrollando el producto como resultado del clima de negocios, por lo tanto si la visión llega a sufrir modificaciones el producto se deberá ir ajustando a estas modificaciones. Capturar y comunicar esta visión central es el elemento más importante para mantener un proyecto enfocado. Cuando se tiene una visión clara, se comienza a entender cuál será la relación de los usuarios con el producto. La visión y sus actividades están relacionadas con la creación de la base sólida sobre la cual un producto puede ser creado (Microsoft, 2006).

2.2.1 Descripción del módulo.

Debido a la gran amplitud y cantidad de personal con el que cuentan las organizaciones y empresas hoy en día, se hace engorrosa la tarea manual de aprovisionamiento de cuentas de usuarios. Un Sistema de Aprovisionamiento de Usuarios es el encargado de realizar esta tarea de forma automática, centralizada y segura.

El Subsistema de Aprovisionamiento de Usuarios del Sistema Administración de Identidades es el encargado de crear las cuentas de usuarios y sus privilegios en las aplicaciones que hacen uso de sus servicios. Este subsistema cuenta con varios módulos, siendo el Módulo de Conectores el encargado de realizar el aprovisionamiento de las cuentas de usuario de manera transaccional en los recursos con los que cuente la institución. Para una primera versión deben quedar desarrollados conectores para 4 tipos de recursos diferentes.

2.2.2 Definir Personas.

Las personas no interactúan directamente con el Módulo de Conectores debido a que este es solamente una librería de clases encargada del aprovisionamiento. A continuación se muestra el proceso que interactúa directamente con este módulo.

Servicio de Aprovisionamiento: Realiza las peticiones de aprovisionamiento a ser ejecutadas.

2.3 Planificación.

La Planificación como fase comienza en un período temprano en la vida de un proyecto. En esta fase el grupo de trabajo analiza, identifica y prioriza los requerimientos que describen de forma total la solución, conjuntamente se genera algunos artefactos, entre los cuales encontramos la lista de escenarios y la lista de requerimientos de calidad de servicios (Microsoft, 2006).

2.3.1 Determinar la longitud de las iteraciones.

Una iteración es un conjunto de tareas programadas para ocurrir en un determinado periodo de tiempo. Determinar la longitud de una iteración incluye tener en cuenta factores claves incluyendo la fecha de entrega del proyecto, el tamaño de los escenarios y el tiempo de integración (Microsoft, 2006).

Dada la fecha de entrega se ha considerado realizar tres iteraciones, las cuales tomarían un tiempo de 10 semanas aproximadamente para su total desarrollo.

Iteración #1: Se implementarán los escenarios de mayor prioridad en el módulo, siendo necesario un total de 4 semanas aproximadamente.

Iteración #2: Se implementarán los escenarios de prioridad media en el módulo, siendo necesario un total de 5 semanas aproximadamente.

Iteración #3: Se implementarán los escenarios de prioridad baja en el módulo, siendo necesario un total de 1 semana aproximadamente.

2.3.2 Escenarios del módulo.

Los escenarios estimados son usados para proveer una imagen general para entender cuanto esfuerzo requiere el escenario. Estos estimados ayudan al analista de negocios a tomar decisiones de acuerdo a la prioridad (Microsoft, 2006). (Ver Anexo 1)

Para alcanzar los objetivos del proyecto se identificaron los siguientes escenarios:

Lista de Escenarios.

- Aprovisionar en el recurso Servidor de Base Datos Oracle 11g.
- Aprovisionar en el recurso Servicio de Directorio OpenLdap.
- Aprovisionar en el recurso Servidor de Correo Zimbra.
- Aprovisionar en el recurso Servidor de Mensajería Instantánea Openfire.

Lista de sub escenarios válidos para cada escenario:

- Gestionar Conexión.
  - Inicializar conexión.
  - Abrir conexión.
  - Cerrar conexión.
- Gestionar Cuentas de Usuario.
  - Crear cuenta.
  - Eliminar cuenta.
  - Modificar cuenta.
  - Desactivar cuenta.
  - Bloquear cuenta (No queda incluido dentro del escenario “Aprovisionar en el recurso Servicio de Directorio OpenLdap” ya que depende de la configuración que tenga el servicio del directorio).
  - Activar cuenta.
- Gestionar Roles (Solo es válido para los escenarios “Aprovisionar en el recurso Servidor de Base Datos Oracle 11g” y “Aprovisionar en el recurso Servidor de Correo Zimbra”).
  - Asignar rol.
  - Eliminar rol.
  - Buscar roles.
  - Listar roles.

2.3.3 Priorizar escenarios del sistema.

Los escenarios se priorizan en dependencia de la importancia que tienen para los usuarios y para la validez de la aplicación. Para priorizar la lista de escenarios es necesario identificar aquellos que sean más

importantes en el momento de implementar y de esta manera sean desarrollados en las primeras iteraciones (Microsoft, 2006). Calificar al escenario con 5 puntos equivale a tener una prioridad “Alta”, con 4 puntos “Media” y con 3 puntos “Baja”.

Tabla 1 Listado de Escenarios y sus prioridades.

No.	Escenario	Prioridad
1	Aprovisionar en el recurso Servidor de Base Dato Oracle 11g.	5
2	Aprovisionar en el recurso Servicio de Directorio OpenLdap.	4
3	Aprovisionar en el recurso Servidor de Correo Zimbra.	4
4	Aprovisionar en el recurso Servidor de Mensajería Instantánea Openfire.	3

2.3.4 Estimar escenarios de mayor prioridad.

La estimación de los escenarios es llevada a cabo para tener conocimiento de cuanto trabajo es necesario realizar para llegar a completar un escenario (Microsoft, 2006).

Tabla 2 Escenarios de mayor prioridad.

No.	Escenarios	Prioridad	Riesgo	Esfuerzo(días)
1	Aprovisionar en el recurso Servidor de Base Datos Oracle 11g.	5	Alto	23
2	Aprovisionar en el recurso Servicio de Directorio OpenLdap.	4	Medio	15
3	Aprovisionar en el recurso Servidor de Correo Zimbra.	4	Medio	16

2.3.5 Cronometrar escenarios.

Los escenarios son programados para el desarrollo y pruebas preliminares en una iteración específica. El plan de iteración refleja el entendimiento más actual de lo que debe llevarse a cabo dentro de una iteración. Dependiendo de la prioridad de los escenarios se decide cuáles de ellos se implementarán en las primeras iteraciones del ciclo.

Iteración 1: Se codificarán los escenarios que presentan prioridad alta.

- ✓ Aprovisionar en el recurso Servidor de Base Dato Oracle 11g.

Iteración 2: Se codificarán los escenarios que presentan prioridad media.

- ✓ Aprovisionar en el recurso Servicio de Directorio OpenLdap.

- ✓ Aprovisionar en el recurso Servidor de Correo Zimbra.

Iteración 3: Se codificarán los escenarios que presentan prioridad baja.

- ✓ Aprovisionar en el recurso Servidor de Mensajería Instantánea Openfire.

Una vez definidos los escenarios se gráfica la planificación realizada según las iteraciones necesarias para el desarrollo del sistema. (Ver Anexo 3).

Tabla 3 Diagrama de Iteración.

No.	Nombre de la Tarea	Duración	Comienzo	Final	Predecesor
1	**Primera Iteración.**	23 días	01/03/11	31/03/11	
2	Aprovisionar en el recurso Servidor de Base Datos Oracle 11g.	23 días	01/03/11	31/03/11	
3	➢ Gestionar Conexión.	9 días	01/03/11	11/03/11	
4	➢ Gestionar Cuentas de Usuario.	14 días	14/03/11	31/03/11	3
5	➢ Gestionar Roles.	2 días	25/03/11	27/03/11	3
6	**Segunda Iteración.**	31 días	01/04/11	13/05/11	
7	Aprovisionar en el recurso Servicio de Directorio OpenLdap.	15 días	01/04/11	21/04/11	
8	➢ Gestionar Conexión.	6 días	01/04/11	08/04/11	
9	➢ Gestionar Cuentas de Usuario.	9 días	11/04/11	21/04/11	8
10	Aprovisionar en el recurso Servidor de Correo Zimbra.	16 días	22/04/11	13/05/11	
11	➢ Gestionar Conexión.	6 días	22/04/11	29/04/11	
12	➢ Gestionar Cuentas de Usuario.	10 días	02/05/11	13/05/11	11
13	➢ Gestionar Roles.	2 días	02/05/11	04/05/11	11
14	**Tercera Iteración.**	9 días	11/05/11	23/05/11	
15	Aprovisionar en el recurso Servidor de mensajería Instantánea Openfire.	9 días	11/05/11	23/05/11	
16	➢ Gestionar Conexión.	3 días	11/05/11	13/05/11	
17	➢ Gestionar Cuentas de Usuario.	6 días	16/05/11	23/05/11	16

2.3.6 Requerimientos de calidad de servicio.

Los requerimientos de calidad de servicios representan una lista de requisitos no funcionales del sistema (Microsoft, 2006). A continuación se muestra una lista de los requerimientos con los que debe contar el módulo. (Ver Anexo 4)

- Diseño e Implementación
  - El módulo debe contar con un conjunto de patrones de diseño que permitan la reutilización del código, así como una librería de clases (API) para la futura extensión del módulo.
  - Debe contar con un modelo de transacciones que haga posible que el módulo ejecute las funciones de aprovisionamiento de manera transaccional.
  - El código debe cumplir con los estándares de codificación establecidos por el equipo de desarrollo.
- Seguridad
  - La seguridad en el módulo estará implementada por protocolos seguros SSL/TSL entre otros que soporte el tipo de recurso específico con el que se va a realizar la conexión para garantizar que esta conexión y el envío de datos se realice de forma satisfactoria.

2.3.7 Descripción de los escenarios.

Los escenarios proporcionan un medio para que los desarrolladores, usuarios finales y trabajadores lleguen a una comprensión común del sistema propuesto. Estos son usados para la validación de la arquitectura a lo largo del desarrollo. La descripción de estos permite a los desarrolladores entender las necesidades de las personas que interactúan con los escenarios, además las descripciones deben realizarse de manera que permitan realizar las pruebas (Microsoft, 2006). (Ver Anexo 2).

Tabla 4 Descripción del Escenario “Gestionar Conexión en Servidor de Correo ZImbra”.

Título:	**Inicializar Conexión.**
Área:	Aprovisionar en el recurso Servidor de Correo Zimbra. Gestionar Conexión.
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando se selecciona un recurso en el cual se va a aprovisionar, los parámetros de conexión son cargados son inicializados pero no se llega a abrir una conexión con. En caso de algún fallo se notifica el

	error.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	IcZ1
Orden de magnitud:	1

Tabla 5 Descripción del Escenario “Gestionar Conexión en Base de Datos Oracle 11g”.

Título:	**Abrir Conexión.**
Área:	Aprovisionar en el recurso Base de Datos Oracle 11g. Gestionar Conexión.
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando se selecciona un recurso en el cual se va a aprovisionar, los parámetros de conexión son cargados y utilizados para establecer la comunicación con dicho recurso. En caso de fallar la conexión se notifica el error.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	AcO1
Orden de magnitud:	1

Tabla 6 Descripción del Escenario “Crear Cuenta en Base de Datos Oracle 11g”.

Título:	**Crear cuenta.**
Área:	Aprovisionar en el recurso Base de Datos Oracle 11g. Gestionar cuenta.
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Crear Cuenta” y en cual recurso va a aprovisionar. Después

	se establece la conexión mediante un usuario con privilegios administrativos y se procede a validar los datos de la nueva cuenta, en caso de no ocurrir ningún error se crea la cuenta ejecutando consultas SQL en el servidor de Base de Datos Oracle. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	CcO1
Orden de magnitud:	1

Tabla 7 Descripción del Escenario “Eliminar cuenta en Servicio de Directorio OpenLdap”.

Título:	**Eliminar cuenta.**
Área:	Aprovisionar en el recurso Servicio de Directorio OpenLdap. Gestionar cuenta.
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Eliminar Cuenta” en un recurso específico. Se establece la conexión con el recurso mediante un usuario con privilegios administrativos y se procede a la eliminación de la cuenta seleccionada. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Media
Identificador:	EcL2
Orden de magnitud:	2

Tabla 8 Descripción del Escenario “Modificar cuenta en Servidor de Correo Zimbra”.

Título:	**Modificar cuenta.**

Área:	Aprovisionar en el recurso Servidor de Correo Zimbra. Gestionar cuenta.
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Modificar Cuenta” y elige la cuenta que será objeto de cambios. Se establece la conexión con el recurso mediante un usuario con privilegios administrativos, se validan los nuevos datos de la cuenta y en caso que no ocurra error se procede a la modificación con los nuevos datos de la cuenta a través de comandos enviados hacia la consola del servidor. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Media
Identificador:	McZ2
Orden de magnitud:	2

Tabla 9 Descripción del Escenario “Desactivar cuenta Servidor de Mensajería Instantánea Openfire”.

Título:	**Desactivar cuenta.**
Área:	Aprovisionar en el recurso Servidor de Mensajería Instantánea Openfire. Gestionar cuenta.
Iteración:	3
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Desactivar Cuenta” y elige la cuenta a desactivar en un recurso específico. Se establece la conexión con el recurso mediante un usuario con privilegios administrativos y se produce la desactivación de la cuenta mediante consultas y procedimientos realizados al Servicio de Directorio OpenLdap que este Servidor tenga asociado. En caso de ocurrir

	algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Baja
Identificador:	DcIM3
Orden de magnitud:	3

Tabla 10 Descripción del Escenario “Activar cuenta Servidor de Correo Zimbra”.

Título:	**Activar cuenta.**
Área:	Aprovisionar en el recurso Servidor de Correo Zimbra. Gestionar cuenta.
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Activar Cuenta” y elige la cuenta que será activada en un recurso específico. Se establece la conexión con el recurso mediante un usuario con privilegios administrativos y se produce la activación de la cuenta a través de comandos enviados hacia la consola del servidor. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Media
Identificador:	AcZ2
Orden de magnitud:	2

Tabla 11 Descripción del Escenario “Bloquear cuenta en Base de Datos Oracle 11g”.

Título:	**Bloquear cuenta.**
Área:	Aprovisionar en el recurso Base de Datos Oracle 11g. Gestionar cuenta.
Iteración:	1
Estado:	Activo

Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza de manera automática cuando un usuario realiza una determinada cantidad de intentos fallidos en un sistema que se autentique mediante la Base de Datos Oracle. El sistema procede a bloquear la cuenta automáticamente por un período de tiempo determinado.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	BcO1
Orden de magnitud:	1

Tabla 12 Descripción del Escenario “Cerrar conexión en Servidor de Correo Zimbra.”

Título:	**Cerrar conexión.**
Área:	Aprovisionar en el recurso Servidor de Correo Zimbra. Gestionar conexión.
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario se inicia de manera automática. Al ser ejecutada y finalizada una tarea de aprovisionamiento entonces se cierra la conexión con la consola de comandos del Servidor de Correo.
Historial:	Inicio de descripción del escenario.
Prioridad:	Media
Identificador:	CcZ2
Orden de magnitud:	2

Tabla 13 Descripción del Escenario “Asignar rol en Base de Datos Oracle 11g.”

Título:	**Asignar rol.**
Área:	Aprovisionar en el recurso Base de Datos Oracle 11g. Gestionar rol.
Iteración:	1
Estado:	Activo

Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Asignar rol” además de la cuenta y el recurso en el cual va a realizarse esta acción. Después se establece la conexión con el recurso mediante un usuario con privilegios, son validados los datos y en caso de no ocurrir ningún error se asigna el rol a la cuenta de usuario ejecutando consultas SQL en el servidor de Base de Datos. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	ArO1
Orden de magnitud:	1

Tabla 14 Descripción del Escenario “Eliminar rol en Base de Datos Oracle 11g.”

Título:	**Eliminar rol.**
Área:	Aprovisionar en el recurso Base de Datos Oracle 11g. Gestionar rol.
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Eliminar rol” además de la cuenta y el recurso en el cual va a realizarse esta acción. Después se establece la conexión con el recurso mediante un usuario con privilegios, son validados los datos y en caso de no ocurrir ningún error queda eliminado el rol de la cuenta de usuario ejecutando consultas SQL en el servidor de Base de Datos. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	ErO1
Orden de	1

magnitud:	

Tabla 15 Descripción del Escenario “Buscar roles en Servidor de Correo ZImbra.”

Título:	**Buscar roles.**
Área:	Aprovisionar en el recurso Servidor de Correo ZImbra.Gestionar rol.
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Buscar roles (clases de servicio)” además de la cuenta y el recurso en el cual va a realizarse esta acción. Después se establece la conexión con el recurso mediante un usuario con privilegios, son validados los datos y en caso de no ocurrir ningún error se muestra la clase de servicio a la cual pertenece la cuenta de usuario seleccionada, estas acciones son ejecutadas a través de comandos enviados a la consola del servidor de correo Zimbra. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	BrZ1
Orden de magnitud:	1

Tabla 16 Descripción del Escenario “Listar roles en Servidor de Correo Zimbra“.

Título:	**Listar roles.**
Área:	Aprovisionar en el recurso Base de Datos Oracle 11g. Gestionar rol.
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz

	Web la opción “Listar roles (clases de servicio)” y el recurso en el cual va a realizarse esta acción. Después se establece la conexión con el recurso mediante un usuario con privilegios, son validados los datos y en caso de no ocurrir ningún error se muestran todas las clases de servicio existentes en el servidor de correo Zimbra, estas acciones son ejecutadas a través de comandos enviados a la consola del servidor de correo Zimbra. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	LrZ1
Orden de magnitud:	1

2.4 Conclusiones.

Luego de finalizar el capítulo se arribaron a las siguientes conclusiones:

- La descripción del Módulo de Conectores permitió conocer de una mejor manera en que consiste el mismo.
- Es necesario determinar la longitud de las iteraciones y establecer los escenarios de mayor prioridad para garantizar el desarrollo de estos en las primeras etapas.
- El plan de iteraciones sirvió de conocimiento para calcular el tiempo estimado de desarrollo.
- Los requisitos de calidad de servicio deben quedar correctamente plasmados para conocer cuáles son las características adicionales con las que se debe cumplir.
- Mediante una correcta identificación y descripción de los escenarios se obtiene una mejor comprensión de las funcionalidades del módulo.

Capítulo 3

Construcción y Pruebas

3.1 Introducción.

En el presente capítulo se presenta la arquitectura del sistema de manera general y se muestra dónde es que se encuentra ubicado el Módulo de Conectores, también es realizado el diagrama de clases que será implementado. Son utilizados diferentes patrones que permiten la extensibilidad del módulo además de dar la posibilidad de ejecutar las operaciones de manera transaccional. Se describe el funcionamiento de los diferentes recursos a aprovisionar. Quedan descritas las funcionalidades de las clases y de los conectores específicos siendo finalmente presentados los resultados de las pruebas unitarias realizadas al módulo.

3.2 Arquitectura.

El subsistema de Aprovisionamiento del Sistema de Administración de Identidades está compuesto por los siguientes módulos: Servicio de Aprovisionamiento, Aplicación para la Gestión de Solicitudes de Aprovisionamiento, Motor de Aprovisionamiento y Portal de Administración. Cada uno de estos componentes sigue una arquitectura en capas. Las librerías de clases que constituyen el Módulo de Conectores estarán integradas al Servicio de Aprovisionamiento, este servicio es el punto común para el resto de los módulos del subsistema y es quien a través de las librerías realizará las funciones de provisión.

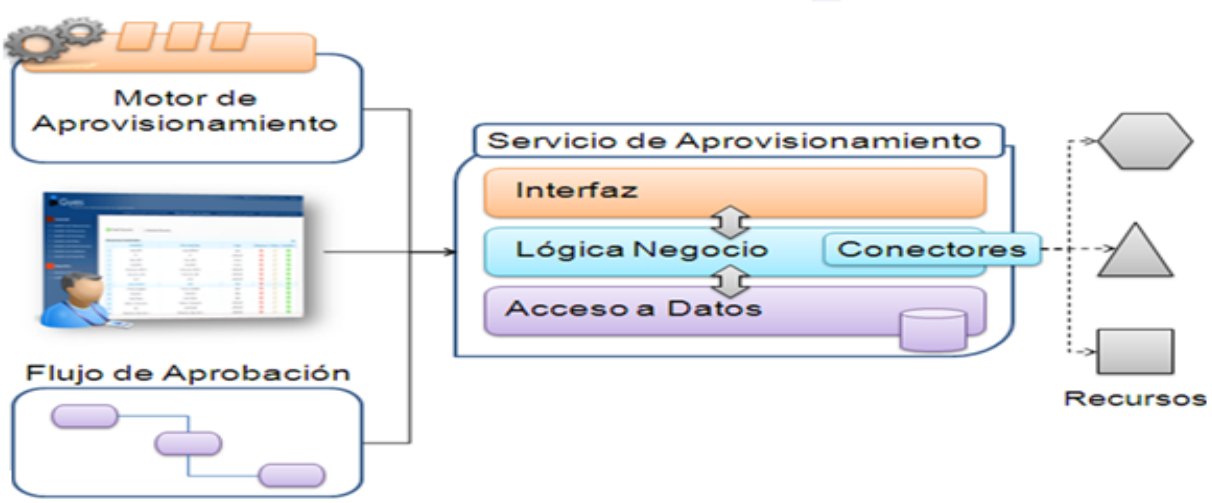


Figura 7 Arquitectura del Servicio de Aprovisionamiento

Según la figura 7, los conectores se utilizan en la capa de negocio. Esta capa recibe información (datos de las cuentas de usuarios), de las interfaces externas (otros módulos) que desean realizar operaciones de aprovisionamiento, a través de la Interfaz; de la capa de Acceso a Datos se extrae información referente a los parámetros de conexión de los diferentes recursos. Finalmente son invocadas las funcionalidades de los conectores a partir de los datos provenientes de las diferentes capas.

3.3 Diagrama de clases.

Los diagramas de clases ayudan a entender la estructura de clases del proyecto. Se pueden utilizar para personalizar, compartir y presentar la información del proyecto con los demás. Estos diagramas muestran sus clases, atributos y las relaciones (Microsoft, 2010).

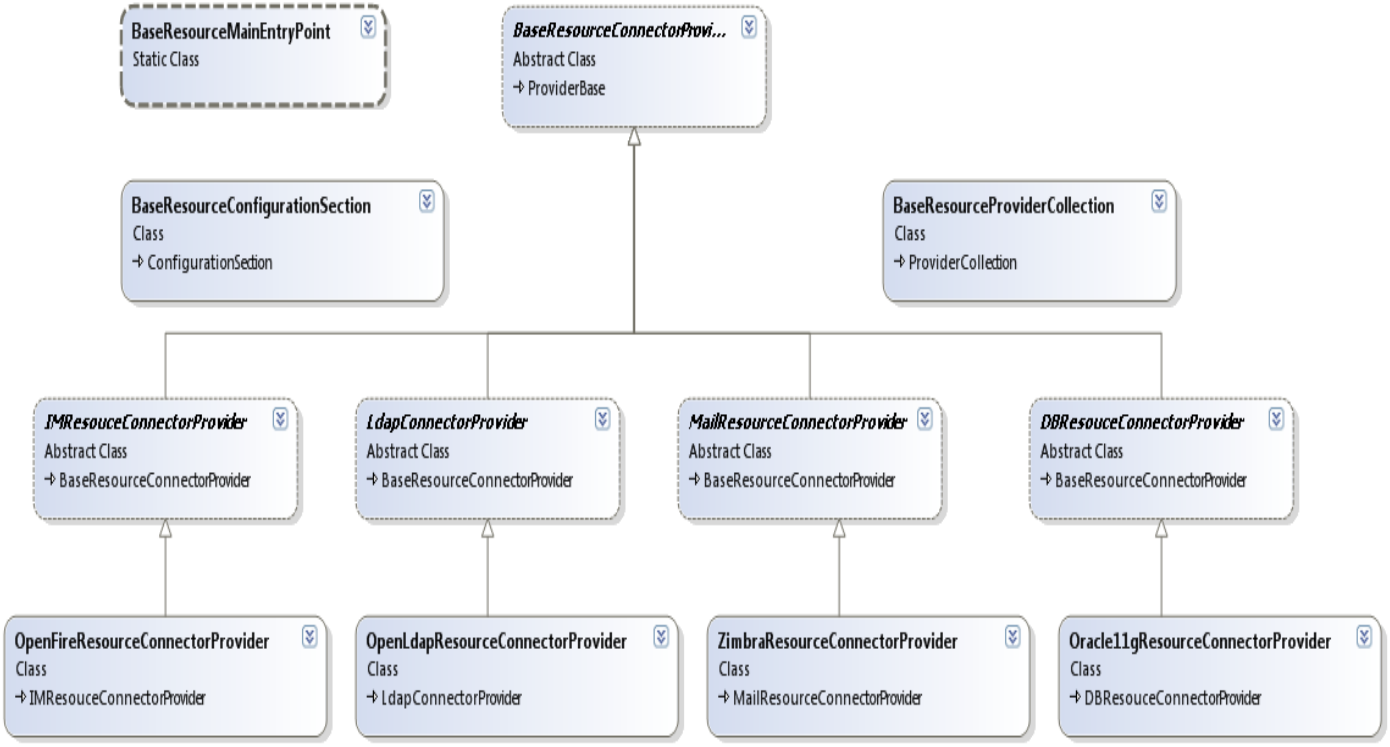


Figura 8 Fragmento del Diagrama de clases del Módulo de Conectores utilizando el patrón Provider.

3.4 Utilización del Modelo Provider en el Módulo de Conectores.

El Patrón Proveedor es extremadamente simple, este proporciona la funcionalidad de una API. Un proveedor es más que un contrato entre el API y la lógica de negocios / Capa de abstracción de datos (Howard, 2004).

El Modelo de Proveedor permite cambiar la lógica personalizada en la aplicación de tal manera que ningún fragmento del código ya existente necesita ser tocado o recompilado. Sin embargo, hay proveedores de APIs bien definidas que se pueden codificar en contra de crear su propia lógica de negocios personalizada

y reglas de negocio. Si se desea, se pueden escribir aplicaciones para tener una dependencia directa del código personalizado, pero este definitivamente no es un requisito (Schackow, 2006). La implementación de un proveedor puede ser definida en el web.config, o sea en este fichero de configuración se define cual será la clase que implementará cada proveedor aquí descrito. (Ver fragmento de código #1).

A continuación se muestra una imagen que muestra un ejemplo de utilización del Modelo Provider en el Módulo de Conectores.

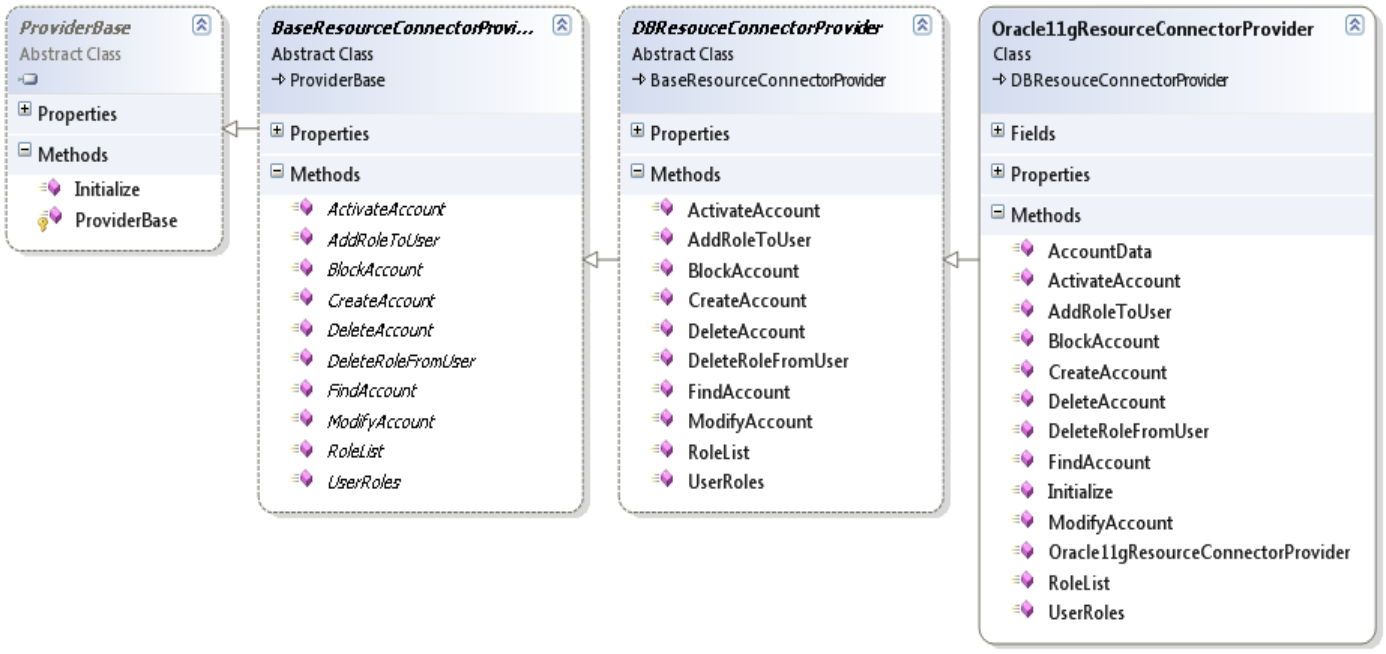


Figura 9 Estructura de clases con la utilización del Modelo Provider.

De la clase ProviderBase se extienden las funcionalidades por la clase abstracta llamada BaseResourceConnectorProvider que será la que defina cada uno de los métodos. Por otro lado DBResourceConnectorProvider es una clase genérica ya que existen varias implementaciones diferentes de ella y por lo tanto se convierte en abstracta y hereda de BaseResourceConnectorProvider que es la clase base de todos los conectores o clases genéricas. Los métodos abstractos definidos en la clase BaseResourceConnectorProvider son implementados por la clase OracleResourceConnectorProvider que extiende las funcionalidades de DBResourceConnectorProvider ya que corresponde a una implementación de esta última clase. De esta forma se pueden crear todas las clases necesarias para aprovisionar en diferentes Base de Datos con solo heredar de la clase DBResourceConnectorProvider e implementar las funcionalidades declaradas abstractas. De la misma manera ocurre con las demás clases que heredan de la clase BaseResourceConnectorProvider, las cuales son: IMResourceConnectorProvider, MailResourceConnectorProvider y OpenLdapResourceConnectorProvider, las cuales son genéricas para

aprovisionar en servidores de mensajería instantánea, servidores de correo electrónico y servicio de directorios respectivamente y son implementadas cada una por su conector específico correspondiente.
Para implementar el Modelo Provider también es necesario programar aquellas clases que garantizarán su correcto funcionamiento:

- **BaseResourceConfigurationSection.**
  Esta clase hereda de la clase de la plataforma .NET System.Configuration.ConfigurationSection, esta última se utiliza para implementar un tipo de sección personalizado. De esta manera al heredar de esta clase se puede proporcionar control y acceso mediante programación a las secciones de configuración personalizadas.
- **Web.config o App.config.**
  Este es el fichero de configuración donde pueden ser configurados los diferentes proveedores con sus características propias de cada uno. A continuación un fragmento del archivo de configuración:

```
<provisioningConnectors>
       <providers>
       <add name="Oracle11gResourceConnectorProvider"
       type="Gyes.Provisioning.ResourceConnectors.Oracle11gResourceConnectorProvider
       , Gyes.Provisioning.ResourceConnectors"
       connectionStringName="SomeConnectionString"/>
       </providers>
</provisioningConnectors>
```

Fragmento de código 1 Fichero *app.config* del Módulo de Conectores.

- **BaseResourceMainEntryPoint.**
  En esta clase son cargados los datos de los proveedores configurados en el app.config y son verificadas las inicializaciones de estos para asegurar que no haya proveedores nulos o mal configurados.
- **BaseResourceProviderCollection.**
  Esta clase hereda de la clase ProviderCollection y es creada por conveniencia ya que permite indexar para obtener una clase proveedor por su nombre con el cual fue configurado. Esta clase guarda las instancias de los proveedores registrados programáticamente o a través del archivo de configuración.

3.5 Descripción de las Clases Comunes (Common).

Dentro de la implementación del Modelo Provider se encuentran una variedad de clases abstractas que son las encargadas de encapsular todas las funcionalidades que sus subclases deben implementar. Estas clases en el Módulo de Conectores son llamadas Clases Comunes o Conectores Comunes, ya que para

cada Conector Común pueden existir varias implementaciones. Por ejemplo la clase ZimbraResourceConnectorProvider implementa la clase abstracta MailResourceConnectorProvider, siendo esta última una clase genérica que tiene todas las funcionalidades que deben ser desarrolladas, que además puede ser implementada por otro conector para otro servidor de correo, que podría ser por ejemplo MDaemonResourceConnectorProvider.

Dentro de las clases comunes se encuentran los siguientes conectores comunes:

- DBResouceConnectorProvider.
- IMResouceConnectorProvider.
- LdapConnectorProvider.
- MailResourceConnectorProvider.

A continuación se muestra una descripción más detallada de una clase común para varios conectores específicos del Módulo de Conectores.

Tabla 17 Descripción de la clase MailResourceConnectorProvider.

Nombre	MailResourceConnectorProvider	
Descripción	Contiene los atributos y las funciones que deben ser implementados por aquellos conectores específicos para aprovisionar en cualquier base de datos.	
Atributos	**Tipo**	**Descripción**
Name	string	Nombre del conector.
Description	string	No implementado.
ConnectionString	string	string de conexión del OpenLdap.
Métodos	**Descripción**	
CreateAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx);	Override abstract. No implementado.	
BlockAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx);	Override abstract. No implementado.	
ActivateAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx)	Override abstract. No implementado.	
DeleteAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx);	Override abstract. No implementado.	

ModifyAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx);	Override abstract. No implementado.
FindAccount(ResourceConnection resConn, Dictionary<object, object> obj);	Override abstract. No implementado.
AddRoleToUser(ResourceConnection resConn, string user, string role, Transaction trx);	Override abstract. No implementado.
DeleteRoleFromUser(ResourceConnection resConn, string user, string role, Transaction trx);	Override abstract. No implementado.
UserRoles(ResourceConnection resConn, string user);	Override abstract. No implementado.
RoleList(ResourceConnection resConn);	Override abstract. No implementado.

3.6 Descripción de los Conectores Específicos.

Aquellas clases que implementan las funcionalidades de las Clases Comunes son llamadas Conectores Específicos, estas responden a realizar tareas de aprovisionamiento en un recurso único que depende del fabricante y la versión, por ejemplo la clase Oracle11gResourceConnectorProvider que hereda las funcionalidades de la clase DBResourceConnectorProvider es utilizada para aprovisionar en una Base de Datos Oracle 11g.

El grupo de los Conectores Específicos está formado por:

- Oracle11gResourceConnectorProvider.
- OpenfireResourceConnectorProvider.
- OpenLdapResourceConnectorProvider.
- ZimbraResourceConnectorProvider.

A continuación se muestra una descripción más detallada de un Conector Específico del Módulo de Conectores.

Tabla 18 Descripción de la clase OpenLdapResourceConnectorProvider.

Nombre	OpenLdapResourceConnectorProvider
Descripción	Contiene los atributos y las funciones para aprovisionar en un Servicio de Directorio OpenLdap.

Atributos	Tipo	Descripción
Name	string	Nombre del conector.
Description	string	No implementado.
ConnectionString	string	string de conexión del OpenLdap.

Métodos	Descripción
Constructor y Propiedades: Get(), Set()	Constructor y propiedades de todos los atributos Name y ConnectionString.
Initialize(string name, NameValueCollection config)	Inicializar el proveedor con la configuración del fichero *web.config*.
CreateAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx)	Crear cuenta en el servidor OpenLdap recibiendo por parámetros una conexión, un diccionario de objetos y una transacción.
ModifyAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx)	Modificar cuenta en el servidor OpenLdap recibiendo por parámetros una conexión, un diccionario de objetos y una transacción.
DeleteAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx)	Eliminar cuenta en el servidor OpenLdap recibiendo por parámetros una conexión, un diccionario de objetos y una transacción.
BlockAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx)	Bloquear cuenta en el servidor OpenLdap recibiendo por parámetros una conexión, un diccionario de objetos y una transacción.
ActivateAccount(ResourceConnection resConn, Dictionary<object, object> obj, Transaction trx)	Activar cuenta en el servidor OpenLdap recibiendo por parámetros una conexión, un diccionario de objetos y una transacción.
FindAccount(ResourceConnection resConn, Dictionary<object, object> obj)	Buscar los datos de una cuenta en el servidor OpenLdap recibiendo por parámetros una conexión y un diccionario de objetos con los filtros de búsqueda deseados.
FindAllUidAccount(ResourceConnection resConn, Dictionary<object, object> obj)	Buscar los datos de todas las cuenta en el servidor OpenLdap recibiendo por parámetros una conexión y un diccionario de objetos vacío.
FindAttrLdapAccount(ResourceConnection resConn, string uid)	Buscar los datos de una cuenta específicos definidos para luego poder ejecutar una operación “undo”.

AddRoleToUser(ResourceConnection resConn, string user, string role, Transaction trx)	Adicionar un rol a un usuario. No implementado.
DeleteRoleFromUser(ResourceConnection resConn, string user, string role, Transaction trx)	Eliminar un rol a un usuario. No implementado.
UserRoles(ResourceConnection resConn, string user)	Listar los roles con los que cuenta un usuario. No implementado.
RoleList(ResourceConnection resConn)	Listar todos los posibles roles existentes en el servicio de directorio. No implementado.

3.6.1 OpenLdapResourceConnectorProvider.

Un Servicio de Directorio es un conjunto de objetos con atributos organizados en una manera lógica y jerárquica. El ejemplo más común es el directorio telefónico, que consiste en una serie de nombres (personas u organizaciones) que están ordenados alfabéticamente, con cada nombre teniendo una dirección y un número de teléfono adjuntos. También es considerado una base de datos a la que pueden realizarse consultas.

Este conector es el encargado de realizar todas las tareas de aprovisionamiento en un Servicio de Directorio OpenLdap. Son utilizadas las librerías de Novell para Ldap y con estas es posible realizar operaciones sobre las cuentas de usuario en este recurso. El proceso de aprovisionamiento se logra a través de consultas enviadas al servidor. Por ejemplo para crear una cuenta es necesaria una conexión al recurso, un diccionario de objetos con los datos de la cuenta y una transacción asociada. Luego se inicializan los parámetros de ejecución de la consulta así como los parámetros para la consulta inversa. Finalmente se construye el DN y se invoca el método *Execute()* de la clase OpenLdapCommand siendo este el encargado de realizar la ejecución de la creación de la cuenta de manera que en caso que ocurra un fallo pueda ser reversible.

A continuación un fragmento de código utilizado para realizar la creación de una cuenta de usuario en un Servicio de Directorio OpenLdap:

```
public override void CreateAccount(ResourceConnection resConn, Dictionary<object,
object> obj, Transaction trx)
        {
        var oLdapConn = resConn as OpenLdapConnection;
        ValidationUtils.EntryLdapValidation(obj);
        var oLdapCmd = new OpenLdapCommand("create", oLdapConn, trx);
        oLdapCmd.ExecutionParams = new object[] { obj };
        oLdapCmd.Dn = String.Format("uid={0},{1}", obj["uid"],
        oLdapConn.LdapData);
        oLdapCmd.TextValue = "Create";
        oLdapCmd.UndoTextValue = "Delete";
        oLdapCmd.UndoExecutionParams = new object[] { obj["uid"],
        oLdapConn.LdapData };
        oLdapCmd.Dn = String.Format("uid={0},{1}", obj["uid"],
        oLdapConn.LdapData);
        oLdapCmd.Execute();
}
```

Fragmento de código 2 Creación de una cuenta de usuario en un Servicio de Directorio Openldap.

3.6.2 Oracle11gResourceConnectorProvider.

Oracle es un sistema de gestión de base de datos objeto-relacional desarrollado por Oracle Corporation. Cuenta entre otros productos con una variedad de versiones de base de datos de gran rendimiento y calidad mundial.

Este conector es el responsable de aprovisionar cuentas de usuario en una Base de Datos Oracle 11g. Utiliza para su correcto funcionamiento la librería de la plataforma .NET Oracle.DataAccess.Client. Todas las tareas son realizadas a través de consultas SQL enviadas a la base de datos mediante una cuenta con privilegios administrativos para poder ejecutarlas. Por ejemplo, en el momento de realizar una eliminación de cuenta en Oracle 11g lo primero que se necesita es una conexión, un diccionario de objetos con los datos de la cuenta y una transacción asociada. Después son inicializados los parámetros de ejecución de la consulta SQL a realizar en el servidor. Es necesario conocer los datos de la cuenta antes de realizar la eliminación, los cuales son almacenados en un arreglo de objetos, ya que en caso que sea necesario deshacer esta operación son indispensables los valores anteriores de la cuenta para poder regresar al estado anterior a la eliminación. Finalmente es invocado el método *Execute()* el cual se encarga de llevar a cabo el procedimiento.

A continuación se muestra un fragmento de código utilizado para eliminar una cuenta de usuario en Oracle 11g:

```
foreach (object item in obj.Keys)
       {if (ValidationUtils.InvalidChars(obj[item].ToString()))
       throw new Exception("Character not valid");}
var conn = resConn as Oracle11gConnection;
var oraCmd = new Oracle11gCommand("delete", conn, trx);
oraCmd.ExecutionParams = new object[]{obj[USER]};
oraCmd.TextValue = "drop user {0}";
Dictionary<object,object> accountData = AccountData(conn, obj[USER].ToString());
oraCmd.UndoTextValue = "create user {0} identified by {1} default tablespace {2}
temporary tablespace {3}";
oraCmd.UndoExecutionParams = new object[]{
accountData["username"],
accountData["username"],
accountData["default_tablespace"],
accountData["temporary_tablespace"]};
oraCmd.Execute();}
```

Fragmento de código 3 Eliminar cuenta de usuario en una base de datos Oracle 11g.

3.6.3 OpenfireResourceConnectorProvider.

Openfire es un servidor de mensajería instantánea sobre el protocolo XMPP programado en java (multiplataforma) y con un WebApp para administrarlo. Se encuentra bajo licencia GPL y con él se puede administrar a los usuarios, compartir archivos, auditar mensajes, mensajes offline, mensajes broadcast, grupos, y además contiene plugins para ampliar sus funcionalidades.

Este conector para Openfire tiene la tarea de aprovisionar cuentas de usuario en este servidor de mensajería. Este tipo de servidor utiliza para tener el control de sus usuarios y mensajes un Servicio de Directorio y una Base de Datos MySQL. Por lo tanto para aprovisionar cuentas de usuario es necesario realizar consultas al servidor LDAP que el Openfire tenga asociado, debido a esto el aprovisionamiento se realiza de la misma manera que para el Servicio de Directorio OpenLdap, mediante la utilización de la librería de Novell (Novell.Directory.Ldap.dll) son realizadas las creaciones de cuenta y actualizaciones de estas. La Base de Datos MySQL es utilizada para tener el control de los mensajes enviados y recibidos por los usuarios además de los estados en que se encuentran las cuentas, o sea, solo se realizan consultas SQL en el servidor de base de datos para bloquear y desbloquear las cuentas existentes, estas consultas son realizadas utilizando la clase MySqlUtils.

A continuación se muestra un el código utilizado para eliminar una cuenta de usuario donde el nombre de usuario es recibido por parámetros y son almacenados los parámetros de eliminación de cuenta y son consultados los datos de la cuenta de usuario y guardados para en caso que exista un fallo se pueda revertir la eliminación de cuenta quedando como estaba anteriormente. Finalmente es llamado el método *Execute()* al igual que en los demás conectores que es el encargado de continuar el procedimiento.

```
var openFConn = resConn as OpenFireConnection;
ValidationUtils.EntryLdapValidation(obj);
var openFCmd = new OpenFireCommand("delete", openFConn, trx);
openFCmd.ExecutionParams = new object[] { obj["uid"], openFConn.LdapData };
openFCmd.Dn = String.Format("uid={0},{1}", obj["uid"], openFConn.LdapData);
openFCmd.TextValue = "Delete";
Dictionary<object, object> accountAttr = FindAttrLdapAccount(openFConn,
obj["uid"].ToString());
openFCmd.UndoTextValue = "Create";
openFCmd.UndoExecutionParams = new object[] { accountAttr };
openFCmd.Execute();
```

Fragmento de código 4 Bloquear cuenta en un Servicio de Directorio OpenLdap.

3.6.4 ZimbraResourceConnectorProvider.

Zimbra es un cliente/servidor de correo y calendario de código libre, gratuito y descargable, tanto su cliente como el servidor. Soporta acceso POP, acceso IMAP, entre otros; e incluye protección anti-spam y antivirus.

Este conector contiene todas las funcionalidades implementadas para realizar las tareas de aprovisionamiento en un Servidor de Correo Zimbra corriendo sobre la plataforma Linux. Para controlar las cuentas de usuario en un servidor de este tipo es necesario acceder directamente a la consola del servidor (Shell) mediante un usuario con permisos sobre el servidor de correo para ejecutar los comandos que permiten tanto la creación, modificación además de otras funcionalidades. Para poder acceder a la consola del servidor se hace uso de la librería de JCraft Inc. Llamada Tamir.SharpSsh liberada bajo la licencia BSD, esta librería realiza las conexiones con la consola mediante el protocolo ssh.

A continuación se muestra un fragmento de código utilizado para bloquear una cuenta de usuario donde son inicializados los parámetros necesarios para poder ejecutar y deshacer el comando en caso que sea necesario.

```
var openFConn = resConn as OpenFireConnection;
ValidationUtils.EntryLdapValidation(obj);
      if (!Regex.IsMatch(obj["endTime"].ToString(), "^[0-9]*$") ||
      !Regex.IsMatch(obj["startTime"].ToString(), "^[0-9]*$")) {
      throw new Exception("Invalid Character");}

var openFCmd = new OpenFireCommand("Block", openFConn, trx);
openFCmd.ExecutionParams = new object[] { obj };
openFCmd.TextValue = "Block";
openFCmd.UndoExecutionParams = new object[] { obj["uid"] };
openFCmd.UndoTextValue = "Active";
openFCmd.Execute();
```

Fragmento de código 5 Bloquear cuenta en un Servidor Zimbra sobre Linux.

3.7 Validaciones de datos.

Los agujeros de seguridad suelen generarse en la negligencia o la inexperiencia de un programador. Puede haber otras causas ligadas al contexto. Una vulnerabilidad por lo general permite que el atacante pueda engañar a la aplicación, por ejemplo, esquivando los controles de acceso o ejecutando comandos en el sistema donde se aloja la aplicación.

Algunas vulnerabilidades se producen cuando la entrada de un usuario no es controlada, permitiendo la ejecución de comandos o solicitudes SQL (conocidos como Inyección SQL). Otras provienen de errores de un programador en la comprobación de los buffers de datos (que puede ser desbordados), provocando una corrupción de la pila de memoria (y, por tanto, permitiendo la ejecución de código suministrado por el atacante) (Wikipedia, 2011).

EL Módulo de Conectores se protege de este tipo de ataques con la utilización de la clase ValídationUtils, esta última está compuesta por un grupo de métodos los cuales son utilizados por los conectores para validar los datos que son suministrados por otros módulos y de esta manera asegurar que no existen entradas invalidas; además se chequea que no esté en presencia de una posible inyección maliciosa que pueda cambiar la ejecución de un comando ejecutado sobre un recurso.

Entre los métodos más utilizados se encuentra la “sanitación” de cadenas de entrada correspondiente a los parámetros de las consultas. Se aplican filtros basados en expresiones regulares que impidan la entrada de caracteres que invaliden parte de las consultas y permitan concatenar nuevos comandos. Los campos de contraseñas son convertidos a valores string literales, de esta manera se permite que puedan contener caracteres especiales, siendo estos últimos muy utilizados y en algunos casos obligatorios para las contraseñas de usuarios.

3.8 Modelo Transaccional

Una transacción es generalmente definida como una unidad de trabajo que se hace a nombre de una aplicación o componente. Cada transacción puede estar compuesta de múltiples operaciones realizadas en datos que están dispersos en uno o varios procesos o en una o varias máquinas. Cada transacción asegura el trabajo de proteger la integridad del estado de un sistema al proveer cuatro garantías básicas conocidas como las propiedades ACID: atomicidad *(atomicity)*, consistencia *(consistency)*, aislamiento *(isolation)* y durabilidad *(durability)* y que se explican a continuación (Lemoine).

- Atomicidad (Atomicity)

  Una transacción tiene que ser atómica lo que significa que es indivisible; todas las operaciones deben ejecutarse o ninguna en lo absoluto. No debe haber posibilidad de que solo una parte se ejecute.
- Consistencia (Consistency)

  Una transacción mantendrá la consistencia de cualquier recurso que esté asociado a la transacción. Esto es, por ejemplo, si la base de datos se encuentra en un estado consistente antes de ejecutar la transacción, una vez que ésta termine la consistencia de la base de datos deberá conservarse. Por consistente se debe entender, internamente consistente.
- Aislamiento (Isolation)

  La tercera propiedad de una transacción es el aislamiento. Se dice que un conjunto de transacciones está aislado si el efecto del sistema que las ejecuta es el mismo que si ejecutara cada una a la vez; las transacciones se ejecutan en secuencia.
- Durabilidad (Durability)

  Cuando una transacción termina de ejecutarse, todas sus actualizaciones se graban en algún tipo de medio de almacenamiento, típicamente disco, en donde se asegura que las actualizaciones no se perderán. Aún si el sistema operativo falla, los resultados de la transacción son almacenados en disco y podrán ser encontrados ahí cuando se recupere el sistema operativo.

El Módulo de Conectores permite la transaccionalidad de las operaciones ejecutadas sobre los recursos a aprovisionar; para logar su funcionamiento fue utilizado el patrón Command, este es un patrón de diseño en el cual los objetos son usados para representar acciones. Un objeto de comando encapsula una acción y sus parámetros (Carata, 2007). Las transacciones han sido implementadas de tal manera que cumplen con la “Atomicidad”, o sea, o se ejecuta de manera completa una transacción con todos los comandos involucrados o no se ejecuta ningún comando y son deshechos los que hayan sido ejecutados de manera satisfactoria. Estos comandos deben permitir realizar una función de retroceso en caso que ocurra algún error durante la ejecución de una transacción. De esta manera se garantiza que si son realizadas varias tareas de aprovisionamiento simultáneamente y alguna llegara a fallar entonces se daría vuelta atrás a los comandos que fueron ejecutados antes de suceder el fallo, quedando el sistema en el mismo estado que se encontraba antes de realizar la transacción y cumpliendo con la característica “Consistencia”. Las propiedades de “Aislamiento” y “Durabilidad” no son implementadas en el modelo desarrollado ya que además están fuera del alcance del trabajo.

Fueron definidas 6 clases para el diseño del modelo transaccional, estas son mencionadas a continuación:

- Transaction

  Contiene la lógica de programación para hacer y deshacer una transacción que puede contener varios comandos asociados.
- CommandBase

  Esta clase contiene la lógica de programación para ejecutar un comando y para deshacerlo. Define un enlace entre el receptor y la acción de ejecución de comandos.
- ResourceConnection

  Es una clase abstracta la cual contiene funcionalidades para inicializar, abrir y cerrar una conexión asociada a un recurso.
- IUndoable

  Es una clase interfaz que contiene el método Undo para pedir la ejecución del código del receptor y realizar la acción de retroceso.
- TransactionException

  Es una clase que hereda de la clase Exception de la plataforma .NET. Se utiliza para capturar excepciones que puedan ocurrir durante una transacción.
- CommandException

  Es una clase que hereda de la clase Exception de la plataforma .NET. Se utiliza para capturar excepciones que puedan ocurrir durante la ejecución de un comando.

Asociadas a las clases involucradas en una transacción fueron implementadas aquellas que contiene los métodos para iniciar, abrir y cerrar una conexión con el recurso al cual pertenecen (Base de Datos Oracle 11g, Servidor de Mensajería Openfire, Servidor de Correo Zimbra y Servicio de Directorio OpenLdap), estas heredan de la clase ResourceConnection.

- **Oracle11gConnection**
- **OpenfireConnection**
- **ZimbraConnection**
- **OpenLdapConnection**

También fueron implementadas las clases que heredan de CommandBase y por lo tanto definen los métodos para ejecutar y deshacer un comando independientemente del comando que sea.

- **Oracle11gCommand**
- **OpenfireCommand**
- **ZimbraCommand**
- **OpenLdapCommand**

3.9 Integración del Módulo de Conectores con el Subsistema de Aprovisionamiento de Usuarios.

Una vez compilada la librería de clases pertenecientes al Módulo de Conectores esta es referenciada y agregada en la carpeta ***bin*** del servicio de aprovisionamiento que se encuentra dentro del Subsistema de Aprovisionamiento de Usuarios, además son agregados los ***tags*** con las configuraciones de los proveedores en el fichero de configuración ***web.config*** para que puedan ser cargados y utilizados los diferentes proveedores implementados.

Para realizar nuevos conectores a partir de la librería de clases de conectores ya creada es necesario referenciarla en el proyecto donde se vaya implementar el/los nuevos conectores. En el caso que sea necesario crear un conector a partir de los ya existentes, por ejemplo desarrollar un conector para base de datos, se necesita extender las funcionalidades de la clase genérica DBResourceConnectorProvider e implementar todos sus métodos abstractos. Si la necesidad fuera crear un nuevo conector genérico para aprovisionar en un tipo de recurso que no exista definido en la librería de conectores, entonces el procedimiento sería crear una clase la cual debe extender las funcionalidades de la clase BaseResourceConnectorProvider y también se crea la clase que implementaría al conector genérico, por ejemplo, pudiera ser como clase genérica FileSystemResourceConnectorProvider la cual tendría las funcionalidades y atributos para aprovisionar cuentas de usuario en sistemas operativos y como conector específico la clase WindowsXPSP1ResourceConnectorProvider la cual realizaría funciones de aprovisionamiento en una computadora que tenga instalado como sistema operativo Windows XP SP1. Además para permitir que las operaciones de un nuevo conector se ejecuten de manera transaccional es necesaria la implementación de una nueva clase que debe heredar de CommandBase y de esta manera implementar los métodos para ejecutar (*ExecuteCommand()*) y para deshacer un comando (*UndoCommand()*). También es imprescindible la creación e implementación de una clase que herede de ResourceConnection para poder realizar las operaciones relacionadas con la conexión al recurso para el que se esté implementando este nuevo conector, ya que sin la conexión a un recurso es imposible el aprovisionamiento.

Al terminar con el proceso se realiza la compilación del nuevo conector y es agregado nuevamente a la carpeta ***bin*** y adicionada la configuración del nuevo proveedor en el fichero ***web.config***.

3.10 Patrones de diseños utilizados.

Los patrones arquitectónicos representan un problema de diseño recurrente que surge en algunas situaciones específicas y propone un esquema genérico contrastado para su resolución (Falgueras, 2003). A continuación se mencionan los patrones utilizados para la implementación del módulo y como fueron utilizados.

- **Factory Method**

  En el momento de instanciar un proveedor se le solicita a la clase BaseResourceMainEntryPoint que construya un objeto del tipo de conector que se necesita y esta clase devuelve una única instancia del proveedor. A continuación se muestra un ejemplo:

```
BaseResourceConnectorProvider Oracle =
BaseResourceMainEntryPoint.Providers["Oracle11gResourceConnectorProvider"];
```

- **Singleton**

  Con la utilización de este patrón se garantiza que para poder instanciar un proveedor este tiene que estar registrado en el archivo de configuración *web.config* y de esta manera se asegura que existirá una instancia única para cada tipo de proveedor.

- **Command**

  De la clase CommandBase hereden aquellas clases que contienen la lógica de programación para ejecutar un comando en cada uno de los conectores. CommandBase contiene atributos para almacenar tanto el comando que se va a ejecutar como su contra-comando en caso que sea necesario deshacer uno o varios comandos. De esta manera cualquier acción de aprovisionamiento que se vaya a realizar sobre un recurso es convertida en uno o más comandos y de esta manera pueden ser ejecutados dentro de una transacción.

3.11 Pruebas de software

Durante la creación de un producto de software se llevan a cabo disímiles actividades en las que es muy posible la aparición de errores, ya sea en los primeros o posteriores momentos del desarrollo. La significativa participación de las personas en este proceso impone la realización de acciones que contribuyan a la calidad del producto debido a la imposibilidad del humano de comunicarse y trabajar de forma perfecta. Para lograr este objetivo tiene un papel fundamental la aplicación de pruebas para la detección y corrección de errores.

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación (Pressman, 2005). Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Básicamente es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas.

3.11.1 Pruebas Unitarias.

Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. La finalidad de estas pruebas es separar el código en partes y demostrar que estas partes están libres de errores. Las Pruebas Unitarias facilitan que el programador cambie el código para mejorar su estructura, puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores. Estas pruebas permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente.

Visual Studio Team System 2010 proporciona herramientas para crear, editar y mostrar los datos alcanzados por cada una de las pruebas realizadas (Microsoft, 2010), por este motivo fue utilizado para realizar las pruebas unitarias, ya que no es necesaria la instalación de ninguna herramienta extra para su uso. Después de realizada una primera iteración de Pruebas Unitarias fueron encontradas no conformidades en el código los cuales se muestran a continuación:

Tabla 19 Errores detectados en la clase Oracle11ResourceConnectorProvider.

Clase:	**Oracle11ResourceConnectorProvider.**
Área:	Aprovisionar en el recurso Base de Datos Oracle 11g. Gestionar cuenta.
Fallo:	Existía una mala referencia a los atributos usados para crear el string de consulta, quedando esta con valores incorrectos.
Ubicación:	Métodos CreateAccount, ActivateAccount, BlockAccount.
Solución:	Fueron reemplazadas las referencias a los atributos incorrectos, creando la consulta a partir de los datos obtenidos en los parámetros del método donde eran introducidos.
Responsable:	Wilhem Verano.
Prioridad:	Alta

Tabla 20 Errores detectados en la clase OpenLdapConnection.

Clase:	**OpenLdapResourceConnectorProvider.**
Área:	Aprovisionar en el recurso Servicio de Directorio OpenLdap. Gestionar conexión.
Fallo:	A la hora de realizar una consulta al servidor de OpenLdap se hacía una mala referencia a un atributo que no contenía el valor de los datos de conexión del OpenLdap.

Ubicación:	Método OpenConnection.
Solución:	Se creó un nuevo atributo para poder asignar estos parámetros de conexión al objeto del tipo de esta clase, los datos de conexión pueden ser los siguientes: "ou=provtest,dc=proiden,dc=cu".
Responsable:	Wilhem Verano.
Prioridad:	Alta

Tabla 21 Errores detectados en la clase ZimbraResourceConnectorProvider.

Clase:	**ZimbraResourceConnectorProvider.**
Área:	Aprovisionar en el recurso Servidor de correo Zimbra. Gestionar cuenta.
Fallo:	En el momento de modificar la contraseña del usuario no existía el atributo de la clave a ser modificada dentro del comando enviado al servidor.
Ubicación:	Método ModifyAccount.
Solución:	Se sustituyó al comando anterior y se envió otro al servidor encargado de cambiar la contraseña solicitada por el usuario, un ejemplo de esta cadena es la siguiente: “zmprov sp nombreusuario sucontraseñanueva”.
Responsable:	Ernesto Martín.
Prioridad:	Alta

Tabla 22 Errores detectadas en la clase ZimbraResourceConnectorProvider.

Clase:	**ZimbraResourceConnectorProvider.**
Área:	Aprovisionar en el recurso Servidor de correo Zimbra. Gestionar cuenta.
Fallo:	A la hora de capturar la información que mostraba el comando de búsqueda de usuarios en el servidor este devolvía una cadena vacía cuando existían varias cuentas creadas y no se conocía el error ocurrido.
Ubicación:	Método FindAccount.
Solución:	Se sustituyó el fragmento del método con el que se obtenía esa información “(connection as SshExec).RunCommand(command)” por “(connection as SshExec).RunCommand(command, ref stdOut, ref stdError)”, donde stdOut es el mensaje de salida de la consola y stdError es el error que se produce en caso de existir algún fallo.
Responsable:	Ernesto Martín.
Prioridad:	Alta

En la siguiente iteración de Pruebas Unitarias realizadas pudo verificarse que los errores o malos funcionamientos habían sido solucionados. No fueron encontrados errores lógicos en la programación asegurando que ante una entrada de datos determinada los resultados obtenidos fueran los esperados, se determinó que el código escrito estaba libre de errores y que las funcionalidades trabajaban de forma correcta.

A continuación se muestran algunos ejemplos de los resultados obtenidos después de corregidas los errores:

Tabla 23 Descripción de la prueba unitaria *OpenConnectionTest.*

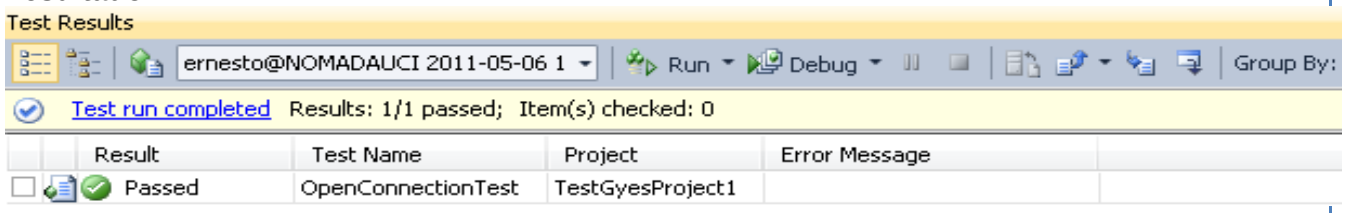
Prueba de Unidad		
Nombre Prueba: OpenConnectionTest	**Recurso:** Servidor de correo Zimbra	
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Última ejecución:**
Ejecutado por: Ernesto Raúl Martín Suárez	**Verificado por:** Roberto Quiñones	
Descripción: **Para poder ejecutar la prueba previamente se deben haber asignado los valores de conexión del recurso solicitado (*host, port, username, password*) si los datos son correctos la conexión se realiza de forma satisfactoria, de lo contrario, se lanza un mensaje mostrando el error que se originó.**		
Entrada:		
Criterio de aceptación: Abrir conexión		
Resultado: 		

Tabla 24 Descripción de la prueba unitaria *CreateAccountTest.*

Prueba de Unidad		
Nombre Prueba: CreateAccountTest	**Recurso:** Servidor de Mensajería OpenFire	
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Última ejecución:**
Ejecutado por: Ernesto Raúl Martín Suárez	**Verificado por:** Roberto Quiñones	
Descripción: **Para la ejecución de la prueba se deben introducir los datos del usuario, si el usuario no existe, la función del método es crear una cuenta de usuario, de lo contrario si el usuario ya se encuentra en el servidor se lanza un mensaje mostrando el error que se originó.**		
Entrada: *Dictionary<object, object>*		

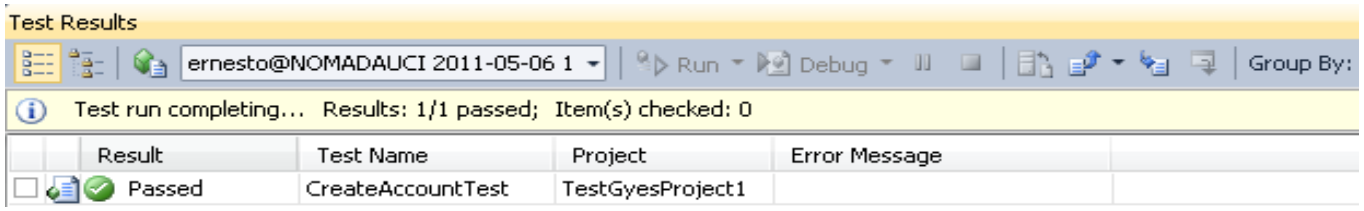
Criterio de aceptación: Crea la cuenta de usuario
Resultado: 

Tabla 25 Descripción de la prueba unitaria *ModifyAccountTest.*

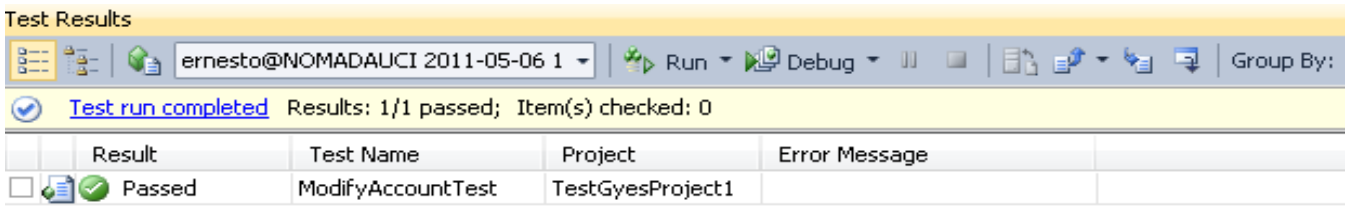
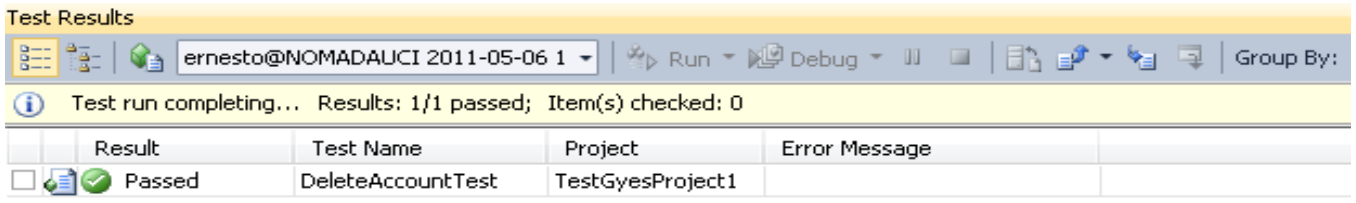
Prueba de Unidad		
Nombre Prueba: ModifyAccountTest	**Recurso:** Directorio de identidades OpenLdap	
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Última ejecución:**
Ejecutado por: Wilhem Verano	**Verificado por:** Roberto Quiñones	
Descripción: **Para la ejecución de la prueba se deben introducir los datos del usuario, si el usuario existe, se procede modificar los datos, de lo contrario si el usuario no se encuentra en el servidor se lanza un mensaje mostrando el error que se originó.**		
Entrada: *Dictionary<object, object>*		
Criterio de aceptación: Modifica la cuenta de usuario		
Resultado: 		

Tabla 26 Descripción de la prueba unitaria *DeleteAccountTest.*

Prueba de Unidad		
Nombre Prueba: DeleteAccountTest	**Recurso:** Servidor de Base de datos Oracle 11g	
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Última ejecución:**
Ejecutado por: Wilhem Verano	**Verificado por:** Roberto Quiñones	
Descripción: **Para la ejecución de la prueba se deben introducir el identificador del usuario, se hace una búsqueda en el servidor y si el usuario existe, se elimina la cuenta del servidor, de lo contrario si el usuario no se encuentra en el servidor se lanza un mensaje mostrando el error que se originó.**		
Entrada: *Dictionary<object, object>*		

Criterio de aceptación: Elimina la cuenta de usuario

Resultado:



3.12 Conclusiones

En este capítulo se realizó la construcción de la solución propuesta del Módulo de Conectores a través de las herramientas y tecnologías seleccionadas siguiendo el diagrama de clases diseñado. Fueron desarrollados los Conectores para aprovisionar en cuatro tipos de recursos distintos. Mediante el uso de patrones se posibilitó que el Módulo tenga la capacidad de ser extensible. Fueron chequeadas todas las funcionalidades del Módulo desarrollado para comprobar su correcto funcionamiento a través de pruebas unitarias realizadas.

CONCLUSIONES GENERALES

AL concluir el desarrollo del trabajo se arribaron a las siguientes conclusiones:

- El estudio de diferentes Sistemas de Aprovisionamiento de Usuarios permitió llegar a un mejor entendimiento del proceso.
- Durante la etapa de Planificación la lista de los escenarios permitió conocer las funcionalidades con las que debía cumplir el Módulo de Conectores.
- Mediante el plan de iteraciones se estimó el tiempo de desarrollo que tomaría cada iteración y el módulo en general.
- La especificación de los escenarios ayudó a realizar la implementación de las funcionalidades del Módulo de Conectores.
- La utilización de algunos patrones arquitectónicos permitió que se pudieran extender las funcionales y continuar desarrollando nuevos conectores además de permitir realizar las funciones del módulo de manera transaccional.
- Las pruebas unitarias realizadas permitieron validar cada uno de los escenarios y verificar su correcto funcionamiento, estas arrojaron resultados satisfactorios otorgando la evaluación final del producto.

RECOMENDACIONES

Debido a la posible extensión del Módulo de Conectores se recomienda:

- Desarrollar nuevos conectores basados en el modelo ya implantado.
- Ampliar el modelo de transacciones.
- Integrar con el servicio de aprovisionamiento.

BIBLIOGRAFÍA

Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal and Timothy Wood. 2008. *Agile dynamic provisioning of multi-tier Internet applications.* 2008.

Carata, Francesco. 2007. *CodeProject.* [Online] 8 23, 2007. [Cited: 6 5, 2011.] http://www.codeproject.com/KB/architecture/Command_Pattern.aspx.

Crispin, Mark R. 1996. [Online] 12 1996. http://www.faqs.org/rfcs/rfc2060.html.

dofactory. 2001. dofactory. [Online] 2001. [Cited: 6 5, 2011.] http://www.dofactory.com/Patterns/PatternCommand.aspx.

Falgueras, Campderrich. 2003. *Ingeniería del software.* 2003.

G. Myers, John and T. Rose, Marshall. 1996. FAQS.ORG. [Online] 5 1996. http://www.faqs.org/rfcs/rfc1939.html.

Guessant, Mickaël. [Online] http://mguessan.free.fr/nt/openldap_en.html.

Howard, Rob. 2004. msdn. [Online] 3 2, 2004. http://msdn.microsoft.com/en-us/library/ms972319.aspx.

IBM. 2009. [Online] 2009. http://www.ibm.com/developerworks/wikis/display/tivoliprovisioningmanager/Tivoli+Provisioning+Manager+7.x+Guide+%28FAQ%29+-+IBM+information.

—. 2009. [Online] 2009. http://publib.boulder.ibm.com/infocenter/tivihelp/v13r1/index.jsp?topic=/com.ibm.tivoli.tpm.ovw.doc/overview/covw_architect.html.

—. 2009. [Online] 2009. http://www-01.ibm.com/software/tivoli/products/prov-mgr/.

—. 2009. [Online] 2009. [Cited: 01 22, 2011.] http://www-01.ibm.com/software/tivoli/products/identity-mgr/.

IETF. 2006. IETF TOOLS. [Online] 2006. [Cited: 5 31, 2011.] http://tools.ietf.org/html/rfc4511.

ITU. 2008. International Telecomunication Union. [Online] 11 2008. [Cited: 5 31, 2011.] http://www.itu.int/itu-t/recommendations/index.aspx?ser=X.

Jackson. 2006. [Online] 2006. http://es.scribd.com/doc/12983329/Metodologia-de-Desarrollo-de-Software.

Lasater, Chris. 2007. codeproject. [Online] 2007. http://www.codeproject.com/KB/books/DesignPatterns.aspx.

Lemoine, Francisco Javier Baños. Sitio de Francisco Javier Baños Lemoine. [Online] http://dixi.members.winisp.net/escalabilidad/I.14.Transacciones.htm.

Microsoft. 2008. [Online] 2008. http://technet.microsoft.com/en-us/library/cc526764.aspx.

—. 2008. [Online] 2008. [Cited: 01 10, 2011.] http://technet.microsoft.com/es-es/library/cc775788%28WS.10%29.aspx.

—. 2009. [Online] 2009. http://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx.

—. 2011. [Online] 2011. [Cited: 01 18, 2011.] http://www.microsoft.com/spain/visualstudio/products/2010-editions/ultimate.

—. 2005. [Online] Octubre 2005. http://msdn.microsoft.com/en-us/library/aa479030.aspx.

—. 2010. [Online] 2010. http://msdn.microsoft.com/es-es/library/ms182520%28v=vs.80%29.aspx.

—. 2006. [Online] 2006. http://msdn.microsoft.com/en-us/vstudio/aa718801.aspx.

—. 2006. [Online] 2006. http://msdn.microsoft.com/en-us/vstudio/aa718801.aspx.

—. 2010. MSDN. *MSDN.* [Online] 2010. [Cited: 05 05, 2011.] http://msdn.microsoft.com/en-us/library/33864ckt.aspx.

Oracle. 2009. [Online] 2009. http://www.oracle.com/technetwork/middleware/id-mgmt/overview/index.html.

Postel, Jonathan B. 1982. [Online] 8 1982. http://www.faqs.org/rfcs/rfc821.html.

Pressman, Roger. 2005. *Ingeniería de software Un enfoque práctico.* 2005.

Rodríguez, Eric. 2010. [Online] 11 28, 2010. [Cited: 3 25, 2011.] http://www.slideboom.com/presentations/291754/EL-INTERNET-Y-SUS-BENEFICIOS.

Saint-Andre, Peter. 2011. [Online] 3 2011. [Cited: 6 6, 2011.] http://xmpp.org/rfcs/rfc6121.html.

Schackow, Stefan. 2006. *ASP.NET 2.0 Security,Membership, and Role Management.* s.l. : Wiley Publishing, Inc., Indianapolis, Indiana, 2006.

—. 2006. *Professional ASP.NET 2.0 Security, Membership, and.* Indiana,EUA : Wiley Publishing, Inc., 2006.

Sierra, Francisco Javier Ceballos. 2006. *El lenguaje de programación C#.* 2006.

Wesley, Adisson. 2003. *Design Patterns. Elements of Reusable Object-Oriented.* 2003.

—. 2003. *Design Patterns. Elements of Reusable Object-Oriented.* 2003.

Wikipedia. 2011. Wikipedia. [Online] 18 3, 2011. [Cited: 6 5, 2011.] http://es.wikipedia.org/wiki/Agujero_de_seguridad.

ANEXOS

Anexo 1: Lista de los escenarios.

Aprovisionar en el recurso Base de Datos Oracle 11g	
1	**Gestionar conexión.**
	1.1 Abrir conexión.
	1.2 Cerrar conexión.
2	**Gestionar Cuentas de Usuario**
	2.1 Crear cuenta.
	2.2 Modificar cuenta.
	2.3 Eliminar cuenta
	2.4 Suspender cuenta
	2.5 Bloquear cuenta
	2.6 Activar cuenta
3	**Gestionar Roles**
	3.1 Adicionar rol
	3.2 Eliminar rol
	3.3 Buscar roles
	3.4 Listar roles

Aprovisionar en el recurso Servidor de Correo Zimbra	
1	**Gestionar conexión.**
	1.1 Abrir conexión.
	1.2 Cerrar conexión.
2	**Gestionar Cuentas de Usuario**
	2.1 Crear cuenta.
	2.2 Modificar cuenta.
	2.3 Eliminar cuenta
	2.4 Suspender cuenta
	2.5 Bloquear cuenta
	2.6 Activar cuenta
3	**Gestionar Roles**
	3.1 Adicionar rol
	3.2 Eliminar rol
	3.3 Buscar roles
	3.4 Listar roles

Aprovisionar en el recurso Servicio de Directorio Openldap	
1	**Gestionar conexión.**
	1.1 Abrir conexión.
	1.2 Cerrar conexión.
2	**Gestionar Cuentas de Usuario**
	2.1 Crear cuenta.
	2.2 Modificar cuenta.
	2.3 Eliminar cuenta
	2.4 Suspender cuenta
	2.5 Bloquear cuenta
	2.6 Activar cuenta

Aprovisionar en el recurso Servidor de Mensajería Instantánea Openfire	
1	**Gestionar conexión.**
	1.1 Abrir conexión.
	1.2 Cerrar conexión.
2	**Gestionar Cuentas de Usuario**
	2.1 Crear cuenta.
	2.2 Modificar cuenta.
	2.3 Eliminar cuenta
	2.4 Suspender cuenta
	2.5 Bloquear cuenta
	2.6 Activar cuenta

Anexo 2: Descripción de los escenarios.

Descripción de Escenarios

Sistema de Administración de Identidades

Subsistema Aprovisionamiento

Hoja de Revisión

Elaborado por:

Fecha	Autor	Versión	Descripción
17/04/2011	Wilhem M. Verano Escalona	1.0	Especificación de escenarios
19/04/2011	Ernesto Raúl Martín Suárez	1.0	Especificación de escenarios

Revisado por:

Equipo	Versión	Fecha
Roberto Quiñones Bondartchuk	1.0	23/04/2011

Descripción de Escenarios

Para la implementación del sistema se realiza la descripción de los escenarios según las necesidades del cliente.

Recurso Servidor de Correo Zimbra.

Título:	**Abrir Conexión.**
Área:	Aprovisionar en el recurso Servidor de Correo Zimbra. Gestionar conexión.
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción.

Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando se selecciona un recurso en el cual se va a aprovisionar, los parámetros de conexión son cargados y utilizados para establecer la comunicación con dicho recurso. En caso de fallar la conexión se notifica el error.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	AcZ1
Orden de magnitud:	1

Título:	**Crear cuenta.**
Área:	Aprovisionar en el recurso Servidor de Correo Zimbra.. Gestionar cuenta.
Iteración:	1
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Crear Cuenta” y en cual recurso va a aprovisionar. Después se establece la conexión, se validan los datos de la cuenta y en caso de no ocurrir ningún error se procede a la creación de la cuenta enviando comandos a la consola del servidor con privilegios de administración en el servidor del Zimbra. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión con el recurso.
Historial:	Inicio de descripción del escenario.
Prioridad:	Alta
Identificador:	CcZ1
Orden de magnitud:	1

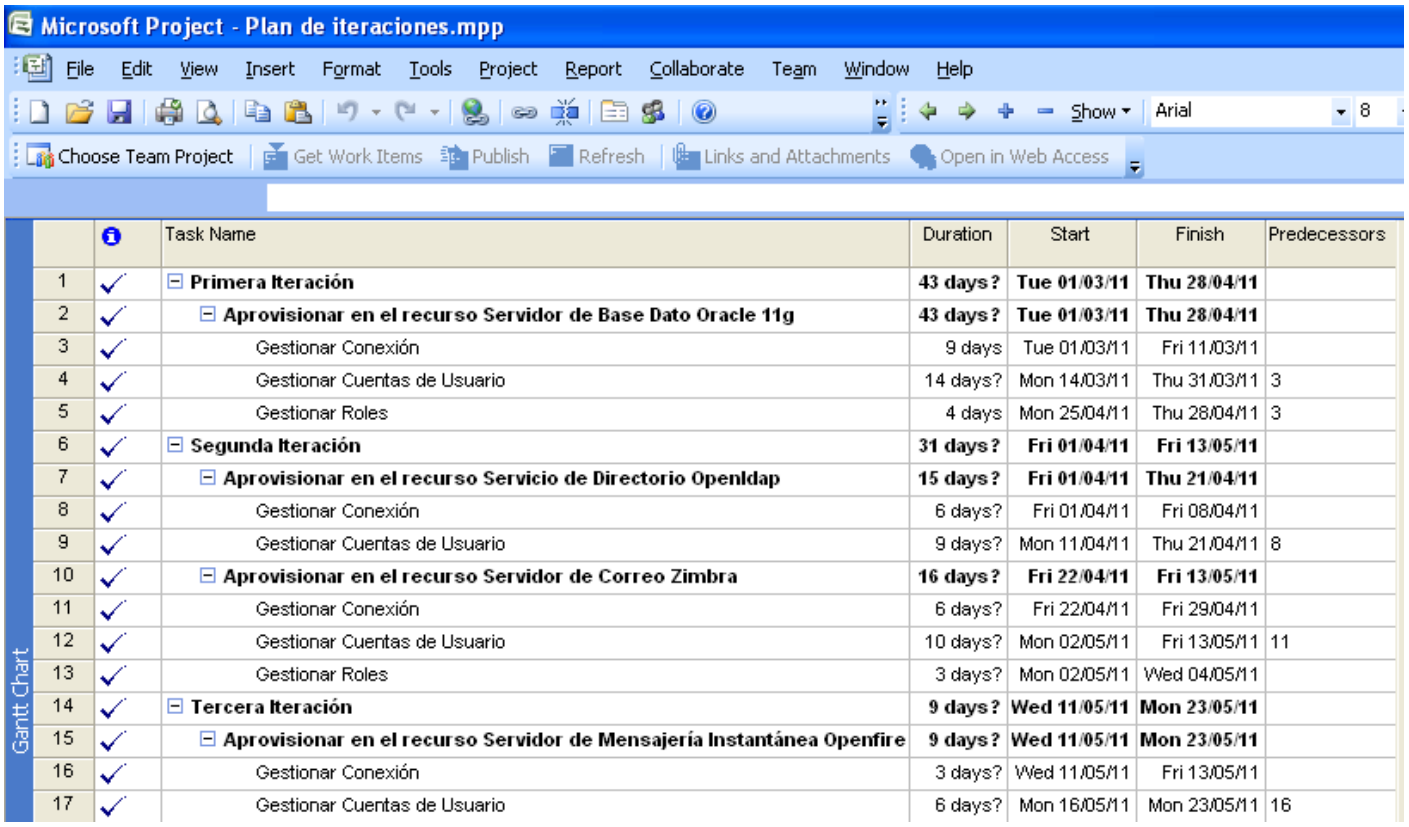
Título:	**Eliminar cuenta.**
Área:	Aprovisionar en el recurso Servidor de Correo Zimbra. Gestionar cuenta.
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Eliminar Cuenta” y el recurso específico en el cual va a realizar esta acción. Se establece la conexión con el recurso, se envían comandos específicos a la consola del servidor con privilegios de administración y se procede a la eliminación de la cuenta seleccionada. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión con el recurso.
Historial:	Inicio de descripción del escenario.
Prioridad:	Media
Identificador:	EcZ2
Orden de magnitud:	2

Título:	**Modificar cuenta.**
Área:	Aprovisionar en el recurso Servidor de Correo Zimbra. Gestionar cuenta.
Iteración:	2
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Modificar Cuenta” y elige la cuenta que será objeto de cambios en un recurso específico. Se establece la conexión con el recurso, se validan los datos y en caso de no existir ningún error se procede a la modificación de la cuenta a través de comandos enviados hacia la consola del servidor. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión con el recurso.

Historial:	Inicio de descripción del escenario.
Prioridad:	Media
Identificador:	McZ2
Orden de magnitud:	2

Título:	**Desactivar cuenta.**
Área:	Aprovisionar en el recurso Servidor de Correo Zimbra. Gestionar cuenta.
Iteración:	3
Estado:	Activo
Razón de estado:	Proceso de construcción.
Responsable:	Wilhem Verano, Ernesto Martín.
Descripción:	El escenario comienza cuando el Administrador selecciona en la Interfaz Web la opción “Desactivar Cuenta” y elige la cuenta a desactivar en un recurso específico. Se establece la conexión con el recurso y se envían comandos a través de la consola del servidor para realizar la desactivación de la cuenta en el servidor. En caso de ocurrir algún error durante el procedimiento se notifica. Al finalizar se cierra la conexión con el recurso.
Historial:	Inicio de descripción del escenario.
Prioridad:	Baja
Identificador:	DcZ3
Orden de magnitud:	3

Anexo 3: Plan de Iteraciones.



	Task Name	Duration	Start	Finish	Predecessors
1	**Primera Iteración**	**43 days?**	**Tue 01/03/11**	**Thu 28/04/11**	
2	**Aprovisionar en el recurso Servidor de Base Dato Oracle 11g**	**43 days?**	**Tue 01/03/11**	**Thu 28/04/11**	
3	Gestionar Conexión	9 days	Tue 01/03/11	Fri 11/03/11	
4	Gestionar Cuentas de Usuario	14 days?	Mon 14/03/11	Thu 31/03/11	3
5	Gestionar Roles	4 days	Mon 25/04/11	Thu 28/04/11	3
6	**Segunda Iteración**	**31 days?**	**Fri 01/04/11**	**Fri 13/05/11**	
7	**Aprovisionar en el recurso Servicio de Directorio Openldap**	**15 days?**	**Fri 01/04/11**	**Thu 21/04/11**	
8	Gestionar Conexión	6 days?	Fri 01/04/11	Fri 08/04/11	
9	Gestionar Cuentas de Usuario	9 days?	Mon 11/04/11	Thu 21/04/11	8
10	**Aprovisionar en el recurso Servidor de Correo Zimbra**	**16 days?**	**Fri 22/04/11**	**Fri 13/05/11**	
11	Gestionar Conexión	6 days?	Fri 22/04/11	Fri 29/04/11	
12	Gestionar Cuentas de Usuario	10 days?	Mon 02/05/11	Fri 13/05/11	11
13	Gestionar Roles	3 days?	Mon 02/05/11	Wed 04/05/11	
14	**Tercera Iteración**	**9 days?**	**Wed 11/05/11**	**Mon 23/05/11**	
15	**Aprovisionar en el recurso Servidor de Mensajería Instantánea Openfire**	**9 days?**	**Wed 11/05/11**	**Mon 23/05/11**	
16	Gestionar Conexión	3 days?	Wed 11/05/11	Fri 13/05/11	
17	Gestionar Cuentas de Usuario	6 days?	Mon 16/05/11	Mon 23/05/11	16

Anexo 4: Requerimientos de calidad de servicio.

Requerimientos de Calidad de Servicios	
Campo	Descripción
Título	Requerimiento de Diseño e Implementación
Area	Autorización
Iteración	1
Tipo	Diseño e Implementación
Responsable	Ernesto Raúl Martín Suárez, Wilhem Manuel Verano
Descripción	El módulo debe contar con un conjunto de patrones de diseño que permitan la reutilización el código así como una librería de clases (API) para la futura extensión del módulo. El código debe cumplir con los estándares de codificación establecidos por el cliente. Se hará uso del Estilo de codificación CAMEL. Este emplea una mezcla de mayúsculas y minúsculas concatenando varias palabras para simplificar la nomenclatura. Los nombres de clases, espacios de nombres, métodos y propiedades tendrá como característica que la primera letra de cada palabra comenzará con mayúscula. Ej. ZimbraResourceConnectorProvider, LdapResourceConnectorProvider Los atributos de las clases y los parámetros de los métodos mostrará la primera letra con minúscula y para cada primera letra de las demás palabras se empleará la mayúscula. Ej. zimbraHost, zimbraUsername, openLdapHost Las constantes se escribirán completamente con mayúscula. Ej. const string USERPASSATTR Las interfaces y los enumerativos deberán contener en el inicio de la palabra una I en el caso de las interfaces y una E en el caso de los enumerativos. Ej. ILdapResourceConnectorProvider Todos los nombres y documentación debe ser en inglés y deben describir claramente su finalidad. Ej. ZimbraResourceConnectorProvider Como el lenguaje de programación que se usa es C#, se empleara la función de documentación XML.
Prioridad	Alta
Orden de magnitud	1

Requerimientos de Calidad de Servicios	
Campo	Descripción
Título	Requerimiento de Seguridad
Area	Autorización
Iteración	1
Tipo	Seguridad
Responsable	Ernesto Raúl Martín Suárez, Wilhem Manuel Verano
Descripción	La seguridad en el módulo estará implementada por protociolos seguros SSL/TSL que soporte el tipo de recurso especifico con el que se va a realizar la conexion para garantizar que esta conexión y el envio de datos se realice de forma saticfactoria.
Prioridad	Alta
Orden de magnitud	1

GLOSARIO DE TÉRMINOS Y SIGLAS

A

AJAX: Técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

B

Base de datos: Conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo.

C

Common Language Runtime (CLR): Es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET. El CLR es el encargado de compilar una forma de código intermedio llamada Common Intermediate Language, al código de maquina nativo, mediante un compilador en tiempo de ejecución.

Cronometrar: Cronometrar el desarrollo de un software es tener el control del tiempo de vida de este.

D

DLL: Dynamic Link Library ("Biblioteca de vínculos dinámicos") es un archivo que contiene funciones que se pueden llamar desde aplicaciones u otras DLL. Los desarrolladores utilizan las DLL para poder reciclar el código y aislar las diferentes tareas.

F

Framework: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

I

IBM: Empresa multinacional estadounidense que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. Tiene su principal presencia en prácticamente todos los segmentos relacionados con las tecnologías de la información; de hecho, en los años recientes, más de la mitad de sus ingresos vienen de sus ramas de consultoría y servicios, y no de la fabricación de equipos.

IDE: Software informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

L

LDAP: Protocolo Ligero de Acceso a Directorios es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

Licencia GPL: La licencia GPL o General Public License es desarrollada por la FSF o Free Software Foundation. Permite instalar y usar un programa GPL en un ordenador o en tantos como se quiera, sin limitación. También puedes modificar el programa para adaptarlo a lo que uno desee que haga. Además, se puede distribuir el programa GPL tal cual o después de haberlo modificado.

Licencia BSD: Es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Es una licencia de software libre permisiva. Permite el uso del código fuente en software no libre.

M

Microsoft Corporation: Empresa multinacional de origen estadounidense. Dedicada al sector de la informática. Microsoft desarrolla, fabrica, licencia y produce software y equipos electrónicos.

MSF: Microsoft Solutions Framework (MSF) es un conjunto de principios, modelos, disciplinas, conceptos y directrices para la entrega de soluciones de tecnología de la información de Microsoft. MSF no se limita

solo al desarrollo de aplicaciones, si no también se aplican a otros proyectos de las tecnologías de la información como los proyectos de implementación, la creación de redes o infraestructura.

MVC: Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que divide en tres componentes distintos los datos de una aplicación, la interfaz de usuario y la lógica de control. El patrón de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el encargado de la Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

MPS: Plataforma extensible establecida en Windows que proporciona un marco basado en XML el cual permite crear soluciones personalizadas de aprovisionamiento para la Web, datos y alojamiento de aplicaciones. Esta solución incluye tareas tales como agregar nuevos usuarios, la actualización de las entradas de la guía, y aprovisionamiento de aplicaciones y servicios.

O

Openfire: Sistema de mensajería instantánea GPL y hecho en java, utiliza el protocolo XMPP con el podrás tener tu propio servidor de mensajería donde puedes administrar a tus usuarios, compartir archivos, auditar mensajes, mensajes offline, mensajes broadcast, grupos y además contiene plugins gratuitos con diferentes funciones extras.

P

Plugins: Conjunto de componentes de software que pueden ser adicionados a una aplicación para agregar funciones específicas.

S

SQL: Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar información de interés de una base de datos de una forma sencilla, así como también hacer cambios sobre ella.

Ssh: este intérprete de órdenes es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host

remotamente. A diferencia de otros protocolos de comunicación remota este encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptados.

X

XMPP: Protocolo extensible de mensajería y comunicación de presencia, es un protocolo abierto y extensible basado en XML, originalmente ideado para mensajería instantánea. Con el protocolo XMPP queda establecida una plataforma para el intercambio de datos XML que puede ser usada en aplicaciones de mensajería instantánea. Las características en cuanto a adaptabilidad y sencillez del XML son heredadas de este modo por el protocolo XMPP.

Z

ZCS: Zimbra Collaboration Suite es un software de correo electrónico que permite a los empleados de oficina enviar, recibir, guardar y buscar los miles de mensajes procesados cada día. Posee tanto el componente de servidor como su respectivo cliente. Hace uso de tecnologías Open Source existentes tales como Postfix, MySQL y OpenLdap.