Universidad de las Ciencias Informáticas

Facultad#1

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

Paquete de conectores para el sistema de Aprovisionamiento de Usuarios de la Universidad de las Ciencias Informáticas

Autores:

Marisol Viera Figueroa

Alfredo Gonzáles Fraxedas

Tutor:

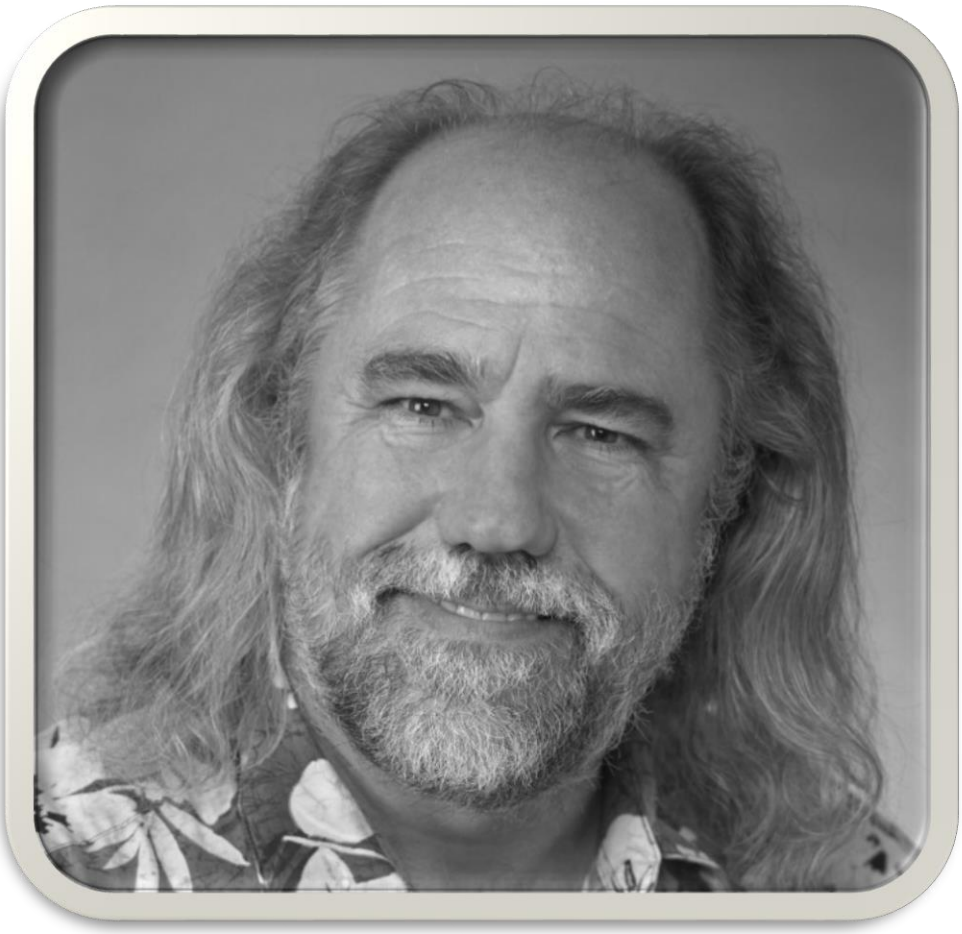
Ing. Yendry Machado García

Ing. Vladimir Campos Kindelan

La Habana, 3 de junio de 2015

Año 56 de la Revolución

Frase



"La función de un buen *software* es hacer que lo complejo aparente ser simple"

Grady Booch

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Autorizamos a dicho centro para que haga el uso que estime pertinente de este trabajo.

Para que así conste firmamos la presente a los ___ días del mes de _____del año_____.

Autores:

Marisol Viera Figueroa _____

Alfredo Gonzáles Fraxedas _____

Ing. Vladimir Campos Kindelan

_____

Tutor

Ing. Yendry Machado García

_____

Tutor

Agradecimientos

A mi madre por su amor incondicional, por tantos sacrificios, por ser un ejemplo para mí, por tanta devoción, dedicación, porque después de tantos sufrimientos y tantas lágrimas va a ver su sueño realizado, sus valores inculcados y por la confianza que siempre ha tenido en mí.

A mi padre que ha sido ejemplo de sacrificio, fuerza y batallar. Por confiar en mí, por no rendirse nunca y salir hacia adelante, por sus consejos. Porque sin él darse cuenta me demostró que soy una persona más fuerte de lo que yo me imaginaba que era. Por demostrarme que siempre se puede y por quererme tanto.

A mi novio Yubenis por entenderme mejor que nadie, por apoyarme y comprenderme en los momentos de estrés cuando yo misma ni me soportaba y darme fuerzas para seguir adelante. Gracias por estar ahí a mi lado, por darme consuelo y ayudarme cuando más lo necesité. Gracias por tu apoyo incondicional.

A mi hermanita Lilian por ser muy importante y especial para mí. A toda mi familia, por su comprensión y apoyo incondicional durante todos estos años.

A los que creyeron en mí, que lo podía lograr, y a los que no también, porque su desconfianza fue la fuerza que me empujó a seguir adelante. A mis amigos, a todos los de mi primer grupo 1008, a los de mi grupo actual 1502, a los que hice en el camino y a los que en algún momento me dijeron, "Hola, ¿Cómo va la tesis?". A las chicas del apartamento Claudia, Ively, Liliana, por apoyarme y ayudarme de una forma u otra cuando lo he necesitado. A mis amigas Orlay, Arletis, Lien, Yennis y Lisdania por estar ahí y nunca olvidarse que existo.

A mi compañero de tesis por aguantarme todo este año y trabajar cada día para que la tesis saliera adelante. A mis tutores por brindarme su ayuda, conocimientos y apoyo, pues fueron vitales para la realización de esta investigación.

Ah y a mí, por haber luchado aún sin fuerzas, haber vencido todas las barreras y obstáculos en la vida y lograr atrapar el sueño que por momentos se me iba de las manos.

Marisol

A mis padres, por toda su confianza, por haberme apoyado en todo momento a pesar de las circunstancias, daré lo mejor de mí para compensar todo lo que me han dado.

A mi novia, por ser mi fiel compañera en las buenas y malas, por confiar en mí en todo momento.

A mi familia, por todo el cariño, ayuda y dedicación que me han brindado sin esperar nada a cambio.

A mis compañeros de aula el piquete del 1501, por todo el apoyo y la colaboración que me han brindado.

A mis amigos del apartamento, del edificio, en fin a todos los que he conocido en la UCI, gracias a los que están y a los que conocí que no están en estos momentos.

A los tutores, por todo el apoyo brindado en la realización de esta investigación.

A mi compañera de tesis, por su paciencia y por confiar en mí.

Alfredo

Dedicatoria

A la memoria de mi abuela Cachín, porque a pesar de no estar físicamente conmigo, sé que estaría muy orgullosa de mí.

A 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.

En especial a mi hermanita Lilian, de la cual espero siga mi ejemplo y se convierta en una profesional.

A mi querido novio que tanto me ha apoyado en estos seis años y me ha brindado todo su amor.

Marisol

A lo más sagrado de mi vida, mis padres, por ser los motores que me impulsan y me dan fuerzas para salir adelante.

A mis amigos, por estar presentes en cada momento feliz y triste de mi vida.

Alfredo

Resumen

La presente investigación tiene como objetivo desarrollar un paquete de conectores que permita al Sistema de Aprovisionamiento de Usuarios la interacción con los servidores de aplicaciones y servicios, con la intención de dar solución a los problemas existentes que giran en torno a cómo establecer la interacción con servidores de aplicaciones y servicios que utiliza el Sistema de Aprovisionamiento de Usuarios de la Universidad de las Ciencias Informáticas (UCI) para realizar las tareas de aprovisionamiento.

En el sistema existen insuficiencias de prestaciones debido a que no se puede acceder a todos los servidores que se utilizan en la UCI. Además no cuenta con un mecanismo que le permita deshacer acciones para recuperarse ante la ocurrencia de errores. Para cumplir el objetivo general se realizó un estudio de la interacción entre servidores de aplicaciones y servicios en el proceso de aprovisionamiento de usuarios, con el fin de elaborar la propuesta de solución.

El paquete de conectores permite la conexión a diferentes recursos específicos, brinda la posibilidad de incluir nuevos conectores para nuevos recursos sin realizar cambios en la interfaz principal y permitir al sistema que utilice el paquete de conectores deshacer acciones en caso de ocurrencia de errores. Para la elaboración de la propuesta de solución se utilizó el lenguaje de programación Python y el entorno de desarrollo integrado PyCharm. Para guiar el desarrollo del *software* se utilizó la Programación Extrema (XP). Se realizaron las pruebas unitarias y de aceptación para verificar el correcto funcionamiento del paquete de conectores.

Palabras claves: conectores, interacción, servidores.

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

El desarrollo de la Informática y los vertiginosos cambios producidos en el ámbito de las tecnologías de la información y las comunicaciones (TIC) ha cambiado la forma en que se realiza la gestión de la información. Las organizaciones y empresas de hoy en día han realizado un uso masivo de las nuevas tecnologías, automatizando sus procesos pasando así del contexto físico al digital (1).

Una de las variables fundamentales en la organización empresarial es el flujo informativo y la disponibilidad de los datos a través de los sistemas informáticos en el momento apropiado. De igual forma surge la necesidad de integrar la información, para facilitar el nivel de gestión del flujo informativo que se genera en las empresas y así poder planificar y controlar mejor sus procesos (2).

Las grandes organizaciones de hoy en día suelen tener procesos complejos con diseños poco satisfactorios para aprovisionar usuarios a los sistemas de información. En ocasiones estos sistemas emplean un tiempo prolongado para tener acceso a aquellas aplicaciones que necesitan para realizar su trabajo. Además del conjunto de actividades que realizan de forma manual afines con el aprovisionamiento de usuarios, agregan gastos generales y retrasan la productividad de los trabajadores (3).

En los últimos años Cuba, debido al recio bloqueo al cual se encuentra sometida hace más de 50 años por el gobierno de los Estados Unidos de América, tiene una infraestructura débil para soportar la informatización de los procesos en las empresas. Esto se ha visto reflejado en lo dispuesto por el gobierno norteamericano a través de la ley Torricelli respecto al acceso a las TIC. De igual forma se restringe el acceso a importantes sitios exclusivos sobre tecnologías de *software* que impiden a los cubanos estar en contacto directo con tan importante campo de la ciencia (4). Precisamente por esta causa, la economía nacional ha tenido que adaptarse a la fluctuante economía mundial, buscando variantes y oportunidades de negocios que permitan crear en el país una plataforma de bienes y servicios asociados a la industria del *software*, como una de las principales fuentes de ingresos para el país.

El Comandante en Jefe, uno de los principales estadistas del siglo XX, fue el creador intelectual de la Universidad de las Ciencias Informáticas (UCI). Fue creada con el compromiso para con la Revolución de desarrollar sistemas informáticos que automaticen procesos y permitan el desarrollo en el campo de la informática y las telecomunicaciones, tanto a nivel nacional como internacional. Con la visión estratégica que lo caracteriza, Fidel Castro Ruz planteó: “Esta Universidad debe caracterizarse por la gran variedad de

formas diferentes de enseñar, de preparar. Debe tener un nivel de flexibilidad alto. centro docente experimental, centro docente–productor" (5).

Dentro de los entornos productivos de la UCI se encuentra el Centro de Identificación y Seguridad Digital (CISED), el cual está encargado del desarrollo de aplicaciones destinadas a la gestión de información en diversas ramas como la biometría, tarjetas inteligentes, criptografía y por último la administración de entidades, de la cual se encarga el departamento de desarrollo. Uno de los sistemas que se desarrolla en este departamento es el Sistema de Aprovisionamiento de Usuarios, el cual permite que se realice de manera centralizada el trabajo con cuentas de usuarios. Proporciona entre otras funciones la conexión a recursos tales como: directorios de identidades, servidores de mensajería instantánea y servidores de correo.

Este sistema cuenta con recursos de lectura/escritura que facilitan la interacción con la información que brindan otros servidores, pero aun así, cuenta con deficiencias. En el proceso de aprovisionamiento existe inconsistencia de la información, por tal motivo si durante el proceso de creación de cuentas de un usuario algún servidor falla, el proceso estará incompleto y no se podrán deshacer los cambios realizados en los restantes servidores. El sistema de aprovisionamiento utiliza los conectores de manera estática, puesto que en cualquier contexto en el cual deba utilizarlos, los debe importar todos. Además, a la hora de utilizarlos, debe comprobar qué conector se está manejando, para así poder ejecutar las funciones que brinda el mismo.

La implementación de los conectores genera un alto nivel de acoplamiento, puesto que existe una alta dependencia entre la aplicación principal y los conectores. Esto es debido a que los conectores están embebidos en el sistema, y se requiere que estén independientes del mismo. Actualmente el Sistema de Aprovisionamiento de Usuarios solamente se conecta a 3 servidores: MySQL, PostgreSQL y LDAP, sin embargo este sistema tiene la intención de desplegarse en la UCI y no cuenta con los principales servicios que se utilizan en la universidad**.**

Todo lo anteriormente expuesto, conlleva a expresar el siguiente **problema de investigación**: ¿Cómo establecer la interacción con los servidores de aplicaciones y servicios que utiliza el Sistema de Aprovisionamiento de Usuarios de la Universidad de las Ciencias Informáticas para realizar las tareas de aprovisionamiento?

Con el propósito de responder esta interrogante la investigación estará enmarcada en el siguiente **objeto**

de estudio: La interacción entre servidores de aplicaciones y servicios en el proceso de aprovisionamiento de usuarios.

Identificado el objeto de estudio se propone el siguiente **objetivo general**: Desarrollar un paquete de conectores que permita al Sistema de Aprovisionamiento de Usuarios la interacción con los servidores de aplicaciones y servicios.

Dentro del cual se incide en el **campo de acción:** La interacción entre servidores de aplicaciones y servicios mediante el uso de conectores en el proceso de aprovisionamiento de usuarios en la UCI.

En función del cumplimiento del objetivo general se llevarán a cabo los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación a partir del estudio de diferentes sistemas que utilizan los conectores para interactuar con servidores de aplicaciones y servicios.
- Definir la metodología y las tecnologías a utilizar para el desarrollo del paquete de conectores.
- Diseñar el paquete de conectores.
- Implementar el paquete de conectores.
- Validar el desarrollo del paquete de conectores a través de las pruebas unitarias y de aceptación.

Con vistas al cumplimiento de cada objetivo específico, se ejecutarán en el orden de aparición cada una de las **tareas de investigación** correspondientes:

- Comprobación de la validez del problema a resolver.
- Identificación de las necesidades del sistema a desarrollar.
- Selección y observación de los casos de estudios más significativos vinculados al objetivo general.
- Recopilación y ordenamiento de la información obtenida a partir de la bibliografía utilizada.
- Caracterización de la metodología de desarrollo a utilizar.
- Definición de las tecnologías y las herramientas a utilizar para el desarrollo del Paquete de conectores.
- Elaboración de la propuesta de solución.
- Especificación de los requisitos funcionales y no funcionales.
- Discusión y definición de la arquitectura del paquete de conectores.
- Definición de los patrones de diseño.
- Confección del diagrama de clases del diseño.

- Definición del estándar de codificación.
- Definición de las tareas de ingeniería.
- Codificación de las historias de usuario.
- Validación de funcionalidades de la solución propuesta mediante pruebas unitarias y de aceptación.

Para llevar a cabo las tareas de investigación se utilizarán los siguientes **métodos de la investigación:**

Métodos teóricos: Histórico-lógico, Analítico-sintético y Modelado.

Histórico-lógico: a través de este método se realizó el análisis de cómo se utilizan los conectores para la interacción de servidores de aplicaciones y servicios en diferentes sistemas, obteniéndose así las características, aspectos negativos y positivos del proceso en cuestión, lo cual sirvió para llegar a la conclusión de la necesidad de realizar un sistema que facilite este proceso con la arquitectura seleccionada.

Análisis - síntesis: a través de este método se revisó y se analizó la bibliografía disponible para realizar una investigación lo más completa posible sobre la interacción de servidores de aplicaciones y servicios. Se sintetizaron los aspectos más importantes para la investigación, para buscar vías y solucionar los problemas que se generaron.

Modelado: a partir de todo lo investigado se realizaron los modelos correspondientes a la metodología XP, que ayudaron a dar cumplimiento a las tareas de diseño de los procesos involucrados en la solución, como son las historias de usuario y las tarjetas CRC.

Métodos empíricos: Observación, Entrevista.

Observación: permitió analizar cada fase del proceso, cada tarea que se realizó y tomar experiencia de esta para aplicarla en todas las demás, por lo que se observó exhaustivamente el funcionamiento de los conectores dentro de diferentes sistemas, lo cual permitió identificar todos los problemas que se deseaban resolver.

Entrevista: se utilizó para interactuar con especialistas que poseen un amplio conocimiento en el tema de los conectores, como es el caso de la entrevista realizada al Ing. Yendry Machado García para obtener información del Sistema de Aprovisionamiento de Usuarios del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas. (Ver **Anexo 1**)

Posible resultado

- Consolidar a través del paquete de conectores las variantes de interacción a los servidores que utiliza el Sistema de Aprovisionamiento de Usuarios de la Universidad de las Ciencias Informáticas.

Justificación de la investigación

Con el desarrollo de esta investigación se proporcionará un paquete de conectores que facilitará al Sistema de Aprovisionamiento de Usuarios la interacción con diferentes servidores de aplicaciones y servicios para realizar las tareas de aprovisionamiento de usuarios en la UCI. Se permitirá la adición de nuevos conectores sin realizar cambios en la interfaz principal. El sistema que utilice la solución puede deshacer acciones en caso de ocurrencia de errores. Además se realizará un aporte tecnológico al departamento de desarrollo del Centro de Identificación y Seguridad Digital.

El trabajo de diploma está compuesto por 3 capítulos:

Capítulo 1: En este capítulo se relacionan todos los conceptos, paradigmas y soluciones que permiten obtener un mejor entendimiento para dar solución a la problemática planteada. También se llevará a cabo un análisis general en cuanto a las herramientas, lenguajes de programación y tecnologías que serán usadas durante el proceso de desarrollo del paquete de conectores.

Capítulo 2: El objetivo que se persigue con la elaboración de este capítulo es mostrar la evolución del paquete de conectores a desarrollar, se presentan las fases de Planificación y Diseño definidas por la metodología Programación Extrema (XP), para dar solución al problema científico. Se define la arquitectura para desarrollar la solución. Se identifican las historias de usuarios y los requisitos no funcionales, se realiza el plan de iteraciones y plan de entregas.

Capítulo 3: Se muestran algunos de los artefactos relacionados con la implementación del paquete de conectores, como los diagramas de componentes y despliegue. Se valida el sistema implementado mediante pruebas unitarias y de aceptación.

Capítulo 1: La interacción entre servidores de aplicaciones y servicios en el proceso de aprovisionamiento de usuarios.

Este capítulo aborda los elementos teóricos que sustentan el objetivo de la investigación. Se relacionan todos los conceptos, paradigmas y soluciones que permiten obtener un mejor entendimiento del objeto de estudio definido. Se precisa la metodología, las tecnologías, así como las herramientas que se utilizan en el desarrollo de este tipo de solución para lograr el objetivo propuesto.

1.1 Aprovisionamiento de usuarios

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. Este proceso encierra la gestión del ciclo de vida de las cuentas de usuarios y se basa en crear, leer, modificar y eliminar la información de los usuarios en las cuentas de los recursos de una organización. Además facilita la gestión centralizada de los atributos de usuarios que se encuentran de manera local en cada recurso (6).

Generalmente están compuestos por determinados elementos como una interfaz de usuario, donde los mismos pueden realizar solicitudes y aprobar o denegar cambios propuestos de aprovisionamiento. Además se pueden encontrar el repositorio de datos, que es el encargado de rastrear los objetos de usuarios y otros datos de sistemas integrados y aplicaciones. Cuentan con un motor de aprovisionamiento que interpreta las solicitudes enviadas y las distribuye entre los recursos apropiados (7). También entre dichos elementos se encuentran los conectores, los cuales leen la información acerca de los usuarios y realizan actividades de actualización como crear, modificar y eliminar cuentas de usuarios (6).

Partiendo de lo anteriormente expresado, el aprovisionamiento de usuarios es la gestión del ciclo de vida de las cuentas de usuarios y la administración de los recursos asociados a estos. Para realizar este proceso se utilizan, a nivel general, un motor de aprovisionamiento, una interfaz principal y los conectores para la interacción con los recursos de la entidad en cuestión.

1.2 Los conectores y su rol en la interacción entre servidores de aplicaciones y servicios

Varios han sido los conceptos emitidos sobre los conectores, ejemplo de ello son los que se mencionan a continuación: según plantea Laura Barros, Patricia López y José M. Drake un conector es un elemento especializado que incorpora en su código el mecanismo de comunicación que se necesita para establecer

la interacción entre un componente cliente y una faceta de un componente servidor. La funcionalidad que debe ser implementada por un conector es la serialización y deserialización de la información que intercambian, la transferencia de las invocaciones a través de la red, la sincronización del cliente a la espera o no de que la invocación termine, y la invocación local del servicio en el componente servidor (8).

José Luis Pastrana, Ernesto Pimentel y Miguel Katrib puntualizan que un conector es el componente encargado de manejar la conexión y adaptar el comportamiento de otro componente que se desee utilizar. El cliente es quién decide las restricciones y comportamientos que desea o necesita de dicho componente servidor para que su funcionalidad satisfaga sus necesidades. Permite que un cliente trabaje a través de él con uno o más servidores diferentes. Pueden ser enriquecidos tanto estática como dinámicamente con el fin de resolver problemas de adaptación de servicios (9).

Gary T. Leavens y Murali Sitaraman en su libro *Foundations of Component Based Systems* afirman que los conectores son los encargados de mediar las actividades de comunicación y coordinación entre los componentes, como pueden ser clientes, objetos, bases de datos o servidores. Representan las interacciones más complejas, tales como un servidor-cliente o un enlace entre una base de datos y una aplicación. Tienen interfaces que definen los roles interpretados por los diferentes participantes en la interacción representada por el conector (10).

A partir de los conceptos anteriormente consultados, para esta investigación, un conector contiene las funcionalidades necesarias para interactuar con los servidores. Para lograr esta interacción, utiliza las funcionalidades básicas de comunicación de una API o librería. Los conectores emplean las peticiones del sistema, y a través de las funcionalidades que tiene implementadas, pueden realizar los cambios pertinentes en el servidor para el cual fue implementado.

La Interfaz de Programación de Aplicaciones (API del inglés Application Programming Interface) es una interfaz de comunicación entre componentes de *software*. Uno de sus principales objetivos es proporcionar un conjunto de funciones de uso general además de representar un método para conseguir abstracción en la programación. Permite al programador acceder a servicios de una aplicación a través del uso de un lenguaje de programación. Esta le ofrece al programador un cierto nivel de abstracción que enmascara la complejidad de acceso a un sistema o aplicación, proponiéndole un conjunto de funciones de las cuales solo se conocen los parámetros y los valores devueltos.

Francis y Dominique, en su libro La alquimia de las multitudes, define a una API como: “Puertas que los creadores de los programas abren de forma voluntaria para permitir que otros creadores puedan apropiarse de los elementos que interesen de dichos programas y añadir sus propios servicios” (11). Juan Desongles, en conjunto con varios autores, definen a una API como: “un conjunto de rutinas, protocolos y herramientas para construir aplicaciones de interfaz” (12).

1.3 Conectores de Identidad del Sistema de Administración de Identidades de Oracle

El Sistema de Administración de Identidades de Oracle es una plataforma completa e integrada, que permite a las organizaciones gestionar eficazmente el ciclo de vida de extremo a extremo de las identidades de los usuarios en todos los recursos de la empresa. Se encuentra construido sobre una arquitectura moderna e innovadora que combina escalabilidad extrema con flexibilidad por medio de su *framework* de conectores, que proporciona una rápida integración con los sistemas (13). Según el artículo reglas y licenciamientos *de Oracle Identity Manager,* un conector enlaza el producto de *software* con un producto externo y se requiere de un conector único por cada producto que el *software* requiera para interconectarse (14).

Como componente de esta solución, se encuentran los Conectores de Identidad (*Identity Connectors*), que permiten al sistema integrarse con aplicaciones externas enfocadas en la administración de identidades y llevar a cabo el aprovisionamiento de usuarios. Cada conector contiene una serie de adaptadores, donde cada uno ejecuta una operación determinada en el sistema externo. Los conectores de identidad se encuentran en una plataforma única, la instalación es sencilla, permiten la ejecución remota y la reutilización de código (13).

Al realizar el proceso de aprovisionamiento de un usuario, el sistema ejecuta una serie de tareas de procesamiento. Cada tarea de procesamiento ejecuta una operación determinada durante el proceso de aprovisionamiento. Un adaptador es el componente que llama directamente el código para realizar una operación de aprovisionamiento específica en el sistema objetivo. A la vez, un adaptador es llamado por una tarea de procesamiento, por lo que existe una relación de uno a uno entre un adaptador y una tarea de procesamiento. El código que llama el adaptador está implementado para ser compatible con las características o servicios que provee el sistema objetivo para realizar operaciones de aprovisionamiento iniciadas desde otros sistemas.

Los conectores de identidad están compuestos por los siguientes componentes:

- ✓ El marco de conector de identidad: Proporciona características comunes a los desarrolladores. Cuenta con una plataforma única, la instalación es sencilla y permite la reutilización. Se separa en dos partes:
  - ➢ La API conector que es la encargada de presentar una visión coherente de un conector, independientemente de las operaciones que esté aplicando, utilizan la API para llamar a los conectores.
  - ➢ El SPI (Servicio de Interfaz de Proveedores) es un conector que está formado por varias interfaces para que el desarrollador sólo tenga que implementar las interfaces que la aplicación o el sistema de destino soporta.
- ✓ El servidor de conexión (opcional): el cual permite a una aplicación ejecutar de forma remota uno o más paquetes de conectores que se implementan en otro sistema (15).

El Sistema de Administración de Identidades de Oracle permite conocer cómo se realiza una rápida integración con los sistemas y este es un aspecto que el paquete de conectores a desarrollar debe ser capaz de poder realizar. No se hará uso de conectores predefinidos que estarán diseñados específicamente para una aplicación de destino, debido a que los conectores pueden utilizarse por cualquier aplicación con necesidades similares al Sistema de Aprovisionamiento de Usuarios. Es necesario aclarar que no serán utilizados los conectores de identidad al igual que Oracle, puesto que no es necesario la división de cada conector en varios adaptadores. En la solución esperada se requiere que el conector brinde las funcionalidades requeridas para interactuar con el servidor para el cual fue concebido.

1.4 Sistema de Aprovisionamiento de Microsoft

Una de las empresas que ha apostado por el desarrollo de gestión de identidades integrados que combinen los mejores productos y prestaciones de sistemas de administración de identidades y ha alcanzado éxito a nivel internacional es el Sistema de Aprovisionamiento de Microsoft (*Microsoft Provisioning System* por sus siglas en inglés MPS). Es una plataforma basada en Windows extensible que proporciona un marco basado en el Lenguaje de Etiquetado Extensible XML que permite crear soluciones de aprovisionamiento a medida para web, datos y alojamiento de aplicaciones. Esta solución incluye tareas tales como añadir nuevos usuarios, la actualización de las entradas de directorio, y el aprovisionamiento de aplicaciones y servicios.

Para iniciar los servicios de aprovisionamiento, se provee una petición XML como entrada al MPS. Luego el sistema convierte el XML especificado a llamadas a funciones de la API hacia aplicaciones externas como Microsoft Active Directory[1] donde las tareas apropiadas de aprovisionamiento son ejecutadas. Cuenta con un motor de aprovisionamiento transaccional que permite que las transacciones que fallen sean revertidas. Dicho sistema también posee un historial de transacciones de aprovisionamiento. MPS brinda facilidades de colas de espera para aceptar peticiones que pueden ser ejecutadas más adelante. Los componentes de MPS proporcionan un entorno robusto de aprovisionamiento con una arquitectura extensible que permite el empleo de gran variedad de aplicaciones y servicios.

En este sistema los conectores se encuentran en los proveedores, que es la biblioteca de vínculos dinámicos que contienen los procedimientos programáticos que hacen llamadas de la API a las aplicaciones externas que implementan las tareas de bajo nivel iniciados por las solicitudes de aprovisionamiento. Hacen posible automatizar las tareas necesarias para la prestación y gestión de servicios en un centro de datos. Aceptan solicitudes XML desde el motor de aprovisionamiento y ejecutan tareas de aprovisionamiento. Uno de los beneficios de MPS es la posibilidad de la adición de proveedores personalizados (16).

La solución del sistema de aprovisionamiento de Microsoft (MPS) brinda la posibilidad de conocer cómo utilizan el motor de aprovisionamiento transaccional que permite que en caso de que falle alguna transacción, se puedan deshacer las operaciones realizadas hasta ese instante. El paquete de conectores a desarrollar requiere que el sistema que lo utilice pueda deshacer acciones en caso de ocurrencia de errores por lo cual es factible conocer como este sistema logra realizar este proceso para obtener una visión de cómo realizar esta funcionalidad en el sistema a desarrollar aunque no se va a implementar de la misma forma.

El paquete de conectores no puede construirse de la misma forma que MPS debido a que se consideró que recibir las peticiones en formato XML es más costoso que utilizar un tipo de dato o formato más sencillo y rápido de utilizar, como los diccionarios, que son un tipo de colección del lenguaje, con los cuales no habría que leer el documento XML para poder extraer la información necesaria para ejecutar las operaciones. Sin

[1] Microsoft Active Directory: Es el término que utiliza *Microsoft* para referirse a su implementación de servicio de directorio en una red distribuida de computadores.

embargo, utilizando un diccionario, para obtener la información, solo es necesario extraer el valor a través de la llave o clave.

1.5 WSO2 Enterprise Service Bus (ESB)

WSO2 Enterprise Service Bus (ESB) es una aplicación de código abierto que se encarga de la mediación entre aplicaciones cuyos mensajes de comunicación son diferentes. Brinda un mecanismo para que el flujo de mensajes entre estas aplicaciones sea entendible entre ambas partes. Para interactuar con los diferentes servicios o funcionalidades que podría brindar una determinada aplicación externa, en ESB se implementan conectores que son los encargados de interactuar con la API indicada para acceder a estos servicios o funcionalidades (17).

Según la página oficial de WSO2 *Enterprise Service Bus* un conector es una colección de plantillas que definen los usuarios y pueden llamar a las operaciones de sus configuraciones ESB para acceder fácilmente a la lógica específica para el procesamiento de los mensajes. Típicamente, los conectores se utilizan para envolver el API[2] de un servicio externo (18).

ESB posibilita que mediante archivos de configuración escritos en XML se definan y configuren todos los componentes necesarios para desplegar un conector. Cada fichero de configuración contiene, respectivamente:

- Las dependencias necesarias para que el conector pueda acceder a los repositorios que contienen las librerías requeridas para implementar las funcionalidades básicas del mismo.
- La descripción general con parámetros como el nombre del conector.
- El paquete donde se encontrará disponible.
- Las diferentes categorías de operaciones que tendrá definidas.
- Una descripción para cada operación por categoría, donde ya se definen las operaciones que contendrá el conector.
- Para cada operación se configuran los pasos necesarios para que esta interactúe con la API indicada.

[2] Interfaz de Programación de Aplicaciones traducida del inglés como *Application Programming Interface*

Los ficheros de configuración están diseñados de forma jerárquica, descendentemente en la jerarquía. Cada fichero configura una parte más específica del conector, o sea, desde los aspectos informativos más generales hasta la definición en sí de las operaciones que implementará el conector. Mediante esta implementación de conectores a través de ficheros de configuración XML se obtiene un mecanismo de reutilización del código. Cuando se crean los conectores, que utilizan el mismo recurso o API, simplemente se tiene que hacer referencia en los repositorios a las librerías adecuadas y realizar las configuraciones deseadas a las operaciones del conector.

El proceso de creación de conectores es más sencillo gracias a que ESB cuenta con una infraestructura de repositorios de APIs y conectores. Esta aplicación posee conectores por defecto para un grupo de aplicaciones externas las cuales destacan por sus servicios como Twitter o Salesforce. Por tal motivo si se desea interactuar con alguna de estas aplicaciones, solamente se tendría que adicionar este conector desde el repositorio (17).

Con respecto a los conectores en la solución de ESB debido a que, para el desarrollo del paquete de conectores, no se cuenta con una infraestructura de conectores como la que esta empresa posee, por tal motivo no se puede implementar los conectores a través de archivos de configuración XML solamente. No obstante, esta visión de cómo configuran en ESB los diferentes componentes de un conector brinda una idea de cómo deben organizarse y separarse estos componentes en la arquitectura de las librerías a implementar, además de qué componentes no deben faltar en estas librerías.

1.6 Línea de ensamblaje de IBM Tivoli Directory Integrator

La principal herramienta dentro de Tivoli Directory Integrator es la línea de ensamblaje (en inglés *AssemblyLine*). Los componentes de la línea de ensamblaje son los conectores, las funciones, *scripts*, atributos mapas y las ramas. La principal tarea es procesar datos como entradas, registros, elementos y objetos de un directorio LDAP, lo transforma y lo envía a la base de datos de Perfiles. Cuando importa datos desde varios directorios LDAP, la línea de ensamblaje procesa, transforma y combina todos los datos de origen antes de enviarlos. Está diseñado y optimizado para trabajar con un elemento cada vez.

Dispone de una rica biblioteca de componentes con los que puede trabajar. Cada componente está especializado para manejar una API, un protocolo o un formato en particular. Los conectores son los componentes que necesita para crear una línea de ensamblaje. Ellos están diseñados para que no tengan que ocuparse de los detalles técnicos a la hora de trabajar con diversos almacenes de datos, sistemas, servicios

o transportes. Cada tipo de conector utiliza un protocolo específico o API para gestionar los detalles del acceso a fuente de datos. Puede crear sus propios conectores para soportar diferentes funciones o utilizar los conectores que se proporcionan con *IBM Connections*[3]. Posee dos categorías de conectores:

- ✓ Se conoce el mecanismo de transporte, pero no el contenido estructurado que normalmente se ve como un flujo de datos. Esta categoría requiere un analizador para interpretar o generar la estructura de contenidos con el fin de que la línea de ensamblaje funcione correctamente.
- ✓ El transporte y la estructura de contenido de datos es conocido por el conector; es decir, el esquema de la fuente de datos se puede consultar o detectarse con las APIs bien conocidos tales como JDBC[4] o LDAP[5] (19).

El modo de un conector de ensamblaje define cuál es el papel que juega el conector en el flujo de datos y controla su comportamiento en la línea de ensamblaje. Cada conector admite sólo un subconjunto de modos que son adecuados para el sistema al cual está conectado. Los conectores pueden ajustarse a uno de estos modos estándar: iterar, buscar, solo adición, actualización, borrar, llamada-respuesta y servidor.

Cuando se selecciona un conector para la línea de ensamblaje, un cuadro de diálogo se muestra lo que le permite elegir el tipo de conector que desea heredar. Los conectores ofrecen una opción de inicialización para controlar que el componente se instala. De forma predeterminada, todos los conectores se inicializan en el momento que la línea de ensamblaje se pone en marcha.

Los conectores pueden ser transferidos a su biblioteca de conectores y se pueden reutilizar para crear nuevas soluciones. La utilización de la biblioteca de IBM Tivoli Directory Integrator hace que el mantenimiento y la mejora de los conectores sean más sencillos. Siempre que actualiza la plantilla del conector en la biblioteca, todas las líneas de ensamblaje derivadas del conector heredan los cambios y las mejoras (20).

---

[3] IBM Connections: es una plataforma de software que tiene como objetivo capacitar a las empresas para que sean innovadoras.

[4] JDBC: *Java Database Connectivity* (en español Conectividad a Bases de Datos de Java) es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

[5] LDAP: *Lightweight Directory Access Protocol* (en español Protocolo Ligero/Simplificado de Acceso a Directorios) que hacen referencia a 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.

La solución de IBM Tivoli Directory Integrator permite obtener otra visión del uso de los conectores y de la arquitectura que se puede utilizar. Sin embargo, no se desarrollará el paquete de conectores de la misma forma, puesto que no es necesario implementar un mecanismo con tantos procesos intermedios como la línea de ensamblaje. Estos procesos que requerirían de la avanzada infraestructura de la solución estudiada para funcionar correctamente. Por otra parte, vale destacar que para cada tipo de conector a implementar se va a utilizar un protocolo específico o API, con el objetivo de lograr que el mismo interactúe correctamente con el servidor deseado.

1.7 Módulo de conectores para el subsistema de Aprovisionamiento de Usuarios del sistema Administración de Identidades.

Uno de los sistemas desarrollados en el Centro de Identificación y Seguridad Digital (CISED), perteneciente a la Universidad de las Ciencias Informáticas (UCI) es el Módulo de conectores para el subsistema de Aprovisionamiento de Usuarios del sistema Administración de Identidades. Este sistema provee múltiples beneficios debido a que permite la gestión y centralización de cuentas de usuarios solicitadas de una manera rápida, segura y sencilla.

Para establecer la comunicación con los recursos se vale de conectores predefinidos en los cuales recaen las tareas de aprovisionamiento para cada tipo de recurso, estos son los encargados de traducir los comandos del sistema de aprovisionamiento 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 (21).

El modulo Conectores está compuesto por los siguientes conectores:

- ✓ OpenLDAPResourceConnectorProvider: Este conector es el encargado de realizar todas las tareas de aprovisionamiento en un Servicio de Directorio OpenLDAP[6]. 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.

[6] OpenLDAP: es una implementación libre y de código abierto del protocolo *Lightweight Directory Access Protocol* (LDAP por sus siglas en inglés) desarrollada por el proyecto OpenLDAP.

- ✓ Oracle11gResourceConnectorProvider: 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.
- ✓ OpenfireResourceConnectorProvider: Este conector para Openfire[7] 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[8]. Por lo tanto para aprovisionar cuentas de usuario es necesario realizar consultas al servidor LDAP que el Openfire tenga asociado. Por tal motivo 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. Solamente se realizan consultas SQL en el servidor de base de datos para bloquear y desbloquear las cuentas existentes.
- ✓ ZimbraResourceConnectorProvider: Este conector contiene todas las funcionalidades implementadas para realizar las tareas de aprovisionamiento en un Servidor de Correo Zimbra corre 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. Este proceso se realiza 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.

[7] Openfire: (anteriormente llamado *Wildfire* y *Jive Messenger*) es un sistema de mensajería instantánea GPL.

[8] MySQL: es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.

Los patrones de diseño que fueron utilizados en el desarrollo para lograr la extensibilidad de las funcionalidades del Módulo de Conectores fueron: *Provider, Factory Method, Singleton* y *Facad*e. Para hacer posible que el módulo soporte las transacciones se utilizó el patrón *Command*.

El Módulo de conectores para el subsistema de Aprovisionamiento de Usuarios del sistema Administración de Identidades en su solución solamente implementó 4 conectores que son el conector para Servidor de Correo Zimbra, el conector para Servidor de Mensajería Instantánea Openfire, el conector para Servidor de Base de Datos Oracle 11g y el conector para Servicio de Directorio OpenLDAP con estos conectores no es suficiente para poder interactuar con todos los servidores a los que brinda acceso la UCI. Este sistema sirve de referencia debido a que utilizan un mecanismo transaccional y lo implementan a través del patrón de diseño *Command*. En el paquete de conectores que se pretende desarrollar se permitirá que el sistema que haga uso de la solución pueda deshacer acciones en caso de ocurrencia de errores para poder realizar esta acción se utilizará un patrón de diseño diferente al empleado por el sistema analizado.

A continuación se presenta una tabla comparativa realizada entre los sistemas analizados donde se tuvieron en cuenta 3 factores diferentes. Primeramente determinar si el sistema es multiplataforma, es decir, si el sistema funciona en las plataformas Microsoft y Linux. Otro factor es comprobar si el sistema utiliza el lenguaje de etiquetado extensible XML para crear los conectores y por último si el sistema permite deshacer acciones para recuperarse ante la ocurrencia de errores.

Nombre del sistema	Multiplataforma	XML	Deshacer acciones
Sistema de Administración de Identidades de Oracle.	Si	No	Si
Sistema de Aprovisionamiento de Microsoft	No	Si	Si
WSO2 Enterprise Service Bus (ESB).	Si	Si	Si
Tivoli Directory Integrator	Si	No	Si
Módulo de conectores	No	No	Si
Sistema de Aprovisionamiento de Usuarios.	Si	Si	No

Tabla 1: Comparación entre sistemas

Fuente: Elaboración propia

1.8 Metodología, tecnologías y herramientas a utilizar

Metodología de desarrollo de *software*

Una metodología de desarrollo de *software* es un marco de trabajo donde se definen los pasos a seguir, cómo y con qué se llevará a cabo un nuevo *software*. Está compuesta por tareas para dividir los procesos del proyecto, procedimientos que definen cómo es que deben realizarse las distintas tareas y técnicas que permiten aplicar los procedimientos. Su objetivo es ayudar a los ingenieros de *software* a obtener un producto final con calidad. Se tienen en cuenta una serie de elementos como los recursos disponibles, la opinión del cliente y el costo asociado al proyecto además de evitarles el caos que se podría generar si no siguen una serie de pasos para desarrollar un *software*.

Las metodologías se pueden agrupar en metodologías pesadas o tradicionales y en metodologías ligeras o ágiles. Es esta última clasificación la que se tiene en cuenta para el desarrollo del paquete de conectores. Las metodologías tradicionales son aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos, tienen una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Sin embargo, las metodologías ágiles están más orientadas a la generación de código con ciclos cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso (22).

Basándose en este planteamiento, se hizo un estudio de dos metodologías ágiles, *Feature Driven Development* (FDD) y la Programación Extrema (XP).

- ***Feature Driven Development***

Feature Driven Development (FDD por sus siglas en inglés) define un proceso iterativo que consta de cinco procesos secuenciales durante los cuales se diseña y construye el sistema. Cada fase del proceso tiene un criterio de entrada, tareas, pruebas y un criterio de salida. Típicamente, la iteración de un rasgo insume de una a tres semanas. Las fases son: desarrollo de un modelo global, construcción de la lista de funcionalidades, planificación por funcionalidad, diseño por funcionalidad e implementación por funcionalidad. El trabajo tanto de modelado como de desarrollo se realiza en grupo.

Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el *software*. FDD es un método ágil, iterativo y adaptativo. A diferencia de otros métodos ágiles, no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción. Esta metodología se considera adecuada para proyectos mayores y de misión crítica. FDD no requiere un modelo específico de proceso y se complementa con otras metodologías (23).

- **Programación Extrema**

La Programación Extrema o *Extreme Programming* (XP por sus siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de *software*. Otras de sus características es que promueve el trabajo en equipo, se preocupa por el aprendizaje de los desarrolladores, y propicia un buen clima de trabajo. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. El ciclo de vida ideal de XP es iterativo e incremental consiste de seis fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto (24).

Lo fundamental en este tipo de metodología es:

- ✓ La simplicidad al desarrollar y codificar los módulos del sistema.
- ✓ La retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.
- ✓ La comunicación entre los usuarios y los desarrolladores (25).

Según lo expresado por algunos autores XP se definen como:

- ✓ "Un proceso ligero, de bajo riesgo, flexible, predecible, científico y divertido de desarrollar *software*".
- ✓ "Una metodología ágil que requiere gran disciplina" (26).

Luego de realizar un análisis se identificaron ciertos factores que permitieron la selección de XP como metodología guía para el proceso de desarrollo del paquete de conectores. El desarrollo en este caso es realizado por pocas personas y con pocos roles. El desarrollo comienza a partir de los requerimientos básicos y a partir de ahí se añaden funcionalidades que tanto los desarrolladores como el cliente entiendan necesarias.

A medida que el proyecto avanza pueden surgir nuevas expectativas o ideas que pueden ser incorporadas fácilmente permitiéndole mayor adaptabilidad al producto. La comunicación entre el equipo de desarrollo y el cliente será constante y directa. Lo antes descrito evidencia cómo las características del paquete de conectores se inclinan a adoptar las prácticas y valores de la metodología XP, por lo cual se selecciona la misma como guía del proceso de desarrollo del *software*.

Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado (IDE del inglés *Integrated Development Environment*) es un entorno de programación. Es un editor de código, compilador, depurador y constructor de interfaz gráfica. Facilita la interacción con los lenguajes de programación.

- ***PyCharm***

El entorno de desarrollo integrado *PyCharm* 3.4, proporciona características como análisis de código, depuración gráfica y entorno de pruebas de unidad integrada. Incluye un editor de código inteligente que entiende las capacidades específicas de *Python*. Presenta múltiples comodidades como: formateo de código automático y completamiento del código para palabras reservadas, variables y clases. Automáticamente detecta qué tipo de consola debería correr basado en el tipo de proyecto abierto. Tiene capacidad remota de desarrollo: puede correr, depurar, probar y desarrollar aplicaciones en hosts remotos o máquinas virtuales (27). Se decide utilizar *PyCharm* 3.4 debido a las características que proporciona y por ser multiplataforma. Las facilidades que brinda serán aprovechadas en el desarrollo del paquete de conectores que dará solución al objetivo de la presente investigación.

Lenguajes utilizados

- **Lenguaje de modelado**

Se emplea el Lenguaje Unificado de Modelado o *Unified Modeling Language* (UML por sus siglas en inglés) en su versión 2.0, por ser un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema. Brinda un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de *software* reutilizables (28).

Se empleó este lenguaje para realizar un adecuado proceso de análisis y diseño orientado a objetos del sistema. Además fue seleccionado por ser el lenguaje utilizado por la herramienta de modelado *Visual Paradigm* seleccionada para la presente investigación y permite al mismo tiempo modelar todos los artefactos generados por la metodología XP.

- **Lenguaje de programación**

El uso del lenguaje de programación *Python 2.7.X* en la siguiente investigación está regido principalmente a la petición realizada por el cliente. El cliente identificó las potencialidades de este lenguaje y expresó la necesidad de contar con un paquete de conectores en este lenguaje para que a la vez sea compatible con el Sistema de Aprovisionamiento de Usuarios debido a que se realiza utilizando el mismo lenguaje.

Es un lenguaje de programación multiparadigma, más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación estructurada y programación funcional. Utiliza tipado dinámico y fuertemente tipado. Además posee una

sintaxis cómoda para desarrollar sobre él y permite grandes facilidades por la cantidad de estructuras de datos que ya tiene implementada. Estas estructuras de datos permiten crear sentencias complejas en pocas líneas de código para ganar así eficiencia y rapidez en la escritura del paquete de conectores (29).

- **Lenguaje de Etiquetado Extensible**

Se hará uso del Lenguaje de Etiquetado Extensible (XML por sus siglas en inglés) en su versión 1.0, que describe los datos y cómo estos se estructuran, mediante el cual los desarrolladores pueden crear sus propios elementos para alcanzar sus necesidades de información (31). A continuación se presentan las características principales del lenguaje:

- ✓ El XML no es un lenguaje de presentación, sino para describir datos, lo que facilita una búsqueda más precisa, permite compartir datos de forma eficiente.
- ✓ XML permite definir nuevas etiquetas y atributos en un documento donde las estructuras de contenido pueden anidarse a cualquier nivel de complejidad, lográndose un control absoluto sobre dichas estructuras y permite la adición de nuevos atributos para incrementar el control de la información.
- ✓ Se pueden transportar enormes cantidad de datos bien organizados junto con descriptores que indican los elementos y la estructura de datos que contiene el documento.
- ✓ Un documento XML puede contener una descripción opcional de su gramática para uso de las aplicaciones que necesiten realizar una validación de la estructura del mismo (32).

Es factible el uso del XML en el paquete de conectores debido a que con el uso de un fichero XML se puede guardar de manera organizada toda la información relacionada a la descripción de un conector, y así usar esta información para gestionar las facilidades que brinda el mismo.

- **Lenguaje de Definición de Esquema**

Será utilizado el Lenguaje de Definición de Esquema (XSD por sus siglas en inglés) en su versión 1.1, para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma precisa. Un XSD es el encargado de definir la correcta estructura de los elementos del documento XML, define los elementos que pueden aparecer en dicho documento así como los atributos de esos elementos. Precisa además, qué elementos son hijos de los elementos principales del documento XML y la secuencia en la cual los hijos de los elementos pueden aparecer en él. Los XSD facilitan la reutilización de código.

El modelo de datos de XSD incluye: el vocabulario (nombres de elemento y atributo), el contenido modelo (relaciones y estructura) y los tipos de datos (33). Es necesario el uso del XSD debido a que permite comprobar que el XML de cada uno de los conectores está formado correctamente y cumple con la estructura definida.

Herramientas de desarrollo

- **Herramienta XML Copy Editor**

Para la creación del XSD se utilizará la herramienta *XML Copy Editor* en su versión 1.2.1.3 que es un editor rápido, de código abierto y gratuito. Cuenta con una interfaz intuitiva y bien diseñada para la vista, por lo que estas tareas no resultarán difíciles de llevar a cabo ya sea en el sistema operativo Windows o en Linux, y las principales características que posee es que permite validar XSD, resalta la síntesis, comprueba la ortografía y permite el plegado de fragmentos de código (34).

- **Herramienta *VMware Workstation***

Se hará uso de la herramienta *VMware Workstation* en su versión 8.04 la cual es un sistema de virtualización por *software*. Es un programa que simula un sistema físico con unas características de *hardware* determinadas. Permite ejecutar varios computadores (sistemas operativos) dentro de un mismo *hardware* de manera simultánea, para lograr así el mayor aprovechamiento de recursos. Sin embargo, al ser una capa intermedia entre el sistema físico y el sistema operativo que funciona en el *hardware* emulado, la velocidad de ejecución de este último es menor, pero en la mayoría de los casos suficiente para usarse en entornos de producción (35). Este programa será utilizado durante la implementación del paquete de conectores y para la realización de las pruebas.

- **Herramienta CASE**

Las herramientas de Ingeniería de *Software* Asistida por Computadora (CASE por sus siglas en inglés). Son sistemas de *software* que intentan proporcionar ayuda automatizada a las actividades del proceso de *software*. Se encuentran destinadas a aumentar la productividad en el desarrollo de *software* reducen el costo de las mismas en términos de tiempo y de dinero, a través del apoyo a la realización de los diseños del proyecto, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras según lo indicado por (36).

Se decide utilizar la herramienta CASE *Visual Paradigm* 10.0 porque la misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta

la generación del código fuente de los programas y la documentación. Ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste ya que permite generar código desde diagramas y generar documentación.

Esta herramienta se puede ejecutar sobre diferentes sistemas operativos lo que le confiere la característica de ser multiplataforma. Permite soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de todo tipo de diagramas (37). Se decidió utilizar debido a que es una herramienta de fácil entendimiento y de la cual se tiene mayor conocimiento pues fue estudiada y empleada en las asignaturas de Ingeniería de *Software* 1 e Ingeniería de *Software* 2.

1.9 Conclusiones parciales

En el presente capítulo se plantearon conceptos fundamentales para lograr una mejor comprensión de la problemática. Se realizó un análisis de las tendencias actuales y se tuvieron en cuenta empresas que son líderes en el mundo en el uso de conectores en sus aplicaciones informáticas, se realizó un análisis para determinar cómo implementan los conectores, las metodologías, los lenguajes y el entorno de desarrollo integrado que utilizan. Se puede concluir que para dar solución a la problemática planteada es necesario desarrollar un paquete de conectores para la interacción de servidores de aplicaciones y servicios. Para ello, la selección del lenguaje de programación *Python* 2.7.X. La herramienta CASE que se utilizará es *Visual Paradigm* 10.0 y el IDE *PyCharm* 3.4, las cuales permitirán la agilización del desarrollo de la investigación. La metodología de desarrollo de *software* XP será de suma importancia para guiar el proceso de desarrollo del sistema.

Capítulo 2: Análisis y diseño del paquete de conectores

El objetivo que se persigue con la elaboración de este capítulo es mostrar la evolución del paquete de conectores a desarrollar. Se realizará un análisis y una amplia comprensión de todos los elementos que se relacionan con los conectores para la interacción de servidores y aplicaciones. Al mismo tiempo se desarrollarán las fases iniciales de la metodología de desarrollo XP, planificación y diseño. Además se presentan los artefactos realizados, los cuales serán premisas cruciales para la entrega final de la solución.

2.1 Acercamiento al Sistema de Aprovisionamiento de Usuarios

El Sistema de Aprovisionamiento de Usuarios del centro CISED utiliza los conectores con el propósito que persigue el paquete de conectores a desarrollar. En este sistema los conectores son usados para tener acceso a algunos de los servidores que brinda la Universidad y gestionar las cuentas de usuarios. Actualmente no permite aprovisionar en otros recursos que deben ser utilizados, como por ejemplo Oracle, Openfire y Zimbra. Carece de un mecanismo que le permita recuperarse ante la ocurrencia de fallas por lo cual si en el proceso de creación de cuentas de un usuario algún servidor falla entonces el proceso de aprovisionamiento estará incompleto y no se pueden deshacer los cambios realizados en los restantes servidores.

Los conectores están implementados de manera independiente, o sea que no cuenta con una interfaz única para manipular las operaciones. Por tal motivo la información para ejecutar funcionalidades similares no es compatible entre recursos. En el momento que se ejecuta una funcionalidad, primero se debe comprobar a que recurso se accede para poder saber que parámetros necesita esta funcionalidad.

La implementación de los conectores genera un alto acoplamiento, por lo que existe una alta dependencia entre la aplicación principal y los conectores, esto es debido a que los conectores están embebidos en el sistema y se requiere que estén independientes de la aplicación. Los conectores están estáticos por tanto en los formularios de la aplicación hay que importar directamente cada una de las librerías de estos conectores; lo cual conlleva a tener mucha codificación en los formularios para lograr cumplir con el negocio.

2.2 Propuesta de solución

La solución consiste en un paquete de conectores que permitirá al Sistema de Aprovisionamiento de Usuarios, o cualquier otro sistema con necesidades similares, el acceso a los servidores de la Universidad, sea correo,

chat, base de datos o directorio activo. Con el objetivo de poder crear a los nuevos usuarios sus cuentas en estos recursos y realizar otras funciones de aprovisionamiento requeridas. Como la solución a la que se pretende llegar es que desde una misma clase se pueda acceder a las funcionalidades de cada conector, lo más óptimo es utilizar las interfaces como facilidad o componente de un lenguaje el cual permite implementar este tipo de comportamiento.

Debido a que el lenguaje que se utiliza es *Python* y el mismo no cuenta con interfaces, se utilizarán los ficheros de configuración XML para poder describir la información de cada conector y así poder implementar una clase que, leyendo esta información, pueda acceder a las funcionalidades de los conectores, en otras palabras, que haga función de interfaz. Será utilizado el esquema XSD para definir la correcta estructura de los elementos del documento XML, el cual permitirá comprobar que el XML de configuración del conector tenga la estructura correcta.

Durante los procesos de aprovisionamiento, se dará la opción de deshacer acciones en caso de ocurrencia de errores. En caso de que ocurra un error de conexión, se realizan una serie de reintentos de conexión a determinados intervalos de tiempo, y además se comprueba si los datos del usuario se encuentran presentes o no en el servidor correspondiente, para que en caso de no estar presentes, proceder a insertar esta información.

Los conectores estarán independientes de la aplicación para que la codificación en el sistema superior no tenga que variar cuando se necesite utilizar otro conector. Por esta razón se realiza esta solución que brinda al sistema superior un módulo que cumple la función de interfaz principal, para que este pueda trabajar a través de este módulo con los conectores de manera unificada y sin tener que integrar estos conectores directamente en el código. Todo lo anteriormente expuesto permite que no sea necesario realizar cambios en el código del sistema cuando se tenga que trabajar con varios conectores. Deben quedar desarrollados conectores para 7 tipos de recursos diferentes.

El paquete de conectores brinda la opción de adicionar nuevos conectores sin tener que realizar cambios en la interfaz principal. Para lograr esto se debe añadir en la carpeta “Paquetes” una nueva carpeta con los componentes del nuevo conector, los cuales deben ser los módulos con las funciones del mismo y el fichero de configuración XML que describe a este conector. Se debe crear una nueva entrada en el fichero de configuración principal con la información general del conector y la ubicación del fichero de configuración del nuevo conector. Por último se debe añadir al lenguaje de programación *Python* la API que se va a utilizar.

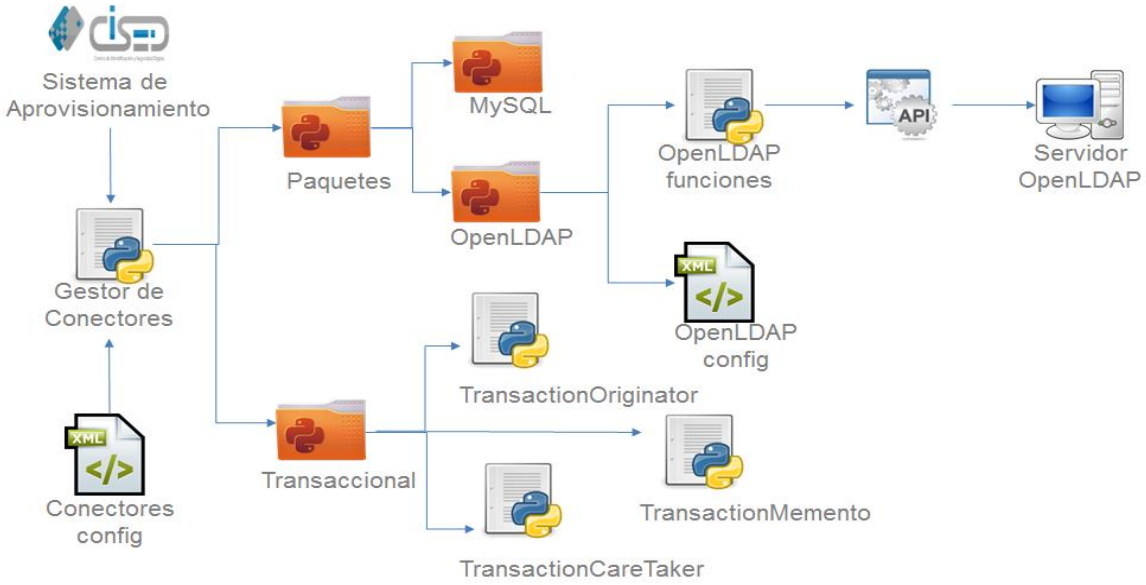


Figura 1: Propuesta de solución

Fuente: Elaboración propia.

2.3 Modelo de dominio

Con el objetivo de representar el vocabulario y los conceptos clave del dominio del problema se realizó el modelo de dominio. En este se identifican las relaciones entre todas las entidades comprendidas en el ámbito del dominio del problema. El modelo de dominio ilustra los principales conceptos con los que trabaja el paquete de conectores a desarrollar, ayuda a comprender su entorno y a definir sus clases más significativas. El mismo se realiza cuando los procesos de negocio no están bien estructurados, por lo que se presenta una vista conceptual de forma general de la aplicación para capturar los objetos más importantes, los eventos que suceden en el entorno donde estará el sistema y la relación existente entre ellos.

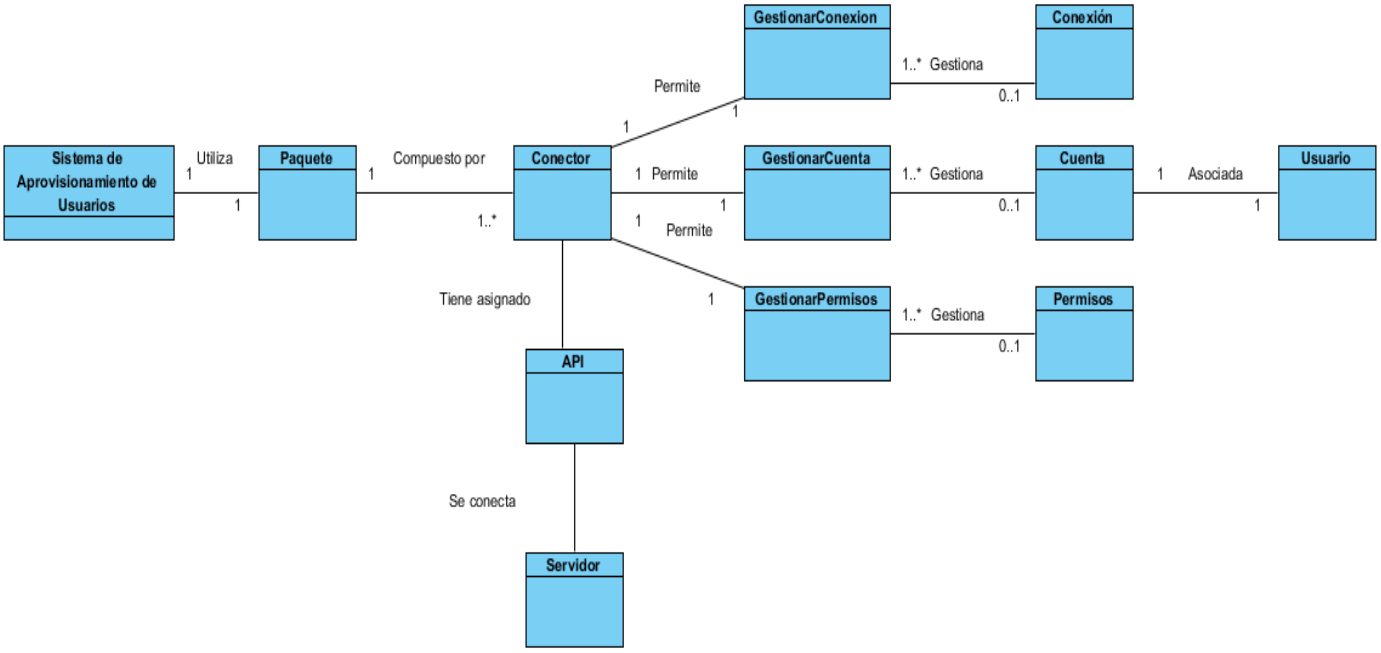


Figura 2: Modelo de dominio.

Fuente: Elaboración propia.

Descripción:

- ✓ **Sistema de Aprovisionamiento de Usuarios:** Representa la aplicación del Sistema de Aprovisionamiento de Usuarios del centro CISED de la Universidad de las Ciencias Informáticas.
- ✓ **Paquete:** Representa al paquete de conectores para la interacción de servidores de aplicaciones y servicios.
- ✓ **Conector:** Representa cada uno de los conectores que se encuentran en el Paquete de conectores para la interacción de servidores de aplicaciones y servicios.
- ✓ **API:** Representa al componente fundamental de los conectores, puesto que el conector no hace más que utilizar las funciones que brinda la API, modificarlas de acuerdo al negocio y así poder realizar cambios en el servidor.
- ✓ **Servidor:** Representa los diferentes servidores de aplicaciones y servicios con los cuales el paquete de conectores va a interactuar.
- ✓ **GestionarConexión:** Gestiona todas las acciones que se realizan sobre la conexión.
- ✓ **GestionarCuenta:** Gestiona todas las acciones que se realizan sobre las cuentas.
- ✓ **GestionarPermisos:** Gestiona todas las acciones que se realizan sobre los permisos.
- ✓ **Cuenta:** Representa las cuentas de usuario creadas en el sistema de Aprovisionamiento de Usuarios.
- ✓ **Conexión:** Representa las conexiones realizadas.
- ✓ **Permisos:** Representa los permisos que se pueden tener en el sistema.
- ✓ **Usuario:** Representa los usuarios que interactúan con el sistema de Aprovisionamiento de Usuarios.

2.4 Captura de requisitos

La obtención de los requisitos es un paso importante para las siguientes etapas del desarrollo del *software*, pues un error en estas fases iniciales puede ocasionar que el sistema no cumpla las expectativas de los usuarios y difícilmente aporte valor agregado al negocio para el que debe ser concebido (25).

Requerimientos funcionales (RF)

- **RF1 Gestionar conexión en el recurso Servidor de Base de Datos Oracle.**
  RF1. 1 Abrir conexión en el recurso Servidor de Base de Datos Oracle.
  RF1. 2 Cerrar conexión en el recurso Servidor de Base de Datos Oracle.
  RF1. 3 Probar conexión en el recurso Servidor de Base de Datos Oracle.
- **RF2 Gestionar conexión en el recurso Servidor de Base de Datos MySQL.**
  RF2. 1 Abrir conexión en el recurso Servidor de Base de Datos MySQL.
  RF2. 2 Cerrar conexión en el recurso Servidor de Base de Datos MySQL.
  RF2. 3 Probar conexión en el recurso Servidor de Base de Datos MySQL.
- **RF3 Gestionar conexión en el recurso Servidor de Base de Datos PostgreSQL.**
  RF3. 1 Abrir conexión en el recurso Servidor de Base de Datos PostgreSQL.
  RF3. 2 Cerrar conexión en el recurso Servidor de Base de Datos PostgreSQL.
  RF3. 3 Probar conexión en el recurso Servidor de Base de Datos PostgreSQL.
- **RF4 Gestionar conexión en el recurso Servicio de Directorio Activo.**
  RF4. 1 Abrir conexión en el recurso Servicio de Directorio Activo.
  RF4. 2 Cerrar conexión en el recurso Servicio de Directorio Activo.
  RF4. 3 Probar conexión en el recurso Servicio de Directorio Activo.
- **RF5 Gestionar conexión en el recurso Servicio de Directorio OpenLDAP.**
  RF5. 1 Abrir conexión en el recurso Servicio de Directorio OpenLDAP.
  RF5. 2 Cerrar conexión en el recurso Servicio de Directorio OpenLDAP.
  RF5. 3 Probar conexión en el recurso Servicio de Directorio OpenLDAP.
- **RF6 Gestionar conexión en el recurso Servidor de Correo Zimbra.**
  RF6. 1 Abrir conexión en el recurso Servidor de Correo Zimbra.
  RF6. 2 Cerrar conexión en el recurso Servidor de Correo Zimbra.
  RF6. 3 Probar conexión en el recurso Servidor de Correo Zimbra.

- **RF7 Gestionar conexión en el recurso Servidor de Mensajería Instantánea Openfire.**
  RF7. 1 Abrir conexión en el recurso Servidor de Mensajería Instantánea Openfire.
  RF7. 2 Cerrar conexión en el recurso Servidor de Mensajería Instantánea Openfire.
  RF7. 3 Probar conexión en el recurso Servidor de Mensajería Instantánea Openfire.
- **RF8 Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos Oracle.**
  RF8. 1 Crear cuenta de usuario en el recurso Servidor de Base de Datos Oracle.
  RF8. 2 Eliminar cuenta de usuario en el recurso Servidor de Base de Datos Oracle.
  RF8. 3Modificar cuenta de usuario en el recurso Servidor de Base de Datos Oracle.
  RF8. 4 Buscar por criterio cuenta de usuario en el recurso Servidor de Base de Datos Oracle.
  RF8. 5 Bloquear cuenta de usuario en el recurso Servidor de Base de Datos Oracle.
  RF8. 6 Desbloquear cuenta de usuario en el recurso Servidor de Base de Datos Oracle.
- **RF9 Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos MySQL.**
  RF9. 1 Crear cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
  RF9. 2 Eliminar cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
  RF9. 3 Modificar cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
  RF9. 4 Buscar por criterio cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
  RF9. 5 Bloquear cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
  RF9. 6 Desbloquear cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
- **RF10 Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos PostgreSQL.**
  RF10. 1 Crear cuenta de usuario en el recurso Servidor de Base de Datos PostgreSQL.
  RF10. 2 Eliminar cuenta de usuario en el recurso Servidor de Base de Datos PostgreSQL.
  RF10. 3 Modificar cuenta de usuario en el recurso Servidor de Base de Datos PostgreSQL.
  RF10. 4 Buscar por criterio cuenta de usuario en el recurso Servidor de Base de Datos PostgreSQL.
  RF10. 5 Bloquear cuenta de usuario en el recurso Servidor de Base de Datos PostgreSQL.
  RF10. 6 Desbloquear cuenta de usuario en el recurso Servidor de Base de Datos PostgreSQL.
- **RF11 Gestionar cuentas de usuarios en el recurso Servicio de Directorio Activo.**
  RF11. 1 Crear cuenta de usuario en el recurso Servicio de Directorio Activo.
  RF11. 2 Eliminar cuenta de usuario en el recurso Servicio de Directorio Activo.
  RF11. 3 Modificar cuenta de usuario en el recurso Servicio de Directorio Activo.
  RF11. 4 Bloquear cuenta de usuario en el recurso Servicio de Directorio Activo.

RF11. 5 Desbloquear cuenta de usuario en el recurso Servicio de Directorio Activo.

- **RF12 Gestionar cuentas de usuarios en el recurso Servicio de Directorio OpenLDAP.**

  RF12. 1 Crear cuenta de usuario en el recurso Servicio de Directorio OpenLDAP.

  RF12. 2 Eliminar cuenta de usuario en el recurso Servicio de Directorio OpenLDAP.

  RF12. 3 Modificar cuenta de usuario en el recurso Servicio de Directorio OpenLDAP.

  RF12. 4 Bloquear cuenta de usuario en el recurso Servicio de Directorio OpenLDAP.

  RF12. 5 Desbloquear cuenta de usuario en el recurso Servicio de Directorio OpenLDAP.

- **RF13 Gestionar cuentas de usuarios en el recurso Servidor de Correo Zimbra.**

  RF13. 1 Crear cuenta de usuario en el recurso Servidor de Correo Zimbra.

  RF13. 2 Eliminar cuenta de usuario en el recurso Servidor de Correo Zimbra.

  RF13. 3 Modificar cuenta de usuario en el recurso Servidor de Correo Zimbra.

  RF13. 4 Buscar por criterio cuenta de usuario en el recurso Servidor de Correo Zimbra.

  RF13. 5 Bloquear cuenta de usuario en el recurso Servidor de Correo Zimbra.

  RF13. 6 Desbloquear cuenta de usuario en el recurso Servidor de Correo Zimbra.

- **RF14 Gestionar cuentas de usuarios en el recurso Servidor de Mensajería Instantánea Openfire.**

  RF14. 1 Crear cuenta de usuario en el recurso Servidor de Mensajería Instantánea Openfire.

  RF14. 2 Eliminar cuenta de usuario en el recurso Servidor de Mensajería Instantánea Openfire.

  RF14. 3 Modificar cuenta de usuario en el recurso Servidor de Mensajería Instantánea Openfire.

  RF14. 4 Buscar por criterio cuenta de usuario en el recurso Servidor de Mensajería Instantánea Openfire.

  RF14. 5 Bloquear cuenta de usuario en el recurso Servidor de Mensajería Instantánea Openfire.

  RF14. 6 Desbloquear cuenta de usuario en el recurso Servidor de Mensajería Instantánea Openfire.

- **RF15 Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos Oracle.**

  RF15. 1 Asignar permiso a los usuarios en el recurso Servidor de Base de Datos Oracle.

  RF15. 2 Eliminar permiso a los usuarios en el recurso Servidor de Base de Datos Oracle.

  RF15. 3 Listar permisos a los usuarios en el recurso Servidor de Base de Datos Oracle.

- **RF16 Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos MySQL.**

  RF16. 1 Asignar permiso a los usuarios en el recurso Servidor de Base de Datos MySQL.

  RF16. 2 Eliminar permiso a los usuarios en el recurso Servidor de Base de Datos MySQL.

  RF16. 3 Listar permisos a los usuarios en el recurso Servidor de Base de Datos MySQL.

- **RF17 Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos PostgreSQL.**

RF17. 1 Asignar permiso a los usuarios en el recurso Servidor de Base de Datos PostgreSQL.

RF17. 2 Eliminar permiso a los usuarios en el recurso Servidor de Base de Datos PostgreSQL.

RF17. 3 Listar permisos a los usuarios en el recurso Servidor de Base de Datos PostgreSQL.

- **RF18 Gestionar permisos a los usuarios en el recurso Servicio de Directorio Activo.**

  RF18. 1 Asignar permiso a los usuarios en el recurso Servicio de Directorio Activo.

  RF18. 2 Eliminar permiso a los usuarios en el recurso Servicio de Directorio Activo.

  RF18. 3 Listar permisos a los usuarios en el recurso Servicio de Directorio Activo.

- **RF19 Gestionar permisos a los usuarios en el recurso Servicio de Directorio OpenLDAP.**

  RF19. 1 Asignar permiso a los usuarios en el recurso Servicio de Directorio OpenLDAP.

  RF19. 2 Eliminar permiso a los usuarios en el recurso Servicio de Directorio OpenLDAP.

  RF19. 3 Listar permisos a los usuarios en el recurso Servicio de Directorio OpenLDAP.

- **RF20 Gestionar permisos a los usuarios en el recurso Servidor de Correo Zimbra.**

  RF20. 1 Asignar permiso a los usuarios en el recurso Servidor de Correo Zimbra.

  RF20. 2 Eliminar permiso a los usuarios en el recurso Servidor de Correo Zimbra.

  RF20. 3 Listar permisos a los usuarios en el recurso Servidor de Correo Zimbra.

- **RF21 Gestionar permisos a los usuarios en el recurso Servidor de Correo Zimbra.**

  RF21. 1 Asignar permiso a los usuarios en el recurso Servidor de Mensajería Instantánea Openfire.

  RF21. 2 Eliminar permiso a los usuarios en el recurso Servidor de Mensajería Instantánea Openfire.

  RF21. 3 Listar permisos a los usuarios en el recurso Servidor de Mensajería Instantánea Openfire.

Requerimientos no funcionales

Diseño e Implementación

- El paquete debe contar con un conjunto de patrones de diseño que permitan la reutilización del código.
- Debe permitir al sistema que utilice el paquete de conectores que pueda deshacer las acciones realizadas en caso de ocurrencia de errores de conexión.
- El paquete de conectores debe permitir la adición de nuevos conectores sin tener que realizar cambios en la interfaz principal.
- El código debe cumplir con los estándares de codificación establecidos por el equipo de desarrollo.
- Lenguaje de programación: *Python* 2.7.X.

2.5 Historias de usuario (HU)

Las historias de usuario (HU) representan las funcionalidades que el cliente desea que estén presentes en el sistema; por lo tanto, todo el trabajo futuro debe girar en torno a satisfacer estas expectativas (25). A continuación se presentan algunas de las historias de usuario desarrolladas las cuales están relacionadas con el recurso Servidor de Base de Datos MySQL. El resto de las historias de usuario se encuentran descritas en el **Anexo 2**.

Historia de usuario	
Número: HU2	**Usuario:** -
Nombre historia: Gestionar conexión en el recurso Servidor de Base de Datos MySQL.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**1
Programador responsable: Alfredo Gonzales Fraxedas, Marisol Viera Figueroa	
Descripción: Se encarga de gestionar los parámetros de abrir conexión, además de cerrar y probar la conexión si esta estaba abierta. En caso de fallar la conexión se notifica el error.	
Observaciones:	

Tabla 2: Descripción de la HU Gestionar conexión en el recurso Servidor de Base de Datos MySQL.

Fuente: Elaboración propia.

Historia de usuario	
Número: HU9	**Usuario:** -
Nombre historia: Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos MySQL.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**1
Programador responsable: Alfredo Gonzales Fraxedas, Marisol Viera Figueroa	
Descripción: Se encarga de crear, modificar, buscar por criterio, eliminar, bloquear y desbloquear las cuentas de los usuarios. En caso de fallar el procedimiento se notifica el error.	
Observaciones:	

Tabla 3: Descripción de la HU Gestionar cuenta de usuario en el recurso Servidor de Base de Datos MySQL.

Fuente: Elaboración propia.

Historia de usuario	
Número: HU16	**Usuario:** -
Nombre historia: Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos MySQL.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados:1	**Iteración asignada:**1
Programador responsable: Alfredo Gonzales Fraxedas, Marisol Viera Figueroa	
Descripción: Se encarga de asignar, eliminar y listar los permisos a un usuario determinado. En caso de fallar el procedimiento se notifica el error.	
Observaciones:	

Tabla 3: Descripción de la HU Gestionar permisos en el recurso Servidor de Base de Datos MySQL.

Fuente: Elaboración propia.

2.6 Fase de planificación

En la metodología XP, la planificación se plantea como un diálogo continuo entre las partes involucradas en el proyecto. Es una fase corta, donde el cliente y el grupo de trabajo acuerdan el orden en que deberán implementarse las historias de usuario, así como también su entrega. Esta planificación debe de ser flexible para que pueda adaptarse a los posibles cambios que puedan surgir (38).

Plan de iteraciones

Después de ser identificadas y descritas las HU, se procede a la planificación de la fase de implementación donde se establece una división de tres iteraciones.

Iteración 1: Se realizarán las historias de usuario que presentan prioridad alta: se codificarán todas las historias de usuario relacionadas con en el recurso Servidor de Base de Datos Oracle, recurso Servidor de Base de Datos MySQL y recurso Servidor de Base de Datos PostgreSQL.

Iteración 2: Se realizarán las historias de usuario que presentan prioridad media: se codificarán todas las historias de usuario relacionadas con el recurso Servicio de Directorio OpenLDAP, el recurso Servicio de Directorio Activo y en el recurso Servidor de Correo Zimbra.

Iteración 3: Se realizarán las historias de usuario que presentan prioridad baja: se codificarán todas las historias de usuario relacionadas con el recurso Servidor de Mensajería Instantánea Openfire.

Plan de duración de las iteraciones

El plan de duración de las iteraciones se encarga de mostrar las HU en el orden en que se implementarán en cada iteración así como la duración estimada de las mismas. Las estimaciones de esfuerzo asociado a la implementación de las historias de usuario la establecen los programadores y se utiliza como medida el punto. Un punto, equivale a una semana ideal de programación, que es la cantidad de trabajo que se puede realizar durante una semana sin distracciones. El tiempo de implementación de una historia de usuario generalmente es de uno a tres puntos (39).

Para realizar la estimación el principal factor que se tuvo en cuenta fue la experiencia media del equipo de desarrollo con el lenguaje de programación y el conocimiento sobre los diferentes conectores. La siguiente tabla muestra el plan de duración de las iteraciones que permite el desarrollo del paquete de conectores.

Iteración	Historias de usuarios	Duración estimada(semanas)
Iteración 1	✓ **HU1** Gestionar conexión en el recurso Servidor de Base de Datos Oracle. ✓ **HU8** Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos Oracle. ✓ **HU15** Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos Oracle.	2
	✓ **HU2** Gestionar conexión en el recurso Servidor de Base de Datos MySQL. ✓ **HU9** Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos MySQL. ✓ **HU16** Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos MySQL.	2
	✓ **HU3** Gestionar conexión en el recurso Servidor de Base de Datos PostgreSQL. ✓ **HU10** Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos PostgreSQL. ✓ **HU17** Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos PostgreSQL.	2
Iteración 2	✓ **HU4** Gestionar conexión en el recurso Servicio de Directorio Activo.	3

	✓ **HU11** Gestionar cuentas de usuarios en el recurso Servicio de Directorio Activo. ✓ **HU18** Gestionar permisos a los usuarios en el recurso Servicio de Directorio Activo.	
	✓ **HU5** Gestionar conexión en el recurso Servicio de Directorio OpenLDAP. ✓ **HU12** Gestionar cuentas de usuarios en el recurso Servicio de Directorio OpenLDAP. ✓ **HU19** Gestionar permisos a los usuarios en el recurso Servicio de Directorio OpenLDAP.	3
	✓ **HU6** Gestionar conexión en el recurso Servidor de Correo Zimbra. ✓ **HU13** Gestionar cuentas de usuarios en el recurso Servidor de Correo Zimbra. ✓ **HU20** Gestionar permisos a los usuarios en el recurso Servidor de Correo Zimbra.	3
Iteración 3	✓ **HU7** Gestionar conexión en el recurso Servidor de Mensajería Instantánea Openfire. ✓ **HU14** Gestionar cuentas de usuarios en el recurso Servidor de Mensajería Instantánea Openfire. ✓ **HU21** Gestionar permisos a los usuarios en el recurso Servidor de Mensajería Instantánea Openfire.	3

Tabla 4: Plan de duración de las iteraciones.

Fuente: Elaboración propia.

La Iteración 1 durará en total 1 mes y 2 semanas.

La Iteración 2 durará en total 2 meses y 1 semana.

La iteración 3 durará en total 3 semanas.

Plan de entregas

Una vez que se concluye la tarea por parte del cliente de elaborar las distintas HU, se comienza con la creación del plan de entrega, para estimar el tiempo de desarrollo de las mismas. Este artefacto se elabora con la intención de fijar qué período de tiempo puede tardar la implementación de cada una de las historias en cada iteración, definiéndose las fechas en que serán liberadas las versiones funcionales del producto (38). El plan

de entrega del paquete de conectores para la interacción de servidores de aplicaciones y servicios es el siguiente:

1ra Entrega (iteración 1)	2da Entrega (iteración 2)	3ra Entrega (iteración 3)
16 de febrero de 2015	24 de abril de 2015	16 de mayo de 2015

Tabla 5: Plan de entregas.

Fuente: Elaboración propia.

2.7 Fase de diseño

Para guiar la metodología XP debe aparecer un diseño que describa qué clases hay y cómo interactúan. Se indica cómo se construirá la solución mediante la definición de una estructura lo más sencilla posible para satisfacer los requisitos funcionales y no funcionales. La metodología XP no requiere la representación del sistema mediante diagramas de clases; en su lugar utiliza una técnica para representar clases, mediante las tarjetas Clase-Responsabilidad-Colaboración (CRC) (25). (Ver **Anexo 3**)

Nombre de la clase: MySQLConector	
Responsabilidades	**Colaboradores**
1. open_connection 2. test_connection 3. close_connection 4. create_cursor 5. get_all_databases_info 6. get_all_tables_info 7. get_table_info 8. get_column_info 9. get_all_users_info 10. get_users_info_by_criteria 11. get_user_info 12. create_user 13. undo_create_user 14. delete_user 15. undo_delete_user	MySQLdb

16. modify_user 17. lock_user 18. unlock_user 19. grant 20. grant_all_privileges 21. grant_privilege_on_table 22. revoke 23. revoke_all_privileges 24. revoke_privilege_on_table 25. list_priveleges 26. execute_query	

Tabla 6: Tarjeta CRC correspondiente a la clase MySQL Conector.

Fuente: Elaboración propia.

2.8 Diagrama de clases del diseño

Después de definidas las tarjetas CRC, donde se identificaron las clases y sus principales responsabilidades, se está en condiciones de confeccionar el diagrama de clases del diseño (Ver **Anexo 4**). Este diagrama aunque no es un artefacto propio de XP, permitirá representar gráficamente las clases (atributos y métodos) y las relaciones entre clases (dependencias y asociaciones) lo cual permitirá un mejor entendimiento del funcionamiento del paquete de conectores.

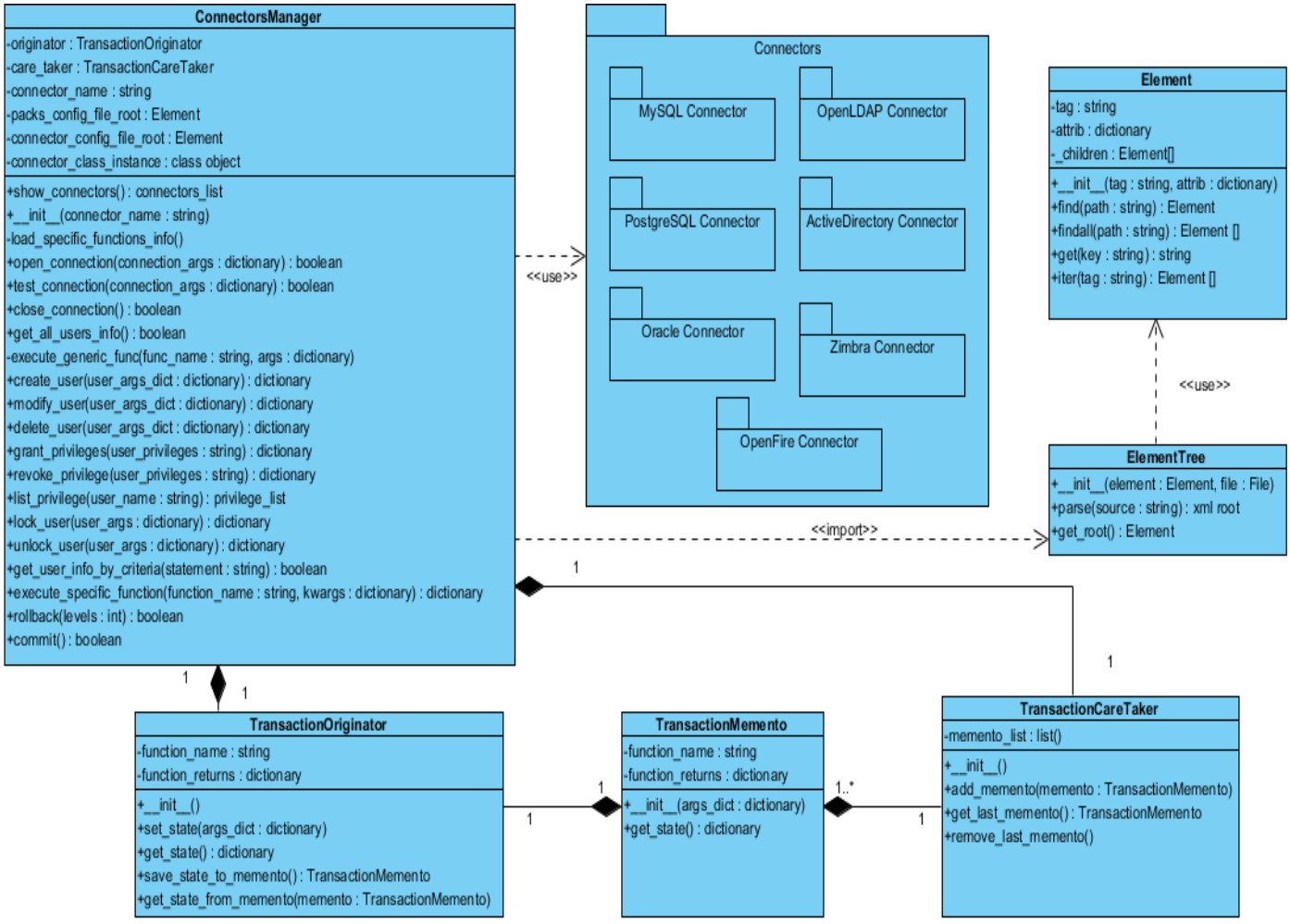


Figura 3: Diagrama de clases del paquete de conectores.

Fuente: Elaboración propia.

2.9 Especificación de la arquitectura

La arquitectura es el diseño de más alto nivel de la estructura de un sistema informático. Permite obtener una vista del sistema, y es en gran medida responsable de permitir o no ciertos atributos de calidad del sistema entre los que se destacan la confiabilidad y el rendimiento del *software*. Establece los fundamentos para que analistas, diseñadores, programadores y otros roles, trabajen en una línea común que permita alcanzar los objetivos del sistema y cubra todas las necesidades (40).

Para el desarrollo del paquete de conectores se utilizó el patrón arquitectónico n-capas (ver **figura 3)**; el cual permite la distribución jerárquica de los roles y responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican la forma de interacción con otras capas y las responsabilidades la funcionalidad que implementa. Garantiza el desarrollo paralelo en cada capa y permite aumentar la flexibilidad porque cada capa puede ser manejada y escalada de forma independiente. Debido a que cada capa es independiente de la otra los cambios o actualizaciones pueden ser realizados sin afectar la aplicación como un todo (41).

El paquete de conectores estará integrado a una aplicación que utilizará los conectores para realizar funciones de aprovisionamiento. Los conectores se utilizan en la capa Lógica de Negocio. Esta capa recibe información (datos de las cuentas de usuarios) para realizar operaciones de aprovisionamiento, a través de la capa Interfaz de Programación; la capa de Acceso a Datos interactúa directamente con los recursos a través de las API, las cuales brindan las funcionalidades necesarias para acceder y modificar la información en estos recursos.

La capa Lógica de Negocio se relaciona con esta de tal forma que cada conector está relacionado con la API que le permitirá acceder al recurso para el cual está concebido. Finalmente son invocadas las funcionalidades de los conectores a partir de los datos provenientes de las diferentes capas. Con el objetivo de exponer claramente el funcionamiento de la propuesta arquitectónica para el desarrollo del paquete de conectores, se presenta el siguiente diagrama que muestra la estructura de la arquitectura:

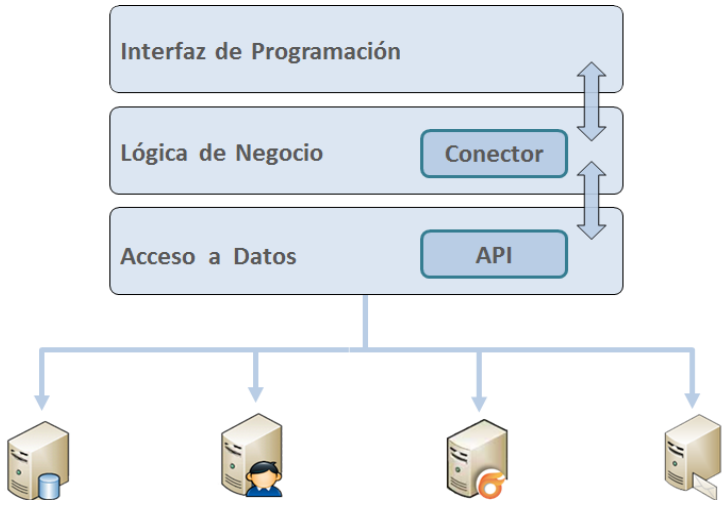


Figura 4: Arquitectura del paquete de conectores.

Fuente: Elaboración propia.

2.10 Patrones de diseño utilizados

Los patrones de diseño son soluciones para problemas típicos y recurrentes que no se pueden encontrar en el momento de desarrollar una aplicación. Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces (42).

Patrones GRASP

En el diseño de la aplicación se utilizaron los patrones GRASP que son patrones generales de *software* para asignar responsabilidades (en inglés *General Responsibility Asignment Software Patterns*). Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (43). Dentro de los patrones de diseño GRASP se utilizó:

- ✓ Experto en información: Se utiliza porque conserva el encapsulamiento, pues los objetos se valen de su propia información para realizar la tarea encomendada. Soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento (42). Además, el comportamiento se distribuye entre las clases que cuentan con la información requerida, haciéndolas más fáciles de comprender y de responsabilidades mantener, lo cual garantiza una alta cohesión. Este patrón se utiliza en todas las clases del paquete de conectores.
- ✓ Creador: El patrón Creador guía la asignación de relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos (42). El patrón se pone de manifiesto en la clase Connectors_Manager a la hora de crear varias instancias de clases que se relacionan con ella.
- ✓ Alta cohesión: Expresa que la información que almacena una clase debe de ser coherente y debe estar relacionada con la clase en la medida de lo posible (42). La información que manipula cada clase está estrechamente relacionada con la funcionalidad que realiza la misma como componente del paquete de conectores.

Patrones GOF

Los patrones GOF (en inglés *Gang of Four*) describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad (44).

- ➢ **Patrones de comportamiento:** Definen la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos (45).

- ✓ *Memento* (Momento): Tiene la finalidad de almacenar el estado de un objeto (o del sistema completo) en un momento dado de manera que se pueda restaurar en ese punto de manera sencilla (42) . Se utiliza para guardar las acciones realizadas, para que en caso de fallos, se puedan deshacer, y así lograr que el paquete de conectores permita la implementación de un mecanismo transaccional.

  Ejemplo:

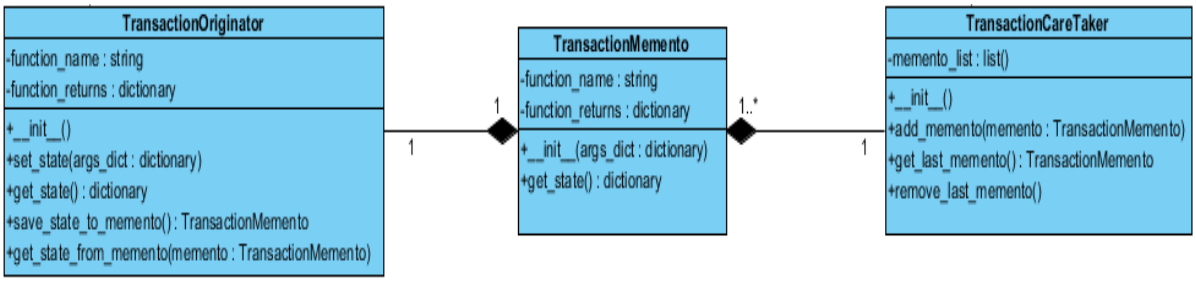


Figura 5: Patrón memento

Fuente: Elaboración propia.

- ➢ **Patrones estructurales:** Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades (42).
  - ✓ *Facade* (Fachada): Define una interfaz de alto nivel para permitir que un subsistema sea más fácil de utilizar (42). En el paquete de conectores se implementó un módulo con la clase ConnectorsManager, que contiene un grupo de funciones relacionadas con las funcionalidades de los conectores. Lo cual permite al sistema que utilicé la solución el acceso a las funcionalidades de manera unificada y sencilla, para reducir la complejidad y minimizar las dependencias en el sistema.

Ejemplo:

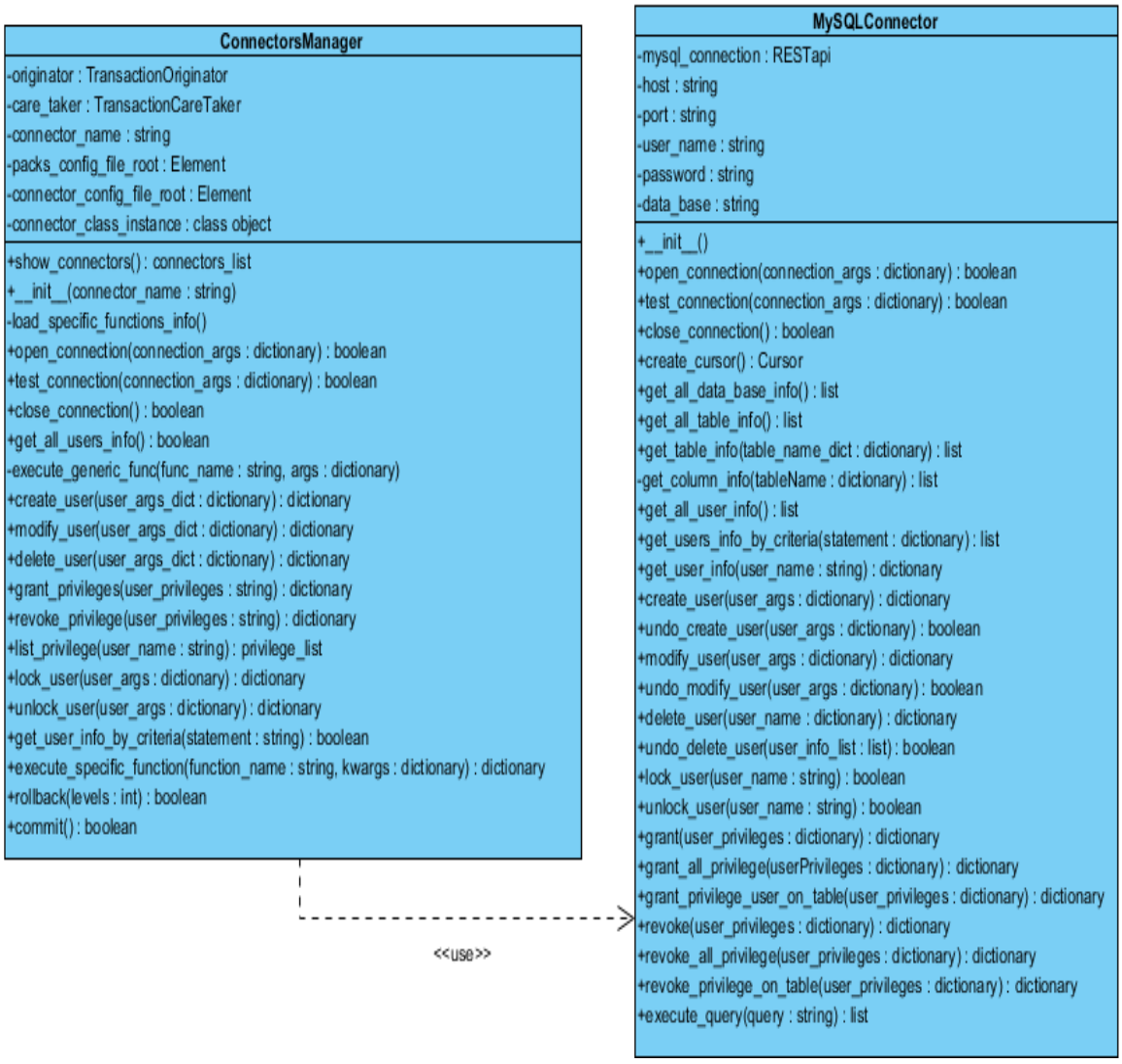


Figura 6: Ejemplo del uso del patrón fachada.

Fuente: Elaboración propia.

2.11 Conclusiones parciales

Luego de definir las características que debe cumplir el paquete de conectores y de realizar el análisis y diseño correspondiente, se concluye que durante el desarrollo del capítulo se definió el modelo de dominio que ayudó a identificar los requisitos funcionales y no funcionales descritos en las historias de usuario. Se realizó una estimación del tiempo utilizado por los programadores para implementar las historias de usuario, lo cual permitió planificar dicha implementación en 3 iteraciones. El plan de iteraciones sirvió de conocimiento para calcular el tiempo estimado de desarrollo.

El estudio realizado en este capítulo ha facilitado un entendimiento de la dinámica y estructura del paquete de conectores, y contribuyó a una mejor comprensión del problema que la solución debe resolver para ganar claridad en cuanto a su concepción, además se sentaron las bases para las restantes fases del proceso.

Capítulo 3: Implementación y Pruebas

Luego de realizar el diseño de la solución propuesta y de definir su arquitectura, se procede a implementar el paquete de conectores que cumpla con los requisitos que se necesitan. El presente capítulo está dedicado a detallar las tres iteraciones llevadas a cabo durante la etapa de construcción de la solución, para mostrar los principales artefactos de la implementación de la misma. Se describen las pruebas realizadas al *software* obtenido para determinar el nivel de calidad que posee al efectuar pruebas unitarias y de aceptación, las cuales permiten conocer cuántas de las especificaciones planteadas fueron cumplidas en su totalidad y sin errores.

3.1 Estándar de codificación

Establecer un estándar de codificación que sea aceptado por todo el equipo de desarrollo es muy importante en una metodología como XP, ya que este representa pautas o reglas creadas para escribir el código en la programación. Usar estas reglas implica obtener un código de alta calidad que juega un papel importante en la calidad del *software* y en su rendimiento. Además, la utilización de forma eficiente de estos estándares de codificación permite una mejor comprensión del código por otras personas (26).

Se utilizó el estándar de codificación de *Python* (46) que a su vez es utilizado por el Sistema de Aprovisionamiento de Usuarios, lo cual permite que al integrar el paquete de conectores, se mantenga una misma línea estética con respecto a la legibilidad del código.

3.2 Tareas de ingeniería

XP plantea que la implementación de un *software* se realiza iterativamente, y permite obtener al culminar cada iteración un producto funcional, que debe ser probado y mostrado al cliente. Durante el transcurso de las iteraciones, se realiza la implementación de las HU definidas por el cliente y descritas por el equipo de desarrollo en la etapa de planificación (25). Como parte de este plan, se descomponen estas HU en tareas de ingeniería, las cuales son asignadas a los programadores para ser implementadas durante la iteración correspondiente.

En la tabla se describen las tareas correspondientes a las historias de usuario en la primera iteración del proceso de desarrollo del paquete de conectores. Las tareas de las otras iteraciones se encuentran en los anexos (ver **Anexo 5**).

<table>
<tr><th colspan="2">Iteración 1</th></tr>
<tr><th>Historia de usuario</th><th>Tareas</th></tr>
<tr><td>✓ HU1 Gestionar conexión en el recurso Servidor de Base de Datos Oracle.
✓ HU8 Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos Oracle.
✓ HU15 Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos Oracle.</td><td rowspan="2">Gestionar conexión.
Abrir conexión
✓ Obtener una API o librería que permita la conexión con el servidor.
✓ Validar que no falte ningún parámetro de conexión.
✓ Ejecutar la función de conexión con el servidor.

Cerrar conexión.
✓ Comprobar que existe una conexión activa.
✓ Cerrar la conexión.

Probar conexión.
✓ Comprobar que no falten parámetros en la conexión.
✓ Ejecutar la función de conexión y en caso de que exista un error, lanzar una excepción.

Gestionar cuentas de usuario.
✓ Comprobar que se poseen los parámetros correctos.
✓ Ejecutar las funciones necesarias para crear, eliminar, modificar o buscar por criterio una cuenta en el servidor.
✓ Si la operación es exitosa devolver los datos que fueron creados o eliminados, en el servidor para posteriormente poder deshacer esa operación. En el caso de los datos</td></tr>
<tr><td>✓ HU2 Gestionar conexión en el recurso Servidor de Base de Datos MySQL.
✓ HU9 Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos MySQL.
✓ HU16 Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos MySQL.</td></tr>
</table>

	modificados se debe devolver los datos previos a la modificación. ✓ En el caso que se desee buscar una cuenta y la información del usuario no aparezca en el servidor se debe lanzar una excepción para informar el error. **Gestionar permisos** ✓ Comprobar que se poseen los parámetros correctos. ✓ Asignar o eliminar los permisos a los usuarios en las tablas definidas. Mostrar para un usuario que permisos tiene.
✓ **HU3** Gestionar conexión en el recurso Servidor de Base de Datos PostgreSQL. ✓ **HU10** Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos PostgreSQL. ✓ **HU17** Gestionar permisos a los usuarios en el recurso Servidor de Base de Datos PostgreSQL.	

Tabla 7: Tareas de ingeniería en la primera iteración.

Fuente: Elaboración propia.

3.3 Diagrama de componentes

Para un mejor entendimiento del funcionamiento y relación entre los archivos del sistema desarrollado se realiza el diagrama de componentes, se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes, donde un componente se corresponde con una o varias clases, interfaces o colaboraciones (45).

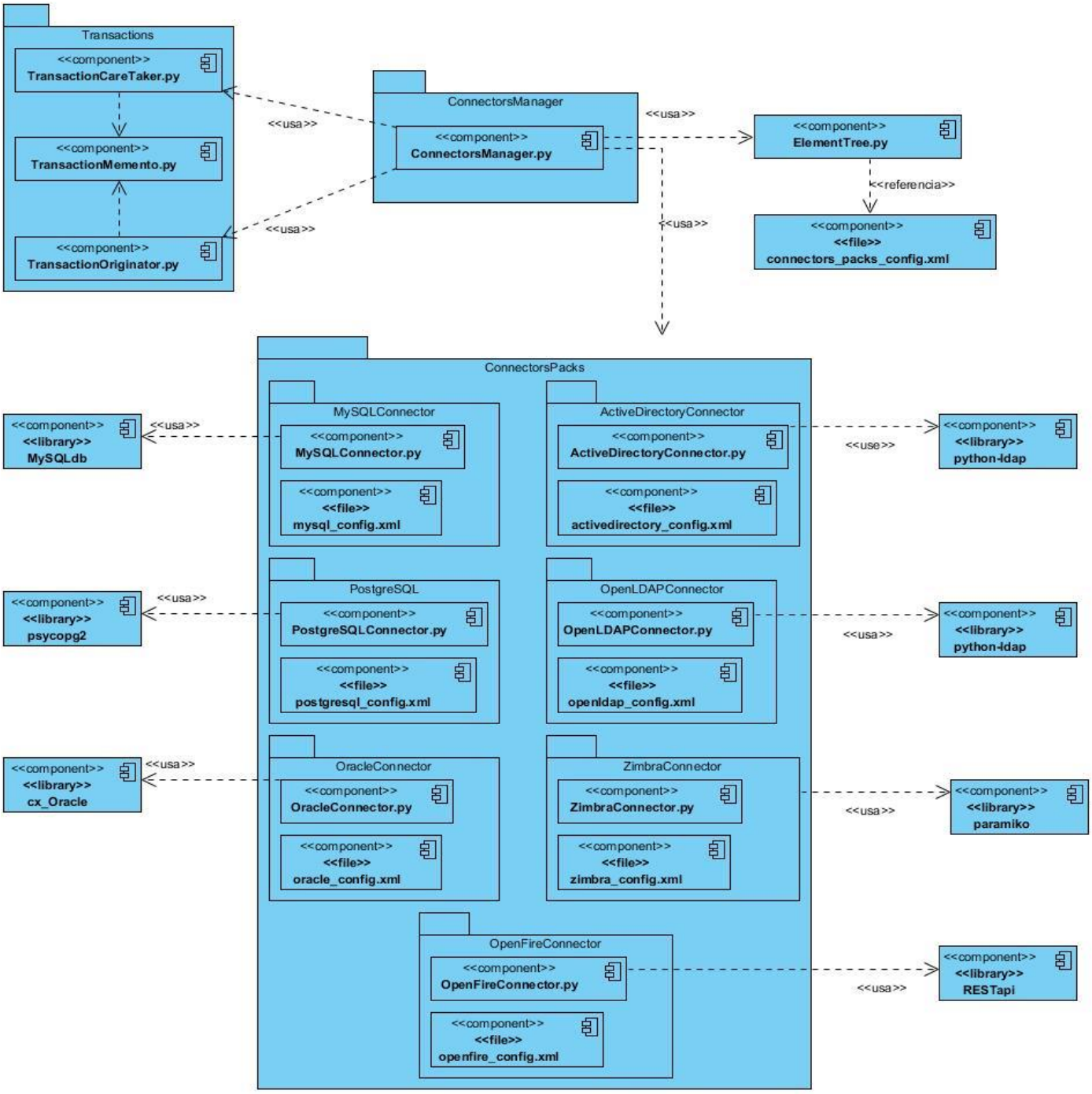


Figura 7: Diagrama de componentes.

Fuente: Elaboración propia.

Descripción de componentes:

El paquete principal es *ConectorsManager*, el cual contiene:

- El componente *ConnectorsManager.py*, el cual contiene las funcionalidades principales del sistema.
- El componente *ElementTree.py*, responsable de acceder a los ficheros XML.
- El fichero de configuración principal *connectors_packs_config.xml,* que es el componente que cuenta con la ubicación del paquete de cada conector en el paquete *ConnectorPacks*, así como la información general de cada uno de los conectores.

El paquete *Transaction* es el encargado de permitir al paquete de conectores brindar la funcionalidad de deshacer acciones en caso de ocurrencia de errores. *ConnectorsManager.py* se relaciona con *TransactionCareTaker.py* y con *TransactionOriginator.py* para de esta forma poder utilizar las funcionalidades que ellos les brindan.

- El componente *TransactionCareTaker.py* está encargado de guardar los objetos *memento*, por lo que es quien guarda las acciones que se van generando.
- El componente *TransactionOriginator.py* se utiliza para hacer seguimiento del estado de una acción y crear objetos *memento*.
- El componente *TransactionMemento.py* guarda el estado de una acción.

El paquete *ConnectorPacks* contiene varios paquetes, cada uno relacionado con un conector específico.

- El paquete de cada conector contiene un componente principal con las funciones del conector y un fichero de configuración XML que define la estructura del conector.
- Cada conector está relacionado con una librería o API externa al *ConnectorPacks* que le permite al conector interactuar con su respectivo servidor.

3.4 Diagrama de despliegue

El diagrama de despliegue describe la distribución física del paquete de conectores. En él se grafican los nodos y las conexiones necesarias para el funcionamiento exitoso del *software*. Muestra también dónde se localizan los componentes del *software*, o sea, en qué servidores (22).

El siguiente diagrama está compuesto por 8 nodos, *ConectorsManager* va a contener dos componentes, uno para el XML y otro para las API que van a utilizar los conectores (Oracle, MySQL, PostgreSQL, OpenLDAP, Directorio Activo, Zimbra, y Openfire), y la plataforma que se utiliza es *Python* en su versión 2.7.X. Los restantes

nodos representan los servidores que se van a utilizar para cada uno de los conectores del sistema. La interacción entre estos nodos se realiza a través de diferentes protocolos tales como: postgresql, mysql, oracle y LDAP, los cuales, para comunicarse con el servidor, están encapsulados en el protocolo TCP/IP[9]. Los restantes protocolos que se utilizan son SSH[10] y HTTP[11].

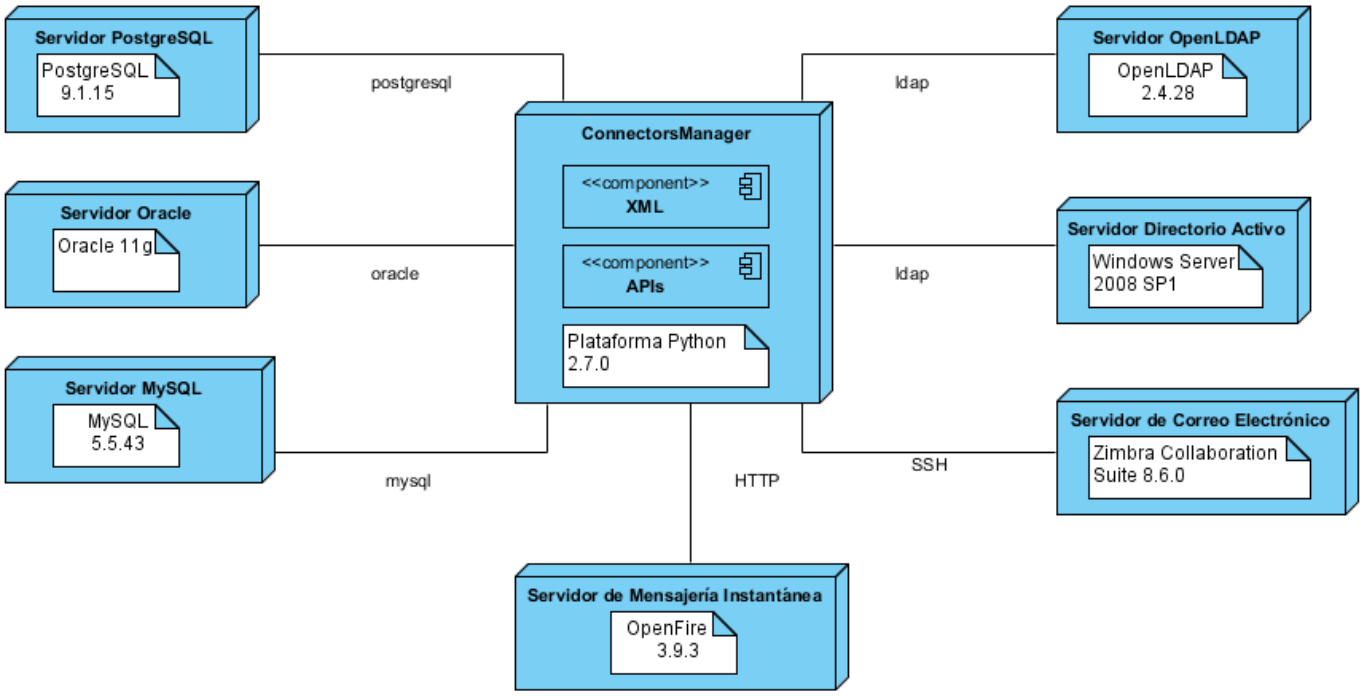


Figura 8: Diagrama de despliegue.

Fuente: Elaboración propia.

3.5 Diseño de los casos de prueba

Una vez generado el paquete de conectores, el *software* debe ser probado para revelar y corregir el máximo de errores posible antes de su entrega al cliente, haciéndolo con la menor cantidad de tiempo y esfuerzo

[9] **TCP/IP** son las siglas del Protocolo de Control de Transmisión/Protocolo de Internet (en inglés *Transmission Control Protocol/Internet Protocol*),

[10] **SSH** son las siglas del protocolo intérprete de órdenes segura (en inglés Secure Shell)

[11] **HTTP** son las siglas del protocolo de transferencia de hipertexto(en inglés Hypertext Transfer Protocol)

posible. Como objetivo fundamental de las pruebas, según *Pressman*, se tiene el siguiente: descubrir la máxima cantidad de errores no detectados hasta entonces (22). Aun así, es necesario señalar que las pruebas no pueden asegurar la ausencia de errores, solo puede demostrar que existen defectos en el *software*. La programación extrema define dos tipos de pruebas las cuales son: pruebas unitarias y pruebas de aceptación.

- ✓ **Pruebas unitarias**: desarrolladas por los programadores y encargadas de verificar el código de forma automática (47).
- ✓ **Pruebas de aceptación:** destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente (38).

Las pruebas unitarias se establecen antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema, se basa en hacer pruebas en pequeños fragmentos al código, encargados de una tarea específica. Esta prueba, conocida también como Caja Blanca, permite asegurar que cada módulo implementado funcione adecuadamente. El objetivo de realizar este tipo de prueba, es demostrar realizándole inspección al código de que no contiene errores, y en caso de contener proceder a su pronta eliminación (25).

Para realizar estas pruebas fue utilizado el IDE *PyCharm* 3.4 el cual permite la realización de pruebas de unidad a los métodos. Se utilizó la herramienta *VMWare Workstation* 8.04 donde se crearon varias máquinas virtuales, para instalar *Ubuntu 12.04* para probar los conectores MySQL, PostgreSQL y OpenLDAP, *Windows 7 Professional 32 bits* para realizar las pruebas al conector Oracle, *Windows Server 2008* para probar el conector de Active Directory, *Ubuntu Server 12.04 LTS* para realizar las pruebas al conector Zimbra y *Windows 8 Pro 64 bits* sobre la máquina virtual de java en su versión 8 para probar el conector de OpenFire. Después de realizada una primera iteración de pruebas unitarias fueron encontradas no conformidades en el código las cuales se muestran a continuación:

Clase:	OracleConnector, MySQLConnector y PostgreSQLConnector
Áreas:	Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos Oracle. Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos MySQL. Gestionar cuentas de usuarios en el recurso Servidor de Base de Datos PostgreSQL.
Fallo:	Después de actualizar la contraseña del usuario durante el proceso de restauración, al conectarse como ese usuario, el proceso fallaba.
Ubicación:	delete_user

Solución:	Ejecutar una cláusula en el Sistema Gestor de Base de Datos cuya función es recargar la configuración de seguridad, para que vuelva a cargar toda la información de seguridad y así reconocer la contraseña actualizada, para que el usuario pueda conectarse con la nueva contraseña de inmediato.
Responsable:	Alfredo Gonzáles Fraxedas
Prioridad:	Alta

Tabla 8: Errores detectados en las clases OracleConnector, MySQLConnector y PostgreSQLConnector.

Fuente: Elaboración propia.

Clase:	OpenFireConnector
Área:	Gestionar permisos a los usuarios en el recurso Servidor de Mensajería Instantánea OpenFire.
Fallo:	No se puede crear usuarios como Administrador porque el plugin que se utiliza no brinda esta funcionalidad, a la hora de asignar los permisos.
Ubicación:	*add_user_to_group*
Solución:	Se van a asignar los permisos de acuerdo a los grupos.
Responsable:	Alfredo Gonzáles Fraxedas
Prioridad:	Alta

Tabla 9: Errores detectados en la clase OpenFireConnector.

Fuente: Elaboración propia.

Clases:	OpenFireConnector y ZimbraConnector
Área:	Gestionar cuentas de usuarios en el recurso Servidor de Mensajería Instantánea OpenFire y Gestionar cuentas de usuarios en el recurso Servidor de Correo Zimbra.
Fallo:	No se puede obtener la contraseña del usuario.
Ubicación:	*delete_user*
Solución:	Tomar la contraseña como el nombre de usuario que se desee eliminar.
Responsable:	Marisol Viera Figueroa
Prioridad:	Alta

Tabla 10: Errores detectados en las clases OpenFireConnector y ZimbraConnector.

Fuente: Elaboración propia.

Clases:	OpenLDAPConnector y ActiveDirectoryConnector
Área:	Gestionar cuentas de usuarios en el recurso Servicio de Directorio LDAP y Gestionar cuentas de usuarios en el recurso Servicio de Directorio Activo.
Fallo:	No hay un atributo que especifique si el usuario está bloqueado.
Ubicación:	*lock_user*
Solución:	Para evitar que un usuario se conecte, se modifica su contraseña corrompiéndola, se puede realizar de esta forma debido a que la contraseña cuenta con una función hash y al quitarle el SHA-1 que tiene delante la contraseña, esta se modifica y es inválida porque el hash va a ser diferente.
Responsable:	Marisol Viera Figueroa
Prioridad:	Alta

Tabla 11: Errores detectados en las clases OpenLDAPConnector y ActiveDirectoryConnector

Fuente: Elaboración propia.

En la segunda iteración de pruebas unitarias se encontraron 4 no conformidades y en la tercera iteración realizada pudo verificarse que los errores o malos funcionamientos habían sido solucionados. No fueron encontrados errores lógicos en la programación lo cual permite asegurar 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 corregidos los errores. Para consultar los restantes casos de prueba ver el **Anexo 6**.

Prueba unitaria		
Nombre de la prueba: *OpenConnection* en MySQL		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Última ejecución:**
Ejecutado por: Marisol Viera Figueroa	**Verificado por:** Alfredo Gonzales Fraxedas	
Descripción: Para poder ejecutar la prueba previamente se deben haber asignado los valores de conexión del recurso solicitado *(host, port, user, password, database)* si los datos son correctos la conexión se realiza de forma satisfactoria, de lo contrario, se lanza un mensaje que muestra el error que se originó.		
Entrada: *host : string, port : string, user : string, password: string*		
Criterio de aceptación: Abrir conexión		

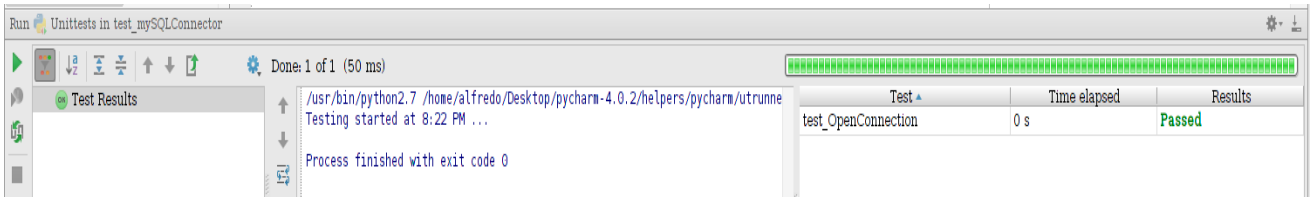
Resultado:


Tabla 12: Prueba unitaria OpenConnection para MySQL.
Fuente: Elaboración propia.

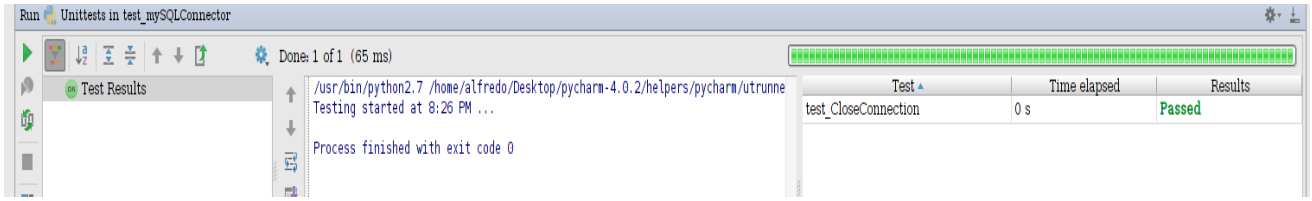
Prueba unitaria		
Nombre de la prueba: *CloseConnection* en MySQL		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Última ejecución:**
Ejecutado por: Marisol Viera Figueroa	**Verificado por:** Alfredo Gonzales Fraxedas	
Descripción: Para la ejecución de la prueba debe de haber alguna conexión abierta de lo contrario si no existe ninguna conexión abierta se lanza un mensaje que muestra el error que se originó.		
Entrada:		
Criterio de aceptación: Cerrar la conexión.		
Resultado: 		

Tabla 13: Prueba unitaria CloseConnection para MySQL.
Fuente: Elaboración propia.

Prueba unitaria		
Nombre de la prueba: *CreateUser* en MySQL		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Última ejecución:**
Ejecutado por: Marisol Viera Figueroa	**Verificado por:** Alfredo Gonzales Fraxedas	
Descripción:		

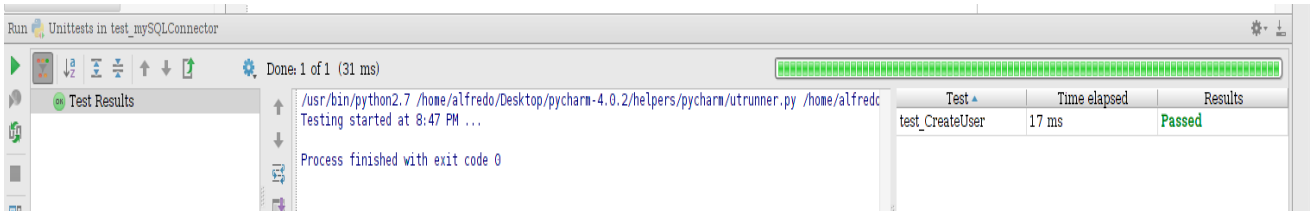
Para la ejecución de la prueba se debe establecer la conexión con el servidor, luego introducir los datos del usuario, si el usuario no existe, la función del método es crear una cuenta de usuario, de lo contrario si el usuario ya se encuentra en el servidor se lanza un mensaje que muestra el error que se originó.
Entrada: *user_info_dict {'user_name': 'valor', 'user_pasword': 'valor'}*
Criterio de aceptación: Crear la cuenta de usuario
Resultado: 

Tabla 14: Prueba unitaria CreateUser para MySQL.

Fuente: Elaboración propia.

Prueba unitaria		
Nombre de la prueba: *DeleteUser* en MySQL		
Estado: Satisfactoria	**Tipo:** Caja Blanca	**Última ejecución:**
Ejecutado por: Marisol Viera Figueroa	**Verificado por:** Alfredo Gonzales Fraxedas	
Descripción: Para la ejecución de la prueba se debe establecer la conexión con el servidor, luego 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 que muestra el error que se originó.		
Entrada: *user_name: string*		
Criterio de aceptación: Elimina la cuenta de usuario.		

Resultado:

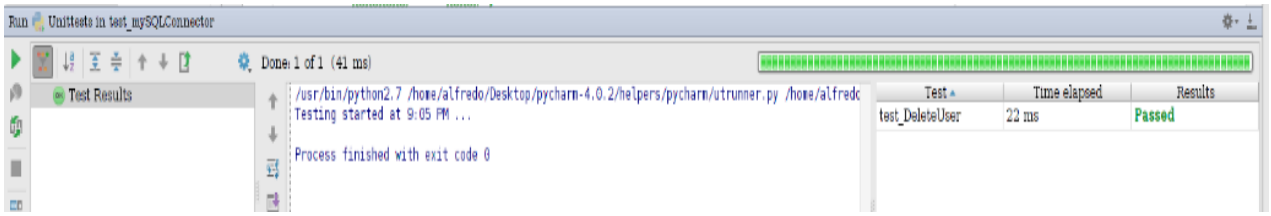


Tabla 15: Prueba unitaria DeleteUser para MySQL.

Fuente: Elaboración propia.

Las pruebas han sido aplicadas a través de los casos de prueba correspondientes a los historias de usuarios, lo cual permite comprobar que los conectores desarrollados funcionan acorde a los requisitos definidos por el cliente. A continuación se muestran algunos casos de pruebas de aceptación realizados a la aplicación. Para consultar los restantes casos de prueba ver el **Anexo 7**.

Caso de prueba de aceptación	
Código:28	**Historia de usuario:** Crear cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
Responsable de la prueba: Marisol Viera Figueroa	
Descripción: Se registran los datos de la cuenta de usuario en el recurso Servidor de Base de Datos MySQL y se realiza la acción de crear la cuenta.	
Condiciones de ejecución: 1. El paquete de conectores debe contar con la librería *MySQLdb.* 2. Debe existir alguna conexión abierta en el recurso Servidor de Base de Datos MySQL.	
Entrada/Pasos de ejecución: 1. Entrada de los datos de la cuenta de usuario: *user_name* y *password.* 2. Crear la cuenta de usuario. 3. Devolver datos del usuario creado para luego poder deshacer la opción.	
Resultado esperado: Mensaje que indique la creación correcta de la cuenta de usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 16: Caso de prueba de funcionalidad crear cuenta de usuario en el recurso Servidor de Base de Datos MySQL.

Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código:30	**Historia de usuario:** Modificar cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
Responsable de la prueba: Marisol Viera Figueroa	
Descripción: Se selecciona el nombre de la cuenta usuario que se desea modificar en el recurso Servidor de Base de Datos MySQL y se realiza la acción de modificar.	
Condiciones de ejecución: 1. El paquete de conectores debe contar con la librería *MySQLdb.* 2. Debe existir alguna conexión abierta en el recurso Servidor de Base de Datos MySQL.	
Entrada/Pasos de ejecución: 1. Entrada de los datos de la cuenta de usuario: *user_name* y *password.* 2. Modificar cuenta de usuario. 3. Devolver datos del usuario modificado para luego poder deshacer la opción.	
Resultado esperado: Mensaje que indique la modificación correcta de la cuenta de usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 17: Caso de prueba de funcionalidad modificar cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código:32	**Historia de usuario:** Bloquear cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
Responsable de la prueba: Marisol Viera Figueroa	
Descripción: Se selecciona la cuenta usuario que se desea bloquear en el recurso Servidor de Base de Datos MySQL y se realiza la acción de bloquear.	
Condiciones de ejecución: 1. El paquete de conectores debe contar con la librería *MySQLdb.* 2. Debe existir alguna conexión abierta en el recurso Servidor de Base de Datos MySQL.	
Entrada/Pasos de ejecución: 1. Entrada de los datos de la cuenta de usuario: *user_name.* 2. Bloquear la cuenta de usuario. 3. Devolver datos del usuario bloqueado para luego poder deshacer la opción.	

Resultado esperado: Mensaje que indique que se bloqueó la cuenta de usuario.
Evaluación de la prueba: Prueba satisfactoria

Tabla 18: Caso de prueba de funcionalidad bloquear cuenta de usuario en el recurso Servidor de Base de Datos MySQL.
Fuente: Elaboración propia.

Caso de prueba de aceptación	
Código:65	**Historia de usuario:** Asignar permiso a los usuarios en el recurso Servidor de Base de Datos MySQL.
Responsable de la prueba: Marisol Viera Figueroa	
Descripción: Se selecciona el usuario al que se le desea asignar los permisos en el en el recurso Servidor de Base de Datos MySQL y se realiza la acción de asignan los permisos.	
Condiciones de ejecución: 1. El paquete de conectores debe contar con la librería *MySQLdb.* 2. Debe existir alguna conexión abierta en el recurso Servidor de Base de Datos MySQL.	
Entrada/Pasos de ejecución: 1. Entrada de los datos del usuario: *user_privileges= {'user_name', 'table_name'}* 2. Asignar permisos al usuario. 3. Devolver datos del usuario con los permisos para luego poder deshacer la opción.	
Resultado esperado: Mensaje que indique que se asignaron los permisos de forma satisfactoria.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 19: Caso de prueba de funcionalidad asignar permiso a los usuarios en el recurso Servidor de Base de Datos MySQL.
Fuente: Elaboración propia.

Resultados de las pruebas

Con el objetivo de obtener un producto de calidad y con la menor cantidad posible de errores se ejecutaron 3 iteraciones de prueba. En una primera iteración fueron detectadas 8 no conformidades en las pruebas unitarias, fundamentalmente relacionadas con la implementación y 9 no conformidades en las pruebas de aceptación. Al aplicar una segunda iteración se corrigieron los errores anteriores y se encontraron 4 no conformidades en las pruebas unitarias y 5 no conformidades en las pruebas de aceptación. En la tercera iteración se verifica

que fueron resueltas todas las no conformidades encontradas durante las iteraciones realizadas anteriormente y no se encontró ninguna no conformidad. Para observar el proceso de iteraciones realizadas al *software*, ver Figura 9.

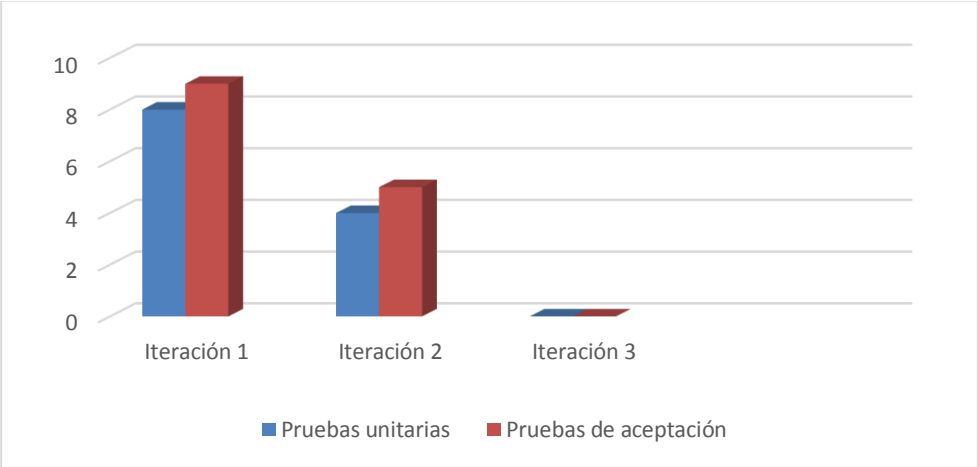


Figura 9: Resultados de las pruebas.
Fuente: Elaboración propia.

3.6 Conclusiones parciales

En este capítulo se precisó el estilo de codificación utilizado en la implementación del paquete de conectores, lo que permitió mantener una uniformidad en la codificación y mayor claridad a la hora de leer o agregar líneas de código. Se realizaron los diagramas de componentes y despliegue, los cuales posibilitaron un mejor entendimiento del *software* desarrollado, para lograr una fácil implementación. Se constató que el desarrollo guiado por pruebas asegura la ejecución correcta de todo el período de implementación.

Conclusiones generales

A partir del desarrollo de la presente investigación se arribaron a las siguientes conclusiones:

- ✓ La elaboración del marco teórico y el estudio de los diferentes sistemas homólogos en Cuba y el mundo, permitió obtener los conocimientos necesarios para una mayor comprensión del objeto de estudio y sentar las bases de la investigación.
- ✓ La selección de la metodología, las herramientas y tecnologías relacionadas con el desarrollo del paquete de conectores permitió crear una solución conforme a las necesidades del cliente.
- ✓ El estudio y análisis realizado en la investigación permitió generar los artefactos definidos por la metodología seleccionada, así como definir los elementos fundamentales de la arquitectura y el diseño de la propuesta de solución.
- ✓ A partir del diseño obtenido se codificaron los componentes necesarios para llegar a obtener el paquete de conectores.
- ✓ La realización de las pruebas unitarias y de aceptación permitió comprobar el buen funcionamiento del paquete de conectores, lo cual demuestra la validez del mismo con resultados satisfactorios.

Recomendaciones

Los resultados obtenidos luego del desarrollo del presente trabajo satisfacen los requerimientos definidos. Sin embargo, se recomienda a toda persona interesada en mejorar o continuar con este trabajo:

- ✓ Desarrollar otras versiones de esta solución con nuevas funcionalidades para que la misma pueda ser utilizada en un contexto diferente al aprovisionamiento de usuarios.
- ✓ Realizar la integración del paquete de conectores con otras aplicaciones.
- ✓ Mejorar el mecanismo de deshacer acciones, con el fin de que el paquete de conectores haga un manejo más completo de las acciones que se realizan sobre el servidor, permitiendo al sistema en el que se integre la implementación de un mecanismo transaccional.
- ✓ Extender el manejo de las peticiones ante fallos de conexión, mediante la posterior ejecución de las mismas en un momento que puede ser programado.

Bibliografía referenciada

1. Alonso, Begoña Eguía e Ixone. El desarrollo de las tecnologías de la información y las comunicación :un nuevo reto para el mercado de trabajo. *Scripta Nova.* [En línea] 1 de agosto de 2002. [Citado el: 10 de 10 de 2014.] http://www.ub.edu/geocrit/sn/sn119-74.htm. 1138-9788.

2. SeaMaster. *SeaMaster.* [En línea] [Citado el: 1 de 10 de 2014.] http://www.sea-master.eu/es/integracion.

3. Microsoft TechNet. *Microsoft TechNet.* [En línea] Microsoft, 2015. [Citado el: 23 de mayo de 2015.] https://technet.microsoft.com/es-es/library/bb821278.aspx#EDAA.

4. Nelsón P. Váldes. Cuba y la tecnología de la información. *Cuba y la tecnología de la información.* [En línea] octubre-diciembre de 2012. [Citado el: 3 de 11 de 2014.] http://www.temas.cult.cu/revistas/31/057-071nelson.pdf. no. 31: 57-71.

5. Ruz, Fidel Castro. *"Discurso en el encuentro con dirigentes de la UCI en el Consejo de Estado".* Cuba : s.n., 19 de agosto 2002.

6. Earl Perkins, Perry Carpenter. Magic Quadrant for User Provisioning. *Magic Quadrant for User Provisioning.* [En línea] 15 de agosto de 2008. [Citado el: 2015 de mayo de 20.] http://www.stsginc.com/marketing/gartner_provisioning_understanding.pdf.

7. Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal and Timothy Wood. *Agile dynamic provisioning of multi-tier Internet applications.* 2010.

8. Laura Barros, Patricia López y José M. Drake. *Tecnologías de componentes CMM basada en conectores.* España : Albacete, 2008. 978-84-691-2813-8.

9. José Luis Pastrana, Ernesto Pimentel,Miguel Katrib. *Coordinación de componentes mediante conectores implementados con sevicios web.*

10. Sitaraman, Gary T. Leavens y Murali. *Foundations of Component Based Systems.* s.l. : Cambrige University Press, 2000. 0 521 77164 1 hardback.

11. Francis Pisani, Dominique Piotet. *La alquimia de las multitudes.* Barcelona, España : Paidós Ibérica ,S.A, 2013. 978-84-493-2196-2.

12. Juan Desongles Corrales, Eduardo Antonio Ponce Cifredo, Ma. Luisa Garzón Villar. *Técnicos de Soporte Informático.* España: Mad : SL, 2012. 84-665-5102-6.

13. OI Architecture. *OI Architecture.* [En línea] julio de 2010. [Citado el: 2014 de 12 de 25.] http://www.oracle.com/technetwork/middleware/id-mgmt/overview/identity-manager-wp-11gr1-156947.pdf.

14. Oracle definiciones y reglas de licenciamiento. *Oracle definiciones y reglas de licenciamiento.* [En línea] [Citado el: 15 de enero de 2015.] http://www.oracle.com/ocom/groups/public/@opnpublic/documents/webcontent/042220_esa.pdf.

15. Oracle . *Oracle.* [En línea] [Citado el: 10 de 10 de 2014.] https://docs.oracle.com/cd/E37115_01/dev.1112/e27150/icf.htm#OMDEV3268.

16. Service Providers. *Service Providers.* [En línea] [Citado el: 16 de 10 de 2014.] http://technet.microsoft.com/en-us/library/cc546865.aspx.

17. WSO2. *WSO2.* [En línea] [Citado el: 9 de 10 de 2014.] https://docs.wso2.com/display/ESB480/Creating+a+Connector.

18. Enterprise Service Bus. *Enterprise Service Bus.* [En línea] WSO2 Inc, 2005-2014. [Citado el: 2015 de enero de 2015.] https://docs.wso2.com/display/ESB480/Creating+a+Connector.

19. IBM. *IBM.* [En línea] [Citado el: 16 de enero de 2015.] http://www-01.ibm.com/support/knowledgecenter/SSCQGF_7.1.0/com.ibm.IBMDI.doc_7.1/theconnector.htm.

20. IBM Knowledge Center. *IBM Knowledge Center.* [En línea] [Citado el: 8 de 11 de 2014.] http://www-01.ibm.com/support/knowledgecenter/SSYGQH_5.0.0/admin/install/c_tdi_about.dita?lang=es.

21. Wilhem Manuel Verano Escalona, Ernesto Raúl Martín Suárez. Sitio de la Biblioteca de la Universidad de las Ciencias Informáticas. *Sitio de la Biblioteca de la Universidad de las Ciencias Informáticas.* [En línea] [Citado el: 2 de 10 de 2014.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04991_11/1/TD_04991_11.pdf.

22. R., Pressman. *Software Engineering: A Practitioner's.Approach.* s.l. : McGraw-Hill., 2009 (7th edition).

23. Reynoso, Carlos. Metodologías de Desarrollo de *Software* Ágiles. *Metodologías de Desarrollo de Software Ágiles.* [En línea] 2004. [Citado el: 5 de 10 de 2014.] http://www.sel.unsl.edu.ar/ApuntesMaes/2004/Metodologias%20Agiles.doc.

24. Roberth G. Figueroa , Camilo J. Solís ,Armando A. Cabrera. *Metodologías tradicionales vs. metodologías ágiles.* 2001-2006.

25. Patricio Letelier y Mª Carmen Penadés. Métodologías ágiles para el desarrollo de *software*: Extreme Programing. *Métodologías ágiles para el desarrollo de software: Extreme Programing.* [En línea] abril-junio de 2013. [Citado el: 5 de 10 de 2014.] http://www.cyta.com.ar/ta0502/b_v5n2a1.htm. ISSN 1666-1680.

26. Reynoso, Carlos. Metodologías de Desarrollo de *Software* Ágiles. *Metodologías de Desarrollo de Software Ágiles.* [En línea] 2004. [Citado el: 5 de 10 de 2014.] http://www.sel.unsl.edu.ar/ApuntesMaes/2004/Metodologias%20Agiles.doc.

27. EngineeringFantasy. *EngineeringFantasy.* [En línea] [Citado el: 8 de 10 de 2014.] http://nafiulis.me/pycharm-the-good-parts-i.html.

28. Orallo, E.H. *El Lenguaje Unificado de Modelado (UML).* 2010.

29. Duque, Raúl González. *Python para todos.* España : Creative Commons Reconocimien-to 2.5 .

30. Raul González Luque. Curso Python. *Curso Python.* [En línea] Creative Commons Reconocimien. [Citado el: 2015 de mayo de 24.] http://dspace.universia.net/bitstream/2024/919/1/Python_para_todos.pdf.

31. Ligth, Richard. *Presenting XML.*

32. Collado, Manuel. *Introducción a las tecnologías y estándares XML.* 2010.

33. Multibody System. *Multibody System.* [En línea] Universidad Técnica de Madrid., 15 de abril de 2012. [Citado el: 16 de 11 de 2014.] http://mat21.etsii.upm.es/mbs/MechXML/que_es_xml_schema.htm.

34. XML Copy Editor. *XML Copy Editor.* [En línea] [Citado el: 15 de abril de 2015.] http://www.portalprogramas.com/xml-copy-editor/.

35. Workstation. *Workstation.* [En línea] [Citado el: 9 de 10 de 2014.] http://www.vmware.com/products/workstation.

36. Scribd. Scribd. [En línea] [Citado el: 25 de Enero de 2015.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

37. López, Pátricia. Herramienta CASE Visual Parading. *Herramienta CASE Visual Parading.* [En línea] [Citado el: 6 de 10 de 2014.] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-*software*-i/practicas-1/is1-p01-trans.pdf.

38. Joskowicz, Ing. José. *Reglas y prácticas en eXtreme Programing.* 2013.

39. Líber Batalla, Javier Garderes,Andrés Gatto. *Gestión de software Extreme Programing(XP).* 2010.

40. Alonso, Fernando Amo, Normand and Segovia, Francisco Javier Pérez. *El Proceso Unificado está centrado en la arquitectura. Introducción a la ingeniería del software.* s.l. : Delta Publicaciones, 2014. pp. 337-338.

41. Sommerville, Ian. *Ingeniería del Software.* Madrid : s.n., 2012.

42. Larman, Craig. *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.*

43. Torres, M. L. *Introducción al diseño con patrones.* 2009.

44. Giraldo Gómez, Gloria Lucia and Acevedo O., Juan F. and Moreno N., David A. . Una ontología para la representación de conceptos de diseño de *software*. *Una ontología para la representación de conceptos de diseño de software.* [En línea] 2012. [Citado el: 20 de enero de 2015.] http://www.bdigital.unal.edu.co/25046/1/22291-106826-1-PB.pdf.

45. Pressman, R. S. *Ingeniería de software. Un enfoque ágil.* McGraw-Hill : Quinta ed. 600 p. ISBN , 2002. 8448132149.

46. Raúl González Duque . Guía de estilo del código Python. *Guía de estilo del código Python.* [En línea] 10 de agosto de 2007. [Citado el: 25 de mayo de 2015.] http://mundogeek.net/traducciones/guia-estilo-python.htm.

47. Patricio Letelier. Métodologías ágiles para el desarrollo de *software*: eXtreme Programming (XP). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* [En línea] Técnica Administrativa, Buenos Aires, junio de 2006. [Citado el: 8 de abril de 2015.] http://www.cyta.com.ar -. ISSN 1666-1680.

48. Facebook Connector. *Facebook Connector.* [En línea] [Citado el: 27 de 9 de 2014.] https://www.mulesoft.com/cloud-connectors/facebook-integration-connector.

49. Reyes, J. L. G. *Plataforma de Gestión de los Servicios Telemáticos en GNU\Linux. Módulo DNS.* Universidad de las Ciencias Informáticas : s.n., 2013.

50. Tecnología de Información y la Comunicación-Correo Electrónico. *Tecnología de Información y la Comunicación-Correo Electrónico.* [En línea] [Citado el: 4 de 10 de 2014.] http://tutoriales.igluppiweb.com.ar/ecorreo.pdf.

51. OpenLDAP, Desarolladores. *OpenLDAP Software 2.4 Administrator's Guide.* 2014.

52. Informatica-hoy. *Informatica-hoy.* [En línea] [Citado el: 18 de 09 de 2014.] http://www.informatica-hoy.com.ar/redes-sociales/La-historia-de-las-redes-sociales.php.

53. Introduction to OpenLDAP Directory Services. *Introduction to OpenLDAP Directory Services.* [En línea] [Citado el: 20 de 09 de 2014.] http://www.openldap.org/doc/admin24/intro.html.

54. Computer World. *Computer World.* [En línea] [Citado el: 18 de 09 de 2014.] http://www.computerworld.com/article/2593623/app-development/application-programming-interface.html.

55. Keiver Hernández Fernández, Amaury Viera Hernández. *Migración del Directorio Activo a Plataforma Libre.* Ciudad de la Habana, Cuba : s.n.

56. Microsoft, C. *Gestión de Identidades Microsoft TechNet.* 2004.

57. affiliates, Oracle Corporation and/or its. NetBeans. *NetBeans.* [En línea] [Citado el: 13 de 10 de 2014.] http://netbeans.org/.

58. Tinoco Gómez, Rosales López. *Criterios de selección de metodologías de desarrollo de software.* Lima,Perú : vol. 13, nº 2, 2010. ISSN 1810-9993..

59. MSF for Agile *Software* Development. *MSF for Agile Software Development.* [En línea] GUIDANCE. [Citado el: 6 de 11 de 2014.] http://www.microsoft.com/downloads/details.aspx?familyid=9F3EA426-C2B2-4264-BA0F35A021D85234&displaylang=en.

60. Hernández, Marcos Legido. *Manual JavaScript.Características.* 2009.

61. Nallathamby, Johann Dilantha. *WSO2 Identity Server 5.0.0.*

62. Microsoft. *Microsoft.* [En línea] [Citado el: 2015 de febrero de 28.] https://msdn.microsoft.com/en-us/library/aa480356.aspx.

63. R, Pressman. *Software Engineering: A Practitioner's Approach.* s.l. : (7th edition),McGraw-Hill, 2009.

64. Onel Roselló Reyes, Carlos Sotolongo Alonso. Microsoft Developer Network. *Microsoft Developer Network.* [En línea] Microsoft, 2014. [Citado el: 2 de 11 de 2014.] https://msdn.microsoft.com/es-es/library/dd831853.aspx.

65. Raul Ruggia, Jorge Besiel,Carla Pais,Dario Sande. *Interoperabilidad entre Servidores de Aplicaciones.*

66. Manual de Referencia MySQL. *Manual de Referencia MySQL.* [En línea] [Citado el: 10 de febrero de 2015.] http://dev.mysql.com/doc/refman/5.5/en/index.html.

67. Conector de MySQL para Python, Guía del desarrollador. *Conector de MySQL para Python, Guía del desarrollador.* [En línea] [Citado el: 12 de febrero de 2015.] http://dev.mysql.com/doc/connector-python/en/index.html.

68. MySQLdb (API). *MySQLdb (API).* [En línea] [Citado el: 16 de febrero de 2015.] http://mysql-python.sourceforge.net/MySQLdb.html.

69. Documentación PostgreSQL 9.1.16. *Documentación PostgreSQL 9.1.16.* [En línea] [Citado el: 18 de febrero de 2015.] http://www.postgresql.org/docs/9.1/static/.

70. Psycopg2 (API). *Psycopg2 (API).* [En línea] [Citado el: 20 de febrero de 2015.] http://initd.org/psycopg/docs/.

71. Documentación en línea para el servidor de bases de datos Oracle 11g. *Documentación en línea para el servidor de bases de datos Oracle 11g.* [En línea] [Citado el: 20 de febrero de 2015.] https://docs.oracle.com/cd/E11882_01/nav/portal_4.htm.

72. cx_Oracle (API). *cx_Oracle (API).* [En línea] [Citado el: 22 de febrero de 2015.] https://cx-oracle.readthedocs.org/en/latest/.

73. Guía del administrador para OpenLDAP. *Guía del administrador para OpenLDAP.* [En línea] [Citado el: 22 de febrero de 2015.] http://www.openldap.org/doc/admin24/.

74. Python-ldap (API). *Python-ldap (API).* [En línea] [Citado el: 2 de marzo de 2015.] http://www.python-ldap.org/doc/html/index.html.

75. Guías para la administración de Active Directory (Windows Server 2008 R2). *Guías para la administración de Active Directory (Windows Server 2008 R2).* [En línea] [Citado el: 3 de febrero de 2015.] https://msdn.microsoft.com/en-us/library/bb742437.aspx.

76. Active Directory. *Active Directory.* [En línea] [Citado el: 6 de febrero de 2015.] https://technet.microsoft.com/en-us/library/dd349801%28v=ws.10%29.aspx.

77. Documentación Zimbra 8.6.0. *Documentación Zimbra 8.6.0.* [En línea] [Citado el: 10 de febrero de 2015.] https://www.zimbra.com/docs/ne/8.6.0/administration_guide/wwhelp/wwhimpl/js/html/wwhelp.htm#href=860_admin_ne.Zimbra_Collaboration.html.

78. Zmprov (Interfaz de Línea de Comandos para el aprovisionamiento de usuarios en el servidor). *Zmprov (Interfaz de Línea de Comandos para el aprovisionamiento de usuarios en el servidor).* [En línea] [Citado el: 12 de febrero de 2015.] https://wiki.zimbra.com/wiki/Zmprov_Examples.

79. Paramiko (librería para el trabajo con comandos en computadoras remotas usando el protocolo SSH). *Paramiko (librería para el trabajo con comandos en computadoras remotas usando el protocolo SSH).* [En línea] [Citado el: 20 de abril de 2015.] https://paramiko-docs.readthedocs.org/en/1.15/.

Glosario de términos

Aplicación: Cualquier programa que corra en un sistema operativo y que haga una función específica para un usuario.

Active Directory: Servicio establecido en uno o varios servidores en donde se crean objetos tales como usuarios, equipos o grupos, con el objetivo de administrar los inicios de sesión en los equipos conectados a la red, así como también la administración de políticas en toda la red. Su estructura jerárquica permite mantener una serie de objetos relacionados con componentes de una red, como usuarios, grupos de usuarios, permisos y asignación de recursos y políticas de acceso.

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.

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.

Multiplataforma: Término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de *software*, que puedan funcionar en diversas plataformas. Una plataforma es una combinación de hardware y *software* usada para ejecutar aplicaciones.

MySQL: Sistema de gestión de bases de datos relacional, multihilo y multiusuario. Es una base de datos *Open Source* muy rápida.

Openfire: Sistema de mensajería instantánea GPL, utiliza el protocolo XMPP, es un servidor de mensajería donde se puede administrar usuarios, compartir archivos, auditar mensajes, mensajes o*ffline*, mensajes *broadcast*, grupos y además contiene plugins gratuitos con diferentes funciones.

Oracle: Sistema de gestión de base de datos objeto-relacional. Se considera a Oracle Database como uno de los sistemas de bases de datos más completos, donde resalta: soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma.

PostgreSQL: Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD, posee alta concurrencia.

Servidor: Un servidor es una computadora que maneja peticiones de datos, email, servicios de redes y transferencia de archivos de otras computadoras (clientes). También puede referirse a un *software* específico. Una computadora puede tener distintos *software* de servidor, lo cual proporciona muchos servidores a clientes en la red.

Usuario: Persona que tiene una cuenta en una determinada computadora por medio de la cual puede acceder a los recursos y servicios que ofrece una red.

Zimbra: Es una plataforma de soporte al trabajo colaborativo de nivel empresarial basada en *software* libre, que ofrece servicios integrados de mensajería y colaboración. Utiliza las infraestructuras de comunicaciones disponibles hoy en día y provee acceso a correo electrónico, calendario, libreta de direcciones y mensajería a través de una interfaz unificada accesible vía Web.