



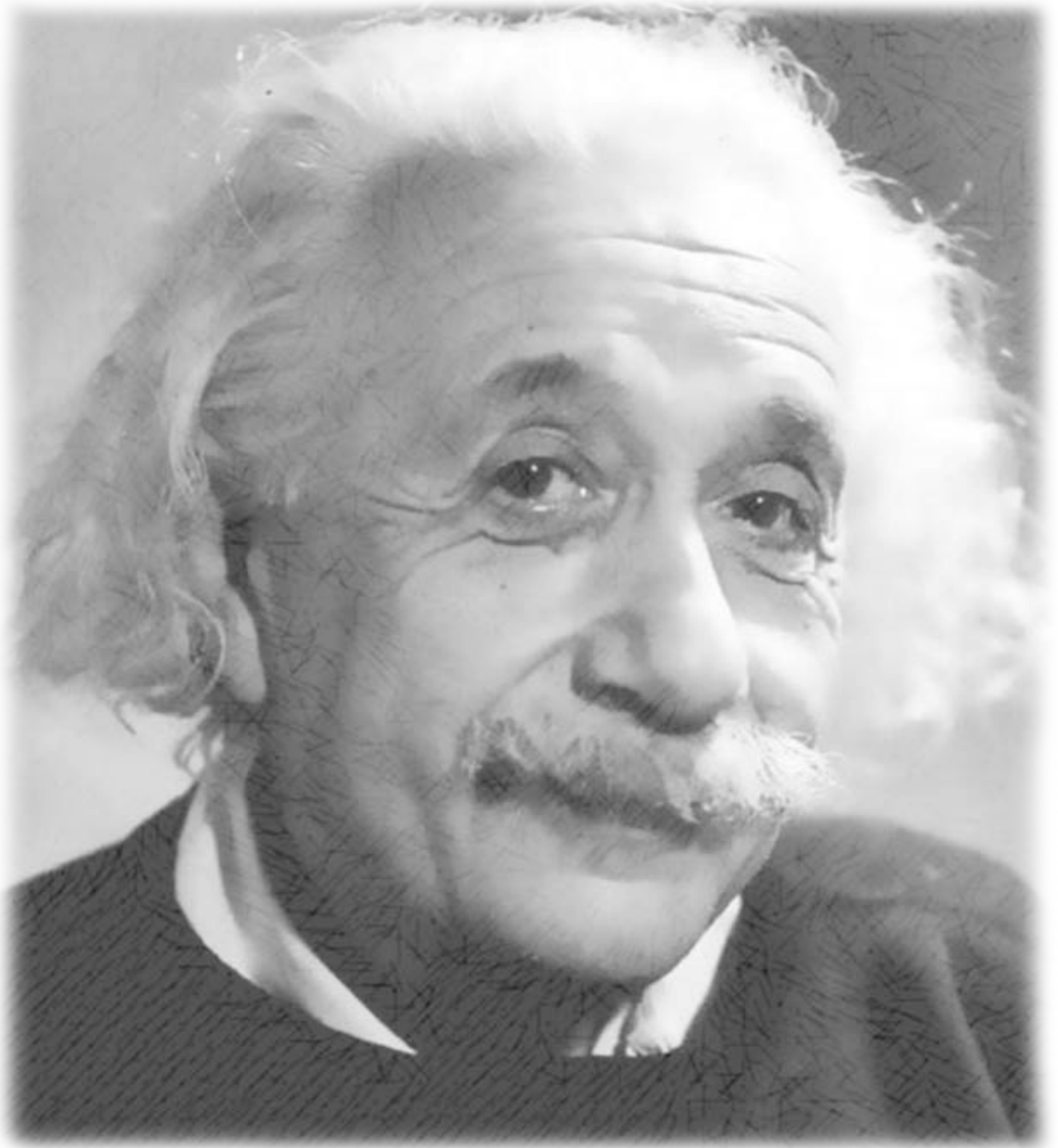
Facultad 5
Centro de Consultoría y Desarrollo de
Arquitecturas Empresariales.

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

Título: Extensión del módulo de seguridad para el
Replicador de datos REKO.

Autor: Rioger López Collar.
Tutora: Ing. Madielennis Núñez de los Ríos

La Habana, Junio de 2015
“Año 57 de la Revolución”



“Sí lo puedo pensar e imaginar, es que lo puedo hacer.”

Albert Einstein

DECLARACIÓN JURADA DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2015.

Firma del autor:

Rioger López Collar.

Firma de la tutora:

Ing. Madielennis Núñez de los Ríos

DATOS DE CONTACTO

Autor

Rioger López Collar

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Correo: rcollar@estudiantes.uci.cu

Tutora

Ing. Madielennis Núñez de los Ríos

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Correo: mnrios@uci.cu

AGRADECIMIENTOS

A mis padres, a mis hermanos, a mi hermana Jimagua, a mi tío y a toda mi familia en general por ayudarme siempre en todo momento brindándome su apoyo.

A mi tutora Made por haberme guiado a como desempeñar una buena investigación y convertirme en un buen ingeniero.

A los profesores Frank Rosales y Gloria Raquel, que a pesar de no ser los tutores directos, siempre estuvieron dispuestos a ayudarme ante cualquier duda que tuviera.

A todo el equipo de desarrollo de REKO, por las experiencias y conocimientos transmitidos.

A mis compañeros de apartamento Felipe, Koko, Walter, Lara, Lorgio y Cuba, que desde el primer año hemos estado luchando por graduarnos.

A las amistades que cultivé a lo largo de la carrera Roberto, Marlon, Mirneris, Orson, Marin, Erduin, Liset, Manuel.

A mi ACER ASPIRE, por estar ahí conmigo para arriba y para abajo, aguantando más de quince horas encendida todos los días para lograr este trabajo.

Quiero agradecer a todas las personas que han hecho posible de una forma u otra que hoy pueda estar optando por el título de Ingeniero Informático.

DEDICATORIA

De manera especial quiero dedicar este trabajo a mis padres, por estar siempre presentes y darme su apoyo incondicional.

RESUMEN

El Replicador de datos REKO es un software de réplica de datos entre bases de datos distribuidas que pretende cubrir las principales necesidades relacionadas con la protección, recuperación, sincronización y transferencia de datos entre diversas localizaciones.

El sistema posee un módulo de seguridad de autenticación basado en roles, que controla el acceso a los recursos y permite la gestión y configuración de los mismos. Dicho módulo no brinda la flexibilidad para adaptarse a nuevos mecanismos de seguridad por lo que el objetivo de esta investigación se centra en la extensión del módulo de seguridad que garantice de forma adecuada la autenticación mediante un Protocolo Ligero de Acceso a Directorios (del inglés *Lightweight Directory Access Protocol*, LDAP), así como la importación y sincronización de usuarios. Todo esto permite que cada usuario tenga acceso solamente al rol o roles que desempeña dentro del sistema, de forma que se automatice la creación de cuentas de usuario.

Para guiar el proceso de desarrollo se empleó la metodología ágil AUP-UCI que abarca todo el proceso de desarrollo del software con iteraciones pequeñas. El módulo ha sido diseñado e implementado en su totalidad usando herramientas de código abierto como: Visual Paradigm, Eclipse Kepler, Acegi Security, entre otras. Se realizaron pruebas al módulo validando el correcto funcionamiento del mismo, concluyendo que con la utilización del protocolo LDAP se aumentan los niveles de seguridad en el acceso a la consola de administración Web del Replicador de datos REKO, disminuyendo el tiempo de respuesta al usuario y el esfuerzo.

Palabras claves: módulo de seguridad, LDAP, REKO, Replicador de datos.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN.....	1
CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.....	5
1.1 Conceptos fundamentales	5
1.1.1 Seguridad de la información.....	5
1.1.2 Control de acceso.....	6
1.2 Estándares de autenticación	6
1.2.1 Radius	6
1.2.2 Kerberos	6
1.2.3 NTLM.....	7
1.3 Técnicas de identificación y autenticación.....	7
1.4 Descripción general del objeto de estudio.....	7
1.4.1 Servicio de directorio LDAP.....	8
1.4.2 Principales implementaciones de servidores LDAP	9
1.4.3 Diferencias entre OpenLDAP y Active Directory	9
1.4.4 Características de un directorio LDAP	10
1.4.5 Estructura del directorio.....	10
1.4.6 Operaciones sobre el directorio LDAP	11
1.4.7 Ventajas de un servidor LDAP	11
1.5 Herramientas con mecanismos similares	12
1.5.1 Ámbito internacional.....	12
1.5.2 Ámbito nacional.....	13
1.6 Metodologías de desarrollo	14
1.6.1 Metodologías tradicionales.....	14
1.6.2 Metodologías ágiles.....	15
1.6.2.1 AUP	15
1.7 Técnicas, métodos, herramientas y lenguajes para el desarrollo.....	19
1.7.1 Técnicas y métodos.....	19
1.7.2 Herramientas y lenguajes de desarrollo	20
1.8 Conclusiones parciales.....	22
CAPÍTULO 2. CARACTERÍSTICAS Y DISEÑO DEL SISTEMA	23
2.1 Flujo actual de los procesos	23
2.2 Propuesta de solución	24
2.3 Modelo de dominio	24
2.3.1 Descripción del diagrama de clases del Modelo de dominio.....	24
2.4 Modelado del sistema.....	25

2.4.1	Requisitos Funcionales	25
2.4.2	Requisitos No Funcionales.....	25
2.4.3	Historias de Usuario	27
2.4.4	Descripción de las Historias de Usuario.....	28
2.5	Descripción de la arquitectura del Replicador de datos REKO	30
2.5.1	Estilos arquitectónicos.....	30
2.5.2	Patrones	32
2.5.3	Patrones arquitectónicos	32
2.5.3.1	Arquitectura en capas.....	33
2.5.4	Patrones de diseño.....	33
2.5.4.1	Patrón de Acceso a Datos.....	33
2.5.4.2	Patrones GRASP.....	34
2.5.4.3	Patrones GOF	35
2.6	Modelo de diseño	35
2.6.1	Diagrama de clases del diseño	35
2.6.2	Descripción de las clases.....	37
2.6.3	Diagrama de despliegue.....	40
2.7	Conclusiones parciales.....	41
CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS.....		42
3.1	Modelo de implementación.....	42
3.1.1	Diagrama de componentes	42
3.2	Código fuente	43
3.2.1	Proceso de filtrado e importación de usuarios de un servidor LDAP	43
3.2.2	Proceso de sincronización de servidor LDAP con BD embebida en REKO.....	48
3.2.3	Proceso de autenticación con servidor LDAP	50
3.3	Modelo de pruebas	52
3.3.1	Técnicas y estrategias de prueba.....	52
3.3.2	Pruebas de Caja Negra	53
3.3.2.1	Casos de pruebas de Caja Negra	54
3.3.3	Pruebas de Caja Blanca.....	58
3.3.4	Resultados obtenidos	60
3.4	Conclusiones parciales.....	60
CONCLUSIONES.....		61
RECOMENDACIONES.....		62
REFERENCIAS BIBLIOGRÁFICAS		63
GLOSARIO DE TÉRMINOS		68

ÍNDICE DE TABLAS

Tabla 1: Diferencias entre OpenLDAP y Active Directory	9
Tabla 2: Fases de AUP-UCI.	16
Tabla 3: Disciplinas de AUP-UCI.....	17
Tabla 4: Roles de AUP-UCI.....	18
Tabla 5: Herramientas y lenguajes para el desarrollo.	20
Tabla 6: Catálogo de los RF.....	25
Tabla 7: Catálogo de los RNF.	26
Tabla 8: HU1. Editar configuración de conexión	28
Tabla 9: HU4. Importar usuarios filtrados	29
Tabla 10: Descripción de la clase “SecurityManager”.	38
Tabla 11: Descripción de la clase “UserDaoImpl”.	39
Tabla 12: Descripción de variables.....	54
Tabla 13: Matriz de datos	55

ÍNDICE DE FIGURAS

Figura 1: Representación del funcionamiento de los servicios de directorio LDAP	8
Figura 2: Ejemplo de la estructura de un árbol de directorio	10
Figura 3: Disciplinas desarrolladas en cada fase según AUP. (30).....	18
Figura 4: Fases e iteraciones de AUP. (31).....	19
Figura 5: Formulario para la creación de cuentas de usuario en REKO.	23
Figura 6: Diagrama del Modelo de dominio.....	24
Figura 7: Vista de los principales componentes de REKO	31
Figura 8: Arquitectura de la seguridad en REKO.	32
Figura 9: Diagrama de clases de diseño “Editar configuración de conexión”.....	35
Figura 10: Diagrama de clases de diseño “Buscar roles”.....	36
Figura 11: Diagrama de clases de diseño que incluye los requisitos “Filtrar usuarios del LDAP, Importar usuarios filtrados, Sincronizar servidor LDAP con BD embebida en REKO, Buscar usuarios”.	36
Figura 12: Diagrama de clases de diseño “Autenticar con LDAP”.	37
Figura 13: Descripción de la clase “SecurityManager”.....	37
Figura 14: Descripción de la clase “UserDaoImpl”.	39
Figura 15: Diagrama de despliegue.....	40
Figura 16: Diagrama de componente “Editar configuración de conexión”.....	42
Figura 17: Diagrama de componente “Importar usuarios filtrados”	43
Figura 18: <i>Bean</i> “daoAuthenticationProvider”.....	51
Figura 19: <i>Bean</i> “berkeleyDaoProviderImpl”.	51
Figura 20: Método de prueba <i>testGetUsersTableLDAP</i> de la clase <i>SecurityUserCfgControllerTest</i> ..	59
Figura 21: Método de prueba <i>testGetsearchUsersTable</i> de la clase <i>SecurityUserCfgControllerTest</i> .	59
Figura 22: Comparación del proceso de creación de cuentas de usuario a través de un servidor LDAP o manualmente usando la BD embebida de REKO.	60

INTRODUCCIÓN

La seguridad informática desempeña un papel clave en el combate del delito cibernético; la tecnología, el software y con ello el *malware* (software malicioso) revolucionan constantemente, lo cual ocasiona que las grandes entidades a nivel mundial destinen cuantiosos recursos para mejorar sus sistemas de seguridad. La fortaleza de estos sistemas depende “en gran medida” de los componentes de autenticación que implementan, puesto que en estos se verifica si el usuario que trata de identificarse es válido, de esta forma se evita que las áreas restringidas sean visitadas por intrusos.

La autenticación, aunque no debe ser el único mecanismo de seguridad implementado en un sistema, es uno de los más importantes, puesto que la mejor prevención ante un ataque es la de denegar el acceso desde un principio al atacante. Este proceso se ha perfeccionado en el transcurso de los años con la incorporación de nuevas técnicas que autentican al usuario con una probabilidad muy baja de errores y hacen cada vez más difícil de burlar la seguridad de los sistemas.

La Universidad de Ciencias Informáticas (UCI), creada a partir del pensamiento visionario del compañero Fidel, es un centro de enseñanza superior de nuevo tipo, que tiene como uno de sus objetivos principales el impulso del software cubano. Para cumplir esta meta se ha trabajado constantemente para elevar el nivel de preparación en los temas de seguridad informática y aún con la escasa disponibilidad de recursos, se aplican iniciativas para elevar la seguridad del software desplegado y se diseñan componentes de autenticación mediante la combinación de estrategias donde se utilizan las nuevas tecnologías.

Dentro de la universidad, específicamente en el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE) se desarrolla una solución para el proceso de réplica de datos llamada Replicador de datos REKO, el mismo pretende cubrir las principales necesidades relacionadas con la protección, recuperación, sincronización y transferencia de datos entre diversas localizaciones. La seguridad de REKO está basada fundamentalmente en los usuarios, los roles y los permisos de navegación que le son asignados a estos.

A pesar de todo lo que se ha avanzado en el trabajo para mejorar la seguridad del Replicador de datos REKO, todavía presenta algunas limitaciones, puesto que depende totalmente del mecanismo de autenticación por roles para gestionar la autenticación y autorización de sus usuarios, la cual se realiza a través de una base de datos embebida. En dicho sistema el proceso de creación de las cuentas de usuario se realiza manual, lo que podría provocar un rechazo por parte de los clientes que optan por usar este producto de réplica, ya que en una red con muchos usuarios de réplica el proceso se volvería muy tedioso. Unido a esto, mantener actualizados los usuarios del sistema con respecto a los existentes en la institución podría requerir un constante seguimiento sobre los cambios ocurridos.

En muchas instituciones en las que puede ser instalado el sistema, ya existen servidores que contienen la información de los usuarios, sin embargo, resulta engorroso para el administrador de la seguridad del Replicador de datos REKO gestionar la información relacionada con los mismos. A

solicitud de los clientes de algunas de estas instituciones fue requerido que el sistema ofreciera los mecanismos necesarios para el uso de un servidor LDAP.

Dentro de las principales limitaciones que presenta la seguridad del Replicador de datos REKO se encuentran:

- ✓ El proceso de creación de cuentas de usuario es engorroso, debido a que se debe introducir de forma manual la información de cada usuario.
- ✓ Depende totalmente del uso de una base de datos embebida para el proceso de autenticación.
- ✓ El administrador debe verificar constantemente que los usuarios con acceso al sistema no han causado baja de la institución.
- ✓ Para la gestión de los usuarios y roles, los procesos de búsqueda no se encuentran bien definidos en el sistema.
- ✓ El nivel de usabilidad de las interfaces es bajo.

Teniendo en cuenta la situación problemática descrita se define como **problema a resolver**: ¿Cómo gestionar la información de los usuarios a través de un servidor LDAP, para aumentar el nivel de seguridad del Replicador de datos REKO?

Para darle solución al problema antes mencionado se define como **objeto de estudio de la investigación**: Las aplicaciones que poseen mecanismos de integración con un servidor LDAP. Centrado en el **campo de acción**: Los procesos de integración y sincronización con un servidor LDAP.

Para darle solución al problema descrito, se define el siguiente **objetivo general**: Extender el módulo de seguridad del Replicador de datos REKO para que permita gestionar la seguridad a través de un servidor LDAP.

Se plantea como **Idea a defender**: La extensión del módulo de seguridad a través de un servidor LDAP en el Replicador de datos REKO aumentará los niveles de seguridad en el acceso a la consola de administración Web.

Para cumplir con el objetivo enunciado se propusieron las siguientes **tareas investigativas**:

- ✓ Elaboración del marco conceptual para definir los conceptos fundamentales de la investigación.
- ✓ Análisis de las principales aplicaciones que poseen mecanismos de integración con un servidor LDAP.
- ✓ Estudio de las herramientas, estándares, tecnologías y lenguajes utilizados en el ambiente de desarrollo del Replicador de datos REKO.
- ✓ Definición de los requerimientos que dan cumplimiento a la solución propuesta.
- ✓ Elaboración de la documentación correspondiente a los flujos de trabajo propuestos por la metodología de desarrollo seleccionada.

- ✓ Diseño e implementación de un módulo que permita importar y sincronizar de manera automatizada los usuarios de un servidor LDAP hacia REKO.
- ✓ Validación de las funcionalidades implementadas.

Para un mejor desarrollo de la investigación se utilizó los siguientes métodos teóricos:

- ✓ **Histórico-Lógico:** Permitió el estudio de las distintas etapas por las que atravesó el protocolo LDAP en un orden cronológico, para conocer su evolución desde su surgimiento y poder determinar sus tendencias. Además, permitió determinar las principales características del protocolo LDAP.
- ✓ **Analítico-Sintético:** Permitió descomponer en varias partes los elementos relacionados con las herramientas que poseen mecanismos de integración con servidores LDAP. También permitió analizar por separado cada elemento y obtener una comprensión total de estos para posteriormente formular conclusiones.
- ✓ **Modelación:** Se aplicó para representar mediante diagramas y modelos el proceso de integración con servidores LDAP, lo cual facilita el entendimiento de la solución desarrollada.

Dentro de los métodos empíricos empleados se encuentran:

- ✓ **Observación:** Permitió conocer la realidad mediante la percepción directa de los objetos y fenómenos relacionados con los mecanismos de integración a servidores LDAP, así como el estudio del módulo de seguridad a extender.
- ✓ **Análisis documental:** En la consulta de la literatura especializada en las temáticas afines a la seguridad y los servidores LDAP.
- ✓ **Consulta de la información en todo tipo de base:** A través de la aplicación de este método se logra obtener los elementos fundamentales a abordar en el marco teórico de la investigación.

El presente trabajo de diploma consta de introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, glosario de términos y anexos.

Capítulo 1: Fundamentación teórica, se expone un análisis de todos los aspectos teóricos necesarios para comprender el problema en cuestión. Se estudia el funcionamiento del servicio de directorio LDAP y se muestra el resultado del estudio del estado del arte de algunas aplicaciones que cuentan con mecanismos de integración con un servidor LDAP. Además, se relacionan las herramientas existentes para el desarrollo de aplicaciones Web y se definen las empleadas para el desarrollo del sistema.

Capítulo 2: Características y diseño del sistema, contiene una descripción detallada del flujo de los procesos que dan surgimiento al problema. Se describe la propuesta de solución a partir de un análisis de las características de REKO y un Modelo de dominio que facilita la comprensión de su funcionamiento. Se relacionan los requisitos funcionales y no funcionales que debe tener el módulo a desarrollar para satisfacer las necesidades del cliente. Se describe la arquitectura y los patrones de diseño a emplear en su implementación.

Capítulo 3: Implementación y Pruebas, se muestra la implementación del módulo y se brinda una solución a los requisitos especificados. Se describe el código fuente de las principales clases y se aplican pruebas al módulo para demostrar la robustez del mismo, y verificar que realmente realice lo que se espera de él.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los aspectos y conceptos esenciales relacionados con los servidores LDAP. Se realiza un estudio del arte de las diversas herramientas que permiten acceder a la información contenida en los directorios LDAP o implementan algún mecanismo de integración con estos, tanto del ámbito internacional como nacional. Además, se caracterizan las técnicas, métodos y herramientas que se utilizan en el ambiente de desarrollo del Replicador de datos REKO y que hacen posible la extensión del módulo.

1.1 Conceptos fundamentales

1.1.1 Seguridad de la información

La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. (1)

La misma se encuentra dividida en dos términos importantes: Seguridad Lógica y Seguridad Física. La Seguridad Física es la aplicación de barreras físicas y procedimientos de control, como medidas de prevención y contramedidas ante amenazas a los recursos e información confidencial.

El activo más importante que poseen las instituciones es la información, por lo tanto, deben existir técnicas más allá de la Seguridad Física que la asegure. Estas técnicas las brinda la Seguridad Lógica que consiste en la aplicación de barreras y procedimientos que resguarden el acceso a los datos y solo se permita acceder a ellos a las personas autorizadas para hacerlo. (1)

La misma se propone como objetivos principales:

- ✓ Restringir el acceso a los programas y archivos.
- ✓ Asegurar que los operadores puedan trabajar sin una supervisión minuciosa y no puedan modificar los programas ni los archivos que no le correspondan.
- ✓ Verificar que se estén utilizando los datos, archivos y programas guiados por el procedimiento correcto.
- ✓ Asegurar que la información transmitida sea recibida solo por el destinatario al cual ha sido enviada y no a otro.
- ✓ Asegurar que la información recibida sea la misma que ha sido transmitida.

La Seguridad de la Información tiene como objetivo la protección de los datos en sí y trata de evitar su pérdida y modificación no autorizada. La protección debe garantizar en primer lugar la confidencialidad e integridad de los datos. (2)

- ✓ **Confidencialidad:** Medidas enfocadas a garantizar que la información está disponible para aquellos que estén autorizados a conocerla.
- ✓ **Integridad:** Garantizar que la información es fiable y que no se ha modificado.

Un término importante a tener en cuenta dentro de la seguridad lógica es el procedimiento que se realiza para verificar si corresponde un permiso de acceso solicitado por un usuario a un determinado recurso. Con el objetivo de brindar mayor facilidad en este proceso de revisión o verificación de acceso a cierto sistema, son creados los controles de accesos.

1.1.2 Control de acceso

Es el proceso de conceder permisos a usuarios o grupos, de acceder a objetos tales como ficheros o impresoras en la red. El control de acceso está basado en tres conceptos fundamentales: identificación, autenticación y autorización.

Los controles de accesos son necesarios para proteger la confidencialidad, integridad y disponibilidad de los objetos, y por extensión de la información que contienen, pues permiten que los usuarios autorizados accedan solo a los recursos que ellos definieron para realizar sus tareas. (3)

✓ Identificación

Es la acción por parte de un usuario de presentar su identidad a un sistema, usualmente se utiliza un identificador de usuario. Establece que el usuario es responsable de las acciones que lleve a cabo en el sistema. Esto está relacionado con los registros de auditorías que permiten guardar las acciones realizadas dentro del sistema y rastrearlas hasta el usuario autenticado. (3)

✓ Autenticación

Proceso por el cual se determina si un usuario tiene permiso para acceder a un determinado servicio de red del que quiere hacer uso. El proceso de autenticación se realiza mediante la presentación de las credenciales por parte del usuario que demanda acceso. (4)

✓ Autorización

Conceder servicios específicos (entre los que se incluye la “negación de servicio”) a un determinado usuario, basándose para ellos en su propia autenticación, los servicios que este solicita, y el estado actual del sistema. (4)

1.2 Estándares de autenticación

Un protocolo o estándar de autenticación (o autenticación) es un tipo de protocolo criptográfico que tiene el propósito de autenticar entidades que desean comunicarse de forma segura. Algunos protocolos de autenticación son: (5)

1.2.1 Radius

Dial de autenticación remoto para acceso a servicios (del inglés *Remote Authentication Dial-In User Service*, RADIUS). Es un protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP. (6)

Cuando se realiza la conexión mediante módem, Ethernet o Wi-Fi, se envía una información que generalmente es un nombre de usuario y una contraseña. Esta información se transfiere a un dispositivo Servidor de Acceso a la Red (del inglés *Network Access Server*, NAS), quien redirige la petición a un servidor RADIUS sobre el protocolo RADIUS. El servidor RADIUS comprueba que la información es correcta. Si es aceptado, el servidor autoriza el acceso al sistema y le asigna los recursos de red como una dirección IP, etc. (6)

1.2.2 Kerberos

La autenticación de los usuarios a los servicios de red puede mostrarse peligrosa cuando el método utilizado por el protocolo es inherentemente inseguro, como se evidencia por la transferencia de

contraseñas sin encriptar sobre la red bajo el Protocolo de Transferencia de Archivos (del inglés *File Transfer Protocol*, FTP). Kerberos es una forma de eliminar la necesidad de aquellos protocolos que permiten métodos de autenticación inseguros, y de esta forma mejorar la seguridad general de la red. Es un protocolo de seguridad que usa una criptografía de claves simétricas para validar usuarios con los servicios de red evitando así tener que enviar contraseñas a través de la red. Al validar los usuarios para los servicios de la red por medio de Kerberos, se frustran los intentos de usuarios no autorizados que intentan interceptar contraseñas en la red. (7)

1.2.3 NTLM

NT LAN Manager (NTLM) se utiliza como protocolo de autenticación para las transacciones entre dos equipos en los que al menos uno de ellos ejecuta Windows NT 4.0 o una versión anterior. Las redes con esta configuración se denominan de modo mixto, que es la configuración predeterminada en la familia de servidores Windows Server 2003. Además, NTLM es el protocolo de autenticación para equipos que no forman parte de un dominio, como los servidores independientes y los grupos de trabajo. (8)

Dentro de los estándares de autenticación estudiados Kerberos es el empleado en nuestra solución, debido a que es el utilizado por los servicios de directorios LDAP para realizar el proceso de autenticación.

Existen diferentes técnicas que permiten realizar la autenticación de la identidad del usuario, las cuales pueden ser utilizadas individualmente o combinadas.

1.3 Técnicas de identificación y autenticación

Técnicas para realizar autenticación de identidad de usuarios: (3)

- Algo que solamente el individuo conoce: Por ejemplo una contraseña.
- Algo que la persona posee: Por ejemplo una tarjeta magnética.
- Algo que el individuo es y que lo identifica unívocamente: Por ejemplo las huellas digitales.
- Algo que solo el individuo es capaz de hacer: Por ejemplo los patrones de escritura.

La fortaleza de la autenticación es mayor mientras más factores se adicionen, generalmente solo se utilizan hasta 3 factores: (9)

- 1 factor = contraseña
- 2 factores = contraseña + token
- 3 factores = contraseña + token + biometría
- 4 factores = contraseña + token + biometría + sistema de posicionamiento global (del inglés *Global Positioning System*, GPS)
- 5 factores = contraseña + token + biometría + GPS + perfil de usuario.

1.4 Descripción general del objeto de estudio

Un directorio es una Base de Datos (BD) optimizada para lectura, navegación y búsqueda conteniendo información descriptiva basada en atributos y tienen capacidades de filtrado muy sofisticadas. Generalmente no soportan transacciones complicadas ni esquemas de vuelta atrás

como los que se encuentran en los sistemas de bases de datos diseñados para manejar grandes y complejos volúmenes de actualizaciones. Las actualizaciones de los directorios son normalmente cambios simples, o todo o nada, siempre y cuando estén permitidos. Estos están afinados para dar una rápida respuesta a grandes volúmenes de búsquedas y tienen la capacidad de replicar la información para incrementar la disponibilidad y la fiabilidad, al tiempo que reducen los tiempos de respuesta. (10)

La integración de las aplicaciones con los directorios LDAP de una institución facilita la gestión de los usuarios, brinda un origen común para los datos de estos y un mecanismo de autenticación centralizado. Los datos almacenados en dichos directorios son empleados por las aplicaciones, principalmente para la definición de la seguridad del sistema basada en usuarios y roles. En aras de dar cumplimiento a los objetivos planteados se hace necesario comprender las características, estructura y funcionamiento de los servicios de directorio LDAP, así como estudiar los mecanismos empleados por las aplicaciones para integrarse a este.

1.4.1 Servicio de directorio LDAP

Protocolo Ligero de Acceso a Directorios (del inglés *Lightweight Directory Access Protocol*, LDAP) 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. El directorio LDAP también se considera una base de datos a la que pueden realizarse consultas.

En la década de los 80 comenzaron a utilizarse los servicios de directorio como X.500, sobre todo por las compañías de telefonía para almacenar los datos de sus clientes, algo parecido a lo que se conoce como páginas amarillas.

Los servicios de X.500 se accedían mediante el Protocolo de Acceso a Directorio (del inglés *Directory Access Protocol*, DAP) que utilizaban el conjunto de protocolos OSI, lo que hacía que fuese muy pesado. LDAP en lugar de utilizar todo el Modelo OSI utiliza el Modelo TCP/IP, en concreto el puerto 389/TCP, lo que hizo que fuese más ligero. (11)

La información contenida en un directorio LDAP normalmente se lee más de lo que se escribe, por lo que están optimizados para dar una respuesta rápida a operaciones de consulta o búsqueda. Los servidores que brindan acceso a este tipo de directorios se denominan servidores LDAP. A continuación se observa un esquema de funcionamiento bien detallado. (Ver Figura 1)



Figura 1: Representación del funcionamiento de los servicios de directorio LDAP

Varias compañías han desarrollado implementaciones de servidores LDAP, a continuación se listan algunas de estas:

- Microsoft Active Directory
- Novell Directory Services
- OpenLDAP
- IBM Security Directory Server
- Oracle Internet Directory

Entre los estándares que componen un directorio activo, y que han ayudado a mejorar su seguridad, destacan: (12)

- ✓ **Sistema de Nombres de Dominio** (del inglés *Domain Name System*, DNS): Es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.
- ✓ **LDAP**: Protocolo ligero de acceso a directorio. Este es el protocolo mediante el cual las aplicaciones acceden y modifican la información existente en el directorio.
- ✓ **Kerberos**: Protocolo utilizado para la autenticación de usuarios y máquinas.

1.4.2 Principales implementaciones de servidores LDAP

Entre las implementaciones anteriores algunas de las más comunes son OpenLDAP y el Directorio Activo de Microsoft (Active Directory), por lo que se desea que la solución implementada soporte ambas.

✓ OpenLDAP

El proyecto OpenLDAP nació como la continuación de la versión 3.3 del servidor LDAP de la Universidad de Michigan. Este es un servidor LDAP que se distribuye bajo licencia GNU (Código Abierto), que permite que el software se pueda usar de forma gratuita tanto de forma educativa como profesional. Además, se dispone del código fuente para poder realizar modificaciones. (13)

✓ Active Directory

Active Directory es una implementación propietaria (creada por Microsoft) de los Servicios de directorio, y proporciona una manera de compartir información entre recursos y usuarios de la red. Además de proporcionar una fuente centralizada para esa información, Active Directory también funciona como autoridad de seguridad centralizada de autenticación para la red. (14)

1.4.3 Diferencias entre OpenLDAP y Active Directory

Existen algunos campos que presentan diferencias en los esquemas de estas dos implementaciones de servidores LDAP. (Ver Tabla 1)

Tabla 1: Diferencias entre OpenLDAP y Active Directory

OpenLDAP	Active Directory
dn	dn
displayname	displayname
givenname	givenname
uid	samaccountname

Fuente: Elaboración propia.

1.4.4 Características de un directorio LDAP

Un directorio LDAP se destaca sobre los demás tipos de bases de datos por las siguientes características: (15)

- Es muy rápido en la lectura de registros.
- Permite replicar el servidor de forma muy sencilla y económica.
- Muchas aplicaciones poseen interfaces de conexión a servidores LDAP y se pueden integrar fácilmente.
- Dispone de un modelo de nombres globales que asegura que todas las entradas son únicas.
- Usa un sistema jerárquico de almacenamiento de información.
- Funciona sobre el Protocolo de Control de Transmisión (de inglés *Transmission Control Protocol*, TCP) y sobre el protocolo criptográfico Capa de Conexión Segura (del inglés *Secure Sockets Layer*, SSL).
- La mayoría de los servidores LDAP son fáciles de instalar, mantener y optimizar.

Las características antes mencionadas hacen de los directorios LDAP un servicio con gran aceptación por parte de los administradores de red. Cada implementación de servidores LDAP posee sus peculiaridades, sin embargo, existen algunos aspectos en la estructura del directorio que son comunes.

1.4.5 Estructura del directorio

Los servidores LDAP almacenan la información en una estructura de datos jerárquica. (Ver Figura 2) Esta estructura puede ser distribuida, o sea, puede existir un servidor LDAP que contenga un subárbol de otro. Los objetos se almacenan en entradas y cada entrada posee un conjunto de atributos y un Nombre Distinguido (DN, por sus siglas en inglés) que la identifica unívocamente. (16)

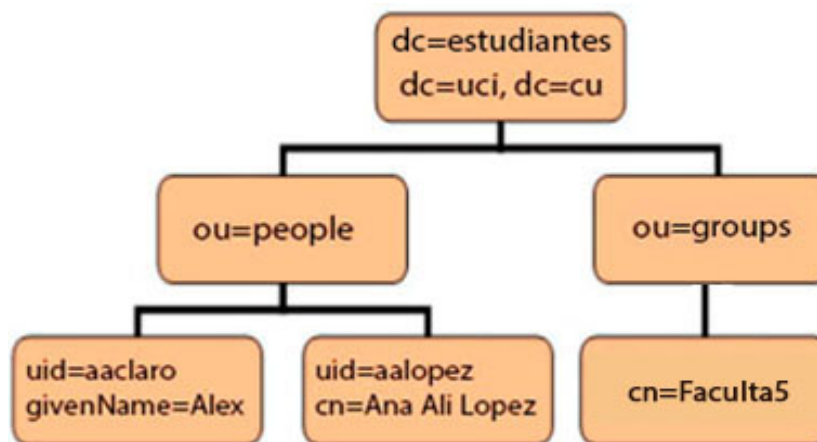


Figura 2: Ejemplo de la estructura de un árbol de directorio

El **dn** de la entrada que almacena los datos de Alex es: (Ver Figura 2) uid=aaclaro, ou=people, dc=estudiantes, dc=uci, dc=cu.

Los datos del directorio se representan mediante atributos y sus valores. Algunos de los tipos de atributos más comunes son:

- **dc:** *Domaincomponent*, componente de dominio.
- **uid:** Identificador de usuario.
- **samaccountname:** Atributo empleado para almacenar el identificador del usuario, en el caso del Directorio Activo se emplea el atributo.
- **ou:** *Organizationalunit*, unidad de la organización
- **c:** País.
- **cn:** *Commonname*, nombre común (nombre y apellidos)
- **givenname:** Nombre.
- **sn:** Apellidos.
- **mail:** Dirección de correo electrónico.

Cada entidad puede definirse con varios parámetros. El atributo *member* permite definir cuáles son los usuarios que pertenecen a un grupo. Por ejemplo, se puede definir el grupo Facultad 5 como en el listado siguiente, en este caso, solo el usuario *aaclaro* (línea 4) es miembro de este grupo.

- **dn:** cn=Facultad 5, ou=groups, dc=estudiantes, dc=uci, dc=cu.
- **objectclass:** AccessGroup.
- **cn:** Facultad 5.
- **member:** uid=aaclaro, ou=people, dc=estudiantes, dc=uci, dc=cu.

Los datos de un directorio LDAP pueden ser exportados a un fichero en Formato Ligero de Intercambio de Directorios (del inglés *Lightweight Directory Interchange Format*, LDIF), utilizado para realizar operaciones por lotes entre directorios. Puede utilizarse para exportar e importar datos del directorio y permite llevar a cabo operaciones como agregar, crear o modificar. (17)

1.4.6 Operaciones sobre el directorio LDAP

Una vez analizada la estructura del directorio resulta necesario conocer las posibles operaciones sobre este. Las operaciones que se realizan sobre el directorio LDAP se basan en el Modelo Cliente-Servidor. Cada cliente utiliza el protocolo LDAP para recuperar los datos almacenados en la base de datos del servidor. Entre las principales operaciones permitidas se encuentran las siguientes: (18)

- **Bind** (autenticación): Inicia la sesión de un usuario luego de reconocerlo como válido.
- **Add** (adiciona un elemento al directorio): Adiciona los registros del usuario a un directorio.
- **Search** (busca elementos en el directorio): Ejecuta la búsqueda con filtros o sin ellos.
- **Modify** (editar el contenido de un elemento): Modifica los registros y permite agregar nuevos atributos a estos.
- **Delete** (eliminar un elemento del directorio): Elimina uno o varios registros de usuario.
- **Unbind** (cierra la conexión): Cierra la sesión en el servidor y termina la conexión.

1.4.7 Ventajas de un servidor LDAP

La mayor ventaja de un servidor LDAP es que se puede consolidar información para toda una organización dentro de un repositorio central. Por ejemplo, en vez de administrar listas de usuarios

para cada grupo dentro de una organización, puede usar LDAP como directorio central, accesible desde cualquier parte de la red. Soporta la Capa de Conexión Segura (SSL) y la Seguridad de la Capa de Transporte (TLS). También, ya que LDAP tiene una interfaz de programación de aplicaciones (API) bien definida el número de aplicaciones acreditadas para LDAP son numerosas y aumentan en cantidad y calidad. (19)

Todo lo antes mencionado hace de los directorios LDAP un servicio con elevada aceptación por parte de los administradores de red. La integración de las aplicaciones con los directorios LDAP de una institución facilita la gestión de los usuarios, brinda un origen común para los datos de estos y un mecanismo de autenticación centralizado. Los datos almacenados en dichos directorios son empleados por las aplicaciones para la definición de la seguridad del sistema basada en usuarios y roles.

1.5 Herramientas con mecanismos similares

Una vez estudiada la estructura y funcionamiento de los directorios LDAP se han sentado las bases para analizar el estado del arte. Existen diversas herramientas que permiten acceder a la información contenida en los directorios LDAP o implementan algún mecanismo de integración con estos. Con la finalidad de aprovechar funcionalidades y/o conceptos que puedan utilizarse como partida para la solución que se pretende desarrollar se realizó un estudio de estas herramientas, tanto del ámbito internacional como nacional. A continuación se puede observar el resultado de la investigación realizada.

1.5.1 Ámbito internacional

Entre las herramientas internacionales se destacan:

- ✓ **phpLDAPadmin:** Este software es un cliente LDAP basado en la Web, que provee una administración fácil y accesible del servidor LDAP desde cualquier punto con acceso a la red. La herramienta permite administrar los registros del servidor, lo que incluye su creación, importación, exportación, modificación y eliminación. (20) (*Ver Anexo A.1*)
- ✓ **Webmin:** Una herramienta para la administración de servidores Unix accesible desde la Web. Permite controlar muchas aplicaciones, tales como el servidor Web Apache, MySQL, Samba, DHCP, OpenLDAP, entre otras. Webmin está escrito en el lenguaje de programación *Perl* y es multiplataforma. Brinda la posibilidad de manipular las entradas del directorio (adicionar, eliminar y modificar). (21) (*Ver Anexo A.2*)

Las dos herramientas antes mencionadas han sido empleadas en la investigación para analizar la estructura de un directorio LDAP, con el objetivo de comprender la organización de los usuarios dentro de este. Por otra parte, el estudio de estas herramientas ha aportado elementos conceptuales necesarios para el desarrollo de la solución. Sin embargo, el autor del presente trabajo considera que no son herramientas que puedan ser adaptadas a las necesidades de REKO, debido a que este posee una arquitectura muy peculiar, tema que se aborda con profundidad en el próximo capítulo.

- ✓ **KnowledgeBase Manager Pro:** Software que permite automatizar los procesos de gestión del conocimiento. Este software está desarrollado en el lenguaje de programación PHP. Este sistema cuenta con un módulo para la integración con servidores LDAP, a través de una interfaz bastante intuitiva donde se configuran los principales parámetros de conexión. (22) (Ver Anexo A.3)

Sin embargo, ha resultado imposible la obtención del código fuente de este sistema, ya que el software no es gratuito, lo que limita su estudio y empleo en la solución.

- ✓ **SmarterMail:** Desarrollado por la compañía SmarterTools. Este servidor de correo permite a sus administradores adicionar nuevos usuarios desde un Directorio Activo mediante el uso del protocolo LDAP. Esta característica puede ser usada para ahorrar tiempo siendo útil cuando se desea importar múltiples usuarios de un servidor LDAP existente. Los pasos necesarios para realizar la importación de usuarios en este sistema son: (23) (Ver Anexo A.4)
 - Primeramente se configuran los parámetros de conexión al servidor LDAP.
 - Luego se obtienen los datos de los usuarios y se muestran en un listado.
 - A continuación se seleccionan los usuarios que se desean importar.
 - Por último, se importan los usuarios al sistema.

Aunque esta herramienta cuenta con un mecanismo similar al que se necesita para dar solución al problema planteado, al ser privativa no brinda la posibilidad de reutilizar de forma parcial o total su código. Los pasos mencionados anteriormente representan una guía para el desarrollo de la solución.

Además de las herramientas antes mencionadas, existen *plugins* relacionados con el trabajo con directorios LDAP. A continuación se muestra una descripción de uno de los *plugin* analizados.

- ✓ **upSimpleLdapPlugin:** *Plugin* que brinda una clase para LDAP bastante fácil de utilizar. Cuenta con las siguientes características: (24)
 - Permite autenticar un usuario con las credenciales del directorio LDAP.
 - Permite obtener la información del directorio, dado un usuario específico.
 - Un usuario puede actualizar su contraseña.

Teniendo en cuenta las características antes mencionadas, se puede determinar que upSimpleLdapPlugin se enfoca en la obtención de los datos de un usuario específico y no en explorar todo el directorio como se desea.

1.5.2 Ámbito nacional

Entre las herramientas nacionales se destaca por las características que posee:

- ✓ **Módulo de directorio de la Plataforma de Gestión de Servicios Telemáticos en GNU/Linux:** Este módulo fue desarrollado en el lenguaje de programación *Python* por varios ingenieros de la UCI. Además de realizar las operaciones básicas para la gestión de un

servidor de directorio, también permite un conjunto de operaciones tales como las mencionadas a continuación: (25) (Ver Anexo A.5)

- La gestión de imágenes.
- La migración de datos de un Directorio Activo de Windows al servidor OpenLDAP.
- La gestión de más de un servidor de directorio.

A pesar de que la aplicación permite la migración antes mencionada, no cuenta con un mecanismo que le permita importar los usuarios a una base de datos embebida como la de REKO. Además, el hecho de ser una aplicación de escritorio, dificulta su integración con la consola de administración Web de REKO. Las condiciones anteriores impiden su empleo en la solución.

Las herramientas anteriormente analizadas tanto del ámbito internacional como nacional, poseen características que impiden su utilización en la solución al problema presente. Sin embargo, el estudio de estas herramientas aportó elementos teóricos y prácticos necesarios para la materialización de la solución deseada. Finalmente, no se encontró una herramienta que pueda ser utilizada de forma íntegra en la solución, por lo que se evidencia la necesidad de realizar una implementación propia.

1.6 Metodologías de desarrollo

Las metodologías para el desarrollo imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo, haciendo énfasis en la calidad y menor tiempo de construcción del software o lo que es lo mismo “producir lo esperado en el tiempo esperado y con el coste esperado”. (26)

En los últimos tiempos la cantidad y variedad de metodologías de software han aumentado de manera impresionante. Han surgido dos corrientes, los llamados métodos pesados y los métodos ágiles. La diferencia fundamental entre ellos es que, mientras los métodos pesados buscan cumplir el objetivo común mediante el orden y la documentación, los métodos ágiles constituyen una solución a la medida para entornos cambiantes, lo cual aporta una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

1.6.1 Metodologías tradicionales

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas, notaciones para el modelado y documentación detallada. (27)

Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar.

1.6.2 Metodologías ágiles

Las metodologías ágiles son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno. En esencia, las empresas que apuestan por esta metodología consiguen gestionar sus proyectos de forma eficaz reduciendo los costes e incrementando su productividad. (28)

Debido a que el equipo de desarrollo está compuesto por una persona, el tiempo disponible es pequeño y se trabaja en un entorno donde los requisitos pueden variar, se decide el uso de una metodología ágil.

Para la extensión del módulo de seguridad se emplea la metodología de desarrollo AUP adaptada al ciclo de vida definido para la actividad productiva de la UCI, siendo esta la designada por la universidad para guiar los procesos de desarrollo de software.

1.6.2.1 AUP

El Proceso Unificado Ágil (del inglés *Agile Unified Process*, AUP) es una versión simplificada del Proceso Unificado de Rational (del inglés *Rational Unified Process*, RUP) desarrollada por Scott Ambler, que describe una aproximación al desarrollo de aplicaciones que combina conceptos propios del proceso unificado tradicional con técnicas ágiles, con el objetivo de mejorar la productividad.

En general, AUP supone un enfoque intermedio entre la Programación Extrema (del inglés *eXtreme Programming*, XP) y RUP, y tiene la ventaja de ser un proceso ágil que incluye explícitamente actividades y artefactos a los que la mayoría de los desarrolladores ya están acostumbrados. (29)

Características de AUP (29)

- ✓ Versión simplificada de la metodología RUP.
- ✓ Abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de Configuración, Gestión de Proyectos y Entorno.
- ✓ El modelado agrupa los tres primeros flujos de RUP (Modelado de Negocio, Requerimientos y Análisis y Diseño).
- ✓ Dispone de cuatro fases igual que RUP: Incepción o Creación, Elaboración, Construcción y Transición.

Principios de AUP (30)

- ✓ Simplicidad: Todo se describe concisamente utilizando poca documentación.
- ✓ Centrarse en actividades de alto valor: La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.
- ✓ Herramienta de la independencia: Usted puede usar cualquier conjunto de herramientas que desea con la metodología AUP. Se sugiere utilizar las herramientas más adecuadas para el trabajo, que a menudo son las herramientas simples o incluso herramientas de código abierto.

Ciclo de vida de AUP (29)

El Proceso Unificado Ágil consta de cuatro fases que el proyecto atraviesa de forma secuencial. Dichas fases son, al igual que en RUP:

- ✓ **Iniciación:** El objetivo de esta fase es identificar el alcance inicial del proyecto, una arquitectura potencial para el sistema y obtener, si procede, financiación para el proyecto y la aceptación por parte de los promotores del sistema.
- ✓ **Elaboración:** Mediante esta fase se pretende identificar y validar la arquitectura del sistema.
- ✓ **Construcción:** El objetivo de esta fase consiste en construir software desde un punto de vista incremental basado en las prioridades de los participantes.
- ✓ **Transición:** En esta fase se valida y despliega el sistema en el entorno de producción.

De las cuatro fases que propone AUP (Inicio, Elaboración, Construcción y Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes tres fases de AUP en una sola, a la que se le llama Ejecución y se agrega una fase de Cierre. (Ver Tabla 2)

Tabla 2: Fases de AUP-UCI.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración Construcción Transición	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Fuente: (31)

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de Configuración, Gestión de Proyectos y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de

Negocio, Requisitos y Análisis y Diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplina. (Ver Tabla 3)

Tabla 3: Disciplinas de AUP-UCI

Disciplinas AUP	Disciplinas Variación AUP-UCI	Objetivos Disciplinas (Variación AUP-UCI)
Modelo	Modelado de Negocio (opcional)	Disciplina destinada a comprender cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito.
	Requisitos	Desarrollar un modelo del sistema que se va a construir. Comprende la administración de los requisitos funcionales y no funcionales del producto.
	Análisis y Diseño	Los requisitos pueden ser refinados y estructurados para conseguir una comprensión y descripción más precisa de estos. Se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales.
Implementación	Implementación	En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
Prueba	Prueba Interna	Se verifica el resultado de la implementación probando cada construcción. Se deben desarrollar artefactos de prueba como: Diseños de casos de prueba, listas de chequeo y componentes de prueba ejecutables para automatizar las pruebas.
	Prueba de Liberación	Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
	Prueba de aceptación	Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y puede ser usado por usuarios finales para ejecutar aquellas funciones para las cuales fue construido.
Despliegue	Despliegue (opcional)	Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente.

Fuente: (31)

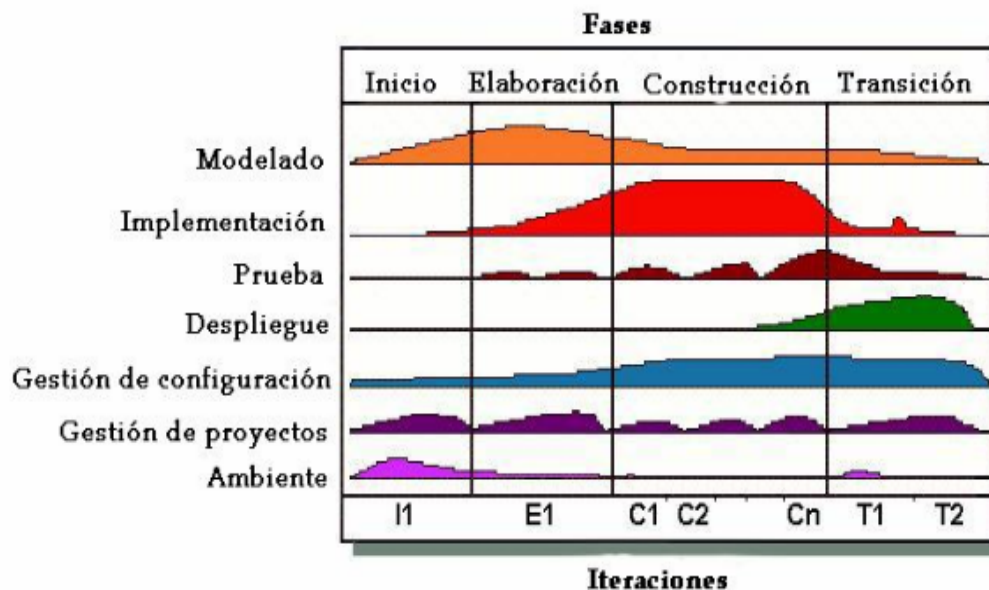


Figura 3: Disciplinas desarrolladas en cada fase según AUP. (30)

Descripción de roles

AUP propone 9 roles (Administrador de proyecto, Ingeniero de procesos, Desarrollador, Administrador de BD, Modelador ágil, Administrador de la configuración, Stakeholder, Administrador de pruebas, Probador), se decide para el ciclo de vida de los proyectos de la UCI tener 11 roles, manteniendo algunos de los propuestos por AUP y unificando o agregando otros. (Ver Tabla 4)

Tabla 4: Roles de AUP-UCI

Roles AUP	Roles Variación AUP-UCI
Administrador de proyecto	Jefe de proyecto
	Planificador
Ingeniero de proceso	Analista
Modelador ágil	Arquitecto de información (Opcional)
Desarrollador	Desarrollador
Administrador de la configuración	Administrador de la configuración
Stakeholder	Stakeholder (Cliente/Proveedor de requisitos)
Administrador de pruebas	Administrador de la calidad
Probador	Probador
Administrador de BD	Arquitecto de software (Sistema)
	Administrador de BD

Fuente: (31)

Todas las disciplinas antes definidas (desde Modelado de Negocio hasta Despliegue) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen iteraciones y se obtengan resultados incrementales. (Ver Figura 4)

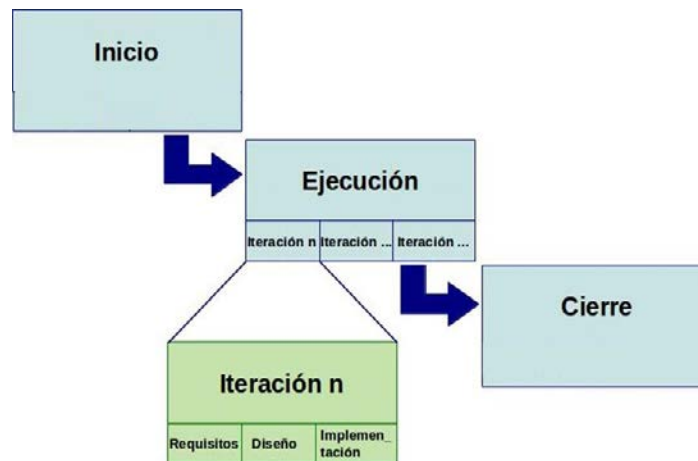


Figura 4: Fases e iteraciones de AUP. (31)

Ventajas y desventajas de AUP (29)

Ventajas:

- ✓ Es una metodología ágil.
- ✓ Se puede adaptar con otros procesos.

Desventajas:

- ✓ A veces omite contenido que puede ser de interés en el proyecto.
- ✓ Se espera que cubra un amplio sistema de necesidades para los proyectos de desarrollo en un plazo muy corto.

AUP permite abordar ágilmente el desarrollo de proyectos pequeños y tiene un enfoque centrado en el cliente con iteraciones cortas. Este proceso de desarrollo unificado está basado en las mejores prácticas de RUP, lo que permite que el tiempo de capacitación (curva de aprendizaje) del personal sea mínimo ya que el mismo posee experiencia con el uso de RUP. Permite el desarrollo de artefactos ligeros y apropiados, utilizando el Lenguaje de Modelado Unificado (del inglés *Unified Modeling Language*, UML), aplica un enfoque iterativo e incremental dentro de su estructura del ciclo de vida y tiene la peculiaridad de tener solamente cuatro fases. AUP permite disminuir las probabilidades de fracaso e incrementa las probabilidades de éxito y permite detectar errores tempranos a través de un ciclo iterativo.

1.7 Técnicas, métodos, herramientas y lenguajes para el desarrollo

Para el desarrollo del módulo se emplearon las técnicas, métodos y herramientas definidas en la arquitectura de REKO que a continuación se describen.

1.7.1 Técnicas y métodos

✓ Programación Orientada a Objetos (POO)

La POO es una de las formas más populares de programar, la cual tiene gran acogida en el desarrollo de proyectos de software desde los últimos años. Esta acogida se debe a sus grandes capacidades y ventajas frente a las antiguas formas de programar. La POO viene de la evolución de la programación estructurada; básicamente simplifica la programación con la nueva filosofía y nuevos conceptos que tiene. Se basa en dividir el programa en pequeñas unidades lógicas de código denominadas objetos. (32)

✓ **Programación Orientada a Aspectos (POA)**

La POA permite a los programadores escribir, ver y editar un aspecto diseminado por todo el sistema como una entidad por separado, de una manera inteligente, eficiente e intuitiva.

Es un nuevo paradigma de programación que aspira a soportar la separación de propiedades. Esto implica separar la funcionalidad básica y los aspectos entre sí, a través de mecanismos que permitan abstraerlos y componerlos para formar todo el sistema. (33)

Permite insertar funcionalidades sobre otras ya implementadas sin necesidad de modificar el código original. (34)

1.7.2 Herramientas y lenguajes de desarrollo

Tabla 5: Herramientas y lenguajes para el desarrollo.

Frameworks	
Nombre	Descripción
Spring 2.0	Es un <i>framework</i> (marco de trabajo) de aplicación de código abierto que ayuda a hacer el desarrollo en JEE mucho más fácil. Ayuda a estructurar aplicaciones completas en una manera consistente y productiva para crear arquitecturas coherentes. Presenta un diseño que brinda una gran flexibilidad arquitectónica e interviene en todas las capas de una aplicación. (35) El mismo presenta varios módulos de los cuales los más importantes son: <ul style="list-style-type: none"> • Acceso a datos / Integración • Web • Módulo para la Programación Orientada a Aspectos (POA) • Pruebas
Dojo 1.2.0	Dojo es un <i>framework</i> que posee Interfaz de Programación de Aplicaciones (del inglés <i>Application Programming Interface, API</i>) y <i>Widgets</i> (pequeña aplicación o programa) para facilitar el desarrollo de aplicaciones Web modernas. Tiene como características principales que es proyecto de código abierto, presenta una multitud de componentes visuales que permiten el desarrollo rápido de interfaces de usuario complejas y permite importar Hojas de Estilo en Cascada (del inglés <i>Cascading Style Sheets, CSS</i>). (35)
Acegi Security 1.0.7	Acegi Security es un <i>framework</i> de código abierto altamente usado por la comunidad de Spring para proveer los servicios de seguridad a las aplicaciones basadas principalmente en Spring Framework. Es utilizado para asegurar numerosos entornos exigentes, incluyendo organismos gubernamentales, y bancos centrales. Permite a las aplicaciones sobre Java aplicar los tres requerimientos de seguridad más comunes de las aplicaciones empresariales: autenticación, seguridad sobre las peticiones web y seguridad a las instancias de los objetos de dominio. (35)
JSON	Notación de Objetos de JavaScript (del inglés <i>JavaScript Object Notation, JSON</i>) es un formato de intercambio de datos de peso ligero. Es completamente independiente del lenguaje, pero usa convenciones que son familiares a los

	programadores de los lenguajes de la familia de C, como son C, C++, C#, Java, JavaScript, Perl, Python, entre otros. Estas propiedades hacen a JSON un lenguaje de intercambio de datos ideal. (35)
JUnit V 4	<i>Framework</i> de código abierto para diseñar, construir y ejecutar pruebas unitarias de aplicaciones Java. Produce varios tipos de reportes a partir de los resultados obtenidos en la ejecución de las pruebas. Soporta la jerarquía entre las pruebas pudiendo establecerse la prioridad y el orden de unas con otras. (36)
Herramientas	
Nombre	Descripción
Apache Tomcat 7.0	Apache Tomcat es un contenedor de <i>servlet</i> usado en la implementación de referencia oficial para la tecnología Java Server Pages (JSP). Es desarrollado en un ambiente participativo y abierto. (37)
IDE de desarrollo	
Nombre	Descripción
Eclipse Kepler Release	Eclipse es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios Web, programas en C++ o Java. Es un Entorno de Desarrollo Integrado (del inglés <i>Integrated Development Environment</i> , IDE) en el que se encuentran todas las herramientas y funciones necesarias para el trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar. (38)
Herramienta CASE	
Nombre	Descripción
Visual Paradigm para UML 8.0	Herramienta profesional que soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases, código inverso y generar documentación. Cuenta con la ventaja de ser una herramienta multiplataforma y permite la integración con Eclipse IDE. (39) Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar, y compatible entre ediciones. Genera código para Java y exportación como HTML.
Plataforma de desarrollo	
Nombre	Descripción
JEE	Java EE (del inglés <i>Java Platform Enterprise Edition</i> , JEE) es un entorno independiente de la plataforma centrado en Java para desarrollar, crear e implementar en línea aplicaciones empresariales basadas en Web. Consta de un conjunto de servicios, APIs y protocolos que proporcionan las funcionalidades necesarias para el desarrollo. (40)
Lenguaje de programación	
Nombre	Descripción
Java 7.0	Lenguaje de programación orientado a objetos. Los programas que se crean son portátiles en una red. Su programa fuente se compila en lo que se llama Java <i>bytecode</i> , que se puede ejecutar en cualquier parte de una red que contenga una

	máquina virtual Java. (41)
Lenguaje de modelado	
Nombre	Descripción
UML 2.0	Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. (42)

Fuente: Elaboración propia.

1.8 Conclusiones parciales

A partir del análisis e investigación de los aspectos abordados en este capítulo se puede concluir que:

- ✓ Atendiendo al contexto de la problemática planteada, no es viable la utilización total o parcial de herramientas que implementan mecanismo de integración con directorios LDAP que ya han sido desarrolladas, por lo que se propone una implementación propia.
- ✓ Se describieron las principales características de las herramientas y metodología que se usan en el desarrollo de la aplicación, las cuales guían el proceso de desarrollo.
- ✓ La implementación de un módulo que permita importar y sincronizar los usuarios de un servidor LDAP en REKO aumenta la seguridad de dicho sistema, permitiendo a su vez automatizar el proceso de creación de cuentas de usuario.

CAPÍTULO 2. CARACTERÍSTICAS Y DISEÑO DEL SISTEMA

En el presente capítulo se describe la propuesta de solución para la extensión del módulo de seguridad del Replicador de datos REKO. Para ello se detalla el Modelo de dominio, se definen los requisitos funcionales y no funcionales así como se identifican los actores del sistema. Además, se presentan los diagramas de clases, favoreciendo un mejor entendimiento para el desarrollo de la solución.

2.1 Flujo actual de los procesos

Para comprender el problema planteado se hace necesario describir el flujo actual de los procesos. El Replicador de datos REKO cuenta con un mecanismo de gestión de usuarios interno e independiente, basado en la autenticación por roles. Este mecanismo sienta las bases para toda la administración de los permisos de navegación de cada usuario, asignándole roles a cada uno, y dándole permisos a los roles sobre los distintos módulos.

Toda la información de estos usuarios se encuentra almacenada en la BD embebida de REKO, de esta forma, para habilitar la navegación a cierto usuario este debe primeramente estar registrado en el sistema. El proceso de creación de cuentas de usuarios en REKO se realiza de forma manual, a través de un formulario donde se introducen los datos del usuario y se le asignan los roles determinados. (Ver Figura 5)

The image shows a web application interface titled "Gestionar Seguridad". It contains two main sections:

- Gestionar Usuario:** This section has a form with fields for "Nombre", "Contraseña", "Repita la Contraseña", and "Descripción". To the right, under "Roles Existentes", there is a list of roles with checkboxes: ROLE_ADMIN, ROLE_EXPORTACION, ROLE_INICIO, ROLE_INICIO2, ROLE_INICIO_3, ROLE_INICIO_MONITOR, ROLE_MONITOR, and ROLE_SINCRONIZACION. At the bottom of this section are buttons for "Adicionar", "Cancelar", and "Confirmar".
- Gestionar Rol:** This section has a form with fields for "Nombre" and "Descripción". To the right, under "Módulos", there is a list of modules with expandable arrows: Configuración, Sincronización, Inicio, Exportar, Importar, Seguridad, Monitor, and Conflictos. At the bottom of this section are buttons for "Adicionar", "Cancelar", and "Confirmar".

Figura 5: Formulario para la creación de cuentas de usuario en REKO.

2.2 Propuesta de solución

Se propone extender el módulo para la seguridad del Replicador de datos REKO de tal forma que permita filtrar e importar los usuarios de un servidor LDAP hacia la BD embebida de REKO, para automatizar el proceso de creación de cuentas de usuarios. El sistema debe permitir realizar el proceso de sincronización, con el objetivo de mantener actualizados los datos de los usuarios de REKO con respecto a los nuevos cambios que ocurran en el servidor y poder gestionar los parámetros de configuración de la conexión para el servidor LDAP existentes en la institución. Unido a todo esto, el usuario podrá autenticarse a través de la BD embebida de REKO y a través de un servidor LDAP.

2.3 Modelo de dominio

Un Modelo de dominio captura los tipos más importantes de objetos en el entorno del sistema. Los objetos o clases del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que el sistema trabaja. Se describe mediante diagramas de UML que muestran las clases del dominio y como se relacionan unas con otras mediante asociaciones. (43)

Dicho modelo permite describir el estado del problema planteado, así como obtener una vista estática vinculada con las reglas del negocio. (Ver Figura 6)

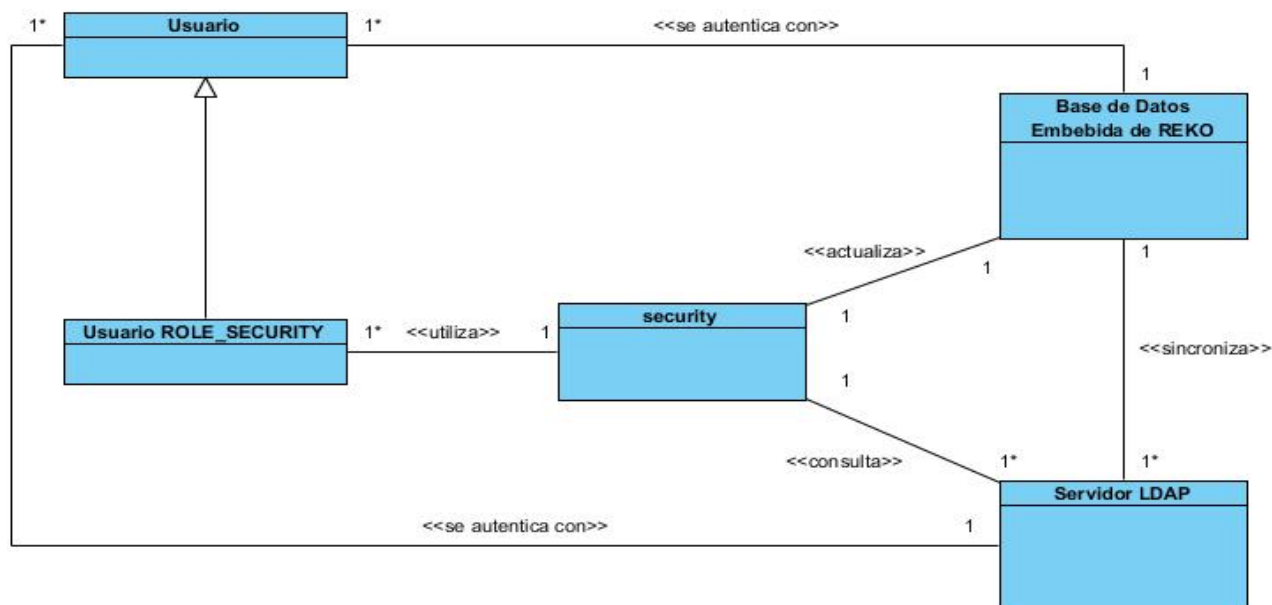


Figura 6: Diagrama del Modelo de dominio.

2.3.1 Descripción del diagrama de clases del Modelo de dominio

A continuación se describen los conceptos empleados en el diagrama mostrado:

- ✓ **Usuario:** Cualquier persona que solicita acceder al Replicador de datos REKO.
- ✓ **Usuario ROLE_SECURITY:** Usuario que utiliza el módulo propuesto, desempeña el rol de administrador o rol de seguridad.
- ✓ **security:** Paquete *security*, perteneciente al componente Administración, donde se realizan los cambios necesarios.

- ✓ **BD embebida de REKO:** BD embebida propia de REKO, en la que se encuentra los datos de los usuarios del sistema.
- ✓ **Servidor LDAP:** Servidor que brinda el servicio de directorio basado en el protocolo LDAP y contiene la información de los usuarios de la institución.

2.4 Modelado del sistema

La especificación de requisitos en el proceso de desarrollo del software es de vital importancia. Tener los requisitos bien definidos permite comprender desde un inicio la línea a seguir en el desarrollo y así garantizar la eficiencia y calidad del software.

2.4.1 Requisitos Funcionales

Los Requisitos Funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los RF de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (44)

Tabla 6: Catálogo de los RF.

No	Nombre	Descripción	Complejidad
[RF1.]	Editar configuración de conexión.	Permite establecer y editar los parámetros de configuración de conexión a un servidor LDAP.	Alta
[RF2.]	Autenticar con LDAP.	Permite la autenticación a través de un servidor LDAP.	Alta
[RF3.]	Filtrar usuarios del LDAP.	Permite filtrar usuarios del servidor LDAP.	Alta
[RF4.]	Importar usuarios filtrados.	Permite importar los usuarios obtenidos como resultado del filtrado.	Alta
[RF5.]	Sincronizar servidor LDAP con BD embebida en REKO.	Permite realizar el proceso de sincronización teniendo en cuenta los parámetros de conexión.	Media
[RF6.]	Buscar usuarios	Permite realizar la búsqueda de usuarios en el sistema, para eliminarlos o modificarlos.	Baja
[RF7.]	Buscar roles	Permite realizar la búsqueda de roles en el sistema, para eliminarlos o modificarlos.	Baja

Fuente: Elaboración propia.

2.4.2 Requisitos No Funcionales

Los Requisitos No Funcionales (RNF) son restricciones de los servicios o funciones ofrecidos por el sistema. Los RNF a menudo se aplican al sistema en su totalidad, normalmente apenas se aplican a características o servicios individuales del sistema. Son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema. (44)

Los RNF del módulo a extender son equivalentes a los del software al que se integra, REKO.

Tabla 7: Catálogo de los RNF.

No	Nombre	Descripción
[RNF1.]	Usabilidad	<p>El módulo debe ser de fácil manejo para los usuarios que tengan niveles básicos de trabajo con ordenadores y en la Web.</p> <ul style="list-style-type: none"> - <u>Operabilidad</u>: Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad. - <u>Protección contra errores de usuarios</u>: Capacidad del producto para proteger a los usuarios de cometer errores.
[RNF2.]	Fiabilidad	<p>El módulo debe ser capaz de recuperarse ante fallos, así como notificar al usuario sobre cualquier error o excepción que ocurra durante la ejecución de las acciones solicitadas.</p> <ul style="list-style-type: none"> - <u>Tolerancia a fallos y recuperabilidad</u>: Capacidad del producto para operar según lo previsto en presencia de fallos de hardware o software. - <u>Disponibilidad</u>: Capacidad del producto de estar operativo y accesible para su uso cuando se requiere.
[RNF3.]	Mantenibilidad	<p>Mide la facilidad con que puede darse mantenimiento al producto con la finalidad de desarrollar nuevos requerimientos, corregir defectos y atender las demandas del entorno cambiante.</p> <ul style="list-style-type: none"> - <u>Analizabilidad</u>: Facilidad con la que se puede evaluar el impacto de un determinando cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar. - <u>Modificabilidad</u>: Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño. - <u>Capacidad para ser probado</u>: Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.
[RNF4.]	Portabilidad	<p>El módulo debe ser capaz de adaptarse de forma efectiva y eficiente a diferentes entornos.</p> <ul style="list-style-type: none"> - <u>Adaptabilidad</u>: Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso. - <u>Instalabilidad</u>: Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.
[RNF5.]	Compatibilidad	<ul style="list-style-type: none"> - <u>Co-existencia</u>: Grado en que un producto puede realizar sus funciones necesarias de manera eficiente mientras comparte un

		entorno común y recursos con otros productos, sin impacto perjudicial en cualquier otro producto.
[RNF6.]	Adecuación funcional	<p>El desarrollo del módulo está guiado por las necesidades expresadas por parte de los proveedores de requisitos, dándole total cumplimiento a sus especificaciones declaradas e implícitas.</p> <ul style="list-style-type: none"> - <u>Integridad funcional</u>: Grado en el que el conjunto de funciones cubre todas las tareas y objetivos del usuario especificado. - <u>Corrección funcional</u>: Grado en que un producto o sistema proporciona los resultados correctos con el grado necesario de precisión.
[RNF7.]	Seguridad	<p>Grado de protección de los datos, software y plataforma de tecnología de posibles pérdidas, actividades no permitidas o uso para propósitos no establecidos previamente.</p> <ul style="list-style-type: none"> - <u>Confidencialidad</u>: La información manejada por el sistema está protegida de acceso no autorizado y divulgación. - <u>Integridad</u>: La información manejada por el sistema es objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma es considerada igual a la fuente de los datos.
[RNF8.]	Eficiencia en el rendimiento	<p>El módulo debe estar concebido para el consumo mínimo de recursos. Los clientes no necesitarán más de 128 MB de RAM, lo suficiente para ejecutar un navegador Web.</p> <ul style="list-style-type: none"> - <u>Comportamiento en el tiempo</u>: Grado en que los tiempos de respuesta, procesamiento y las tasas de rendimiento de un producto o sistema, al realizar sus funciones cumplen con los requisitos. - <u>Utilización de recursos</u>: Grado en el que las cantidades y tipos de recursos utilizados por un producto o sistema, al realizar sus funciones cumplen con los requisitos.

Fuente: Elaboración propia.

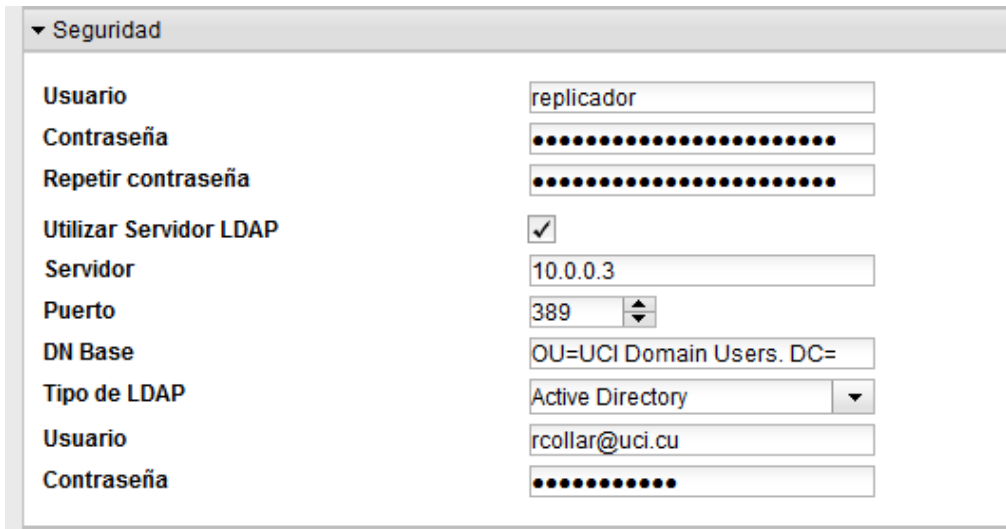
La aplicación de los RNF antes mencionados se puede observar en los Anexos del F1 al F7.

2.4.3 Historias de Usuario

Una Historia de Usuario (HU) es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario. Son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos y son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Permiten responder rápidamente a los requisitos cambiantes. (45)

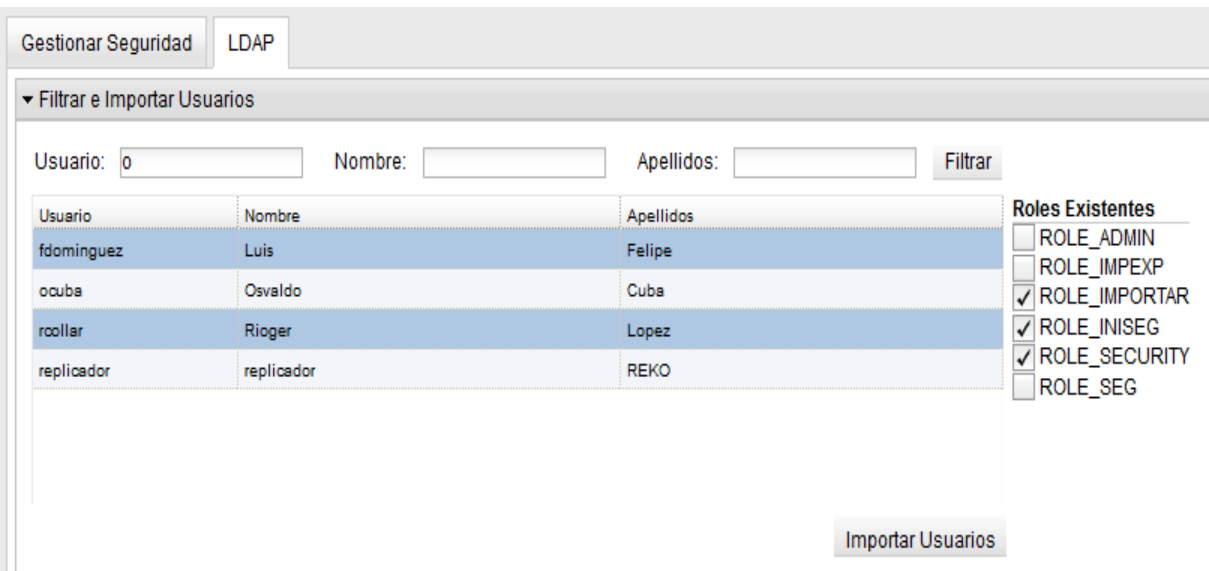
2.4.4 Descripción de las Historias de Usuario

Tabla 8: HU1. Editar configuración de conexión

Historias de Usuario	
Número: HU1	Nombre del requisito: Editar configuración de conexión.
Programador: Rioger López Collar	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en Desarrollo: <ul style="list-style-type: none"> • Problemas eléctricos. • Problemas técnicos de los servicios de redes. 	Tiempo Real: 1 semana
Descripción: En la consola de administración Web del Replicador de datos REKO, dentro del módulo correspondiente a la “Configuración General” en la sección “Seguridad”, específicamente en “Utilizar Servidor LDAP” se deben establecer los parámetros de configuración de la conexión a un servidor LDAP específico (Servidor, Puerto, DN Base, Tipo, Usuario y Contraseña). Estos parámetros correctamente configurados son empleados más adelante a la hora de filtrar e importar usuarios del servidor, en la sincronización de la BD embebida de REKO con dicho servidor y para la autenticación de los usuarios. Una vez guardados los parámetros de configuración, los mismos son escritos en un archivo denominado replication.properties donde se guardan todas las configuraciones de REKO. Estos parámetros pueden ser leídos y modificados. A través de los botones “Guardar” o “Cancelar” puede realizar la acción de edición o no de los parámetros de configuración.	
Observaciones: Ninguno de los campos debe estar vacío, sino el sistema muestra una alerta “Llene los campos requeridos”.	
Prototipo de interfaz: 	

Fuente: Elaboración propia.

Tabla 9: HU4. Importar usuarios filtrados

Historias de Usuario	
Número: HU4	Nombre del requisito: Importar usuarios filtrados.
Programador: Rioger López Collar	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 14 días
Riesgo en Desarrollo: <ul style="list-style-type: none"> • Problemas eléctricos. • Problemas técnicos de los servicios de redes. 	Tiempo Real: 2 semanas
Descripción: Una vez filtrados los usuarios del servidor LDAP, en el módulo de “Seguridad” en la sección “LDAP” el administrador del sistema tiene la posibilidad de seleccionar los usuarios y los roles correspondiente que desea importar (persistir en la BD embebida). Cada usuario seleccionado debe poseer un rol determinado y no puede existir previamente en la BD embebida de REKO, el mismo una vez importado posee acceso a la consola de administración Web de REKO según sus permisos. Para la ejecución del proceso de importación debe presionar el botón “Importar Usuarios”. El sistema muestra un mensaje “Importación satisfactoria”.	
Observaciones: <ul style="list-style-type: none"> • Se debe seleccionar el menos un rol para los usuarios, sino el sistema muestra una alerta “Debe seleccionar el menos un rol”. • Se debe seleccionar el menos un usuario a importar, sino el sistema muestra una alerta “Debe seleccionar el menos un usuario a importar”. 	
Prototipo de interfaz: 	

Fuente: Elaboración propia.

La descripción de las HU restantes se pueden observar en los Anexos del B.1 al B.5.

2.5 Descripción de la arquitectura del Replicador de datos REKO

El Instituto de Ingeniería Eléctrica y Electrónica (del inglés *Institute of Electrical and Electronics Engineers*, IEEE) define a la arquitectura de software como: “La organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.” (46)

2.5.1 Estilos arquitectónicos

Un estilo arquitectónico se define como: “Una lista de tipos de componentes que describen los patrones o las interacciones a través de ellos. Un estilo afecta a toda la arquitectura de software y puede combinarse en la propuesta de solución.” (47)

El estilo arquitectónico “Llamada y retorno” es el utilizado en la solución propuesta ya que se obtiene una estructura del programa que resulta relativamente fácil de modificar y cambiar de tamaño. Los datos son pasados como parámetros y el manejador principal proporciona un ciclo de control sobre las subrutinas. Son los estilos más generalizados en sistemas en gran escala. Miembros de la familia son: (46)

- ✓ Arquitecturas en capas.
- ✓ Modelo Vista Controlador.
- ✓ Sistemas orientados a objeto.
- ✓ Sistemas basados en componentes.

REKO presenta una arquitectura que puede definirse como basada en componentes, debido a que todas sus partes encapsulan un conjunto de comportamientos que pueden ser reemplazados por otros. Se descompone en componentes lógicos o funcionales y se comunican a través de interfaces bien definidas que contienen métodos, eventos y propiedades. Los principales componentes presentes en el software son: (Ver Figura 7)

- ✓ **Capturador_Cambios:** Encargado de capturar los cambios que se realizan en la BD y entregarlos al Distribuidor.
- ✓ **Distribuidor:** Determina el destino de cada cambio realizado en la BD, los envía y se responsabiliza de su llegada.
- ✓ **Aplicador:** Ejecuta en la BD los cambios que son enviados hacia él desde otro nodo de réplica.
- ✓ **Administrador:** Permite realizar las configuraciones principales de REKO, como el registro de nodos, configuración de las tablas a replicar y el monitoreo del funcionamiento.

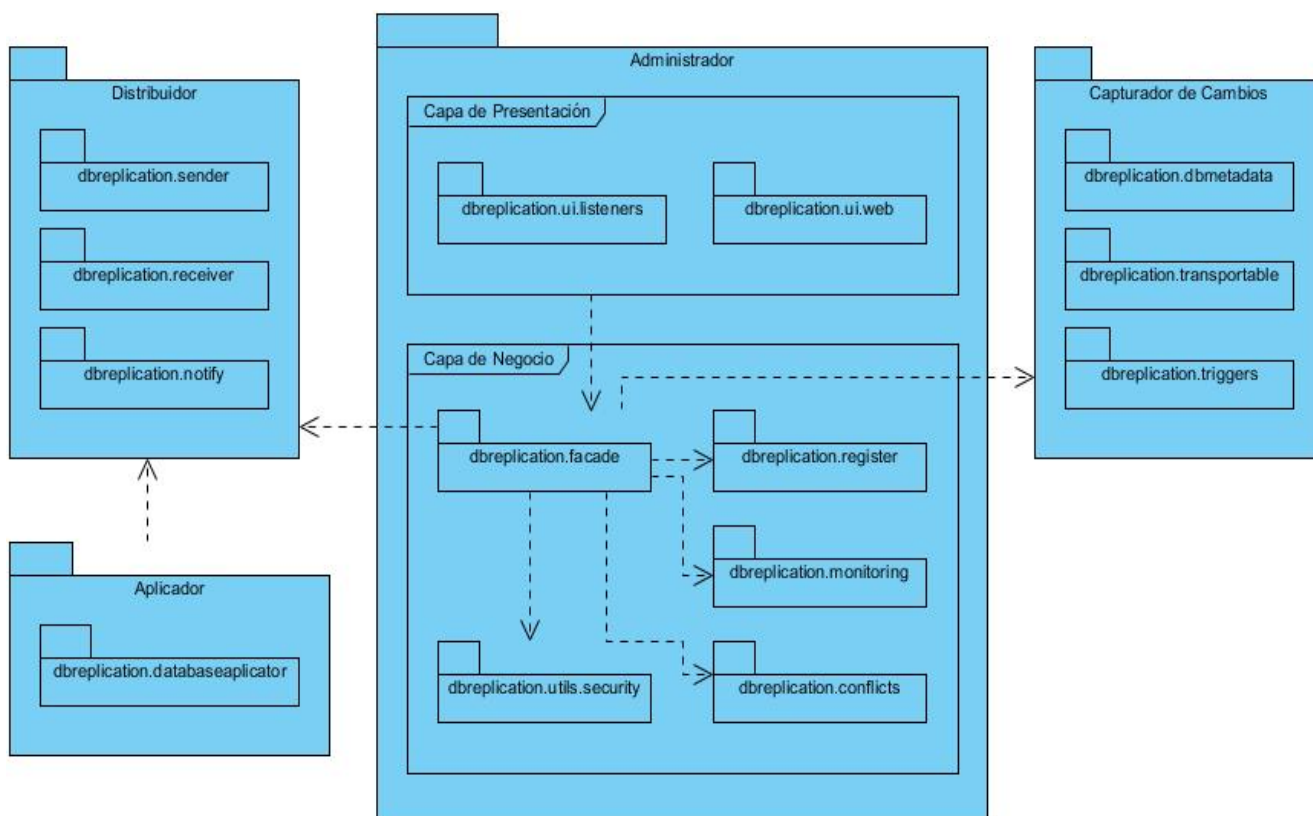


Figura 7: Vista de los principales componentes de REKO.

Independientemente de lo antes expuesto, el componente Administrador responde a un modelo multicapa, donde cada capa tiene funcionalidades y objetivos precisos, así su implementación se encuentra desacoplada de la programación de cualquier otra y la comunicación con una capa inferior ocurre a través de interfaces.

La **Capa de Presentación** es la que el usuario ve en su ordenador, es donde se tratan los datos que se van a mostrar. Esta capa se comunica únicamente con la Capa de Negocio.

La **Capa de Negocio** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina además como lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. (46)

El objetivo primordial de este modelo es la separación de la lógica de negocios y la lógica de diseño, es decir, separar la Capa de Negocios de la Capa de Presentación al usuario. La ventaja principal es que el desarrollo se puede llevar a cabo en dos niveles, en caso de cambios, solo se afecta al nivel requerido, sin tener que revisar todo el código.

Para realizar la extensión del módulo de seguridad se hizo necesario hacer cambios en el paquete *security*, perteneciente al componente Administrador, donde hasta el momento existía una implementación de este módulo basada en el mecanismo de autenticación por roles, por lo que no existían funcionalidades capaces de adaptarse a nuevos mecanismos de seguridad como son los servicios de directorios LDAP. (Ver Figura 8)

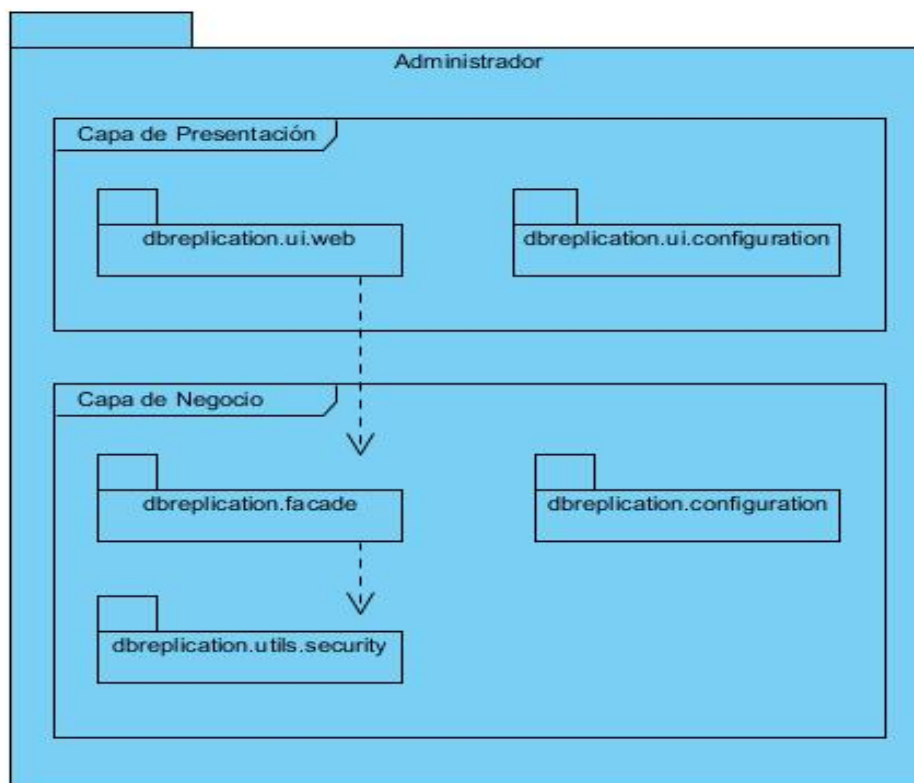


Figura 8: Arquitectura de la seguridad en REKO.

2.5.2 Patrones

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a ese problema, de tal manera que puedes usar esa solución un millón de veces más, sin hacer jamás la misma cosa dos veces. (48)

Según la escala o nivel de abstracción presentan tres categorías:

- ✓ Patrones arquitectónicos: Estos patrones definen la estructura general del software, indican las relaciones entre los subsistemas y los componentes del software y definen las reglas para especificar las relaciones entre los elementos (clases, paquetes, componentes, subsistemas) de la arquitectura.
- ✓ Patrones de diseño: Estos patrones se aplican en un elemento específico del diseño como un agregado de componentes para resolver algún problema de diseño, relaciones entre los componentes o los mecanismos para efectuar la comunicación de componente a componente.
- ✓ Idiomas: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

2.5.3 Patrones arquitectónicos

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionando un conjunto de sub-sistemas predefinidos, especificando

sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos. (49)

2.5.3.1 Arquitectura en capas

Los sistemas o arquitecturas en capas constituyen uno de los estilos que aparecen con mayor frecuencia mencionados en Internet. El estilo en capas es definido como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inferior. Esta se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades.

Capas:

- ✓ **Capa de Presentación:** Es la que ve el usuario, captura la información del mismo en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la Capa de Negocio. (50)
- ✓ **Capa de Negocio:** Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la Capa de Presentación, para recibir las solicitudes y presentar los resultados, y con la Capa de Datos, para solicitar al gestor de BD almacenar o recuperar datos de él. (50)
- ✓ **Capa de Datos:** Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de BD que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la Capa de Negocio. (50)

2.5.4 Patrones de diseño

Los patrones de diseño constituyen 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. Es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. (48)

2.5.4.1 Patrón de Acceso a Datos

Los Objetos de Acceso a Datos (del inglés, *Data Access Object*, DAO) vienen a resolver el problema de contar con diversas fuentes de datos (BD, archivos, servicios externos, etc.), de tal forma que se encapsula y oculta la forma de acceder a estos. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento. El DAO tiene una interfaz común, sea cual sea el modo y fuente de acceso a datos. No es imprescindible, pero en proyectos de cierta complejidad resulta útil que el DAO implemente una interfaz. De esta forma, los objetos cliente tienen una forma unificada de acceder a los DAO. (51)

2.5.4.2 Patrones GRASP

Los Patrones de Software para Asignar Responsabilidades (del inglés *General Responsibility Assignment Software Patterns*, GRASP) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática y racional. (52)

- ✓ **Experto:** Asigna la responsabilidad de crear objetos e implementar métodos a la clase que contiene la información necesaria para realizar estas acciones. Un ejemplo de clase experto sería *SecurityRoleCfgController* debido a que la misma contiene la información necesaria para realizar acciones con las demás clases.

Beneficios:

- El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello el beneficio de clases sencillas y más cohesivas, que son más fáciles de comprender. Así se brinda soporte a una Alta Cohesión. (52)
- ✓ **Controlador:** Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Un ejemplo de clase controladora sería *SecurityUserCfgController* debido a que esta es la que recibe la información relacionada con los usuarios, y la envía a las distintas clases que trabajan con esta información.
 - Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. (52)
 - ✓ **Creador:** Consiste en asignarle a una clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:
 - B agrega los objetos A.
 - B contiene los objetos A.
 - B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A. (52)

Este patrón fue utilizado para el diseño de la clase *SecurityUserCfgController* debido a que la misma es la encargada de la instanciación de los usuarios.

- ✓ **Alta Cohesión:** Como cada clase tiene un conjunto de funcionalidades relacionadas directamente con la entidad que representan, no realizan un trabajo enorme y por tanto pueden ser calificadas como de alta cohesión. (53)
- ✓ **Bajo Acoplamiento:** Las clases se comunican solo con las clases necesarias para desarrollar cada flujo de evento. (53)

Los patrones Alta Cohesión y Bajo Acoplamiento fueron utilizados para el diseño de las clases *SecurityUserController* y *SecurityRoleCfgController*, debido a que las mismas son las encargadas del trabajo con los usuarios y los roles.

2.5.4.3 Patrones GOF

Los patrones Gang of Four (GOF) describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos.

- ✓ **Fachada (Facade):** Provee una interfaz unificada simple, para acceder a una interfaz o grupo de interfaces de un subsistema. Se ha utilizado este patrón para reducir la dependencia entre clases, ofreciendo un punto de acceso, de manera que si estas cambian o se sustituyen por otras, solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones cliente. REKO tiene el componente Facade que aprovisiona estas ventajas. (54)

2.6 Modelo de diseño

Un Modelo de diseño, a diferencia del Modelo de dominio, se encuentra más cerca de la solución que se desea obtener y adquiere algunas de las entidades reflejadas en el Modelo de dominio para convertirlas en clases. Su propósito es especificar una solución que trabaje y pueda convertirse fácilmente en código fuente. (55)

2.6.1 Diagrama de clases del diseño

Un diagrama de clases es una representación de las clases que involucran al sistema y las relaciones entre ellas que pueden ser de asociación, de herencia, de uso y de agregación. Una clase es una definición de un conjunto de entidades u objetos que comparten los mismos atributos, operaciones y relaciones. (55) (Ver Figura 9)

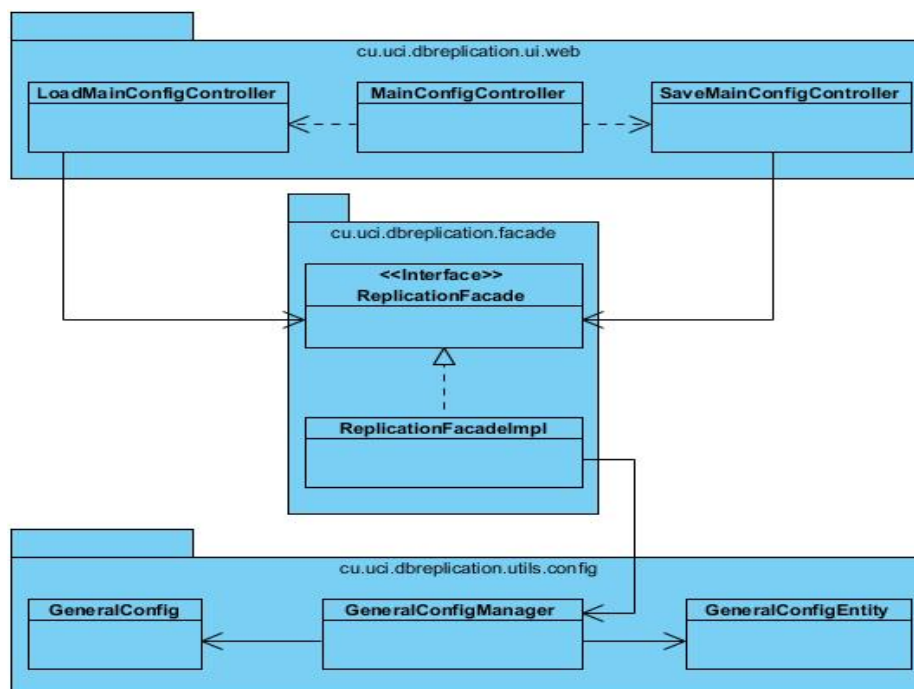


Figura 9: Diagrama de clases de diseño “Editar configuración de conexión”.

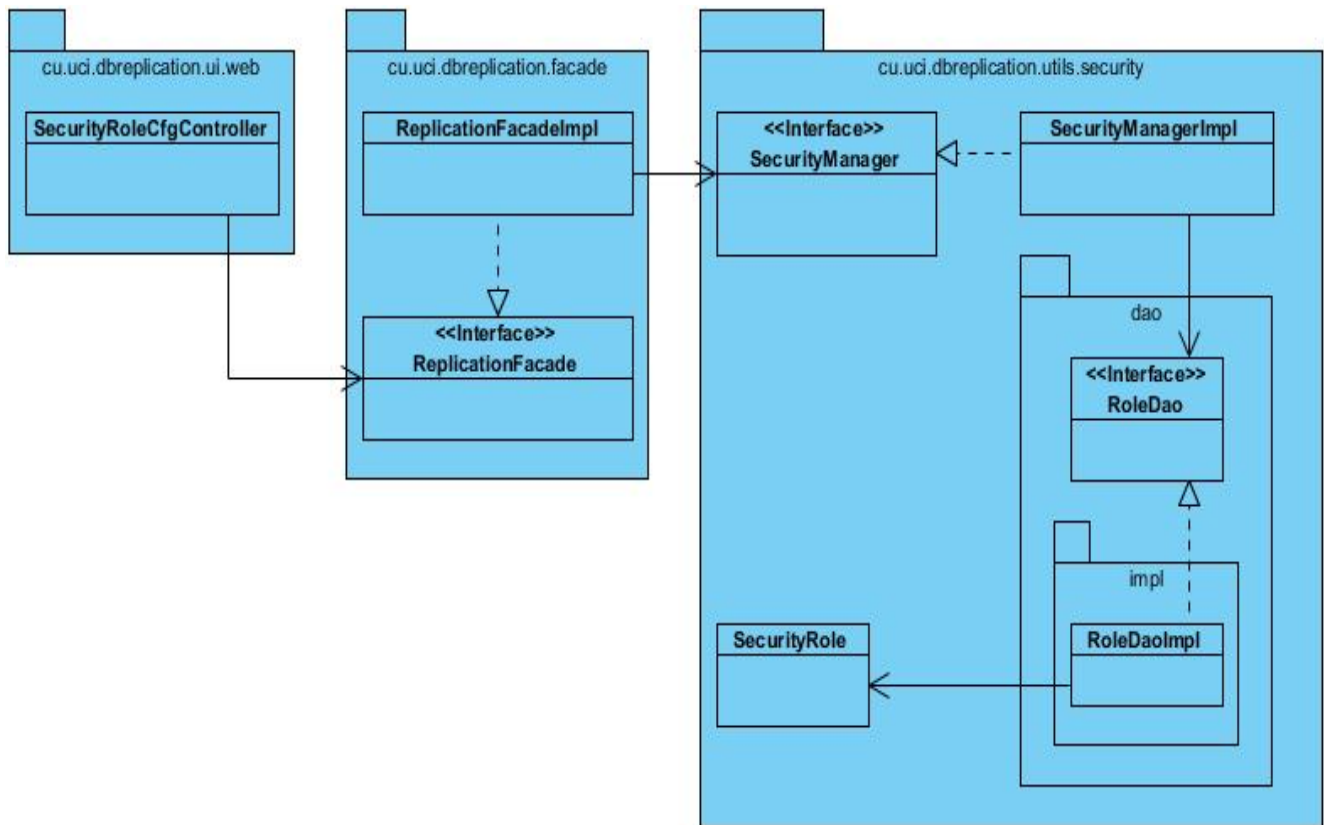


Figura 10: Diagrama de clases de diseño “Buscar roles”.

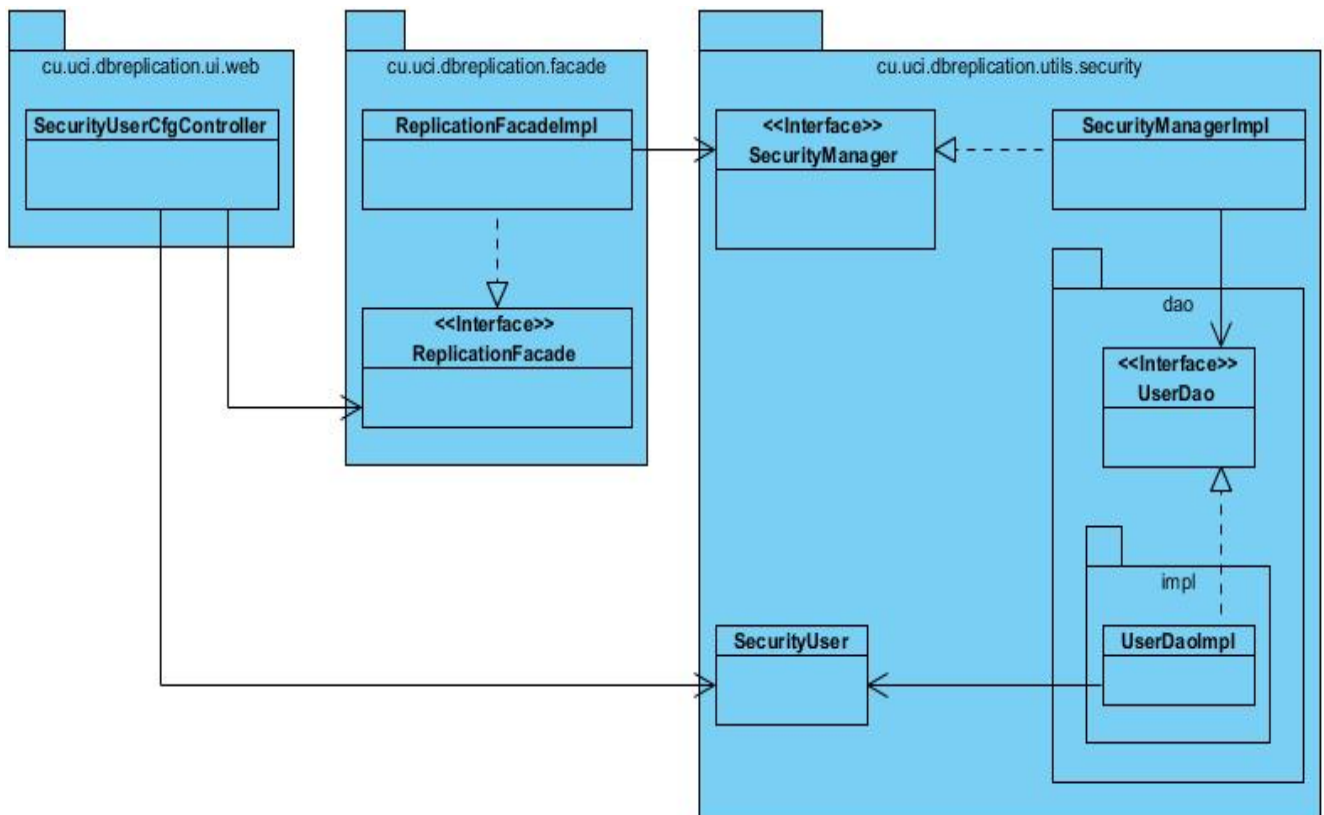


Figura 11: Diagrama de clases de diseño que incluye los requisitos “Filtrar usuarios del LDAP, Importar usuarios filtrados, Sincronizar servidor LDAP con BD embebida en REKO, Buscar usuarios”.

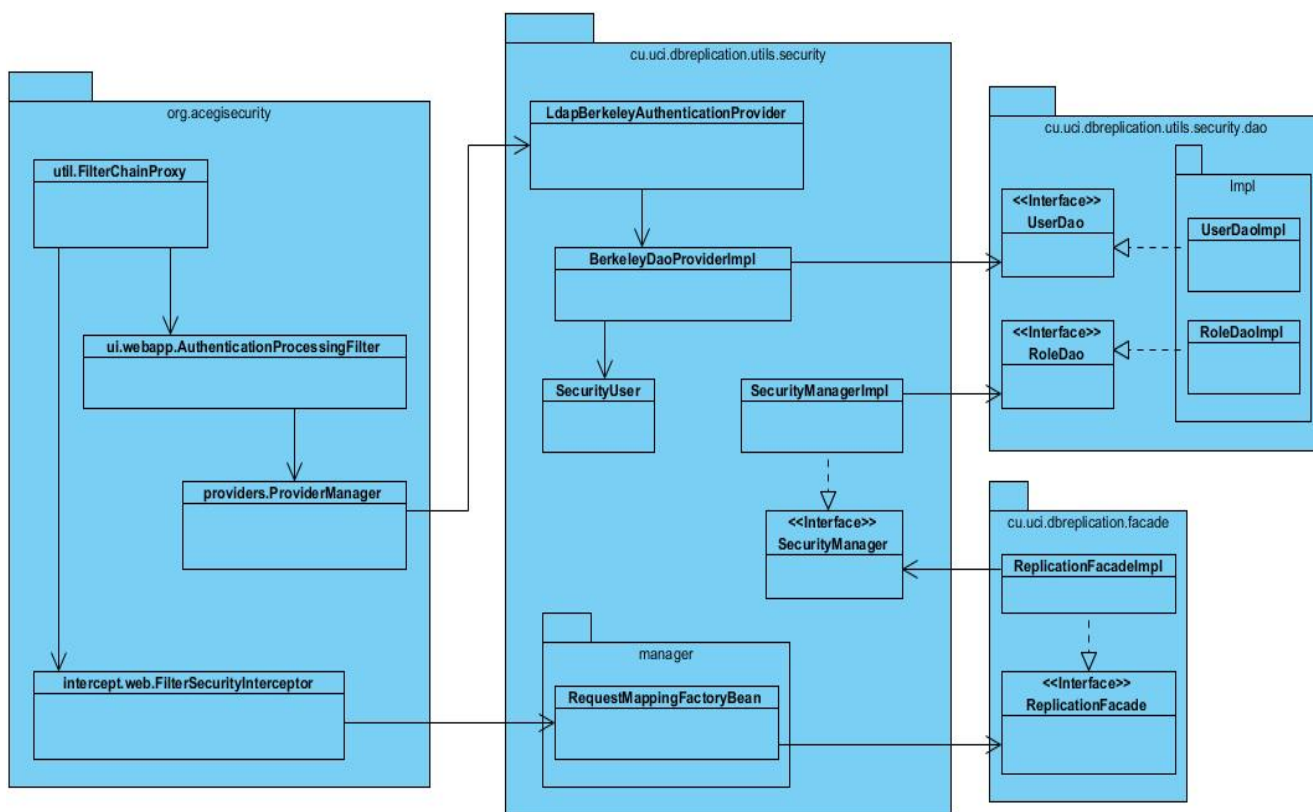


Figura 12: Diagrama de clases de diseño “Autenticar con LDAP”.

La descripción de los diagramas de paquetes correspondientes a cada diagrama de clases se puede observar en los Anexos del C.1 al C.4.

2.6.2 Descripción de las clases

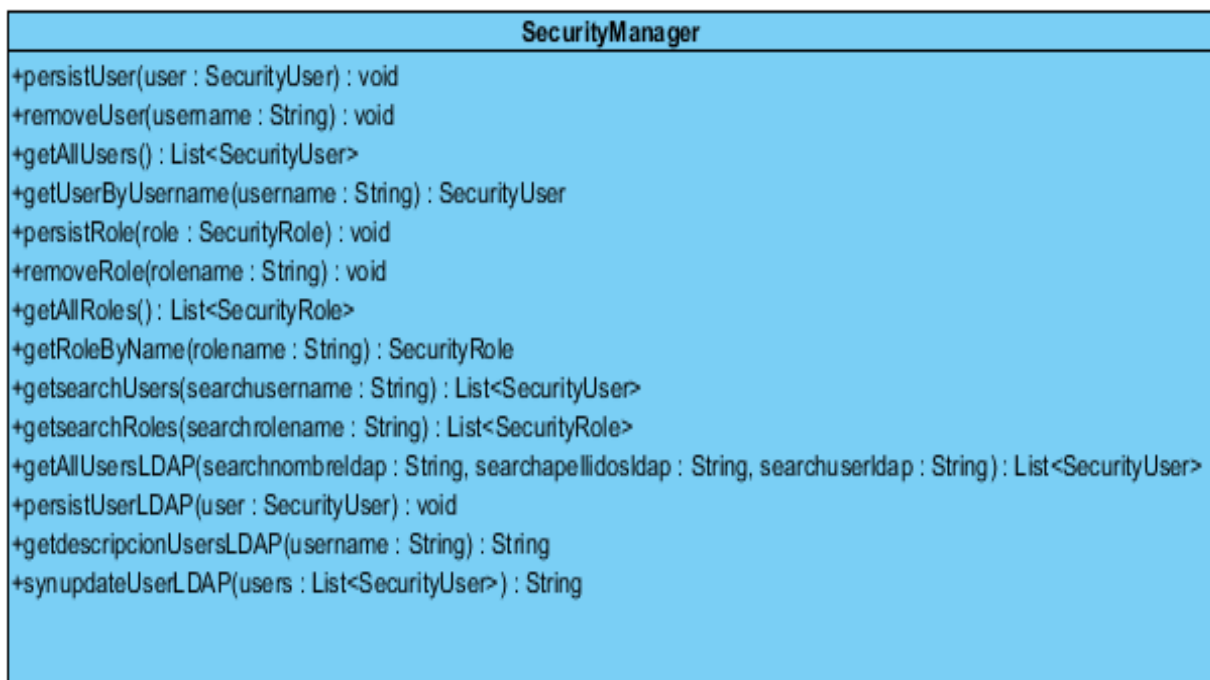


Figura 13: Descripción de la clase “SecurityManager”.

Tabla 10: Descripción de la clase “SecurityManager”.

Descripción de la clase SecurityManager	
Nombre:	SecurityManager
Tipo de clase:	Controladora.
Responsabilidades	
Nombre:	getAllUsers():List<SecurityUser>
Descripción:	Función para devolver todos los usuarios del sistema.
Nombre:	getUserByUsername(String username): SecurityUser
Descripción:	Función para devolver un usuario específico del sistema.
Nombre:	persistUser(SecurityUser user): void
Descripción:	Función para persistir los usuarios en el sistema.
Nombre:	removeUser(String username): void
Descripción:	Función para eliminar usuarios del sistema.
Nombre:	getAllRoles():List<SecurityRole>
Descripción:	Función para devolver todos los roles del sistema.
Nombre:	getRoleByName(String rolename): SecurityRole
Descripción:	Función para devolver un rol específico del sistema.
Nombre:	persistRole(SecurityRole role): void
Descripción:	Función para persistir los roles en el sistema.
Nombre:	removeRole(String rolename): void
Descripción:	Función para eliminar roles del sistema.
Nombre:	getsearchUsers(String searchusername): List<SecurityUser>
Descripción:	Función para la búsqueda de usuarios en el sistema.
Nombre:	getsearchRoles(String searchrolename): List<SecurityRole>
Descripción:	Función para la búsqueda de roles en el sistema.
Nombre:	getAllUsersLDAP(String searchnombreldap, String searchapellidosldap, String searchuserldap): List<SecurityUser>
Descripción:	Función para el filtrado de usuarios de un servidor LDAP, acorde a un criterio de búsqueda.
Nombre:	persistUserLDAP(SecurityUser user): void
Descripción:	Función para importar hacia la BD embebida de REKO los datos de usuarios filtrados de un servidor LDAP.
Nombre:	getdescripcionUsersLDAP(String username): String
Descripción:	Función para obtener la descripción de los usuarios importados hacia la BD embebida de REKO, de un servidor LDAP.
Nombre:	synupdateUserLDAP(List<SecurityUser> users): String
Descripción:	Función que permite eliminar del sistema los usuarios que no existen en el servidor LDAP.

Fuente: Elaboración propia.

UserDaolmpl	
<pre>#logger : Log = LogFactory.getLog(getClass()) -secuserDatabase : Database -SECURITY_USER_DATABASE_NAME : String = "su-database" -dbEnvironment : Environment -environmentFactory : EnvironmentFactory -use : String -nomb : String -apellidos : String</pre>	
<pre>+persist(user : SecurityUser) : void +getAll() : List<SecurityUser> +getUser(username : String) : SecurityUser +remove(username : String) : void +searchUser(searchusername : String) : List<SecurityUser> +getAllLDAP(nombrep : String, apellidosp : String, usuariop : String) : List<SecurityUser> +persistLDAP(user : SecurityUser) : void +getdescripcionLDAP(username : String) : String +synupdUserLDAP(users : List<SecurityUser>) : String</pre>	

Figura 14: Descripción de la clase "UserDaolmpl".

Tabla 11: Descripción de la clase "UserDaolmpl".

Descripción de la clase UserDaolmpl	
Nombre: UserDaolmpl	
Tipo de clase: Modelo / Entidad.	
Atributos	Tipos
logger	protected
secuserDatabase	private
SECURITY_USER_DATABASE_NAME	private
dbEnvironment	private
environmentFactory	private
use	private
nomb	private
apellidos	private
Responsabilidades	
Nombre:	persist(SecurityUser user): void
Descripción:	Función para persistir los usuarios en el sistema.
Nombre:	getAll(): List<SecurityUser>
Descripción:	Función para devolver todos los usuarios del sistema.
Nombre:	getUser(String username): SecurityUser
Descripción:	Función para devolver un usuario específico del sistema.
Nombre:	remove(String username): void

Descripción:	Función para eliminar usuarios del sistema.
Nombre:	searchUser(String searchusername): List<SecurityUser>
Descripción:	Función para la búsqueda de usuarios en el sistema.
Nombre:	getAllLDAP(String nombrep, String apellidosp, String usuariop): List<SecurityUser>
Descripción:	Función para el filtrado de usuarios de un servidor LDAP, acorde a un criterio de búsqueda.
Nombre:	persistLDAP(SecurityUser user): void
Descripción:	Función para importar hacia la BD embebida de REKO los datos de usuarios filtrados de un servidor LDAP.
Nombre:	getdescripcionLDAP(String username): String
Descripción:	Función para obtener la descripción de los usuarios importados hacia la BD embebida de REKO, de un servidor LDAP.
Nombre:	synupdUserLDAP(List<SecurityUser> users): String
Descripción:	Función que permite eliminar del sistema los usuarios que no existen en el servidor LDAP.

Fuente: Elaboración propia.

Se realizó la descripción de las demás clases que se utilizaron para llevar a cabo el desarrollo de la solución. (Ver Anexos del D.1 al D.18)

2.6.3 Diagrama de despliegue

El Diagrama de despliegue visualiza los elementos de hardware y software utilizados en la implementación del sistema y además las relaciones entre ellos. Es utilizado para modelar la topología de procesadores y dispositivos empleados en el software. Muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. (47) (Ver Figura 15)

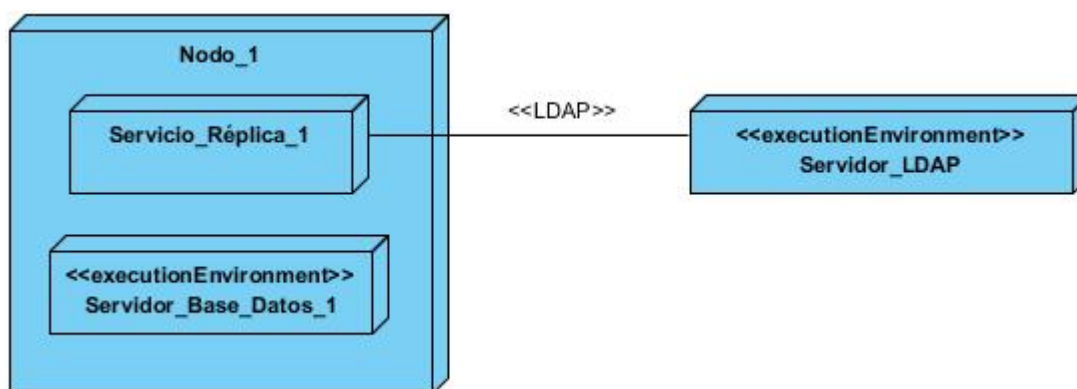


Figura 15: Diagrama de despliegue

2.7 Conclusiones parciales

Luego de realizarse el diseño del sistema se obtuvieron las siguientes conclusiones:

- ✓ Se realizó la descripción de la arquitectura utilizada en el software REKO y se fundamentaron y ejemplificaron los patrones de diseño utilizados.
- ✓ Fueron descritas las principales funcionalidades que conforman la solución, desglosándose las mismas en requisitos funcionales, haciendo énfasis en una amplia descripción de los mismos.
- ✓ Se describen además los requisitos no funcionales, los cuales representan las características físicas y lógicas que debe presentar la entidad que opte por REKO como solución para la réplica de datos.
- ✓ Se presentó los diagramas de clases y fueron brevemente descritas las clases principales.
- ✓ Se construyó el Diagrama de despliegue para lograr un mejor entendimiento de cómo está físicamente distribuido el sistema.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

El presente capítulo contiene los artefactos correspondientes al flujo de trabajo de implementación y pruebas. En el desarrollo del mismo se muestra la descripción de los diagramas de componentes, el cual no es más que la implementación del Modelo de diseño en términos de *scripts*, ficheros de código fuente, y ejecutables. Se presenta además el Modelo de pruebas donde se abordan detalles de la ejecución y los resultados de las mismas luego de haberlas aplicado al sistema.

3.1 Modelo de implementación

El Modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del Modelo de diseño a subsistemas y componentes físicos. (56)

3.1.1 Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el Modelo de diseño. Los diagramas de componentes describen los elementos físicos y sus realizaciones en el entorno de implementación mostrando las organizaciones y dependencias lógicas entre componentes de software, sean código fuente, binarios, archivos, bibliotecas cargadas dinámicamente o ejecutables. Modelan la vista estática de un sistema. (57)

A continuación se muestran los diagramas de componentes de la solución.

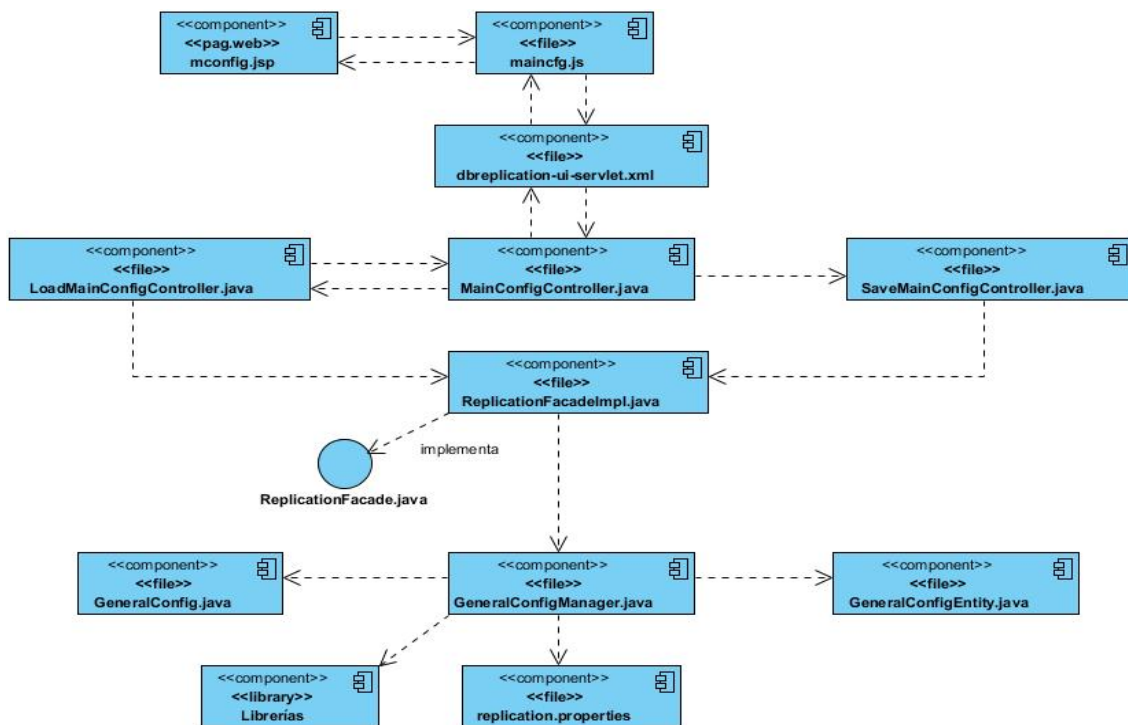


Figura 16: Diagrama de componente "Editar configuración de conexión"

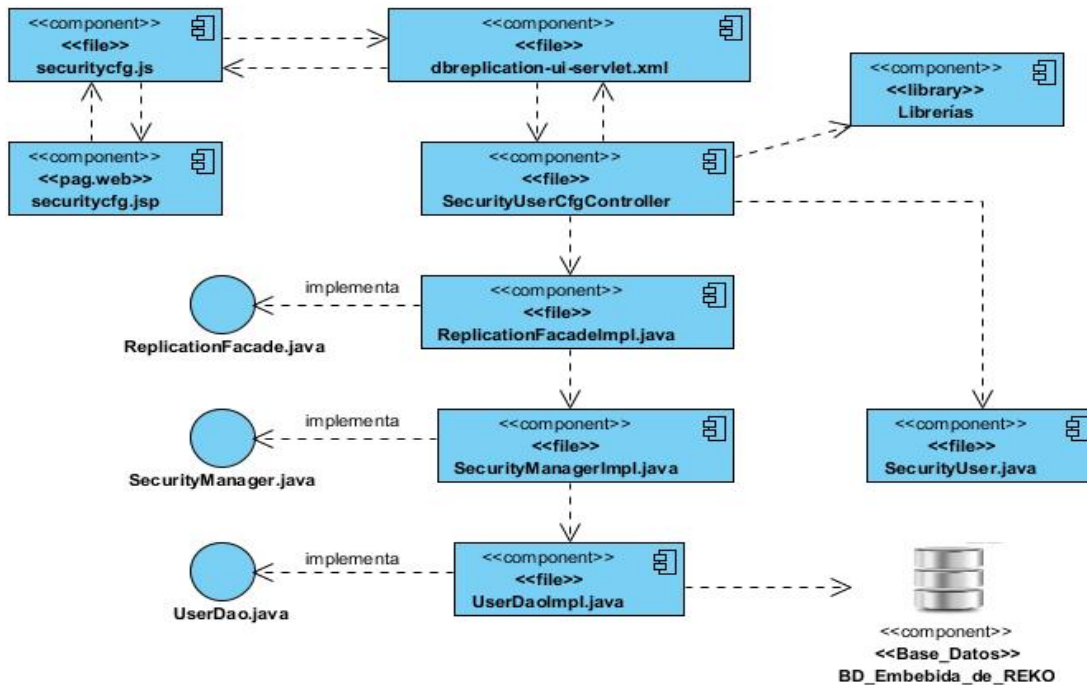


Figura 17: Diagrama de componente "Importar usuarios filtrados"

Los diagramas de componentes restantes se pueden observar en los Anexos del E.1 al E.5.

3.2 Código fuente

El Código fuente es un conjunto de líneas que conforman un bloque de texto, escrito según las reglas sintácticas de algún lenguaje de programación destinado a ser legible por humanos. Es un programa en su forma original, tal y como fue escrito por el programador, no es ejecutable directamente por el computador, debe convertirse en lenguaje de maquina mediante compiladores, ensambladores o intérpretes. (58)

A continuación se muestra la implementación de los requisitos más importantes:

3.2.1 Proceso de filtrado e importación de usuarios de un servidor LDAP

El administrador del sistema puede filtrar usuarios del servidor LDAP acorde a tres criterios de búsquedas (usuario, nombre y apellidos). Estos usuarios son obtenidos del servidor LDAP se define con anterioridad en el módulo de "Configuración General". Los usuarios son almacenados en memoria, para que posteriormente el administrador seleccione cuales van a ser importados a la BD embebida de REKO.

Una vez filtrados los usuarios del servidor LDAP, el administrador del sistema selecciona los usuarios a importar con sus correspondientes roles (persistirlos en la BD embebida de REKO). Cada usuario seleccionado debe poseer un rol determinado y no puede existir previamente en la BD embebida de REKO, el mismo una vez importado posee acceso a la consola de administración Web de REKO según sus permisos.

El siguiente fragmento de código pertenece a la clase *UserDaolmpl*. El método *getAllLDAP(String nombrep, String apellidosp, String usuariop)* es el encargado de realizar el proceso de filtrado de usuarios de un servidor LDAP.

```

UserDaolmpl
@Override
public List<SecurityUser> getAllLDAP(String nombrep, String apellidosp, String
usuariop) {
    List<SecurityUser> listUsers = new LinkedList<SecurityUser>();
    List<SecurityUser> listUserslimit = new LinkedList<SecurityUser>();
    String typeLDAP= GeneralConfig.getLDAPType();
    boolean activeLDAP=GeneralConfig.isLDAPUse();
    if (activeLDAP==true && typeLDAP.equals("OpenLDAP") && typeLDAP!=null) {
    if (!nombrep.equals("") || !usuariop.equals("") || !apellidosp.equals("")) {
        try {
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
            env.put(Context.PROVIDER_URL,
"ldap://" + GeneralConfig.getLDAPHost() + ":" + GeneralConfig.getLDAPPort() + "/" + GeneralConfig.ge
tLDAPDNBase().replace('.', ','));
            env.put(Context.SECURITY_PRINCIPAL, GeneralConfig.getLDAPUser());
            env.put(Context.SECURITY_CREDENTIALS, GeneralConfig.getLDAPPass());
            DirContext ctx = new InitialLdapContext(env, null);
            SearchControls searchCtrls = new SearchControls();
            NamingEnumeration<SearchResult> results = null;
            String returnedAtts[] = {"uid", "givenName", "sn"};
            searchCtrls.setReturningAttributes(returnedAtts);
            searchCtrls.setSearchScope(SearchControls.SUBTREE_SCOPE);
            String searchBase = "";
            if (nombrep.equals("") && apellidosp.equals("")) {
                results = ctx.search(searchBase, "&(objectclass=person)(uid=" + "*" +
usuariop + "*" + ")", searchCtrls);
            } else if (nombrep.equals("") && usuariop.equals("")) {
                results = ctx.search(searchBase, "&(objectclass=person)(sn=" + "*" +
apellidosp + "*" + ")", searchCtrls);
            } else if (apellidosp.equals("") && usuariop.equals("")) {
                results = ctx.search(searchBase, "&(objectclass=person)(givenName=" + "*"
+ nombrep + "*" + ")", searchCtrls);
            } else if (nombrep.equals("")) {
                results = ctx.search(searchBase, "&(objectclass=person)(sn=" + "*" +
apellidosp + "*" + ")(uid=" + "*" + usuariop + "*" + ")", searchCtrls);
            } else if (apellidosp.equals("")) {
                results = ctx.search(searchBase, "&(objectclass=person)(givenName=" + "*"
+ nombrep + "*" + ")(uid=" + "*" + usuariop + "*" + ")", searchCtrls);
            } else if (usuariop.equals("")) {
                results = ctx.search(searchBase, "&(objectclass=person)(givenName=" + "*"
+ nombrep + "*" + ")(sn=" + "*" + apellidosp + "*" + ")", searchCtrls);
            } else {
                results = ctx.search(searchBase, "&(objectclass=person)(givenName=" + "*"
+ nombrep + "*" + ")(uid=" + "*" + usuariop + "*" + ")(sn=" + "*" + apellidosp + "*" +
"))", searchCtrls);
            }
            while (results.hasMoreElements()) {
                SearchResult searchResult = (SearchResult) results.next();
                Attributes attrs = searchResult.getAttributes();

```

```

        Attribute usua = attrs.get("uid");
        if (usua != null) {
            use = usua.get().toString();
        } else {
            use = "";
        }
        Attribute no = attrs.get("givenName");
        if (no != null) {
            nomb = no.get().toString();
        } else {
            nomb = "-";
        }
        Attribute ap = attrs.get("sn");
        if (ap != null) {
            apellidos = ap.get().toString();
        } else {
            apellidos = "-";
        }
        SecurityUser userqq = new SecurityUser();
        userqq.setUsername(use);
        userqq.setPassword(null);
        userqq.setDescripcion(nomb+", "+apellidos);
        userqq.setRolenames(null);
        listUsers.add(userqq);
    }
    if (listUsers.isEmpty()) {
        logger.info("No existen usuarios en el servidor LDAP con esos datos");
    }
    ctx.close();
}
catch (NamingException ex) {
    logger.info("Error de conexión con el servidor LDAP");
}finally {
}
}
}

else if (activeLDAP==true && typeLDAP.equals("Active Directory") &&
typeLDAP!=null) {
    if (!nombrep.equals("") || !usuariop.equals("") || !apellidosp.equals("")) {
        String server =
"ldap://" + GeneralConfig.getLDAPHost() + ":" + GeneralConfig.getLDAPPport();
        try {
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
            env.put(Context.PROVIDER_URL, server);
            env.put(Context.SECURITY_PRINCIPAL, GeneralConfig.getLDAPUser());
            env.put(Context.SECURITY_CREDENTIALS, GeneralConfig.getLDAPPass());
            DirContext ctx = new InitialLdapContext(env, null);
            SearchControls searchCtls = new SearchControls();
            NamingEnumeration<SearchResult> results = null;
            String returnedAtts[] = {"samaccountname", "givenName", "sn"};
            searchCtls.setReturningAttributes(returnedAtts);
            searchCtls.setSearchScope(SearchControls.SUBTREE_SCOPE);
            String searchBase=GeneralConfig.getLDAPDNBase().replace('.', ',');
            if (nombrep.equals("") && apellidosp.equals("")) {

```



```

        results = ctx.search(searchBase,
"&(objectclass=person)(samaccountname=" + "*" + usuariop + "*" + ")", searchCtls);
        } else if (nombrep.equals("") && usuariop.equals("")) {
            results = ctx.search(searchBase, "&(objectclass=person)(sn=" +
"*" + apellidosp + "*" + ")", searchCtls);
        } else if (apellidosp.equals("") && usuariop.equals("")) {
            results = ctx.search(searchBase,
"&(objectclass=person)(givenName=" + "*" + nombrep + "*" + ")", searchCtls);
        } else if (nombrep.equals("")) {
            results = ctx.search(searchBase, "&(objectclass=person)(sn=" +
"*" + apellidosp + "*" + ")(samaccountname=" + "*" + usuariop + "*" + ")", searchCtls);
        } else if (apellidosp.equals("")) {
            results = ctx.search(searchBase,
"&(objectclass=person)(givenName=" + "*" + nombrep + "*" + ")(samaccountname=" + "*" +
usuariop + "*" + ")", searchCtls);
        } else if (usuariop.equals("")) {
            results = ctx.search(searchBase,
"&(objectclass=person)(givenName=" + "*" + nombrep + "*" + ")(sn=" + "*" + apellidosp +
"*" + ")", searchCtls);
        } else {
            results = ctx.search(searchBase,
"&(objectclass=person)(givenName=" + "*" + nombrep + "*" + ")(samaccountname=" + "*" +
usuariop + "*" + ")(sn=" + "*" + apellidosp + "*" + ")", searchCtls);
        }
        while (results.hasMoreElements()) {
            SearchResult searchResult = (SearchResult) results.next();
            Attribute attrs = searchResult.getAttributes();
            Attribute usua = attrs.get("samaccountname");
            if (usua != null) {
                use = usua.get().toString();
            } else {
                use = "";
            }
            Attribute no = attrs.get("givenName");
            if (no != null) {
                nomb = no.get().toString();
            } else {
                nomb = "-";
            }
            Attribute ap = attrs.get("sn");
            if (ap != null) {
                apellidos = ap.get().toString();
            } else {
                apellidos = "-";
            }
            SecurityUser userqq = new SecurityUser();
            userqq.setUsername(use);
            userqq.setPassword(null);
            userqq.setDescripcion(nomb+", "+apellidos);
            userqq.setRolenames(null);
            listUserslimit.add(userqq);
        }
        if(listUserslimit.size()>=200){
            for (int i=0; i<200; i++){
                listUsers.add(listUserslimit.get(i));
            }
        }
    }
}

```

```

        }else {
            for (int i=0; i<listUserslimit.size(); i++){
                listUsers.add(listUserslimit.get(i));
            }
        }
        if (listUsers.isEmpty()) {
            logger.info("No existen usuarios en el servidor LDAP con esos datos");
        }
        ctx.close();
    } catch (NamingException ex) {
        logger.info("Error de conexión con el servidor LDAP");
    }finally {
    }
}
}
return listUsers;
}
}

```

El siguiente fragmento de código pertenece a la clase *SecurityUserCfgController*. El método *importUserLDAP(HttpServletRequest request, HttpServletResponse response)* es el encargado de realizar el proceso de importación de usuarios de un servidor LDAP hacia la BD embebida de REKO.

SecurityUserCfgController

```

public void importUserLDAP(HttpServletRequest request,
    HttpServletResponse response) throws IOException,
    DataAccessException {
    String username = request.getParameter("username");
    String password = "";
    String rolenamesLDAP = request.getParameter("rolenamesLDAP");
    String[] arrayColores = rolenamesLDAP.split(" ");
    String rolenamesLDAPimport="";
    for (int i = 0; i < arrayColores.length; i++) {
        rolenamesLDAPimport=rolenamesLDAPimport.concat(arrayColores[i]);
    }
    List<SecurityUser> users = new ArrayList<SecurityUser>();
    users = replicationFacade.getAllSecurityUsers();
    try {
        if (username == null || username.equals("")) {
            response.getWriter().print(
                "Debe seleccionar al menos un usuario a importar");
            response.getWriter().flush();
            response.getWriter().close();
        }
        else if (existUser(users, username)) {
            response.getWriter().print(
                "El usuario especificado ya existe en el sistema");
            response.getWriter().flush();
            response.getWriter().close();
        }
        else {
            Md5PasswordEncoder md5PasswordEncoder = new Md5PasswordEncoder();
            String encodePassword = md5PasswordEncoder.encodePassword(
                password, username);

```

```

        new JSONArray();
        JSONArray listURLJson =
JSONArray.fromObject(rolenamesLDAPimport);
        List<String> roleList = new LinkedList<String>();
        for (int i = 0; i < listURLJson.size(); i++) {
            roleList.add(listURLJson.getString(i));
        }
        if (roleList.isEmpty()) {
            response.getWriter().print("Debe seleccionar al menos un rol");
            response.getWriter().flush();
            response.getWriter().close();
        } else {
            String descripcion =
replicationFacade.getdescripcionSecurityUsersLDAP(username);
            SecurityUser user = new SecurityUser();
            user.setUsername(username);
            user.setPassword(encodePassword);
            user.setDescripcion(descripcion);
            user.setRolenames(roleList);
            replicationFacade.importSecurityUserLDAP(user);
            providerImpl.authoritiesRolesProvider(roleList);
            response.getWriter().print(
                "Usuarios importados satisfactoriamente");
            response.getWriter().flush();
            response.getWriter().close();
        }
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

3.2.2 Proceso de sincronización de servidor LDAP con BD embebida en REKO

El administrador del sistema puede optar por dos criterios de sincronización (Eliminar y Actualizar) para mantener la BD embebida de REKO actualizada con respecto al servidor LDAP en uso. La opción de “Eliminar” borra del sistema todos los usuarios que fueron importados de un servidor LDAP y la de “Actualizar” borra del sistema aquellos usuarios que no existen en el servidor LDAP en uso.

El siguiente fragmento de código pertenece a la clase *SecurityUserCfgController*. El método *synchronizedelUserLDAP(HttpServletRequest request, HttpServletResponse response)* es el encargado de eliminar del sistema los usuarios que fueron importados de un servidor LDAP.

SecurityUserCfgController

```

public void synchronizedelUserLDAP(HttpServletRequest request,
    HttpServletResponse response) throws IOException, DataAccessException {
    List<SecurityUser> users = replicationFacade.getAllSecurityUsers();
    List<SecurityUser> listUsers = new LinkedList<SecurityUser>();
    for (SecurityUser user1 : users) {
        if (user1.getDescripcion().contains("(LDAP)")){
            listUsers.add(user1);
        }
    }
}

```

```

    }
    if (listUsers.size()==0)
    {
        response.getWriter().print("No hay usuarios para eliminar");
        response.getWriter().flush();
        response.getWriter().close();
    }
    else {
        for (SecurityUser user : users) {
            if (user.getDescripcion().contains("(LDAP)")){
                String username=user.getUsername();
                replicationFacade.removeSecurityUser(username);
            }
        }
    }
    response.getWriter().print("Usuarios eliminados satisfactoriamente, BD actualizada");
    response.getWriter().flush();
    response.getWriter().close();
}
}

```

El siguiente fragmento de código pertenece a la clase *UserDaoImpl*. El método *synupdUserLDAP(List<SecurityUser users>)* es el encargado de eliminar del sistema los usuarios que no existen en el servidor LDAP.

UserDaoImpl

```

@Override
public String synupdUserLDAP(List<SecurityUser> users){
    String typeLDAP= GeneralConfig.getLDAPType();
    boolean activeLDAP=GeneralConfig.isLDAPUse();
    String resp="update";
    String username=null;
    if (activeLDAP==true && typeLDAP.equals("OpenLDAP") && typeLDAP!=null)
    {
        try {
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
            env.put(Context.PROVIDER_URL,
"ldap://" + GeneralConfig.getLDAPHost() + ":" + GeneralConfig.getLDAPPport() + "/" + GeneralConfig.ge
tLDAPDNBase().replace('.', ','));
            env.put(Context.SECURITY_PRINCIPAL, GeneralConfig.getLDAPUser());
            env.put(Context.SECURITY_CREDENTIALS, GeneralConfig.getLDAPPass());
            DirContext ctx = new InitialLdapContext(env, null);
            SearchControls searchCtrls = new SearchControls();
            NamingEnumeration<SearchResult> results = null;
            String returnedAtts[] = {"uid"};
            searchCtrls.setReturningAttributes(returnedAtts);
            searchCtrls.setSearchScope(SearchControls.SUBTREE_SCOPE);
            String searchBase = "";
            for (SecurityUser user : users) {
                username=user.getUsername();
                results = ctx.search(searchBase, "(uid="+username+")", searchCtrls);
                if(!results.hasMoreElements()) {
                    this.remove(username);
                }
            }
        }
    }
}

```

```

        }
    }
    ctx.close();
}
catch (NamingException ex) {
    logger.info("Error de conexión con el servidor LDAP");
}finally {
}
}

else if (activeLDAP==true && typeLDAP.equals("Active Directory") &&
typeLDAP!=null) {
    String server =
"ldap://" + GeneralConfig.getLDAPHost() + ":" + GeneralConfig.getLDAPPport();
    try {
        Hashtable env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
        env.put(Context.PROVIDER_URL, server);
        env.put(Context.SECURITY_PRINCIPAL, GeneralConfig.getLDAPUser());
        env.put(Context.SECURITY_CREDENTIALS, GeneralConfig.getLDAPPass());
        DirContext ctx = new InitialLdapContext(env, null);
        SearchControls searchCtls = new SearchControls();
        NamingEnumeration<SearchResult> results = null;
        String returnedAtts[] = {"samaccountname"};
        searchCtls.setReturningAttributes(returnedAtts);
        searchCtls.setSearchScope(SearchControls.SUBTREE_SCOPE);
        String searchBase=GeneralConfig.getLDAPDNBase().replace('.', ',');
        for (SecurityUser user : users) {
            username=user.getUsername();
            results = ctx.search(searchBase,
"(samaccountname="+username+")", searchCtls);
            if(!results.hasMoreElements()) {
                this.remove(username);
            }
        }
        ctx.close();
    } catch (NamingException ex) {
        logger.info("Error de conexión con el servidor LDAP");
    }finally {
    }
}
return resp;
}
}

```

3.2.3 Proceso de autenticación con servidor LDAP

El usuario que desee acceder a la consola de administración Web del Replicador de datos REKO puede autenticarse a través de la BD embebida de REKO, o usando un servidor LDAP predefinido anteriormente por un usuario con permiso de administrador. Si el administrador define que la autenticación es a través de un servidor LDAP, los usuarios que se desean que tengan acceso a REKO deben ser importados a la BD embebida del mismo con anterioridad. El usuario a autenticarse debe poseer las credenciales (usuario y contraseña) correctas en el servidor LDAP y estar persistido en REKO como usuario LDAP con roles definidos.

El proceso de lectura de las configuraciones de seguridad, usuarios y roles, desde la BD Berkeley así como la verificación de estos en el servidor LDAP es justo en el momento de la autenticación, para proceder al posterior autorizo del usuario y de esta forma dotarlo de permisos para realizar acciones dentro del sistema. Este proceso comienza en la configuración del contexto de seguridad del sistema que tiene el nombre *dbreplication-security-context.xml*, específicamente el *bean* con el id "daoAuthenticationProvider", perteneciente a la clase de *Acegi org.acegisecurity.providers.dao.DaoAuthenticationProvider*.

En algunos casos fue necesario variar el comportamiento de algunas funcionalidades de Acegi para poder desarrollar las funcionalidades previstas. Como técnica común ante este tipo de situación, se realizó el procedimiento de heredar de las clases que se querían variar e implementar nuevamente las funcionalidades necesarias. Este método evita tener que cambiar código dentro de los *frameworks*. El *bean* con el id "daoAuthenticationProvider" fue modificado, haciendo referencia a la clase *cu.uci.dbreplication.utils.security.LdapBerkeleyAuthenticationProvider*. En esta clase es donde se realiza la variación del comportamiento de la funcionalidad *additionalAuthenticationChecks*, encargada de realizar el proceso de autenticación a través de un servidor LDAP.

En este contexto de seguridad es donde se realizan todas las configuraciones de la misma. Este tiene la siguiente estructura:

```
<bean id="daoAuthenticationProvider"
      class="cu.uci.dbreplication.utils.security.LdapBerkeleyAuthenticationProvider">
  <property name="passwordEncoder">
    <bean class="org.acegisecurity.providers.encoding.Md5PasswordEncoder" />
  </property>
  <property name="saltSource">
    <bean class="org.acegisecurity.providers.dao.salt.ReflectionSaltSource">
      <property name="userPropertyToUse" value="getUsername" />
    </bean>
  </property>
  <property name="userDetailsService">
    <ref bean="berkeleyDaoProviderImpl" />
  </property>
</bean>
```

Figura 18: Bean "daoAutenticationProvider".

Dentro de los atributos que contiene la clase *LdapBerkeleyAuthenicationProvider* para proveer seguridad a un sistema, se encuentra el *userDetailsService*, este se encarga de suministrarle a Acegi los elementos de seguridad necesarios para administrar el sistema. Los valores de este atributo son asignados desde la clase *BerkeleyDaoProviderImpl*.

```
<bean id="berkeleyDaoProviderImpl" class="cu.uci.dbreplication.utils.security.BerkeleyDaoProviderImpl">
  <property name="userDao" ref="userDaoImpl"/>
  <property name="roleDao" ref="roleDaoImpl"></property>
  <property name="replicationFacade"><ref bean="replicationFacade"/></property>
</bean>
```

Figura 19: Bean "berkeleyDaoProviderImpl".

La clase *BerkeleyDaoProviderImpl* implementa la interfaz *UserDetailsService*. Al implementar esta interfaz se puede redefinir el método llamado *loadUserByUsername*, el cual partiendo del usuario que se desea autenticar en el sistema es capaz de verificar si el mismo está persistido o no en la BD embebida de REKO y luego crea un objeto de tipo *GrantedAuthority*, el mismo es un arreglo que contiene los roles pertenecientes al usuario que fue pasado como parámetro a la función. De este modo el método retorna un objeto de tipo *UserDetails* (definición de seguridad que contiene el usuario que se desea autenticar con los roles pertenecientes a este) el cual modifica el valor actual del atributo *UserDetailsService* en la clase *cu.uci.dbreplication.utils.security.LdapBerkeleyAuthenticationProvider*.

Luego de realizar una breve descripción de las funcionalidades implementadas se procede a realizar las pruebas para verificar la solidez de las mismas.

3.3 Modelo de pruebas

Las pruebas de software desempeñan un papel decisivo cuando se desea obtener un producto de alta calidad y buen funcionamiento, por lo que su objetivo es verificar y validar que realmente el software realice lo que se espera de él, dándole solución a los RF definidos. “Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente”. (59)

3.3.1 Técnicas y estrategias de prueba

Existen dos técnicas para realizar pruebas, las técnicas de Caja Blanca y las técnicas de Caja Negra. Las de Caja Blanca se basan en un riguroso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Las de Caja Negra es donde se realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa, no es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. (60)

Para evaluar dinámicamente un sistema de software se definen estrategias o niveles de prueba que permiten probar desde los componentes más simples y pequeños e ir avanzando hasta probar todo el software en su conjunto. En total se dividen en 5 niveles de pruebas: (60)

- ✓ **Pruebas Unitarias:** Son la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, esté correctamente codificado. En estas pruebas cada módulo es probado por separado y lo hace, generalmente, la persona que lo creó.
- ✓ **Pruebas de Integración:** Son realizadas para probar el software ensamblando todos los módulos probados previamente, a partir del esquema del diseño.

- ✓ **Pruebas del Sistema:** Son efectuadas cuando el software se encuentra ensamblado totalmente e integrado con cualquier componente hardware, permite comprobar que se cumplen los RF.
- ✓ **Pruebas de Aceptación:** Son aplicadas cuando el producto está listo para implantarse en el entorno del cliente y son realizadas por el usuario en conjunto con las personas del equipo de pruebas, siendo deseable, que sea el mismo usuario quien aporte los casos de prueba. Las pruebas de aceptación son pruebas de Caja Negra que se crean a partir de las HU, 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.
- ✓ **Pruebas de Regresión:** Consisten en la repetición selectiva de pruebas para detectar fallos introducidos durante la modificación de un sistema o componente de un sistema, y son efectuadas para comprobar que los cambios no han originado efectos adversos.

A las funcionalidades implementadas para la extensión del módulo de seguridad a través de LDAP en el Replicador de datos REKO se le hicieron **Pruebas de Aceptación**, utilizando la técnica de **Caja Negra** para validar el correcto funcionamiento de las interfaces.

3.3.2 Pruebas de Caja Negra

Las pruebas de Caja Negra también conocidas como pruebas funcionales, pruebas de caja opaca o pruebas de entrada/salida, son las que no toman en cuenta el código, el usuario que prueba no sabe cómo está estructurado por dentro el programa, solo necesita saber cuáles pueden ser las posibles entradas sin necesidad de entender cómo se deben obtener las salidas, donde se trata de encontrar errores en la interfaz mientras se está usando, el cómo luce, se maneja, etc. (61)

Objetivos: Revelar el incorrecto o incompleto funcionamiento del sistema, así como los errores de interfaz, rendimiento y errores de inicialización y terminación. (61)

Descripción: Se llevan a cabo sobre la interfaz del software y son completamente indiferentes al comportamiento interno y a la estructura del programa. Pretenden demostrar que: (61)

- ✓ Las funciones del software son operativas.
- ✓ La entrada se acepta de forma adecuada.
- ✓ Se produce una salida correcta.
- ✓ La integridad de la información externa se mantiene.

Para confeccionar los casos de prueba de Caja Negra existen distintas técnicas. Entre las que se encuentran: (61)

- ✓ **Particiones de Equivalencia:** Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Esta se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de

casos de prueba que hay que desarrollar. Se basa en una evaluación de las clases de equivalencia para una condición de entrada.

- ✓ **Análisis de Valores Límite (AVL):** Es una técnica de diseño de casos de prueba que complementa la partición equivalente. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.
- ✓ **Guiada por Casos de prueba:** Verifican las especificaciones funcionales y no consideran la estructura interna del programa. Se realiza sin el conocimiento interno del producto. No validan funciones ocultas por tanto los errores asociados a ellas no son encontrados.

Con el objetivo de comprobar el correcto funcionamiento de las funcionalidades implementadas, las mismas fueron sometidas a una serie de pruebas, a través de las cuales se pudo corroborar que el sistema responde positivamente a toda actividad que el usuario realice sobre él. A continuación se muestran los resultados obtenidos al aplicar las pruebas de **Caja Negra** utilizando la técnica de **Particiones de Equivalencia**.

3.3.2.1 Casos de pruebas de Caja Negra

Para evidenciar la aplicación de las pruebas de Caja Negra, a continuación se detallan las características de la aplicación de la misma al requisito funcional "Filtrar usuarios del LDAP".

Escenario: Filtrar usuarios del LDAP

Descripción general: En el módulo de "Seguridad", específicamente en la sección "LDAP", el administrador del sistema puede filtrar usuarios del servidor LDAP acorde a tres criterios de búsquedas (usuario, nombre y apellidos). Estos usuarios son obtenidos del servidor LDAP el cual se define con anterioridad en el módulo de "Configuración General". Los usuarios son almacenados en memoria, para que posteriormente el administrador seleccione cuales son importados a la BD embebida de REKO. Para ejecutar la acción de "Filtrar usuarios del LDAP" debe presionar el botón "Filtrar". El sistema muestra un mensaje "Usuarios filtrados del servidor LDAP".

Condiciones de ejecución:

- ✓ El usuario debe estar autenticado en el sistema.
- ✓ Los parámetros de conexión al servidor LDAP deben estar correctamente configurados, sino el sistema muestra una alerta "Error de conexión con el servidor LDAP".

Tabla 12: Descripción de variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto	Si	Especificar el usuario para filtrarlo en el servidor LDAP.
2	Nombre	Campo de texto	Si	Especificar el nombre para filtrarlo en el servidor LDAP.

3	Apellidos	Campo de texto	Si	Especificar los apellidos para filtrarlo en el servidor LDAP.
---	-----------	----------------	----	---

Tabla 13: Matriz de datos

Escenario	Descripción	Usuario	Nombre	Apellidos	Respuesta del sistema	Flujo central
EC 1.1 Filtrar usuarios del servidor LDAP a través de tres criterios de búsqueda (usuario, nombre y apellidos), introduciendo los datos correctamente.	Filtrar usuarios del servidor LDAP acorde a tres criterios de búsqueda (usuario, nombre y apellidos), introduciendo los datos correctamente.	V	V	V	2- El sistema muestra el módulo de seguridad. 5- El sistema filtra y muestra el/los usuario/s correspondientes al/los criterio/s especificados. 6- El sistema muestra el mensaje "Usuarios filtrados del servidor LDAP".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" escribir los criterios de filtrado en los campos "Usuario", "Nombre" y "Apellidos". 4- Clic en el botón "Filtrar".
		rol	Riog	Collar		
		N/A	V	V		
		Vacío	Riog	Collar		
V	N/A	V	V	2- El sistema muestra el módulo de seguridad. 5- El sistema filtra y muestra el/los usuario/s correspondientes al/los criterio/s especificados. 6- El sistema muestra el mensaje "Usuarios filtrados del servidor LDAP".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" escribir los criterios de filtrado en los campos "Nombre" y "Apellidos". 4- Clic en el botón "Filtrar".	
						rol
V	V	V	V	V	2- El sistema muestra el módulo de seguridad. 5- El sistema filtra y muestra el/los usuario/s correspondientes al/los criterio/s especificados. 6- El sistema muestra el mensaje "Usuarios filtrados del servidor LDAP".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" escribir los criterios de filtrado en los campos "Usuario" y "Apellidos". 4- Clic en el botón "Filtrar".

					"Usuarios filtrados del servidor LDAP".	
		V rrol	V Riog	N/A Vacío	2- El sistema muestra el módulo de seguridad. 5- El sistema filtra y muestra el/los usuario/s correspondientes al/los criterio/s especificados. 6- El sistema muestra el mensaje "Usuarios filtrados del servidor LDAP".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" escribir los criterios de filtrado en los campos "Usuario" y "Nombre". 4- Clic en el botón "Filtrar".
		V rrol	V Riogerre r	V Collar	2- El sistema muestra el módulo de seguridad. 5- El sistema muestra el mensaje "No existen usuarios en el servidor LDAP con esos datos".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" escribir los criterios de filtrado en los campos "Usuario", "Nombre" y "Apellidos". 4- Clic en el botón "Filtrar".
EC 1.2	Filtrar usuarios del servidor LDAP acorde a tres criterios de búsqueda (usuario, nombre y apellidos), introduciendo o los datos incorrectame	I Vacío	I Vacío	I Vacío	2- El sistema muestra el módulo de seguridad. 5- El sistema muestra una alerta "Debe especificar un criterio de filtrado".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" no se especifica ningún criterio de filtrado. 4- Clic en el botón "Filtrar".
		V	V	I	2- El sistema	1- Clic en el link

	nte.	rcol	Riog	Coll4*	muestra el módulo de seguridad. 5- El sistema muestra una alerta "El valor especificado no es válido".	"Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" se especifica un apellido incorrecto en el campo "Apellido". 4- Clic en el botón "Filtrar".
		V rcol	I Riog8/	V Collar	2- El sistema muestra el módulo de seguridad. 5- El sistema muestra una alerta "El valor especificado no es válido".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" se especifica un nombre incorrecto en el campo "Nombre". 4- Clic en el botón "Filtrar".
		I rcol*9	V Riog	V Collar	2- El sistema muestra el módulo de seguridad. 5- El sistema muestra una alerta "El valor especificado no es válido".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" se especifica un usuario incorrecto en el campo "Usuario". 4- Clic en el botón "Filtrar".
EC 1.3	Filtrar usuarios del servidor LDAP con los parámetros de conexión al servidor LDAP incorrectamente configurados en el módulo de "Configuración General".				2- El sistema muestra el módulo de seguridad. 5- El sistema muestra una alerta "Error de conexión con el servidor LDAP".	1- Clic en el link "Seguridad" 3- En la opción "LDAP" específicamente en la sección "Filtrar e Importar Usuarios" escribir los criterios de filtrado en los campos "Usuario", "Nombre" y "Apellidos". 4- Clic en el botón "Filtrar".

La descripción de los casos de prueba de Caja Negra restantes se puede observar en los Anexos del G.1 al G.6.

Para comprobar cada una de las funcionalidades por separado y definir si el software está listo y correctamente terminado se realizaron **Pruebas Unitarias** utilizando la técnica de **Caja Blanca**.

3.3.3 Pruebas de Caja Blanca

Las pruebas de Caja Blanca, en ocasiones llamada pruebas de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de pruebas. Al emplear los métodos de pruebas de Caja Blanca, el ingeniero de software puede derivar casos de pruebas que: (62)

- Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por al menos una vez.
- Ejerciten todos los lados verdaderos y falsos de todas las decisiones lógicas.

La aplicación de estas pruebas permitieron verificar el comportamiento de las clases *BerkeleyDaoProviderImpl* (encargada de realizar el proceso de autenticación), *SecurityRoleCfgController* (encargada de manejar las peticiones para gestionar los roles dentro del sistema) y *SecurityUserCfgController* (encargada de manejar las peticiones para gestionar los usuarios dentro del sistema).

Mediante estas pruebas fue posible comparar la respuesta que debería obtenerse de un método con lo que realmente se adquiriría de él.

Existen varias herramientas que facilitan la elaboración y desarrollo de casos de pruebas, además de darle seguimiento a los errores. Un ejemplo de ello es JUnit utilizada para realizar pruebas unitarias de aplicaciones Java.

Para este proceso se hizo necesario diseñar las clases *BerkeleyDaoProviderTest*, *SecurityRoleCfgControllerTest* y *SecurityUserCfgControllerTest*, las cuales extienden de la clase *TestCase*, perteneciente al *framework* de pruebas de Java JUnit. Estas clases contienen la implementación de prueba de los métodos de las clases *BerkeleyDaoProviderImpl*, *SecurityRoleCfgController* y *SecurityUserCfgController* respectivamente. Fue modificada la clase *TestRunnerAllTest*, en la cual se llaman los métodos de las clases *BerkeleyDaoProviderTest*, *SecurityRoleCfgControllerTest* y *SecurityUserCfgControllerTest* para su ejecución en tiempo real.

A continuación se muestran algunas de las pruebas realizadas y los resultados que arrojaron:

```

@Test
public void testGetUsersTableLDAP() {
    System.out.println("Probando el Filtrar Usuarios del LDAP");
    long t = System.currentTimeMillis();

    List<SecurityUser> users1 = securitymanager.getAllUsersLDAP("ri", "lop", "");
    assertTrue(users1.size()>0);
    System.out.println("Cantidad de usuarios obtenidos:"+ users1.size());

    System.out.println("Tiempo de ejecución:"+ (System.currentTimeMillis()-t));
}

```

Figura 20: Método de prueba *testGetUsersTableLDAP* de la clase *SecurityUserCfgControllerTest*

```

@Test
public void testGetsearchUsersTable() {
    System.out.println("Probando el Buscar Usuarios en BD");
    long t = System.currentTimeMillis();

    List<SecurityUser> users2 = securitymanager.getsearchUsers("o");
    assertTrue(users2.size()>0);
    System.out.println("Cantidad de usuarios obtenidos:"+ users2.size());

    System.out.println("Tiempo de ejecución:"+ (System.currentTimeMillis()-t));
}

```

Figura 21: Método de prueba *testGetsearchUsersTable* de la clase *SecurityUserCfgControllerTest*

Las funcionalidades fueron sometidas a cuatro iteraciones de prueba: en la iteración inicial los resultados obtenidos fueron: un WARNING, dos FAILURE, dos OK, debido a errores de conexión hacia el servidor LDAP en uso, puesto que la aplicación debía mostrar una alerta de error de conexión al servidor LDAP en el proceso de filtrado cuando los parámetros de conexión a este sean incorrectos o no exista ese servidor LDAP. Al realizar la segunda iteración de prueba el resultado obtenido fue: cero WARNING, un FAILURE, tres OK, debido a errores de filtrado del servidor LDAP en uso, no se obtenía lo que realmente el usuario esperaba como resultado del filtrado acorde a los criterios especificados por este. Una vez ejecutadas la tercera y cuarta iteración de pruebas, los resultados fueron satisfactorios, mostrándose de la siguiente manera: cero WARNING, cero FAILURE, cuatro OK, lo que se traduce como: cero advertencias, cero errores y que las cuatro funcionalidades probadas muestran un correcto funcionamiento.

A continuación se ilustran algunos de los resultados obtenidos en la vista de la consola del IDE Eclipse Kepler.

```

-----
.Probando el Buscar Usuarios en BD
Cantidad de usuarios obtenidos:8
Tiempo de ejecución:43

.Probando el Filtrar Usuarios del LDAP
Cantidad de usuarios obtenidos:8
Tiempo de ejecución:23

Time: 0,066
OK (2 tests)
-----
    
```

Figura 22: Comparación del proceso de creación de cuentas de usuario a través de un servidor LDAP o manualmente usando la BD embebida de REKO.

En la figura 22 se observa el proceso de filtrado de usuarios de un servidor LDAP, el cual arroja ocho resultados en un tiempo de 23 milisegundos, mientras que buscar estos mismos usuarios en la BD embebida de REKO tomó un tiempo de 43 milisegundos; lo cual demuestra que mediante el filtrado e importación de usuarios de un servidor LDAP se agiliza el proceso de creación de cuentas de usuario.

3.3.4 Resultados obtenidos

- ✓ Todos los problemas de tratamiento de errores que fueron descubiertos durante el desarrollo fueron solucionados.
- ✓ La tercera y cuarta iteración de las pruebas dieron resultados satisfactorios, de esta manera, se demuestra que el sistema cumple con cada uno de los requisitos que lo conforman, siendo el módulo completamente funcional.

3.4 Conclusiones parciales

Una vez terminada la implementación y aplicadas las pruebas, se obtuvieron las siguientes conclusiones:

- ✓ Se expusieron los diagramas de componentes correspondientes a cada diagrama de clases, así como algunos de los algoritmos más relevantes utilizados en el desarrollo de la solución.
- ✓ Las pruebas de Caja Blanca y Caja Negra aplicadas mostraron una respuesta efectiva en cuanto a la obtención de la información requerida por el usuario, validando de esta forma el cumplimiento de los requisitos funcionales.
- ✓ El empleo de estas pruebas permitió verificar la solidez de la implementación del sistema, así como validar que realmente el módulo realice lo que se espera de él.

CONCLUSIONES

Con la realización del presente trabajo se arriba a las siguientes conclusiones:

- ✓ Mediante el análisis de la literatura especializada fueron identificados los elementos teóricos que sustentan el desarrollo de la investigación.
- ✓ El estudio de los sistemas que poseen mecanismos de integración con servidores LDAP, tanto a nivel nacional como internacional, sentó las bases para el desarrollo de la solución.
- ✓ La extensión del módulo de seguridad para el Replicador de datos REKO permite gestionar la seguridad a través de un servidor LDAP, garantizando la autenticación, filtrado, importación, búsqueda y sincronización; lo que automatiza la creación de cuentas de usuarios.
- ✓ Las pruebas aplicadas permitieron validar las funcionalidades desarrolladas, verificando que cumplan con todos los requisitos definidos.

RECOMENDACIONES

Para el desarrollo de futuras versiones del Replicador de datos REKO se recomienda:

- ✓ Incluir soporte para importar los usuarios a partir de un fichero LDIF o desde un fichero en texto plano.
- ✓ Agregar al módulo la posibilidad de integrarse con otros servidores LDAP como Oracle Internet Directory y Novell Directory Services.

REFERENCIAS BIBLIOGRÁFICAS

1. Definición de seguridad informática. [En línea] [Citado el: 5 de octubre de 2014.] <http://definicion.de/seguridad-informatica/#ixzz2E0VwS7Ek5>.
2. **Erb, Markus.** Gestión de Riesgo en la Seguridad Informática. [En línea] [Citado el: 5 de octubre de 2014.] http://protejete.wordpress.com/gdr_principal/seguridad_informacion.
3. **Meneses, Dorela B. Carrasquel.** [En línea] [Citado el: 6 de octubre de 2014.] <http://carmen.jimdo.com/informaci%C3%B3n-personalizada/dorela-carrasquel/>.
4. Instalación y configuración de un servidor RADIUS. *Instalación y configuración de un servidor RADIUS.* [En línea] 16 de abril de 2014. [Citado el: 6 de octubre de 2014.] <http://www.grc.upv.es/docencia/tra/PDF/Radius.pdf>.
5. **Leandro Alegsa.** ALEGSA. [En línea] [Citado el: 6 de octubre de 2014.] <http://www.alegsa.com.ar/Dic/protocolo%20de%20autenticacion.php>.
6. **Cayuqueo, Sergio.** Cayu- Wiki. [En línea] 17 de enero de 2014. [Citado el: 7 de octubre de 2014.] http://wiki.cayu.com.ar/doku.php?id=manuales:servidor_freeradius.
7. Red Hat Enterprise Linux 4: Manual de referencia. *Capítulo 19.* [En línea] 2005. [Citado el: 6 de octubre de 2014.] <http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-kerberos.html>.
8. **Microsoft.** Autenticación NTLM. [En línea] enero de 2005. [Citado el: 7 de octubre de 2014.] <http://msdn.microsoft.com/es-es/library/cc783005%28v=ws.10%29.aspx>.
9. **Balmaceda, Luis Carlos Flores.** Generalidades Seguridad de Redes. [En línea] 12 de marzo de 2014. [Citado el: 7 de octubre de 2014.] <http://luiscafb.blogspot.com/>.
10. **Gonzalez, Gonzalez, Sergio.** Integración de redes con OpenLDAP, Samba, CUPS y PyKota. [En línea] 2004. [Citado el: 7 de octubre de 2014.] <http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-openldap-samba-cups-python/html/ldap+samba+cups+pykota.html#openldap-que-es>.
11. Servicios de directorio, LDAP. *Capítulo 6.* [En línea] 3 de febrero de 2014. [Citado el: 7 de octubre de 2014.] www.profesordeinformatica.com/descargas/capitulo6-ldap.pdf.
12. **Sergio de los Santos.** HISPASEC. [En línea] 25 de junio de 2009. [Citado el: 7 de octubre de 2014.] <http://unaaldia.hispasec.com/2009/06/mitos-y-leyendas-el-directorio-activo-i.html>.
13. Proyectos ULS. *¿Qué es un servicio de directorio?* [En línea] 2014. [Citado el: 7 de octubre de 2014.] <http://proyectos.uls.edu.sv/wiki/images/b/be/Presentacion-ldap-redesII-2014-equipo-4.pdf>.
14. **Correa, Yefeson Gómez.** slideshare. [En línea] 26 de marzo de 2014. [Citado el: 7 de octubre de 2014.] <http://es.slideshare.net/yefesongomezcorrea/yeferson-ldap>.
15. **Ferrando, Antonio Félix.** Openaccess. *Desarrollo de Aplicación Catalogo TIC de la Dirección General de Tecnologías de Información de la Generalitat Valenciana.* [En línea] 10 de enero de 2014.

[Citado el: 7 de octubre de 2014.] <http://openaccess.uoc.edu/webapps/o2/bitstream/10609/28521/3/afelixfeTFM0114memoria.pdf>.

16. **Pradas, Rafael Calzada.** Introducción al servicio de directorio. [En línea] 2001. [Citado el: 7 de octubre de 2014.] <http://www.rediris.es/ldap/doc/ldap-intro.pdf>.

17. Microsoft. *Utilización de LDIFDE para importar y exportar a Active Directory objetos del directorio.* [En línea] 2006. [Citado el: 7 de octubre de 2014.] <http://support.microsoft.com/kb/237677/es>.

18. E-CENTRO. [En línea] 2012. [Citado el: 7 de octubre de 2014.] http://centrodeartigo.com/articulos-enciclopedicos/article_92576.html.

19. **Bedoya, Yina Paola Garzon.** slideshare. *LDAP.* [En línea] 28 de marzo de 2014. [Citado el: 7 de octubre de 2014.] <http://es.slideshare.net/YinaGarzon/presentacin1-32838976>.

20. phpLDAPadmin. *phpLDAPadmin main page.* [En línea] 8 de abril de 2013. [Citado el: 7 de octubre de 2014.] http://phpldapadmin.sourceforge.net/wiki/index.php/Main_Page.

21. Webmin. *Introduction to Webmin.* [En línea] 2011. [Citado el: 7 de octubre de 2014.] Disponible en: <http://www.webmin.com/intro.html>.

22. WebSite Scripts. *KnowledgeBase Manager Pro.* [En línea] 2011. [Citado el: 7 de octubre de 2014.] <http://www.web-site-scripts.com/knowledge-management/overview.html>.

23. SmarterMail. *Import New Users Using LDAP.* [En línea] 13 de enero de 2013. [Citado el: 7 de octubre de 2014.] <http://portal.smartertools.com/kb/a2713/import-new-users-using-ldap.aspx>.

24. Symfony Project. *upSimpleLdapPlugin.* [En línea] 2011. [Citado el: 9 de octubre de 2014.] <http://www.symfony-project.org/plugins/upSimpleLdapPlugin>.

25. **Martínez, Ramón A. Anglada.** *Plataforma de Gestión de Servicios Telemáticos en GNU/LINUX.* 2012. págs. 46-57. Vol. 11.

26. **PRESSMAN, R. S.** *Ingeniería del Software, Un Enfoque Práctico.* 6. s.l. : McGraw-Hil, 2002. pág. 640. Técnicas para facilitar las especificaciones de la aplicación.

27. **Rengifo, Adrian.** [En línea] 12 de agosto de 2014. [Citado el: 11 de octubre de 2014.] <http://arengifoingsoft.blogspot.com/2014/08/metodologias-tradicionales.html>.

28. **Eduardo Martinez Fustero.** Comunidad. [En línea] 7 de febrero de 2014. [Citado el: 11 de octubre de 2014.] <http://comunidad.iebschool.com/iebs/agile-scrum/que-es-agile/>.

29. **Torrecilla, Pablo.** nosolopau. [En línea] 7 de junio de 2013. [Citado el: 13 de octubre de 2014.] <http://nosolopau.com/2012/06/07/mas-sobre-el-proceso-unificado-agil-fases-y-disciplinas/>.

30. ingenieriadesoftware. [En línea] 2013. [Citado el: 13 de octubre de 2014.] http://ingenieriadesoftware.mex.tl/63758_aup.html.

31. **Sánchez, T.R.** *Metodología de desarrollo para la actividad productiva de la UCI*. La Habana : s.n., 2014.
32. **Saco, Guillermo González-Vallés.** Ciberaula. *POO (Programación Orientada a objetos)*. [En línea] 2014. [Citado el: 13 de octubre de 2014.] http://www.ciberaula.com/articulo/tecnologia_orientada_objetos/.
33. **Q Chanatasig.** Paradigma de la programación orientada a aspectos. [En línea] 2011. [Citado el: 13 de octubre de 2014.] <http://repositorio.utc.edu.ec/bitstream/27000/1843/1/T-UTC-1334.pdf>.
34. **Asteasuain, Fernando.** *PROGRAMACIÓN ORIENTADA A ASPECTOS. Análisis del paradigma*. 2012.
35. **Suarez, Eivys Hernández.** *Administración de las Configuraciones y Definiciones de Seguridad del SIGEP*. 2008.
36. **D, PARSONS.** *Foundational Java: Key Elements and Practical Programming*. 2012. pág. 532.
37. Apache Tomcat. *THE APACHE SOFTWARE FOUNDATION*. [En línea] 9 de mayo de 2013. [Citado el: 23 de noviembre de 2014.] <http://tomcat.apache.org/>.
38. Definición de Eclipse. *bilib*. [En línea] [Citado el: 24 de noviembre de 2014.] <http://www.bilib.es/recursos/catalogo-de-aplicaciones/ficha-de-aplicacion/app/eclipse/>.
39. **Maria Isabel Pacheco.** Scribd. *Herramientas CASE para el proceso de desarrollo de Software*. [En línea] [Citado el: 15 de octubre de 2014.] <http://es.scribd.com/doc/52162574/Herramientas-CASE-para-el-proceso-de-desarrollo-de-Software>.
40. ¿Qué es Java Enterprise Edition ? [En línea] [Citado el: 15 de noviembre de 2014.] <http://webcache.googleusercontent.com/search?q=cache:ONUZnvPeqslJ:www.java.com/es/download/faq/techinfo.xml+&cd=1&hl=es&ct=clnk&gl=cu>.
41. Searchsoa. *Java*. [En línea] 2014. [Citado el: 15 de noviembre de 2014.] <http://searchsoa.techtarget.com/definicion/Java>.
42. Unified Modeling Language. *Resource Page*. [En línea] 22 de mayo de 2014. [Citado el: 20 de noviembre de 2014.] <http://www.uml.org/>.
43. **JACOBSON, I. B. G. R.** *El Proceso Unificado de Desarrollo de Software*. s.l. : Pearson Education S.A, 2010. pág. 464. Vol. 2.
44. **Sosa, Ángel Gabriel Olivera.** Planificación y Modelado. [En línea] 6 de septiembre de 2010. [Citado el: 24 de noviembre de 2014.] <http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales>.
45. **SUAZA, KATERINE VILLAMIZAR.** *Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP*. Facultad de Minas - Departamento de Ciencias de la Computación y de la Decisión : Universidad Nacional de Colombia, 2013.

46. **REYNOSO, C. K., NICOLAS.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Buenos Aires : s.n., 2004. pág. 56.
47. **SCHMULLER, J.** Aprendiendo UML en 24 horas. [En línea] 2000. [Citado el: 24 de enero de 2015.] <https://notasdelprofesor.files.wordpress.com/2013/08/adaptacion-de-uml.pdf>.
48. **GARCÍA LIRA, K. G. D., AILEC y TEJERA HERNÁNDEZ, DAYANA CARIDAD.** *Arquitectura y Patrones de diseño*. Universidad de la Ciencias Informáticas : s.n. Vol. Tema I. Culminación de la Fase de Elaboración.
49. Ingenio DS. *Patrones Arquitectónicos*. [En línea] 16 de septiembre de 2013. [Citado el: 24 de enero de 2015.] <https://ingeniods.wordpress.com/2013/09/16/patrones-arquitectonicos/>.
50. **Peláez, Juan.** Arquitectura basada en capas. [En línea] 26 de mayo de 2009. [Citado el: 24 de enero de 2015.] <http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-capas/>.
51. *ingenieriadelsoftware-iutep. DISEÑO ORIENTADO A OBJETOS.* [En línea] 2005. [Citado el: 24 de enero de 2015.] <http://ingenieriadelsoftware-iutep.wikispaces.com/>.
52. Patrones de Diseño GRASP. [En línea] [Citado el: 24 de enero de 2015.] <http://es.scribd.com/doc/53315954/Patrones-de-Diseno-GRASP>.
53. **GROSSO, A.** Patrones GRASP. [En línea] 2011. [Citado el: 24 de enero de 2015.] <http://www.practicadesoftware.com.ar/2011/03/patrones-grasp/>.
54. Unidad II Analisis y Diseño _ Arquitectura y Diseño de Software. [En línea] 5 de julio de 2014. [Citado el: 24 de enero de 2015.] <https://mariamerica15.wordpress.com/2014/07/05/unidad-ii-analisis-y-diseno/>.
55. **Pressman, Roger S.** *Ingeniería del software, un enfoque práctico*. 7ma. 2014.
56. Modelo de Implementación. [En línea] 1 de junio de 2013. [Citado el: 1 de marzo de 2015.] <http://ithleovi.blogspot.com/>.
57. **TRACY, GARCIA SAAVEDRA MADELINE.** Diagrama de Componentes. [En línea] [Citado el: 1 de marzo de 2015.] <http://virtual.usalesiana.edu.bo/web/practica/archiv/componn.doc>.
58. **Martínez, Rafael.** god..INFORMATIc.com. [En línea] 24 de enero de 2014. [Citado el: 1 de marzo de 2015.] <http://rafmartz1999.blogspot.com/2014/01/codigo-de-fuente.html>.
59. **Ramírez, Adilén Sánchez.** laccei. *Procedimiento para la realización de ensayos de Aceptación y/o Piloto*. [En línea] 24 de julio de 2014. [Citado el: 1 de marzo de 2015.] <http://www.laccei.org/LACCEI2014-Guayaquil/RefereedPapers/RP049.pdf>.
60. **JURISTO, N., MORENO, ANA M., VEGAS, SIRA.** Técnicas de evaluación de software. [En línea] 2005. [Citado el: 1 de marzo de 2015.] http://eva.uci.cu/file.php/257/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_3/Comun/Tecnicas_de_evaluacion_de_software_Jurisco-Moreno.pdf.

61. **Reyna Cruz, Arely.** Facultad de Ingeniería. *Universidad Autónoma de Chihuahua.* [En línea] 2011. [Citado el: 1 de marzo de 2015.] <http://www.buenastareas.com/ensayos/Pruebas-De-Caja-Negra/1477804.html>.

62. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* 6ta. 2002. pág. 423.

GLOSARIO DE TÉRMINOS

BD: Base de Datos. Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Framework: Plataformas o herramientas del mundo de la informática que le proveen a los programadores un grupo de facilidades en el ámbito para la cual han sido creadas.

GPS: Sistema que permite determinar en todo el mundo la posición de un objeto con una precisión de hasta centímetros.

HTML: Lenguaje compuesto de una serie de etiquetas o marcas que permite definir el contenido y la apariencia de las páginas Web.

HTTPS: Protocolo Seguro de Transferencia de Hipertexto. Garantiza la seguridad de las comunicaciones entre el usuario y el servidor Web al que este se conecta.

IDE: Entorno de Desarrollo Integrado. Herramienta que se usa para facilitar el desarrollo de software.

JSP: Tecnología que ayuda a los desarrolladores de software a crear páginas Web dinámicas basadas en HTML.

LDAP: Protocolo Ligero de Acceso a Directorios. Protocolo basado en el Modelo Cliente-Servidor para acceder a un servicio de directorio.

LDIF: Formato de Intercambio de LDAP. Archivo utilizado para importar y exportar información entre servidores de directorio basados en LDAP.

Malware: Tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario.

NTLM: Es un conjunto de protocolos de seguridad de Microsoft que proporciona autenticación, integridad y confidencialidad de los usuarios.

PHP: Lenguaje de código abierto muy popular y adecuado para el desarrollo Web y que puede ser embebido en HTML.

Plugin: Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Servlet: Clases escritas en Java que se utilizan en un servidor Web para extender sus capacidades de respuesta a los clientes.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

SSL: Proporciona cifrado de datos, autenticación de servidores, integridad de mensajes y, opcionalmente, autenticación de cliente para conexiones TCP/IP.

TCP/IP: Protocolo de Control de Transmisión/ Protocolo de Internet: Protocolo de red en los que se basa Internet y que permiten la transmisión de datos entre computadoras.

Token: Pequeño dispositivo de hardware que los usuarios cargan consigo para autorizar el acceso a un servicio de red. El dispositivo puede ser en forma de una tarjeta inteligente o puede estar incorporado en un objeto utilizado comúnmente, como un llavero.

UML: Lenguaje de modelado para visualizar, especificar, construir y documentar un sistema.