

Universidad de las Ciencias Informáticas

Facultad 1



**Módulo para la importación y sincronización de usuarios de un
LDAP en Smart Keeper**

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

Autores: Eddy Yusiel Pupo Rodríguez
Maidelys Rodríguez Fernández

Tutor: Ing. Luis Enrique Sánchez Arce
Co-Tutora: Ing. Yurisleidy Hernández Moya

La Habana, Junio de 2012

Declaración de autoría

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

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

Autores:

Maidelys Rodríguez Fernández _____


Eddy Yusiel Pupo Rodríguez _____

Ing. Luis Enrique Sánchez Arce

Tutor

Ing. Yurisleidy Hernández Moya

Co-Tutora



"El aspecto fundamental en el cual la juventud debe señalar el camino es precisamente en el aspecto de ser vanguardia en cada uno de los trabajos que le compete."

Che

Datos de contactos

Ing. Luis Enrique Sánchez Arce

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de la Ciencias Informáticas en el año 2008. Pertenece al proyecto FCI desde el año 2006. Actualmente se desempeña como Líder del proyecto antes mencionado.

Correo electrónico: lesanchez@uci.cu

Ing. Yurisleidy Hernández Moya

Graduada de Ingeniera en Ciencias Informáticas en el 2008. Pertenece al proyecto FCI desde el año 2010 donde se desempeña como desarrolladora. Ha ejercido como oponente en varias tesis de grado y ha formado parte de tribunales de tesis con otros roles. Posee categoría docente de Instructor.

Correo electrónico: ymoya@uci.cu.

Agradecimientos

A mis papis lindos (Olga y Leudelio) por ser lo más grande que tengo en mi vida. Gracias por su cariño, apoyo y confianza, sin lo cual no hubiese podido hacer realidad dos de mis más grandes sueños: culminar mi carrera profesional y hacerlos sentirse orgullosos de esta persona que los adora. Los amo mucho y le pido a Dios a diario mucha salud y felicidad para que puedan estar siempre a mi lado y nunca se me alejen porque los necesito mucho.

A mi tata, mi hermanita por ser mi otra mitad y por haberme dado ese tesoro que es parte de mi felicidad y mi corazón.

A mis abuelos que adoro un montón. Gracias por apoyarme, quererme y aconsejarme. Los quiero siempre a mi lado.

A mi compañero de tesis, amigo y esposo, por enseñarme el significado del amor, por estar a mi lado desde el primer día que entré a la universidad, por hacerme reír siempre, por ayudarme y cuidar de mí. Juntos hemos logrado hacernos profesionales, para un día servir de ejemplo ante nuestros hijos. Te amo mi puchi.

A mis suegros por ser tan especiales, por quererme como una hija más y darme tanto amor y confianza. Los quiero mucho y son como unos padres para mí. Siempre estaré aquí para ayudarlos y nunca los sacaré de mi mente.

A mi rubio lindo Davicito, que lo adoro con el alma. Gracias por ser tan dulce conmigo. Quiero ser un ejemplo para ti y te deseo todo lo bueno del mundo.

A mi princesita Lily por quererme tanto y siempre tener guardadas para mí flores, cartas, dibujos y por ser tan linda y tierna conmigo.

A mi cuñado por apreciarme, ayudarme y por haber hecho posible la llegada de mi tesoro a este mundo.

A mi tío Mundy, por estar ahí siempre que lo necesité. Gracias tío, te quiero como un padre y eres muy excepcional para mí.

A mis tíos Cary y Evelio por ayudarme siempre desinteresadamente. Sepan que soy la hija que no tuvieron, son más que especiales en mi vida y nunca los dejaré solos.

A mis tías Ivette, Negra y Leida por preocuparse, aconsejarme y ayudarme.

A mis hermanos y sobrinos por darme su cariño y saber que puedo contar con ellos porque estarán ahí.

A mi cuñada Maily por ser una hermana más para mí.

A mis amigas que las admiro mucho Yari, Lianis por estar ahí y nunca olvidarse que existo.

A Jenlys y Elier. Gracias por estar ahí a mi lado apoyándome, por darme consuelo y ayudarme cuando más lo necesité. Gracias por su apoyo incondicional.

A las chicas del apartamento, Yanet, Soyma, Isabel, Jennifer, Adis, Alianny, Nana, por apoyarme y ayudarme de una forma u otra cuando lo he necesitado.

Maidelys Rodríguez Fernández

Agradezco en primer lugar a mis padres. Gracias por entregarlo todo para construir mi futuro, por educarme de la forma que lo han hecho, por darme todo su amor y cariño.

Gracias a mi hermana por siempre estar apoyándome en todos mis pasos. Gracias a mi sobrina Lili por su ternura. Sé que un día seguirá mis pasos.

A mis abuelas Rosa y Orismelda porque de estar vivas se sentirían muy orgullosas de su nieto, el profesional que siempre predijeron.

Gracias a mi compañera de tesis por ser mi esposa y amiga. Gracias a ti conocí el verdadero amor, solo espero poder contar con tu compañía por el resto de la vida. Me haces realmente feliz. Este es uno de nuestros sueños hecho realidad gracias a nuestro esfuerzo. Te amo.

Gracias a mis tíos por ayudarme en todo momento, en las buenas y las malas. Muchísimas gracias a mis nuevos tíos Cary y Evelio por comportarse como unos padres para mí. Gracias a ustedes el calor de madre y padre nunca nos faltó.

Gracias a mis suegros Olga y Leudelio por formar parte de mi vida, son personas muy especiales que tendré siempre en mi corazón.

Gracias a los compañeros de ETECSA de Holguín por haber contribuido a mi formación profesional.

Eddy Yusiél Pupo Rodríguez

Agradecemos a nuestro tutor por su apoyo incondicional, dedicación, preocupación e interés de ayudarnos en todo momento. Sin su ayuda no hubiese sido posible lograr un trabajo con calidad. Gracias por compartir tu gran experiencia y profesionalidad, seguiremos tu ejemplo.

A nuestros compañeros de grupo y al equipo de desarrollo de Smart Keeper y AiresProxy, en especial a Eduardo y Rubén.

A nuestros profesores por ayudar a nuestra formación.

A todos los que de una forma u otra han contribuido a la realización de este sueño.

Dedicatoria

Dedico este trabajo a mis padres, a quienes jamás encontraré la forma de agradecer el amor, aliento, comprensión y apoyo brindado en los momentos buenos y malos de mi vida, este logro es dedicado a ellos y quiero que sepan que mis esfuerzos son inspirados en cada uno de ustedes.

Gracias por demostrarme que todo en la vida se puede lograr.

Gracias por ser parte de lo que más amo.

Gracias por estar en mi vida.

....Gracias....

Maidelys Rodríguez Fernández

A mis padres por ser lo que más quiero en el mundo, por tantos años de sacrificio y esmero, por educarme y convertirme en un hombre de bien.

Eddy Yusiel Pupo Rodríguez

Resumen

En la actualidad existe una creciente tendencia a centralizar la gestión de los usuarios de las redes informáticas, debido a las grandes ventajas que esto proporciona. En este sentido, han ganado gran importancia los servicios de directorio. Por este motivo un gran número de instituciones cuentan con un servidor LDAP para almacenar la información de sus usuarios. La creciente popularidad de esta tecnología se debe a su flexibilidad y al hecho de que puede ser integrada a un número creciente de sistemas. Los filtros de contenido web generalmente poseen mecanismos de integración con LDAP a fin de identificar usuarios y grupos relevantes. Sin embargo, el filtro de contenido web Smart Keeper, desarrollado en la Universidad de las Ciencias Informáticas, carece de dicha característica.

La problemática anteriormente planteada justifica la presente investigación por lo que se define como objetivo principal, desarrollar un módulo que permita importar y sincronizar los usuarios de un servidor LDAP en Smart Keeper. De esta forma, se logra automatizar el proceso de creación de cuentas de usuarios en dicho sistema. Para lograr el objetivo planteado se hizo uso del *framework* Symfony y el proceso de desarrollo estuvo guiado por la metodología RUP. Las herramientas empleadas facilitaron el desarrollo de un *plugin* para la Interfaz de Administración Web de Smart Keeper. Dicho *plugin* fue probado e integrado al sistema con resultados satisfactorios.

Palabras clave: filtro de contenido web, LDAP, plugin, servicio de directorio, Smart Keeper.

Índice General

INTRODUCCIÓN	1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA	5
1.1 Servicio de directorio LDAP	5
1.2 Características de LDAP	6
1.3 Estructura del directorio	7
1.4 Operaciones sobre el directorio LDAP	8
1.5 Herramientas con mecanismos similares	9
1.5.1 Ámbito Internacional	9
1.5.2 Ámbito Nacional	11
1.6 Tecnologías y herramientas	12
1.6.1 Lenguaje de programación	13
1.6.2 Sistema Gestor de Base de Datos	14
1.6.3 Framework	14
1.6.4 Entorno de Desarrollo Integrado (IDE)	15
1.6.5 Herramienta CASE	15
1.7 Metodología de desarrollo de software	16
1.8 Conclusiones parciales	17
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA	18
2.1 Flujo actual de los procesos	18
2.2 Propuesta de solución	20
2.3 Modelado del dominio	21
2.3.1 Diagrama de clases del Modelo de Dominio	22
2.3.2 Definición de las Clases del Modelo de Dominio	22

2.4 Modelado del sistema	22
2.4.1 Requerimientos funcionales	23
2.4.2 Requerimientos no funcionales	24
2.4.3 Diagrama de Casos de Uso del Sistema	25
2.4.4 Descripción de Casos de Uso del Sistema	26
2.5 Conclusiones parciales	31
CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA	32
3.1 Estilos arquitectónicos y patrones de diseño	32
3.2 Diagramas de clases del análisis	35
3.3 Diagramas de clases del diseño	36
3.4 Diagramas de interacción	41
3.5 Diagrama de despliegue	45
3.6 Modelo de datos	46
3.6.1 Tablas del modelo de datos	48
3.7 Conclusiones parciales	50
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS	51
4.1 Diagrama de componentes	51
4.1.1 Descripción de los componentes	53
4.2 Pantallas principales de la aplicación	55
4.3 Validación del sistema	57
4.3.1 Técnica de prueba	57
4.3.2 Tipos de pruebas aplicadas	57
4.3.3 Resultado de las pruebas	58
4.4 Conclusiones parciales	58
CONCLUSIONES	59

RECOMENDACIONES	60
GLOSARIO DE TÉRMINOS	61
REFERENCIA BIBLIOGRÁFICA	63
BIBLIOGRAFÍA	65
A. FIGURAS RELACIONADAS	66
B. DESCRIPCIÓN DE CASOS DE USO DEL SISTEMA	70
C. DIAGRAMAS DE CLASES DEL DISEÑO	77
D. DIAGRAMAS DE INTERACCIÓN	82

Índice de figuras

1.1: Árbol de directorio.....	7
2.1: Formulario para la creación de cuentas de usuario en Smart Keeper.....	19
2.2: Esquema de autenticación de Smart Keeper a través de squid_ldap_auth.....	20
2.3: Diagrama del Modelo de dominio.....	22
2.4: Diagrama de Casos de Uso del Sistema.....	26
3.1: Arquitectura Modelo Vista Controlador.....	33
3.2: Diagrama de clases del análisis. CU Gestionar configuración de conexión.....	35
3.3: Diagrama de clases del análisis. CU Importar usuarios.....	35
3.4: Diagrama de clases del análisis. CU Sincronizar usuarios.....	36
3.5: Diagrama de clases del diseño. CU Importar usuarios. Sección Listar usuarios del LDAP.....	37
3.6: Diagrama de clases del diseño. CU Importar usuarios. Sección Actualizar listado.....	38
3.7: Diagrama de clases del diseño. CU Importar usuarios. Sección Importar un usuario.....	39
3.8: Diagrama de clases del diseño. CU Importar usuarios. Sección Importar usuarios seleccionados.....	40
3.9: Diagrama de clases del diseño. CU Importar usuarios. Sección Importar usuarios filtrados.....	41
3.10: Diagrama de secuencia. CU Importar usuarios. Sección Listar usuarios del LDAP.....	42
3.11: Diagrama de secuencia. CU Importar usuarios. Sección Actualizar listado de usuarios.....	42
3.12: Diagrama de secuencia. CU Importar usuarios. Sección Importar un usuario.....	43
3.13: Diagrama de secuencia. CU Importar usuarios. Sección Importar usuarios seleccionados.....	44
3.14: Diagrama de secuencia. CU Importar usuarios. Sección Importar usuarios filtrados.....	45
3.15: Diagrama de despliegue.....	46
3.16: Modelo de datos.....	47
4.1: Diagrama de componentes de LdapPlugin.....	51
4.2: Diagrama de componentes del sub-módulo ldap_config.....	52
4.3: Diagrama de componentes del sub-módulo ldap_user.....	53
4.4: Pantalla principal del sub-módulo ldap_user.....	55
4.5: Pantalla principal del sub-módulo ldap_config.....	56
4.6: Formulario de edición del sub-módulo ldap_config. Pestaña Sincronización.....	56
4.7: Formulario de configuración de los parámetros de los usuarios a importar.....	57

Índice de tablas

2.1: Requerimientos funcionales.....	23
2.2: Descripción textual del Caso de Uso Importar usuarios.	30
3.1: Descripción de la tabla ldap_config.	49
3.2: Descripción de la tabla ldap_user.	50
4.1: Descripción del componente importLdapUsersTask.....	54
4.2: Descripción del componente synchronizeTask.	55

Introducción

Dada la creciente necesidad de información, sobre todo en Internet, la popularidad de los servicios de directorio ha aumentado en la última década y son hoy una opción común para muchas instituciones alrededor del mundo. Tal es el caso de los servidores LDAP (*Lightweight Directory Access Protocol*), los cuales son servidores de datos optimizados para dar una respuesta rápida a consultas de lectura y están orientados al almacenamiento de datos de los usuarios a modo de directorio. La gran aceptación de esta tecnología se debe a su flexibilidad y al hecho de que se integra con un número creciente de sistemas.

Entre las principales ventajas de emplear servidores LDAP se encuentra la posibilidad de centralizar la administración de los usuarios en un único lugar. Por otra parte, son muy utilizados como mecanismo de autenticación para los distintos servicios de un sistema informático, gracias a que la mayoría de las aplicaciones cuentan con mecanismos de integración con LDAP. De esta forma, bastaría con solo crear las cuentas de usuarios en el servidor y dichos usuarios tendrían acceso al resto de las aplicaciones. Además, al dar de baja a un usuario, este perdería el acceso al resto de los sistemas sin necesidad de eliminarlo por separado en cada uno. LDAP es empleado además, como libreta de direcciones para el completamiento automático de direcciones en los servicios de correo electrónico.

Los filtros de contenidos web, con frecuencia admiten la conexión con un Directorio Activo¹ a fin de identificar usuarios y grupos relevantes y permitir la definición de políticas en función de ellos [1]. Un filtro de contenido web es un *software* que permite controlar el acceso de los usuarios a determinados contenidos de Internet, a partir de un conjunto de reglas predeterminadas. En la Universidad de las Ciencias Informáticas (UCI) se desarrolla el filtro de contenido Smart Keeper, siendo este el único de su tipo en Cuba. El funcionamiento de este sistema está basado fundamentalmente en los usuarios, los grupos y las políticas de navegación que le son asignadas a estos.

Actualmente en Smart Keeper el proceso de creación de las cuentas de usuario se realiza manualmente. Este hecho puede provocar un rechazo por parte de los posibles clientes del sistema a la hora de adquirir el producto, ya que en una red con gran cantidad de usuarios el proceso se volvería muy tedioso. Además,

¹ Implementación de LDAP de Microsoft.

Introducción

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 existe un servidor LDAP que contiene la información de los usuarios, sin embargo, Smart Keeper no cuenta con un mecanismo que le permita aprovechar la información almacenada en dichos servidores. Esto constituye una desventaja para el sistema Smart Keeper en comparación con otros filtros similares tales como: Websense², CyberPatrol SiteSURV³, Optenet⁴ y Puresight⁵, los cuales cuentan con soporte para integrarse a LDAP.

Teniendo en cuenta la situación problemática descrita anteriormente se define como **problema a resolver**: ¿Cómo lograr que Smart Keeper pueda aprovechar la información almacenada en los servidores LDAP para automatizar el proceso de creación de cuentas de usuarios?

Se plantea como **Idea a defender**: La implementación de un módulo que permita importar y sincronizar los usuarios de un servidor LDAP en Smart Keeper automatizará el proceso de creación de cuentas de usuarios.

En vista a darle solución al problema antes mencionado se define como **objeto de estudio** de la investigación, las aplicaciones que poseen mecanismos de integración con LDAP.

Para darle solución al problema descrito, se ha planteado el siguiente **objetivo general**: Desarrollar un módulo que permita automatizar el proceso de creación de cuentas de usuarios en Smart Keeper, a partir de los datos de un servidor LDAP.

Del cual se desglosan los siguientes **objetivos específicos**:

- Sistematizar acerca de las principales aplicaciones que poseen mecanismos de integración con LDAP.
- Diseñar un módulo que permita importar y sincronizar de manera automatizada los usuarios de un servidor LDAP hacia Smart Keeper.

² <http://www.websense.com/>

³ <http://www.cyberpatrol.com/cpnetfilter.asp>

⁴ <http://www.optenet.com/>

⁵ <http://www.puresight.com/>

Introducción

- Implementar las funcionalidades diseñadas.
- Validar las funcionalidades de la aplicación desarrollada.
- Integrar la aplicación al sistema Smart Keeper.

Resultados esperados:

Se pretende que al concluir la investigación, el sistema Smart Keeper cuente con una funcionalidad que permita realizar el proceso de importación y sincronización de los usuarios de un servidor LDAP, de forma que se automatice la creación de cuentas de usuario.

Para un mejor desarrollo de la investigación se utilizaron los siguientes métodos científicos:

- **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 de la versión actual del protocolo LDAP.
- **Analítico-Sintético:** permitió descomponer en varias partes los elementos relacionados con las herramientas que poseen mecanismos de integración con LDAP. También permitió analizar por separado cada elemento y obtener una comprensión total de los elementos analizados para posteriormente formular conclusiones.

A continuación se hace una breve descripción de cada uno de los capítulos del trabajo de diploma:

Capítulo 1: Fundamentación teórica, se expone un análisis de todos los aspectos teóricos necesarios para comprender el problema en cuestión y todos los conceptos relacionados con este. Se estudia el funcionamiento del servicio de directorio LDAP y se relacionan las distintas implementaciones de este. También se muestra el resultado del estudio del estado del arte de algunas aplicaciones que cuentan con mecanismos de integración con 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 del sistema, contiene una descripción detallada del flujo de los procesos que dan surgimiento al problema. Se muestra la propuesta de solución a partir de un análisis de las

Introducción

características de Smart Keeper y un modelo de dominio que facilita la comprensión de su funcionamiento. Además, se relacionan los requisitos funcionales y no funcionales que debe tener el módulo a desarrollar para satisfacer las necesidades del cliente. También se define el diagrama de Casos de Uso del Sistema, así como la descripción de estos.

Capítulo 3: Análisis y diseño del sistema, se describe la arquitectura del módulo y se muestran los patrones de diseño a emplear en su implementación. Además, se relacionan algunas ventajas de usar el *framework* Symfony en el desarrollo de aplicaciones web. Se describe el modelo de diseño, el diagrama de despliegue y el modelo de datos.

Capítulo 4: Implementación y pruebas, se muestra el diagrama de componentes y su descripción, para así lograr comprender los elementos de *software* del módulo y las relaciones entre ellos. También se muestran las pantallas principales del módulo desarrollado donde se evidencian las funcionalidades implementadas. Por último, se describen las pruebas realizadas para la validación del sistema y el resultado obtenido.

Capítulo 1

Capítulo 1: Fundamentación teórica

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 LDAP, así como estudiar los mecanismos empleados por las aplicaciones para integrarse a este.

1.1 Servicio de directorio LDAP

A finales de la década de los 70 y principio de los 80 la ITU (*International Telecommunication Union*) comenzó a trabajar en la serie X.400 de los estándares de correo electrónico, los cuales requerían de un directorio de nombres que se pudiera acceder desde la red. Esta necesidad de un directorio global basado en red llevó a la ITU a desarrollar la serie de estándares X.500, que define el Protocolo de Acceso a Directorios (DAP, por sus siglas en inglés).

Las series X.400 y X.500 utilizaban toda la pila de protocolos OSI (*Open Systems Interconnection*) por lo que consumían recursos importantes. Por tal motivo, a principios de la década de los 90 se evidenció la necesidad de crear un protocolo de acceso a directorios más ligero. Como resultado, surge en 1993 una versión del protocolo DAP denominada LDAP. Posteriormente, en el año 1995 fue adoptada la versión 2 y en 1997 fue publicada la versión 3, utilizada hasta la actualidad.

LDAP es un protocolo para el acceso a un servicio de directorio que emplea una estructura de datos similar a X.500 y que opera sobre TCP/IP (*Transfer Control Protocol / Internet Protocol*). Los servicios de directorio facilitan el acceso a información organizada a una gran variedad de aplicaciones. Estos servicios le permiten a los usuarios y aplicaciones autorizadas buscar información de personas, computadoras, aplicaciones y dispositivos de red [2].

Capítulo 1: Fundamentación teórica

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. Estos directorios suelen tener la capacidad para replicar datos y con esto lograr una mayor disponibilidad de la información y reducir los tiempos de respuesta. Los servidores que brindan acceso a este tipo de directorios se denominan servidores LDAP.

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

- Microsoft Active Directory
- Novell Directory Services
- iPlanet - Sun ONE Directory Server
- OpenLDAP
- Red Hat Directory Server
- Apache Directory Server (ApacheDS)
- Fedora Directory Server
- IBM Trivoli DS
- Oracle Internet Directory

Entre las implementaciones anteriores algunas de las más comunes son OpenLDAP y el Directorio Activo de Microsoft, por lo que se desea que la solución implementada soporte ambos. A continuación, las principales características de un directorio LDAP.

1.2 Características de LDAP

Un directorio LDAP se destaca sobre los demás tipos de bases de datos por las siguientes características [3]:

- 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 LDAP y se pueden integrar fácilmente.

Capítulo 1: Fundamentación teórica

- Dispone de un modelo de nombres globales que asegura que todas las entradas son únicas.
- Usa un sistema jerárquico de almacenamiento de información.
- Permite múltiples directorios independientes.
- Funciona sobre TCP/IP y SSL (*Secure Socket Layer*).
- La mayoría de los servidores LDAP son fáciles de instalar, mantener y optimizar.

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

1.3 Estructura del directorio

Los servidores LDAP almacenan la información en una estructura de datos jerárquica (Ver figura 1.1). Esta estructura puede ser distribuida, o sea, puede existir un servidor LDAP que contenga un subárbol de otro. El servidor, al recibir una consulta de un cliente, le responde o le indica dónde puede encontrar la respuesta. 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 [4].

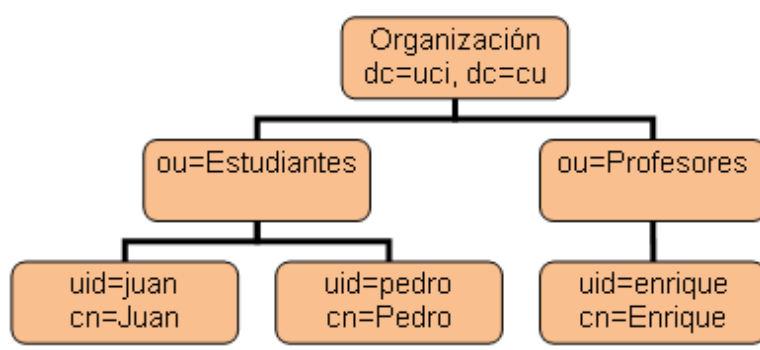


Figura 1.1: Árbol de directorio.

Por ejemplo, el DN de la entrada que almacena los datos de Juan según la *Figura 1.1* es:

`uid=juan, ou=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:

Capítulo 1: Fundamentación teórica

- **uid**: Identificador de usuario.
- **samaccountname**: atributo empleado para almacenar el identificador del usuario en el caso del Directorio Activo se emplea el atributo.
- **ou**: Unidad Organizacional.
- **o**: Organización.
- **c**: País.
- **cn**: Nombre y apellidos.
- **givenname**: Nombre.
- **sn**: Apellidos.
- **mail**: Dirección de correo electrónico.

Los datos de un directorio LDAP pueden ser exportados a un fichero en formato LDIF (*LDAP Interchange Format*). LDIF es un formato utilizado para la realización de operaciones por lotes entre directorios según los estándares del protocolo LDAP. LDIF puede utilizarse para exportar e importar datos del directorio y permite llevar a cabo operaciones como agregar, crear o modificar [5].

1.4 Operaciones sobre el directorio LDAP

Una vez analizada la estructura del directorio resulta necesario estudiar las posibles operaciones sobre este. Las operaciones que se realizan sobre el directorio LDAP se basan en el modelo cliente-servidor. Cada cliente LDAP 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:

- **Bind** (autenticación): inicia la sesión de un usuario luego de reconocerlo como válido.
- **Add** (adiciona un elemento al directorio): adiciona los registros del usuario al 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.

Capítulo 1: Fundamentación teórica

- 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.

1.5 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. Actualmente existen diversas herramientas que permiten acceder a la información contenida en los directorios LDAP o implementan algún mecanismo de integración con este. Con la finalidad de aprovechar funcionalidades y/o conceptos que puedan servir 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 muestra el resultado de la investigación realizada.

1.5.1 Ámbito Internacional

Entre las herramientas internacionales se destaca **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. Además, la herramienta permite administrar los registros del servidor, lo que incluye su creación, importación, exportación, modificación y eliminación [7] ([Ver anexo A.1](#)).

Otra de las herramientas existentes es **Webmin**, una herramienta para la administración de servidores Unix accesible desde la web. Permite controlar muchas aplicaciones, tales como el servidor web Apache, MySQL, BIND, Samba, DHCP, OpenLDAP, entre otras. Webmin está escrito en el lenguaje de programación Perl bajo la licencia BSD y es multiplataforma. Brinda la posibilidad de manipular las entradas del directorio (adicionar, eliminar y modificar) [8] ([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 que se pretende realizar. Sin embargo, los autores del presente trabajo consideran que no son herramientas que puedan ser adaptadas a las necesidades de Smart Keeper, debido a que este posee una arquitectura muy peculiar, tema que se abordará con más profundidad en el próximo capítulo.

Capítulo 1: Fundamentación teórica

También se ha analizado **KnowledgeBase Manager Pro**, un *software* que permite automatizar los procesos de gestión del conocimiento. Este software está desarrollado en el lenguaje de programación PHP (*Hypertext Preprocessor*) y liberado bajo la licencia GNU GPL versión 2. Este sistema cuenta con un módulo para la integración con LDAP, a través de una interfaz bastante intuitiva donde se configuran los principales parámetros de la conexión [9] ([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.

Otra herramienta que posee un mecanismo de integración con LDAP es el servidor de correo **SmarterMail**, desarrollado por la compañía SmarterTools. Este servidor de correo permite a sus administradores adicionar nuevos usuarios desde un Directorio Activo, empleando el protocolo LDAP. Esta característica puede ser usada para ahorrar tiempo y es muy útil cuando se desea importar múltiples usuarios de una base de datos LDAP existente [10] ([Ver anexo A.4](#)). Los pasos necesarios para realizar la importación de usuarios en este sistema son:

1. Primeramente se configuran los parámetros de conexión al servidor LDAP.
2. Luego se obtienen los datos de los usuarios y se muestran en un listado.
3. A continuación se seleccionan los usuarios que se desean importar.
4. 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. Sin embargo, los pasos mencionados anteriormente representan una guía para el desarrollo de la solución.

Además de las herramientas antes mencionadas, existen varios *plugins* desarrollados para el *framework* Symfony relacionados con el trabajo con directorios LDAP. A continuación se muestra una descripción de los *plugins* analizados.

slapORM es un ORM (*Object Relational Mapping*) simple para LDAP, escrito en PHP para el *framework* Symfony. Su objetivo principal es permitir realizar consultas LDAP y manipular datos en una capa orientada a objetos. Esto hace que recuperar datos de un LDAP sea tan fácil como al utilizar los ORM *Propel* o *Doctrine* [11]. Este *plugin* para Symfony está en fase de desarrollo y no se ha liberado una

Capítulo 1: Fundamentación teórica

versión estable, por tal motivo se considera que no debe ser empleado en la solución.

dcLdapAbstractionPlugin es un *plugin* para Symfony que provee una abstracción LDAP a través de objetos definidos en un fichero de configuración. Este *plugin* permite utilizar múltiples servidores LDAP y facilita las búsquedas en el directorio a través del uso de criterios de filtrado [12].

Luego de realizar algunas pruebas de rendimiento, se ha apreciado que el tratamiento en forma de objetos que realiza esta herramienta con las entradas del directorio, repercute directamente en la estabilidad del sistema y esto se aprecia en el uso excesivo de memoria. Por tal motivo, los autores de la presente investigación recomiendan evitar el empleo de esta herramienta en aplicaciones donde se manipule gran cantidad de datos, como es el caso de los procesos de importación y sincronización del presente problema.

upSimpleLdapPlugin es un *plugin* para Symfony que brinda una clase para LDAP bastante fácil de utilizar. Fue desarrollado por Leo Cacheux⁶ y liberado bajo la licencia MIT⁷ (*Massachusetts Institute of Technology*). Cuenta con las siguientes características [13]:

- 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

En el caso particular de Cuba, se han encontrado referencias al **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 Universidad de las Ciencias Informáticas. Además de realizar las operaciones básicas para la gestión de un servidor de directorio, también permite un conjunto

⁶ Desarrollador de *plugins* para Symfony de la compañía SensioLabs.

⁷ Instituto tecnológico de Massachusetts: Dedicado a la promoción del conocimiento y la educación de los estudiantes en las áreas que contribuyen a prosperar en un ambiente de ciencia y tecnología.

Capítulo 1: Fundamentación teórica

de operaciones tales como las mencionadas a continuación [6] ([Ver anexo A.5](#)).

- La gestión de imágenes.
- La replicación de la información.
- La migración de datos de un Directorio Activo de Windows al servidor OpenLDAP.
- La gestión de esquemas.
- La gestión de configuraciones del servidor.
- La certificación del servidor.
- 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 externa como la de Smart Keeper. Además, el hecho de ser una aplicación de escritorio, dificulta su integración con la Interfaz de Administración Web de Smart Keeper (en lo adelante IAW). 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 Tecnologías y herramientas

La selección de las tecnologías a utilizar en el desarrollo de un *software* es un paso muy importante ya que influye directamente en la calidad del producto final y en el esfuerzo necesario para obtenerlo. La IAW de Smart Keeper fue creada usando el *framework* Symfony y el lenguaje de programación PHP. Dado que esta interfaz se ha convertido en una poderosa herramienta para los administradores, se desea que el módulo desarrollado se pueda integrar a esta, lo cual se ha tenido en cuenta a la hora de definir las tecnologías a emplear.

1.6.1 Lenguaje de programación

Actualmente existe una gran variedad de lenguajes de programación para desarrollar en la web tales como: Perl, Java, PHP, Ruby y Python, los cuales permiten interactuar con los usuarios y utilizar sistemas de bases de datos. Entre estos se destaca **PHP**, un lenguaje de código abierto, interpretado en el lado del servidor, utilizado para la generación de páginas web dinámicas y que puede ser embebido en páginas HTML (*HyperText Markup Language*). A continuación se muestran las ventajas y desventajas de emplear dicho lenguaje:

Ventajas [14]:

- Velocidad: está escrito en C, por lo que se ejecuta rápidamente utilizando poca memoria.
- Estabilidad: utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- Simplicidad: los usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente debido a que la sintaxis es similar.
- Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix y PostgreSQL.
- Lenguaje de código abierto, lo que posibilita la actualización y corrección de problemas por la amplia comunidad con que cuenta.
- Multiplataforma, permite ser ejecutado bajo varias versiones de Unix, Windows y Macs.

Desventajas [15]:

- Todo el trabajo lo realiza el servidor ya que el código PHP se ejecuta en este. Por tanto, puede ser más ineficiente a medida que las solicitudes aumentan.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.

PHP cuenta además con soporte para LDAP desde su versión 3.0. Estas capacidades hacen que PHP sea muy popular entre los desarrolladores que necesitan crear aplicaciones web que interactúan con directorios LDAP [16]. Además, emplearlo facilitará la integración de la solución con la IAW. Luego de analizar estas características se ha decidido emplear dicho lenguaje, en su versión 5.3.

1.6.2 Sistema Gestor de Base de Datos

La persistencia de los datos es un aspecto clave en cualquier sistema de gestión de la información. **PostgreSQL** es un sistema gestor de bases de datos relacional de código abierto y es el que utiliza Smart Keeper para almacenar los datos. Al usarlo se garantiza que el módulo pueda integrarse a la Base de datos del sistema, de forma que se evite utilizar una base de datos adicional. La selección de dicho sistema está sustentada además por sus características, las cuales se mencionan a continuación.

Características de PostgreSQL [17]:

- Puede ejecutarse sobre plataformas Linux, UNIX y Windows.
- Tiene interfaces nativas de programación para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC y otros.
- Cuenta con una documentación bastante amplia.
- Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.
- Cuenta con numerosos tipos de datos y la posibilidad de definir nuevos.
- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido).

1.6.3 Framework

La utilización de un *framework* puede facilitar y agilizar el desarrollo de un sistema. En el desarrollo de la IAW de Smart Keeper se emplea **Symfony**, un *framework* para PHP con gran aceptación entre los desarrolladores de aplicaciones web. Este *framework* hace un uso bastante eficiente de varios patrones de diseño. Elegir Symfony (versión 1.2) para el desarrollo de la solución facilitará el proceso de integración del módulo a desarrollar con la IAW, además permitirá mantener cierta homogeneidad con los demás componentes del sistema. A continuación se relacionan algunas características de Symfony que refuerzan su elección.

Características de Symfony [18]:

- Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona

Capítulo 1: Fundamentación teórica

correctamente en los sistemas Windows y Unix estándares).

- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además es lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.6.4 Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) es una herramienta informática que facilita el trabajo a los programadores, brindándoles un conjunto de herramientas de programación. **Netbeans** es un IDE gratuito y de código abierto, empleado en el desarrollo de aplicaciones web, de escritorio y para móviles. Se encuentran disponibles versiones de Netbeans para los sistemas operativos Windows, Linux, Mac OS X y Solaris [19].

Se recomienda utilizar la versión 7.0 del IDE. La base en la que se sustenta su elección es que permite desarrollar aplicaciones utilizando el *framework* Symfony y ejecutar los comandos de este directamente desde la interfaz del IDE. Por otra parte se posee gran dominio y experiencia en el uso de este IDE para el desarrollo de aplicaciones web en PHP.

1.6.5 Herramienta CASE

Visual Paradigm para UML es una herramienta de diseño multiplataforma diseñada para asistir el desarrollo de *software*. Soporta los principales estándares de la industria, tales como; UML, SysML, BPMN, XMI. Ofrece un conjunto completo de herramientas que los equipos de desarrollo necesitan para la captura de requisitos, planificación de *software*, planificación de pruebas, modelamiento de clases, modelamiento de datos y otras actividades [20]. Todas estas características unidas a la experiencia que se

posee en la utilización de esta herramienta apoyaron su elección. La versión a utilizar es *Visual Paradigm for UML 8.0*.

1.7 Metodología de desarrollo de software

El empleo de una metodología de desarrollo de *software* es un aspecto muy importante, ya que indica el camino a seguir para obtener un producto con calidad. Existen dos grandes enfoques, las metodologías tradicionales y las metodologías ágiles. Las metodologías tradicionales están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto. Algunos ejemplos de metodologías tradicionales son: RUP (*Rational Unified Process*) y MSF (*Microsoft Solution Framework*). En cambio, las metodologías ágiles centran su atención en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y a mantener una buena relación con el cliente. Ejemplos de metodologías ágiles son: XP (*eXtreme Programming*), AUP (*Agil Unified Process*), Scrum e Iconix [21].

RUP es precisamente la metodología empleada en Smart Keeper para la gestión de sus procesos. Por otra parte, se puede considerar que el equipo de desarrollo del proyecto Smart Keeper es muy inestable y varía mucho a causa de estar formado fundamentalmente por estudiantes. Lo anterior constituye un motivo adicional para emplear RUP debido a que este genera una gran cantidad de artefactos.

Los artefactos generados permiten tener una amplia y detallada documentación sobre el proceso de desarrollo de *software*, de modo que los nuevos miembros del proyecto puedan conocer en detalle lo que se ha desarrollado. Además, RUP brinda una guía para la producción de *software* de alta calidad, que satisfaga las necesidades de los clientes. A continuación se relacionan algunas de las características principales de esta metodología.

Características de RUP [22]:

- Es orientado a objetos.
- Utiliza **UML** como lenguaje de representación visual.
- Dirigido por Casos de Uso: Los Casos de Uso son el hilo conductor que orientan las actividades de desarrollo y son usados para guiar el flujo de procesos desde la definición de requisitos hasta las pruebas. Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo, dispositivo) que interactúa con él.

Capítulo 1: Fundamentación teórica

- Centrado en la arquitectura: Propone la definición de una arquitectura robusta, lo que facilita el desarrollo del sistema en paralelo, aumentando las posibilidades de reutilización de componentes y el mantenimiento del sistema. Proporciona las guías para configurar el proceso de modo que se adapte a cada situación.
- Iterativo e Incremental: Propone la descomposición de proyectos grandes en mini-proyectos, cada mini-proyecto es una iteración, y cada iteración debe estar controlada y tratar un determinado grupo de Casos de Usos.

1.8 Conclusiones parciales

El estudio del estado del arte ha permitido identificar algunas características a incluir en la solución al problema planteado. Sin embargo, no se ha encontrado una herramienta que pueda ser utilizada íntegramente en dicha solución, lo cual evidencia la necesidad de crear una herramienta propia. El empleo del lenguaje de programación PHP y el *framework* Symfony permitirá una rápida integración del módulo con la IAW de Smart Keeper. La metodología RUP y las herramientas seleccionadas facilitarán el desarrollo de un producto de calidad.

Capítulo 2

Capítulo 2: Características del sistema

El filtro de contenido Smart Keeper está orientado a regular y monitorear el acceso a Internet a través del análisis de las peticiones de los usuarios y del tratamiento de la respuesta. Entre las funcionalidades de este sistema se encuentra la gestión de políticas de navegación a nivel de usuario o de grupo. Además, posee capacidades de filtrado adicionales basadas en programación de tiempo, control de subredes de acceso, expresiones regulares y excepciones que pueden ser aplicadas a nivel global, de política y de usuario. La base de la gestión de estas características gira alrededor de los usuarios del sistema, de ahí la importancia de estos.

2.1 Flujo actual de los procesos

Para poder comprender mejor el problema planteado se hace necesario describir el flujo actual de los procesos. El filtro de contenido Smart Keeper cuenta con un mecanismo de gestión de usuarios interno e independiente, basado en la utilización del *plugin sfGuardPlugin*⁸. Este mecanismo sienta las bases para toda la administración de los permisos de navegación de cada usuario, permitiendo aplicar a cada uno cierta política, programación de tiempo u otros mecanismos de control.

En el sistema existen dos tipos de usuarios: los usuarios estándar y los usuarios subred. Los usuarios estándar son el personal de la institución, mientras que los usuarios subred se emplean en caso de que se desee controlar la navegación por dirección IP o subred desde la que se accede al servicio de Internet. Toda la información de estos usuarios se encuentra almacenada en la base de datos de Smart Keeper, alojada en un servidor PostgreSQL, de esta forma para habilitar la navegación a cierto usuario este debe primeramente estar registrado en el sistema, ya sea como usuario estándar o como subred.

Actualmente el proceso de creación de cuentas de usuarios en Smart Keeper se realiza de forma manual, a través de un formulario en la IAW donde se introducen los datos del usuario y se establecen algunos parámetros adicionales (*Ver figura 2.1*).

⁸ *Plugin* para Symfony que permite incluir la gestión de usuarios y grupos a una aplicación.

Nuevo Usuario

Perfil	Navegación	Grupos y permisos
Tipo de usuario	<input checked="" type="radio"/> Estándar <input type="radio"/> Subred	
Usuario	<input type="text"/>	
Nombre	<input type="text"/>	
Apellido(s)	<input type="text"/>	
Correo	<input type="text"/>	
Contraseña	<input type="text"/>	
Repita la contraseña	<input type="text"/>	
Idioma	español ▲▼	
← Atrás <input type="button" value="Guardar"/> <input type="button" value="Guardar y crear otro"/>		

Figura 2.1: Formulario para la creación de cuentas de usuario en Smart Keeper.

Es importante destacar que Smart Keeper ya cuenta con la posibilidad de habilitar la autenticación a través de un servidor LDAP mediante el mecanismo de autenticación squid_ldap_auth⁹, pero este mecanismo tiene cierta dificultad. El problema radica en que si un usuario estándar no está registrado en la base de datos de Smart Keeper y no existe una configuración para la subred en la que se encuentra navegando, entonces se deniega la navegación, aunque sus datos de autenticación contra LDAP sean correctos (Ver figura 2.2). Se evidencia de esta forma la necesidad de crear el usuario de forma local en el sistema.

⁹ Mecanismo de autenticación de Smart Keeper mediante LDAP (tomado del manual de usuario de Smart Keeper).

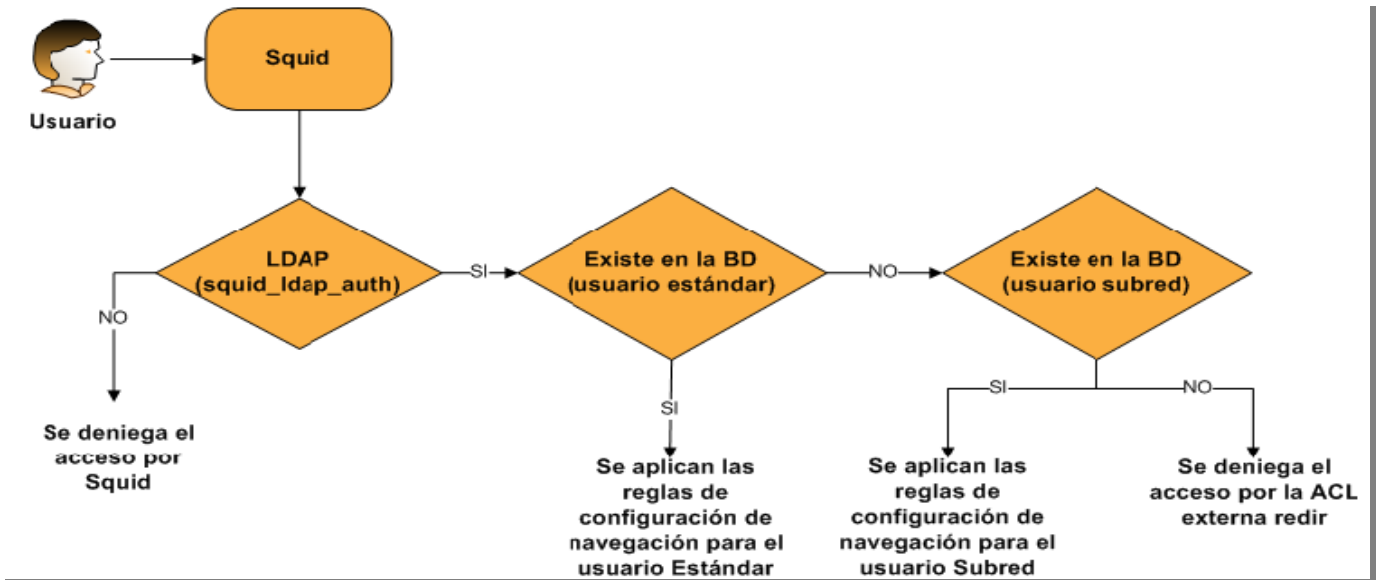


Figura 2.2: Esquema de autenticación de Smart Keeper a través de squid_ldap_auth.

2.2 Propuesta de solución

Para concretar la propuesta de solución se considera importante adentrarse un poco más en la arquitectura del sistema Smart Keeper. La IAW de Smart Keeper está desarrollada utilizando el *framework* Symfony y el lenguaje de programación PHP, como se ha explicado anteriormente. Además, el sistema posee una arquitectura basada en *plugin*, característica novedosa que se ha ido incorporando en el sistema y que ha tenido gran aceptación por el equipo de desarrollo. Por otra parte la base de datos de Smart Keeper se encuentra alojada en un servidor PostgreSQL.

Se propone desarrollar un módulo para la IAW de Smart Keeper en forma de *plugin*. Dicho módulo debe permitir importar los usuarios de un servidor LDAP hacia la base de datos de Smart Keeper. De esta forma se automatiza el proceso de creación de cuentas de usuarios. Por otro lado, el sistema debe permitir realizar el proceso de sincronización, con el objetivo de mantener actualizados los datos de los usuarios de Smart Keeper con respecto a los nuevos cambios que puedan ocurrir en el servidor LDAP.

Debido a que el proceso de sincronización y la importación de un gran número de usuarios son tareas que requieren un tiempo considerable de procesamiento, los autores de la presente investigación consideran necesaria la implementación de dichos mecanismos en forma de tareas de Symfony.

Capítulo 2: Características del sistema

Las tareas de Symfony permiten ejecutar cierto código de forma independiente a la interfaz administrativa de un sistema, o sea, desde la línea de comandos. Con esto se logra evitar que la aplicación supere el tiempo límite de los navegadores en espera de una respuesta del servidor. Además, el sistema debe almacenar el resultado de la última consulta realizada al servidor LDAP en una tabla temporal creada a tal efecto, esto permite reducir el número de consultas realizadas al servidor.

El módulo a desarrollar se nombrará **LdapPlugin** y estará compuesto por 2 sub-módulos:

- **ldap_user**: Permitirá listar los usuarios del LDAP para realizar el proceso de importación a un usuario específico, a varios usuarios seleccionados o a todos los usuarios encontrados dado un criterio de búsqueda.
- **ldap_config**: Permitirá gestionar los parámetros de configuración para los distintos servidores LDAP existentes en la institución. Esto incluye la configuración de la conexión y el comportamiento del proceso de sincronización para cada servidor existente.

Al utilizar Symfony y PHP en el desarrollo del módulo y además en forma de *plugin*, se facilita su integración con la IAW.

2.3 Modelado del dominio

El modelo de dominio es una representación visual de los objetos del negocio donde se identifican elementos y eventos que suceden en dicho ambiente. Dicho modelo permite describir el estado del problema planteado, así como obtener una vista estática vinculada con las reglas del negocio (*Ver figura 2.3*).

2.3.1 Diagrama de clases del Modelo de Dominio

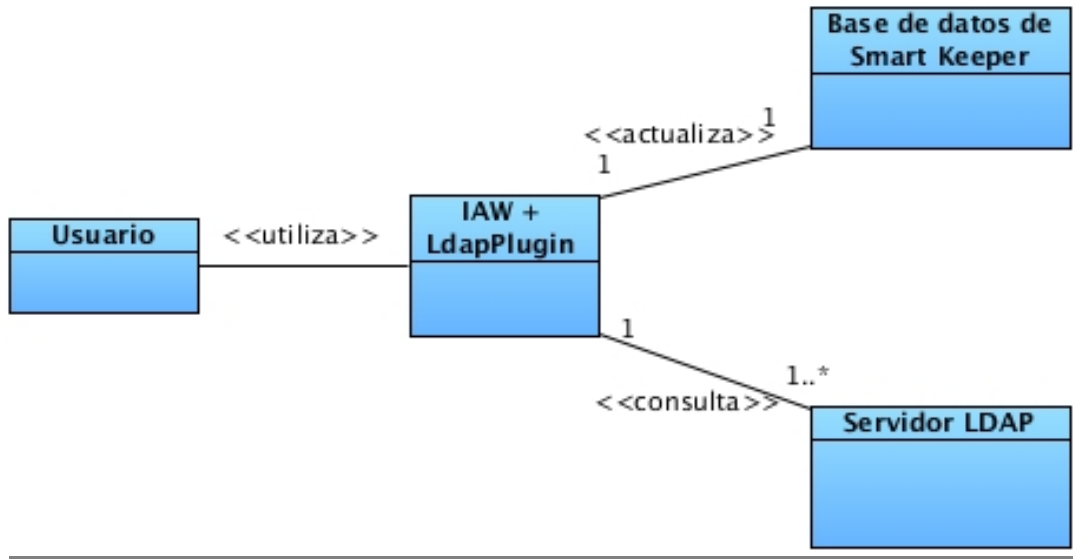


Figura 2.3: Diagrama del Modelo de dominio.

2.3.2 Definición de las Clases del Modelo de Dominio

Usuario: Persona que utiliza el módulo propuesto, juega el rol de actor e inicializa los Casos de Uso del Sistema.

IAW + LdapPlugin: Módulo propuesto a desarrollar, integrado a la IAW de Smart Keeper.

Base de datos de Smart Keeper: Base de datos propia de Smart Keeper, en la que se encuentra la información 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 claros y definidos permite comprender desde un inicio la línea a seguir en el desarrollo y así garantizar la eficiencia y calidad del *software*.

Capítulo 2: Características del sistema

2.4.1 Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que un sistema determinado debe cumplir, estas capacidades dan la medida de la utilidad del sistema. A continuación se describen los requerimientos funcionales identificados.

Nº	Nombre	Descripción
[RF1.]	Adicionar configuración de conexión	El sistema debe permitir adicionar los parámetros de conexión que se usarán para contactar con un servidor LDAP específico, así como los parámetros para la sincronización.
[RF2.]	Listar configuración de conexión	El sistema debe mostrar un listado de todas las configuraciones de conexión existentes.
[RF3.]	Editar configuración de conexión	El sistema debe permitir modificar los datos de una configuración de conexión a un servidor LDAP y los parámetros para la sincronización.
[RF4.]	Eliminar configuración de conexión	El sistema debe permitir eliminar una configuración de conexión.
[RF5.]	Listar usuarios del LDAP	El sistema debe mostrar el listado de los usuarios del servidor LDAP que se obtuvieron en la última consulta.
[RF6.]	Actualizar listado de usuarios del LDAP	El sistema debe permitir actualizar el listado de usuarios de acuerdo a los existentes en el servidor LDAP activo.
[RF7.]	Importar un usuario del LDAP	El sistema debe permitir importar un usuario del LDAP.
[RF8.]	Importar usuarios filtrados	El sistema debe permitir importar los usuarios obtenidos como resultado del filtrado.
[RF9.]	Importar usuarios seleccionados	El sistema debe permitir importar todos los usuarios que hayan sido seleccionados.
[RF10.]	Sincronizar	El sistema debe permitir realizar el proceso de sincronización teniendo en cuenta los parámetros de conexión y la configuración de sincronización establecida para el servidor activo.

Tabla 2.1: Requerimientos funcionales.

Capítulo 2: Características del sistema

2.4.2 Requerimientos no funcionales

Los requerimientos no funcionales detallan las propiedades o cualidades que el producto debe tener y hacen al producto atractivo, fácil de usar, rápido y confiable. Como parte de la captura de requerimientos no funcionales se determinaron los siguientes requisitos.

Usabilidad:

- El sistema debe proporcionar un acceso fácil, rápido e intuitivo al usuario, a través de una interfaz web integrada a la IAW de Smart Keeper.

Software:

- La PC cliente debe contar con un navegador web Mozilla Firefox versión 5 o superior.

Hardware:

- El servidor web debe contar con al menos 1 Gb de memoria RAM.
- El servidor web debe poseer un procesador Intel o similar, a 3.0 GHz o superior.

Seguridad:

- El acceso al módulo se realizará mediante una conexión segura por HTTPS (*Hypertext Transfer Protocol Secure*) y la autenticación de los usuarios mediante los mecanismos de autenticación de Smart Keeper.

Confiabilidad:

- El sistema debe notificar al usuario sobre cualquier error o excepción que ocurra durante la ejecución de las acciones solicitadas.

Eficiencia:

- El sistema debe permitir la ejecución de solo una tarea de importación o sincronización simultáneas, para que el rendimiento del sistema no se vea afectado.

Restricciones de diseño:

- El sistema se desarrollará utilizando el lenguaje de programación PHP 5.3 con el módulo para LDAP.

Capítulo 2: Características del sistema

- Se debe emplear el *framework* Symfony.
- El servidor web deberá ser Apache 2.
- Se utilizará el Entorno de Desarrollo Integrado Netbeans 7.
- Se usará Visual Paradigm for UML 8.0 para la modelación del sistema.
- El proceso de desarrollo estará guiado por la metodología de desarrollo de *software* RUP.
- El lenguaje de modelado será UML.

Interfaz:

- El módulo debe poseer los colores de Smart Keeper para así lograr una vista común.
- Los mensajes de error deben mostrarse en el mismo formato que en el resto de los módulos de la interfaz.

Requisitos de Licencia:

- La licencia del módulo será GNU GPL v2.

Requisitos Legales, de Derecho de Autor y otros:

- Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres, bajo las condiciones de *software* libre. Para la herramienta Visual Paradigm se debe emplear la licencia que posee la universidad.

2.4.3 Diagrama de Casos de Uso del Sistema

Los Casos de Uso son una técnica para capturar y especificar los requisitos del sistema, así como para guiar su diseño, implementación y prueba. Cada Caso de Uso facilita un escenario sin ambigüedad en la interacción entre el actor y el *software*, así como describe un conjunto de funcionalidades que el sistema desarrollado debe poseer [23].

A continuación se muestra el diagrama de Casos de Uso del Sistema (CUS) y la descripción del CUS Importar usuarios.

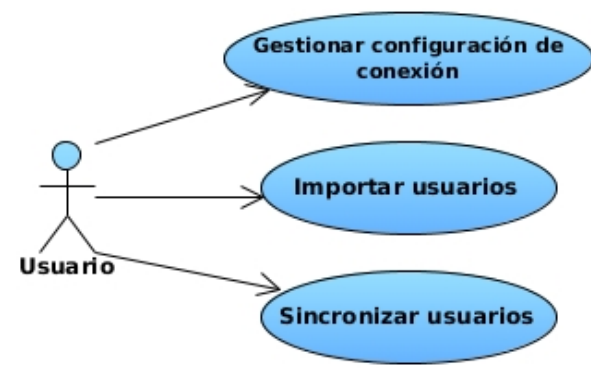


Figura 2.4: Diagrama de Casos de Uso del Sistema.

2.4.4 Descripción de Casos de Uso del Sistema

Descripción textual del Caso de Uso Importar usuarios.

Objetivo	Actualizar listado de usuarios del LDAP de modo que se puedan importar a la base de datos del sistema Smart Keeper.
Actores	Usuario: (Inicia) Lista y actualiza el listado de usuarios obtenido del LDAP y realiza la importación de usuarios hacia el sistema.
Resumen	El CU se inicia cuando el usuario decide realizar una de las siguientes acciones: <ul style="list-style-type: none"> - Listar usuarios del LDAP - Actualizar listado de usuarios del LDAP - Importar un usuario - Importar usuarios seleccionados - Importar usuarios filtrados
Complejidad	Alta
Prioridad	Crítica
Precondiciones	El usuario ha iniciado sesión en el sistema y cuenta con los privilegios necesarios para interactuar con el módulo. Debe existir una configuración de conexión activa.
Flujo de eventos	
Flujo básico “Importar usuarios”	

Actor		Sistema
1.	Selecciona la opción “Usuarios LDAP”.	
2.		Muestra el listado de los usuarios existentes en el LDAP según los datos de la última consulta realizada.
3.		Permite realizar varias acciones: <ul style="list-style-type: none"> • Actualizar listado de usuarios del LDAP (Ver Sección 1: “Actualizar listado de usuarios del LDAP”). • Importar un usuario (Ver Sección 2: “Importar un usuario”). • Importar usuarios filtrados (Ver Sección 3: “Importar usuarios filtrados”). • Importar usuarios seleccionados (Ver Sección 4: “Importar usuarios seleccionados”).
4.		Termina el Caso de Uso.

Sección 1: “Actualizar listado de usuarios del LDAP”

Flujo básico “Actualizar listado de usuarios del LDAP”

Actor		Sistema
1.	Selecciona la opción “Actualizar listado de usuarios”.	
2.		Borra el listado actual de usuarios.
3.		Determina la configuración de la conexión que va a emplear para obtener los datos del LDAP.
4.		Contacta al servidor LDAP con la configuración de la conexión activa y realiza la consulta obteniendo los datos de los usuarios.
6.		Muestra notificación con el resultado del proceso de actualización.

7.		Termina el Caso de Uso.
Flujos alternos		
4a. No se pudo contactar al servidor LDAP		
Actor		Sistema
1.		Muestra un mensaje indicando los errores existentes.
2.		Termina el Caso de Uso.
Sección 2: “Importar un usuario”		
Flujo básico “Importar un usuario”		
	Actor	Sistema
1.	Selecciona la opción “Importar” de un usuario específico.	
2.		Solicita la siguiente información mediante un cuadro de diálogo: <ol style="list-style-type: none"> 1. Cuota límite. 2. Política de navegación. 3. Idioma de la interfaz. 4. Marcar como activo. 5. Habilitar navegación.
3.	Completa los datos solicitados.	
4.		Valida los datos introducidos.
5.		Realiza el proceso de importación.
6.		Muestra un mensaje indicando el resultado de la operación.
7.		Termina el Caso de Uso.
Flujos alternos		
5ª. Existen errores en los datos		
Actor		Sistema
1.		Muestra un mensaje indicando los errores

		existentes.
2.		Termina el Caso de Uso.
Sección 3: “Importar usuarios filtrados”		
Flujo básico “Importar usuarios filtrados”		
	Actor	Sistema
1.	Introduce valores en los campos del formulario de filtrado y presiona “Filtrar”.	
2.		Actualiza el listado de usuarios basándose en los criterios de filtrado especificados por el usuario.
3.	Selecciona la opción “Importar usuarios”.	
4.		Solicita la siguiente información mediante un cuadro de diálogo: <ol style="list-style-type: none"> 1. Cuota límite. 2. Política de navegación. 3. Idioma de la interfaz. 4. Marcar como activo. 5. Habilitar navegación.
5.	Completa los datos solicitados.	
6.		Valida los datos introducidos.
7.		Realiza el proceso de importación de los usuarios obtenidos como resultado de aplicar ciertos criterios de búsqueda.
8.		Muestra notificación con resultados del proceso de importación.
9.		Termina el Caso de Uso.
Flujos alternos		
6ª. Existen errores en los datos		
	Actor	Sistema
1.		Muestra un mensaje indicando los errores

		existentes.
Sección 4: “Importar usuarios seleccionados”		
Flujo básico “Importar usuarios seleccionados”		
	Actor	Sistema
1.	Selecciona los usuarios a importar y hace clic en la opción Importar de la lista de acciones por lotes.	
2.		Solicita la siguiente información mediante un cuadro de diálogo: <ol style="list-style-type: none"> 1. Cuota límite. 2. Política de navegación. 3. Idioma de la interfaz. 4. Marcar como activo. 5. Habilitar navegación.
3.	Completa los datos solicitados.	
4.		Valida los datos introducidos.
5.		Realiza el proceso de importación para cada usuario seleccionado.
6.		Muestra un mensaje indicando el resultado de la operación.
7.		Termina el Caso de Uso.
Flujos alternos		
4ª. Existen errores en los datos		
	Actor	Sistema
1.		Muestra un mensaje indicando los errores existentes.

Tabla 2.2: Descripción textual del Caso de Uso Importar usuarios.

Nota: Las descripciones de los Casos de Usos restantes pueden ser consultadas en el anexo [B](#).

2.5 Conclusiones parciales

El análisis del flujo actual de los procesos ha permitido ganar en comprensión sobre el problema presente. La solución propuesta permitió establecer una línea por la cual guiarse durante el desarrollo del *plugin*. A través de la captura de requerimientos se estableció qué debe hacer exactamente el sistema, de modo que se pueda determinar posteriormente si cumple o no con los requisitos establecidos. Las especificaciones de Casos de Uso del Sistema permitieron detallar las interacciones entre el actor y el sistema. Al realizar todas las especificaciones y diagramas descritos en el presente capítulo han quedado sentadas las bases para comenzar el diseño del módulo propuesto.

Capítulo 3

Capítulo 3: Análisis y diseño del sistema

Durante el análisis se estudian los requisitos que fueron identificados con el fin de refinarlos y estructurarlos. El objetivo del análisis es conseguir una comprensión más precisa de los requisitos y una descripción de estos, de forma que ayude a estructurar el sistema, incluyendo su arquitectura [22]. De esta manera se garantiza la obtención de una arquitectura robusta, eficaz, eficiente y capaz de sobrevivir a cambios. El resultado final de este flujo es el modelo del análisis que constituye la entrada principal en el diseño.

Por otro lado, el diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Durante este flujo se modela el sistema y se encuentra su forma para que soporte todos los requisitos, incluyendo los no funcionales [22]. El resultado final más importante de este flujo de trabajo será el modelo de diseño.

3.1 Estilos arquitectónicos y patrones de diseño

Un patrón de diseño es la base para la búsqueda de soluciones a problemas comunes en el desarrollo de un *software*. Entre los patrones utilizados en la solución propuesta se encuentran los patrones de arquitectura y los patrones de diseño. Los patrones de arquitectura son aquellos que expresan un esquema estructural para sistemas de *software*. Los patrones de diseño expresan un esquema para definir las estructuras de diseño con las que se va a construir el *software* [24].

Symfony está basado en un patrón arquitectónico muy clásico en el diseño web, conocido como **arquitectura MVC** (Modelo - Vista - Controlador), que está formado por tres niveles:

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.

- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

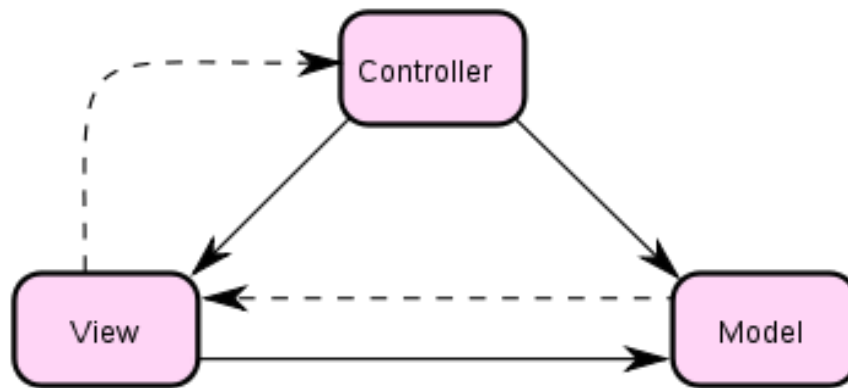


Figura 3.1: Arquitectura Modelo Vista Controlador.

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador.

Symfony toma lo mejor de esta arquitectura y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. Hace una separación de las capas más allá del MVC, por ejemplo, la capa del modelo se divide en la capa de acceso a los datos y en la capa de abstracción de la base de datos. Las clases de la capa del modelo se generan automáticamente, en función de la estructura de datos de la aplicación. Esto se logra gracias a la librería *Propel*, que se encarga de esta generación automática.

Además, *Propel* genera el esqueleto o estructura básica de las clases y el código necesario. Esto permite que si se cambia de sistema gestor de bases de datos, solamente sea necesario cambiar un parámetro en un archivo de configuración y no haya que reescribir ni una línea de código [18].

Implementación del MVC que realiza Symfony:

- La capa del Modelo
 - Abstracción de la base de datos
 - Acceso a los datos

Capítulo 3: Análisis y diseño del sistema

- La capa de la Vista
 - Vista
 - Plantilla
 - Layout
- La capa del Controlador
 - Controlador frontal
 - Acción

La arquitectura MVC proporciona grandes ventajas, como la organización, la reutilización y la flexibilidad del código. Crear la aplicación con Symfony permite generar páginas XHTML (*Extensible HyperText Markup Language*) válidas, depurar fácilmente las aplicaciones, contar con una configuración sencilla, realizar una abstracción de la base de datos, utilizar un enrutamiento con URL limpias, emplear varios entornos de desarrollo y muchas otras utilidades para el desarrollo de aplicaciones.

Por otro lado Symfony implementa muchos patrones de diseño y aprovecha buenas prácticas en el desarrollo web. Como parte de la implementación del MVC separa la vista en un *layout* y en una plantilla. Normalmente, el *layout* es global en toda la aplicación y contiene el código HTML que es común en la mayoría de las páginas. La plantilla sólo se encarga de visualizar las variables definidas en el controlador, esto se puede ver como que el *layout* decora la plantilla, lo cual es una implementación del patrón **Decorator**. El controlador frontal es un componente que sólo tiene código relativo al MVC y Symfony también lo genera de forma automática [18].

El patrón **Singleton** es usado en Symfony para permitir el acceso desde cualquier lugar de la aplicación a los objetos relacionados con el núcleo del *framework*. Esto ocurre a través de la llamada a la función `sfContext::getInstance()`, desde cualquier parte del código. Las clases que conforman el núcleo están basadas en el patrón **Factory Method** permitiendo la creación fácil de los objetos y su extensión de forma sencilla.

3.2 Diagramas de clases del análisis

Los diagramas de clases del análisis constituyen una vista de la futura composición de las clases del *software*. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama y se modifica para satisfacer los detalles de las implementaciones [25]. A continuación se muestran los diagramas de clases del análisis de la solución propuesta:

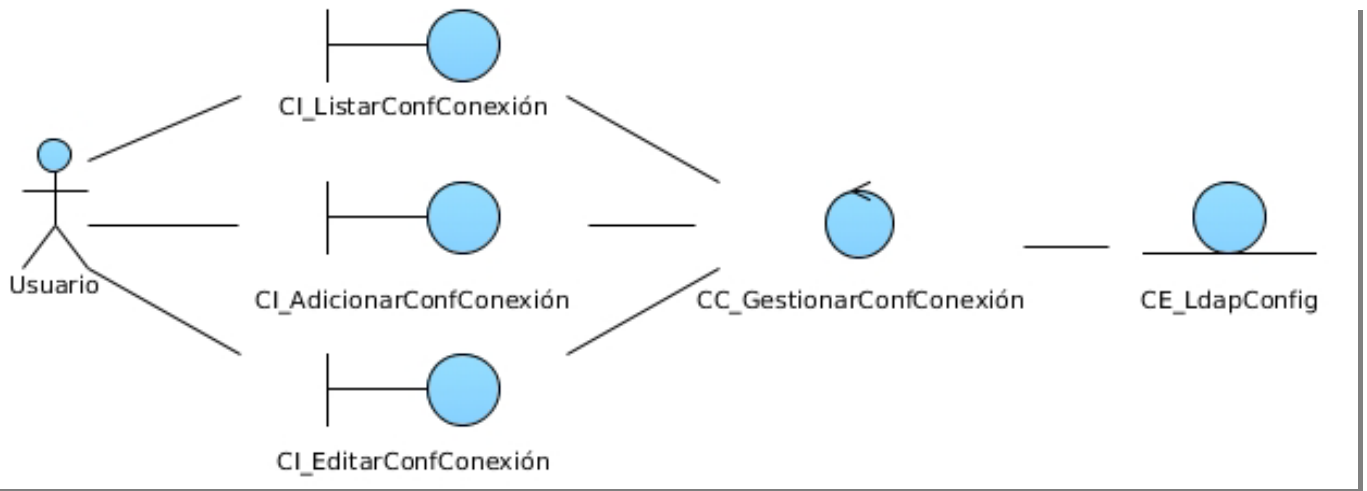


Figura 3.2: Diagrama de clases del análisis. CU Gestionar configuración de conexión.

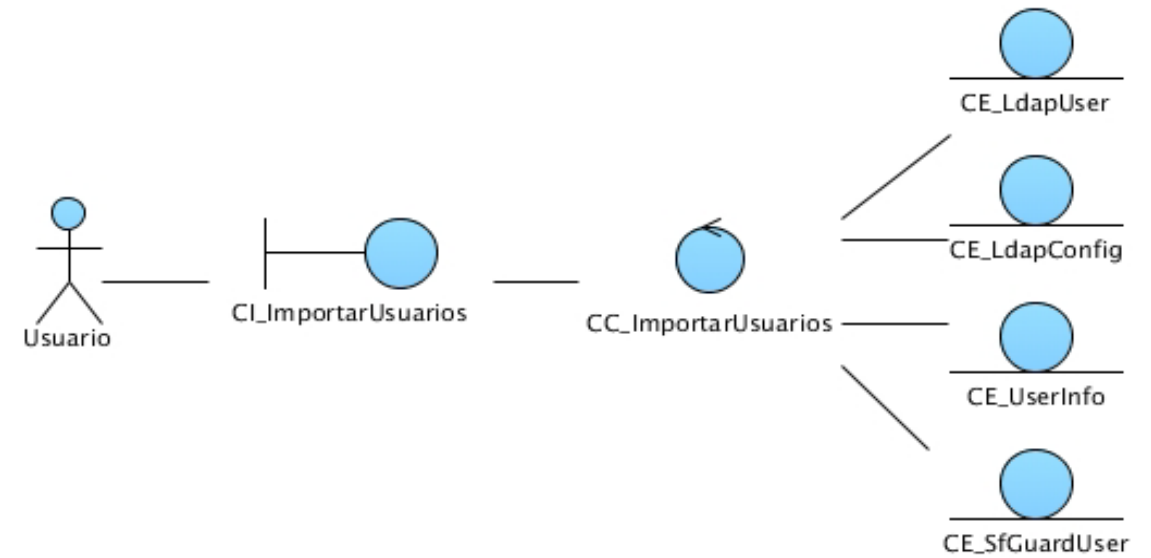


Figura 3.3: Diagrama de clases del análisis. CU Importar usuarios.

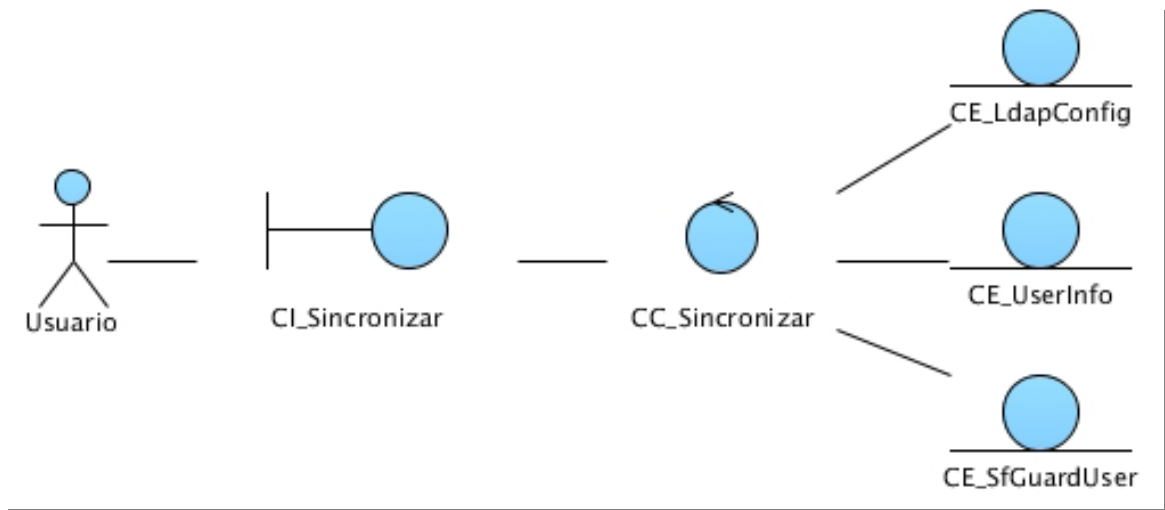


Figura 3.4: Diagrama de clases del análisis. CU Sincronizar usuarios.

3.3 Diagramas de clases del diseño

Los diagramas de clase del diseño se usan para describir la estructura de un sistema y formalizar el conocimiento del dominio de la aplicación. Las clases son abstracciones que especifican los atributos y comportamientos de un conjunto de objetos [26]. Los siguientes diagramas muestran las clases del diseño del Caso de Uso Importar usuarios.

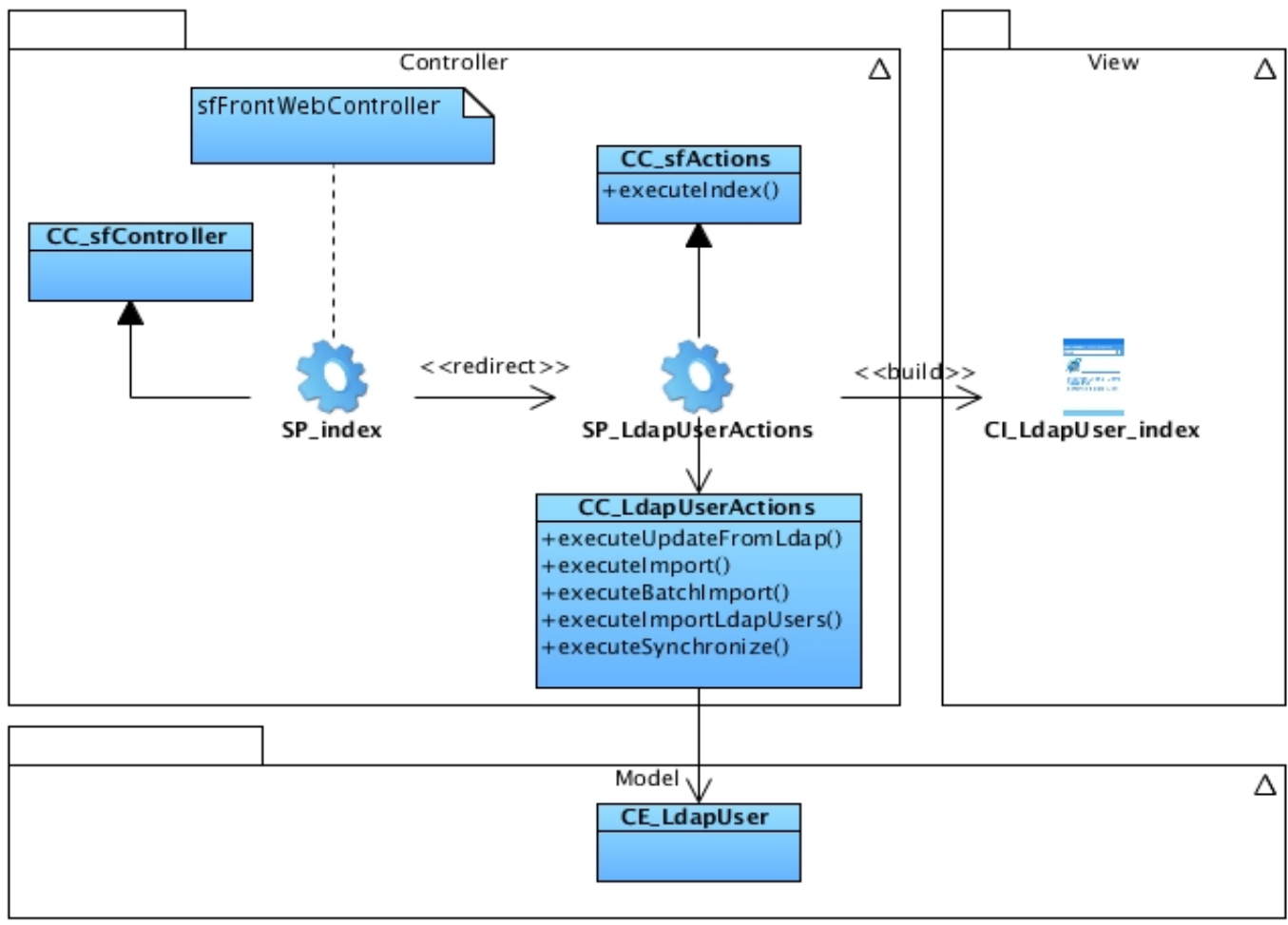


Figura 3.5: Diagrama de clases del diseño. CU Importar usuarios. Sección Listar usuarios del LDAP.

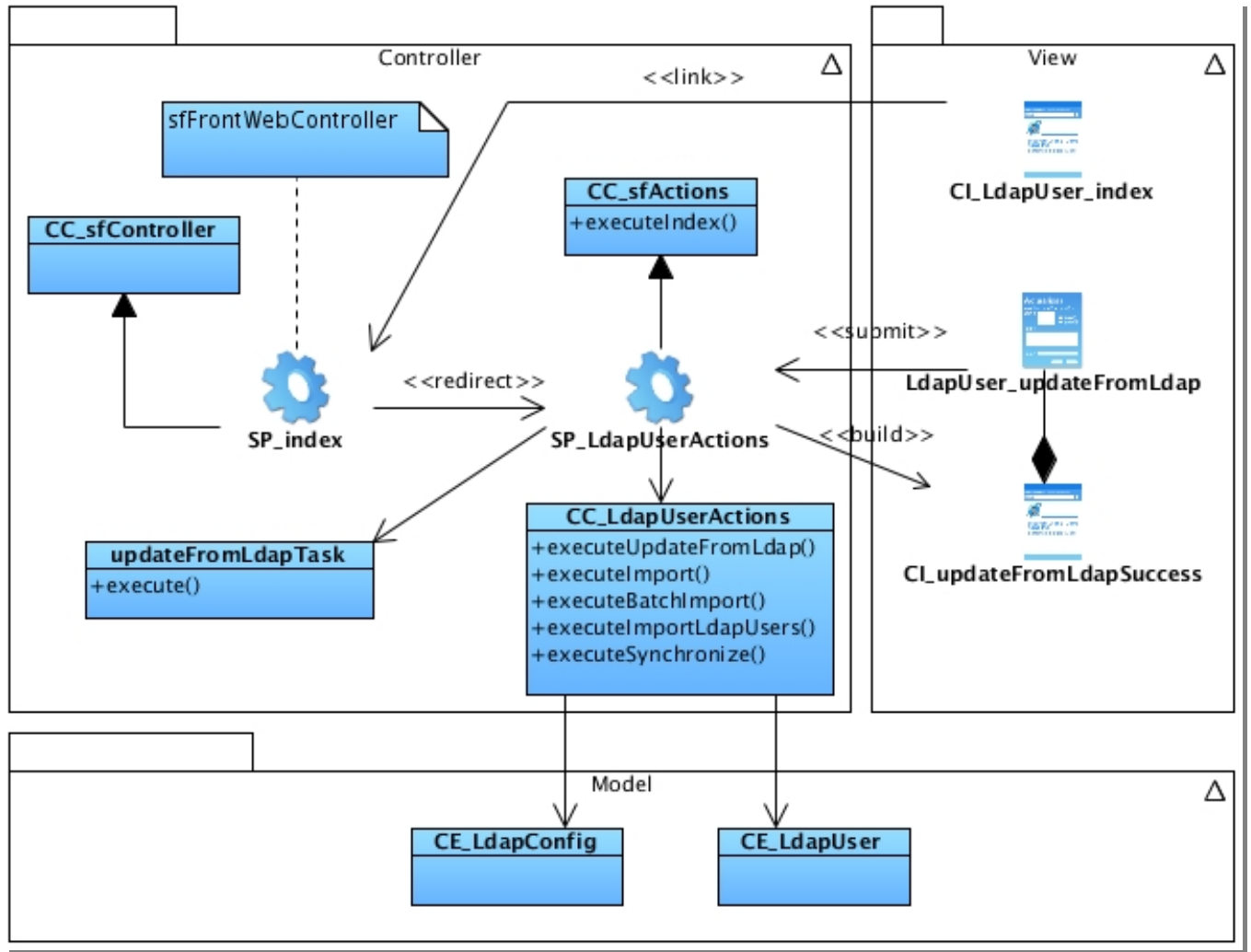


Figura 3.6: Diagrama de clases del diseño. CU Importar usuarios. Sección Actualizar listado.

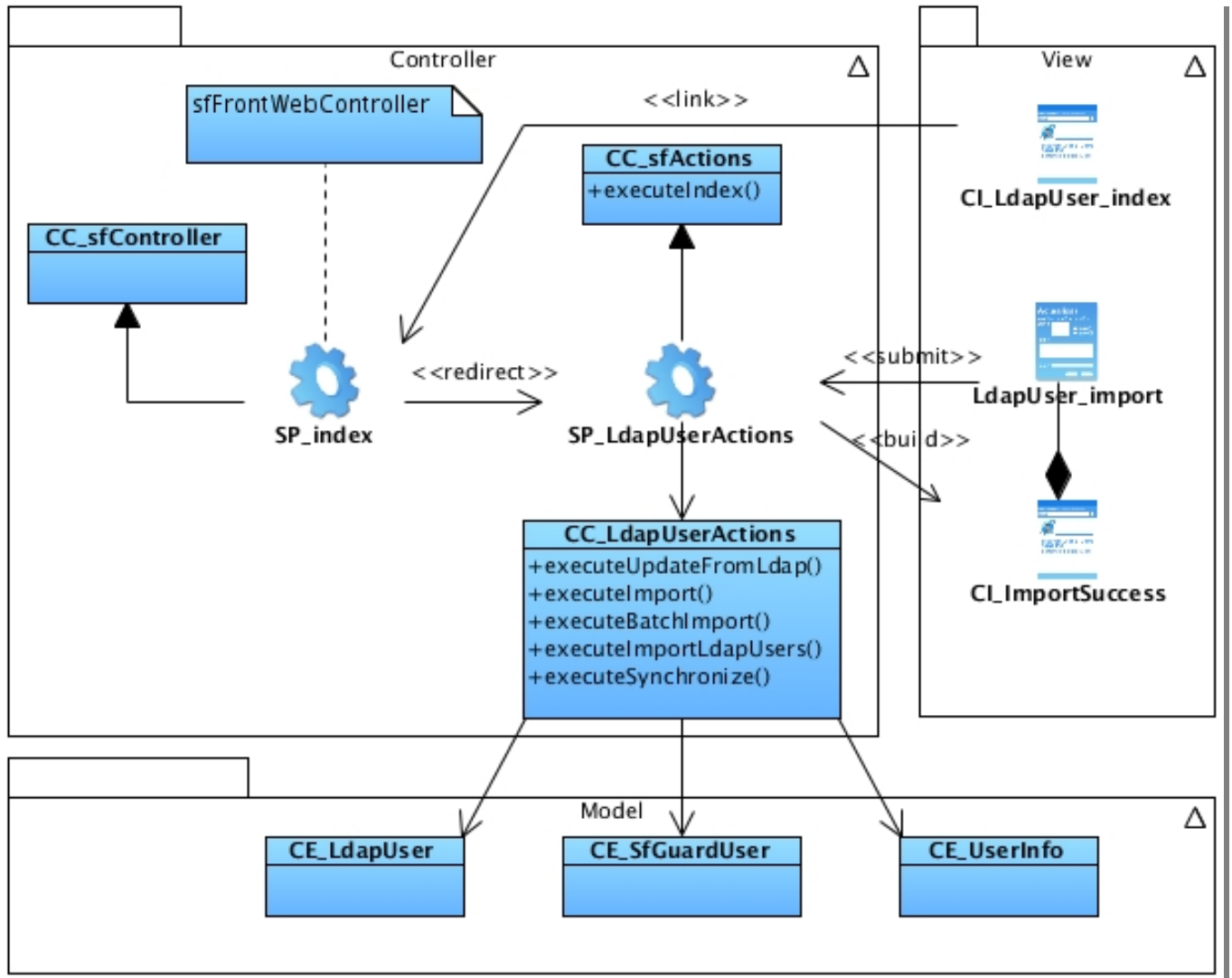


Figura 3.7: Diagrama de clases del diseño. CU Importar usuarios. Sección Importar un usuario.

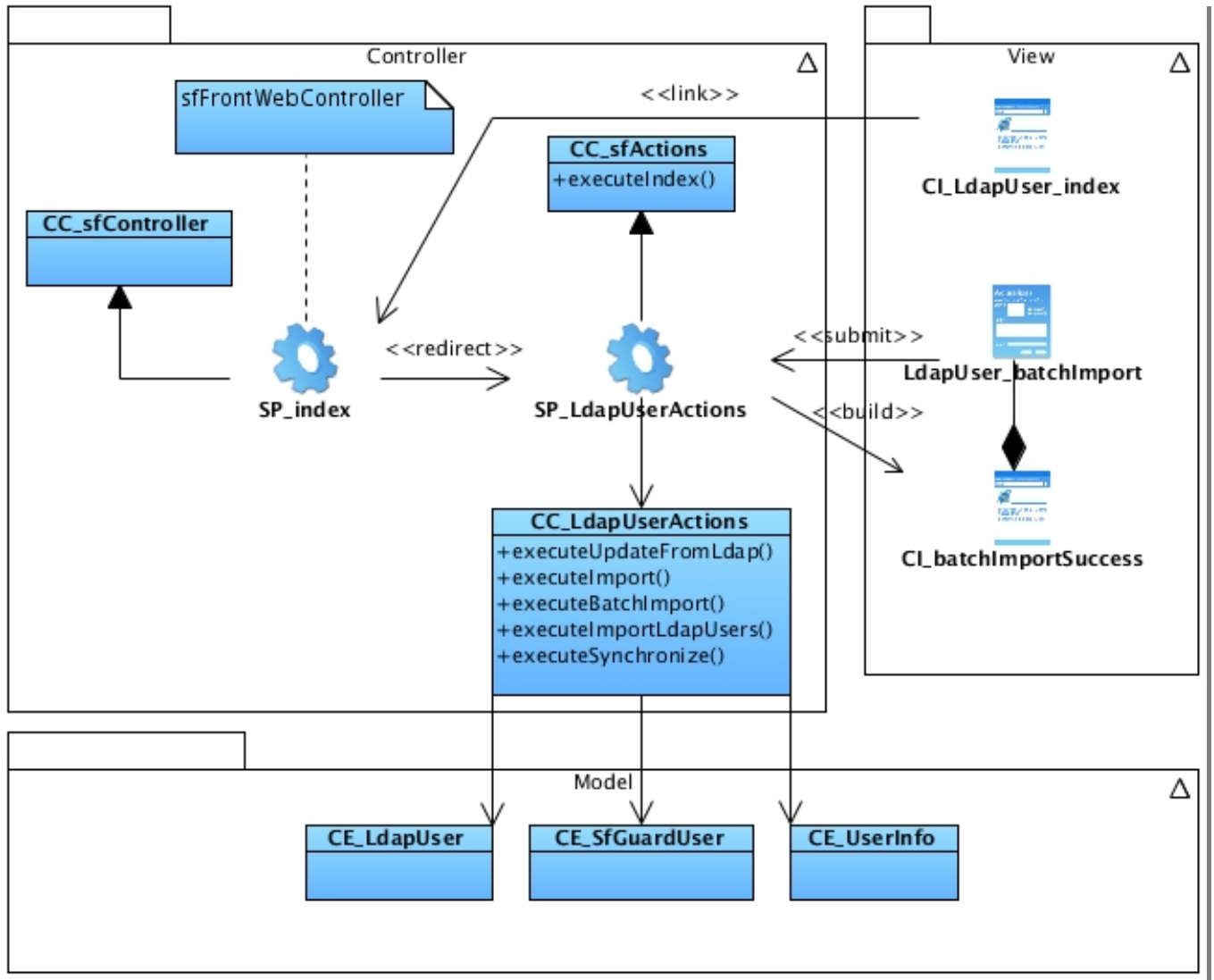


Figura 3.8: Diagrama de clases del diseño. CU Importar usuarios. Sección Importar usuarios seleccionados.

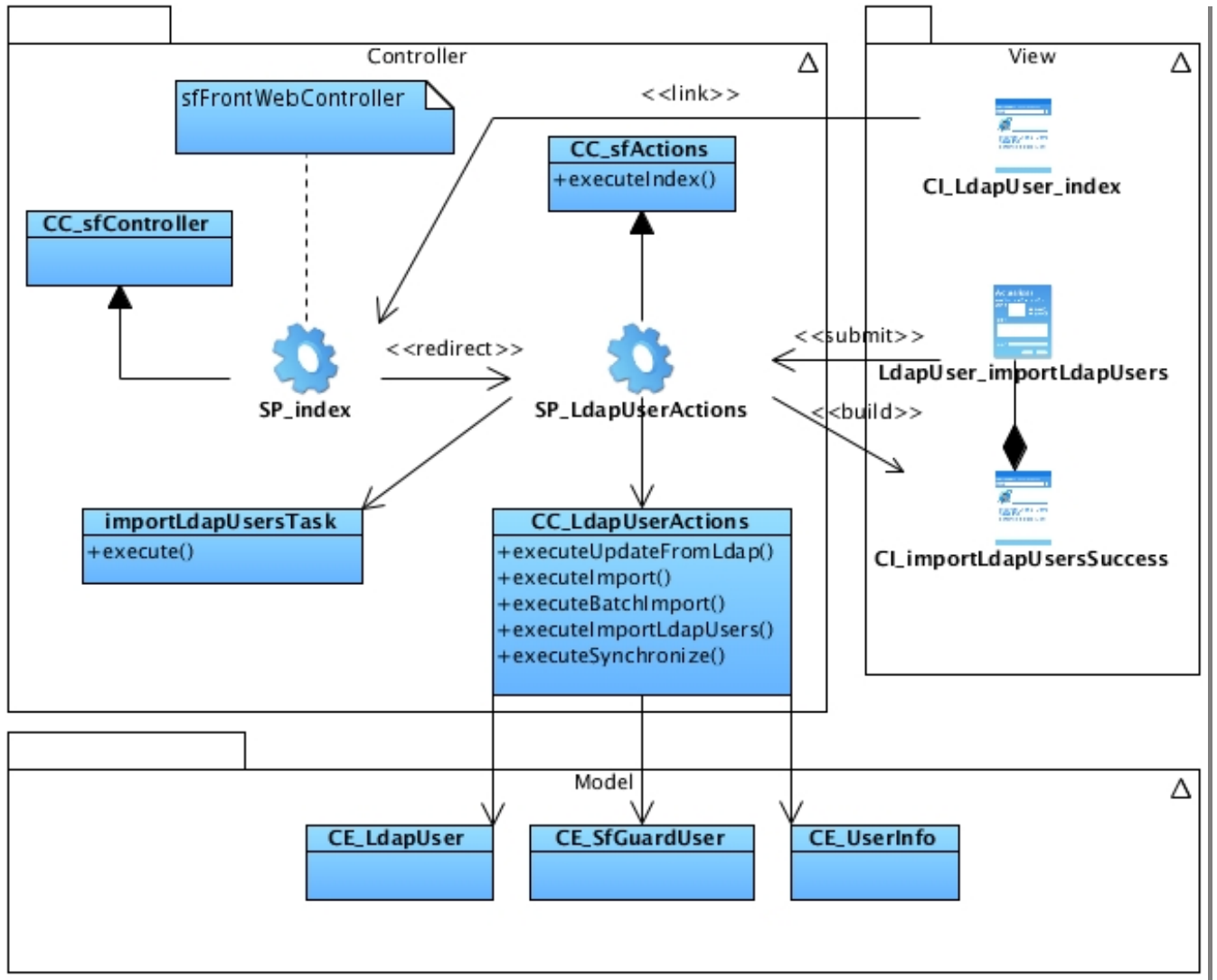


Figura 3.9: Diagrama de clases del diseño. CU Importar usuarios. Sección Importar usuarios filtrados.

Nota: Los restantes diagramas de clases del diseño pueden ser consultados en el anexo [C](#).

3.4 Diagramas de interacción

Los diagramas de interacción representan la forma en que un cliente u objeto se comunica entre sí en petición a un evento. No son sólo importantes para modelar los aspectos dinámicos de un sistema, sino

también para construir sistemas ejecutables por medio de ingeniería directa e inversa [27].

Los siguientes diagramas muestran la interacción entre las clases y los actores del CU Importar usuarios.

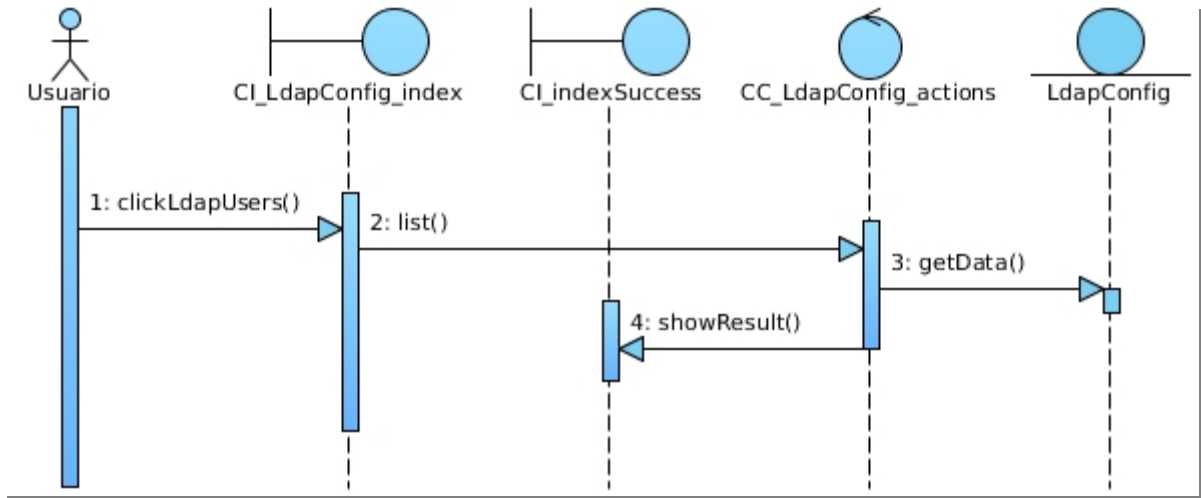


Figura 3.10: Diagrama de secuencia. CU Importar usuarios. Sección Listar usuarios del LDAP.

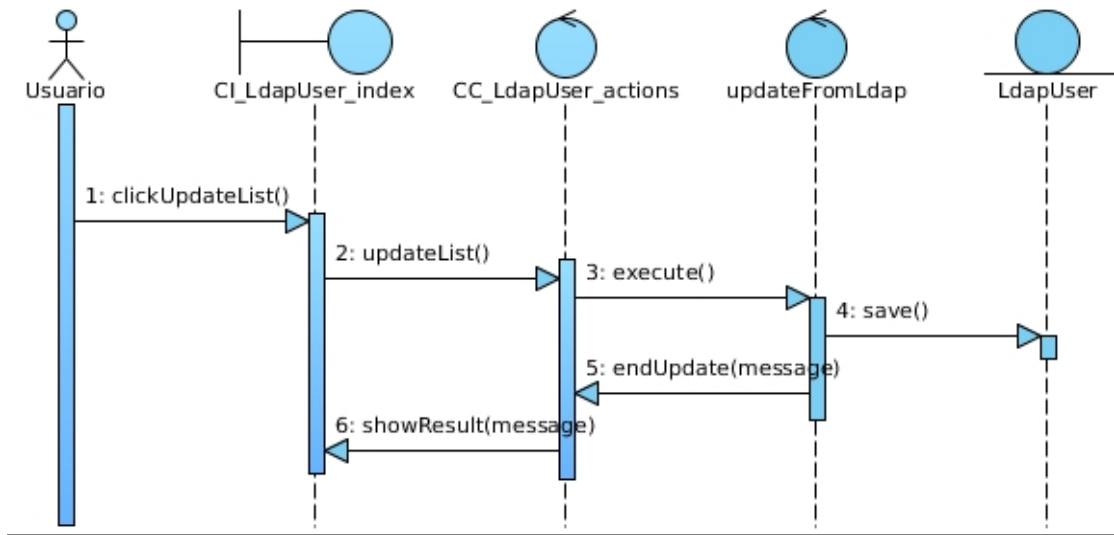


Figura 3.11: Diagrama de secuencia. CU Importar usuarios. Sección Actualizar listado de usuarios.

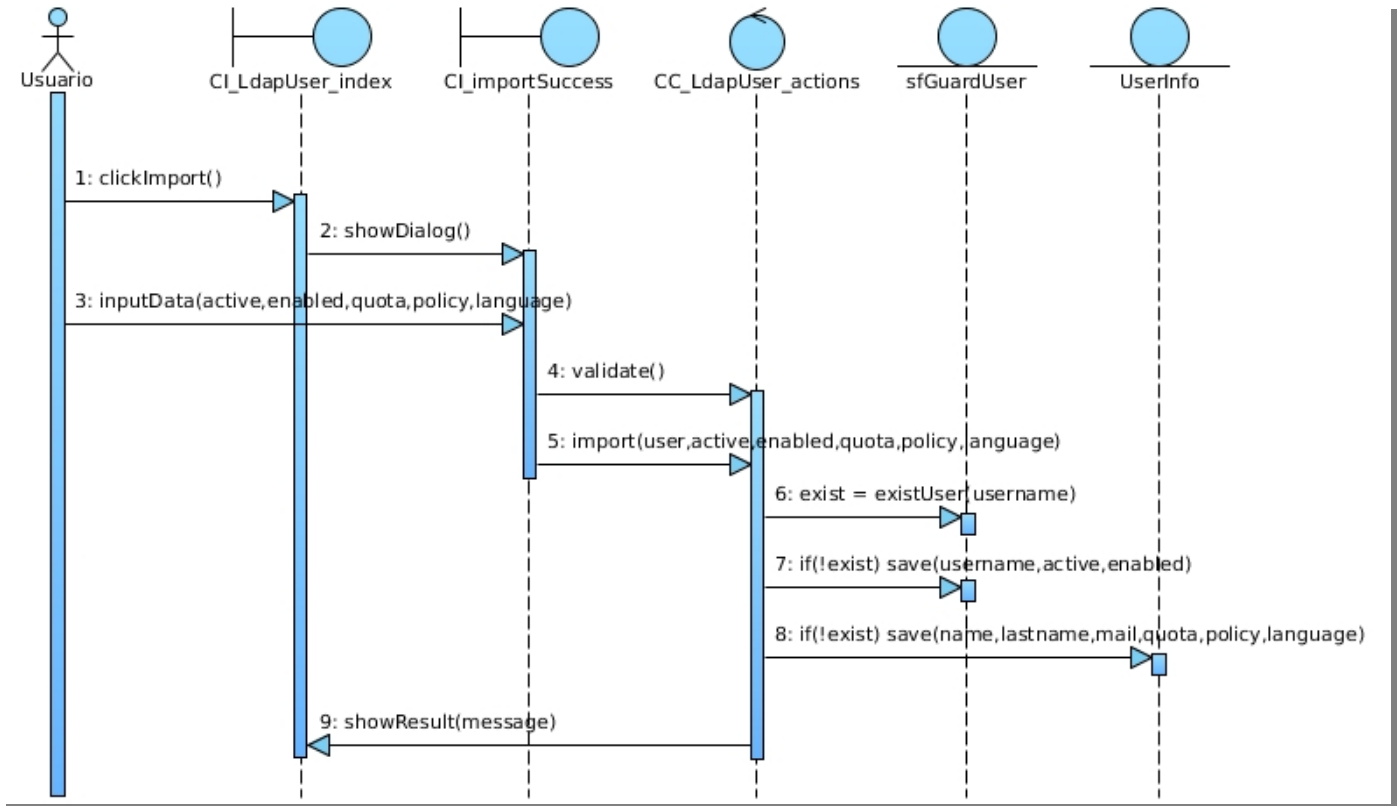


Figura 3.12: Diagrama de secuencia. CU Importar usuarios. Sección Importar un usuario.

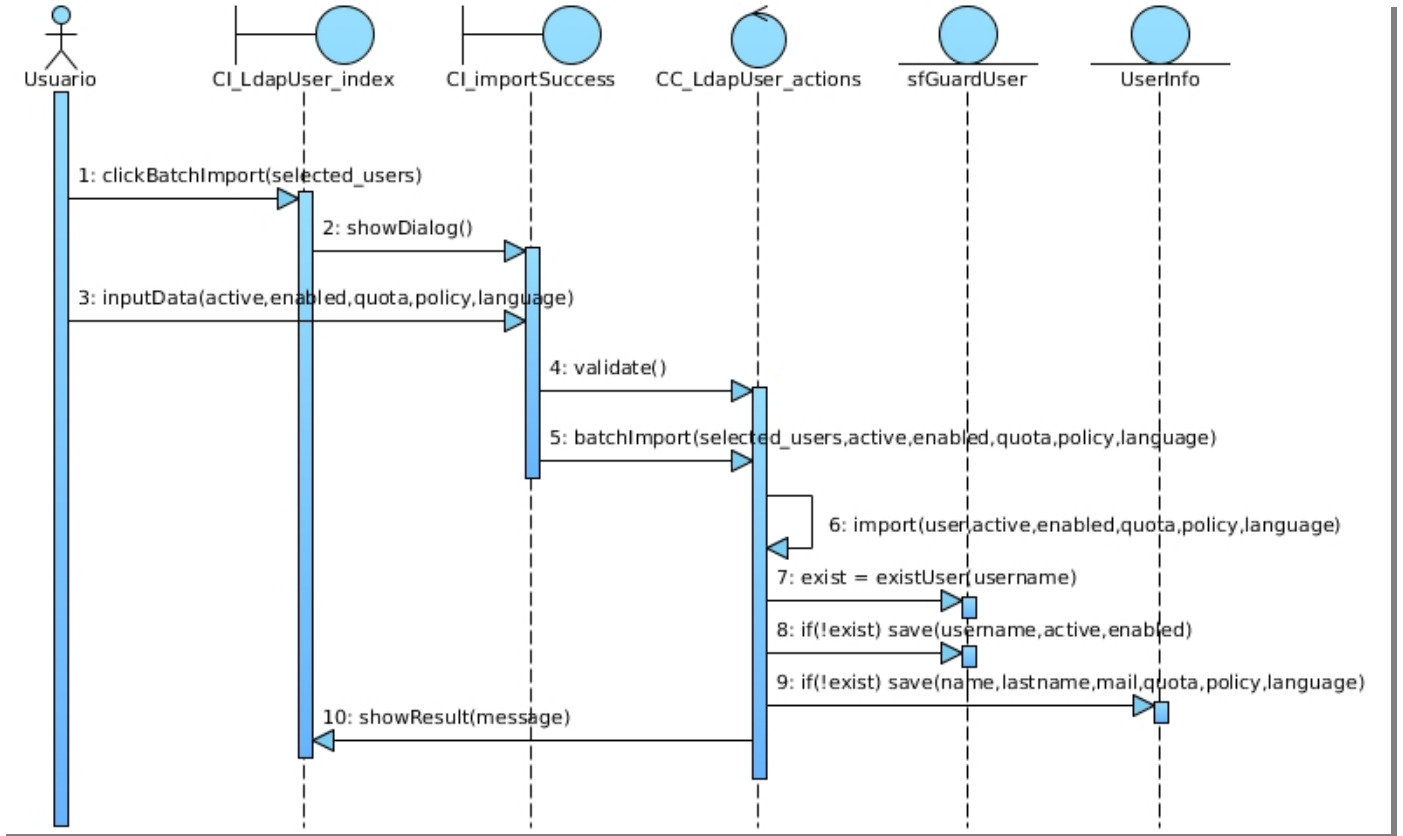


Figura 3.13: Diagrama de secuencia. CU Importar usuarios. Sección Importar usuarios seleccionados.

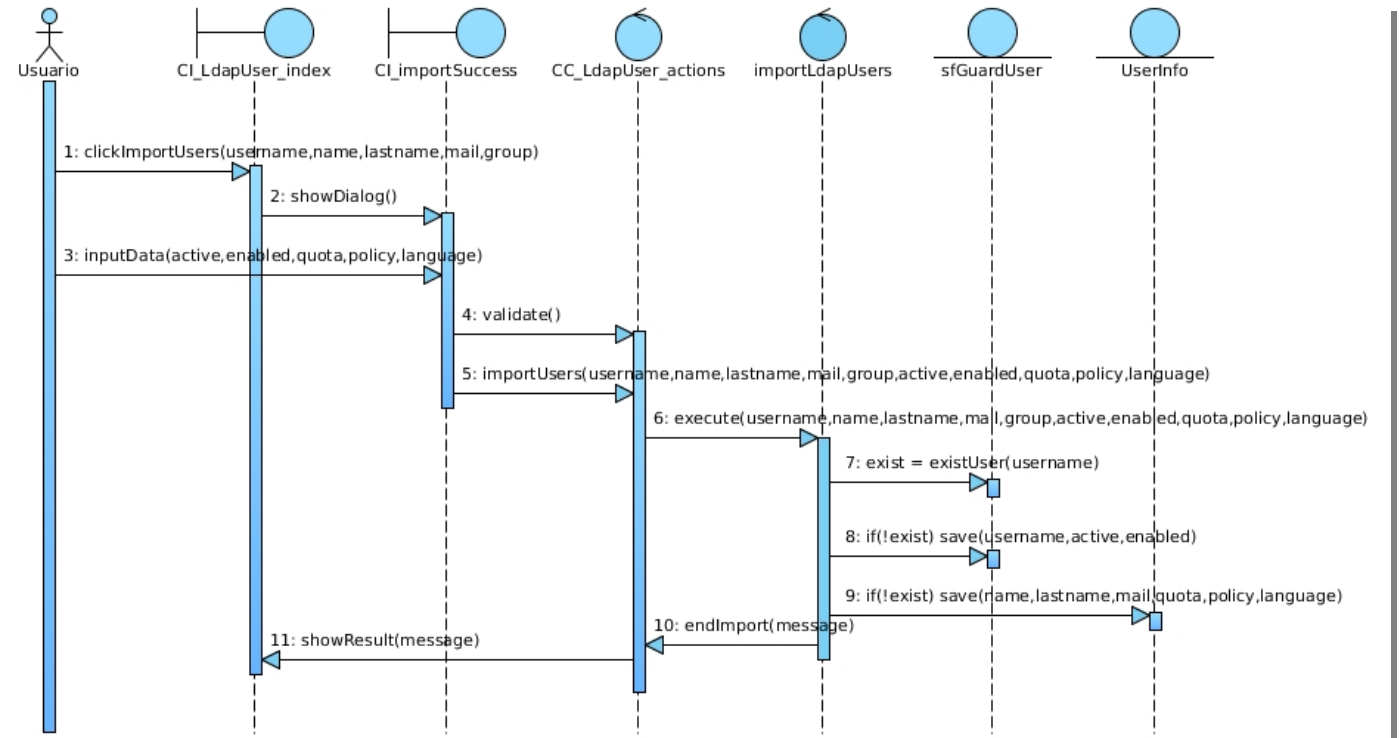


Figura 3.14: Diagrama de secuencia. CU Importar usuarios. Sección Importar usuarios filtrados.

Nota: Los restantes diagramas de secuencia pueden ser consultados en el anexo [D](#).

3.5 Diagrama de despliegue

El diagrama de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Dichos nodos son elementos de hardware sobre los cuales pueden ejecutarse los elementos de *software* [22]. El diagrama de despliegue se utiliza para visualizar la distribución de los componentes de *software* en los nodos físicos. A continuación se muestra una descripción de los elementos que componen un diagrama de este tipo.

Nodos: elementos de procesamiento con al menos un procesador, memoria y posiblemente otros dispositivos.

Dispositivos: nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.

Conectores: expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo. Como se puede apreciar en la figura 3.16, el *plugin* LdapPlugin se integrará a la IAW de Smart Keeper. Dicho *plugin* será accedido por los usuarios usando un navegador web desde una PC cliente, a través de una conexión segura por HTTPS. El módulo realizará consultas al servidor LDAP a través del protocolo LDAP y actualizará la base de datos de usuarios de Smart Keeper.

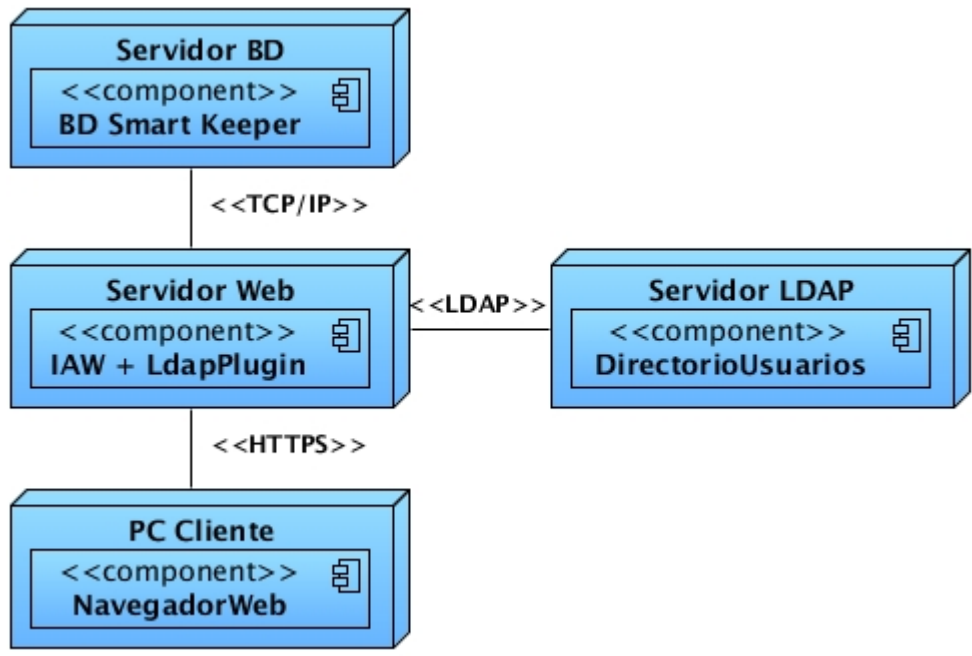


Figura 3.15: Diagrama de despliegue.

3.6 Modelo de datos

Un modelo de datos es aquel que describe la representación lógica y física de los datos persistentes en el sistema. En el desarrollo de la solución se ha hecho necesario garantizar la persistencia de los datos.

En el caso del sub-módulo `ldap_config`, se empleará una tabla con el mismo nombre. En dicha tabla se almacenan los parámetros de conexión de cada servidor LDAP y la configuración de la sincronización.

El módulo `ldap_user` también se relaciona con una tabla, con su mismo nombre. En este caso, la tabla `ldap_user` tiene como objetivo principal garantizar la persistencia de los datos de los usuarios del LDAP, de

forma temporal en la base de datos de Smart Keeper. El hecho de contar con una tabla en la base de datos que contenga los datos de los usuarios del LDAP brinda varias facilidades. En primer lugar, permite aprovechar las ventajas de Symfony ya que este automatiza la tarea repetitiva de crear módulos para manipular datos, mediante el uso de objetos *Propel*. Por otra parte, se hace posible generar el mecanismo de filtrado de usuarios. En resumen, el modelo de datos está compuesto por dos tablas: `ldap_user` y `ldap_config`, tal como se muestra en el diagrama 3.17.

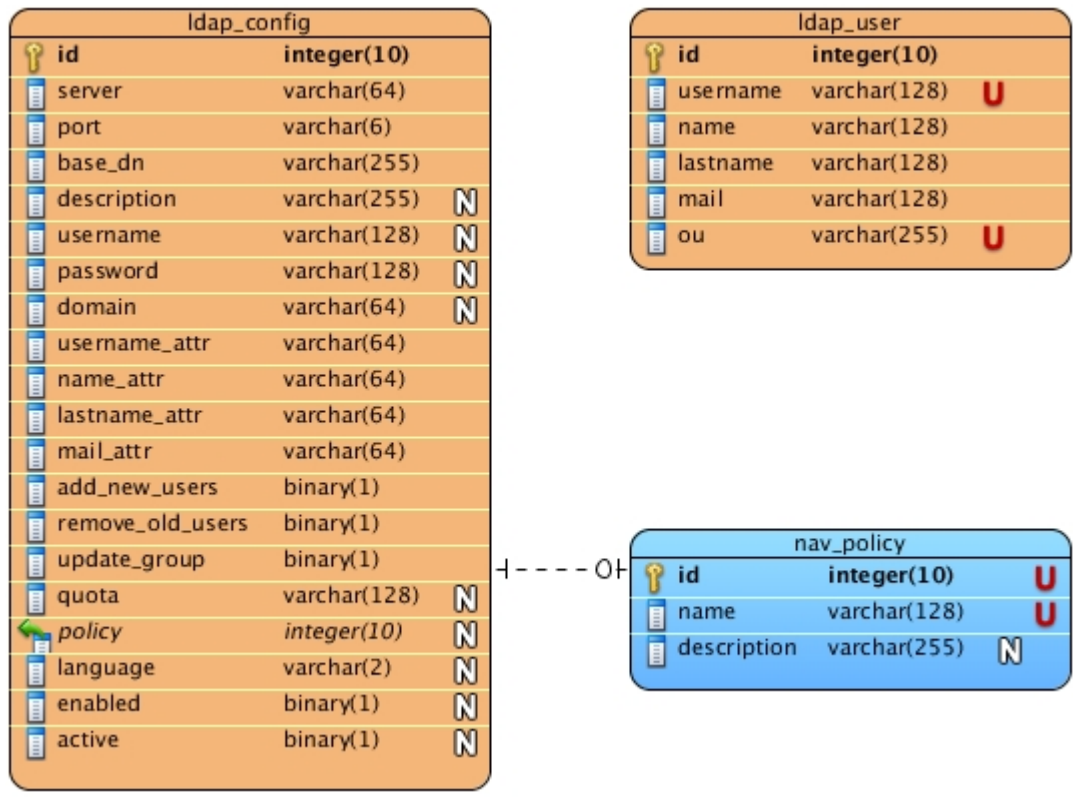


Figura 3.16: Modelo de datos.

Nota: La tabla `nav_policy` forma parte de la base de datos de Smart Keeper. Esta tabla ha sido incluida en el modelo debido a que está relacionada con la tabla `ldap_config`.

Las tablas 3.1 y 3.2 muestran una descripción detallada de cada una de las tablas propuestas.

3.6.1 Tablas del modelo de datos

Nombre: ldap_config		
Descripción: Contiene la configuración de la conexión y el comportamiento del proceso de sincronización para cada servidor existente.		
Atributo	Tipo	Descripción
id	integer(10)	Identificador de la configuración.
server	varchar(64)	Dirección IP o nombre del servidor LDAP.
port	varchar(6)	Puerto del servidor.
base_dn	varchar(255)	Nombre distinguido (DN) donde se encuentra la información de los usuarios.
description	varchar(255)	Almacena una descripción del servidor.
username	varchar(128)	Nombre de usuario para la conexión.
password	varchar(128)	Contraseña del usuario.
domain	varchar(64)	Dominio del usuario.
username_attr	varchar(64)	Almacena el nombre del atributo en el cual se encuentra el identificador de los usuarios en el LDAP.
name_attr	varchar(64)	Almacena el nombre del atributo en el cual se encuentra el nombre de los usuarios en el LDAP.
lastname_attr	varchar(64)	Almacena el nombre del atributo en el cual se encuentran los apellidos de los usuarios en el LDAP.
mail_attr	varchar(64)	Almacena el nombre del atributo en el cual se encuentra la dirección de correo de los usuarios en el LDAP.
add_new_users	binary(1)	Permite decidir si durante la sincronización se deben adicionar los usuarios del LDAP que aún no existen en el

		sistema.
remove_old_users	binary(1)	Permite decidir si durante la sincronización se deben eliminar del sistema los usuarios que ya no existen en el LDAP.
update_group	binary(1)	Permite decidir si durante la sincronización se debe sincronizar el grupo de navegación de los usuarios con respecto a la unidad organizacional de estos en el LDAP.
quota	varchar(128)	Almacena la cuota predeterminada para los nuevos usuarios.
policy	integer(10)	Determina la política a la que pertenecerán los nuevos usuarios.
language	varchar(2)	Almacena el idioma de la interfaz para los nuevos usuarios.
enabled	binary(1)	Determina si la navegación estará habilitada o no para los nuevos usuarios.
active	binary(1)	Determina si los nuevos usuarios tendrán o no acceso a la interfaz.

Tabla 3.1: Descripción de la tabla ldap_config.

Nombre: ldap_user		
Descripción: Contiene la información de los usuarios del LDAP, obtenidos como resultado de la última consulta realizada al servidor, de forma temporal.		
Atributo	Tipo	Descripción
id	integer(10)	Código que identifica al usuario.
username	varchar(128)	Identificador del usuario.
name	varchar(128)	Nombre del usuario.
lastname	varchar(128)	Apellidos del usuario.

mail	varchar(128)	Dirección de correo del usuario.
ou	varchar(255)	Nombre distinguido (DN) de la unidad organizacional a la que pertenece el usuario en el LDAP.

Tabla 3.2: Descripción de la tabla ldap_user.

3.7 Conclusiones parciales

En el presente capítulo se realizó una descripción de los principales patrones de diseño empleados en el desarrollo de la aplicación, los cuales proveerán sin dudas una arquitectura más sólida al sistema. A través de la modelación del análisis y del diseño se han sentado las bases para la implementación. El diagrama de despliegue ha permitido mostrar la distribución de los componentes de *software* en los nodos físicos, así como los protocolos de comunicación entre dichos nodos. El refinamiento del modelo de datos permitió diseñar la base de datos del *plugin*.

Capítulo 4

Capítulo 4: Implementación y pruebas

En el flujo implementación se comienza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, *scripts*, ficheros de código binario, ejecutables y similares. La implementación es el centro durante las iteraciones de construcción, aunque también se lleva a cabo durante la fase de transición, para tratar otros defectos encontrados en dicha fase. En el flujo de trabajo de prueba se verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias [22]. Las pruebas se realizan con el objetivo de validar el sistema desarrollado y detectar posibles fallos.

4.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de *software* que intervienen en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente.

A continuación se muestra el diagrama de componentes del módulo desarrollado:

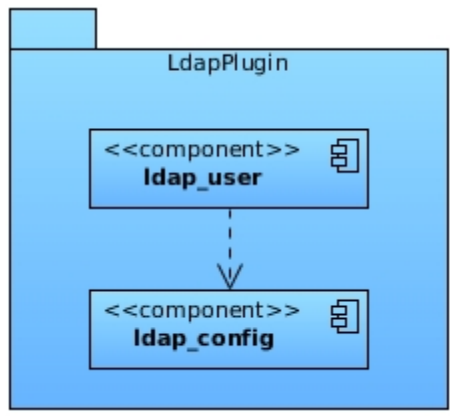


Figura 4.1: Diagrama de componentes de LdapPlugin.

Para una mayor comprensión sobre los componentes desarrollados, se muestra una vista más detallada de los mismos, separada por sub-módulos:

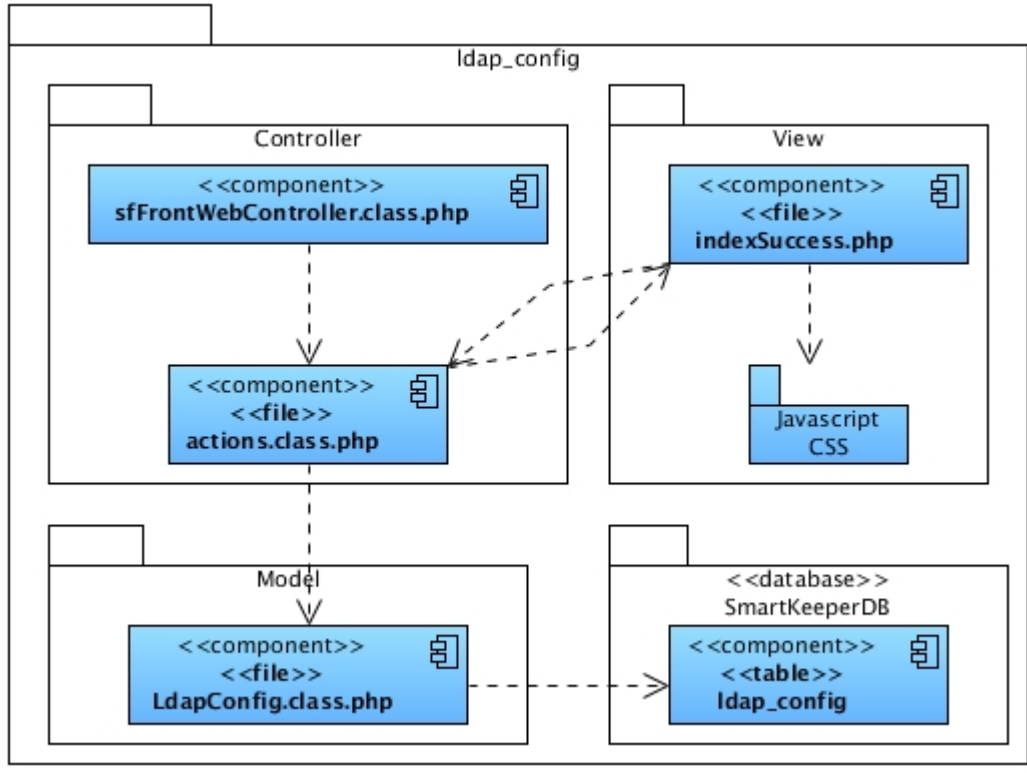


Figura 4.2: Diagrama de componentes del sub-módulo ldap_config.

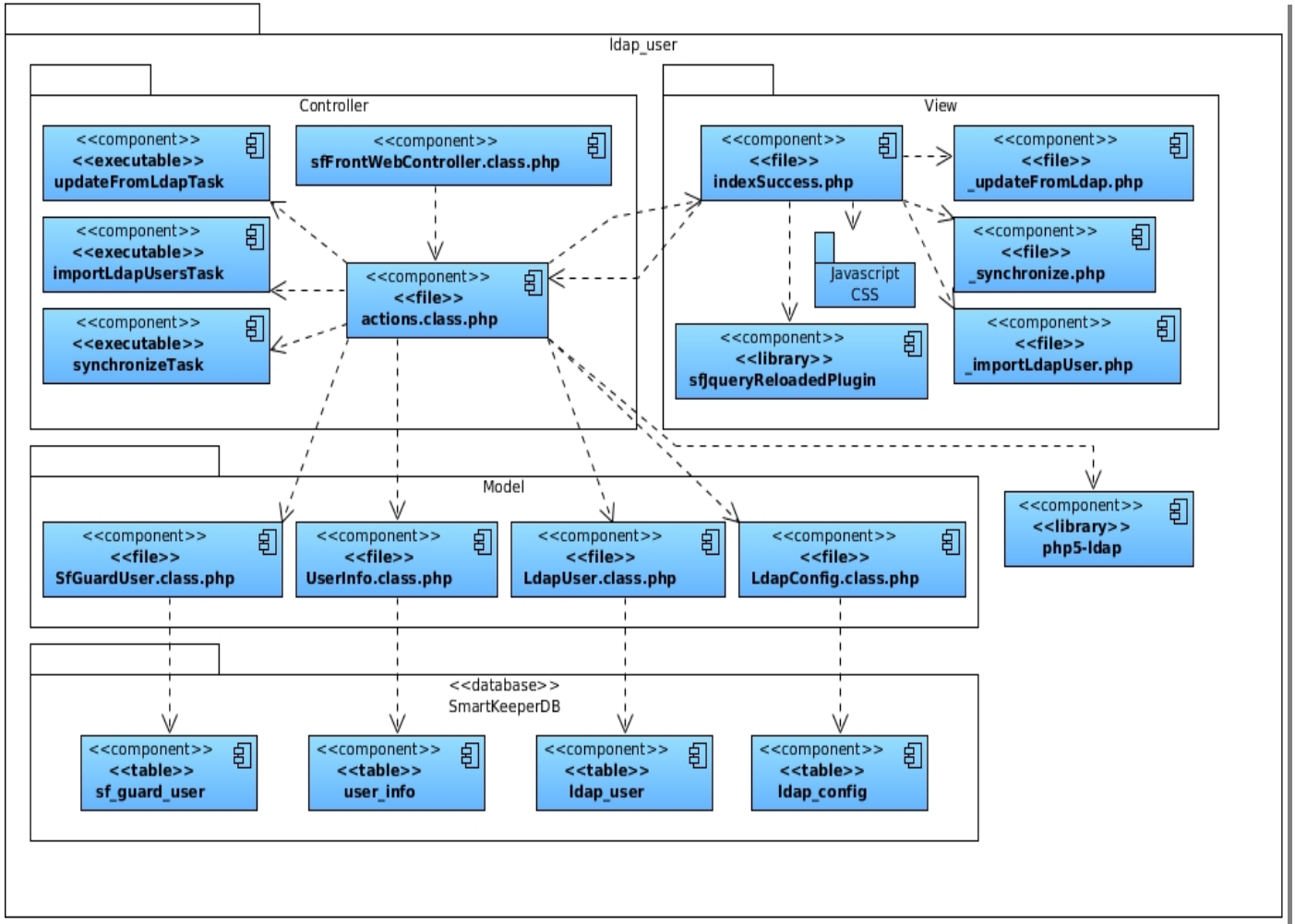


Figura 4.3: Diagrama de componentes del sub-módulo `ldap_user`.

4.1.1 Descripción de los componentes

Nombre del componente:	<code>importLdapUsersTask</code>
Tipo de clase:	Controladora
Atributos	
<code>active</code>	Determina el estado (activo o inactivo) de los usuarios importados.
<code>enabled</code>	Determina el estado de la navegación de los usuarios importados (activa o inactiva).

quota	Cuota límite de los nuevos usuarios.
language	Idioma para la IAW de los nuevos usuarios.
policy	Contiene el identificador de la política de navegación a la que pertenecerán los usuarios a importar.
username	Permiten especificar varios criterios de búsqueda para filtrar los usuarios del LDAP y solo realizar el proceso de importación a los que coincidan con dichos criterios.
name	
lastname	
mail	
ou	
Métodos	
execute	Ejecuta el proceso de importación basándose en la configuración de conexión marcada como activa y los parámetros de configuración establecidos para los nuevos usuarios.

Tabla 4.1: Descripción del componente importLdapUsersTask.

Nombre del componente:	synchronizeTask
Tipo de clase:	Controladora
Atributos	
added	Registro de la cantidad de usuarios adicionados.
removed	Cantidad de usuarios eliminados.
group_updated	Cantidad de usuarios a los que se les ha actualizado el grupo.
Métodos	
loadSetting	Determina las opciones de configuración establecidas para la sincronización.
prepareConnection	Establece la conexión con el servidor LDAP marcado como activo.
execute	Realiza el proceso de sincronización.

addNewUsers	Adiciona los usuarios existentes en el servidor LDAP que no existen en Smart Keeper.
removeDeletedUsers	Elimina los usuarios de Smart Keeper que ya no se encuentran en el servidor LDAP.
updateGroups	Actualiza el grupo de los usuarios en caso de que el mismo haya cambiado en el servidor LDAP.

Tabla 4.2: Descripción del componente synchronizeTask.

4.2 Pantallas principales de la aplicación

Las siguientes imágenes muestran las principales interfaces del *plugin* desarrollado.

Usuarios LDAP



<input type="checkbox"/>	Usuario	Nombre	Grupos	Acciones
<input type="checkbox"/>	aaballi	Abel Aballi Parodis	F01	Importar
<input type="checkbox"/>	aabraham	Angel Fernandez Abraham	F01	Importar
<input type="checkbox"/>	aacastro	Aliane Amelia Gonzalez Castro	F01	Importar
<input type="checkbox"/>	aacebo	Alejandro Acebo Denis	F01	Importar
<input type="checkbox"/>	aaclaro	Alejandro Almarales Claro	F01	Importar
<input type="checkbox"/>	aacortina	Amed Alcides Cortina Marquez	F01	Importar
<input type="checkbox"/>	aacueto	Adrian Avila Cueto	F01	Importar
<input type="checkbox"/>	aafernandez	Adonis Alfredo Fernandez Reyes	F01	Importar
<input type="checkbox"/>	aafundora	Ariel Antonio Fundora Lopez	F01	Importar
<input type="checkbox"/>	aalayo	Alexei Alayo Rondon	F01	Importar

1124 resultados (página 1/113) Resultados por página 10

Selecciona una acción ok Actualizar lista Importar usuarios Sincronizar

Figura 4.4: Pantalla principal del sub-módulo ldap_user.

Configuraciones de conexión

<input type="checkbox"/>	Servidor	DN Base	Descripción	Estado	Acciones
<input type="checkbox"/>	uci.cu	OU=F01,OU=Students,OU=UCI Domain Users,DC=uci,DC=cu	Facultad 1	<input checked="" type="radio"/> Activo	Clonar Editar Borrar
<input type="checkbox"/>	uci.cu	OU=UCI Domain Graduated,DC=uci,DC=cu	UCI Graduados	<input type="radio"/> Inactivo	Clonar Editar Borrar
<input type="checkbox"/>	ldap.uci.cu	DC=uci,DC=cu	LDAP UCI	<input type="radio"/> Inactivo	Clonar Editar Borrar
<input type="checkbox"/>	uci.cu	OU=F02,OU=Students,OU=UCI Domain Users,DC=uci,DC=cu	Facultad 2	<input type="radio"/> Inactivo	Clonar Editar Borrar
<input type="checkbox"/>	uci.cu	OU=UCI Domain Users,DC=uci,DC=cu	UCI	<input type="radio"/> Inactivo	Clonar Editar Borrar

5 resultados Resultados por página 10

Selecciona una acción ok + Adicionar

Filtrar

Servidor

Puerto

DN Base

Descripción

Activo

Campos

Servidor
 DN Base
 Descripción
 Estado

Restablecer

Figura 4.5: Pantalla principal del sub-módulo ldap_config.

Editar configuración LDAP

Configuración de la conexión	Mapeo de atributos	Sincronización
Adicionar usuarios	<input checked="" type="checkbox"/>	Adiciona los usuarios de LDAP que aún no existen en el sistema
Eliminar usuarios	<input checked="" type="checkbox"/>	Eliminar del sistema los usuarios que ya no existen en LDAP
Actualizar grupos	<input checked="" type="checkbox"/>	Sincronizar el grupo de navegación de los usuarios con la unidad organizacional de LDAP
Activo	<input checked="" type="checkbox"/>	
Navegación	<input checked="" type="checkbox"/>	
Cuota	<input type="text" value="100mb/m"/>	
Política	<input type="text" value="default"/>	
Idioma	<input type="text" value="español"/>	

Figura 4.6: Formulario de edición del sub-módulo ldap_config. Pestaña Sincronización.

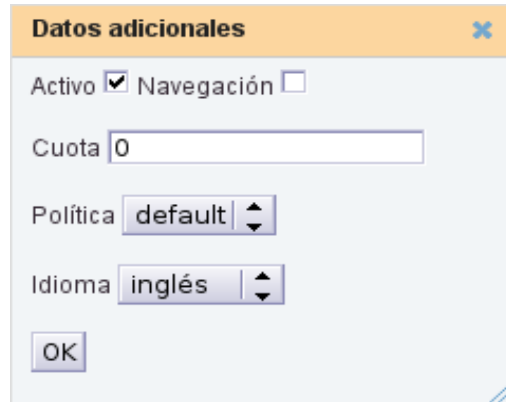


Figura 4.7: Formulario de configuración de los parámetros de los usuarios a importar.

4.3 Validación del sistema

El desarrollo de un *software* implica una serie de actividades en las que existe la posibilidad del fallo humano. Los errores pueden comenzar desde el inicio del proceso, en el que los objetivos pueden estar especificados de forma errónea o imperfecta, así como dentro de pasos de diseño y desarrollo posteriores [23]. Las pruebas no pueden asegurar la ausencia de errores, pero son capaces de demostrar que existen defectos o fallos en el *software*.

4.3.1 Técnica de prueba

El *software* debe ser probado con el objetivo de descubrir y corregir el máximo de errores posibles antes de su entrega al cliente. Para esto es necesario diseñar una serie de casos de pruebas que tengan una alta probabilidad de encontrar errores. Existen dos técnicas de prueba: caja blanca y caja negra. La técnica de caja negra (o funcional) se centra en los requisitos funcionales del *software*, es por ello que ha sido empleada en el diseño de los casos de prueba. También permiten al ingeniero del *software* obtener un conjunto de condiciones de entrada que prueben completamente todos los requisitos funcionales del sistema [23].

4.3.2 Tipos de pruebas aplicadas

Para comprobar la calidad del producto desarrollado, se realizaron pruebas funcionales, las cuales consisten en evaluar las funcionalidades del sistema. Estas están centradas en asegurar que el producto

Capítulo 4: Implementación y pruebas

satisface los requisitos funcionales establecidos por el cliente. Para la realización de dichas pruebas se diseñaron casos de prueba basados en Casos de Uso, los cuales pueden ser consultados en el documento “Diseño de Casos de Prueba basado en Casos de Uso.xls” del expediente del proyecto Smart Keeper.

También se realizaron pruebas de integración. Las pruebas de integración son una técnica sistemática para construir la estructura del programa, así como para detectar errores en la interacción de los componentes [23]. Su aplicación permite detectar errores funcionales incorporados al integrar el sistema como un todo. Para la ejecución de estas pruebas, el *plugin* fue integrado a la IAW de Smart Keeper, por un tiempo aproximado de 2 meses. Este tipo de pruebas resulta de gran importancia, ya que dicho *plugin* comparte con Smart Keeper varios componentes, implicando la introducción de errores y alterando el comportamiento normal del sistema.

Además, fueron aplicadas pruebas de aceptación. Las pruebas de aceptación son realizadas por el usuario final y su objetivo es verificar que el *software* está listo para ser usado [28].

4.3.3 Resultado de las pruebas

Con el objetivo de obtener un producto de calidad y con la menor cantidad posible de errores se ejecutaron 2 iteraciones de prueba. En una primera iteración fueron detectadas 4 no conformidades, fundamentalmente relacionadas con la internacionalización y la validación de los datos en los formularios. Al aplicar una segunda iteración se corrigieron los errores anteriores y se obtuvieron resultados satisfactorios. Como resultado de la integración se concluyó, que el *plugin* se integra al diseño de la IAW y a los mecanismos de seguridad de forma satisfactoria.

4.4 Conclusiones parciales

El diagrama de componentes permitió obtener una visión más clara sobre la estructura del *plugin* desarrollado. Como resultado de la implementación se ha obtenido un sistema funcional, completamente operativo. La interfaz web del *plugin* se adapta a los requisitos definidos y brinda la posibilidad de gestionar las funcionalidades con facilidad. La realización de pruebas al *software* implementado ha permitido validar la solución desarrollada y comprobar que se adapta a los requerimientos definidos.

Conclusiones

La realización del presente trabajo ha posibilitado el cumplimiento de los objetivos propuestos. En general, se ha arribado a las siguientes conclusiones:

- El estudio de los sistemas que poseen mecanismos de integración con LDAP, tanto a nivel nacional como internacional, sentó las bases para el desarrollo de la solución.
- El *plugin* desarrollado para Smart Keeper permite realizar el proceso de importación y sincronización desde la IAW de dicho sistema lo que automatiza la creación de cuentas de usuarios.
- Las pruebas aplicadas permitieron validar que el *plugin* desarrollado cumple con todos los requerimientos de *software* definidos y se integra correctamente a la IAW.

Recomendaciones

Los resultados obtenidos luego del desarrollo del presente trabajo, satisfacen los requerimientos definidos. No obstante, para el desarrollo de futuras versiones del *plugin* se recomienda:

- Incluir soporte para importar los usuarios a partir de un fichero LDIF o desde un fichero en texto plano.
- Valorar la posibilidad de poder programar desde la IAW el horario de ejecución de la tarea de sincronización.

Glosario de términos

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.

HTML: Lenguaje compuesto de una serie de etiquetas o marcas que permiten 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.

IAW: Interfaz de Administración Web.

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

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.

ORM: Mapeo Relacional de Objetos.

OSI: Interconexión de Sistemas Abiertos.

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.

RUP: Proceso Unificado de Rational.

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.

Glosario de términos

TCP: Protocolo de Control de Transferencia.

ITU: Es el organismo especializado de las Naciones Unidas encargado de regular las telecomunicaciones a nivel internacional.

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

XHTML: HTML escrito según las normas que marca XML. Por tanto, se trata de una aplicación concreta de XML y no tienen que confundirse entre si.

Referencia bibliográfica

- [1] Gonzalo Álvarez Marañón, José María Gómez Hidalgo. **Filtrado de contenidos web**, PCWorld, 2011, No. 284 (Marzo): 110-111.
- [2] Vassiliki Koutsonikola y Athena Vakali. **LDAP: Framework, Practices, and Trends**, IEEE Internet Computing, 2004, No. 5 (September-October): 1-7.
- [3] Francisco Olcina Grande. **Análisis, diseño e implantación de un sistema de servidores departamental basado en máquinas virtuales**. Universidad Carlos III de Madrid. Leganés, 2007. 307.
- [4] Rafael Calzada Pradas. **Introducción al servicio de directorio**, 2001. [Disponible en: <http://www.rediris.es/ldap/doc/ldap-intro.pdf>].
- [5] Microsoft. **Utilización de LDIFDE para importar y exportar a Active Directory objetos del directorio**, 2006. [Disponible en: <http://support.microsoft.com/kb/237677/es>].
- [6] Ramón A. Anglada Martínez. **Plataforma de Gestión de Servicios Telemáticos**, 2010. [Disponible en: <http://semanatecnologica.fordes.co.cu/ocs-2.3.2/public/site/152.pdf>].
- [7] phpLDAPadmin. **phpLDAPadmin main page**, 2012. [Disponible en: http://phpldapadmin.sourceforge.net/wiki/index.php/Main_Page].
- [8] Webmin. **Introduction to Webmin**, 2011. [Disponible en: <http://www.webmin.com/intro.html>].
- [9] WebSite Scripts. **KnowledgeBase Manager Pro**, 2011. [Disponible en: <http://www.web-site-scripts.com/knowledge-management/overview.html>].
- [10] SmarterMail. **Import New Users Using LDAP**, 2011. [Disponible en: <http://portal.smartertools.com/KB/a1449/import-new-users-using-ldap.aspx>].
- [11] GitHub. **slapORM**, 2010. [Disponible en: <https://github.com/chanmix51/slapOrm/wiki>].
- [12] Symfony Project. **dcLDAPAbstractionPlugin**, 2011. [Disponible en: <http://www.symfony-project.org/plugins/dcLDAPAbstractionPlugin>].
- [13] Symfony Project. **upSimpleLdapPlugin**, 2011. [Disponible en: <http://www.symfony-project.org/plugins/upSimpleLdapPlugin>].
- [14] Jimi Marley. **¿Por qué elegir PHP?**, 2011. [Disponible en: <http://www.programacion.com/articulo/>]

Referencia bibliográfica

[por que elegir php 143](#)].

[15] Gerardo Herrera. **PHP**, 2006. [Disponible en: <http://www.vaslibre.org.ve/publicaciones/phpflisol2006.pdf>].

[16] Harish Kamath. **Using PHP with LDAP**, 2003. [Disponible en: <http://www.devshed.com/index.php?option=content&task=view&id=193&Itemid=29/>].

[17] Rafael Martínez. **Sobre PostgreSQL**, 2010. Disponible en: <http://www.postgresql.org.es/sobrepostgresql#caracteristicas>].

[18] Fabien Potencier, Francois Zaninotto. **Symfony: La guía definitiva**. LibrosWeb, 2008. Páginas 6-37.

[19] Netbeans. **Welcome to Netbeans**, 2011. [Disponible en: <http://netbeans.org/>].

[20] Visual Paradigm. **Visual Paradigm for UML**, 2011. [Disponible en: <http://www.visual-paradigm.com/product/vpum/>].

[21] Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. **Metodologías tradicionales vs. metodologías ágiles**, 2008. [Disponible en: <http://adonisnet.wordpress.com/2008/06/18/metodologias-tradicionales-vs-metodologias-agiles/>].

[22] Jacobson, I., Booch, G., Rumbaugh J. **El Proceso Unificado de Desarrollo de Software**. Addison Wesley, 2000. Páginas 255-256, 282.

[23] Roger S. Pressman. **Ingeniería del Software, un enfoque práctico**. McGraw-Hill, 2001. Páginas 255-281, 303- 338.

[24] Wikipedia, la enciclopedia libre. **Patrón de diseño**, 2011. [Disponible en: http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o].

[25] Popkin Software and Systems. **Modelado de Sistemas con UML**, 2008. [Disponible en: <http://es.scribd.com/Samirak/d/1838816-docmodeladosistemasuml>].

[26] José Manuel Godoy Giménez. **Diseño de proyectos de software en código abierto**, 2002. [Disponible en: <http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-dise%F1o-software/doc-dise%F1o-software-parte-1.pdf>].

[27] Patricio Salinas Caro. **Tutorial de UML**. [Disponible en: <http://www.dcc.uchile.cl/~psalinas/uml/>].

[28] Mario Barrientos, Yamila Nieves Hernández. **Automatización de Pruebas Funcionales para Aplicaciones Web**. Universidad de las Ciencias Informáticas. Habana, 2010. 106

Bibliografía

1. Vassiliki Koutsonikola y Athena Vakali. **LDAP: Framework, Practices, and Trends**, IEEE Internet Computing, 2004, No. 5.
2. Rafael Calzada Pradas. **Introducción al servicio de directorio**, 2001. [Disponible en: <http://www.rediris.es/ldap/doc/ldap-intro.pdf>].
3. Harish Kamath. **Using PHP with LDAP**, 2003. [Disponible en: <http://www.devshed.com/index.php/?option=content&task=view&id=193&Itemid=29/>].
4. Fabien Potencier, François Zaninotto. **Symfony: La guía definitiva**. LibrosWeb, 2008.
5. Roger S. Pressman. **Ingeniería del Software, un enfoque práctico**. McGraw-Hill, 2001.
6. Jacobson, I., Booch, G., Rumbaugh J. **El Proceso Unificado de Desarrollo de Software**. Addison Wesley, 2000.
7. Jose Manuel Suárez. **Curso OpenLDAP**, 2004. [Disponible en: www.redes-linux.com/manuales/openldap/curso_openldap.pdf].
8. Heinz Johner, Larry Brown, Franz-Stefan Hinner, Wolfgang Reis, Johan Westman. **Understanding LDAP**. IBM International Technical Support Organization, 1998.
9. **Lightweight Directory Access Protocol**, 2012. [Disponible en: <http://www.php.net/manual/en/book.ldap.php>].

Anexo A

Figuras relacionadas

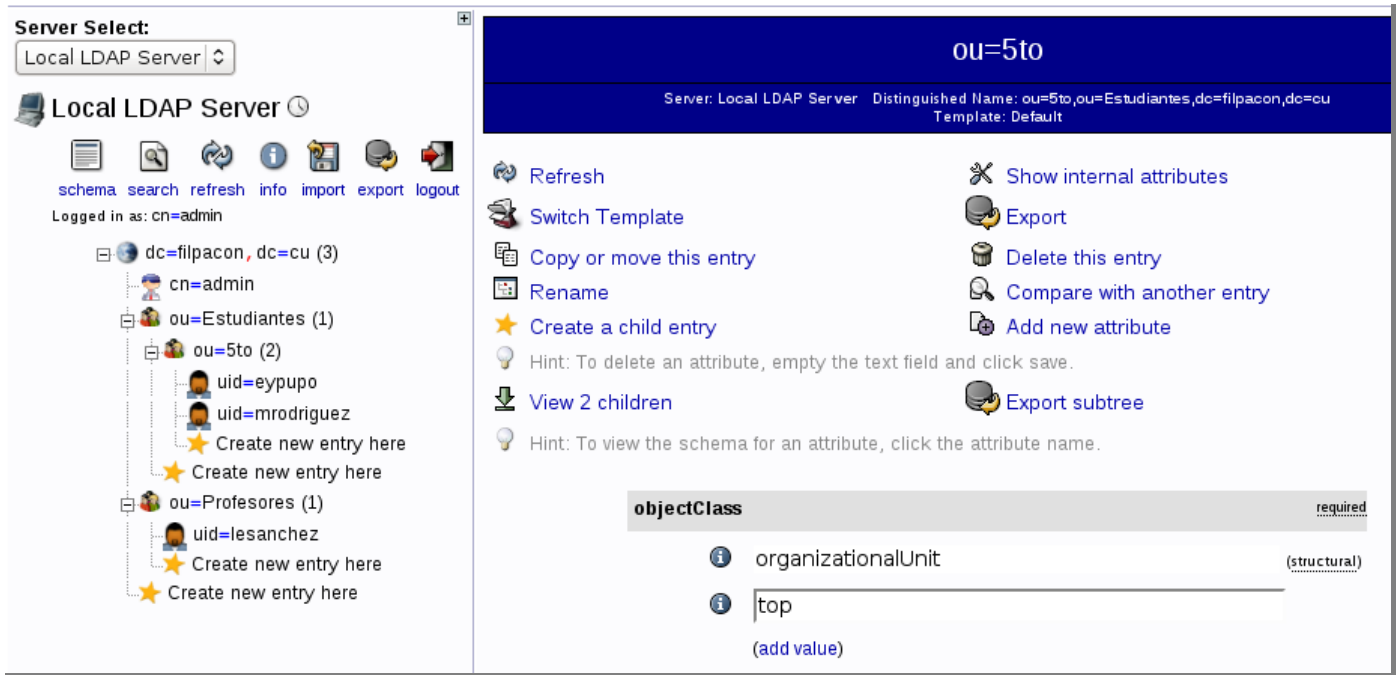


Figura A.1: Interfaz de la herramienta phpLDAPadmin.

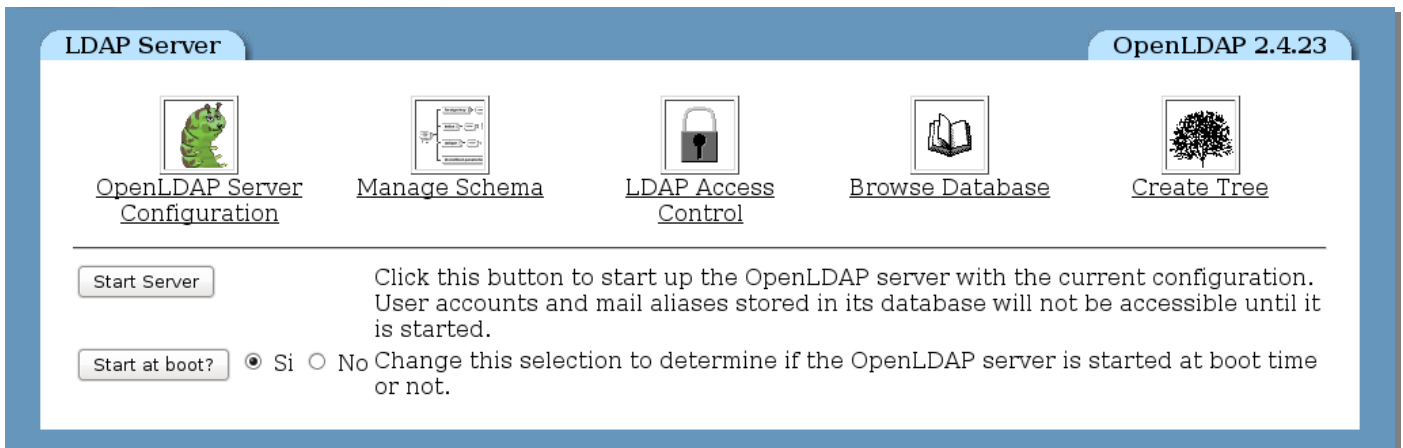


Figura A.2: Vista de la página de administración de LDAP desde Webmin.

General Application Settings

General | SMTP | Attachments | Auto Related Articles | User Preferences | **LDAP Settings** | Search | File Permissions | API Server | Sitemap

Save Test Settings

LDAP Integration Settings

Enable LDAP Authentication?	<input checked="" type="checkbox"/> ?
LDAP Platform: *	MS ActiveDirectory ?
LDAP Host: *	web-site-scripts.com ?
LDAP Port: *	389 ?
Allow Follow Referrals?	<input type="checkbox"/> ?
LDAP Version 3?	<input type="checkbox"/> ?
Negotiate TLS?	<input type="checkbox"/> ?
Search String: *	samaccountname=[login] ?
Base DN: *	ou=mitridat,dc=web-site-scripts,dc=com ?
Connect Username: *	example@web-site-scripts.com ?
Connect Password: ?

LDAP Synchronization Settings

Map Login To: *	samaccountname ?
Map First Name to: *	givenname ?
Map Last Name to: *	sn ?
Map Email to: *	mail ?
Sync User Details?	<input checked="" type="checkbox"/> ?
Sync User Groups?	<input checked="" type="checkbox"/> ?

Figura A.3: Página de configuración de LDAP de KnowledgeBase Manager Pro.

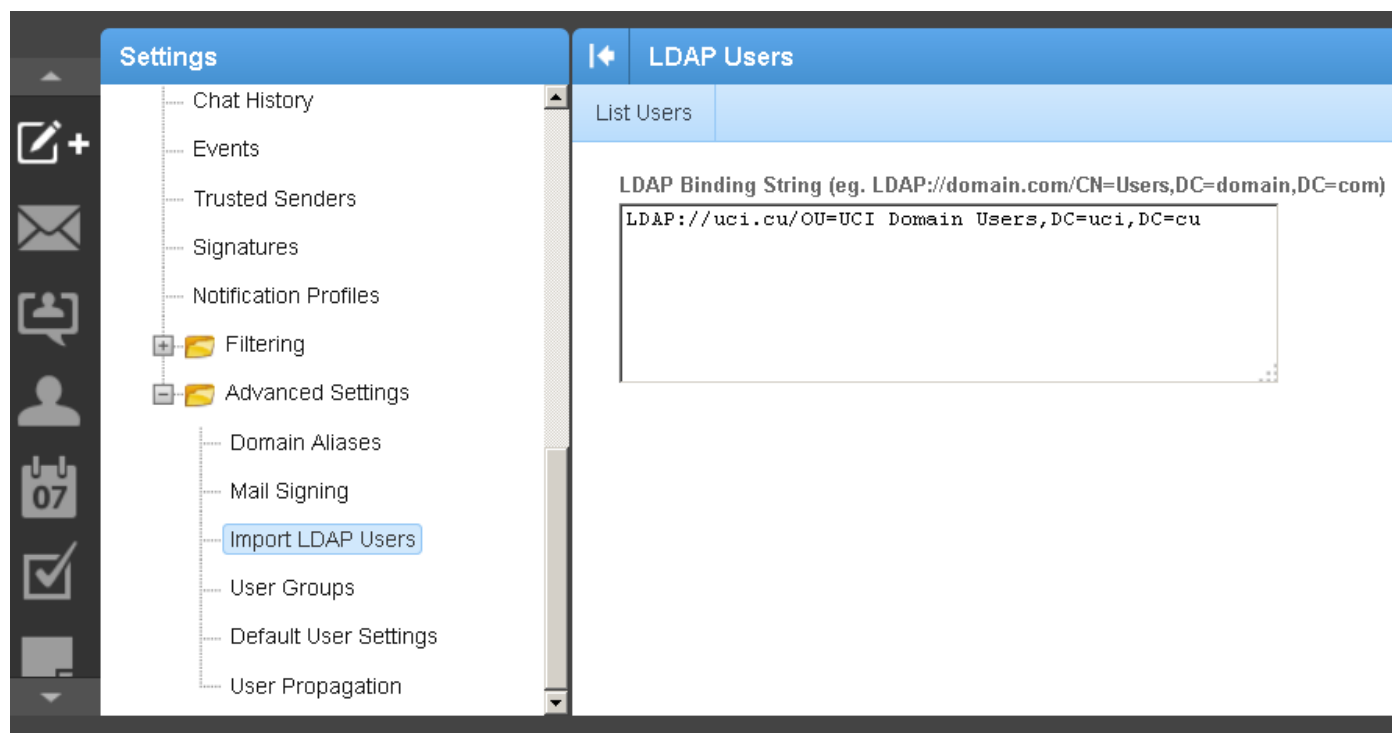


Figura A.4: Formulario para configurar la importación de usuarios LDAP en SmarterMail 8.

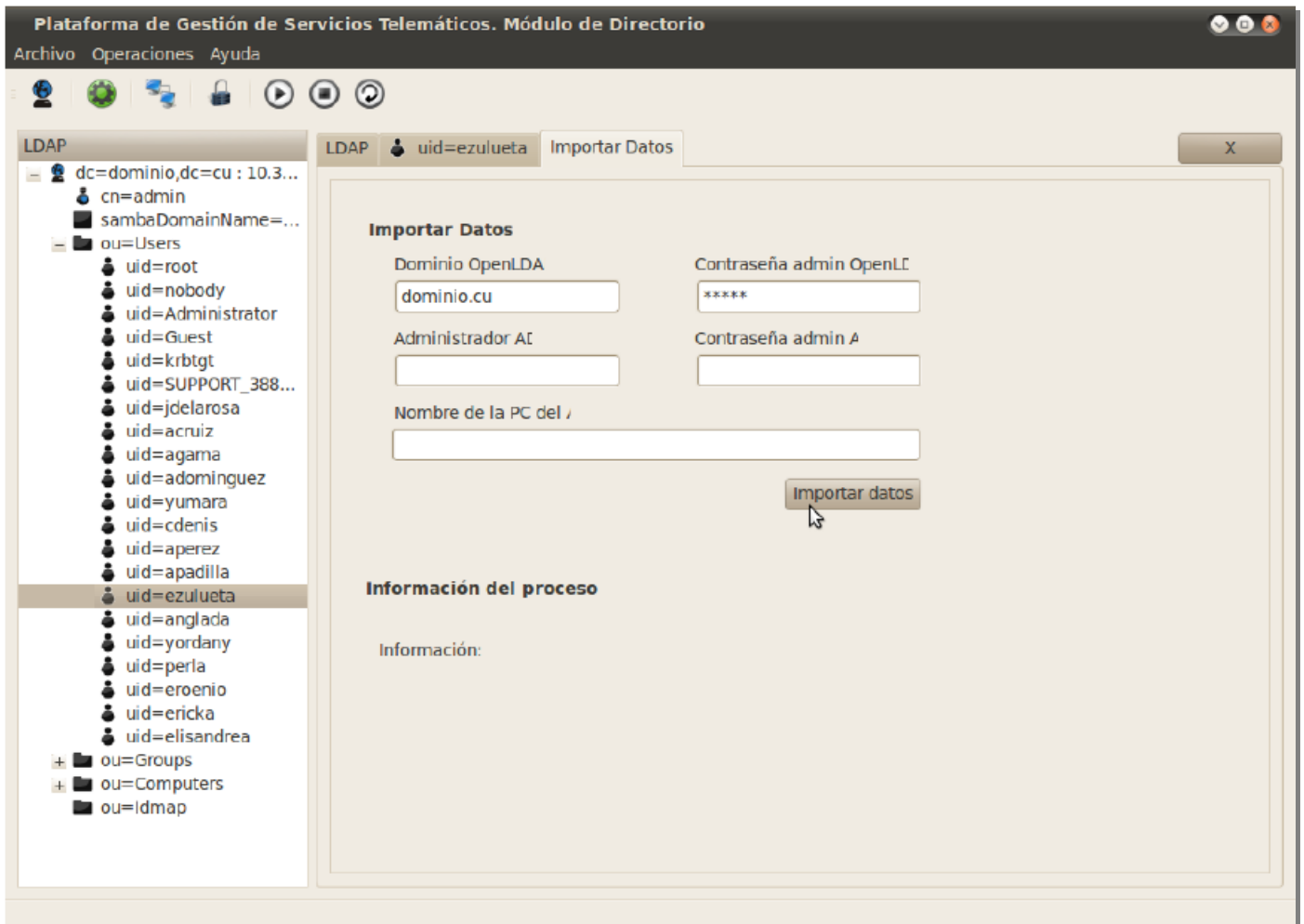


Figura A.5: Plataforma de Gestión de Servicios Telemáticos en GNU/Linux. Módulo de directorio.

Anexo B

Descripción de Casos de Uso del Sistema

Caso de Uso Gestionar configuración de conexión

Objetivo	Establecer los parámetros de conexión a los servidores LDAP.	
Actores	Usuario: (Inicia) Adiciona, modifica, lista y elimina la configuración de las conexiones.	
Resumen	<p>El CU se inicia cuando el usuario decide realizar una de las siguientes acciones:</p> <p>Adicionar configuración de conexión, introduce los datos y el sistema los guarda finalizando el CU.</p> <p>Editar configuración de conexión, el usuario selecciona el campo a modificar, inserta los datos, el sistema registra los cambios finalizando así el CU.</p> <p>Eliminar configuración de conexión, el usuario escoge la opción eliminar, el sistema muestra un diálogo de confirmación y el usuario puede aceptar o cancelar, el sistema elimina la configuración finalizando así el CU.</p>	
Complejidad	Media	
Prioridad	Crítica	
Precondiciones	El usuario ha iniciado sesión en el sistema y cuenta con los privilegios necesarios para interactuar con el módulo.	
Postcondiciones		
Flujo de eventos		
Flujo básico “Gestionar configuración de conexión”		
	Actor	Sistema
1.	Selecciona la opción “Configuración LDAP”.	
2.		Muestra el listado de las configuraciones existentes.
3.		Permite realizar varias acciones:

		<ul style="list-style-type: none"> • Adicionar una nueva configuración de conexión. Ver Sección 1: “Adicionar configuración de conexión”. • Editar configuración de conexión. Ver Sección 2: “Editar configuración de conexión”. • Eliminar configuración de conexión. Ver Sección 3: “Eliminar configuración de conexión”.
4.		Termina el Caso de Uso.
Sección 1: “Adicionar configuración de conexión”		
Flujo básico “Adicionar configuración de conexión”		
	Actor	Sistema
1.	Selecciona la opción “Adicionar”.	
2.		<p>Muestra el formulario con los campos de la configuración a adicionar:</p> <p>Pestaña “Configuración de conexión”</p> <ul style="list-style-type: none"> - Servidor - Puerto - DN Base - Usuario - Contraseña - Sufijo de cuenta <p>Pestaña “Mapeo de atributos”</p> <ul style="list-style-type: none"> - Atributo usuario - Atributo nombre - Atributo apellidos - Atributo correo <p>Pestaña “Sincronización”</p> <ul style="list-style-type: none"> - Adicionar usuarios

		<ul style="list-style-type: none"> - Eliminar usuarios - Actualizar grupos
3.	Completa los campos de la configuración y presiona el botón “Guardar”.	
4.		Valida la completitud y corrección de los datos.
5.		Adiciona la configuración.
6.		Muestra una notificación con el resultado de la operación.
7.		Termina el Caso de Uso.
Flujos alternos		
4a. Activa la opción Adicionar usuarios nuevos		
	Actor	Sistema
1.		Solicita información sobre los nuevos usuarios: <ol style="list-style-type: none"> 1. Cuota límite. 2. Política de navegación. 3. Idioma de la interfaz. 4. Marcar como activo. 5. Habilitar navegación.
2.	Completa la información solicitada.	
4a. Los datos son incorrectos o están incompletos		
	Actor	Sistema
1.		Muestra un mensaje indicando los errores existentes.
Sección 2: “Editar configuración de conexión”		
Flujo básico “Editar configuración de conexión”		
	Actor	Sistema
1.	Selecciona la opción “Editar”.	
2.		Muestra el formulario con los campos de la configuración a editar.

		<p>Pestaña “Configuración de conexión”</p> <ul style="list-style-type: none"> - Estado - Servidor - Puerto - DN Base - Usuario - Contraseña - Sufijo de cuenta <p>Pestaña “Mapeo de atributos”</p> <ul style="list-style-type: none"> - Atributo usuario - Atributo nombre - Atributo apellidos - Atributo correo <p>Pestaña “Sincronización”</p> <ul style="list-style-type: none"> - Adicionar usuarios - Eliminar usuarios - Actualizar grupos
3.	Modifica los campos de la configuración y presiona el botón “Guardar”.	
4.		Valida la completitud y corrección de los datos.
5.		Registra los cambios realizados en la configuración.
6.		Muestra un mensaje con el resultado de la operación.
7.		Termina el caso de uso.
Flujos alternos		
2a. Activa la opción Adicionar nuevos usuarios		
	Actor	Sistema
1.		<p>Solicita información sobre los nuevos usuarios:</p> <p>1. Cuota límite.</p>

		<ul style="list-style-type: none"> 2. Política de navegación. 3. Idioma de la interfaz. 4. Marcar como activo. 5. Habilitar navegación.
2.	Completa la información solicitada.	
4a. Los datos son incorrectos o están incompletos		
	Actor	Sistema
1.		Muestra un mensaje indicando los errores existentes.
Sección 3: “Eliminar configuración de conexión”		
Flujo básico “Eliminar configuración de conexión”		
	Actor	Sistema
1	Selecciona la opción “Eliminar”.	
2		Muestra un diálogo de confirmación.
3	Selecciona la opción “Aceptar”.	
4		Elimina la configuración.
5		Muestra un mensaje con el resultado de la operación.
6		Termina el Caso de Uso.
Flujos alternos		
3a. El usuario elige la opción “Cancelar”		
	Actor	Sistema
1.		Termina el Caso de Uso.

Tabla B.1: Descripción textual. CU Gestionar configuración de conexión.

Caso de Uso Sincronizar usuarios

Objetivo	Garantizar la sincronización de los usuarios de Smart Keeper con los usuarios del LDAP.	
Actores	Usuario: (Inicia) ejecuta la tarea de sincronización.	
Resumen	El CU se inicia cuando el usuario decide realizar la sincronización.	
Complejidad	Alta	
Prioridad	Crítica	
Precondiciones	El usuario ha iniciado sesión en el sistema y cuenta con los privilegios necesarios para interactuar con el módulo.	
Postcondiciones		
Flujo de eventos		
Flujo básico “Ejecutar sincronización”		
	Actor	Sistema
1	En el módulo “Usuarios LDAP” selecciona la opción “Sincronizar”.	
2		Determina la configuración de la conexión que va a emplear para obtener los datos del LDAP.
3		Contacta al servidor LDAP con la configuración de la conexión determinada.
4		Determina la configuración establecida por el usuario para la sincronización, ya sea Adicionar nuevos usuarios, eliminar usuarios inexistentes, actualizar el grupo de los usuarios o cualquier combinación de estas.
5		Ejecuta el proceso de sincronización.
6		Muestra notificación con el resultado del proceso de sincronización.
7		Termina el Caso de Uso.
Flujos alternos		

3a. No se pudo contactar al servidor LDAP		
	Actor	Sistema
3		Muestra un mensaje informando los errores ocurridos.
4		Termina el Caso de Uso.

Tabla B.2: Descripción textual. CU Sincronizar usuarios.

Anexo C

Diagramas de clases del diseño

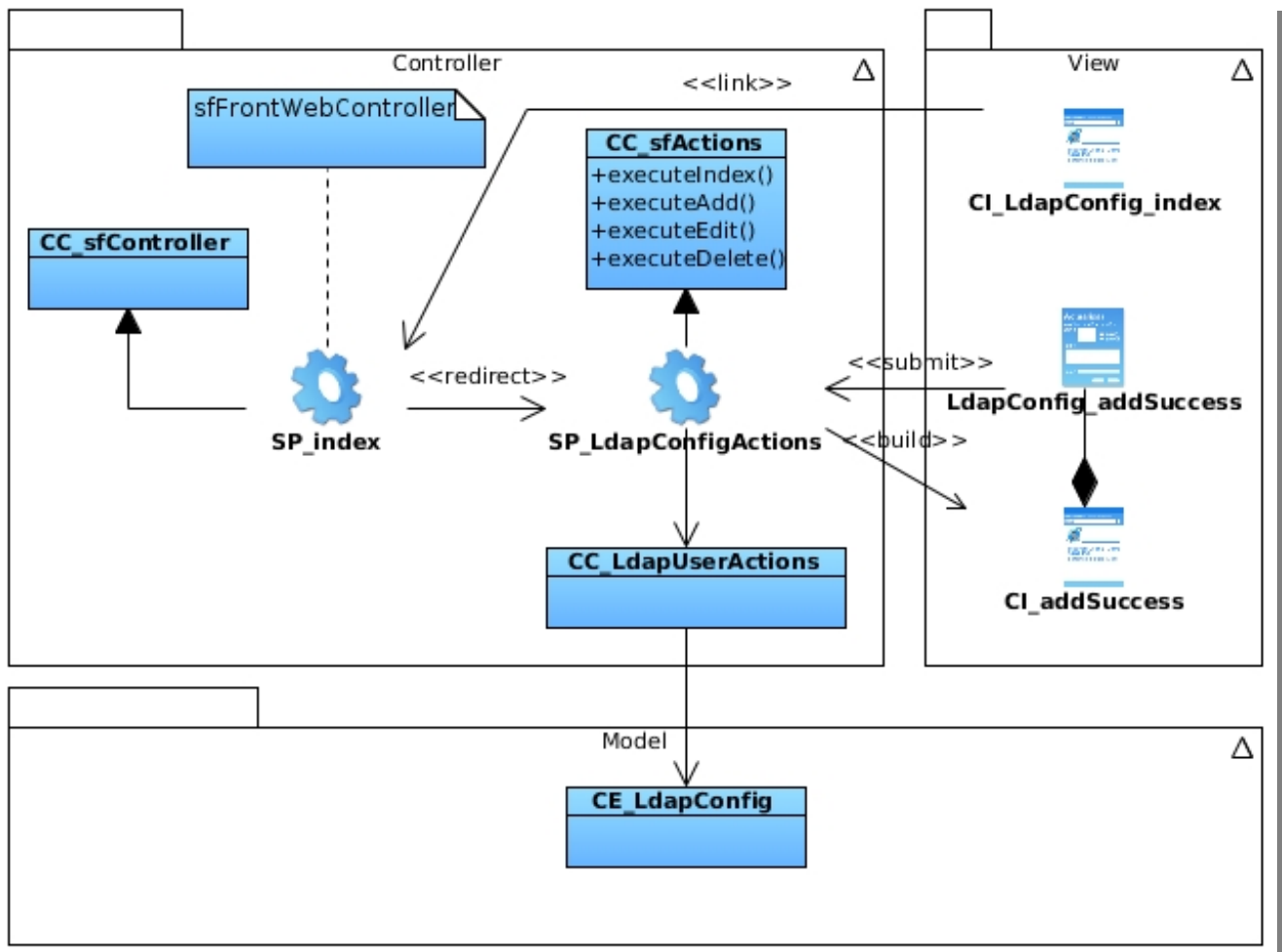


Figura C.1: Diagrama de clases del diseño. CU Gestionar configuración de conexión. Sección Adicionar configuración de conexión.

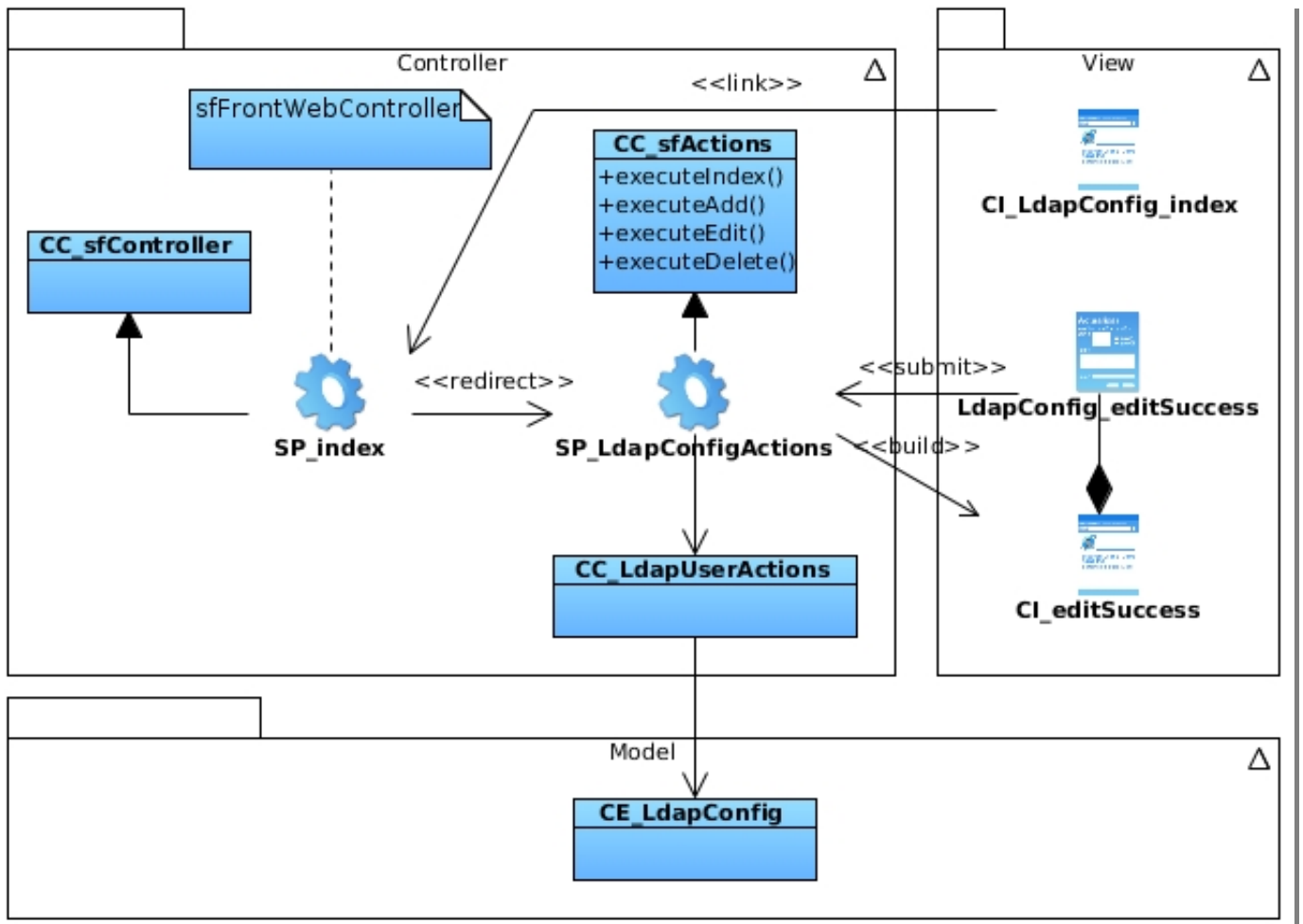


Figura C.2: Diagrama de clases del diseño. CU Gestionar configuración de conexión. Sección Editar configuración de conexión.

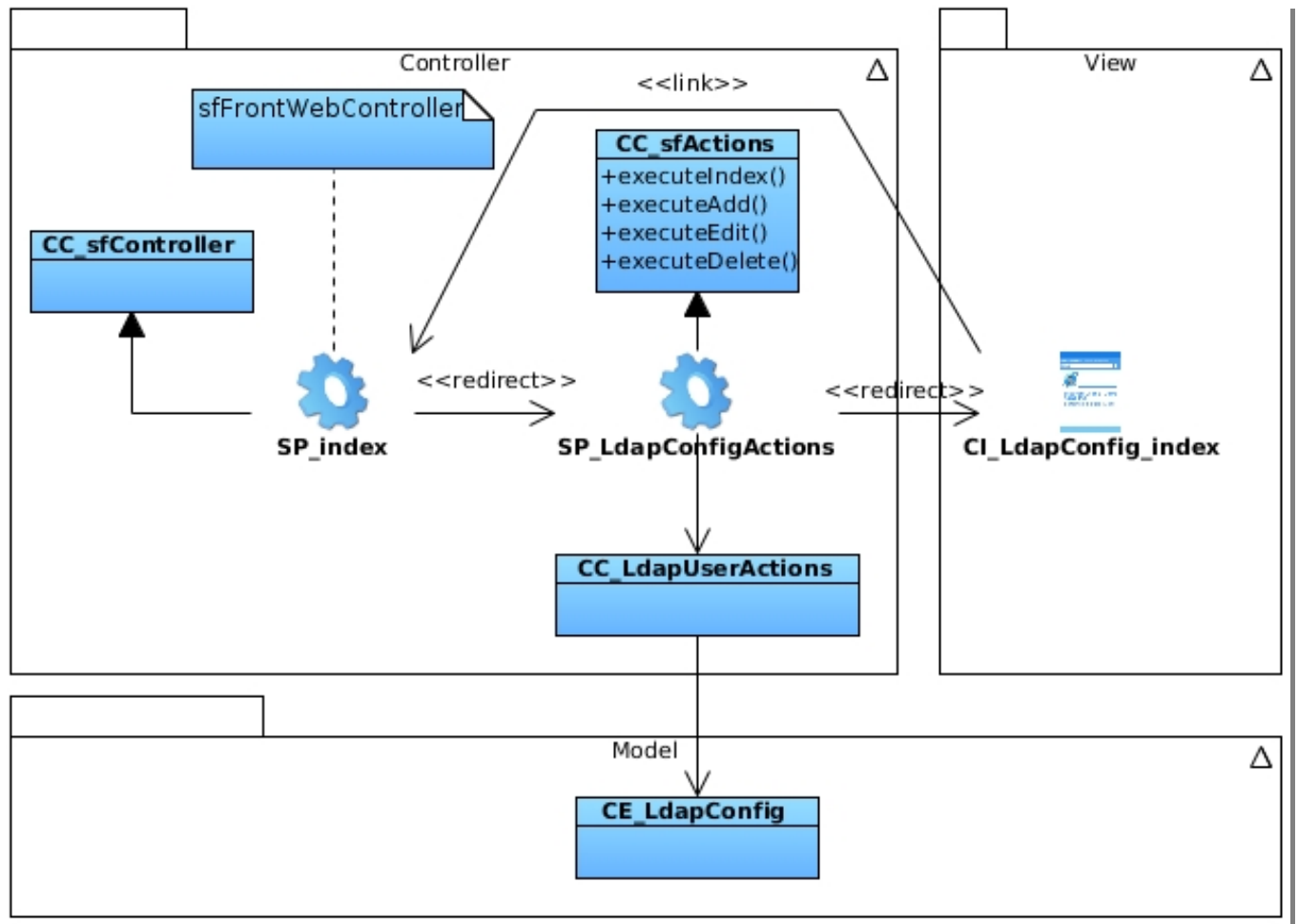


Figura C.3: Diagrama de clases del diseño. CU Gestionar configuración de conexión. Sección Eliminar configuración de conexión.

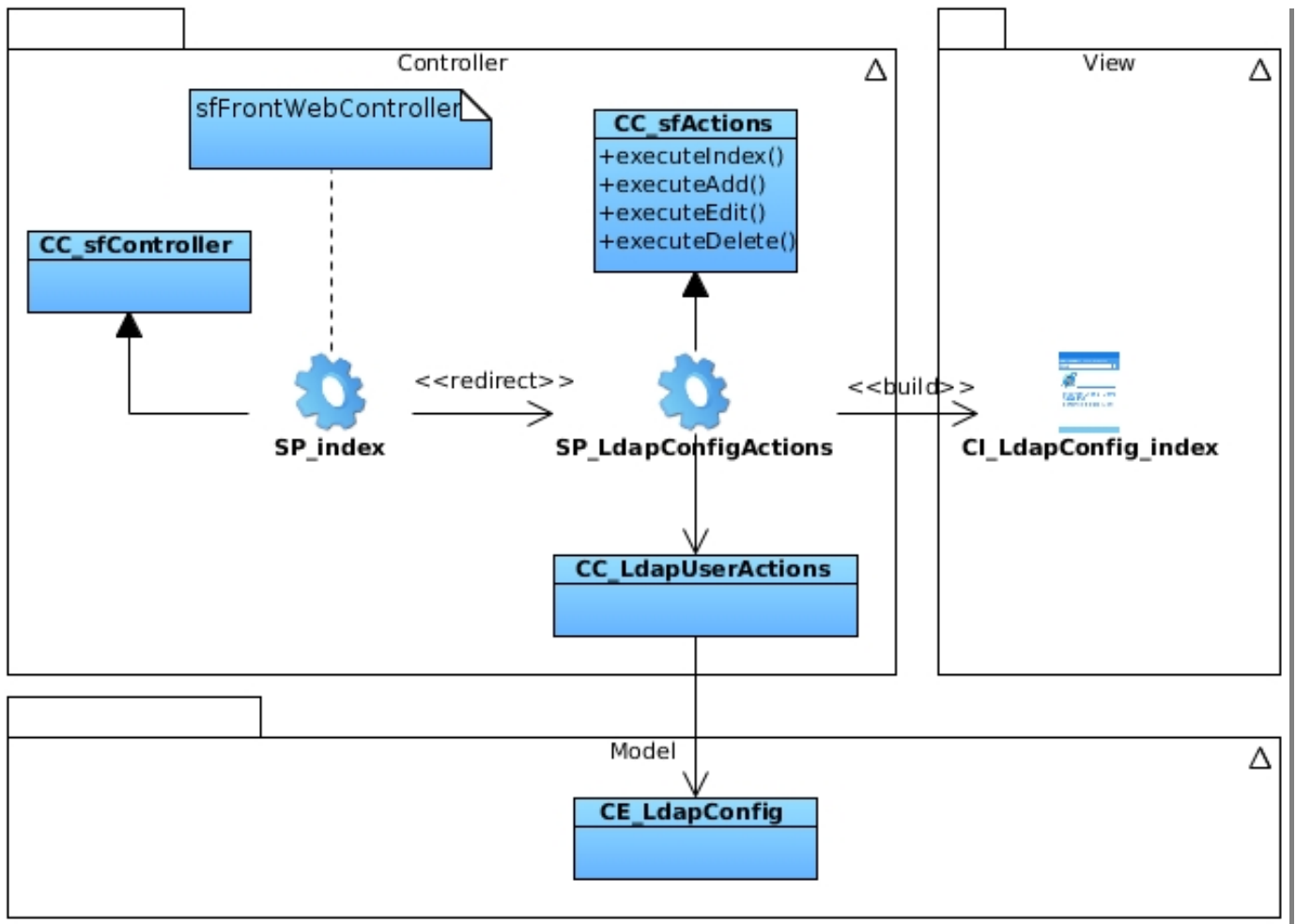


Figura C.4: Diagrama de clases del diseño. CU Gestionar configuración de conexión. Sección Listar configuración de conexión.

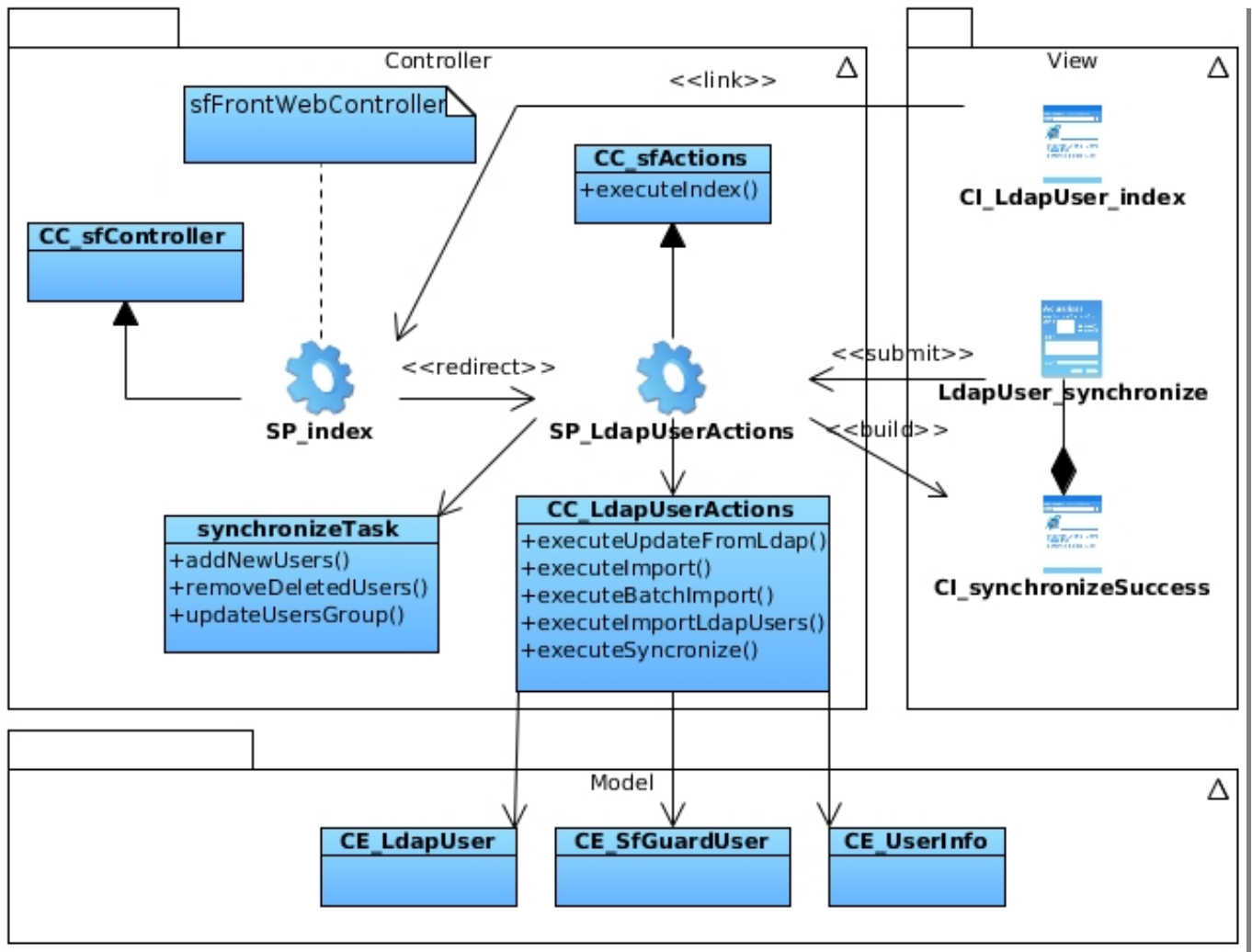


Figura C.5: Diagrama de clases del diseño. CU Sincronizar usuarios. Sección Ejecutar sincronización.

Anexo D

Diagramas de interacción

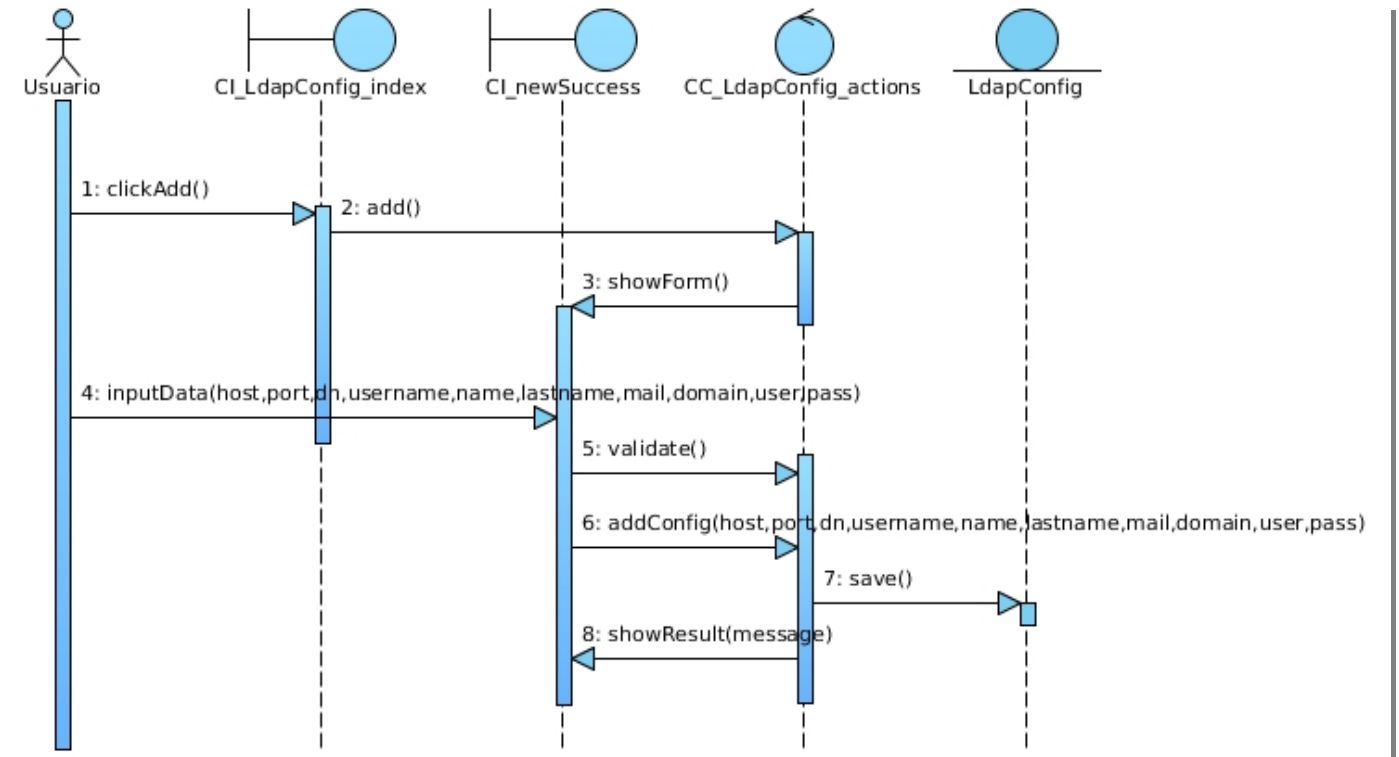


Figura D.1: Diagrama de interacción. CU Gestionar configuración de conexión. Sección Adicionar configuración de conexión.

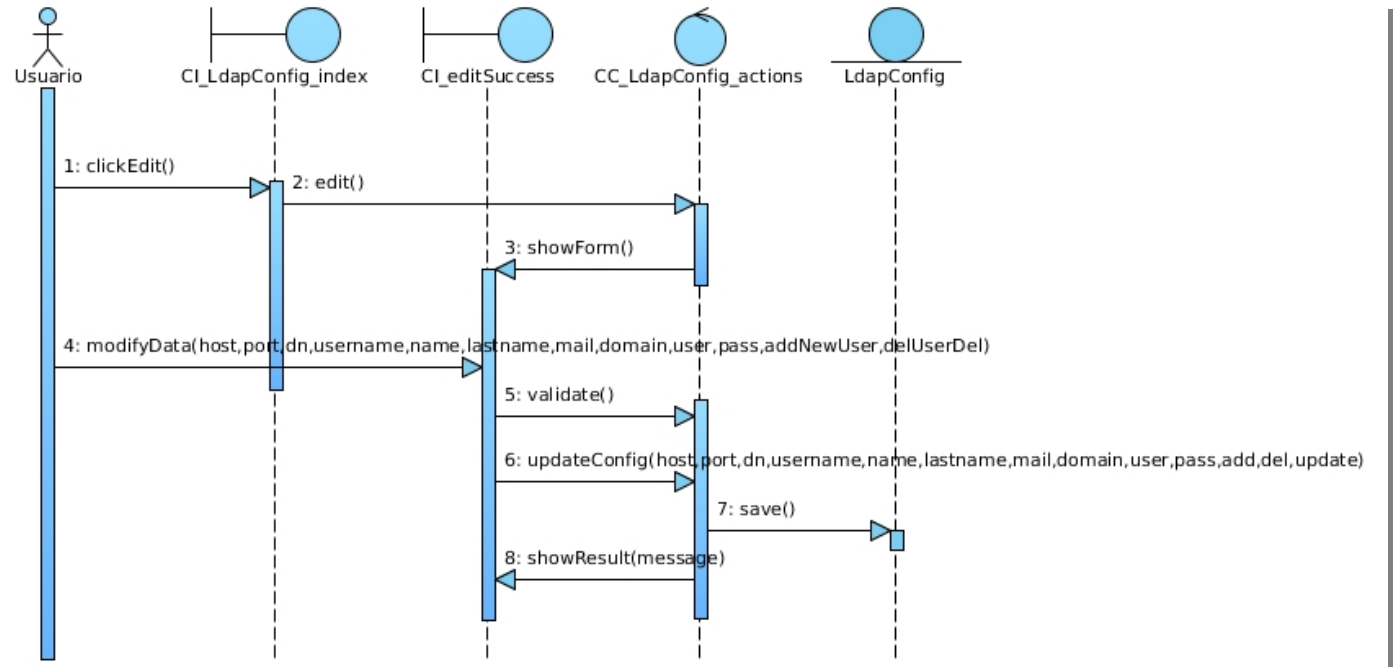


Figura D.2: Diagrama de interacción. CU Gestionar configuración de conexión. Sección Editar configuración de conexión.

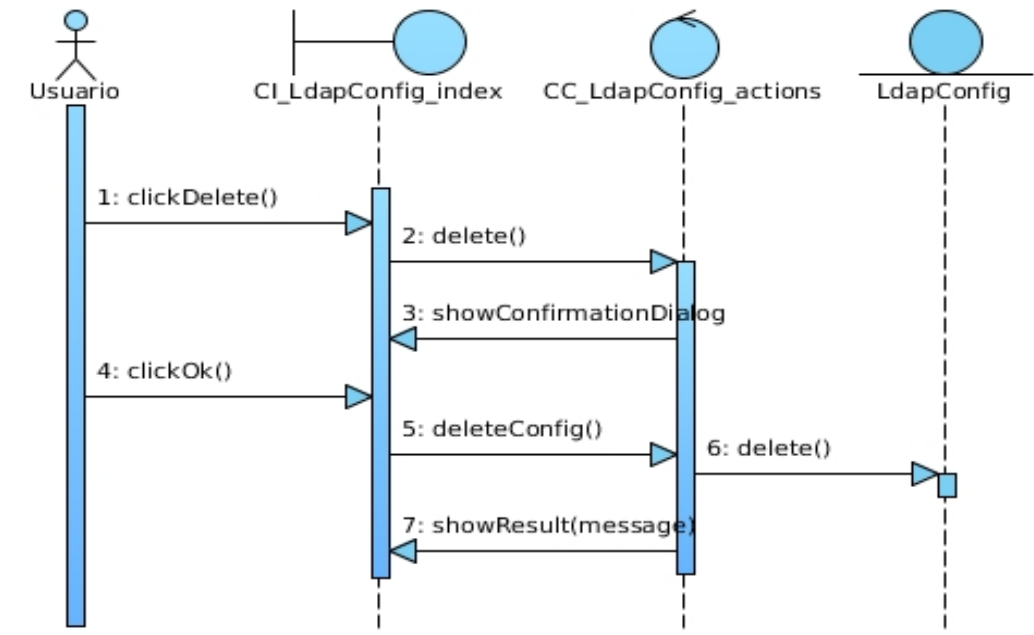


Figura D.3: Diagrama de interacción. CU Gestionar configuración de conexión. Sección Eliminar configuración de conexión.

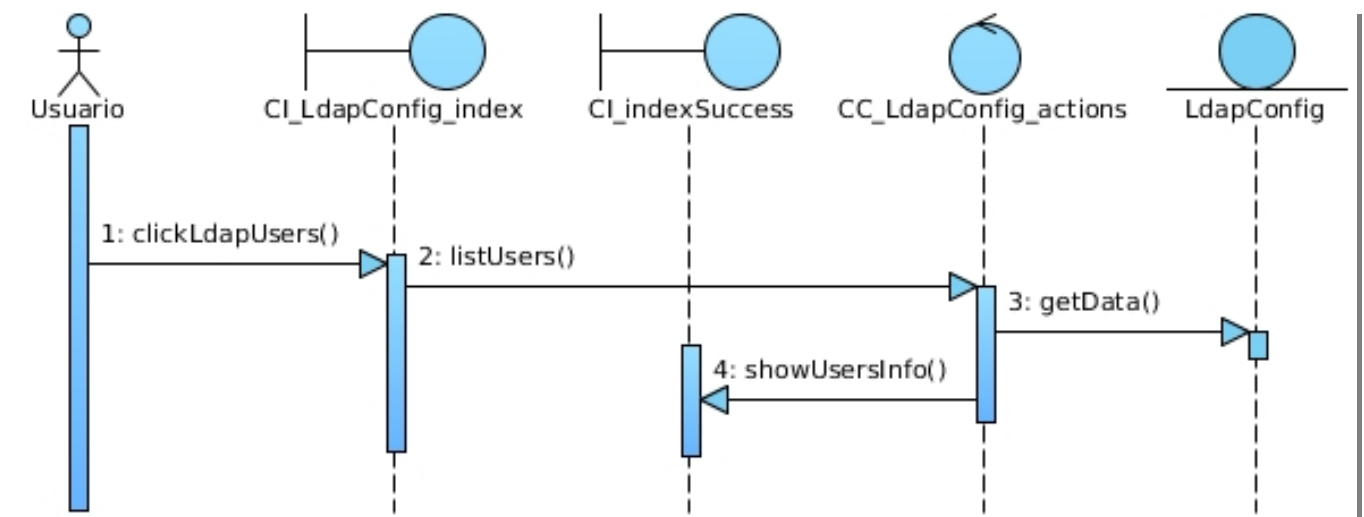


Figura D.4: Diagrama de interacción. CU Gestionar configuración de conexión. Sección Listar configuración de conexión.

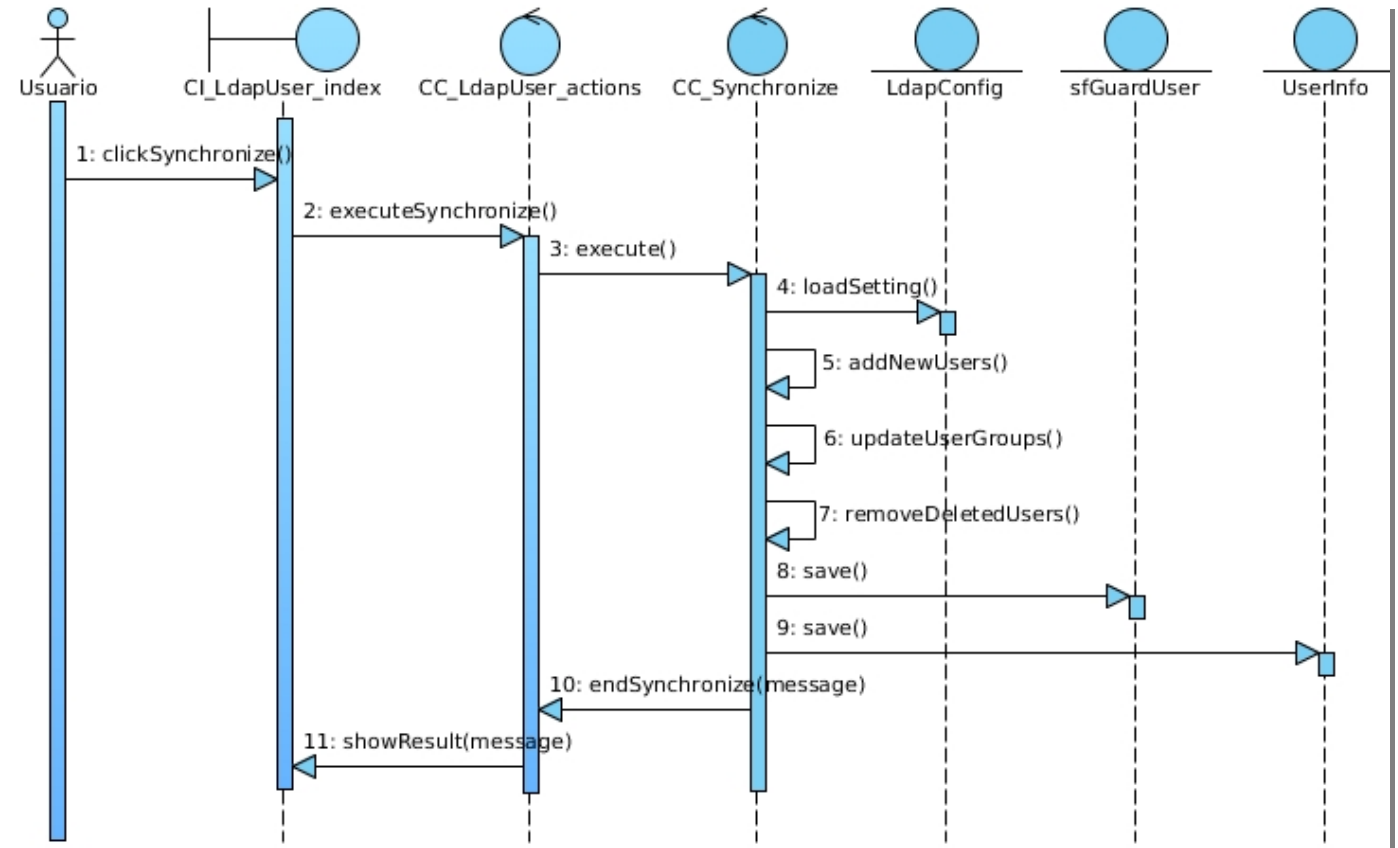


Figura D.5: Diagrama de interacción. CU Sincronizar usuarios. Sección Ejecutar sincronización.