

**UNIVERSIDAD DE SANTIAGO DE CHILE**  
**FACULTAD DE INGENIERÍA**  
**Departamento de Ingeniería Informática**



**Prototipo de sistema multiprotocolo moderno para la administración, autenticación y autorización de clientes en el Departamento de Ingeniería Informática**

**Roberto Ernesto Lillo Toloza**

Profesor guía: Cristóbal Acosta Jurado

Trabajo de titulación presentado en conformidad a los requisitos para obtener el título de Ingeniero de Ejecución en Computación e Informática

Santiago – Chile

2021

© Roberto Ernesto Lillo Toloza , 2021



• Algunos derechos reservados. Esta obra está bajo una Licencia Creative Commons Atribución-CompartirIgual-Chile 3.0. Sus condiciones de uso pueden ser revisadas en:  
<http://creativecommons.org/licenses/by-sa/3.0/cl/>.

## RESUMEN

El Departamento de Ingeniería Informática de la Universidad de Santiago de Chile administra las cuentas de usuarios y el *hardware* mediante un servicio de directorio que es *software* libre. Esta implementación no está del todo actualizada y su último mantenimiento fue hace ya varios años, de esto nacen situaciones problemáticas como tener dos cuentas universitarias con distinto nombre de usuario y contraseña, ingreso manual de datos que podrían estar automatizados y poca integración con sistemas y aplicaciones modernas. Por otro lado, los servicios ofrecidos por el *software* son insuficientes para las necesidades actuales y deja en duda si esta es una herramienta que se encuentre acorde al estado del arte.

Para solucionar este problema se realizó una investigación con el fin de comparar distintos *sistemas de directorio* y también nuevos *software* que sean capaces de integrar estas funciones junto a nuevos protocolos de autenticación y autorización, con el fin de diseñar un sistema centralizado donde cualquier integrante del departamento pueda utilizar sólo un nombre de usuario para ingresar a los distintos sistemas, ya sea plataformas Windows y Linux, así como servicios Web, entre otros.

**Palabras Claves:** *Directory Service, Authentication, Authorization, LDAP, SAMBA, Active Directory, Gluu, Keycloak, User Federation.*

*Dedicado a mis padres Ernesto y Marcia quienes siempre se han esforzado por entregarme las herramientas y oportunidades para formarme, por el amor, cariño y particularmente por darme la libertad que me permitió experimentar el mundo y aprender de mis propias acciones. A mis padrinos Hermes y Juanita quienes me dieron la oportunidad de estudiar en Santiago al recibirmee en su hogar, siempre preocupados por mi bienestar en lo que en algún momento fue una ciudad completamente nueva para mi. Finalmente, dedicado al resto de mi familia y amigos quienes me han acompañado durante toda esta experiencia y en particular a mi grupo de amigos "Freecs", donde se encuentran amistades importantes que incluso se remontan a etapas muy tempranas de mi infancia, con los que comparto diariamente y que considero parte de mi familia.*

## **AGRADECIMIENTOS**

Quiero agradecer a la Universidad de Santiago de Chile y a la Facultad de Ingeniería por la formación que me brindaron durante estos años, también a todos los profesores y ayudantes por el valioso conocimiento que me entregaron especialmente durante los difíciles primeros años de la carrera. Quiero igualmente incluir al personal de funcionarios con los que interactué en diversos momentos, agradezco su generosidad y disposición para solucionar los problemas que en algún momento tuve.

También agradezco al Departamento de Ingeniería Informática y a sus docentes por la formación en esta área tecnológica que es tan importante al día de hoy, permitiendo explorar los distintos terrenos en los que es posible ejercer como profesional. El DIINF es un lugar muy agradable en el cual estar y quiero agradecer también a todas las áreas que hacen esto posible como docencia, operaciones, secretaría, mayordomía, funcionarios del aseo e incluso a los mismos estudiantes que aportan en el centro de alumnos.

Finalmente, quiero agradecer a mi profesor guía Cristóbal Acosta, por ayudarme a cumplir mi deseo de dejar algo como agradecimiento al departamento por el conocimiento que me fue entregado durante estos años. Su paciencia, expertiz y forma de guiar el avance me permitieron completar este proyecto a tiempo y dejar en sus manos un sistema capaz de ayudar al área de TI y operaciones.

# TABLA DE CONTENIDO

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Antecedentes y motivación . . . . .	1
1.2	Descripción del problema . . . . .	1
1.3	Solución propuesta . . . . .	3
1.4	Objetivos y alcance del proyecto . . . . .	4
1.4.1	Objetivo general . . . . .	4
1.4.2	Objetivos específicos . . . . .	4
1.4.3	Alcances . . . . .	4
1.5	Metodología y herramientas utilizadas . . . . .	5
1.5.1	Metodología . . . . .	5
1.5.2	Herramientas de desarrollo . . . . .	7
1.6	Organización del documento . . . . .	7
<b>2</b>	<b>Marco Teórico</b>	<b>9</b>
2.1	Autenticación . . . . .	9
2.1.1	Modelos de autenticación . . . . .	10
2.1.2	Otras formas de reforzar la seguridad . . . . .	11
2.2	Autorización . . . . .	11
2.2.1	Modelos de autorización . . . . .	12
2.3	Estándares y protocolos de autenticación y autorización . . . . .	13
2.3.1	<i>Single Sign-On (SSO)</i> . . . . .	14
2.3.2	<i>Security Assertion Markup Language (SAML)</i> . . . . .	15
2.3.3	OAuth . . . . .	16
2.3.4	<i>OpenID Connect (OIDC)</i> . . . . .	18
2.4	Servicios de directorio . . . . .	19
2.4.1	Estructura de directorios LDAP . . . . .	19
2.4.2	Comunicación mediante el protocolo LDAP . . . . .	23
2.4.3	Algunos servicios de directorio que implementan LDAP . . . . .	24
2.5	Administración de identidad, de acceso y gobernanza . . . . .	25
2.6	Tecnologías de desarrollo . . . . .	26
2.6.1	Ansible . . . . .	26
2.6.2	Docker . . . . .	27
<b>3</b>	<b>Desarrollo</b>	<b>29</b>
3.1	Investigación, comparación y evaluación de productos de <i>software</i> . . . . .	29
3.1.1	Criterios de comparación . . . . .	29
3.1.2	Productos de <i>software</i> escogidos . . . . .	31
3.1.3	Proceso de evaluación . . . . .	31
3.2	Rediseño del directorio y formato de credenciales . . . . .	33
3.2.1	Árbol del directorio . . . . .	34
3.2.2	Formato de credenciales . . . . .	36
3.3	Integración de interfaz Web . . . . .	36
3.3.1	phpLDAPadmin . . . . .	37
3.3.2	Gluu y oxTrust . . . . .	38
3.3.3	Keycloak, consola de administración y de usuarios . . . . .	38
3.3.4	Localización y <i>Branding</i> . . . . .	40
3.4	Integración de protocolos . . . . .	42
3.4.1	Gluu y <i>Cache Refresh</i> . . . . .	42
3.4.2	Keycloak y <i>User Storage Federation</i> . . . . .	44
3.5	Delegación de autenticación . . . . .	45

3.5.1	Servicio de delegación de Google . . . . .	45
3.5.2	Integración de Google al sistema centralizado . . . . .	46
3.6	Despliegue del sistema mediante Docker . . . . .	48
3.6.1	Contenedor de Samba . . . . .	48
3.6.2	Contenedor de Keycloak . . . . .	48
3.7	Automatización de instalación y configuración mediante Ansible . . . . .	49
3.7.1	Requisitos previos establecidos . . . . .	50
3.7.2	Instalación de herramientas . . . . .	50
3.7.3	Despliegue de contenedores . . . . .	51
3.7.4	Configuración de Keycloak . . . . .	51
<b>4</b>	<b>Conclusiones</b>	<b>53</b>
4.1	Resultados . . . . .	53
4.2	Objetivos . . . . .	54
4.3	Trabajos futuros . . . . .	56
<b>Glosario</b>		<b>57</b>
<b>Referencias bibliográficas</b>		<b>58</b>
<b>Anexos</b>		<b>61</b>
<b>A</b>	<b>Evaluación vigencia y documentación de software</b>	<b>61</b>
A.1	Criterios para determinar la vigencia . . . . .	61
A.2	Criterios para evaluar la documentación . . . . .	61
<b>B</b>	<b>Creación del tema DIINF para la interfaz de cuentas de usuario de Keycloak</b>	<b>63</b>
<b>C</b>	<b>Localización al español de la interfaz de cuentas de usuario de Keycloak</b>	<b>66</b>
<b>D</b>	<b>Configuración del <i>User Storage Federation</i> en Keycloak</b>	<b>67</b>
<b>E</b>	<b>Obtención de credenciales OAuth en <i>Google Cloud Platform</i></b>	<b>69</b>

## ÍNDICE DE TABLAS

Tabla 2.1 Comparación de funciones autenticación vs autorización . . . . .	11
Tabla 3.1 Evaluación de características base . . . . .	31
Tabla 3.2 Evaluación de características de deploy . . . . .	32
Tabla 3.3 Evaluación de características de interacción . . . . .	32
Tabla 3.4 Evaluación de características relacionadas a protocolos . . . . .	33
Tabla 3.5 Resultado final de la evaluación . . . . .	33
Tabla A.1 Evaluación de vigencia de software . . . . .	61
Tabla A.2 Evaluación de documentación de software . . . . .	62

# ÍNDICE DE ILUSTRACIONES

Figura 1.1	Situación apreciable de la interacción de los <i>software</i> en el DIINF . . . . .	3
Figura 1.2	Visión general del sistema propuesto . . . . .	3
Figura 1.3	Ejemplo de un tablero Kanban . . . . .	6
Figura 1.4	Tablero Kanban en Trello . . . . .	6
Figura 2.1	Diagrama de una infraestructura de red de ejemplo . . . . .	9
Figura 2.2	Flujo en el que ocurre el procedimiento de <i>Single Sign-On</i> . . . . .	14
Figura 2.3	Flujo de autenticación mediante el estándar SAML . . . . .	16
Figura 2.4	Flujo de autorización mediante el protocolo OAuth . . . . .	17
Figura 2.5	Flujo de autenticación y autorización mediante los protocolos OIDC y OAuth	18
Figura 2.6	Organización de dos usuarios mediante un grupo, utilizando <i>Common Name</i>	20
Figura 2.7	Organización de tres usuarios mediante dos grupos y una unidad organizativa	20
Figura 2.8	Organización de cinco usuarios mediante cuatro grupos, dos unidades organizativas y un dominio . . . . .	21
Figura 2.9	Construcción de un <i>Distinguished Name</i> a partir del <i>Common Name</i> , <i>Organizational Unit</i> y <i>Domain</i> . . . . .	22
Figura 2.10	Enlace de federación de confianza entre dos árboles de directorio . . . . .	23
Figura 2.11	Secuencia en la que se realiza una conexión con un servidor LDAP . . . . .	23
Figura 2.12	Diagrama de componentes de servicios de identidad . . . . .	26
Figura 2.13	Diagrama de funcionamiento de Ansible . . . . .	27
Figura 2.14	Comparación entre el despliegue de aplicaciones mediante contenedores y mediante máquinas virtuales . . . . .	28
Figura 3.1	Diseño inicial del árbol del directorio . . . . .	34
Figura 3.2	Diseño final del árbol del directorio. . . . .	35
Figura 3.3	Interfaz inicial de phpLDAPadmin . . . . .	37
Figura 3.4	Interfaz inicial de Gluu . . . . .	38
Figura 3.5	Interfaz inicial de Keycloak . . . . .	39
Figura 3.6	Interfaz de cuentas de usuario de Keycloak . . . . .	40
Figura 3.7	Interfaz de Keycloak en español . . . . .	40
Figura 3.8	Vista de cuenta de usuario con el tema DIINF aplicado en Keycloak . . . . .	41
Figura 3.9	Sincronización de usuarios mediante <i>Cache Refresh</i> en Gluu . . . . .	43
Figura 3.10	Flujo de autenticación de un usuario sincronizado por <i>Cache Refresh</i> en Gluu	43
Figura 3.11	Flujo de autenticación de un usuario sincronizado por <i>User Storage Federation</i> en Keycloak . . . . .	44
Figura 3.12	Flujo de inicio de sesión en una aplicación Web mediante delegación a Google . . . . .	45
Figura 3.13	Pestaña de configuración de proveedores de identidad en Keycloak . . . . .	46
Figura 3.14	Encadenamiento de proveedores de identidad mediante Keycloak y Google	47
Figura 3.15	Diagrama de contenedores desplegados del sistema centralizado . . . . .	49
Figura 3.16	Despliegue completo al finalizar el <i>playbook</i> de Ansible . . . . .	52
Figura B.1	Estructura de la carpeta del tema DIINF . . . . .	63
Figura B.2	Vista de inicio de sesión para integrantes del DIINF en Keycloak . . . . .	63
Figura B.3	Contenido del archivo <i>style.css</i> , de la interfaz de cuentas de usuario . . . . .	64
Figura B.4	Contenido del archivo <i>theme.properties</i> , de la interfaz de cuentas de usuario	64
Figura B.5	Vista de cuenta de usuario para integrantes del DIINF en Keycloak . . . . .	65
Figura C.1	Estructura de la carpeta "messages" en el tema base . . . . .	66
Figura C.2	Extracto del archivo "messages_es.properties" del tema DIINF . . . . .	66

Figura D.1	Pestaña de configuración de <i>User Federation</i> en Keycloak . . . . .	67
Figura D.2	Usuario federado por Keycloak desde un servidor de Samba AD . . . . .	68
Figura E.1	Pantalla de consentimiento de Oauth en <i>Google Cloud Platform</i> . . . . .	69
Figura E.2	Configuración de los <i>scopes</i> para la aplicación registrada en <i>Google Cloud Platform</i> . . . . .	70
Figura E.3	Configuración de rango de usuarios permitidos en la aplicación . . . . .	70
Figura E.4	Pantalla de configuración para la creación de credenciales en <i>Google Cloud Platform</i> . . . . .	71
Figura E.5	Ubicación de URI y credenciales de OAuth en Keycloak . . . . .	71

# CAPÍTULO 1. INTRODUCCIÓN

## 1.1 ANTECEDENTES Y MOTIVACIÓN

La administración de una red con una amplia cantidad de usuarios no es una tarea simple, pequeñas y grandes empresas se ven en la situación de establecer reglas para determinar quien tiene acceso a los recursos disponibles ya sea por un tema de facilitar el uso de los mismos o por tener una capa de seguridad sobre los datos importantes que pueden estar guardados.

Para esto suelen implementarse herramientas de *software* de manejo de directorios conocidos como **Servicios de Directorio**, Alpern & Shimonski (2010) los describen como servicios usados para guardar, retirar y administrar información acerca objetos como cuentas de usuario, computadores, servidores y otros recursos, lo que posteriormente puede ser administrado de manera centralizada en este mismo *software*, por ejemplo, estableciendo distintas políticas de acceso, agrupaciones de usuarios dependiendo del área de trabajo, cuales archivos son guardados en cada cuenta de usuario, entre otras opciones.

No obstante, en la actualidad la administración de recursos no se limita a sólo una red interna ya que se pueden tener distintas aplicaciones o servicios alojados externamente en servicios *cloud* o incluso pueden ser aplicaciones de terceros que se desean integrar al sistema. Para esto es necesario que el *software* tenga la capacidad de ofrecer protocolos estándar los cuales puedan ser utilizados en dichas situaciones.

## 1.2 DESCRIPCIÓN DEL PROBLEMA

Actualmente el Departamento de Ingeniería Informática utiliza un servicio de directorio para sus necesidades de manejo de usuarios y recursos, en este se realiza la creación y administración de cuentas de usuario, como también el manejo de otros instrumentos de *hardware* como los computadores de los laboratorios de uso general y los servidores. Para esta labor se utiliza el *software* **Samba**<sup>1</sup>, desarrollado en 1992 y que al día de hoy aún sigue en desarrollo en su cuarta versión.

La creación de cada nueva cuenta de usuario se maneja de la siguiente forma, primero esta debe ser solicitada personalmente por quien la necesita, los estudiantes deben acercarse a la ventanilla de operaciones y llenar un formulario físico con sus datos personales, acompañándolo con el certificado de su matrícula al día, mientras los académicos al ser una

---

<sup>1</sup><https://www.samba.org/>

menor cantidad y no requerir la matrícula se pueden contactar directamente por correo con la sala de operaciones para la creación de la cuenta. Posterior a la recepción de la solicitud se procede a crear la cuenta, asignando un nombre de usuario en base a la inicial del nombre y el apellido paterno (en caso de ya existir el usuario se utilizan otras combinaciones), dejando una contraseña arbitraria que se cambia en el primer uso. Otro paso extra para los estudiantes, es activar nuevamente su cuenta al inicio de cada semestre, acercándose otra vez a la ventanilla de operaciones para mostrar su certificado de matrícula y mencionando su nombre de usuario. Lo que ofrece la cuenta depende del tipo de usuario, en el caso de los estudiantes se otorga acceso al uso de los laboratorios de computación y a una cantidad fija de impresiones semestrales, por otro lado, los académicos tienen derecho a impresiones ilimitadas y si lo solicitan, acceso a servidores para trabajos de investigación o similares. Observando esta situación se puede apreciar un primer problema, el donde se genera una situación en la que los integrantes del DIINF poseen dos cuentas institucionales para la universidad, cada una con su propio nombre de usuario y contraseña.

En relación al servicio de directorio que se utiliza, Samba *Active Directory* cubre las necesidades mínimas que tiene el departamento en este momento, permitiendo el manejo de estaciones de trabajo Windows 10 y los servidores Linux, pero dejando fuera a las estaciones de trabajo Linux, en donde los alumnos no pueden ingresar al sistema mediante la cuenta de usuario, estando obligados a utilizar una cuenta de invitado que todos comparten. En relación a la administración, todo el manejo de cuentas de usuario se realiza de forma manual, ya sea la creación o activación de cuentas como también la recuperación de contraseñas, la forma de hacerlo es mediante **Interfaz de Línea de Comandos/Command Line Interface** (CLI) sin ninguna interfaz gráfica. En otro aspecto técnico, si bien la versión de Samba que se utiliza en el departamento es la cuarta, la instalación no está al día con los últimos *releases*, por lo que se puede dar el caso de que el sistema posea alguna falla de funcionamiento o seguridad que ya ha sido reparada en alguna nueva actualización.

Un último problema apreciable es la situación de las aplicaciones creadas para el departamento, Samba no ofrece un punto centralizado de autenticación para aplicaciones Web o móviles, por lo que estas necesitan crear su propio sistema de autenticación o buscar formas de delegarla a terceros. En primeras instancias se encontró la posibilidad de hacer un *bypass* al software Samba, conectándose directamente a su base de datos para así obtener información con respecto a los usuarios, esto puede resultar en problemas de seguridad al exponer directamente la información sin un intermediario. Actualmente algunas aplicaciones utilizan a Google como sistema de identificación de usuarios, así aprovechando el correo universitario pero perdiendo control centralizado sobre quienes pueden acceder a la aplicación ya que cada aplicación debe manejar esto por separado.

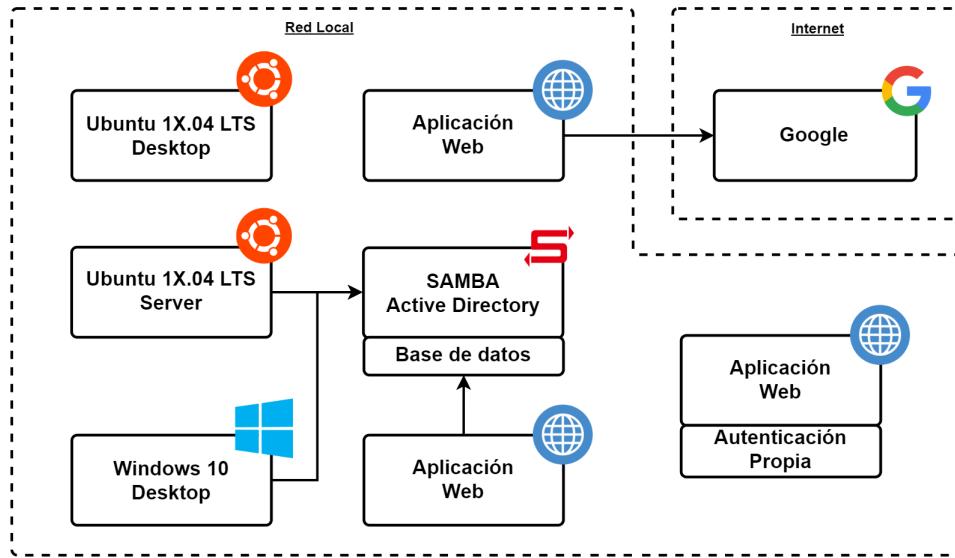


Figura 1.1: Situación apreciable de la interacción de los *software* en el DIINF.  
Fuente: Elaboración propia (2021).

### 1.3 SOLUCIÓN PROPUESTA

Como respuesta a este problema, se contempla el desarrollo de un prototipo para un sistema de directorio actualizado. Este debe ser capaz de responder las problemáticas presentadas anteriormente, principalmente las relacionadas a la actualización del sistema y al uso de protocolos modernos enfocándose principalmente en la identificación de usuarios, con el fin de ofrecer un sistema centralizado capaz de reconocer la identidad de los usuarios en distintas aplicaciones y ofreciendo una forma de determinar a qué recursos tiene acceso. La variedad de protocolos que ofrece este sistema, cubriendo sistemas Windows 10, Linux, Web y móviles lleva a la idea de determinarlo como multiprotocolo.

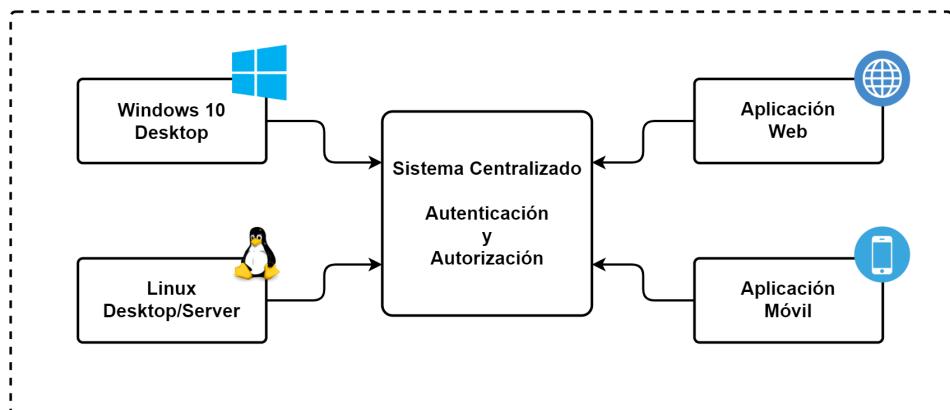


Figura 1.2: Visión general del sistema propuesto.  
Fuente: Elaboración propia (2021).

## **1.4 OBJETIVOS Y ALCANCE DEL PROYECTO**

### **1.4.1 Objetivo general**

Desarrollar un prototipo de sistema multiprotocolo moderno para la administración, autenticación y autorización de clientes en el Departamento de Ingeniería Informática.

### **1.4.2 Objetivos específicos**

1. Comparar nuevas tecnologías de sistemas de directorio y distinguir qué funciones únicas ofrecen a diferencia de lo que puede hacer Samba.
2. Rediseño del árbol de directorio e implementación de nuevo formato para credenciales.
3. Integrar una interfaz Web para reemplazar la administración por línea de comandos.
4. Integrar el protocolo OAuth2 al sistema, para centralizar la autenticación de usuarios.
5. Implementar un sistema para delegar la autenticación de los usuarios a Google mediante el correo USACH, dejando el lado de autorización al servicio de directorio del DIINF.
6. Empaquetar los servicios en contenedores utilizando Docker.
7. Automatizar la instalación de los servicios mediante Ansible.

### **1.4.3 Alcances**

Dentro de los alcances se contempla el rediseño del árbol del directorio, con grupos específicos para las distintas unidades del DIINF (estudiantes, académicos, administrativos, entre otros). También darle un nuevo formato a las credenciales de usuario basándose en el correo institucional.

La integración de protocolos modernos para Web y móvil se hará mediante otro producto de *software* que sea compatible con el protocolo LDAP, con el fin de tener integración mediante estos sistemas en caso de ser necesario. Así mismo, la integración de la interfaz Web también corresponde a otro *software* que sea compatible.

Finalmente, se tomó la decisión de que el sistema centralizado no debe superar el máximo de cinco productos de *software* distintos. Esto es una preferencia personal para evitar mayores problemas de sincronización y configuración entre los mismos.

## 1.5 METODOLOGÍA Y HERRAMIENTAS UTILIZADAS

### 1.5.1 Metodología

La metodología Kanban comenzó a ser considerada parte de las metodologías ágiles alrededor del 2005, gracias a que permite desarrollos graduales y evolutivos de sistemas de *software* (Anderson, 2010). Generalmente los tableros Kanban son utilizados por otras metodologías ágiles (como por ejemplo Scrum), pero no se debe confundir esto con la metodología completa.

Se escogió esta como metodología para el desarrollo del proyecto gracias al constante uso de metodologías ágiles en las distintas asignaturas del Departamento de Ingeniería Informática, por lo que se tiene un buen entendimiento del flujo de trabajo que se necesita realizar. Gracias a la metodología Kanban, el desarrollo del sistema se podrá ir haciendo de manera gradual a medida que se va necesitando cumplir con las distintas funcionalidades previamente establecidas en los objetivos específicos. Además, permite que ciertas funciones más básicas del sistema se puedan implementar de manera rápida y ser probadas de inmediato antes de empezar con tareas de mayor complejidad. Según el mismo Anderson (2020), el método Kanban posee las siguientes seis características principales.

- **Visualización**, donde se debe observar qué es lo que se hace y cuál es el flujo de trabajo que se necesita realizar.
- **Limitación de tareas**, estableciendo la cantidad de trabajo que puede ingresar a cada etapa con el fin de no saturar una o generar cuellos de botella.
- **Seguimiento y gestión del flujo**, supervisar y medir el trabajo realizado en base al tiempo utilizado para medir la eficacia. Indicadores y políticas claras, usar colores para diferenciar de forma fácil los tipos de trabajo y definir cuando una tarea puede pasar de una columna a otra.
- **Ciclos de feedback**, para poder detectar los problemas en el flujo de trabajo y ajustarlo a medida que sea necesario.

- **Modelos o métodos científicos** que ayudan a identificar los problemas que ocurren en el flujo de trabajo, como por ejemplo *The Theory of Constraints* para el estudio de los cuellos de botella.

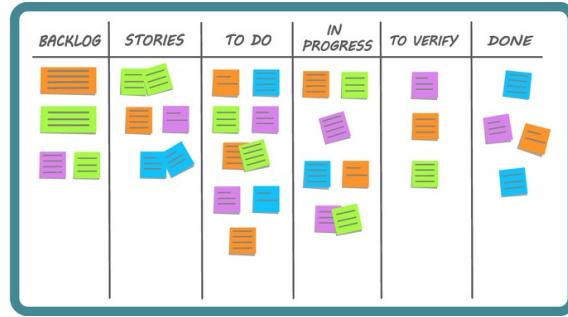


Figura 1.3: Ejemplo de un tablero Kanban.  
Fuente: David J. Anderson School of Management (2021).

Como se puede observar en la figura 1.3, el tablero se separa en distintas columnas en las que se ubican las tareas según el estado en el que estas se encuentran. La cantidad de columnas que se tiene en el tablero es variable y depende totalmente del flujo de trabajo, además, se puede hacer otra separación extra por filas ya sea para medir la importancia de las tareas, la sección o módulo del sistema al que corresponde o a quien está encargado de hacerla.

Para aprovechar lo que ofrece la metodología se tomó la decisión de utilizar ocho tableros distintos, de los cuales siete corresponden a los objetivos específicos del trabajo y el octavo uno específico para tareas relacionadas a la escritura de la memoria. De esta manera se pueden establecer tareas puntuales para avanzar los objetivos específicos por separado y también determinar cuales ya han sido cumplidos o cuales aún necesitan mas tiempo. La plataforma escogida para aplicar la metodología es Trello en su versión gratuita.

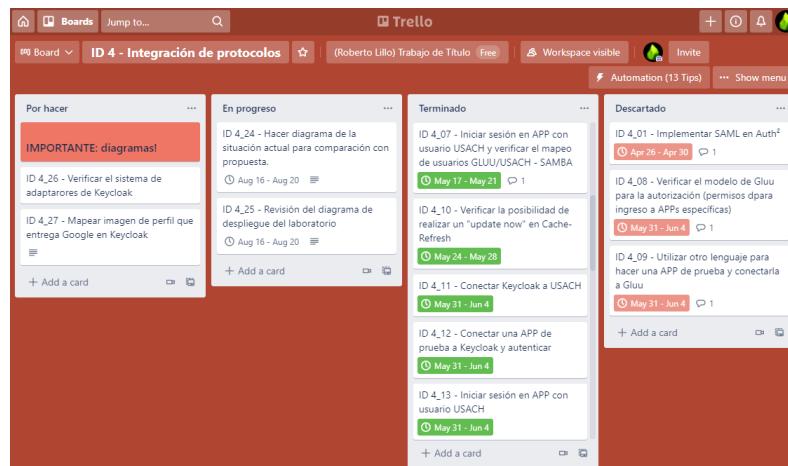


Figura 1.4: Tablero Kanban en Trello.  
Fuente: Elaboración propia (2021).

### **1.5.2 Herramientas de desarrollo**

Para la investigación, desarrollo e implementación se hará uso de las siguientes herramientas.

- **VirtualBox** para levantamiento de máquinas virtuales de prueba.
- **Visual Studio Code** para secciones que requieren código.
- **Firefox** para el ingreso a interfaces Web.
- **Docker** para empaquetamiento de servicios y aplicaciones.
- **Ansible** para creación de scripts de lanzamiento.

Para el desarrollo del sistema y la documentación asociada se utilizarán principalmente dos computadores con las siguientes características.

- **Computador de escritorio**

- Sistema Operativo: Ubuntu 20.04 LTS
  - CPU: Intel Core i7-9700K
  - RAM: 16GB DDR4

- **MacBook Air**

- Sistema Operativo: MacOS 11
  - CPU: Intel Core i5-5350U
  - RAM: 8GB LPDDR3

## **1.6 ORGANIZACIÓN DEL DOCUMENTO**

Este documento está dividido principalmente en cuatro capítulos, los cuales son los siguientes.

- **Capítulo 1. Introducción**, que tiene el fin de dar a conocer la problemática y la solución escogida.
- **Capítulo 2. Marco teórico**, este capítulo presenta la información importante y necesaria para comprender los protocolos utilizados y decisiones tomadas durante el tiempo de trabajo de la solución.

- **Capítulo 3. Desarrollo**, corresponde a la documentación en relación a la investigación e implementación realizada durante las diecisiete semanas de trabajo. Tal como se acaba de mencionar, este capítulo se subdivide en una sección de investigación y otra de implementación.
- **Capítulo 4. Conclusiones**, que corresponde al análisis del trabajo realizado, los resultados obtenidos durante el tiempo de desarrollo y también posibilidades de tareas a futuro para el sistema implementado.

## CAPÍTULO 2. MARCO TEÓRICO

En una empresa o departamento, generalmente se cuenta con una infraestructura en red con la capacidad de proveer servicios a los trabajadores. Estas son diseñadas para que usuarios o *hardware* como computadores puedan acceder a servicios como cuenta de usuario, directorio de archivos, impresoras, aplicaciones de *software*, entre otros.

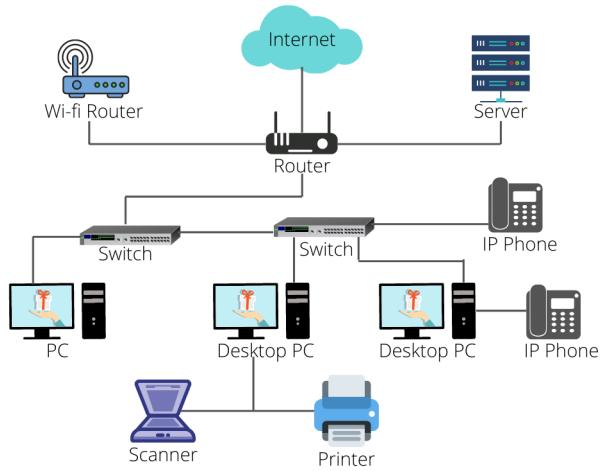


Figura 2.1: Diagrama de una infraestructura de red de ejemplo.  
Fuente: EralInnovator (2021).

Pero una de las piedras angulares del establecimiento de un entorno de red seguro es mantener el acceso restringido a los usuarios y clientes que tienen derecho a acceder a la red (RedHat, 2020), de esta situación nacen términos como la **Autenticación, Autorización** y los **Servicios de Directorio**, conceptos que son descritos en mayor profundidad en las siguientes secciones.

### 2.1 AUTENTICACIÓN

En el contexto o ambiente de redes, la autenticación corresponde al acto o proceso de confirmar la identidad de alguien ya sea un usuario o una máquina, el cual recibe el nombre de cliente. Cuando esta interacción ocurre dentro de una red, es común que interactúen como mínimo dos partes; el cliente que desea autenticarse y un encargado de validar la identidad del usuario conocido como **Proveedor de Identidad/Identity Provider** (IdP), este puede soportar varios tipos de modelos de autenticación, lo cual determinará qué tipo de credenciales necesitará entregar la entidad que busca verificar su identidad.

## 2.1.1 Modelos de autenticación

### *Autenticación por contraseña/Password-based authentication*

También conocida como autenticación simple, la autenticación del usuario es mediante un nombre conocido y una contraseña. Existen otros casos especiales como las **Contraseñas de un Solo Uso/One-Time Password** (OTP), en donde cada vez que el usuario desee autenticarse se genera una nueva contraseña la cual el usuario puede recibir previamente de distintas formas (correo electrónico, mensaje de texto, entre otros). Las autenticaciones de este estilo caen dentro de la categoría de criptografía simétrica, que se basa en la utilización de una única clave secreta que se encargará de cifrar y descifrar la información (López, 2021).

### *Autenticación por certificados/Certificate-based authentication*

Este modelo es parte del protocolo **Secure Socket Layer** (SSL), el que corresponde a un protocolo criptográfico que proporciona privacidad e integridad en la comunicación entre dos puntos de una red, garantizando que la información transmitida por dicha red no pueda ser interceptada ni modificada por elementos no autorizados (Redalia, s.f.). Este cae dentro de la criptografía asimétrica, basada en dos llaves: una privada y una pública, de esta manera el remitente conserva el certificado o llave privada y la pública puede ser entregada a cualquier receptor (Rivas, 2020). Mediante el certificado público es posible encriptar mensajes que sólo pueden ser descifrados mediante la llave privada, mientras que la llave privada entrega autenticidad con respecto al emisor de la información.

### *Autenticación por tarjeta inteligente/Smart card-based authentication*

Este modelo es que agrega al proceso de autenticación un componente físico para confirmar la identidad del usuario. Es una variante de la autenticación por certificados, en donde esta vez el certificado se encuentra guardado en la tarjeta o dispositivo USB que se utiliza como llave, la cual debe ser ingresada en el momento que el usuario desee autenticarse, para que así el IdP pueda verificar la validez de tal llave.

### *Autenticación biométrica/Biometric-based authentication*

Similar a la autenticación por tarjeta inteligente o *token* USB, la autenticación biométrica utiliza un componente físico para confirmar la identidad del usuario, pero a diferencia de utilizar un certificado se utiliza un elemento del cuerpo del usuario (Vinsot, 2008), comúnmente la huella dactilar o el ojo (retina o iris). Para esto el IdP requiere tener previo conocimiento de estas características del usuario en su base de datos.

### 2.1.2 Otras formas de reforzar la seguridad

Para agregar aún más seguridad al proceso se puede añadir al sistema la **Autenticación Multi-Factor/Multi-Factor Authentication** (MFA), la que puede agregar 1 ó más pasos a la verificación de la identidad del usuario. Por ejemplo, una autenticación de 1 solo factor puede ser el uso de ID + Contraseña, pero se puede ampliar a 2 factores haciendo que el usuario reciba en su correo electrónico un *token* generado aleatoriamente y con un tiempo de caducidad, el cual debe ingresar en el sistema para completar la verificación de su identidad. Esto se puede incluso ampliar a 3 factores utilizando también biométrica u otro mecanismo.

## 2.2 AUTORIZACIÓN

Posterior a que un usuario pueda autenticarse exitosamente ocurre el proceso de autorización, del que se encarga un servicio de autorización. Este define qué permisos tiene el usuario y a qué recursos tiene acceso, ya sea servicios, aplicaciones, impresoras, bases de datos u otros. En la mayoría de los sistemas la autenticación y la autorización coexisten, ambos se encargan de funciones de seguridad distintos, pero trabajan en conjunto para asegurar que los recursos de la red sean utilizados por quien corresponda. Las diferencia en las funciones que realizan estos dos procesos se pueden comparar en la tabla 2.1.

Autenticación	Autorización
Verifica la identidad del usuario	Valida los permisos de acceso
Verifica que el usuario realmente sea quien afirman ser	Confirma si el usuario tiene los permisos para acceder a cierto recurso
Determina la identidad del usuario mediante nombre de usuario, contraseña, huella dactilar, etc	Valida los permisos y privilegios de acceso del usuario mediante reglas pre-establecidas en el sistema
Ej: un empleado requiere autenticarse para ingresar a su estación de trabajo	Ej: posterior a la autenticación, el empleado tiene acceso a la impresora

Tabla 2.1: Comparación autenticación vs autorización.

Fuente: Elaboración propia (2021).

Al igual que un IdP, un servidor de autorización puede soportar distintos modelos para definir los permisos del usuario y el acceso a recursos, existen varios **Modelos de Control de Acceso/Access Control Models**.

## 2.2.1 Modelos de autorización

### *Control de acceso obligatorio/Mandatory Access Control (MAC)*

Considerado el modelo más estricto (University of Hawaii West Oahu, 2018), el acceso a toda información en el sistema está predefinido por un administrador y controlado por un sistema operativo. Utiliza un enfoque jerárquico para otorgar permisos en donde se le otorga un nivel de confidencialidad (*secrecy*) a la información y también un nivel de acceso al cliente, de esta forma para que un cliente pueda acceder a un dato su nivel de acceso no puede ser menor del nivel de confidencialidad de la información. Como se mencionó anteriormente, el sistema operativo se encarga de verificar las propiedades de la información y del cliente, otorgando o no el acceso.

### *Control de acceso discrecional/Discretionary Access Control (DAC)*

En este método se hace uso de varias **Listas de control de Acceso/Access Control List** (ACL), estas listas contienen información sobre el dato u objeto que hay en el sistema y qué clientes o grupos tienen acceso a dicho recurso. De esta forma cuando un cliente intenta acceder a una información específica, el sistema operativo verifica si es que él está incluido en la ACL y luego hace una revisión de qué permisos tiene sobre esta información ya sea lectura, escritura, etc. Otro detalle importante de este modelo es que además de tener un administrador encargado de definir los permisos y acceso a la información más importante, permite a los mismos usuarios determinar el acceso a los recursos que ellos crearon y así permitiendo mayor rapidez en la administración de los mismos. Debido a esto, el sistema operativo recibe la tarea extra de verificar que los mismos usuarios no puedan cambiar el acceso a los recursos ya sea por error o maliciosamente.

### *Control de acceso por roles/Role-Based Access Control (RBAC)*

Este modelo es utilizado cuando es necesario que los permisos sean asignados a roles organizacionales en vez de clientes individuales. La idea principal es que los roles sean definidos como abstracciones de los procesos de trabajo de la organización (Ionos, 2020), de esta manera se puede simplificar el acceso a la información que tiene cada rol, por ejemplo, el rol de “Ventas” debería tener acceso a información relacionada a compradores o transacciones, mientras que el rol de “Ingeniero de software” el acceso a los repositorios o documentación. Cuando un cliente es asignado a uno o más roles, este recibe de forma automática los permisos asignados, no obstante, no puede recibir ningún otro permiso fuera de los que estén contenidos en los mismos roles.

#### *Control de acceso por reglas/Rule-Based Access Control*

Toma conceptos del modelo basado en roles y el modelo DAC. En este caso el acceso es otorgado dependiendo de reglas predefinidas por un administrador, en donde al igual que en DAC cada objeto o información tiene su propio ACL (Techotopia, 2016) y el sistema operativo se encarga de verificar si es que cierto cliente o rol tiene permiso para acceder a dicho recurso. Particularmente, este modelo a pesar de trabajar con ACLs no permite a los clientes cambiar los permisos de ningún objeto, dejando esta tarea completamente en manos del administrador de la organización.

#### *Control de acceso por atributos/Attribute-Based Access Control (ABAC)*

En este modelo los derechos son otorgados dependiendo de los atributos o características de un cliente en vez de roles (Casey, 2020). Esto permite crear políticas mucho más específicas sobre quién tiene los derechos de acceso al recurso o información en juego, ya que cada situación de verificación es evaluada no sólo desde el rol del cliente o de la acción que desea realizar, sino que también dependiendo de las condiciones de los diferentes atributos que hayan sido asignados para satisfacer el acceso al recurso. ABAC utiliza lógica booleana con sentencias del tipo “*IF*” y “*THEN*” para verificar si se permite o no el acceso.

## **2.3 ESTÁNDARES Y PROTOCOLOS DE AUTENTICACIÓN Y AUTORIZACIÓN**

Tal como se vio en las secciones anteriores existen diversas formas de mantener la información segura, existiendo así variados modelos encargados de esto, no obstante, para poder implementarlos es necesario definir el estándar o protocolo a utilizar y considerar que este sea compatible con el modelo que sea deseado emplear y del contexto donde se va a utilizar. En esta sección se presentan tres protocolos considerados para el proyecto, además de un procedimiento de inicio de sesión que es transversal. Se encuentran tres términos recurrentes en los protocolos listados, listados con una pequeña definición a continuación:

- **Proveedor de Identidad/Identity Provider (IdP):** componente que se encarga del inicio de sesión y autenticación del usuario.
- **Proveedor de Servicios/Service Provider (SP):** entidad en la que el usuario desea autenticarse para utilizar algún servicio.
- **Agente de Usuario/User Agent:** aplicación que funciona como cliente de acceso al internet, por lo general es un navegador Web o una aplicación similar.

### 2.3.1 Single Sign-On (SSO)

**Single Sign-On** corresponde a un procedimiento de autenticación que ocurre cuando un usuario inicia sesión en una aplicación y automáticamente es iniciado también en otra serie de aplicaciones. El objetivo de este procedimiento es proveer una experiencia fluida al usuario cuando esté utilizando distintas aplicaciones o servicios (Auth0, 2021), por ejemplo, al iniciar sesión en un servicio de Google como Gmail también es posible encontrar automáticamente tiene la sesión iniciada en YouTube, Google Drive, etc. De esta misma forma existe el procedimiento contrario llamado **Single Logout**, permitiendo así cerrar sesión en todas las aplicaciones al momento de hacerlo en sólo una.

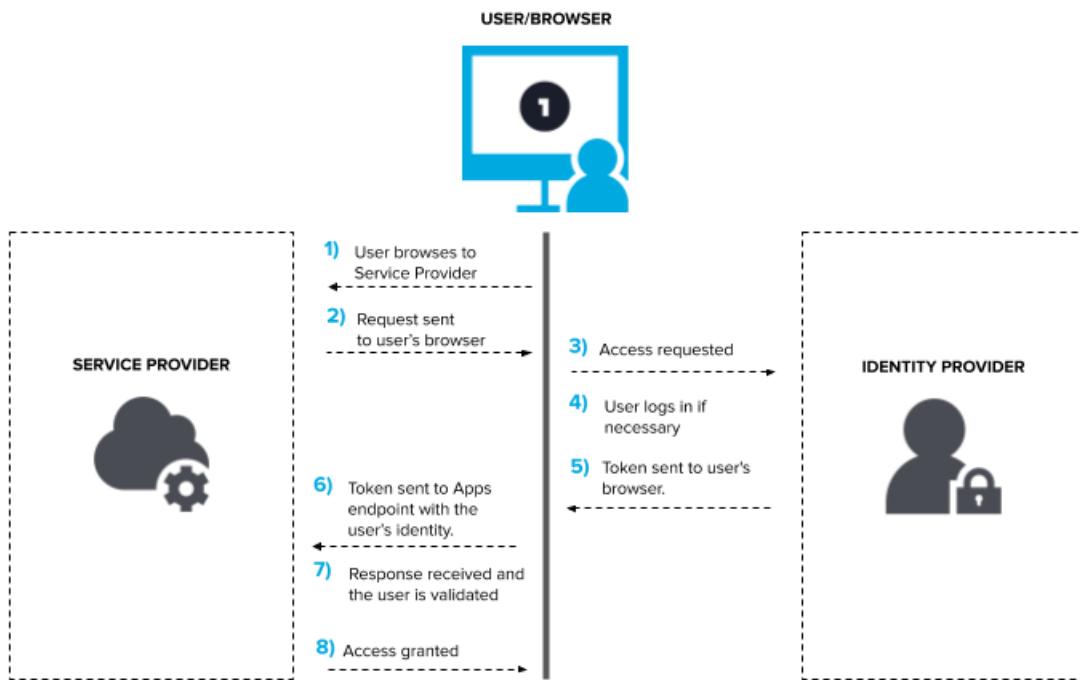


Figura 2.2: Flujo en el que ocurre el procedimiento de *Single Sign-On*.  
Fuente: OneLogin (2021).

En la figura 2.2 se puede apreciar un flujo de inicio de sesión donde se aplica SSO, permitiendo al usuario estar autenticado en múltiples aplicaciones. Los pasos que sigue este proceso son los siguientes:

1. El usuario solicita ingresar al servicio mediante el navegador.
2. El proveedor de servicios responde con la información.
3. El usuario solicita iniciar sesión y es redireccionado.
4. El usuario inicia sesión en el proveedor de identidad.

5. El IdP envía una respuesta positiva al navegador.
6. La respuesta es desplegada en los distintos *endpoints* de las aplicaciones del SP.
7. Se valida el acceso del usuario en las aplicaciones.
8. Se permite el acceso al usuario a las aplicaciones disponibles.

Un aspecto importante del SSO es que para ser implementado necesita estar soportado por el protocolo o estándar utilizado por el proveedor de identidad, en el caso contrario no es posible iniciar sesión en mas de una aplicación a la vez automáticamente. En esta ocasión, los siguientes tres protocolos a presentar poseen la capacidad de ofrecer SSO.

### **2.3.2 Security Assertion Markup Language (SAML)**

Este corresponde a una estándar libre utilizado generalmente como un mecanismo de autenticación para aplicaciones Web (Okta, 2020), su desarrollo comenzó el año 2001 con la versión 1.0 y luego siendo lanzada la versión 2.0 en el año 2005. Actualmente se utiliza la última mencionada, pero ha sido actualizada en base a las nuevas necesidades.

Para solicitar una autenticación al proveedor de identidad mediante SAML es necesario enviar una **SAML Request**, que corresponde a un documento que contiene información del IdP y del SP que está haciendo la solicitud. Luego que el usuario ingresa sus credenciales, el IdP genera y entrega un documento llamado **SAML Assertion**, este contiene la afirmación de la autenticación del usuario y además puede contener otro tipo de información como id, perfil, grupos, método de autenticación, etc. Los documentos en SAML son construidos bajo el formato XML<sup>1</sup>, que corresponde a un lenguaje de marcado similar a otros como HTML.

La figura 2.3 muestra el flujo para iniciar sesión en un SP mediante SAML, muy similar al flujo mostrado en la figura 2.2 pero estableciendo las ubicaciones de la *SAML request* y *SAML assertion*.

El inicio de sesión en SAML puede ocurrir tanto en el proveedor de servicios como en el proveedor de identidad, recibiendo el nombre de **SP-initiated** y **IdP-initiated** respectivamente, en la figura 2.3 pueden ser identificados en los pasos 2 y 4. En el caso que sea iniciado por el IdP, este queda preparado para responder a las solicitudes de los SP.

---

<sup>1</sup>[https://developer.mozilla.org/docs/Web/XML/XML\\_introduction](https://developer.mozilla.org/docs/Web/XML/XML_introduction)

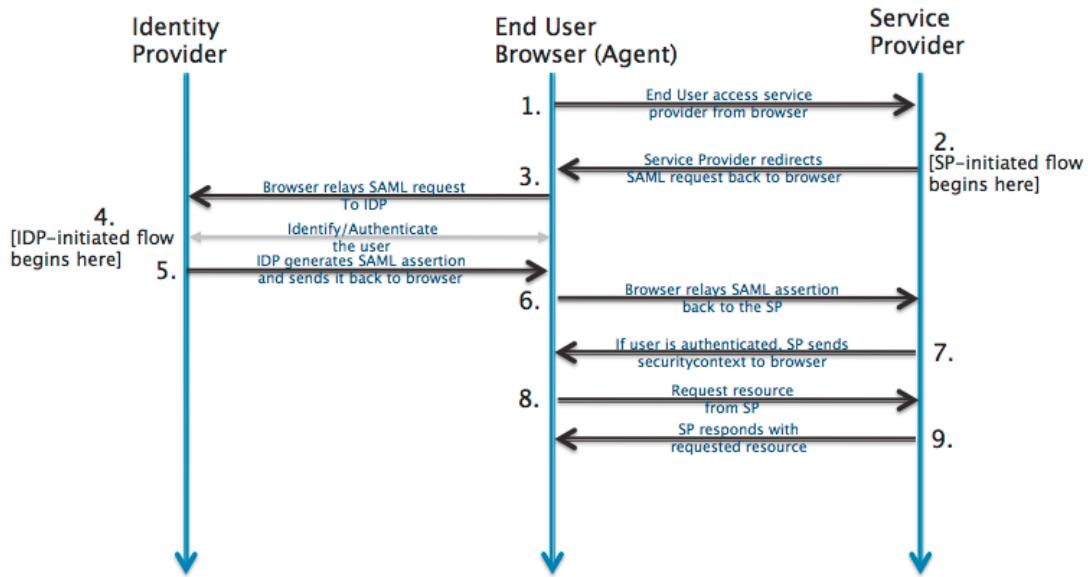


Figura 2.3: Flujo de autenticación mediante el estándar SAML.  
Fuente: Okta (2020).

### 2.3.3 OAuth

**OAuth**<sup>2</sup> corresponde a un protocolo de autorización que permite dar consentimiento para el uso de APIs, permitiendo su uso en aplicaciones Web, de escritorio y móviles (Espinosa, 2021), la versión 1.0 fue lanzada el año 2009 y la versión actual 2.0 en el año 2013. En este estándar interactúan cuatro partes: primero se encuentra el **Recurso Protegido/Protected Resource** que corresponde a la API a la que se desea acceder, segundo el **Cliente/Client** que es una aplicación que quiere acceder al recurso protegido en nombre de alguien, tercero está el usuario que en este caso se llama **Propietario del Recurso/Resource Owner**, se llama de esta forma ya que es el dueño de los datos que son manejados y finalmente el cuarto es el **Servidor de Autorización/Authorization Server** quien es responsable de gestionar las peticiones de autorización.

Para que un cliente pueda utilizar una API requiere del consentimiento del usuario, esto generalmente se despliega en una interfaz que muestra la información a la que el cliente puede y no puede acceder si se aprueba. Una vez se da el consentimiento el cliente puede utilizar el código de autorización entregado para solicitar un **Token de Acceso/Access Token** al servidor de autorización, este token finalmente es utilizado para acceder a la API del recurso protegido al que el cliente quiere acceder. Toda la comunicación realizada por el protocolo es transmitida por HTTPS y en formato JSON<sup>3</sup>.

<sup>2</sup><https://oauth.net/>

<sup>3</sup><https://www.json.org>

El flujo que muestra la figura 2.4 enumera algunos de los pasos previamente descritos. Específicamente, lo que sucede en cada paso de la figura está listado a continuación:

1. El cliente realiza una solicitud de autorización al servidor de autorización.
2. El usuario inicia sesión y da consentimiento al cliente.
3. El servidor de autorización genera y envía un código de autorización al cliente.
4. El cliente utiliza este código para solicitar un *token* de acceso mediante el *endpoint* correspondiente en el servidor de autorización.
5. El servidor de autorización genera y envía un *token* de acceso al cliente.

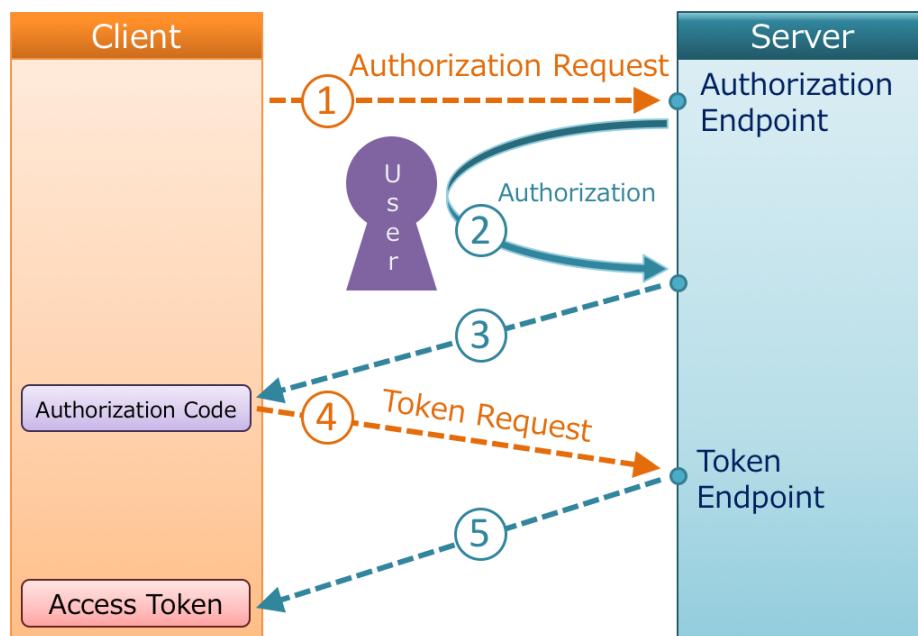


Figura 2.4: Flujo de autorización mediante el protocolo OAuth.  
Fuente: Kawasaki (2017).

Todos los *tokens* generados por el servidor están bajo una política de expiración, usualmente definida por una cantidad de tiempo máxima. Para evitar que el usuario deba entrar constantemente a permitir la generación de nuevas versiones, junto al *token* de acceso se entrega también un **Token de Revalidación/Refresh Token**. Este último también está limitado por un tiempo máximo de validez pero que es mayor, de esta forma mientras este se encuentre vigente es posible revalidar el *token* de acceso las veces que sea necesario.

### 2.3.4 OpenID Connect (OIDC)

Luego de ver el funcionamiento de OAuth, se puede apreciar que al ser solamente un protocolo de autorización carece de forma de identificar a los usuarios que dieron consentimiento. **OpenID Connect<sup>4</sup>** está diseñado como una capa extra para suplir dicha carencia, pensado para encargarse de la autenticación y OAuth de la autorización (Torres, 2019).

Esto se logra al añadir funcionalidades extra, primero se encuentra el **ID Token** que permite saber quién es el usuario y segundo un nuevo *endpoint* en el que se puede consultar mayor información sobre el mismo en el caso de ser necesario. De esta forma, OIDC y OAuth juntos permiten implementar un servicio de autenticación y autorización completo, con la capacidad de entregar un **ID Token** que contiene la información del usuario incluyendo grupos o roles y además un **Access Token** permitiendo autorización de acceso a APIs.

La figura 2.5 muestra las diferencias que ocurren en comparación a la figura 2.4 cuando se agrega la autenticación al flujo. En este caso también se da consentimiento para la autenticación y posteriormente cuando el cliente utiliza el código de autorización en el *endpoint*, el servidor de autorización envía los dos *tokens* de ID y de acceso.

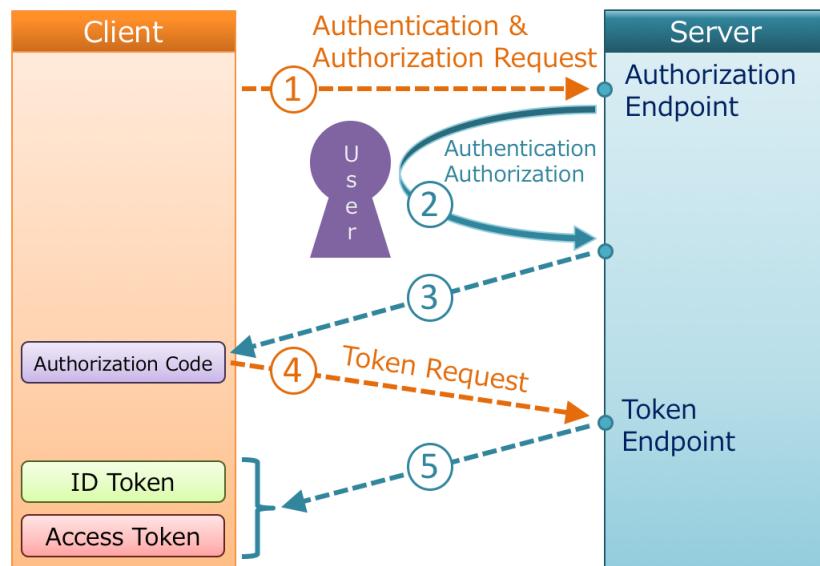


Figura 2.5: Flujo de autenticación y autorización mediante los protocolos OIDC y OAuth.  
Fuente: Kawasaki (2017).

Tanto OAuth como OIDC ofrecen seguridad al usuario y sus credenciales al requerir que los inicios de sesión se realicen frente al servidor de autorización directamente, de esta forma, el cliente nunca tiene la capacidad de ver esta información.

<sup>4</sup><https://openid.net/connect/>

## 2.4 SERVICIOS DE DIRECTORIO

En un aspecto técnico, un directorio es similar a una base de datos pero típicamente contiene información más descriptiva guardada en forma de atributos, con la particularidad que frecuentemente es más leída que escrita (Microsoft, 2018). La utilidad de un servicio de directorio recae en su capacidad de guardar información para la administración de clientes o servicios dentro de una red u organización de gran escala: dicho de otra forma, un servicio de directorio corresponde principalmente a una base de datos que contiene información sobre usuarios y activos de una organización, sobre la que se pueden construir capas para aplicar los modelos de autenticación y autorización previamente descritas.

Para el acceso remoto a la información guardada en el directorio es estándar que se utilice el protocolo **LDAP** (*Lightweight Directory Access Protocol*), este corresponde a un protocolo de comunicación *open source* a nivel de aplicación que permite el acceso a un servicio de directorio para buscar información. LDAP define el método en el que se accede al directorio, los requerimientos para crear entradas de información y establece también la forma en que esta información es guardada (Ellingswood, 2015).

### 2.4.1 Estructura de directorios LDAP

La forma básica que establece LDAP para guardar información es mediante **atributos**, los cuales son pares de llave-valor en los que las llaves tienen nombres predefinidos por el directorio. Para que la información que contienen estos atributos tenga sentido, son agrupados en colecciones denominadas **entradas/entries**, por ejemplo, la entrada de un usuario particular puede contener los atributos nombre, apellido, correo electrónico, etc. Una entrada puede representar variadas cosas por lo que algunas de estas tienen simplemente la labor de organizar el directorio, permitiendo así utilizar el concepto de **grupos/groups** para asociar distintas entradas. Siguiendo el ejemplo anterior, la creación del grupo "Usuarios" permitiría reunir todas las entradas que contengan información de usuarios. Para encontrar las distintas entradas se puede utilizar el atributo estándar **Common Name** (CN), que corresponde a un identificador que tiene solamente un pequeño alcance (Willeke, 2018), este puede ser definido de forma arbitraria o construido a partir de ciertos atributos, por ejemplo, el CN de un usuario puede ser el resultado de unir su nombre y apellido.

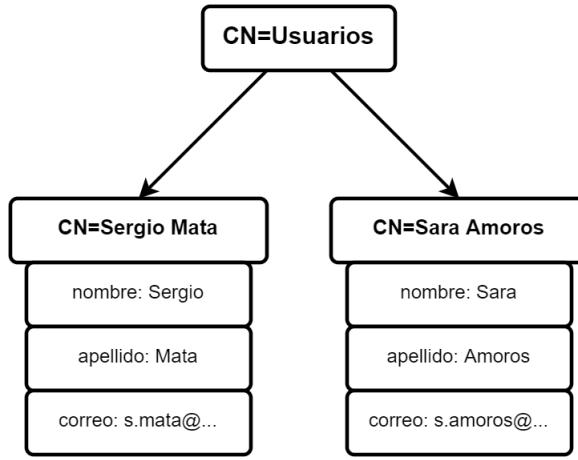


Figura 2.6: Organización de dos usuarios mediante un grupo, utilizando *Common Name*.  
Fuente: Elaboración Propia (2021).

El concepto de grupos no basta para representar la estructura de una organización dentro de un directorio, generalmente existen subdivisiones en unidades que están encargadas de distintas áreas. Para representar esto LDAP provee la **Unidad Organizativa/Organizational Unit** (OU), permitiendo así establecer distintas unidades o áreas dentro de un directorio y luego ubicando en ellas los grupos y entradas correspondientes. En una escala jerárquica las unidades organizativas están por encima de los grupos, además, la forma de ser identificadas es mediante el nombre de la OU a diferencia de los anteriores que utilizan el CN.

En el ejemplo de la figura 2.7, se ve la situación hipotética de una organización que tiene su área de desarrollo de *software*, la que a su vez está dividida en el grupo de desarrolladores de *backend* y desarrolladores de *frontend*.

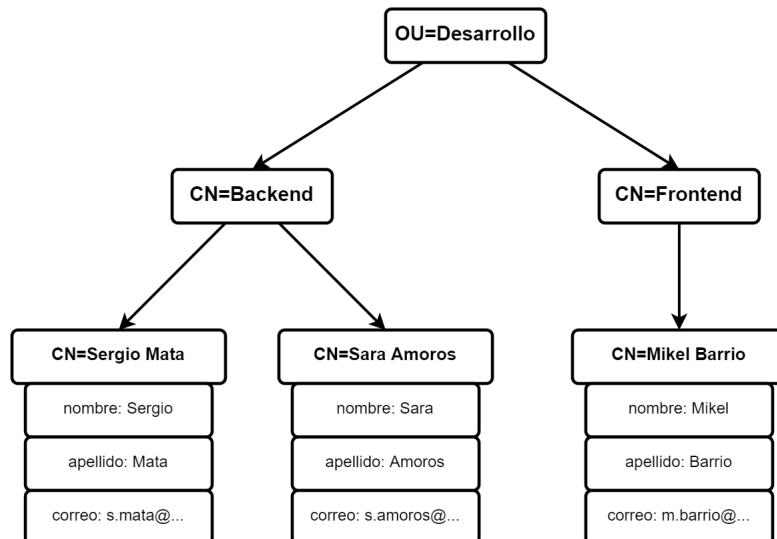


Figura 2.7: Organización de tres usuarios mediante dos grupos y una unidad organizativa.  
Fuente: Elaboración Propia (2021).

Para completar la representación dentro del directorio existe el concepto del **Controlador de Dominio/Domain Controller** (DC), el cual contiene y controla las distintas entradas que se encuentran en el directorio. Este suele recibir un nombre de dominio completo, incluyendo sufijos como por ejemplo ".lan" o ".org", según sea el caso.

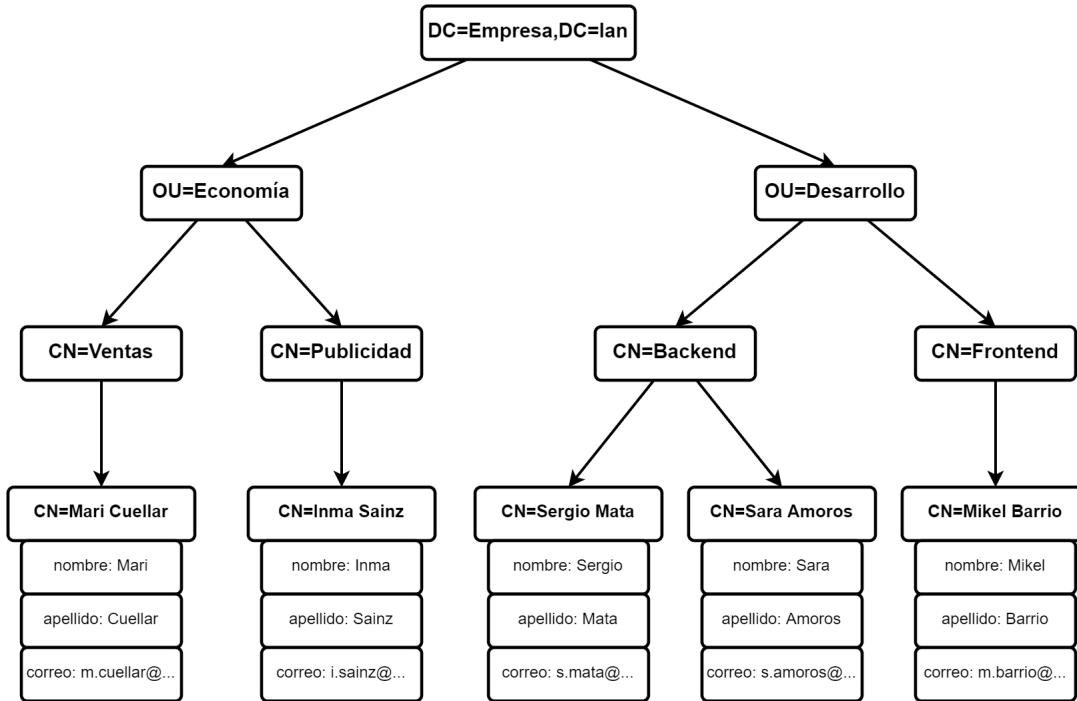


Figura 2.8: Organización de cinco entradas mediante cuatro grupos, dos unidades organizativas y un dominio.

Fuente: Elaboración Propia (2021).

Como se aprecia en las figuras 2.7 y 2.8, la estructura que sigue LDAP para organizar la información es mediante árboles jerárquicos llamados **Directory Information Trees** (DITs), estos finalmente son los que representan la estructura organizativa y qué tan amplio es el directorio, un detalle importante es que en estos árboles cada entrada tiene exactamente un solo parente, pero puede tener cualquier número de hijos.

Con la amplitud que tienen estos árboles el *Common Name* no basta para encontrar a un usuario y corresponde utilizar el **Distinguished Name** (DN), este identificador incluye el CN pero también utiliza información extra como el grupo o unidad organizativa donde se encuentra la entrada y además, siempre se indica el dominio al que pertenece: en otras palabras, el DN corresponde a la dirección completa de una entrada en el árbol del directorio. La figura 2.9 muestra un ejemplo de la construcción de un *Distinguished Name*, en este caso no es necesario incluir el CN del grupo ya que al encontrar la OU, es posible utilizar el CN de la entrada para buscar dentro de ella.

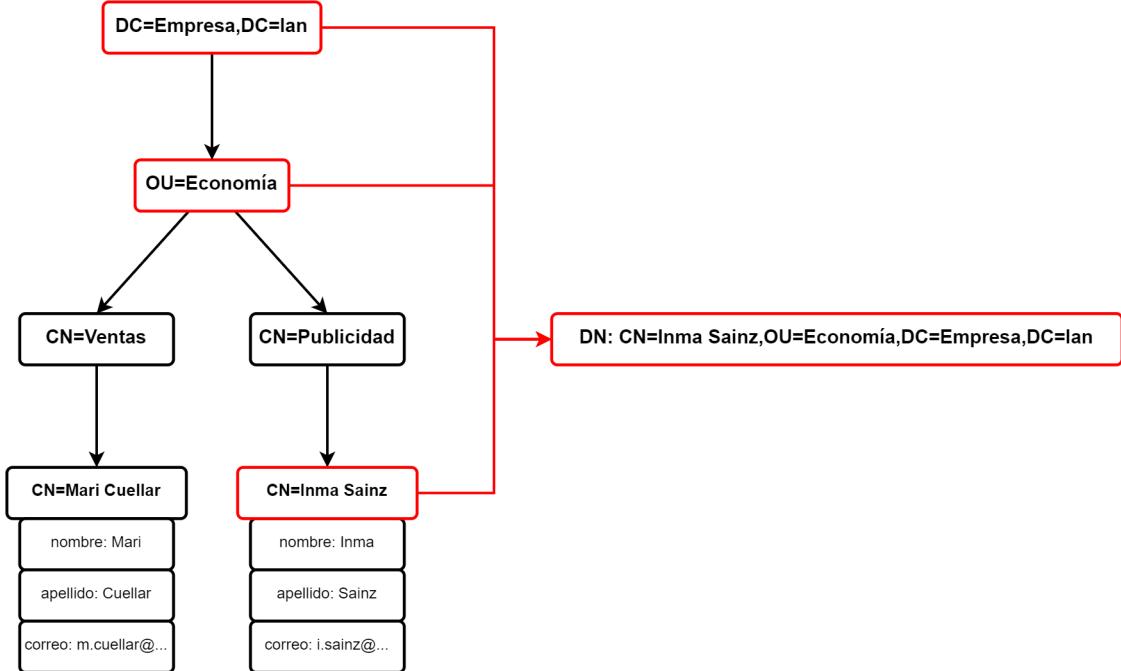


Figura 2.9: Construcción de un *Distinguished Name* a partir del *Common Name*, *Organizational Unit* y *Domain*.

Fuente: Elaboración Propia (2021).

A este punto cada una de las entradas pueden ser tratadas como **Objetos/Objects** que contienen los respectivos atributos previamente determinados por una clase llamada **Object Class**, esta clase especifica los atributos obligatorios y opcionales que pueden ser ingresados una entrada de dicha clase (Oracle, s.f.). También existe otro concepto más amplio que engloba las características de la construcción de un directorio y es el **Esquema/Schema**. Mediante el esquema se definen una serie de elementos como las *Object Classes*, tipo y sintaxis de atributos para las entradas y reglas de comparación del directorio (LDAP, 2018c)

Si bien un árbol tiene la capacidad de representar una organización, está limitado a un cierto alcance que usualmente está definido por el espacio físico donde esta se encuentra, es decir, un árbol de directorio está pensado para representar un edificio u oficina.

Para los casos donde existen variadas sucursales que requieren ser administradas es posible utilizar el concepto de **Federaciones de Confianza/Federation Trust**. Este método permite que dos o más árboles puedan establecer una comunicación y puedan intercambiar entradas de información, de esta forma, es posible recuperar información sobre un objeto que se encuentra en otra sucursal mediante el correcto *Distinguished Name* de la entrada.

Esto abre las puertas a otros conceptos como la **Federación de Usuarios/User Federation**, permitiendo así que los usuarios de un directorio puedan ser autenticados y utilizados en otro directorio.

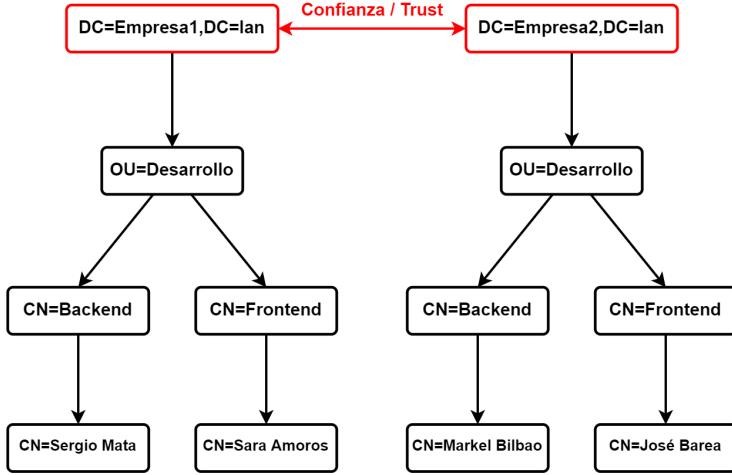


Figura 2.10: Enlace de federación de confianza entre dos árboles de directorio.  
Fuente: Elaboración Propia (2021).

En la figura 2.10 se ve un ejemplo de federación de confianza entre dos árboles, destacando que tanto la comunicación inicial para establecer confianza como las posteriores comunicaciones entre los directorios se realiza mediante el controlador de dominio. Al igual que su contraparte natural, el establecimiento de relación entre variados árboles recibe el nombre de **Bosque/Forest**, este concepto está al tope de la jerarquía de los directorios.

#### 2.4.2 Comunicación mediante el protocolo LDAP

Para conectarse a través LDAP es necesario establecer la conexión mediante una operación **Bind**, estas son usadas para autenticar clientes al directorio, establecer el nivel de autorización permitido para las siguientes operaciones de lectura o escritura y para especificar la versión del protocolo que el cliente usará (LDAP, 2018a).

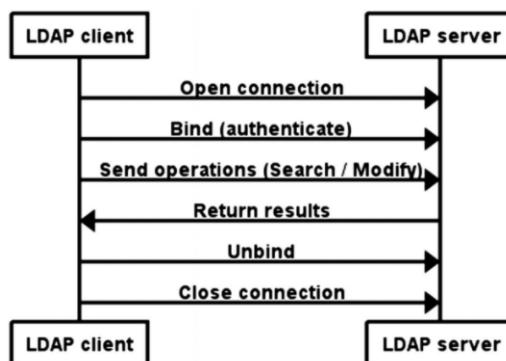


Figura 2.11: Secuencia en la que se realiza una conexión con un servidor LDAP.  
Fuente: Schwartz & Machulak (2018).

Como se puede ver en la figura 2.11, además de la operación de *bind* inicial existe otra llamada ***Unbind*** que ocurre posterior a la lectura o modificación del directorio. Esta operación permite al cliente dar una señal al directorio de que está cercano a cerrar la conexión, de esta forma el servidor también puede cerrar la conexión con el cliente dejando así la instancia de la conexión inutilizable (LDAP, 2018b). Para casos especiales, el servidor debe estar preparado para también cerrar la conexión cuando no reciba una señal de *Unbind* por parte del cliente, por ejemplo, cuando la conexión no haya sido usada por una cantidad arbitraria de tiempo.

### 2.4.3 Algunos servicios de directorio que implementan LDAP

Si bien existe una variada gama de servicios de directorio que funcionan bajo el protocolo LDAP, esta sección presenta principalmente los que son relevantes para el posterior desarrollo del sistema centralizado. Tradicionalmente estos servicios son instalados en una red local, pero hoy en día gracias a la proliferación de los servicios en la nube también es posible encontrarlos como ***Software as a Service*** (SaaS).

#### *OpenLDAP*

OpenLDAP es un esfuerzo colaborativo para desarrollar una implementación robusta, de nivel comercial y libre (*open source*) de un conjunto de aplicaciones y herramientas de desarrollo (Chu et al., 2014). LDAP en algunos casos también es considerado una base de datos jerárquica, donde las herramientas de OpenLDAP permiten levantar, configurar y acceder a esta mediante variados tipos de consultas. Algo importante a mencionar es que LDAP como protocolo no depende de la plataforma donde se encuentre, en general distribuciones Linux incluyen OpenLDAP para dar soporte al protocolo, pero este también puede ser utilizado por otras variantes como Android, MacOS y Windows.

#### *Microsoft Active Directory*

Corresponde a la implementación de un servicio de directorio realizado por Microsoft en específico para su sistema operativo Windows, la documentación lo describe como una estructura jerárquica que guarda información acerca de objetos en una red (Microsoft, 2017), estos objetos generalmente incluyen recursos compartidos como servidores, archivos, impresoras, computadores, usuarios y grupos de usuarios. Al igual que OpenLDAP, también contempla un conjunto de aplicaciones y herramientas para configurar los objetos que se encuentran en la red y además las políticas de acceso a estos. Durante mucho tiempo fue sólo un servicio local con instalación directa en un servidor dentro de la organización, pero actualmente

Microsoft también lo ofrece como un servicio en la nube bajo el modelo SaaS llamado Azure Active Directory. Estos dos son servicios de pago y el costo está asociado a la cantidad de equipos o recursos que hay en la red y que son integrados al directorio.

### Samba

Este es principalmente un *software* libre capaz de proveer servicios de archivos e impresiones a una variedad de clientes, con la notable capacidad de ofrecer esto también a clientes Windows (Seeger, 2006). A partir de la versión 4.0 tiene la capacidad de funcionar como un servidor de *Active Directory* conocido como **Samba Active Directory**, ofreciendo varias de las características que este servicio provee con sistemas operativos Windows, con el gran extra de permitir que recursos como servidores o computadores con sistema operativo Linux sean agregados al directorio. Nuevamente, su importancia recae en la interoperabilidad entre sistemas y en que su estrategia está direccionada a compatibilizar *software* y *hardware* para Windows y Linux, permitiendo así que las organizaciones no estén limitadas por el sistema operativo que es aceptado por su servicio de directorio.

## 2.5 ADMINISTRACIÓN DE IDENTIDAD, DE ACCESO Y GOBERNANZA

Posterior a la implementación de un servicio de directorio para administrar una organización, es necesario determinar los niveles de autenticación y autorización con los que se desea trabajar, existiendo así distintas capas a escoger según la necesidad.

Por ejemplo, si se desea implementar un servicio con el fin único de reconocer al usuario que está iniciando sesión en el sistema, es posible trabajar con un servicio de **Identity Management (IdM)** encargado de permitir la administración de cuentas de usuario, contraseñas y delegar tareas a otros sistemas. Por otro lado, si se desea escalar dicho sistema para que también permita administrar los permisos y políticas de acceso es necesario hablar de **Identity and Access Management (IAM)**. Finalmente, si se necesita de un sistema de mayor complejidad con funciones de administración del ciclo de vida de acceso a los recursos y reportes de qué sucede en la red, **Identity and Access Governance (IAG)** es el modelo a utilizar.

Como se puede ver en la figura 2.12, la cantidad de funciones que cumple el sistema va escalando de izquierda a derecha, pero siempre se encuentra por encima del servicio de directorio, que según el caso podría corresponder a OpenLDAP, Microsoft Active Directory o Samba Active Directory, quienes finalmente son los que se encargan de guardar la información relevante a cuentas de usuario, atributos, credenciales y otros.

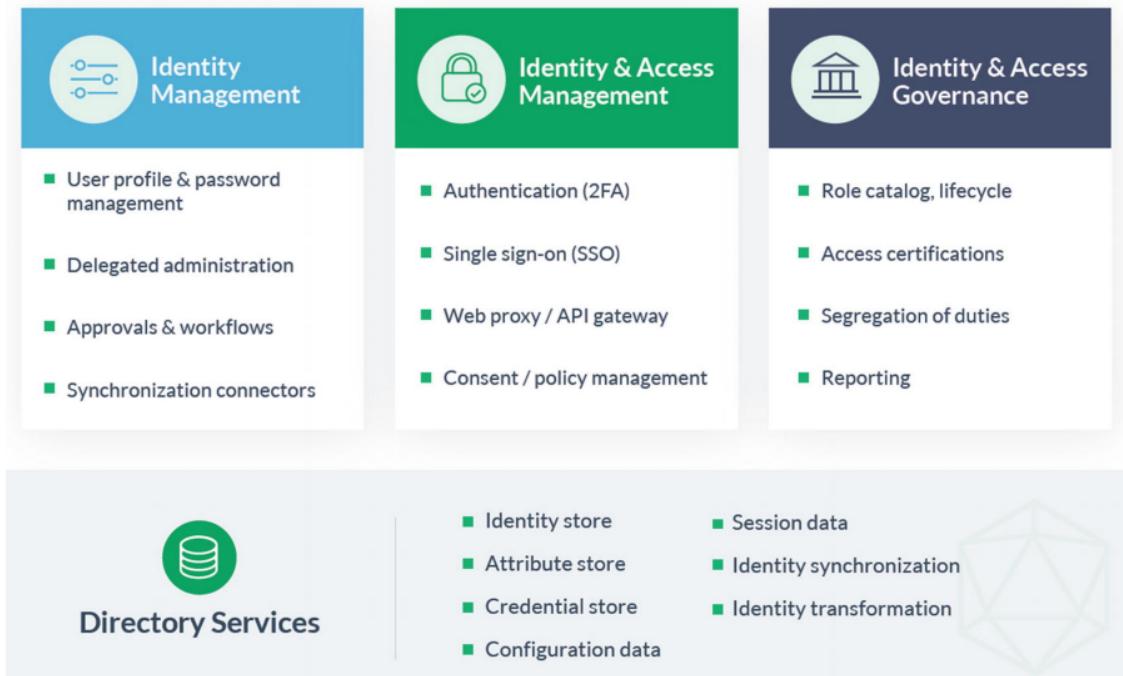


Figura 2.12: Diagrama de componentes de servicios de identidad.  
Fuente: Schwartz & Machulak (2018).

## 2.6 TECNOLOGÍAS DE DESARROLLO

### 2.6.1 Ansible

Ansible es una plataforma *open source* que automatiza los procesos de preparación, configuración, aprovisionamiento y despliegue de *software* (RedHat, 2021a), permitiendo así gestionar y configurar servidores de una forma estándar.

La forma de automatizar estos procesos es mediante el uso de **Playbooks**, estos son un conjunto de tareas que le indican al motor lo que debe realizar en las máquinas especificadas en el **Inventario**, un archivo que contiene los nombres de dominio o direcciones IPs de los *host* a configurar. Su ejecución ocurre conectándose a los distintos nodos y enviando pequeños programas llamados **Módulos**, Ansible envía todos los módulos necesarios para la ejecución de las tareas que contenga un *playbook*. Una vez esto ha sido realizado, se conecta mediante el protocolo SSH para realizar el aprovisionamiento y las configuraciones necesarias, al momento de terminar remueve todos los módulos que haya enviado. La gran mayoría de los documentos de Ansible son escritos en formato YAML<sup>5</sup>.

<sup>5</sup><https://yaml.org/>

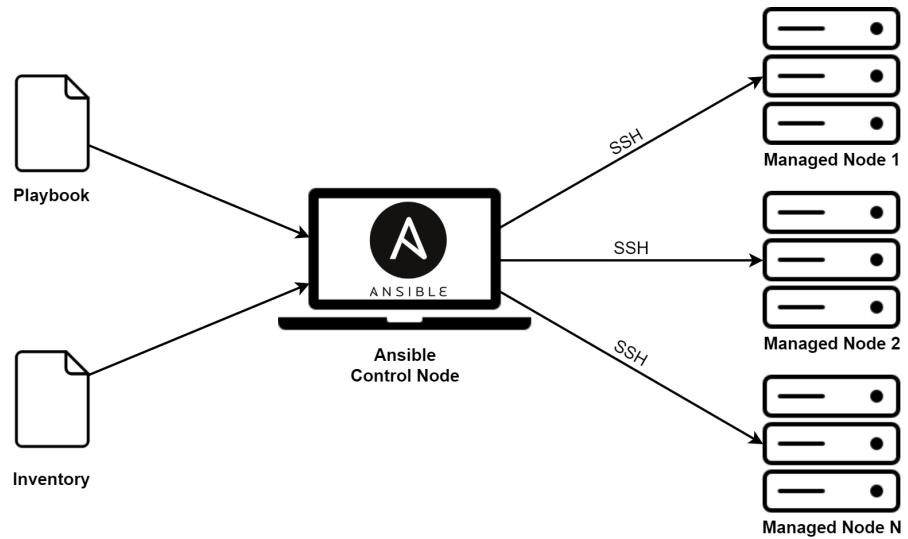


Figura 2.13: Diagrama de funcionamiento de Ansible.

Fuente: Elaboración propia 2021.

La figura 2.13 muestra la interacción de los componentes previamente descritos. El **nodo de control** corresponde a cualquier máquina que tenga Ansible instalado, mientras que los **nodos administrados** pueden incluir computadores de escritorio, servidores, máquinas virtuales u otros tipos de dispositivos.

## 2.6.2 Docker

Docker es una plataforma de software que permite la creación, empaquetamiento y ejecución de aplicaciones mediante el uso de **contenedores**, estos incluyen todo lo necesario para ejecutar el *software* y así permitir su fácil transporte entre distintas máquinas.

Un contenedor es un formato de empaquetado en el que se incluye todo el código y las dependencias de una aplicación (RedHat, 2021b), por lo que dentro se puede encontrar bibliotecas, herramientas de sistema, variables de entorno, entre otros. Los contenedores suelen ser comparados con las máquinas virtuales, no obstante, los primeros están diseñados para ser más livianos e independientes.

Como se puede ver en la figura 2.14, el *stack* de la izquierda solo contiene un sistema operativo sobre el que se ejecuta Docker, de esta forma es posible desplegar múltiples contenedores en la misma máquina quienes a su vez comparten el Kernel del sistema operativo, cada uno corriendo como un proceso aislado (Docker Inc., 2021). Gracias a estas características, los contenedores típicamente utilizan menos espacio, requieren menos recursos y su despliegue es más rápido.

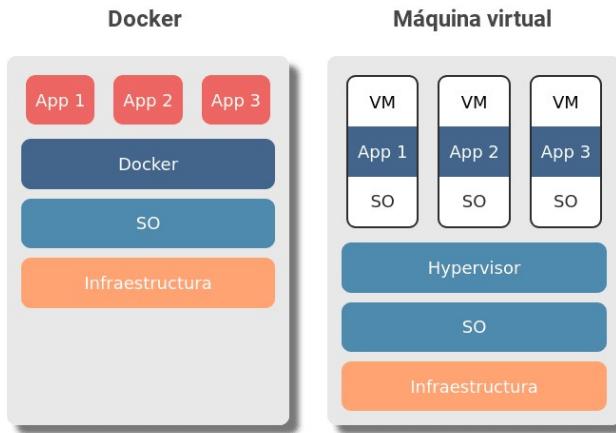


Figura 2.14: Comparación entre el despliegue de aplicaciones mediante contenedores y mediante máquinas virtuales.

Fuente: Zepeda (2020).

Los contenedores son construidos a partir de **imágenes**, estas corresponden a archivos que contienen una captura del estado de un contenedor. Técnicamente, una imagen actúa como un conjunto de instrucciones para construir un contenedor, es decir, una plantilla (Gillis, 2021). Bajo el mismo concepto existe el **Dockerfile**, que es un archivo que contiene una serie de instrucciones para construir una imagen. Usualmente, las imágenes son construidas a partir de otras, aunque es posible encontrar en la documentación oficial formas de crear imágenes desde cero.

## CAPÍTULO 3. DESARROLLO

Esta sección corresponde al reporte y documentación sobre el trabajo realizado para cumplir con cada uno de los objetivos específicos establecidos previamente, por lo que se divide en siete subsecciones en donde se describen los hitos mas importantes ocurridos.

Se tomó la decisión de realizar un mes de trabajo de investigación y tres meses de implementación, por lo que el primer mes estuvo destinado solo al objetivo número 1 mientras que los objetivos 2 al 7 fueron desarrollados en los siguientes tres meses.

### 3.1 INVESTIGACIÓN, COMPARACIÓN Y EVALUACIÓN DE PRODUCTOS DE SOFTWARE

El primer objetivo corresponde a la investigación sobre las tecnologías actuales y los productos de *software* que son utilizados, con el fin de identificar posibles opciones a utilizar en el sistema centralizado. Esto incluye una comparación entre los productos con el fin de distinguir que funciones proveen a diferencia de lo que puede hacer Samba *Active Directory* por si solo.

#### 3.1.1 Criterios de comparación

Para la comparación de los *software* se establecieron trece criterios, los cuales están divididos en cuatro secciones relacionadas a distintos tipos de características. Además, se determinó que los productos a escoger como mínimo deben cumplir los requisitos básicos de ser gratuitos y *open source* para ser llevados a una mayor comparación.

- **Características base**

- Gratis: requerimiento de base, no se busca utilizar productos que sean de pago.
- Open Source: el *software* debe ser código libre, con el fin de poder observar detalladamente su código fuente si es necesario, además de dar la opción de realizar modificaciones.
- Vigente: que se aprecie que el *software* está en constante desarrollo, que los repositorios cuentan con actividad, actualizaciones constantes y que se pueda ver un soporte/comunidad activa actualmente.

- Documentación: una buena documentación en la que se pueda encontrar extensa información con respecto a la instalación y funcionalidades que cubre el *software*, acompañado de diagramas, imágenes y ejemplos explicativos. Idealmente que se encuentren todas las funcionalidades documentadas.

- **Características de *deploy***

- Compatibilidad con Ubuntu: la capacidad de ser desplegado en una máquina con sistema operativo Ubuntu, el cual es el sistema operativo utilizado actualmente en los servidores del departamento.
- Compatibilidad con Docker: la capacidad de ser desplegado en uno o varios contenedores en los servidores de forma nativa (que la opción sea ofrecida por el *software*).

- **Características de interacción**

- Ofrecer CLI: posibilidad de interactuar con el producto mediante línea de comandos (*Command Line Interface*).
- Ofrecer GUI: posibilidad de interactuar con el producto mediante una interfaz, idealmente de tipo Web (*Graphic User Interface*).
- Ofrecer API REST: que el *software* tenga puntos de acceso para que otras aplicaciones puedan realizar consultas directamente.

- **Características relacionadas a protocolos**

- Proveer protocolo SAML: capacidad de ofrecer autenticación de usuarios por SAML para aplicaciones externas.
- Integración de SSO: que venga integrado un sistema para establecer *Single Sign On* en el producto.
- Proveer protocolo OAuth: capacidad de ofrecer autenticación de usuarios por el protocolo OAuth, delegando así el proceso a otras entidades como por ejemplo Google.
- Utilización de directorio LDAP: que el manejo de usuarios y grupos sea mediante una base de datos que se comunique mediante el protocolo LDAP, lo que permite hacer sincronización con otros directorios del mismo tipo.

### 3.1.2 Productos de *software* escogidos

Luego de realizar búsquedas en sitios Web como Gartner<sup>1</sup> y SolutionsReview<sup>2</sup> enfocadas en la popularidad de variados servicios de directorio, se escogieron los siguientes 5 productos que cumplen con ser gratuitos y Open Source.

- **Gluu** de Gluu Inc.
- **FreeIPA** de RedHat
- **OpenIAM** de OpenIAM Inc.
- **Syncopé** de Apache
- **Keycloak** de RedHat

### 3.1.3 Proceso de evaluación

La asignación de valores para cada una de estas características está normalizada entre 0 y 1 con excepción de la vigencia y la documentación. Los valores se obtienen de leer la página oficial para tener una idea general de qué hace el producto, la documentación del mismo con el fin de identificar sus funcionalidades y finalmente de hacer una instalación de cada uno de los *software* y verificar su funcionamiento en un entorno de prueba.

#### *Características base*

De la evaluación de las características base se obtiene la tabla 3.1, es necesario mencionar que para asignar los se utilizaron otros criterios específicos para calcularlos, los que fueron seleccionados en base a la necesidad e importancia que aportan al sistema final, la información completa se encuentra en el anexo A.

C. Base	Gluu	FreeIPA	OpenIAM	Syncopé	Keycloak
Vigencia	0,9	0,7	0,8	0,5	1
Documentación	0,9	0,4	0,7	0,9	0,9
<b>Total</b>	90,0%	55,0%	75,0%	70,0%	95,0%

Tabla 3.1: Evaluación de características base.

Fuente: Elaboración propia (2021).

<sup>1</sup><https://www.gartner.com/reviews/market/access-management>

<sup>2</sup><https://solutionsreview.com/identity-management/the-10-best-free-and-open-source-identity-management-tools/>

### *Características de deploy*

Los resultados de la evaluación de las características de deploy se pueden observar en la tabla 3.2. Aquí se encontró que en relación a la compatibilidad con Ubuntu los 5 productos están diseñados para funcionar en sistemas UNIX, por lo que son compatibles con este sistema operativo de forma nativa con excepción de FreeIPA, el cual fue retirado del repositorio de Ubuntu posterior a la versión 16.04 y se encuentra disponible solo para Fedora y RedHat Enterprise Linux. En relación a Docker, los productos Gluu, OpenIAM, Syncpe y Keycloak cuentan con documentación de instalación y uso mediante contenedores, mientras que FreeIPA hace mención de que sólo es un ambiente de pruebas para verificar la utilidad del sistema y que no debería ser usado en ambientes de producción.

C. Deploy	Gluu	FreeIPA	OpenIAM	Syncpe	Keycloak
Compatibilidad Ubuntu	1	0	0	1	1
Compatibilidad Docker	1	0	1	1	1
<b>Total</b>	100,0%	0,0%	50,0%	100,0%	100,0%

Tabla 3.2: Evaluación de características de deploy.

Fuente: Elaboración propia (2021).

### *Características de interacción*

La evaluación de las características de interacción da como resultado la tabla 3.3. Se observó que la interfaz de línea de comandos de Gluu no permite hacer ajustes mayores en el sistema, OpenIAM no hace mención alguna a una CLI dentro de su documentación y sólo hace enfoque en la interfaz Web, en FreeIPA, Syncpe y Keycloak es posible trabajar mediante la línea de comandos ya sea para la configuración o posterior administración. Gluu, OpenIAM, Syncpe y Keycloak poseen una buena interfaz Web para configuración y administración plenamente documentada, no así con FreeIPA que no ofrece este servicio. En relación a la API REST, Gluu, FreeIPA, OpenIAM y Keycloak poseen una amplia documentación sobre la existencia y uso de la API que ofrecen. En este apartado se puede ver una amplia ventaja de Keycloak al cumplir con los 3 requerimientos.

C. Interacción	Gluu	FreeIPA	OpenIAM	Syncpe	Keycloak
Ofrecer CLI	0,5	1	0	1	1
Ofrecer GUI	1	0	1	1	1
API REST	1	1	1	0	1
<b>Total</b>	83,3%	66,6%	66,6%	66,6%	100,0%

Tabla 3.3: Evaluación de características de interacción.

Fuente: Elaboración propia (2021).

### *Características relacionadas a protocolos*

La tabla 3.4 muestra la capacidad de los *software* de proveer los protocolos listados. Los únicos que cumplen al 100% con ofrecer estas funcionalidades son Gluu y Keycloak, FreeIPA y Syncpe están diseñados para otros casos de uso por lo que no ofrecen OAuth o SSO, mientras que OpenIAM se queda atrás al no tener ningún tipo de sincronización con LDAP.

C. Protocolos	Gluu	FreeIPA	OpenIAM	Syncpe	Keycloak
Protocolo SAML	1	1	1	1	1
Protocolo OAuth	1	0	1	0	1
Integrar SSO	1	0	1	0	1
Directorio LDAP	1	1	0	1	1
<b>Total</b>	100,0%	50,0%	75,0%	50,0%	100,0%

Tabla 3.4: Evaluación de características relacionadas a protocolos.

Fuente: Elaboración propia (2021).

### *Resultado final*

Finalmente, al ponderar todos los valores conseguidos anteriormente se obtiene el la tabla 3.5, la cual contiene el porcentaje final para cada uno de los productos. El porcentaje al que se llega corresponde a qué tanto el *software* cumple con once de los trece puntos listadas inicialmente, los dos que se dejan fuera son los requisitos mínimos de gratuidad y *open source*.

	Gluu	FreeIPA	OpenIAM	Syncpe	Keycloak
<b>Porcentaje Final</b>	93,3%	42,9%	66,6%	71,6%	98,7%

Tabla 3.5: Resultado final de la evaluación.

Fuente: Elaboración propia (2021).

Debido a que Gluu y Keycloak poseen muy buenos porcentajes finales, se tomó la decisión de realizar pruebas con ambos para los siguientes objetivos específicos y así verificar en la práctica cual se adecúa mejor a la situación deseada. La idea final es mantener a Samba como el servicio de directorio y ofreciendo interoperabilidad entre Windows y Linux, agregando a uno de los *software* ofreciendo así prototipos de autenticación para las aplicaciones Web y móviles.

## **3.2 REDISEÑO DEL DIRECTORIO Y FORMATO DE CREDENCIALES**

En este objetivo se busca primero implementar una nueva forma de ordenar los integrantes del DIINF dentro del directorio, como también integrar un nuevo formato para las credenciales de usuario.

### 3.2.1 Árbol del directorio

#### Diseño inicial

Para implementar el primer diseño del árbol se consideraron cuatro herramientas previamente mencionadas en la sección 2.4, las cuales son el dominio, la unidad organizativa, el grupo y el usuario. Se hizo un primer estimado con respecto a qué unidades del departamento requieren estar ingresadas y se llegó a un primer diseño del diagrama del DIINF, correspondiente a la figura 3.1.

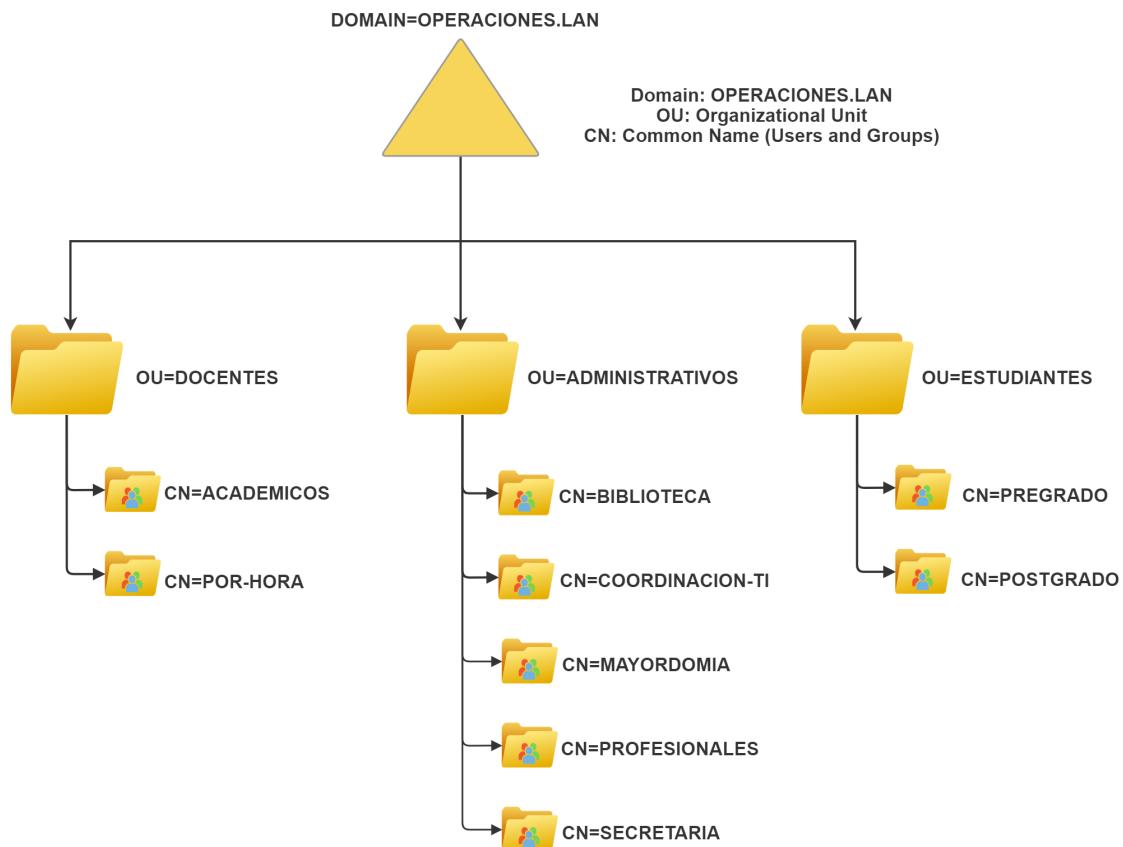


Figura 3.1: Diseño inicial del árbol del directorio.

Fuente: Elaboración propia (2021).

Como se puede apreciar, se separó el departamento en tres unidades organizativas correspondiente a docentes, administrativos y estudiantes, dentro de cada unidad se encuentran los grupos específicos a los que puede pertenecer un usuario de dicha unidad. Este diagrama se diseño de tal forma que primero los usuarios fuesen guardados dentro de sus unidades organizativas correspondientes y luego se les agregara los grupos que fuesen necesarios dependiendo de su situación dentro del departamento.

### Diseño final

Durante el desarrollo de otros objetivos se notaron dos problemas con el diseño planteado previamente, lo que generó una segunda versión del diagrama.

El primer problema tiene relación con los casos en donde un integrante del departamento pertenece a más de una unidad organizativa, Samba Active Directory no permite que un usuario creado dentro de una unidad organizativa pertenezca a un grupo de otra unidad, por ejemplo, un estudiante creado en la unidad "OU=ESTUDIANTES" y perteneciente al grupo "CN=PREGRADO", si trabaja a medio tiempo en el área de operaciones no puede pertenecer al grupo "CN=COORDINACIÓN-TI" que se encuentra en la unidad organizativa "OU=ADMINISTRATIVOS". El segundo problema considera el caso en donde se desee administrar otro sector de informática que no necesariamente esté fusionado con el departamento, como por ejemplo CITIAPS.

Una vez considerados estos problemas y las opciones que provee Samba Active Directory, se creó el segundo diagrama que se puede apreciar en la figura 3.2.

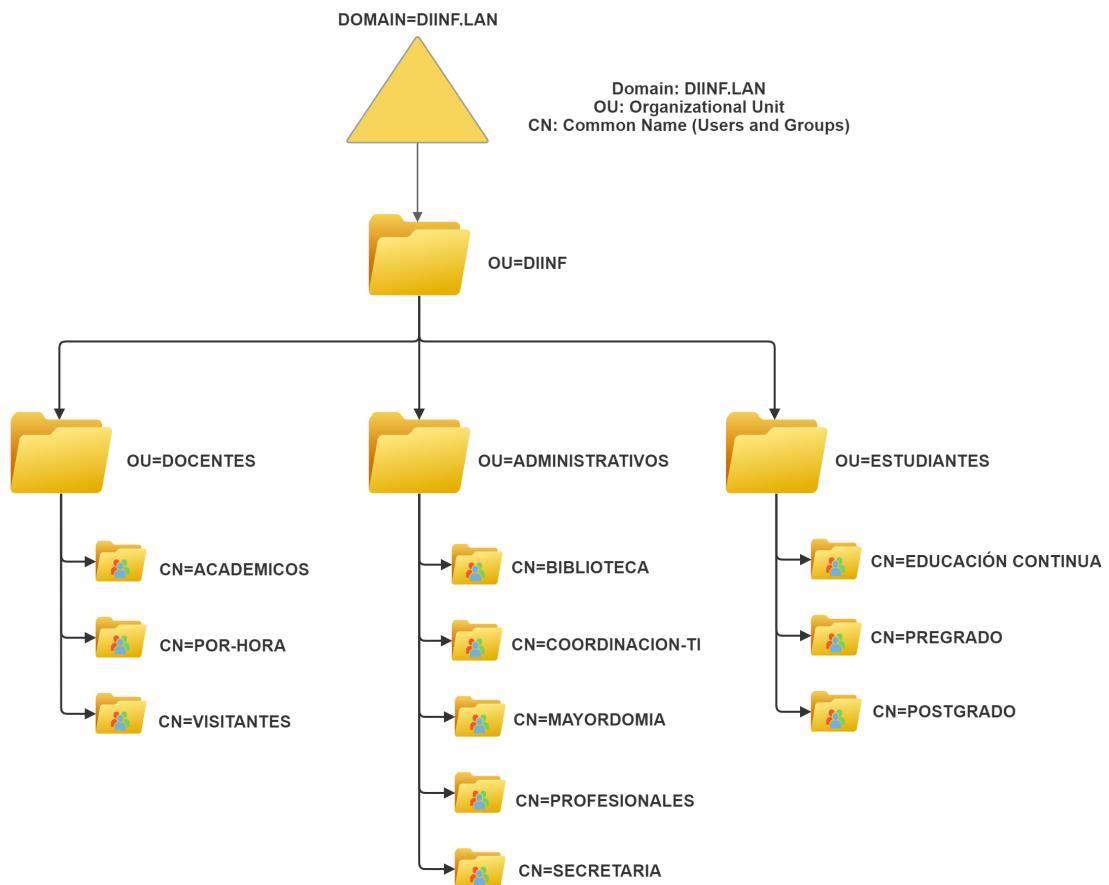


Figura 3.2: Diseño final del árbol del directorio.

Fuente: Elaboración propia (2021).

Este nuevo diseño contempla que la creación de los usuarios se realice dentro de la unidad "OU=DIINF", permitiendo así que posteriormente a una cuenta de usuario se le asigne los grupos a los que pertenece sin estar limitado por su unidad organizativa, pero manteniendo la capacidad de establecer políticas a las distintas sub-unidades según sea necesario. Además, establecer otra unidad organizativa al tope permite mayor flexibilidad al árbol, de esta forma y siguiendo la situación previamente descrita, si fuese necesario administrar CITIAPS se puede agregar otra unidad organizativa al mismo nivel de la unidad "OU=DIINF" llamada "OU=CITIAPS", la cual tendría su propio diseño de unidades y grupos.

### **3.2.2 Formato de credenciales**

#### *Primer inicial*

Como otro de los objetivos específicos de este trabajo contempla la delegación de la autenticación a Google mediante el correo USACH, en primera instancia se tomó la decisión de utilizar el correo completo como nombre de usuario en Samba, donde el formato corresponde a "nombre.apellido@usach.cl".

#### *Formato final*

Más adelante se notó que integrar el "@usach.cl" dentro del nombre de usuario produce un problema al intentar iniciar sesión en el sistema operativo Windows, ya que considera que debe realizar una búsqueda del usuario "nombre.apellido" dentro del dominio "usach.cl", mientras que realmente los usuarios se encuentran dentro del dominio "diinf.lan".

Debido a la situación anterior pero con el objetivo de mantener un formato similar al del correo USACH para facilitar la memorización de credenciales, se tomó la decisión de solo usar la parte de "nombre.apellido" como nombre de usuario en Samba.

## **3.3 INTEGRACIÓN DE INTERFAZ WEB**

Durante el proceso de desarrollo se probaron tres interfaces Web, en donde destacó la interfaz de Keycloak por tener la mayor cantidad de funciones y madurez.

### 3.3.1 phpLDAPadmin

Esta es una interfaz Web que lleva ya muchos años en el mercado y tal como dice en su nombre, es una herramienta escrita en PHP diseñada para administrar servidores de directorio LDAP. Permite múltiples funciones para crear, copiar, editar o borrar entradas en el árbol, además de manejo de políticas de contraseña.

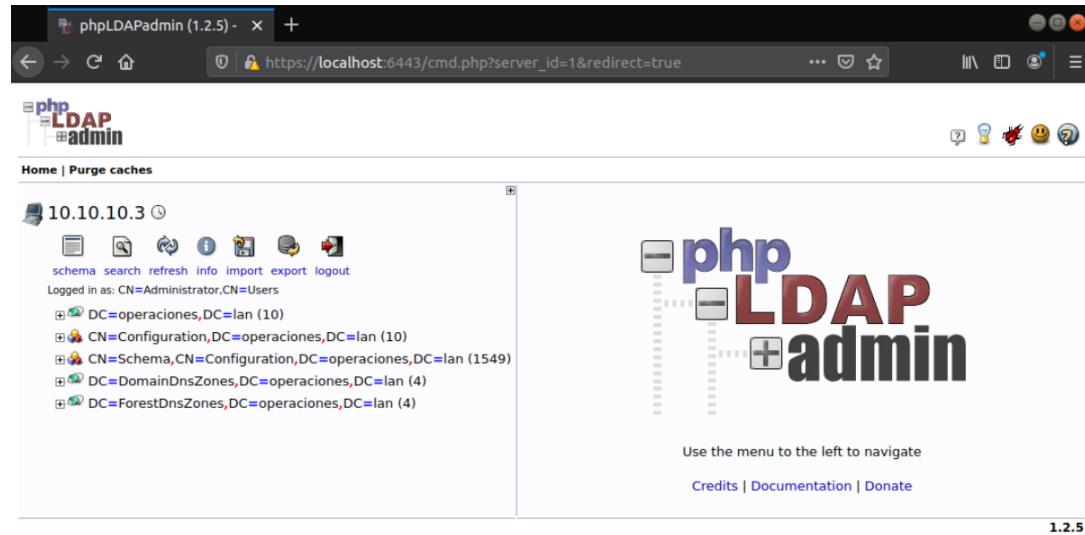


Figura 3.3: Interfaz inicial de phpLDAPadmin.  
Fuente: Elaboración propia (2021).

Si bien phpLDAPadmin ofrece una forma rápida y fácil de integrar una interfaz Web para la administración, los problemas de compatibilidad con Samba *Active Directory* se manifiestan rápidamente. En primer lugar, phpLDAPadmin está diseñado para trabajar con árboles LDAP genéricos, lo que permite compatibilidad con Samba 3 (inclusive phpLDAPadmin por defecto incluye configuraciones para Samba 3) pero no con Samba 4, ya que esta versión posee un esquema modificado específicamente diseñado para asimilarse lo más posible a Microsoft *Active Directory*. En segundo lugar, phpLDAPadmin para conectarse a un servidor requiere la utilización del concepto de *Anonymous Bind*<sup>3</sup>, que corresponde a la realización de una operación de *Bind* al servidor LDAP sin la necesidad de presentar un *Distinguished Name* o una contraseña. Samba *Active Directory* se instala con la opción de permitir *Anonymous Bind* desactivada, debido a que permite el acceso al directorio sin la necesidad de autenticación, terminando en poca seguridad en la integridad de la información guardada.

<sup>3</sup><https://ldapwiki.com/wiki/Anonymous%20bind>

### 3.3.2 Gluu y oxTrust

Gluu provee su propia interfaz de administración la que tiene el nombre de oxTrust<sup>4</sup>, esta permite la administración de cuentas de usuario, grupos y también la configuración de la gran mayoría de las funciones que ofrece (algunas necesitan modificaciones directas en código).

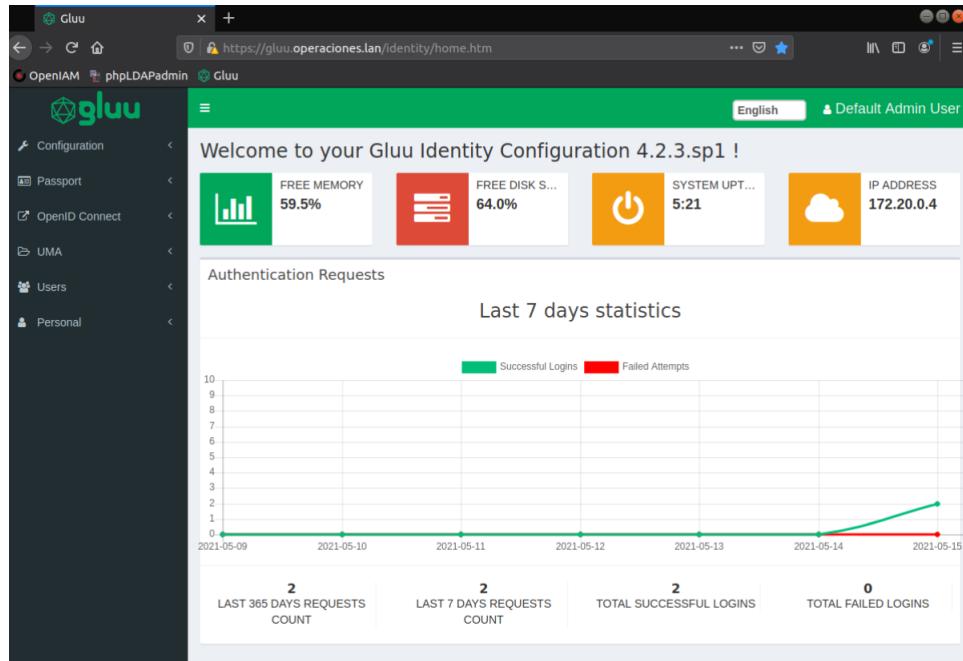


Figura 3.4: Interfaz inicial de Gluu.

Fuente: Elaboración propia (2021).

Si bien la interfaz de Gluu permite la administración sin mayores dificultades deja mucho que desear en el aspecto visual, especialmente en los errores de formato que se pueden encontrar en muchas de las vistas de configuración y en la falta de una mejor implementación del diseño *responsive*. Por otro lado, falla en ofrecer una forma de administrar el servidor LDAP a través de oxTrust, esto se debe a una decisión de diseño en Gluu de la que se habla más en profundidad en la sección 3.4.

### 3.3.3 Keycloak, consola de administración y de usuarios

Keycloak al igual que Gluu provee su propia interfaz Web de administración en la que se puede administrar cuentas de usuario, grupos, políticas de autenticación y aplicaciones,

<sup>4</sup><https://gluu.org/docs/gluu-server/4.0/admin-guide/oxtrust-ui/>

así como también la configuración relacionada a federación de usuarios, proveedores de identidad externos, localización, temas, etc.

Como primera diferencia se puede apreciar una interfaz Web mucho más madura y completa, con más opciones de configuración sin la necesidad de realizar modificaciones en archivos de texto o en código. Por otro lado, el diseño de la federación de usuarios en Keycloak es tratado de forma distinta a Gluu, gracias a esto es posible realizar la administración de usuarios de un servidor LDAP a través de esta interfaz Web.

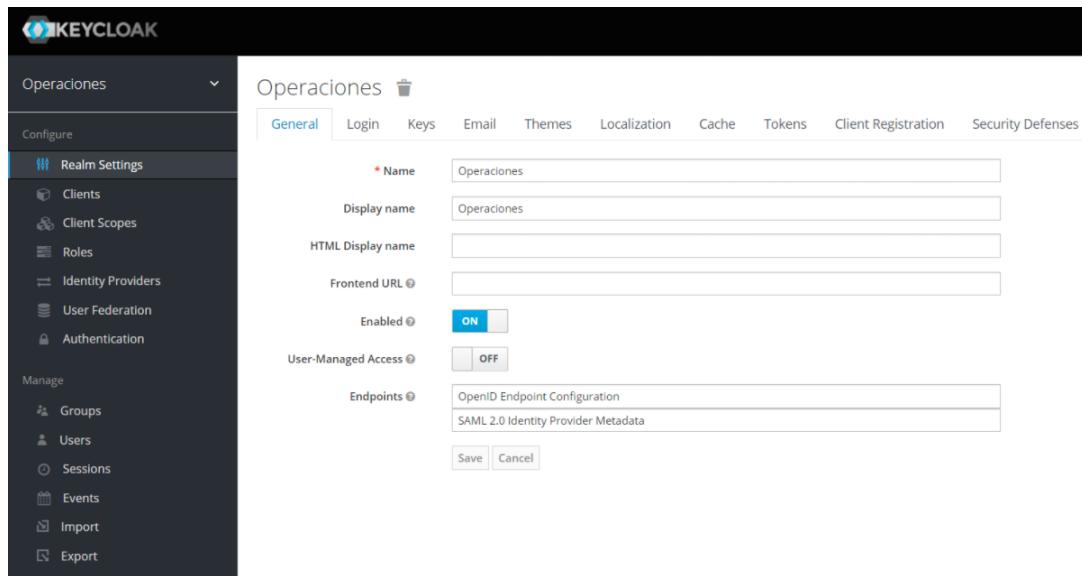


Figura 3.5: Interfaz inicial de Keycloak.  
Fuente: Elaboración propia (2021).

Otro función importante que provee Keycloak por defecto es una interfaz Web de autoservicio<sup>5</sup> como se puede ver en la figura 3.6. En esta es posible modificar la información personal como nombre o apellido, cambiar la contraseña de la cuenta, integrar un segundo factor de autenticación u observar la actividad de la cuenta en los dispositivos en donde se haya iniciado sesión.

Uno de los pocos puntos bajos que se le encontró a esta interfaz tiene relación con la navegación a través de las distintas categorías. Keycloak provee de tantas opciones de administración, integración y configuración que algunas de las configuraciones pueden resultar complejas de encontrar en primera instancia al estar ocultas bajo varios *clicks*.

Finalmente Keycloak es quien provee la interfaz de administración que se acomoda mejor a las necesidades de este proyecto, además de agregar una interfaz para cuentas de usuario sin configuraciones extra. Por todo lo anterior, se tomó la decisión de utilizar esta interfaz Web para la administración del directorio y los demás servicios.

<sup>5</sup>[https://www.keycloak.org/docs/latest/server\\_admin/#\\_account-service](https://www.keycloak.org/docs/latest/server_admin/#_account-service)

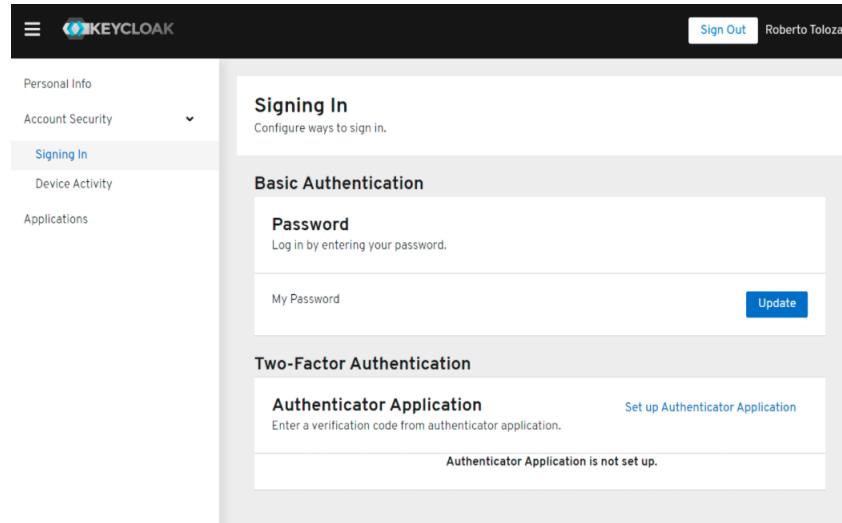


Figura 3.6: Interfaz de cuentas de usuario de Keycloak.

Fuente: Elaboración propia (2021).

### 3.3.4 Localización y *Branding*

#### *Localización*

Keycloak ofrece soporte para 20 idiomas entre los que se encuentra el español. Gracias a esto solamente con cambiar una configuración en la consola de administración es posible cambiar su idioma.

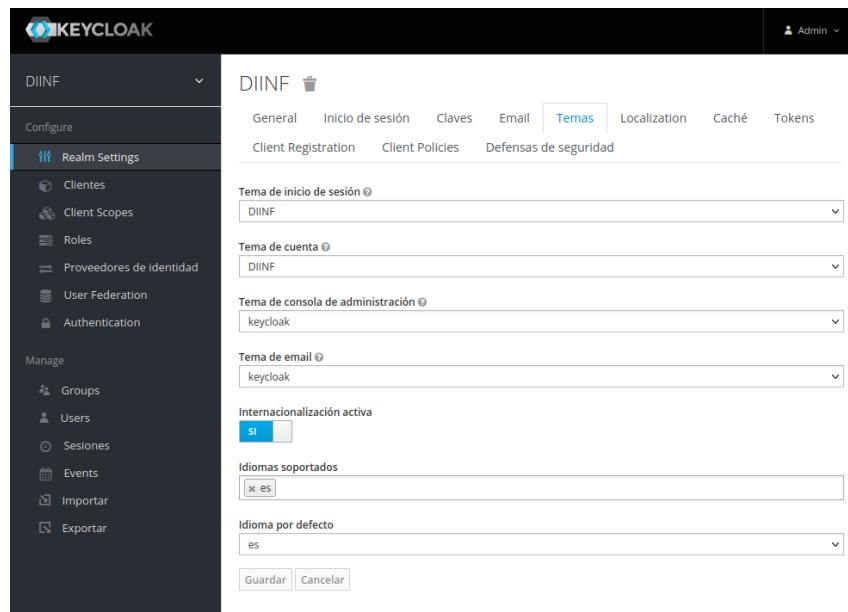


Figura 3.7: Interfaz de Keycloak en español.

Fuente: Elaboración propia (2021).

Si bien el cambio de idioma de la consola de administración es simple, ocurre una situación completamente distinta con la interfaz de autoservicio para usuarios. En septiembre del 2020 Keycloak integró una nueva versión para la consola de cuentas<sup>6</sup>, la que está diseñada como una *single page* en React y PatternFly 4, esta nueva consola aún no tiene una traducción al español oficial en la versión 14.0.0. Para solucionar esto, se investigó el modelo con el que se aplican las traducciones en Keycloak y se tomó la decisión de hacer una traducción propia al español. La implementación se realizó con éxito y la información relevante a los archivos modificados y el proceso se encuentra en el anexo C.

#### *Branding*

Keycloak permite la aplicación de temas a distintos apartados del software<sup>7</sup>, en donde se encuentra la interfaz de *login*, la interfaz de autoservicio para usuarios, la consola de administración y al diseño de los *emails*. Cada una de estas vistas está separada en su propia carpeta que contiene los archivos HTML, CSS, JavaScript, imágenes y otros elementos necesarios.

Para el alcance de este prototipo se tomó la decisión de sólo modificar la interfaz de la consola de cuentas de usuario y la vista de *login*, ya que son las interfaces que ve el usuario final como los estudiantes o académicos del departamento, manteniendo el aspecto por defecto de la consola de administración y de correos. La forma de modificación del tema se puede encontrar en el anexo B.

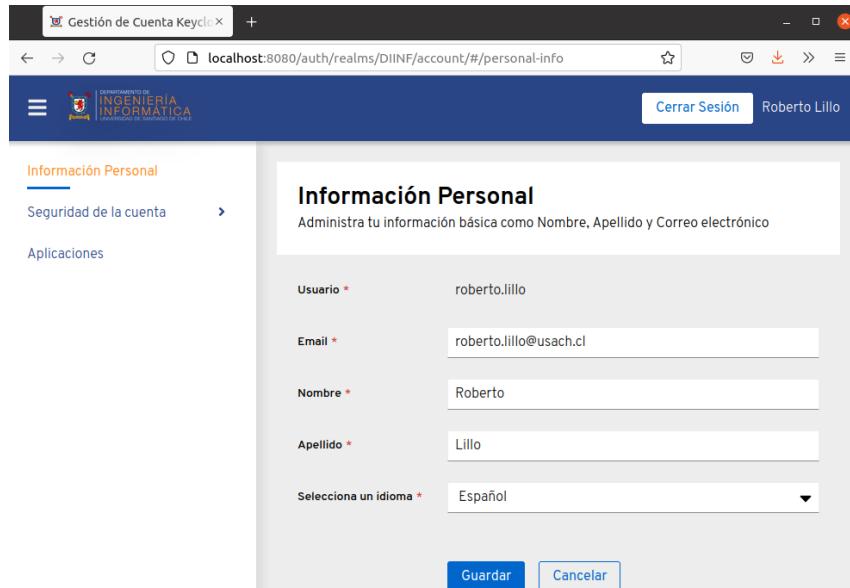


Figura 3.8: Vista de cuenta de usuario con el tema DIINF aplicado en Keycloak.  
Fuente: Elaboración propia (2021).

<sup>6</sup><https://www.keycloak.org/2020/09/new-account-console.adoc>

<sup>7</sup>[https://www.keycloak.org/docs/latest/server\\_admin/#\\_themes](https://www.keycloak.org/docs/latest/server_admin/#_themes)

## 3.4 INTEGRACIÓN DE PROTOCOLOS

La integración de nuevos protocolos de autenticación a Samba mediante el *software* que sea escogido tiene el fin de utilizar el registro de usuarios en el servicio de directorio para iniciar sesión en distintas aplicaciones Web o móviles. Al igual que con el objetivo de buscar una interfaz Web, se realizaron pruebas con los productos Gluu y Keycloak para determinar cuál ofrece mayor ventaja.

En la etapa inicial de investigación se determinó que Gluu y Keycloak ofrecen el protocolo OAuth, no obstante, posteriormente se encontró que lo que ofrecen realmente es *OpenID Connect*, esto es una muy buena noticia ya que esta capa extra permite trabajar con autenticación y autorización a la vez.

A pesar de que ambos productos de *software* tienen la capacidad de ofrecer autenticación mediante OIDC y SAML, se decidió sólo trabajar con OIDC debido a dos razones: la primera es por considerar a JSON como un formato más moderno que XML y la segunda para así mantener un solo estándar dentro del departamento.

Con el protocolo a integrar ya determinado sólo queda establecer la forma de conexión entre Samba y el producto que corresponda. Gluu provee un sistema de sincronización llamado **AD/LDAP Synchronization**<sup>8</sup> o también conocido como **Cache Refresh**, mientras que Keycloak por su parte ofrece el servicio llamado **User Storage Federation**<sup>9</sup>. Si bien tienen nombres distintos, ambos sistemas cumplen la función de sincronizar su base de datos con un servidor LDAP/AD.

### 3.4.1 Gluu y *Cache Refresh*

Gluu provee su propia base de datos de tipo LDAP en la que guarda todos los usuarios, grupos u objetos que sean creados en el sistema. La sincronización mediante *Cache Refresh* permite conectar uno o más servicios de directorio LDAP o Microsoft Active Directory a Gluu, guardando la información que sea extraída de estos servidores en su base de datos para tener acceso rápido cuando sea necesario.

La sincronización realizada frente al servicio de directorio funciona en modo de edición "sólo lectura", es decir, sólo se puede leer la información del servidor para hacer una copia local en Gluu, cualquier modificación que se haga en los atributos de esta copia local no será escrito en el servidor LDAP desde donde se realizó la sincronización.

---

<sup>8</sup><https://gluu.org/docs/gluu-server/4.0/user-management/ldap-sync/>

<sup>9</sup>[https://www.keycloak.org/docs/14.0/server\\_admin/index.html#\\_user-storage-federation](https://www.keycloak.org/docs/14.0/server_admin/index.html#_user-storage-federation)

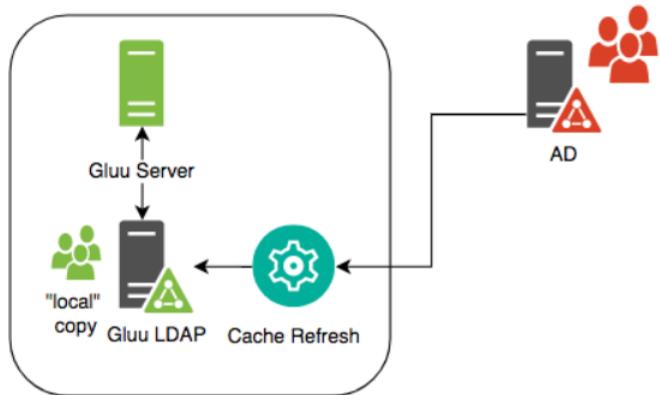


Figura 3.9: Sincronización de usuarios mediante *Cache Refresh* en Gluu.  
Fuente: Chong (2019).

Mediante *Cache Refresh* es posible que los usuarios sincronizados puedan iniciar sesión en las distintas aplicaciones que estén registradas en Gluu, esto tiene un flujo especial debido a que durante la sincronización no se copia ninguna contraseña de usuario. Como se aprecia en el flujo ilustrado en la figura 3.10, primero el usuario intenta autenticarse frente a Gluu (1), luego Gluu verifica si es que el usuario existe en su copia local sincronizada (2) y en el caso de que si exista, realiza la petición de autenticación frente al servicio de directorio con el usuario y contraseña ingresados.

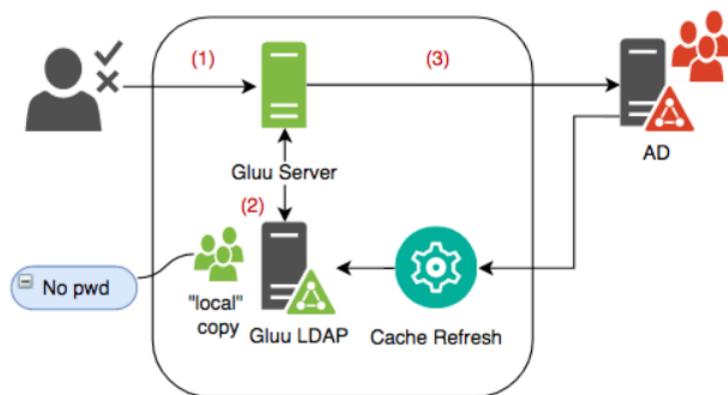


Figura 3.10: Flujo de autenticación de un usuario sincronizado por *Cache Refresh* en Gluu.  
Fuente: Chong (2019).

Finalmente, si bien esta funcionalidad permite la autenticación centralizada de usuarios, se descartó su uso al no permitir la administración del directorio desde el servidor de Gluu, tal como se mencionó previamente en la sección 3.3.2.

### 3.4.2 Keycloak y *User Storage Federation*

En muchos aspectos la sincronización realizada mediante *User Storage Federation* es similar a *Cache Refresh* de Gluu, con la fundamental diferencia de que Keycloak permite cambiar el modo de edición entre "sólo lectura", "escritura" o "no sincronizado".

El modo de edición en "escritura" es el que ofrece la mayor utilidad para el sistema, por defecto este permite modificar los atributos: nombre de usuario, correo electrónico, primer nombre y apellido. Es posible agregar atributos extra a esta lista mediante el uso de *mappers*, que son los encargados de hacer el enlace entre el nombre del atributo en la base de datos de Keycloak y el directorio LDAP.

Keycloak utiliza una base de datos SQL para guardar la información del sistema, por lo que toda la información que es recuperada desde el servidor LDAP es guardada en este formato para su posterior uso, así mismo, cualquier modificación que se realice en los atributos es escrita tanto en la base de datos SQL como en el servidor LDAP. Al igual que Gluu, ninguna contraseña es sincronizada y guardada en la base de datos de Keycloak, por lo que el flujo de autenticación es idéntico al descrito previamente para *Cache Refresh* en donde primero se busca si el usuario existe localmente y luego se autentica frente al servidor LDAP.

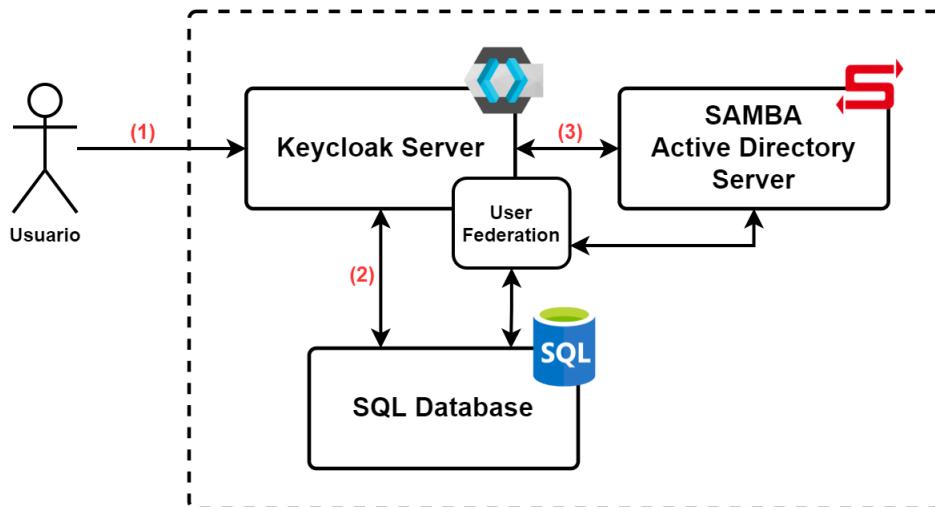


Figura 3.11: Flujo de autenticación de un usuario sincronizado por *User Storage Federation* en Keycloak.

Fuente: Elaboración propia (2021).

Como se puede apreciar en la figura 3.11, la comunicación con Samba es bidireccional, debido a esto se escogió a Keycloak para establecer una autenticación centralizada e integrar el protocolo de *OpenID Connect*. Los detalles específicos de configuración para la sincronización mediante Keycloak se encuentran en el anexo D.

## 3.5 DELEGACIÓN DE AUTENTICACIÓN

Muchos proveedores actuales como Google, Twitter o Github ofrecen un servicio de autenticación para aplicaciones de terceros. Al ser provistos por entidades conocidas permite a los desarrolladores evitar la implementación de su propio sistema de autenticación, como también a los usuarios finales poder utilizar su cuenta de usuario en las aplicaciones que lo soporten. Para poder integrar este modelo de inicio de sesión en una aplicación es necesario integrar un módulo que sea capaz de comunicarse con la entidad que provee el servicio, usualmente también se requiere la generación de un id y *token* secreto que permita asegurar la conexión. Este patrón de autenticación normalmente se conoce como **Social Login** debido a su asociación con las redes sociales, sin embargo la delegación puede ser con cualquier IdP que provea el servicio.

### 3.5.1 Servicio de delegación de Google

Google ofrece un servicio para que aplicaciones de terceros puedan utilizar las cuentas de Gmail para autenticar y autorizar usuarios, este servicio está construido mediante OIDC por lo que la contraseña no es expuesta y se mantiene segura. Una vez el usuario inicia sesión en Google, la aplicación puede utilizar el *ID token* para recuperar información de la cuenta desde los servidores de Google, tal como nombre, correo, imagen de perfil, entre otros.

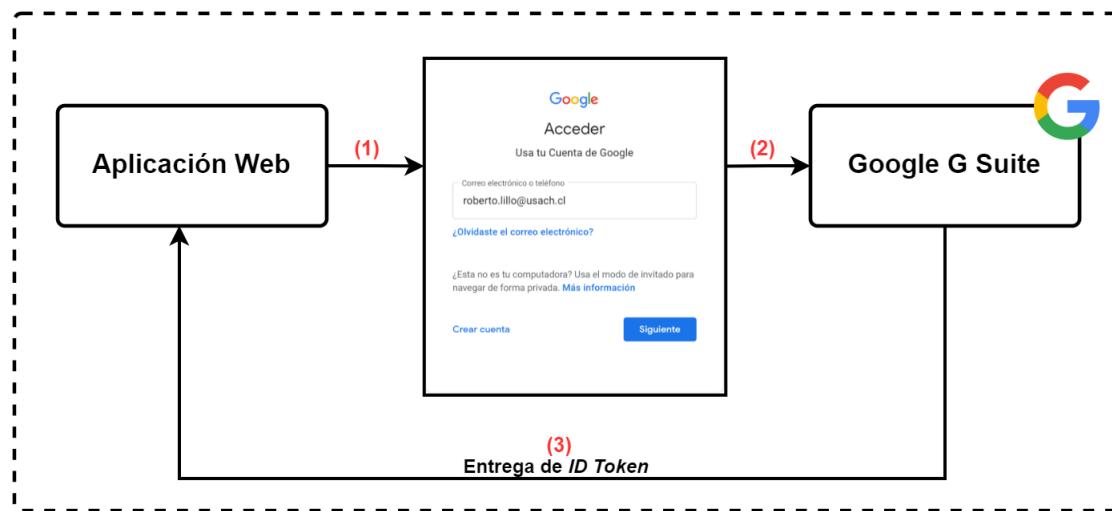


Figura 3.12: Flujo de inicio de sesión en una aplicación Web mediante delegación a Google.  
Fuente: Elaboración propia (2021).

### 3.5.2 Integración de Google al sistema centralizado

Gluu y Keycloak ofrecen los servicios de **Inbound Identity**<sup>10</sup> e **Identity Brokering**<sup>11</sup> respectivamente. Al igual que con los objetivos anteriores se realizaron pruebas con los dos, no obstante, al ser tan similar la configuración y ya haber decidido en los dos objetivos anteriores que Keycloak es el *software* que mejor se adecúa al sistema, se tomó la decisión inmediata de utilizar a Keycloak para cumplir este objetivo.

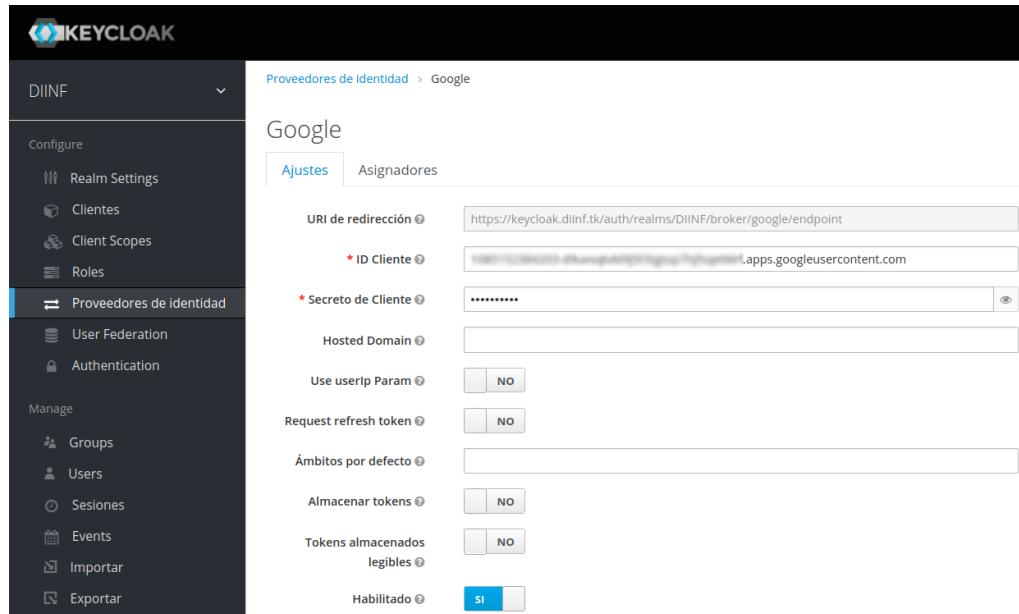


Figura 3.13: Pestaña de configuración de proveedores de identidad en Keycloak.

Fuente: Elaboración propia (2021).

Para poder establecer comunicación con el sistema de Google es necesario obtener credenciales de OAuth desde la consola de **Google Cloud Platform**<sup>12</sup>, esto se puede realizar con cualquier cuenta de Gmail, no obstante, si se utiliza una cuenta institucional del tipo "@usach.cl" es posible limitar el servicio para que sólo puedan iniciar sesión cuentas institucionales que sean parte del dominio, así estableciendo una primera limitación con respecto a los usuarios que pueden ingresar a las aplicaciones del departamento.

Posterior a la obtención de las credenciales de OAuth por parte de Google, es posible configurar Keycloak en el apartado de "Proveedores de identidad": aquí se encuentra una configuración predefinida para Google que sólo necesita el ingreso de las credenciales previamente obtenidas, correspondientes a **clientId** y **clientSecret**, mayor información con respecto a este proceso se encuentra en el anexo E.

<sup>10</sup><https://gluu.org/docs/gluu-server/4.0/authn-guide/inbound-oauth-passport/>

<sup>11</sup>[https://www.keycloak.org/docs/14.0/server\\_admin/index.html#\\_identity\\_broker](https://www.keycloak.org/docs/14.0/server_admin/index.html#_identity_broker)

<sup>12</sup><https://cloud.google.com/>

Al finalizar esta configuración, todo usuario podrá iniciar sesión mediante su correo USACH tanto en la interfaz de cuentas de usuario como en otras aplicaciones que hayan delegado su autenticación a Keycloak, ocurriendo así un flujo de encadenamiento de proveedores de identidad, tal como se puede apreciar en la figura 3.14.

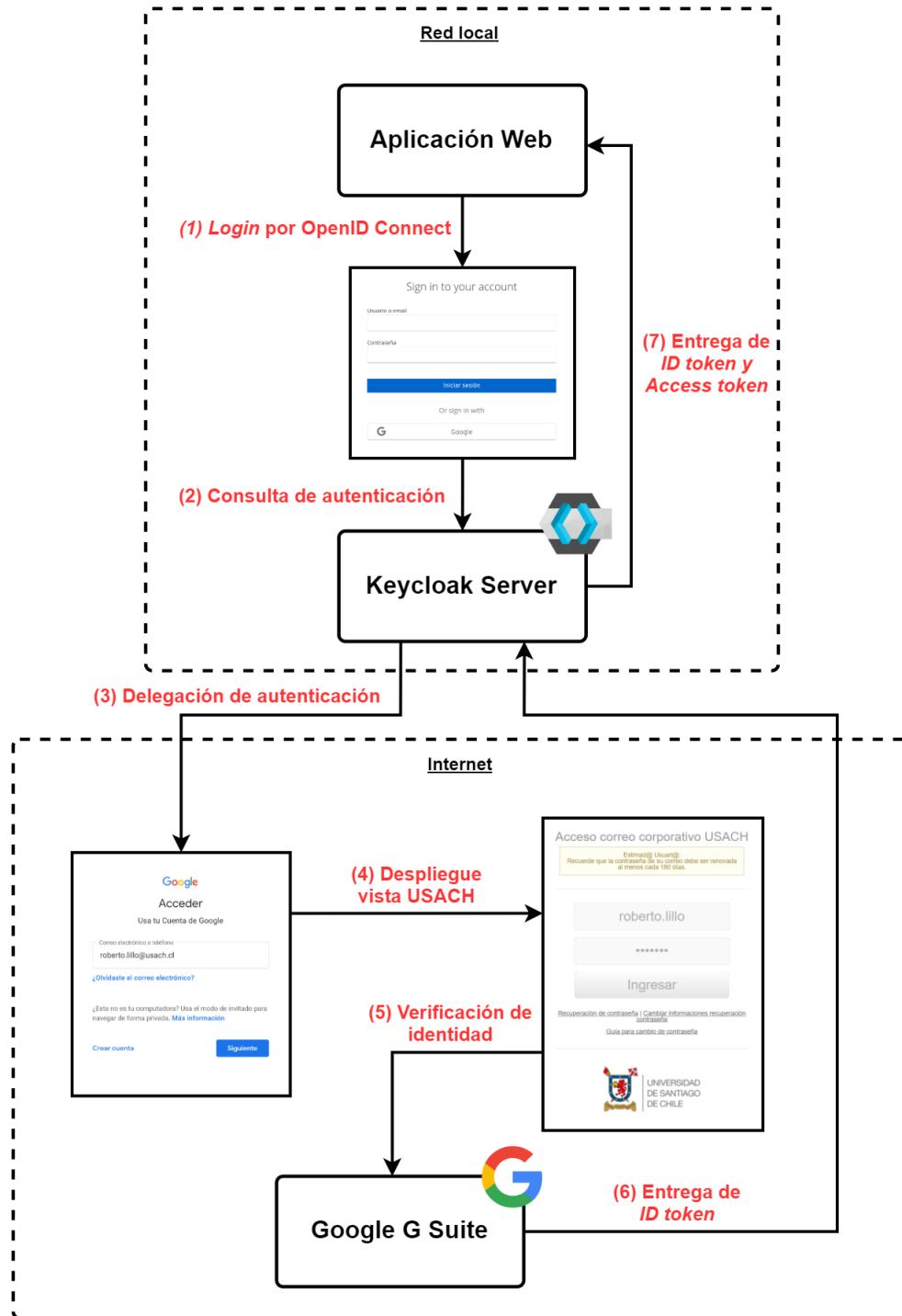


Figura 3.14: Encadenamiento de proveedores de identidad mediante Keycloak y Google.  
Fuente: Elaboración propia (2021).

## 3.6 DESPLIEGUE DEL SISTEMA MEDIANTE DOCKER

La decisión de levantar los servicios de Samba *Active Directory* y Keycloak mediante Docker tiene la intención de simplificar la instalación y configuración de ambos, facilitando el *deploy* del ambiente de pruebas del sistema centralizado y también manteniendo el servidor limpio de paquetes y configuraciones.

### 3.6.1 Contenedor de Samba

El primer servicio a considerar es Samba. Dentro de la documentación oficial no se encuentra ninguna mención con respecto al soporte para levantar un servidor de Samba *Active Directory* mediante contenedores. Durante la mayoría de la fase de implementación se instaló Samba de forma directa en Ubuntu, mientras que cerca de la fase final de automatización de la instalación se escogió utilizar Docker y levantar el servicio mediante un contenedor utilizando una imagen no oficial<sup>13</sup>, esta decisión se tomó al considerar que el sistema a implementar corresponde a un prototipo y se busca rapidez en el despliegue con el fin de realizar pruebas, no obstante, es de vital importancia destacar que si posteriormente este sistema se desea llevar a producción, se debe levantar un servidor de Samba AD como lo indica la documentación.

La imagen escogida requiere definir una gran cantidad de variables para establecer el servicio de directorio, debido a esto se decidió trabajar con un archivo de Docker Compose que contiene la configuración necesaria para levantar el servicio.

### 3.6.2 Contenedor de Keycloak

Entre las formas de instalación que ofrece la documentación de Keycloak se encuentra Docker, la imagen oficial<sup>14</sup> ofrece una amplia variedad de configuraciones para personalizar el deploy y está en constante actualización junto a las nuevas versiones lanzadas. Por defecto, Keycloak incorpora una base de datos SQL llamada H2<sup>15</sup> la cual es muy pequeña y perfecta para ambientes de prueba, no obstante, el sistema tiene soporte para las bases de datos PostgreSQL, MySQL, MariaDB, Oracle y Microsoft SQL. En este caso se tomó la decisión de levantar un contenedor de Keycloak y además un contenedor de PostgreSQL como base de

---

<sup>13</sup><https://hub.docker.com/r/nowsci/samba-domain>

<sup>14</sup><https://hub.docker.com/r/jboss/keycloak>

<sup>15</sup><https://www.h2database.com>

datos para el sistema, esto se debe al conocimiento de PostgreSQL, la facilidad en su despliegue y posterior configuración con Keycloak.

Al igual que con el contenedor de Samba, se implementó un archivo de Docker Compose para levantar los contenedores de Keycloak y PostgreSQL, definiendo las variables necesarias para ambos servicios. Finalmente, la figura 3.15 muestra la distribución de los contenedores luego de ejecutar los dos archivos de Docker Compose.

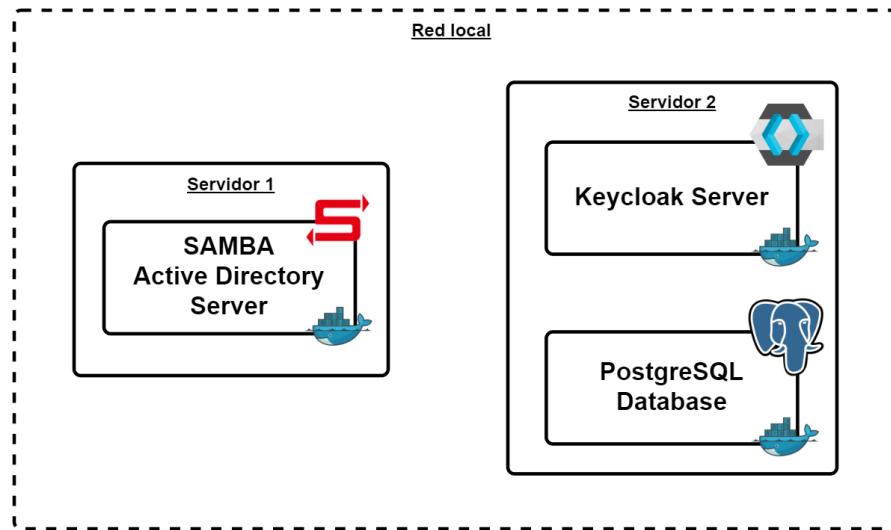


Figura 3.15: Diagrama de contenedores desplegados del sistema centralizado.  
Fuente: Elaboración propia (2021).

### 3.7 AUTOMATIZACIÓN DE INSTALACIÓN Y CONFIGURACIÓN MEDIANTE ANSIBLE

El prototipo obtenido al finalizar los objetivos anteriores requiere una gran cantidad de configuraciones, tanto en el momento de realizar el levantamiento como posteriormente para implementar las integraciones con Samba y Google. Este objetivo contempló la automatización de estos procesos utilizando Ansible, con el fin de levantar el prototipo completo con la menor cantidad de pasos posibles.

El *playbook* implementado consta de cuatro partes: la primera corresponde a tareas comunes para instalar Docker y Docker Compose en todas las máquinas, la segunda realiza el *deploy* del servicio de Samba mediante Docker Compose, la tercera se encarga del *deploy* del servicio de Keycloak con algunas pequeñas configuraciones también mediante Docker Compose y la cuarta implementa las configuraciones necesarias en Keycloak para conectarse con Samba e integrar a Google como proveedor de identidad.

### 3.7.1 Requisitos previos establecidos

Este *playbook* fue diseñado para realizar sólo el *deploy* de los servicios pertenecientes al prototipo del sistema centralizado, por lo tanto, las máquinas en las que se hace el despliegue ya sean maquinas virtuales locales, en la nube o servidores físicos necesitan estar disponibles previamente con el sistema operativo Ubuntu en la versión 20.04 LTS. Además, se establecen dos requerimientos fundamentales para la ejecución del *playbook*:

1. El despliegue del sistema está pensado para ser realizado en dos máquinas *host*, separando el servidor de Samba *Active Directory* del servidor de Keycloak. La IP o nombre de dominio asignado a estas máquinas necesita ser ingresados en el archivo "hosts" que se encuentra en la carpeta "*inventory*" dependiendo de la categoría a la que se quiera asignar, ya sea el servidor de directorio o el servidor de autenticación.
2. Para poder configurar las integraciones con los servicios de Samba y Google es necesario proveer a Keycloak con certificados TLS que no sean auto-firmados, el sistema necesita conocer el emisor de los certificados que posee el servidor, por lo tanto si Keycloak se despliega con certificados auto-firmados todo el rol de configuración del servidor no puede ser ejecutado.

### 3.7.2 Instalación de herramientas

Como todo el *deploy* es realizado mediante Docker y Docker Compose, es necesario instalar como mínimo estas dos herramientas. El rol dedicado a esta tarea se encarga de actualizar el repositorio de paquetes del *host* y verificar si es que Docker está instalado, en el caso que no lo esté se descargan las dependencias necesarias, se agrega el repositorio oficial de Docker y finalmente se instala, además, posterior a terminar la instalación de Docker se descarga Docker Compose desde el repositorio oficial igualmente ofrecido por la documentación.

Para todos los paquetes y dependencias a instalar se establecieron versiones específicas, de esta forma cada vez que se levante el prototipo es posible encontrar el mismo ambiente sin problemas de compatibilidad.

Además de los paquetes de Docker, se realiza la descarga de PIP<sup>16</sup>, que corresponde a un instalador de paquetes especializado para el lenguaje Python, esto se debe a que muchos de los módulos de Ansible requieren dependencias de Python que se pueden obtener desde este administrador de paquetes.

---

<sup>16</sup><https://pypi.org/project/pip>

### 3.7.3 Despliegue de contenedores

Para el despliegue de contenedores se implementaron dos roles, uno para el servidor de Samba AD y otro para el servidor de Keycloak. Ambos roles siguen la misma estrategia de copiar el archivo de Docker Compose al *host* y luego ejecutarlo mediante un comando, pero el rol de Keycloak requiere de una serie de pasos extra.

Tal como se mencionó en la sección 3.7.1, Keycloak requiere de sus propios certificados TLS para su correcto despliegue y configuración, por lo que se integró una tarea especial para copiar los certificados al *host*, además otra para copiar el tema a aplicar en la interfaz de cuentas de usuario descrito en la sección 3.3.4. Existe otra tarea extra destinada a la obtención del certificado público de Samba, el cual requiere ser ingresado al contenedor de Keycloak posterior a la ejecución del archivo de Docker Compose, este archivo es fundamental para realizar la configuración del *User Storage Federation*.

Finalmente, posterior a la ejecución de estos dos roles se puede encontrar ambos servidores aprovisionados con los servicios básicos necesarios para su funcionamiento, listos para ser configurados manualmente o mediante el siguiente rol.

### 3.7.4 Configuración de Keycloak

El último rol se encarga de la configuración con la que se integra Samba para la federación de usuarios y Google como proveedor de identidad a Keycloak. La estrategia que se utilizó para realizar dicha tarea es aprovechar la bien documentada API<sup>17</sup> de Keycloak, con la cual se pueden automatizar los procesos de configuración manual que se requieren hacer mediante la interfaz Web.

Para interactuar con esta API REST se decidió utilizar el módulo URI<sup>18</sup> que viene instalado por defecto con Ansible, este permite especificar distintos atributos como URL, documento a enviar, formato del documento, método, modo, etc. Si bien Ansible ofrece tres módulos específicos para interactuar con Keycloak, estos no cumplen con lo esperado y se decidió no utilizarlos, primero porque la documentación sobre su funcionamiento y utilización está incompleta y segundo por ser enfocados en la creación y manejo de usuarios o grupos, lo cual no cumple con las necesidades de configuración de este *playbook*.

Un aspecto importante de la API de Keycloak es que requiere de la autenticación de un usuario para ser utilizada, la manera de hacerlo es mediante *OpenID Connect* por lo que

---

<sup>17</sup><https://www.keycloak.org/docs-api/14.0/rest-api>

<sup>18</sup>[https://docs.ansible.com/ansible/latest/collections/ansible/builtin/uri\\_module.html](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/uri_module.html)

es necesario primero obtener un *token* de acceso mediante una cuenta que tenga permisos de administración, para efectos de este *playbook* se utilizó la cuenta de administrador que se crea al desplegar el servicio, solicitando el *token* a través de la URL correspondiente mediante el mismo módulo URI. Posterior a la autenticación es posible enviar archivos JSON a través de los *endpoints* correspondientes de la API, haciendo posible automatizar la configuración de la federación de usuarios y también los proveedores de identidad.

Al finalizar la ejecución del *playbook* se puede encontrar el prototipo del sistema centralizado desplegado como se aprecia en la figura 3.16, con los servicios de Samba Active Directory, Keycloak y PostgreSQL funcionando, además, ya configurado para comunicarse mediante *User Storage Federation* con Samba y también comunicándose con Google para delegar autenticación mediante el sistema de *Identity Providers*.

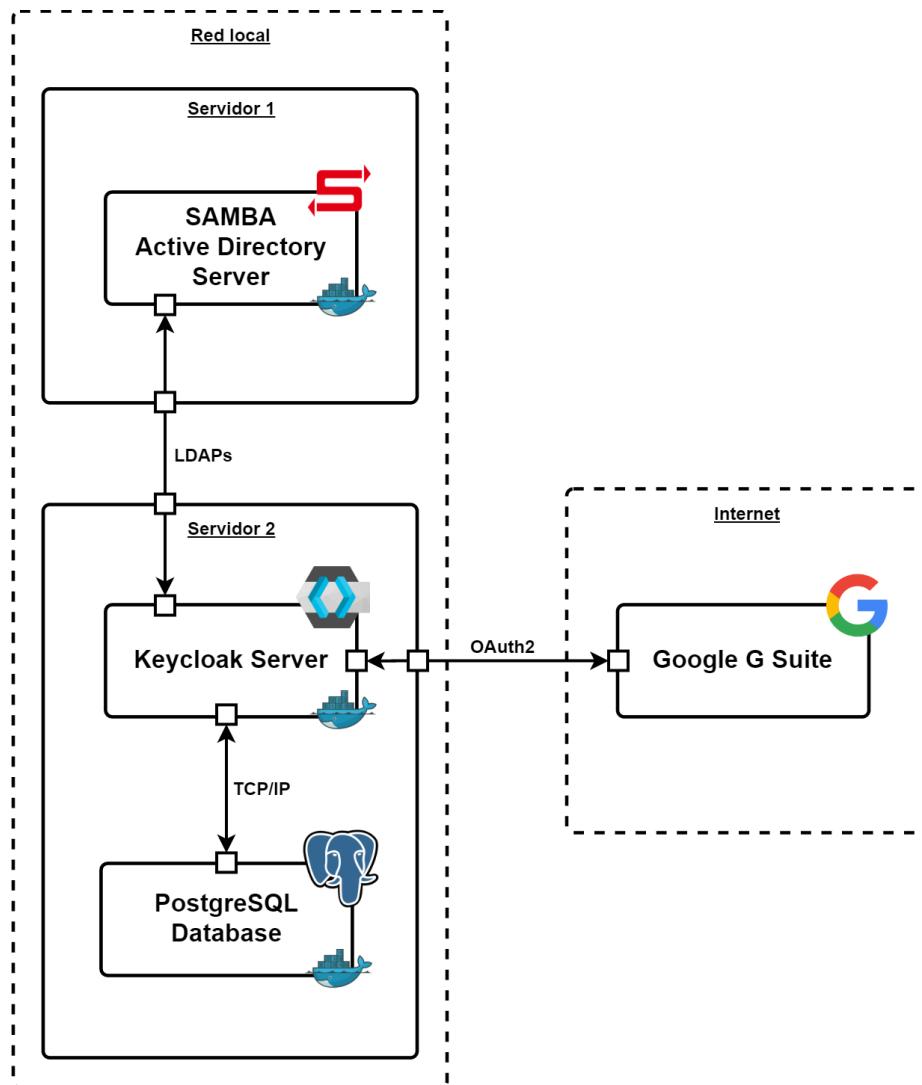


Figura 3.16: Despliegue completo al finalizar el *playbook* de Ansible.

Fuente: Elaboración propia (2021).

## CAPÍTULO 4. CONCLUSIONES

### 4.1 RESULTADOS

Gracias al trabajo realizado se cumplió el objetivo de construir el prototipo del sistema centralizado que se buscaba en la situación deseada, capaz de comunicar un servicio de directorio y un proveedor de identidad para ofrecer en un solo lugar autenticación y autorización en distintos protocolos para variados clientes. Si este prototipo es llevado a producción, permitiría al Departamento de Ingeniería Informática ofrecer un único lugar para autenticar clientes en las distintas aplicaciones que existen actualmente y que pueden ser desarrolladas a futuro.

Ofrecer este punto centralizado puede prevenir muchas situaciones problemáticas al DIINF. El tener la información de los usuarios centralizada implica evitar que cualquier otra aplicación deba tener su propia base de datos de usuarios, de esta forma se elimina la replicación innecesaria de información y se rescatan los recursos asignados a dicho sistema. Por otro lado, implementar un protocolo conocido como *OpenID Connect* permite establecer un estándar para el departamento que puede ser publicado y conocido por los desarrolladores, gracias a esto se puede liberar al área de TI de realizar modificaciones o implementaciones específicas para cada aplicación nueva que desee utilizar el directorio. Finalmente, tanto el encargado de TI como el área de operaciones se ven altamente beneficiados al obtener un sistema capaz de ofrecer el registro automático de usuarios que se sincronizan con el correo USACH, simplificando el proceso de creación de las cuentas de usuario de los estudiantes nuevos cada año, además, la integración de la interfaz Web facilita cualquier tarea de administración de usuario realizada previamente por línea de comandos, como la edición de atributos, reactivación de cuentas en cada semestre o incluso el cambio de contraseñas.

Un último aspecto importante a discutir es la mejora en la situación de las credenciales de la cuenta de usuario del DIINF. El hecho de establecer la combinación de nombre y apellido que utiliza el correo USACH como nombre de usuario para todas las cuentas del departamento simplifica a los integrantes la tarea de recordar sus credenciales, eliminando la situación de tener dos nombres de usuario dentro de la misma institución. Si bien la situación ideal es realmente tener también una sola contraseña y simular la existencia de una sola cuenta institucional, esto no pudo lograrse debido a que el estándar utilizado para sincronizar los usuarios del correo USACH con el DIINF no permite integrar la contraseña del correo al sistema centralizado. En la práctica, esto significa que los usuarios deben utilizar una contraseña específica para iniciar sesión en Windows, esta contraseña puede ser distinta o no del correo USACH dependiendo de los tiempos de caducidad de cada una.

## 4.2 OBJETIVOS

En una mirada general, los objetivos específicos definidos al inicio del proyecto fueron cumplidos y en dos casos incluso se superó la idea inicial esperada. Si bien no es algo reportado por cada objetivo, para cada uno de estos hubo un proceso de investigación asociado a comprender la utilización de los *software* escogidos y su funcionamiento.

1. Comparar nuevas tecnologías de sistemas de directorio y distinguir qué funciones únicas ofrecen a diferencia de lo que puede hacer Samba.

Se realizó un mes de investigación de tecnologías, que incluyó la lectura de distintos indicadores de popularidad de productos de *software* en el área de la administración de identidad, gracias a esto se establecieron 5 productos los cuales fueron instalados para realizar pruebas, además, de cada uno de ellos se examinó la documentación para analizar diversos puntos los cuales fueron utilizados para comparación y determinar cuál era el más adecuado para el sistema centralizado. Finalmente, se optó por tomar los dos mejores productos debido a la similitud en el puntaje final obtenido y determinar el indicado basado su capacidad de cumplir con los siguientes objetivos específicos.

2. Rediseño del árbol de directorio e implementación de nuevo formato para credenciales.

Se diseñaron dos versiones del árbol del directorio, el primero logró definir la estructura general capaz de categorizar muy al detalle los distintos integrantes del departamento basándose en el área a la que pertenecen. Más adelante fue necesario hacer una revisión debido a problemas que fueron encontrados en el primer diseño como no poder permitir que un usuario pertenezca a más de un área, el diseño final fue capaz de solucionar esto al reubicar el lugar específico donde se guardan los usuarios, permitiendo así que un usuario pueda ser incluido en múltiples áreas.

3. Integrar una interfaz Web para reemplazar la administración por línea de comandos.

De los dos *software* posibles, Keycloak fue escogido como la mejor opción para integrar una interfaz Web al directorio. Su amplia documentación de administración mediante la muy acabada interfaz le dio una ventaja por encima de Gluu, la que presenta muchos problemas de diseño y que en general se ve incompleta. Este fue un objetivo en el que se superó lo esperado inicialmente gracias a que Keycloak además de ofrecer la interfaz Web de administración incluye una interfaz para cuentas de usuario, en esta es posible realizar autoservicio para editar información o actualizar la contraseña. Finalmente, también se realizó la modificación de dicha interfaz para incluir colores institucionales y el idioma español.

#### 4. Integrar el protocolo OAuth al sistema, para centralizar la autenticación de usuarios.

Este fue el segundo objetivo donde se superó la idea inicial, tanto Gluu como Keycloak realmente ofrecen *OpenID Connect* como el protocolo para habilitar un punto centralizado de autenticación. Esto fue una muy buena noticia ya que de esta forma es posible ofrecer un sistema centralizado tanto de autenticación mediante OIDC como de autorización mediante OAuth. Con respecto al proceso de integración con Samba, Keycloak fue escogido al permitir modificar información dentro del directorio y haciendo posible llevar toda la administración a este *software*, tarea que no es posible mediante Gluu al permitir sólo lectura.

#### 5. Implementar un sistema para delegar la autenticación de los usuarios a Google mediante el correo USACH, dejando el lado de autorización al servicio de directorio del DIINF.

A esta altura se tomó la decisión de simplemente a escoger a Keycloak como el producto indicado para el sistema centralizado y descartar a Gluu. Para delegar la autenticación de usuarios a Google fue necesario investigar que opciones ofrece, finalmente se encontró que mediante *Google Cloud Platform* es posible generar credenciales de OAuth que pueden ser utilizadas por Keycloak para autenticar usuarios. La ventaja de utilizar este servicio es que permite a cualquier cuenta de Gmail crear estas credenciales, además, en el caso de que se utilice un correo USACH es posible limitar el alcance a solo usuarios de la institución.

#### 6. Empaquetar los servicios en contenedores utilizando Docker.

Esto fue realizado a lo largo de los tres meses de implementación y no presentó mayores complicaciones. Gluu y Keycloak ofrecen formas de ser desplegados mediante contenedores, gracias a esto durante todas las pruebas e implementaciones se trabajó con contenedores. Si bien Samba no ofrece una imagen oficial para Docker, se tomó la decisión de utilizar una bastante completa capaz de ser utilizada para pruebas. En ambos casos, se implementaron archivos de Docker Compose para facilitar el despliegue del sistema.

#### 7. Automatizar la instalación de los servicios mediante Ansible.

Este fue el último objetivo realizado durante el tiempo de implementación, no obstante, fue posible implementar un *playbook* capaz de aprovisionar dos máquinas con Samba y Keycloak. Además del *deploy* es capaz de configurar los servicios de Keycloak para establecer la federación de usuarios con Samba y también la delegación de autenticación a Google mediante el sistema de proveedores de identidad. Si se cumplen los requisitos previamente establecidos, sólo con ejecutar este *playbook* es posible encontrar el prototipo del sistema centralizado listo para ser utilizado, con las configuraciones de lenguaje y temas incluidas.

## 4.3 TRABAJOS FUTUROS

A pesar de haber cumplido todos los objetivos, este como cualquier otro producto de *software* tiene espacio para mejora. Dicho esto, se presentan posibles mejorar a añadir al sistema centralizado:

- Integrar el Proveedor de credenciales de Google para Windows: este *software*<sup>1</sup> permite iniciar sesión en equipos con sistema operativo Windows mediante un correo de Gmail, de esta forma sería posible solucionar el problema de las contraseñas descrito previamente en la sección 4.1 utilizando el correo USACH como cuenta para iniciar sesión en los equipos del departamento. Esto no fue utilizado en la implementación del prototipo ya que su configuración requiere tener acceso de administrador al dominio "usach.cl" en G Suite, por lo que es necesario comunicarse con SEGIC para ver si esto es posible y verificar su factibilidad.
- Modificar la interfaz de la consola de administración de Keycloak: si bien en este proyecto se decidió no hacerlo, es posible modificar la interfaz de la consola de administración de Keycloak y adecuarla a los colores institucionales o integrar el logo del departamento. Un punto extra es también mejorar la traducción al español que esta trae por defecto, la cual contiene algunos mensajes sin traducir.
- Integrar el estándar SAML al sistema centralizado: a pesar de escoger a *OpenID Connect* como el protocolo a ofrecer por el prototipo, Keycloak tiene la capacidad de proveer autenticación y autorización también por SAML. Configurar esta opción para que el sistema centralizado también la ofrezca, entregaría más opciones a los clientes quienes podrían escoger la que más se acomode a su aplicación.
- Implementar un Dockerfile para crear una imagen de Keycloak: si bien el archivo de Docker Compose permite desplegar el contenedor de Keycloak sin problemas, se aplican muchas modificaciones a la imagen que se repiten cada vez que se ejecuta el archivo. Un *Dockerfile* permitiría crear una imagen con estas modificaciones ya integradas.
- Parametrizar el playbook de Ansible: gran parte del *script* implementado está escrito de forma en que los valores de los atributos están indicados de forma directa. Ansible permite el uso de variables las cuales pueden ser definidas en otros archivos, permitiendo así facilidad en su modificación cuando se necesitan cambiar valores específicos o versiones de herramientas a instalar.

---

<sup>1</sup><https://tools.google.com/dlpage/gcpw>

## GLOSARIO

- **Access Control List (ACL)**: es una lista de permisos asociados con un recurso del sistema. Una ACL especifica qué usuarios o procesos del sistema tienen acceso a los objetos, así como qué operaciones están permitidas en determinados objetos.
- **Access Control Models**: se refiere al control de acceso a los recursos del sistema después de que las credenciales y la identidad del usuario han sido autenticadas.
- **Application Programming Interface (API)**: conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro *software* como una capa de abstracción.
- **Command Line Interface (CLI)**: tipo de interfaz de usuario de computadora que permite a los usuarios dar instrucciones a algún programa informático o al sistema operativo por medio de una línea de texto simple.
- **Directory Information Tree (DIT)**: es la información de un directorio representada de una forma jerárquica en una estructura de árbol.
- **Distinguished Name (DN)**: cadena que identifica de forma única una entrada dentro de un DIT.
- **Graphic User Interface (GUI)**: programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.
- **Hardware**: representa los componentes físicos y tangibles de una computadora es decir, los componentes que pueden ser vistos y tocados.
- **Identity Provider (IdP)**: entidad con la autoridad de información relacionada con la identidad de un sistema.
- **Lightweight Directory Access Protocol (LDAP)**: protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.
- **Multi-Factor Authentication (MFA)**: método de control de acceso informático en el que a un usuario se le concede acceso al sistema solo después de que presente dos o más pruebas diferentes de que es quien dice ser.
- **One-Time Password (OTP)**: contraseña válida solo para una autenticación.
- **Secure Socket Layer (SSL)**: protocolo criptográfico que proporciona comunicación segura por una red, comúnmente Internet.
- **Service Provider (SP)**: entidad que presta servicios a otras entidades.
- **Single Sing-On (SSO)**: procedimiento de autenticación que habilita a un usuario determinado para acceder a varios sistemas con una sola instancia de identificación.
- **Software**: sistema formal de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos.
- **Software as a Service (SaaS)**: modelo de distribución de *software* donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación, a los que se accede vía Internet desde un cliente.
- **User Agent**: aplicación informática que funciona como cliente en un protocolo de red; el nombre se aplica generalmente para referirse a aquellas aplicaciones que acceden al Internet.

## REFERENCIAS BIBLIOGRÁFICAS

- Alpern, N. J., & Shimonski, R. J. (2010). *Eleventh Hour Network+*. Syngress.
- Anderson, D. J. (2010). *Kanban - Successful Evolutionary Change for your Technology Business*. Blue Hole Press.
- Anderson, D. J. (2020). *The Principles and General Practices of the Kanban Method*. Recuperado el Lunes 30 de Agosto de 2021, de <https://djaac.com/principles-and-general-practices-of-the-kanban-method/>.
- Auth0 (2021). *Auth0 Documentation: Configure Single Sign-On*. Recuperado el Sábado 4 de Septiembre de 2021, de <https://auth0.com/docs/configure/sso>.
- Casey, K. (2020). *What Is Attribute-Based Access Control (ABAC)?* Recuperado el Jueves 2 de Septiembre de 2021, de <https://www.okta.com/blog/2020/09/attribute-based-access-control-abac/>.
- Chong, C. (2019). *Gluu AD/LDAP Synchronization*. Recuperado el Lunes 30 de Agosto de 2021, de <https://azlabs.blogspot.com/2019/06/gluu-adldap-synchronization.html>.
- Chu, H., Furuseth, H., Gibson-Mount, Q., & Zeilenga, K. (2014). *The OpenLDAP Project Overview*. Recuperado el Jueves 2 de Septiembre de 2021, de <https://www.openldap.org/project/>.
- David J. Anderson School of Management (2021). *4 Common Mistakes Made While Beginning with Kanban*. Recuperado el Viernes 3 de Septiembre de 2021, de <https://djaac.com/4-mistakes-while-beginning-with-kanban/>.
- Docker Inc. (2021). *Use containers to Build, Share and Run your applications*. Recuperado el Domingo 5 de Septiembre de 2021, de <https://www.docker.com/resources/what-container>.
- Ellingwood, J. (2015). *Understanding the LDAP Protocol, Data Hierarchy, and Entry Components*. Recuperado el Jueves 2 de Septiembre de 2021, de <https://www.digitalocean.com/community/tutorials/understanding-the-ldap-protocol-data-hierarchy-and-entry-components>.
- EralInnovator (2021). *Network Infrastructure*. Recuperado el Jueves 2 de Septiembre de 2021, de <https://erainnovator.com/network-infrastructure/>.
- Espinosa, O. (2021). *Okta Documentation: Understanding SAML, SAML Overview*. Recuperado el Sábado 4 de Septiembre de 2021, de <https://www.redeszone.net/tutoriales/seuridad/que-es-oauth/>.
- Gillis, A. S. (2021). *Cloud Computing Definitions: Docker Image*. Recuperado el Domingo 5 de Septiembre de 2021, de <https://searchitoperations.techtarget.com/definition/Docker-image>.
- Ionos (2020). *Role based access control (RBAC): ¿cómo funciona el control de acceso basado en roles?* Recuperado el Jueves 2 de Septiembre de 2021, de <https://www.ionos.es/digitalguide/servidores/seuridad/que-es-el-role-based-access-control-rbac/>.
- Kawasaki, T. (2017). *Diagrams of All The OpenID Connect Flows*. Recuperado el Sábado 4 de Septiembre de 2021, de <https://darutk.medium.com/diagrams-of-all-the-openid-connect-flows-6968e3990660>.
- LDAP (2018a). *The LDAP Bind Operation*. Recuperado el Viernes 3 de Septiembre de 2021, de <https://ldap.com/the-ldap-bind-operation/>.

- LDAP (2018b). *The LDAP Unbind Operation*. Recuperado el Viernes 3 de Septiembre de 2021, de <https://ldap.com/the-ldap-unbind-operation/>.
- LDAP (2018c). *Understanding LDAP Schema*. Recuperado el Viernes 3 de Septiembre de 2021, de <https://ldap.com/understanding-ldap-schema/>.
- López, A. (2021). *Todo sobre criptografía: Algoritmos de clave simétrica y asimétrica*. Recuperado el Jueves 2 de Septiembre de 2021, de <https://www.redeszone.net/tutoriales/seuridad/criptografia-algoritmos-clave-simetrica-asimetrica/>.
- Microsoft (2017). *Active Directory Domain Services Overview*. Recuperado el Jueves 2 de Septiembre de 2021, de <https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview>.
- Microsoft (2018). *What is a Directory Service?* Recuperado el Jueves 2 de Septiembre de 2021, de <https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ldap/what-is-a-directory-service>.
- Okta (2020). *Okta Documentation: Understanding SAML, SAML Overview*. Recuperado el Sábado 4 de Septiembre de 2021, de <https://developer.okta.com/docs/concepts/saml>.
- OneLogin (2021). *How does Single Sign-On work?* Recuperado el Sábado 4 de Septiembre de 2021, de <https://www.onelogin.com/learn/how-single-sign-on-works>.
- Oracle (s.f.). *Object Class Definitions*. Recuperado el Viernes 3 de Septiembre de 2021, de <https://docs.oracle.com/javase/jndi/tutorial/ldap/schema/object.html>.
- Redalia (s.f.). *What is the SSL / TLS Protocol?* Recuperado el Jueves 2 de Septiembre de 2021, de <https://www.redalia.com/ssl/ssl-protocol/>.
- RedHat (2020). *System-Level Authentication Guide*. Recuperado el Jueves 2 de Septiembre de 2021, de [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/system\\_level\\_authentication\\_guide/](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_level_authentication_guide/).
- RedHat (2021a). *Conceptos básicos de Ansible*. Recuperado el Domingo 5 de Septiembre de 2021, de <https://www.redhat.com/es/topics/automation/learning-ansible-tutorial>.
- RedHat (2021b). *¿Qué es Docker?* Recuperado el Domingo 5 de Septiembre de 2021, de <https://www.oracle.com/cl/cloud-native/container-registry/what-is-docker/>.
- Rivas, G. (2020). *Encriptación Simétrica y Asimétrica: Conoce sus diferencias*. Recuperado el Jueves 2 de Septiembre de 2021, de <https://www.gb-advisors.com/es/encriptacion-simetrica-y-asimetrica-conoce-sus-diferencias/>.
- Schwartz, M., & Machulak, M. (2018). *Securing the Perimeter*. Apress.
- Seeger, K. (2006). *SAMBA Overview*. Recuperado el Jueves 2 de Septiembre de 2021, de [https://wiki.samba.org/index.php/Main\\_Page](https://wiki.samba.org/index.php/Main_Page).
- Techotopia (2016). *Mandatory, Discretionary, Role and Rule Based Access Control*. Recuperado el Jueves 2 de Septiembre de 2021, de [https://www.techotopia.com/index.php/Mandatory,\\_Discretionary,\\_Role\\_and\\_Rule\\_Based\\_Access\\_Control](https://www.techotopia.com/index.php/Mandatory,_Discretionary,_Role_and_Rule_Based_Access_Control).
- Torres, G. (2019). *Okta Documentation: Understanding SAML, SAML Overview*. Recuperado el Sábado 4 de Septiembre de 2021, de <https://www.returngis.net/2019/04/oauth-2-0-openid-connect-y-json-web-tokens-jwt-que-es-que/>.
- University of Hawaii West Oahu (2018). *Access Control Models*. Recuperado el Jueves 2 de Septiembre de 2021, de <https://westoahu.hawaii.edu/cyber/best-practices/best-practices-weekly-summaries/access-control/>.

Vinsot, S. (2008). *Los 7 métodos de Autentificación más utilizados*. Evidian.

Willeke, J. (2018). *LDAP CommonName Overview*. Recuperado el Viernes 3 de Septiembre de 2021, de <https://ldapwiki.com/wiki/CommonName>.

Zepeda, E. (2020). *¿Qué es Docker y para qué sirve? Explicación*. Recuperado el Domingo 5 de Septiembre de 2021, de <https://dev.to/silicosis/que-es-docker-y-para-que-sirve-explicacion-5h2n>.

## ANEXO A. EVALUACIÓN VIGENCIA Y DOCUMENTACIÓN DE SOFTWARE

### A.1 CRITERIOS PARA DETERMINAR LA VIGENCIA

- Actividad en Git en el último mes: tiene el fin de medir si es que el software sigue en desarrollo de forma constante. Se le asignó una ponderación del 30% del total de la categoría.
- Última actualización lanzada durante el último semestre: a diferencia de la actividad en Git, esta categoría busca verificar que se hayan lanzado versiones/actualizaciones finales durante el último periodo. Se le asignó una ponderación del 50% del total de la categoría.
- Actividad en el foro en el último mes: esto tiene el fin de ver si existe soporte para el software, esto es de vital importancia cuando se trata de productos gratuitos. Se le asignó una ponderación del 20% del total de la categoría.

Vigencia	%	Gluu	FreeIPA	OpenIAM	Syncope	Keycloak
Actividad Git	30,0%	100,0%	100,0%	66,6%	100,0%	100,0%
Actualizaciones	50,0%	80,0%	80,0%	80,0%	40,0%	100,0%
Actividad foro	20,0%	100,0%	0%	100,0%	0%	100,0%
<b>Porcentaje final</b>	—	90,0%	70,0%	80,0%	50,0%	100,0%

Tabla A.1: Evaluación de vigencia de software.

Fuente: Elaboración propia (2021).

Syncope se dedica principalmente a hacer mantenimiento y *bugfixes* a sus versiones lanzadas ya hace unos años. OpenIAM, FreeIPA y Gluu lanzaron sus últimas versiones durante el año 2020, mientras que Keycloak lanzó su última versión en junio del 2021 al momento de realizar esta investigación. FreeIPA y Syncope no poseen un foro oficial de soporte, por lo que reciben cero puntos en la categoría de actividad en el foro.

### A.2 CRITERIOS PARA EVALUAR LA DOCUMENTACIÓN

- Porcentaje de completación: la intención es medir la cantidad de entradas en la documentación y luego verificar cuántas de estas contienen información real y utilizable. Se le asignó una ponderación del 50% del total de la categoría.
- Uso de diagramas: evaluar si es que se hace utilización de diagramas para simplificar el flujo del sistema o los componentes que interactúan, con el fin de simplificar la explicación de lo que realiza el software. Se le asignó una ponderación del 30% del total de la categoría.
- Tutoriales: medir si la documentación además de explicaciones de instalación, contiene tutoriales para ejemplificar el uso o configuración de otros aspectos específicos. Se le asignó una ponderación del 20% del total de la categoría.

<b>Documentación</b>	<b>%</b>	<b>Gluu</b>	<b>FreeIPA</b>	<b>OpenIAM</b>	<b>Syncope</b>	<b>Keycloak</b>
% Completación	50,0%	100,0%	80,0%	80,0%	80,0%	100,0%
Diagramas	30,0%	66,6%	0,0%	33,3%	100,0%	66,6%
Tutoriales	20,0%	100,0%	0,0%	100,0%	100,0%	100,0%
<b>Porcentaje final</b>	—	90,0%	40,0%	70,0%	90,0%	90,0%

Tabla A.2: Evaluación de documentación de software.

Fuente: Elaboración propia (2021).

FreeIPA presenta una documentación muy poco estructurada, contiene una gran cantidad de información pero hay algunas entradas a las que sólo se puede llegar por otros métodos, además no se hace uso de diagramas de ejemplificación. OpenIAM a la fecha actual acaba de lanzar una nueva versión de su documentación, por lo que aún está poco organizada, faltando imágenes y diagramas. Gluu, Syncope y Keycloak presentan una documentación muy completa, correctamente separada por secciones de instalación, uso, integración etc. Además, cuentan con imágenes y diagramas de ejemplo como también tutoriales específicos para la utilización de ciertas funcionalidades.

## ANEXO B. CREACIÓN DEL TEMA DIINF PARA LA INTERFAZ DE CUENTAS DE USUARIO DE KEYCLOAK

Para crear este nuevo tema es necesario comenzar con una nueva carpeta que contenga los archivos necesarios para las interfaces que son modificadas, de esta forma se crea el tema "DIINF" y se copia la vista "login" desde el tema "keycloak", mientras que la vista "account" se copia desde el tema "keycloak.v2", obteniendo la estructura que se aprecia en la figura B.1.

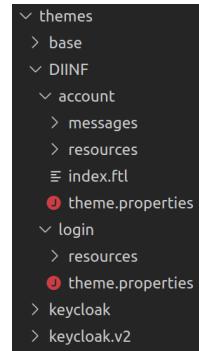


Figura B.1: Estructura de la carpeta del tema DIINF.  
Fuente: Elaboración propia (2021).

Para modificar la vista de *login* se utilizaron dos imágenes nuevas las que deben ser guardadas en la ubicación "*themes/DIINF/login/resources/img*", la primera correspondiente al logo del Departamento de Ingeniería Informática y la segunda es una fotografía del frontis del departamento para utilizar de fondo en el momento que se ingresan las credenciales de usuario. Las configuraciones a realizar para establecer el logo a utilizar y la imagen de fondo necesitan la modificación del archivo "*login.css*" que se encuentra en la ubicación "*themes/DIINF/login/resources/css*", para establecer la imagen de fondo es necesario modificar "*login-pf body*", mientras que para el logo se necesita cambiar "*div.kc-logo-text*". El resultado se puede apreciar en la imagen de la figura B.2.

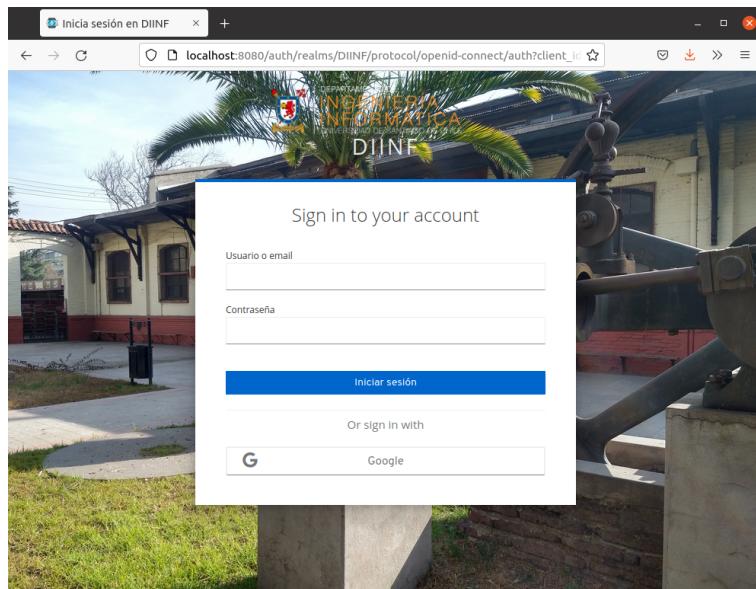
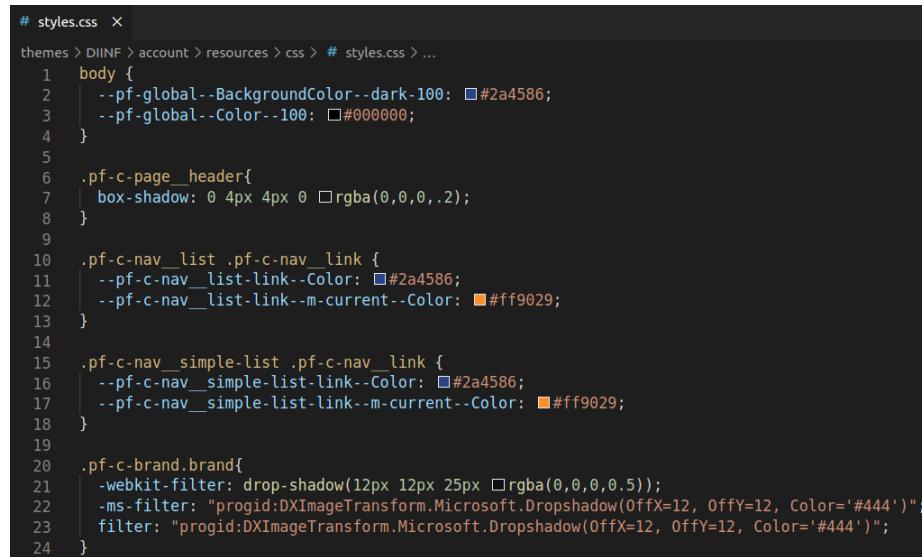


Figura B.2: Vista de inicio de sesión para integrantes del DIINF en Keycloak.  
Fuente: Elaboración propia (2021).

La modificación de la vista de cuentas de usuario requiere de mayor cantidad de trabajo, en especial en el área de la aplicación de colores institucionales y del idioma (descrito posteriormente en la sección de localización). Para la aplicación de los colores es necesario crear un archivo css nuevo en la ubicación "*themes/DIINF/account/resources/css*" y puede tener el nombre que se estime conveniente (en este caso se utilizó "*style.css*").



```

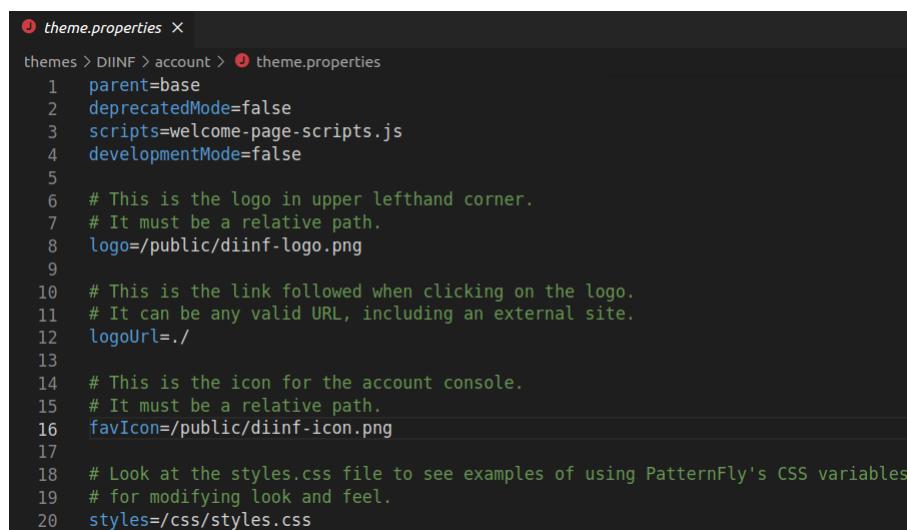
# styles.css ×
themes > DIINF > account > resources > css > # styles.css > ...
1 body {
2   --pf-global--BackgroundColor--dark-100: #2a4586;
3   --pf-global--Color-100: #000000;
4 }
5
6 .pf-c-page_header{
7   box-shadow: 0 4px 4px 0 rgba(0,0,0,.2);
8 }
9
10 .pf-c-nav_list .pf-c-nav_link {
11   --pf-c-nav_list-link--Color: #2a4586;
12   --pf-c-nav_list-link--m-current--Color: #ff9029;
13 }
14
15 .pf-c-nav_simple-list .pf-c-nav_link {
16   --pf-c-nav_simple-list-link--Color: #2a4586;
17   --pf-c-nav_simple-list-link--m-current--Color: #ff9029;
18 }
19
20 .pf-c-brand.brand{
21   -webkit-filter: drop-shadow(12px 12px 25px rgba(0,0,0,0.5));
22   -ms-filter: "progid:DXImageTransform.Microsoft.Dropshadow(OffX=12, OffY=12, Color='#444')";
23   filter: "progid:DXImageTransform.Microsoft.Dropshadow(OffX=12, OffY=12, Color='#444')";
24 }

```

Figura B.3: Contenido del archivo *style.css*, de la interfaz de cuentas de usuario.

Fuente: Elaboración propia (2021).

Este archivo se encarga de integrar los colores institucionales azul y naranjo a la interfaz y también agregar sombras al logo del departamento y al *navbar* para que resalten sobre el fondo. Para aplicar este archivo se necesita configurar mediante el archivo "*theme.properties*" en la ruta "*themes/DIINF/account*" como se ve en la figura B.4, agregando la variable "*styles*" con la dirección del archivo css. En este mismo archivo es posible seleccionar la imagen a utilizar como logo de la institución y el ícono para el navegador Web.



```

❶ theme.properties ×
themes > DIINF > account > ❶ theme.properties
1 parent=base
2 deprecatedMode=false
3 scripts=welcome-page-scripts.js
4 developmentMode=false
5
6 # This is the logo in upper lefthand corner.
7 # It must be a relative path.
8 logo=/public/diinf-logo.png
9
10 # This is the link followed when clicking on the logo.
11 # It can be any valid URL, including an external site.
12 logoUrl=..
13
14 # This is the icon for the account console.
15 # It must be a relative path.
16 favIcon=/public/diinf-icon.png
17
18 # Look at the styles.css file to see examples of using PatternFly's CSS variables
19 # for modifying look and feel.
20 styles=/css/styles.css

```

Figura B.4: Contenido del archivo *theme.properties*, de la interfaz de cuentas de usuario.

Fuente: Elaboración propia (2021).

El resultado final ya con los colores institucionales y el logo del departamento se puede apreciar en la figura B.5.

The screenshot shows a web browser window titled "Gestión de Cuenta Keycloak". The address bar displays "localhost:8080/auth/realms/DIINF/account/#/personal-info". The header includes the logo of the "DEPARTAMENTO DE INGENIERIA INFORMATICA UNIVERSIDAD DE SANTANDER DE CHILE" and a "Cerrar Sesión" button. The main content area is titled "Información Personal" and contains the following fields:

Usuario *	roberto.lillo
Email *	roberto.lillo@usach.cl
Nombre *	Roberto
Apellido *	Lillo
Selecciona un idioma *	Español

At the bottom are "Guardar" and "Cancelar" buttons.

Figura B.5: Vista de cuenta de usuario para integrantes del DIINF en Keycloak.  
Fuente: Elaboración propia (2021).

## ANEXO C. LOCALIZACIÓN AL ESPAÑOL DE LA INTERFAZ DE CUENTAS DE USUARIO DE KEYCLOAK

Los archivos de idioma por defecto se encuentran en el tema base de Keycloak, en cada una de las carpetas para las distintas vistas se puede encontrar la carpeta "messages" que contiene archivos de tipo *properties* tal como se puede apreciar en la figura C.1, cada uno de estos archivos contiene una serie de variables con su correspondiente traducción al idioma que corresponda. De esta forma posteriormente el *software* aplica el mensaje que sea necesario dependiendo de la variable y el idioma que haya sido escogido previamente en la configuración de internacionalización.

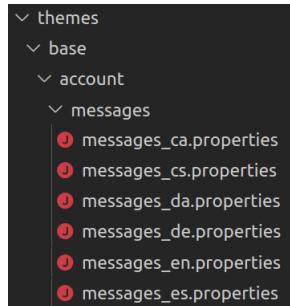


Figura C.1: Estructura de la carpeta "messages" en el tema base.  
Fuente: Elaboración propia (2021).

Para facilitar la personalización, es posible integrar una carpeta "messages" con sus propios archivos de idioma en todo nuevo tema que sea creado. Estos archivos no necesitan ser una copia completa del archivo por defecto que se encuentra en el tema base, ya que es posible personalizar solo las variables que sea necesario y luego Keycloak en caso de no encontrar una variable específica en el tema aplicado, irá a buscar su valor dentro de los archivos del tema base.

Finalmente, como la nueva interfaz de cuentas de usuario utiliza nuevas variables para muchos de sus mensajes, si se aplica la traducción al español por defecto esta no se aplica correctamente y muchas de las líneas se mantienen en inglés. En el nuevo archivo "messages\_es.properties" que se creó para el tema DIINF se realizó la traducción de 88 mensajes, un pequeño extracto se puede apreciar en la figura C.2.

```
① messages_es.properties ×
themes > DIINF > account > messages > ① messages_es.properties
1 # General
2 doSave=Guardar
3 doCancel=Cancelar
4 continue=Continuar
5 update=Actualizar
6 unknown=Desconocido
7 -----
8
9 # Navbar
10 fullName={0} {1}
11 doSignIn=Iniciar Sesi\u00f3n
12 doSignOut=Cerrar Sesi\u00f3n
13 -----
14
15 # Landing Page
16 accountManagementWelcomeMessage=Bienvenido a la administraci\u00f3n de cuentas DIINF
17 personalInfoHtmlTitle=Informaci\u00f3n Personal
18 personalInfoIntroMessage=Gestiona tu informaci\u00f3n b\u00f3sica
19 accountSecurityTitle=Seguridad de la cuenta
20 accountSecurityIntroMessage=Administra tu contrase\u00f1a y acceso a la cuenta
21 applicationsHtmlTitle=Aplicaciones
22 applicationsIntroMessage=Ve las aplicaciones disponibles en el departamento
23 -----
```

Figura C.2: Extracto del archivo "messages\_es.properties" del tema DIINF.  
Fuente: Elaboración propia (2021).

## ANEXO D. CONFIGURACIÓN DEL *USER STORAGE FEDERATION* EN KEYCLOAK

Keycloak utiliza el protocolo Idaps para conectarse de forma segura a los servidores con los que realizará la federación de usuarios, en este caso para establecer la conexión con el servidor de Samba necesita conocer su certificado TLS público o el CA que emitió dicho certificado. En el caso de que ya se tenga un servidor con Samba AD en producción con sus certificados TLS generados por una entidad conocida, no se necesita mayor configuración. Si se está levantando el sistema en un ambiente de prueba, los certificados auto-generados por Samba serán rechazados por Keycloak a menos que el certificado público sea ingresado al archivo "cacerts" de Java en la máquina que contiene a Keycloak. Particularmente si se está usando Docker para levantar el servicio de Keycloak, el certificado público de Samba debe ser ingresado en el archivo "cacerts" dentro del contenedor.

Una vez realizada la integración del certificado se puede proceder a configurar la sincronización mediante la pestaña *User Federation* dentro de la consola de administración de Keycloak, ingresando la información de una cuenta que tenga permisos de administración en el servidor LDAP, tal como se ve en la figura D.1.

The screenshot shows the Keycloak administration interface with the 'User Federation' tab selected in the left sidebar under the 'DIINF' realm. The main content area displays the configuration for the 'Samba.dlinfo.lan' provider. The 'Ajustes' (Settings) tab is active. The configuration form includes the following fields:

- Required Settings**:
  - Provider ID: Samba
  - Habilitado (Enabled): SI (checked)
  - Console Display Name: samba.dlinfo.lan
  - Priority: 0
  - Import Users: SI (checked)
  - Edit Mode: WRITABLE
  - Sync Registrations: SI (checked)
  - \* Vendor: Active Directory
  - \* Username LDAP attribute: sAMAccountName
  - \* RDN LDAP attribute: cn
  - \* UUID LDAP attribute: objectGUID
  - \* User Object Classes: person, organizationalPerson, user
  - \* Connection URL: ldaps://samba.dlinfo.lan
  - \* Users DN: CN=Users,DC=dlinfo,DC=lan
  - Custom User LDAP Filter: LDAP Filter
  - Search Scope: One Level
  - \* Bind Type: simple
  - \* Bind DN: CN=Administrator,CN=Users,DC=dlinfo,DC=lan
  - \* Bind Credential: (redacted)
- Buttons**:
  - Test connection
  - Test authentication

Figura D.1: Pestaña de configuración de *User Federation* en Keycloak.  
Fuente: Elaboración propia (2021).

En el caso de que la configuración sea con un servidor de Samba AD, es importante cambiar el valor de la variable *Username LDAP Attribute* de "cn" a "sAMAccountName", ya que esta es la variable que Windows utiliza como la credencial de usuario en situaciones como el inicio de sesión en un equipo con Windows 10. Una vez que se hayan ingresado los datos necesarios, es posible utilizar los botones *Test Connection* y *Test authentication* para verificar que la conexión se realiza correctamente con el servidor LDAP, el primero verifica que se encuentre el servidor en la red mientras que el segundo confirma si el usuario ingresado para el *bind* existe y tiene los permisos necesarios. En el caso de que no se haya configurado lo mencionado anteriormente con el certificado TLS, el botón de *Test authentication* lanzará un error debido a que esta prueba se realiza por ldaps.

Posterior a la correcta configuración de la federación, cualquier usuario que sea creado dentro de Keycloak será también creado en el servidor LDAP. En el caso de que se cree un nuevo usuario en el servidor LDAP, la homologación de este en Keycloak depende de la política que se haya establecido para los intervalos en los que Keycloak realiza una sincronización de usuarios, en el caso de ser necesario también se puede forzar una sincronización cuando sea necesario en la misma pestaña de *User Federation*.

The screenshot shows the Keycloak administration interface for a user named 'roberto.lillo@usach.cl'. The left sidebar is titled 'DIINF' and includes sections for 'Configure' (Realm Settings, Clients, Client Scopes, Roles, Providers, User Federation, Authentication) and 'Manage' (Groups, Users, Sessions, Events, Import, Export). The 'Users' section is selected. The main content area shows the user details for 'roberto.lillo@usach.cl'. The 'Details' tab is active, displaying the following information:

ID	8e33fbcc-ff07-441b-a614-86ca38f767df
Created At	8/31/21 12:42:59 AM
Usuario	roberto.lillo@usach.cl
Email	roberto.lillo@usach.cl
First Name	Roberto
Last Name	Lillo
User Enabled	<input checked="" type="checkbox"/> SI
Federation Link	samba.diinf.lan
Email Verified	<input checked="" type="checkbox"/> SI
Required User Actions	Select an action...
Locale	Select one...
Impersonate user	Impersonate

At the bottom right are 'Guardar' (Save) and 'Cancelar' (Cancel) buttons.

Figura D.2: Usuario federado por Keycloak desde un servidor de Samba AD.  
Fuente: Elaboración propia (2021).

## ANEXO E. OBTENCIÓN DE CREDENCIALES OAUTH EN GOOGLE CLOUD PLATFORM

En la consola de *Google Cloud Platform* luego de activar el módulo de API, dentro de la sección "API y Servicios" se encuentra la "Pantalla de consentimiento de OAuth". En esta se necesita establecer el nombre de la aplicación, la URL y los dominios autorizados a utilizar el servicio de OAuth, también es posible definir un logo en el caso que sea deseado.

API APIs & Services	Edit app registration
<ul style="list-style-type: none"><li>❖ Dashboard</li><li>☰ Library</li><li>❖ Credentials</li><li><b>❖ OAuth consent screen</b></li><li>☐ Domain verification</li><li>☒ Page usage agreements</li></ul>	<p><b>① OAuth consent screen</b> — <b>② Scopes</b> — <b>③ Summary</b></p> <p><b>App information</b></p> <p>This shows in the consent screen, and helps end users know who you are and contact you</p> <p><b>App name *</b> Keycloak Server</p> <p>The name of the app asking for consent</p> <p><b>User support email *</b> roberto.lillo@usach.cl</p> <p>For users to contact you with questions about their consent</p> <p><b>App logo</b> <b>BROWSE</b></p> <p>Upload an image, not larger than 1MB on the consent screen that will help users recognize your app. Allowed image formats are JPG, PNG, and BMP. Logos should be square and 120px by 120px for the best results.</p>  <p><b>App domain</b></p> <p>To protect you and your users, Google only allows apps using OAuth to use Authorized Domains. The following information will be shown to your users on the consent screen.</p> <p><b>Application home page</b> <a href="https://keycloak.diinft.tk">https://keycloak.diinft.tk</a></p> <p>Provide users a link to your home page</p> <p><b>Application privacy policy link</b> <a href="https://keycloak.diinft.tk">https://keycloak.diinft.tk</a></p> <p>Provide users a link to your public privacy policy</p> <p><b>Application terms of service link</b> <a href="https://keycloak.diinft.tk">https://keycloak.diinft.tk</a></p> <p>Provide users a link to your public terms of service</p> <p><b>Authorized domains</b> <b>?</b></p> <p>When a domain is used on the consent screen or in an OAuth client's configuration, it must be pre-registered here. If your app needs to go through verification, please go to the <a href="#">Google Search Console</a> to check if your domains are authorized. <a href="#">Learn more</a> about the authorized domain limit.</p> <p>diinft.tk</p>

Figura E.1: Pantalla de consentimiento de OAuth en *Google Cloud Platform*.  
Fuente: Elaboración propia (2021).

La siguiente ventana que se despliega corresponde al alcance (*scopes*) que tiene la aplicación para recuperar la información asociada a la cuenta de Google con la que se inicie sesión. Para efectos de este ejemplo se puede mantener con la configuración por defecto.

The screenshot shows the "Edit app registration" page under the "OAuth consent screen" tab. At the top, there are three tabs: "OAuth consent screen" (selected), "Scopes" (highlighted in blue), and "Summary". Below the tabs, a note explains that scopes express permissions requested from users. A button labeled "ADD OR REMOVE SCOPES" is present. The main area is divided into sections for "Your non-sensitive scopes" and "Your sensitive scopes".

API ↑	Scope	User-facing description
...	./auth/userinfo.email	See your primary Google Account email address
...	./auth/userinfo.profile	See your personal info, including any personal info you've made publicly available

**Your sensitive scopes**

Sensitive scopes are scopes that request access to private user data.

API ↑	Scope	User-facing description
No rows to display		

**Your restricted scopes**

Restricted scopes are scopes that request access to highly sensitive user data.

API ↑	Scope	User-facing description
No rows to display		

At the bottom right are "SAVE AND CONTINUE" and "CANCEL" buttons.

Figura E.2: Configuración de los *scopes* para la aplicación registrada en *Google Cloud Platform*.  
Fuente: Elaboración propia (2021).

Con respecto al rango de usuarios permitidos, si se establece los usuarios como "internos" sólo cuentas del dominio "**@usach.cl**" podrán iniciar sesión en las aplicaciones, si se configura como "externos" cualquier cuenta de Google podrá iniciar sesión.

The screenshot shows the "OAuth consent screen" configuration. Under "User type", it is set to "Internal" (highlighted in blue). A "MAKE EXTERNAL" button is visible below the "User type" section.

Figura E.3: Configuración de rango de usuarios permitidos en la aplicación.  
Fuente: Elaboración propia (2021).

La última configuración se realiza en la pestaña "credenciales", al presionar en el botón para crear credenciales se debe seleccionar la opción para crear un nuevo cliente OAuth, luego en el tipo de aplicación seleccionar "aplicación Web" y asignar un nombre. Finalmente es necesario establecer la URI de redireccionamiento autorizada para utilizar estas credenciales, en el caso de Keycloak esta es entregada al momento de crear el proveedor de identidad.

The screenshot shows the 'Client ID for Web application' configuration screen. On the left sidebar, 'Credentials' is selected under 'API & Services'. The main form has the following fields:

- Name \***: Keycloak Server
- Authorized JavaScript origins**: A note states: "The domains of the URIs you add below will be automatically added to your [OAuth consent screen](#) as [authorized domains](#)." Below this is a button labeled "+ ADD URI".
- Authorized redirect URIs**: A note states: "For use with requests from a web server". Below this is a text input field containing "https://keycloak.diinf.tk/auth/realms/DIINF/broker/google/endpoint" and a button labeled "+ ADD URI".
- SAVE** and **CANCEL** buttons at the bottom.

Figura E.4: Pantalla de configuración para la creación de credenciales en *Google Cloud Platform*.  
Fuente: Elaboración propia (2021).

Una vez se presione en el botón guardar, se generan las credenciales que posteriormente deben ser ingresadas en Keycloak tal como se ve en la figura E.5.

The screenshot shows the 'Ajustes' tab for the 'Google' identity provider. The configuration includes:

- URI de redirección**: https://keycloak.diinf.tk/auth/realms/DIINF/broker/google/endpoint
- ID Cliente**: 100-100000-00000000000000000000000000000000.apps.googleusercontent.com
- Secreto de Cliente**: (redacted)

Figura E.5: Ubicación de URI y credenciales de OAuth en Keycloak.  
Fuente: Elaboración propia (2021).