

Universidad de La Habana  
Facultad de Matemática y Computación



# Sistema de Autenticación Central

Autor:  
**Nadia González Fernández**

Tutores:  
**Lic. Roberto Martí Cedeño**

Trabajo de Diploma  
presentado en opción al título de  
Licenciado en Ciencia de la Computación

Noviembre de 2022  
[github.com/nala7/central\\_authentication\\_system](https://github.com/nala7/central_authentication_system)

*A mis padres, por su amor incondicional.  
A mi hermana, por ser mi guía y confidente.  
A Luis, por hacerme feliz.*

# Agradecimientos

A mi familia, que a pesar de la distancia siempre está muy cerca de mi. A mis padres, por siempre priorizar mi felicidad y mi bienestar. Gracias por sus sacrificios, ejemplo y amor, a ellos les debo la persona que soy hoy. A mi hermana, Laila, mi guía y mejor compagnía. Mi primera profesora, la que me enseñó a sumar y restar y la que me enamoró de esta carrera. Gracias por tu paciencia y por siempre estar allí. A Mario, por hacerme sentir parte de su familia. A mis abuelos, por sus mimos. A mis tíos, Raisa, Olguita y Víctor por todas las aventuras. A mis primos, Iván, Mayte, Thalía y Vitico, por las horas de juego y alegría.

A Luis, por ser mi constante y mi impulso cuando hacía falta. Por estar en las buenas y en las malas. Por elegir pasar esta experiencia a mi lado.

A mis compañeros de cinco años de carrera. A Judy, Jose, Labourdette, Carlos, Sandra, Amalia, Gaby y Ariel por las largas madrugadas de proyectos y estudio, por las fiestas y las risas. A mis amigos, por hacer esta experiencia inolvidable.

A mis profesores, por su dedicación y enseñanzas. A ellos les debo la profesional que hoy puedo decir que soy. A mi tutor, Roberto, por asumir esta tesis y todo el tiempo dedicado.

# Opinión del tutor

Opiniones de los tutores

# Resumen

Las deficiencias del sistema de autenticación ofrecido por el Nodo Central de La Universidad de La Habana motivan la puesta en práctica de una nueva solución. Debido a las ventajas de Keycloak, se apuesta por esta herramienta como alternativa al sistema vigente. Se realiza un estudio del estado del arte acerca de Keycloak y el conjunto de protocolos y métodos que utiliza. Además, se presenta una solución a cómo configurar y hacer uso de esta herramienta.

Se determinan las principales dificultades de los métodos de autenticación en uso en el Nodo Central y se procede a diseñar un nuevo sistema de autenticación central que permite integrar a todos los usuarios de la universidad en un único sistema. Este proceso culmina con la propuesta de una metodología que permite a los clientes comunicarse con el nuevo sistema de forma sencilla y repetible. Como parte de la solución se utilizan los protocolos OpenID Connect y SAML para garantizar una comunicación rápida y segura. También se hace uso del método Single Sign-On y la autenticación basada en *tokens* para facilitar la experiencia de los usuarios, y LDAP para el almacenamiento de los datos.

Al aplicar esta metodología se obtienen resultados satisfactorios: un sistema capaz de autenticar a todos los usuarios de la Universidad de La Habana a través de diversos clientes. Los resultados obtenidos validan la efectividad de la metodología desarrollada.

# Abstract

The inadequacy of the authentication system supplied by the Main Network Administration Center of the University of Havana motivates the implementation of a new solution. Due to the advantages offered by Keycloak, it is chosen as an alternative to the current system. A study of the state of the art about Keycloak and the set of protocols and methods it uses is carried out. In addition, a solution is presented on how to configure and make use of this tool.

The main difficulties of the authentication mechanisms in use in the Main Network Administration Center are determined and a new central authentication system is designed in order to integrate all university users into a single system. This process ends with the proposal of a solution that allows customers to communicate with the new system in a simple and repeatable way. As part of the solution, OpenID Connect and SAML are used to guarantee fast and secure communications. In addition, Single Sign-On and token-based authentication are also used to facilitate the user experience. Finally, LDAP is utilized for data storage.

Positive results were obtained through this methodology: a system capable of authenticating all users of the University of Havana through various clients. The results obtained validate the effectiveness of the developed solution.

# Índice general

<b>Introducción</b>	<b>1</b>
<b>1. Métodos para la autenticación central</b>	<b>4</b>
1.1. Autenticación y Autorización . . . . .	4
1.1.1. Autenticación . . . . .	4
1.1.2. Autorización . . . . .	5
1.2. Inicio de Sesión Único . . . . .	6
1.3. Protocolos de autenticación . . . . .	7
1.3.1. OpenID Connect . . . . .	7
1.3.2. SAML . . . . .	8
1.3.3. Kerberos . . . . .	8
1.4. Gestores de usuarios . . . . .	9
1.4.1. LDAP . . . . .	9
1.4.2. Active Directory . . . . .	10
1.5. Servicios administradores de identidad . . . . .	11
1.5.1. Okta . . . . .	11
1.5.2. Gluu . . . . .	11
1.5.3. Auth0 . . . . .	11
1.5.4. Keycloak . . . . .	12
1.5.5. JSON Web Token . . . . .	13
1.5.6. REST API . . . . .	15
<b>2. Propuesta de Sistema Central de Autenticación</b>	<b>17</b>
2.1. Hipótesis . . . . .	17
2.1.1. Requisitos del Software: . . . . .	18
2.2. Bases de Datos . . . . .	19
2.3. Ingestión de Datos . . . . .	20
2.4. Capa de Autenticación . . . . .	20
2.5. Clientes . . . . .	21
2.6. Usuarios . . . . .	23

<b>3. Elección de herramientas y detalles de implementación</b>	<b>24</b>
3.1. Elección de herramientas . . . . .	24
3.2. Ingestión de Datos . . . . .	25
3.2.1. Herramientas para el uso de LDAP . . . . .	25
3.2.2. Keycloak como herramienta para la ingestión de datos . . . . .	27
3.3. Capa de autenticación . . . . .	34
<b>4. Experimentación y resultados</b>	<b>37</b>
4.1. Configuración de cliente Keycloak . . . . .	37
4.2. Creación de cliente Keycloak para obtención de <i>tokens</i> . . . . .	41
4.3. Evaluación . . . . .	43
<b>Conclusiones</b>	<b>48</b>
<b>Recomendaciones</b>	<b>49</b>
<b>Bibliografía</b>	<b>50</b>

# Índice de figuras

1.1. Estructura de JWT . . . . .	14
1.2. Flujo de información en Keycloak . . . . .	16
2.1. Etapas de implementación del software . . . . .	19
2.2. Flujo de información desde el cliente hasta la base de datos . . . . .	22
3.1. Usuario administrador . . . . .	27
3.2. Relación de entidades en Keycloak . . . . .	28
3.3. Keycloak: <i>User Federation</i> . . . . .	28
3.4. Configuración de LDAP en Keycloak(1) . . . . .	29
3.5. Configuración de LDAP en Keycloak (2) . . . . .	30
3.6. Usuarios en LDAP vistos desde Apache Directory Studio . . . . .	31
3.7. Lista de usuarios . . . . .	31
3.8. Configuración para mapear . . . . .	32
3.9. Configuración para mapear email . . . . .	33
3.10. Usuarios con email mapeado . . . . .	33
3.11. Diagrama de flujo de información . . . . .	34
4.1. Nuevo cliente en Keycloak(1) . . . . .	38
4.2. Nuevo cliente en Keycloak (2) . . . . .	38
4.3. Lista de clientes . . . . .	39
4.4. Cliente nodo . . . . .	40
4.5. Credenciales del cliente nodo . . . . .	40
4.6. Interfaz visual . . . . .	42
4.7. Resultado de autenticación exitosa . . . . .	43
4.8. Interfaz visual de Locust . . . . .	45
4.9. Configuración de Locust . . . . .	45
4.10. Estadísticas obtenidas de prueba de Locust . . . . .	46
4.11. Total de pedidos por segundo . . . . .	46
4.12. Tiempo de respuesta . . . . .	47
4.13. Número de usuarios . . . . .	47

# Ejemplos de código

4.1. Conexión de cliente a Keycloak . . . . .	41
4.2. Conexión de Locust a Keycloak . . . . .	44

# Introducción

Hoy en día las personas utilizan muchas aplicaciones con distintas identidades. La identidad del usuario es considerada un conjunto de atributos, permanente o de larga vida, asociados a un usuario. Para obtener acceso a servicios muchas aplicaciones requieren que el usuario esté registrado y haya iniciado sesión. La mayoría de las veces los usuarios están registrados en distintos sitios o servicios con el mismo nombre de usuario y con la misma o similar contraseña, lo cual no es la mejor práctica para preservar la seguridad de las cuentas. Por todo ello el manejo de múltiples nombres de usuarios y contraseñas es una tarea molesta del Internet actual.

Teniendo en cuenta todos esos problemas de seguridad, cada vez más los administradores de sitios web deciden delegar esos servicios de administración y autenticación a terceros, entidades externas especializadas en ese tipo de actividad, que pueden alojar, almacenar, administrar y proteger los datos de los usuarios de un proveedor de servicios en particular. Además, ofrecen una interfaz de programación de aplicaciones (conocida también por la sigla API, en inglés, *Application Programming Interface*), para proporcionar acceso a los datos del usuario a las aplicaciones externas. Estas soluciones permiten compartir los datos con más de un sistema web, permitiendo así el Inicio de Sesión Único (SSO) para aplicaciones de terceros. [1]

El Nodo Central de la Universidad de La Habana tiene entre sus responsabilidades dar las credenciales digitales a todos los usuarios de la Universidad. Los trabajadores y estudiantes de la institución registran sus datos personales en las bases de datos de recursos humanos y secretaría docente, respectivamente. Esta información es utilizada más adelante para que los usuarios puedan autenticarse en los distintos servicios de la institución, respaldado por sus sistemas de origen.

En este trabajo se presenta una solución a los problemas de autenticación, que se ajusta a las necesidades y posibilidades de la Universidad de La Habana.

## Motivación

Actualmente vivimos en un mundo donde la tecnología tiene un papel protagónico. La Universidad de La Habana en los últimos años ha estado inmersa en el proceso

de transformación digital que lleva a cabo nuestro país, como continuidad de la estrategia de informatización de la sociedad cubana, que pretende integrar las tecnologías digitales a todos los ámbitos de la sociedad.

Garantizar un acceso seguro y sencillo a los recursos de la red es esencial para alcanzar este objetivo. Por consiguiente, se deben establecer mecanismos que cuenten con un alto nivel de seguridad y permitan identificar quién realmente está autorizado para acceder a los recursos del sistema.

## **Antecedentes**

Actualmente todos los usuarios de la Universidad de La Habana se almacenan en dos Protocolos Ligeros de Acceso a Directorios (en inglés: *Lightweight Directory Access Protocol*, también conocido por sus siglas como LDAP). En los dos directorios se guardan las cuentas de correo y los datos de sus usuarios. En uno ellos se registra la información de los estudiantes y en el otro la de los trabajadores. A partir de estos, todos los sitios web y aplicaciones de la universidad, individualmente, verifican la pertenencia del usuario a la institución.

Cada aplicación y servicio tiene un mecanismo de autenticación individual que depende de los dos mencionados directorios que contienen a los usuarios.

## **Problemática**

La Universidad brinda servicios como Wi-Fi, correo, proxy y EVEA (Entorno Virtual de Enseñanza y Aprendizaje), imprescindibles para el funcionamiento de la institución. También brinda otros servicios más especializados como los sistemas de contabilidad, recursos humanos, inventarios, el registro de notas, entre otros. Todos ellos utilizan la autenticación de forma individual, lo cual significa que un usuario tiene diversas cuentas a pesar de pertenecer todas a la misma institución.

El sistema implementado actualmente es poco eficiente y requiere de intervención humana constante para corregir y/o restablecer el apropiado funcionamiento de los mecanismos de autenticación. Esto genera tiempos elevados de respuesta y dificulta el mencionado proceso de transformación digital que está siendo llevado a cabo por nuestra universidad.

## **Objetivo**

Con el propósito de presentar una propuesta para solucionar la problemática expuesta anteriormente, se plantean los siguientes objetivos:

## Objetivo General

- Diseñar e implementar un sistema de autenticación centralizada para todos los usuarios de La Universidad de La Habana.

## Objetivos Específicos

- Generar los servicios de autenticación con compatibilidad con todas las tecnologías existentes y previstas en la institución.
- Implementar la gestión de control de acceso a todos los servicios ofrecidos por el Nodo Central de forma extensible a futuros servicios y fuentes de datos.

## Estructura de la Tesis

En el **Capítulo 1** (Métodos para la autenticación central 1) se investiga sobre las herramientas utilizadas actualmente para la realización de sistemas de autenticación y su estado del arte. Se estudia sobre le inicio de sesión único 1.2, protocolos de autenticación 1.3 y gestores de usuarios 1.4. También se analizan varios de los servicios administradores de identidad 1.5 que se utilizan en la industria.

En el **Capítulo 2** (Propuesta de sistema central de autenticación 2) se hace una propuesta detallada de un sistema central de autenticación. Para ello se estudian y analizan las causas por las cuales se decide implementar este sistema. Además, se enumeran los requisitos del software 2.1.1 y se detalla por etapas la implementación del sistema: base de datos 2.2, ingestión de datos 2.3, capa de autenticación 2.4, clientes 2.5 y usuarios 2.6.

En el **Capítulo 3** (Elección de herramientas y detalles de la implementación 3) se desarrolla la implementación del concepto descrito en el capítulo anterior. Se explican las herramientas utilizadas 3.1 para la ingestión de datos y para la autenticación basada en *tokens*. También se lleva a cabo la implementación del sistema 3.2.2 y se describe el flujo de información durante la autenticación 3.3.

En el **Capítulo 4** (Experimentación y resultados 4) se procede a la experimentación del sistema a través de un cliente de Keycloak para la obtención del *token* realizado en *Python* 4.1.

# Capítulo 1

## Métodos para la autenticación central

En este capítulo se brindan las definiciones de herramientas utilizadas para la autenticación. También se realiza un estudio sobre el estado del arte de las mismas. Además, se brindan razones para incluir su utilización como parte de la solución propuesta.

### 1.1. Autenticación y Autorización

La autenticación y la autorización son los primeros parámetros de seguridad que las aplicaciones web consideran para protegerse de accesos no autorizados [2]. Aunque los dos términos aparecen a menudo en el mismo contexto, los dos son conceptualmente muy diferentes. Autenticar es confirmar una identidad, mientras autorizar significa dar acceso a un sistema. En pocas palabras, la autenticación es el proceso de verificar una identidad y la autorización es el proceso de verificar que tiene acceso [3].

#### 1.1.1. Autenticación

La autenticación consiste en validar credenciales, como nombre de usuario/correo electrónico y contraseña, para verificar la identidad del visitante. El sistema determina si el usuario está usando las credenciales correctas. Tanto en redes públicas como privadas, el sistema autentica al usuario a través del inicio de sesión y generalmente se realiza mediante un nombre de usuario y una contraseña. Sin embargo, la autenticación puede realizarse también a través de otros factores como el código de verificación de teléfonos móviles o datos biométricos.

En algunos sistemas de aplicaciones, para lograr una mayor seguridad, requieren el uso conjunto de múltiples factores de autenticación, a menudo llamado *Multifactor*

*Authentication.*

### **Autenticación basada en *tokens***

Los *tokens* de acceso se utilizan en la autenticación basada en *tokens* para permitir que una aplicación acceda a una API. Cuando un usuario se autentica y es autorizado correctamente, la aplicación recibe un *token* de acceso que luego utiliza como credencial cuando llama a la API. El *token* pasado informa a la API que el portador del *token* ha sido autorizado para acceder y realizar un grupo de acciones. [4]

Un *token* consiste en datos relacionados con la identidad de un usuario particular. Los usuarios pueden obtener dichos *tokens* a través de una combinación de nombre de usuario/contraseña, lo que les permite acceder a los recursos solicitados por un periodo específico de tiempo. Durante este periodo no se requieren métodos adicionales de autenticación. Una característica importante de los *tokens* es que permiten ser heredados a otros usuarios, por ello, la autenticación basada en *tokens* es adecuada cuando un *token* puede proveer acceso a múltiples servicios. [5]

Existen diversos protocolos de autenticación basados en *tokens*, los más populares son: SMAL 2.0 [6], OpenID [7] [8] y OAuth [9] [10].

#### **1.1.2. Autorización**

La autorización ocurre después de que el sistema completa la autenticación. Es el proceso de determinar si un usuario autenticado puede acceder a un recurso en particular. Verifica que está autorizado para otorgarle acceso a recursos tales como información, bases de datos, archivos, etc.

Autenticación	Autorización
Confirma la identidad para otorgar acceso al sistema	Determina si tiene acceso a un recurso
Este es el proceso de validación de credenciales de usuario para obtener acceso de usuario	Este es el proceso de verificar que el acceso está permitido
Generalmente requiere un nombre de usuario y una contraseña	Los factores de autenticación son necesarios para que la autorización pueda variar según el nivel de seguridad
La autenticación es el primer paso	La autorización se realiza después de una autenticación exitosa

Tabla 1.1: Comparación entre autenticación y autorización.

## 1.2. Inicio de Sesión Único

El Inicio de Sesión Único (en inglés Single Sign-On o también conocido por sus siglas SSO) ha sido ampliamente adoptado para la autenticación en línea debido a su utilidad y la seguridad que ofrece. Este es un método de autenticación que permite a los usuarios iniciar sesión con un único conjunto de credenciales en varios sistemas independientes. El inicio de sesión único facilita que un usuario no tenga que iniciar sesión en cada aplicación que use. Con este servicio los usuarios pueden acceder a todas las aplicaciones necesarias sin tener que autenticarse con otras credenciales. [11]

En el mundo digital actual, los usuarios acceden a múltiples sistemas para llevar a cabo sus quehaceres. A medida que aumenta la cantidad de sistemas, también aumenta la cantidad de credenciales de cada usuario y, por lo tanto, también crece la posibilidad de perderlas u olvidarlas. El inicio de sesión único se puede utilizar para resolver muchos problemas relacionados con múltiples credenciales para diferentes aplicaciones.

El acceso de inicio de sesión único al centro de autenticación principal permite a los usuarios obtener acceso a todos los demás recursos disponibles. SSO ayuda a mejorar la productividad del usuario y del desarrollador, al evitar que el usuario recuerde

varias contraseñas y también reduce la cantidad de tiempo que este dedica a escribir varias contraseñas para iniciar sesión. SSO también simplifica la administración mediante la gestión de credenciales únicas en lugar de múltiples credenciales. Facilita la gestión de los derechos de un usuario que llega, cambia de función dentro o sale de la empresa, para integrar rápidamente aplicaciones adicionales, delegar derechos de acceso durante las vacaciones sin aumentar la carga de trabajo del equipo de soporte. [12]

Grandes empresas como Google, Facebook y Microsoft utilizan estos servicios. En particular Google permite a sus usuarios iniciar sesión una única vez para tener acceso a todos los servicios de la empresa que se encuentran en la nube. Cuando se configura SSO, los usuarios pueden iniciar sesión en terceros proveedores de identidad y luego acceder a las aplicaciones de Google directamente sin un segundo inicio de sesión [13]. Por ejemplo, si se accede a un servicio de Google como Gmail, se autentica automáticamente en YouTube, AdSense, Google Analytics, y otras aplicaciones de Google. Del mismo modo, si cierra la sesión de su Gmail u otras aplicaciones de Google, se cerrará automáticamente la sesión de todas las otras aplicaciones; esto se conoce como Cierre de Sesión Único (en inglés: Single Logout) [14]

## 1.3. Protocolos de autenticación

### 1.3.1. OpenID Connect

OpenID Connect (también OIDC) es una capa de identidad simple implementada a partir del protocolo OAuth 2.0. Permite a los clientes verificar la identidad del usuario final en función de la autenticación realizada por un servidor de autorización, así como obtener información básica del perfil del usuario final de manera interoperable y similar al protocolo REST. [15].

OpenID Connect es uno de los protocolos de tipo Single Sign-On más utilizado para delegar la autenticación. También tiene un formato simple, por lo que ha ganado popularidad y es soportado por grandes empresas como Google, IBM, Microsoft, Amazon y PayPal [8]. La nueva versión es compatible con *API* y puede ser usada por aplicaciones nativas y móviles. También define mecanismos opcionales más robustos para firmas y cifrados. [15]

Este protocolo está basado en OAuth 2.0 por lo que tiene todas las ventajas de este protocolo. Sin embargo, lo extiende ya que ofrece facilidades para obtener más información de la identidad de los usuarios eficientemente. Permite un flujo de información adicional que genera un *id-token* que contiene datos del usuario. De esta forma, las aplicaciones no solo tienen acceso a los permisos de los usuarios, sino también obtienen información sobre su identidad. [15] [1]

### 1.3.2. SAML

Lenguaje de Marcado para Confirmaciones de Seguridad, conocido como SAML (en inglés: *The Security Assertion Markup Language*) es un marco de trabajo que permite expresar asertos<sup>1</sup> acerca de la identidad, los atributos y las autorizaciones de un sujeto con el objetivo de facilitar las relaciones entre distintas empresas, así como las relaciones de estas con sus usuarios. Este marco de trabajo permite a las compañías crear identidades federadas, lo cual les facilita las tareas de gestión de perfiles, autenticación y autorización de usuarios. El caso típico de uso es el de Single Sign-On (SSO), que permite a los usuarios acceder a diversos sitios en la federación con una única autenticación. [16]

SAML y OpenID Connect son protocolos de identificación, diseñados para autenticar a los usuarios. También proporcionan datos de identidad para el control de acceso y como método de comunicación para la identidad de un usuario.

SAML durante muchos años ha proporcionado un medio seguro de intercambio de datos de identidad, por lo que muchas organizaciones confían en él. También es muy rico en funciones y cubre una amplia gama de requisitos de identidad.

OIDC, al ser más nuevo y encontrarse en desarrollo, todavía está rezagado con respecto a SAML en términos de características. Sin embargo, para muchas aplicaciones donde solo se necesita un requisito simple para los datos de identidad básicos, particularmente en el espacio del consumidor, OIDC es muy útil.

Actualmente, SAML se usa principalmente para la identificación de ciudadanos del gobierno y la autenticación empresarial. Sin embargo, esto está comenzando a cambiar, con sistemas más modernos que utilizan OIDC en lugar de SAML. Esto se debe a que OIDC permite un procesamiento de datos más liviano que SAML, utilizando *tokens* JSON (*token* de ID) en lugar de XML. OIDC es ideal para usar con aplicaciones móviles y aplicaciones web de una sola página, donde el uso de SAML sería complicado. [17]

### 1.3.3. Kerberos

Kerberos es una conexión de software que se emplea en una red grande para establecer la identidad declarada de un usuario. Utiliza una combinación de encriptación y bases de datos distribuidas de tal forma que un usuario pueda registrarse y comenzar una sesión desde cualquier computadora localizada en la red mediante la obtención de tickets para servicios de un servidor especial conocido como TGS (servidor despachador de tickets); cada ticket contiene información para identificar al usuario o servicio

---

<sup>1</sup>Los asertos definen las afirmaciones de seguridad de una entidad dentro de un sistema. Estas afirmaciones pueden ser de tres tipos: asertos de autenticación, de atributos y de decisiones de autorización.

encriptada con la clave privada para el servicio. Como sólo Kerberos y el servicio conocen dicha clave, se considera que el mensaje está genuinamente originado en la fuente y que no fue adulterado en el transporte del mismo. El ticket otorgado por el TGS contiene una nueva clave de sesión que solo conoce el cliente y el servicio afectado. Esta clave será utilizada para encriptar las transacciones que ocurren durante la sesión. Una de las ventajas es que el ticket tiene un tiempo de vida específico, y una vez que éste expira, debe solicitarse un nuevo ticket al TGS para poder seguir utilizando el servicio. Para cada servicio se requiere un ticket distinto. Otra ventaja es que el usuario no debe reingresar la contraseña cada vez que requiere un servicio, porque si el ticket TGS no expiró puede reusarlo para pedir otro ticket de servicio deseado. Por este motivo, el tiempo de vida del ticket TGS deberá ser mayor que el tiempo de vida del ticket de servicio.

## 1.4. Gestores de usuarios

### 1.4.1. LDAP

El Protocolo Ligero de Acceso a Directorios (en inglés: *Lightweight Directory Access Protocol*, también conocido por sus siglas de LDAP) es un conjunto de protocolos de licencia abierta que son utilizados para acceder a la información que está almacenada de forma centralizada en una red. Este protocolo se utiliza a nivel de aplicación para acceder a los servicios de directorio remoto. [18]

LDAP está basado en estándares implementados sobre TCP/IP. Permite a los clientes interactuar directamente con los servidores de los directorios: almacenar y consultar información, buscar datos filtrados, autenticar usuarios, entre otros.

Este protocolo es utilizado actualmente por muchos sistemas que apuestan por el software libre al utilizar distribuciones de Linux para ejercer las funciones propias de un directorio activo en el que se gestionarán las credenciales y permisos de los usuarios y estaciones de trabajo en redes LAN corporativas en conexiones cliente/servidor.

Un directorio remoto es un conjunto de objetos que están organizados de forma jerárquica, tales como: nombre, claves, direcciones, etc. Estos objetos estarán disponibles para una serie de clientes conectados mediante una red, normalmente interna o LAN, y proporcionarán las identidades y permisos para esos usuarios que la utilicen.

LDAP está basado en el protocolo X.500 para compartir directorios, y contiene esta información de forma jerarquizada y mediante categorías para proporcionarnos una estructura intuitiva desde el punto de vista de la gestión por parte de los administradores.

Estos directorios se utilizan generalmente para contener información virtual de usuarios, para que otros usuarios accedan y dispongan de información acerca de los contactos que están aquí almacenados. Además, es capaz de comunicarse de forma

remota con otros directorios LDAP situados en servidores que pueden estar en el otro lado del mundo para acceder a la información disponible. De esta forma se crea una base de datos de información descentralizada y completamente accesible.

El sistema de autenticación vigente en el Nodo Central verifica sus usuarios con dos sistemas implementados con LDAP. Este protocolo se adapta a las necesidades y condiciones actuales de la Universidad con la ventaja importante de que es *Open Source* (OSS o código abierto).

#### 1.4.2. Active Directory

Directorio Activo (en inglés: *Active Directory*, conocido también por sus siglas AD) es un servicio de directorios desarrollado por Microsoft que permite almacenar información como usuarios y dispositivos en una base de datos centralizada y jerárquica. AD brinda servicios como autenticación, políticas de acceso y administración de grupos.

*Active Directory* almacena información sobre objetos en la red y hace que esta información sea fácil de encontrar y usar para administradores y usuarios. Utiliza un almacén de datos estructurados como base para una organización lógica y jerárquica de la información del directorio.

La seguridad está integrada con *Active Directory* a través de la autenticación de inicio de sesión y el control de acceso a los objetos del directorio. Con un solo inicio de sesión en la red, los administradores pueden gestionar los datos del directorio y la organización en toda su red, y los usuarios autorizados de la red pueden acceder a los recursos en cualquier lugar de la red. La administración basada en políticas facilita la gestión incluso de la red más compleja. [19]

	<b>LDAP</b>	<b>AD</b>
<b>Nombre Completo</b>	Protocolo Ligero de Acceso a Directorios	Directorio Activo
<b>Función</b>	Protocolo	Proveedor de servicios de directorios
<b>Standard</b>	Código Abierto	Propietario
<b>Sistemas Soportados</b>	Multiplataforma: Windows, Linux, macOS	Para aplicaciones y usuarios de Windows
<b>Uso principal</b>	Consultar y modificar elementos en proveedores de servicios de directorio	Proveer autenticación, políticas, administración de grupos y usuarios, y muchos otros servicios en forma de una base de datos de directorio

Tabla 1.2: Comparación entre LDAP y Active Directory.

## 1.5. Servicios administradores de identidad

### 1.5.1. Okta

Okta es un servicio que proporciona y simplifica la administración de los sistemas de identidad. Los productos que ofrece más destacados son SSO y la autenticación multifactor.

### 1.5.2. Gluu

Gluu es una plataforma de código abierto gratuita que proporciona a las organizaciones un servicio de autenticación y autorización para las aplicaciones web y móvil. Permite configurar SSO en aplicaciones que tengan soporte para OpenID Connect, SAML o CAS para identidades federadas.

### 1.5.3. Auth0

Auth0 es una plataforma en la nube que ofrece la autenticación y la autorización como un servicio. Auth0 dispone de herramientas para simplificar la autenticación de las aplicaciones y APIs ya que hace uso de estándares como OAuth2.0, OpenID

Connect, SAML 2.0, JSON Web Token o WS-Federation, ofreciendo SSO a entornos empresariales

#### 1.5.4. Keycloak

Keycloak es un software de código abierto que permite el Single Sign-On o Inicio de Sesión Único con *Identity Management* y *Access Management* para aplicaciones y servicios modernos. Esta herramienta facilita la protección de aplicaciones y servicios con poca o ninguna codificación. Un proveedor de identidad (en inglés: *Identity Provider*, también conocido por sus siglas IdP), permite que una aplicación (a menudo llamada *Service Provider* o SP) delegue su autenticación. [20]

- **OAuth2:** se trata de un estándar de código abierto que se ha diseñado como protocolo de comunicación entre servicios con la ventaja de que permite compartir información sin exponer la identidad de los usuarios en las peticiones. Esto es una utilidad bastante relevante y es el motivo principal por el que el protocolo de comunicaciones OAuth ha sido adoptado cada vez por más empresas en la industria del desarrollo de aplicaciones y servicios web. La versión actual de este protocolo es 2.0, de ahí su nombre.
- **IdP:** proveedor de identidad usado por el protocolo OAuth2 que almacena y gestiona las entidades digitales que refieren a los usuarios.
- **Identity Management o IAM:** se refiere al sistema integrado que proporciona las herramientas para gestionar el ciclo de vida de los usuarios y sus accesos dentro de la organización, automatizando las altas, bajas y modificaciones de las cuentas de los usuarios y sus privilegios, atributos, roles, permisos, etc.
- **Access Management:** se refiere a la configuración del intercambio de comunicación y apertura de medios de comunicación entre servicios. Esta configuración garantiza el control y monitorización de los accesos a un servicio. Por ejemplo, se garantiza que un administrador o desarrollador el cual posee mayores privilegios no conozca las contraseñas o credenciales de cuentas con privilegios inferiores, y viceversa.

Este software está escrito en Java y es compatible de forma predeterminada con los protocolos de federación de identidad SAML v2 y OpenID Connect (OIDC) / OAuth2. Está bajo licencia de Apache y es mantenido por Red Hat. [20]

## Características

Los usuarios se autentican en Keycloak en lugar de hacerlo en las aplicaciones. Esto significa que no es necesario que cada aplicación tenga un formulario de inicio de sesión, autentique a los usuarios o almacene sus datos. Una vez entren en Keycloak, los usuarios no tendrán que iniciar sesión en las demás aplicaciones conectadas al software.

Lo mismo sucede cuando un usuario cierra sesión. Keycloak ofrece cierre de sesión único, lo cual significa que los usuarios solo tienen que desconectarse en una de las aplicaciones para salir de su cuenta en el resto.

Otra prestación de Keycloak son las federaciones de usuarios, que facilitan la compatibilidad con LDAP y otros servidores de directorios activos. También admite la implementación de servicios propios para usuarios guardados en otros tipos de almacenamientos como en bases de datos relacionales.

Keycloak ofrece como herramienta una consola de administración de cuentas, a través de la cual los usuarios pueden administrar sus propias cuentas. Pueden actualizar su perfil, cambiar sus contraseñas y configurar la autenticación en dos pasos. También pueden administrar sus sesiones y visualizar el historial de su cuenta.

Otra característica es que es una herramienta extensible porque permite la eliminación, adición y modificación de las bases de datos de usuarios, los métodos de autenticación y los protocolos. Está basada en protocolos estándares y soportan OpenID Connect, OAuth 2.0 y SAML. [20]

Keycloak facilita añadir la autenticación y un servicio seguro a aplicaciones. Permite que los desarrolladores se centren en la funcionalidad empresarial al no tener que preocuparse por los aspectos de seguridad de la autenticación. También posibilita la unificación de los métodos de autenticación de distintas aplicaciones sin modificarlas.

### 1.5.5. JSON Web Token

JSON Web Token (abreviado JWT) es un estándar abierto basado en JSON propuesto por IETF (RFC 7519) para la creación de *tokens* de acceso que permiten la propagación de identidad y privilegios [21].

Esta tecnología define una forma compacta y autónoma para transmitir de forma segura información entre las partes como un objeto JSON. Esta información es verificada y confiable ya que se encuentra firmada digitalmente. Los JWT se pueden firmar usando un secreto (con el algoritmo HMAC) o un par de claves públicas / privadas usando RSA.

JSON Web Token es un método compacto y autónomo para transmitir información, se basa en una cadena de texto que tiene 3 partes (*Header*, *payload*, *signature*) codificadas en Base64, separadas por un punto que es entregado a los clientes de una API como llave de acceso.

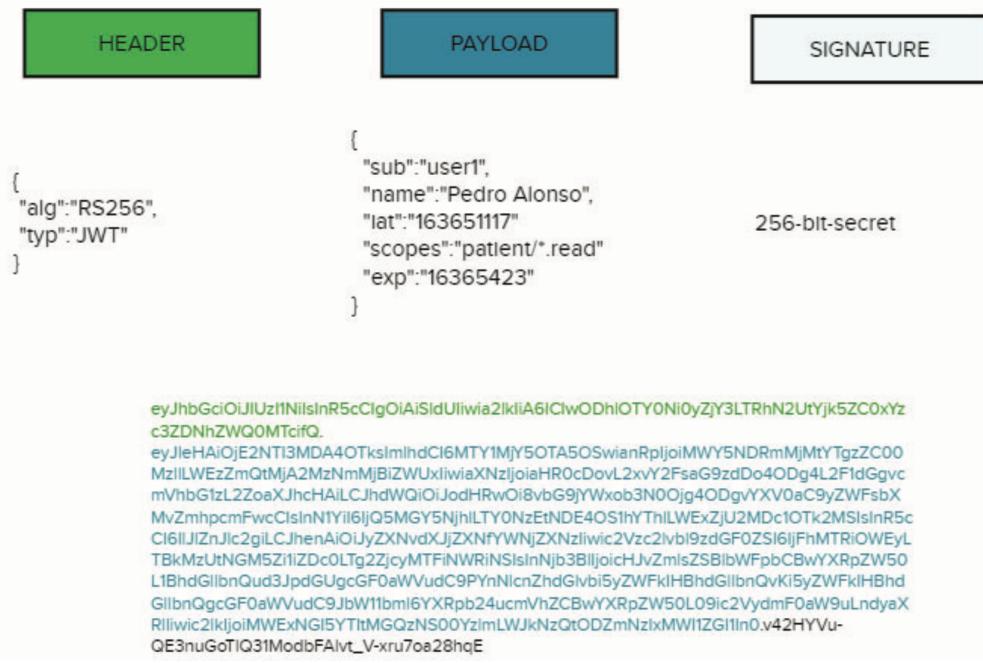


Figura 1.1: Estructura de JWT

- El **header** contiene la cabecera del objeto, con la información del tipo de objeto y el sistema criptográfico que lleva aplicado. En nuestro caso coincide con RS256 asimétrico, como principal algoritmo aceptado en el sistema aunque no es el único contemplado, junto al HS256 simétrico.
- El **payload** contiene la información que le concierne al usuario, así como los permisos y ámbito del mismo, fechas de expiración del *token*, etcétera.
- La **signature** se refiere a la firma del objeto JWT: contiene la información del emisor de dicho objeto para identificarlo y asegurar que dicho mensaje y su contenido no ha sido modificado durante el envío.

Existen diversas herramientas de cifrado y descifrado objetos JWT y cualquier modificación de los mismos en el transcurso de la comunicación los convertiría en un objeto no válido. Se consideran elementos atómicos que solo comparten emisor y receptor mediante una clave pública y privada, únicas y situadas una en cada polo de la comunicación.

El sistema de encriptación que del que se hace uso es RS256 principalmente. Este método consiste en la creación de una clave privada que se situaría en el emisor, y se usa para generar la firma o *signature*; y una clave pública que estaría situada en

el receptor para validar dicha firma. De esta forma, se asegura que emisor y receptor son los únicos participantes en la comunicación. [22]

Keycloak utiliza JWT para transmitir una llave secreta de acceso a usuarios con privilegios. [23]

### **1.5.6. REST API**

Se utiliza API REST como medio de prueba para ejecutar peticiones HTTP que posteriormente serán aseguradas con un Json Web Token emitido por Keycloak. La arquitectura REST enfoca a todo lo que lo conforma como un recurso. Los servicios web REST son livianos, altamente escalables y fáciles de mantener. Se usan muy comúnmente para el intercambio de información, es el estándar más lógico, eficiente y generalizado en la creación de API para servicios de internet.

El siguiente esquema muestra cómo se realiza el flujo de información para autenticarse a través de Keycloak:

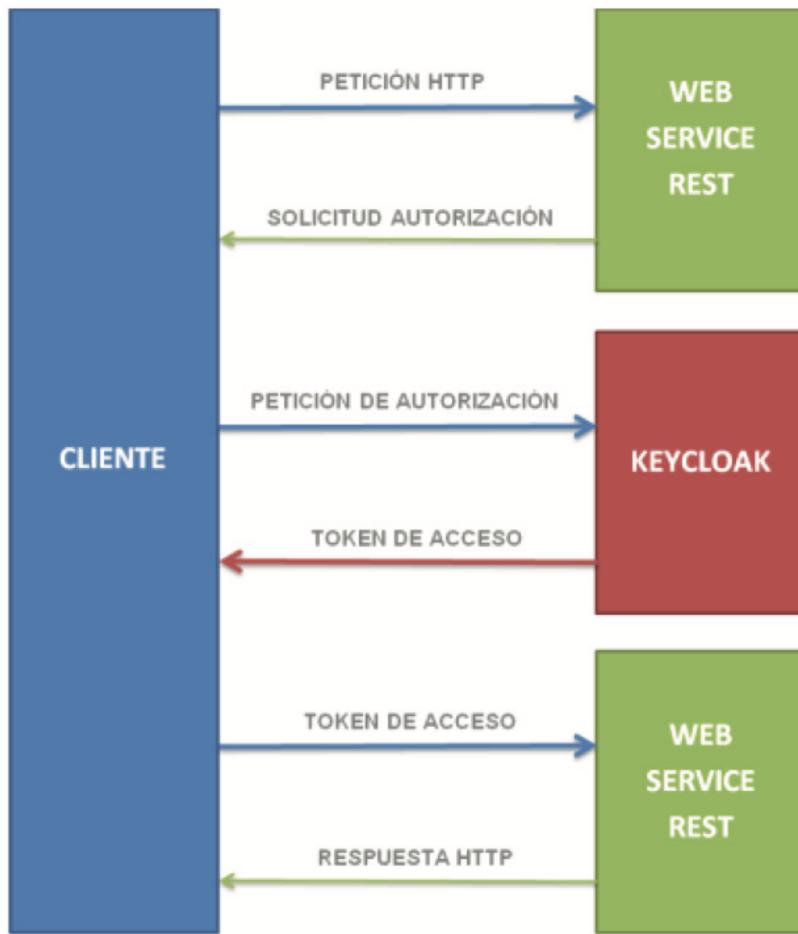


Figura 1.2: Flujo de información en Keycloak

Este flujo asegura que solo los usuarios a los que se les otorgó el acceso al servicio web puedan consumir dicha API. De esta forma se evita el acceso no autorizado a información que se transmite bajo un protocolo no seguro como lo es el protocolo HTTP. [23]

# **Capítulo 2**

## **Propuesta de Sistema Central de Autenticación**

En el siguiente capítulo se presenta una solución al problema principal de la tesis. Se presenta la hipótesis del problema y los requisitos del software. También se explica por capas las varias etapas que el software debe tener: bases de datos, ingestión de datos, capa de autenticación, clientes y usuarios. En cada una de estas se analizan los problemas que debe resolver y se estudia cómo deben ser resueltos.

### **2.1. Hipótesis**

La Universidad de la Habana cuenta con varios sistemas en la red donde se utiliza el método tradicional de usuario/contraseña como mecanismo de seguridad para acceder a diversas aplicaciones, este proceso de autenticación se hace muy complejo al tener que acceder a cada uno de ellos de forma independiente. Con cada servicio nuevo se debe crear un sistema de autenticación que garantice la seguridad de sus datos y se debe hospedar la información de los usuarios repetidas veces, lo cual utiliza una mayor cantidad de recursos y es más propenso a fallas.

Para eliminar estas dificultades se diseña un sistema central de autenticación que se basa en el método Inicio de Sesión Único. Esta propuesta tiene como propósito un aumento en la productividad, ofrecer mayor facilidad de acceso a los recursos, funciones de autenticación a través de una única plataforma, una administración sencilla de credenciales y sobre todo garantizar un aumento de la seguridad. Mediante este servicio el usuario podrá registrarse en el sistema una sola vez, con lo cual podrá acceder a todos los recursos sin tener que volver a autenticarse.

### 2.1.1. Requisitos del Software:

- Unificar las distintas fuentes de datos.
- Que el usuario inicie sesión y, hasta que cierre sesión, sea capaz de realizar operaciones sin tener que volver a introducir credenciales.
- El servicio ofrecido al cliente debe permitir que el usuario extienda la sesión una vez pasado el tiempo de expiración de la misma sin tener que volver a introducir credenciales.
- Reconocer la identidad de los usuarios durante el proceso de autenticación para garantizar un adecuado control de acceso a los recursos del sistema.
- Proteger los recursos del sistema, permitiendo que estos sean solamente usados por aquellos usuarios a los que se les ha concedido autorización.
- Que el usuario inicie sesión con dos únicos campos: nombre de usuario y contraseña. Este requisito será suficiente para garantizar la interoperabilidad del sistema, que debe ser capaz de generar un objeto encriptado con toda la información relativa a dicho usuario y viajar por la red de comunicaciones entre las distintas entidades.
- La respuesta obtenida al iniciar sesión de forma exitosa debe ser un objeto que le dé portabilidad y reusabilidad al software.
- En caso de obtener un inicio de sesión erróneo, retornar un error.
- Garantizar el control de errores y excepciones.
- La evaluación de permisos de acceso.

El software consta de varias etapas:

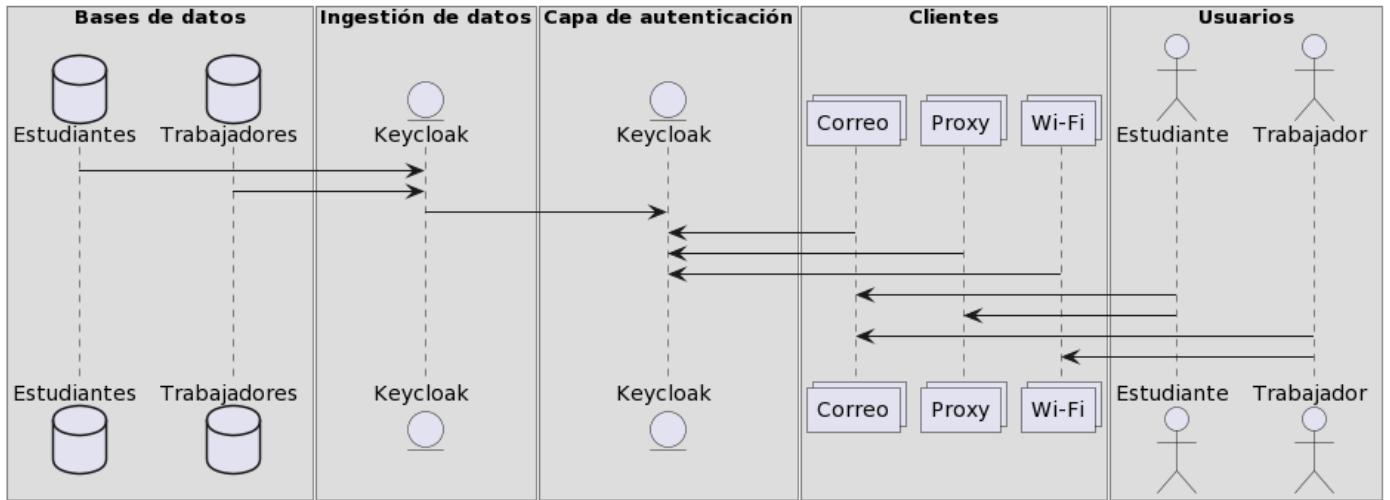


Figura 2.1: Etapas de implementación del software

## 2.2. Bases de Datos

Actualmente los estudiantes de toda la Universidad de La Habana al matricularse se inscriben en Secretaría en el Sistema de Gestión de la Nueva Universidad (SIGE-NU). En este sistema se almacenan todos los datos de los alumnos: datos personales, proveniencia, notas, grados.

Por otra parte, los datos de los profesores y del resto de los trabajadores de la universidad son guardados en bases de datos de ASSETS, software contratado. Cada unidad presupuestada de la universidad tiene su propio ASSET donde se almacena los datos de recursos humanos y del inventario.

El Nodo Central es el responsable de todas las comunicaciones de la Universidad de La Habana. Presta servicios a todas las facultades, desde las que se encuentran en la Colina, hasta facultades externas como Economía, el Jardín Botánico y la Quinta de los Molinos. Cada facultad tiene su sistema independiente, que responde a sus necesidades, donde gestiona a todo su personal.

La información es almacenada en varias bases de datos, lo cual hace difícil la gestión de todos los usuarios de la Universidad. Crear un sistema único al cual todas las facultades pudieran acceder de forma remota simplificaría el trabajo, no solo a los programadores, sino también a los usuarios. Por ejemplo, cada estudiante podría matricularse en su propia facultad y se evitarían las aglomeraciones y la confusión creada cada año.

Sin embargo, sería complicado cambiar el software utilizado por toda la institución. Como se mencionó anteriormente, algunas facultades se encuentran alejadas de La

Colina, lo cual dificulta la comunicación y la instalación de nuevos sistemas. También se debe tener en cuenta que la experiencia del usuario sería diferente con un nuevo software. El personal de la universidad es de variadas edades, por lo que podría ser complejo la adaptación a una nueva interfaz.

Por lo tanto, en el presente trabajo se ha decidido no cambiar los sistemas. La unificación de las bases de datos de usuario de toda la Universidad de La Habana es un proceso complejo que no se encuentra en los objetivos del presente trabajo.

### 2.3. Ingestión de Datos

La ingesta de datos es el proceso mediante el cual se introducen datos, de diferentes fuentes, estructura o características dentro de otros sistemas de almacenamiento o procesamiento de datos. [24]

Teniendo varias fuentes de datos, para garantizar una ingestión de datos exitosa, la solución debe ser capaz de leer información de distintas fuentes de datos.

Por otra parte, el estado de un usuario puede cambiar y sus permisos de acceso pueden variar. Por ejemplo, a los estudiantes se les puede dar baja o un trabajador puede terminar su contrato y, por lo tanto, se les debe retirar sus credenciales de forma inmediata. Un estudiante después de graduarse puede pasar a ser trabajador de la institución, por lo que sus permisos de acceso deben ser cambiados. Otro caso sería cuando un estudiante o trabajador sale de licencia, deben suspenderse sus credenciales temporalmente hasta que la persona regrese.

Se necesita que el sistema de autenticación se actualice constantemente ya que la información en las bases de datos cambia todo el tiempo.

La Universidad se encuentra en constante cambio. Por ejemplo, en 2017 culminó el proceso de adscripción del Instituto Superior de Diseño (ISDI) a la Universidad de La Habana [25]. Para ello fue necesario incorporar todos los datos de los usuarios del ISDI al sistema de autenticación vigente para otorgarles las credenciales correspondientes. Por ello, el nuevo sistema debe ser capaz de admitir nuevas fuentes de datos de forma sencilla en un futuro.

En el nodo central, la ingesta de datos se hace a través de LDAP.

### 2.4. Capa de Autenticación

El proceso de autenticación consiste en la verificación de la identidad de un usuario o una entidad [26]. La certificación de credenciales es un área clave en la seguridad de la información. En la modernidad los usuarios necesitan acceder a muchos servicios digitales imprescindibles para su vida cotidiana. Las contraseñas basadas en caracteres alfanuméricos han sido las más comunes en todo tipo de sistemas por su fácil

implementación [27].

En este caso la capa de autenticación es la encargada de recibir las peticiones de los distintos clientes y verificar la identidad de un usuario a partir de la información recopilada por la ingestión de datos.

Los usuarios de la Universidad de La Habana utilizan diariamente distintos servicios, que van desde acceder a Internet por la red WI-FI o por proxy, hasta servicios web como EVEA y correo. Cada servicio tiene su propio sistema de autenticación, lo cual causa molestias y dificultades en la interacción con las plataformas de los usuarios y complejiza el trabajo de los programadores.

Resulta engorroso crearse una cuenta en cada uno de los sistemas y autenticarse cada vez que uno acceda a un sitio nuevo. La creación de un sistema en el cual todos los usuarios de la Universidad puedan autenticarse con su correo de la institución y su contraseña facilitaría la interacción diaria con los servicios ofrecidos por el Nodo Central.

Como se ha explicado, el nuevo sistema unificará la autenticación de todos los servicios prestados por la Universidad. Para ello se utilizará el método Single Sign-On, mecanismo que permite a un usuario acceder a múltiples servicios utilizando una sola credencial. Esto significaría que si se tiene una cuenta de correo con la que se puede acceder a un sitio, no será necesario registrarse en el resto de los servicios.

El programa también debe garantizar la seguridad de las cuentas. Por ejemplo, los estudiantes utilizan las máquinas de los laboratorios y se autentican con su cuenta personal. Es común que las personas olviden cerrar sus sesiones, por lo que la cuenta debe expirar al pasar un tiempo prudencial, de lo contrario, su información puede ser utilizada por otra persona. La utilización de la autenticación basada en *tokens* facilita el cierre de sesión de forma automática luego de un determinado tiempo.

Este sistema permitirá a todos sus clientes conectarse de forma segura y sencilla para que verifiquen las credenciales de sus usuarios. La universidad actualmente ofrece muchos servicios, por lo que se debe encontrar la forma óptima de migrar los clientes a la nueva forma de autenticar.

## 2.5. Clientes

El cliente es un programa ejecutable que participa activamente en el establecimiento de las conexiones. Envía una petición al servidor y se queda esperando por una respuesta. [28]

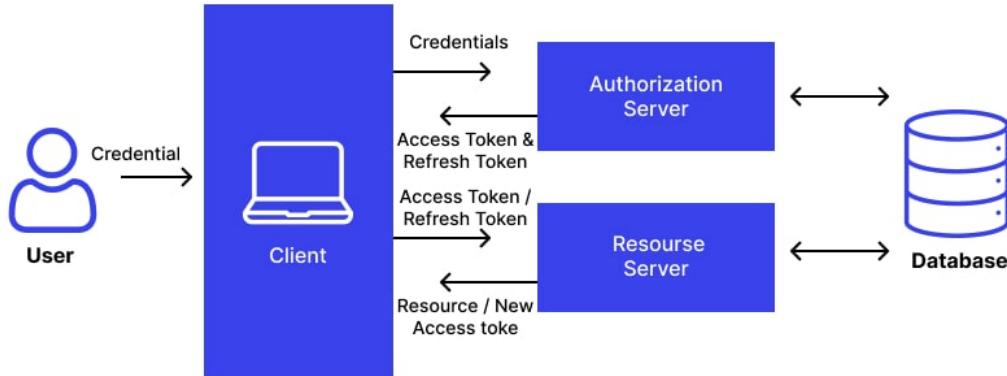


Figura 2.2: Flujo de información desde el cliente hasta la base de datos

El Nodo Central brinda infraestructura a los clientes de la Universidad vitales para el correcto funcionamiento de la institución. Entre sus responsabilidades se encuentran la gestión de correo, el acceso a Internet, las cuentas de usuarios y en el caso de los profesores gestión de cuentas de VPN. También tiene como clientes todos los servicios web de la casa de altos estudios, entre ellos el sitio oficial de la Universidad <https://www.uh.cu/> y los sistemas utilizados en recursos humanos y en los departamentos de contabilidad.

La creación de un sistema central de autenticación facilitaría el trabajo de los programadores que se ven obligados a crear un nuevo servicio de autenticación para cada aplicación. Sería más eficiente respecto a los recursos utilizados porque se unificarían todas las bases de datos utilizadas por cada programa para mantener los registros de sus usuarios. El tráfico de información se minimizaría ya que los usuarios solo tendrían que autenticarse una única vez para tener acceso a todos los servicios de la Universidad, por lo que se reducen las congestiones en los servidores y el tiempo de respuesta. También disminuirían las brechas de seguridad que existen y, por tanto, se minimizaría el volumen de información sensible en la red expuesta a ataques externos.

El software debe ser capaz de admitir un alto número de clientes y debe facilitar la adición de uno nuevo.

Uno de los clientes más importantes es el Entorno Virtual de Enseñanza y Aprendizaje, más conocido como EVEA, creado en 2018. El sitio es una plataforma informática encargada de orientar la comunicación pedagógica entre los universitarios que intervienen en el proceso educativo. También, tiene la misión de crear espacios o comunidades organizadas en torno al estudio. [29]

Este entorno virtual representó un apoyo para el sistema universitario cuando los

estudiantes no podían asistir a sus facultades debido a la pandemia provocada por la Covid-19 en diciembre del 2019 [30]. En la primera mitad del año 2020 la cantidad de usuarios aumentó considerablemente en poco tiempo. El sistema era lento y tenía muchas fallas. Uno de los mayores retos fue lograr registrar a todos sus usuarios y que estos se pudieran autenticar. De haber existido un sistema que ya autenticara a todos los estudiantes y profesores de la universidad, los programadores habrían tenido un problema menos a la hora de modificar la plataforma para adaptarla al nuevo uso que se le daría.

## 2.6. Usuarios

Los usuarios de la Universidad de La Habana se encuentran en un amplio rango de edades. Se encuentran matriculados 38 678 estudiantes en cursos de pregrado, maestrías, doctorados y otros cursos. Entre los trabajadores se encuentran 600 docentes, personal de administración y servicio, economistas, entre otros. Es un grupo muy heterogéneo, lo cual se debe tener en cuenta para el desarrollo de una aplicación.

Con el uso de distintos sistemas de autenticación, los usuarios se ven obligados a tener más de una credencial lo cual crea confusión. Frecuentemente se olvidan los nombres de usuarios o contraseñas o se utilizan las mismas credenciales en varios sitios, lo cual es una mala práctica desde el punto de vista de la seguridad y es confuso y complejo para muchos. Es muy fácil olvidar las credenciales, lo cual obliga a los gestores del sistema del Nodo Central a enviar un correo no encriptado con información confidencial. Por otra parte, frecuentemente los usuarios se ven obligados a dirigirse a los especialistas por problemas con sus cuentas.

Un nuevo sistema centralizado de autenticación beneficiaría a los usuarios ya que solo tendrían una cuenta para acceder a todas las aplicaciones. También será más fácil gestionar una sola cuenta.

# Capítulo 3

## Elección de herramientas y detalles de implementación

En el siguiente capítulo se toman decisiones claves respecto a qué herramientas utilizar para lidiar con los problemas que surgen al autenticarse. También se proveen instrucciones específicas para la ejecución y configuración de las herramientas mencionadas. Además, se brindan los pasos a seguir para la implementación de un sistema de autenticación central.

### 3.1. Elección de herramientas

En la siguiente propuesta se utiliza LDAP como herramienta para ingerir datos. Como ya se ha explicado anteriormente, LDAP es el programa que se utiliza actualmente para unificar las distintas bases de datos de usuarios de toda la Universidad de La Habana. Para el manejo de LDAP se utilizará *Apache Directory Studio* (*ApacheDS*), un software libre que permite la gestión de directorios.

También se utilizará Keycloak, una solución de gestión de acceso e identidad de código abierto dirigida a aplicaciones y servicios modernos. Esta facilita la protección de aplicaciones y servicios con poco o ningún código [23].

Keycloak es una herramienta que se encarga de abstraer la parte de autenticación de usuarios y almacenamiento de la información privada de estos, del flujo de comunicación entre cliente y servidor. Ofrece Single Sign-On a través de una autenticación basada en *tokens* y está diseñada para que sea sencillo añadir nuevos servidores de usuarios y clientes.

Esta herramienta contiene una alta elegibilidad de configuración en sí misma. Su papel es meramente de intermediario entre servicios externos, por lo que cuando un usuario quiere iniciar sesión en una aplicación, la autenticación se le transfiere directamente al servicio Keycloak, que se encarga de generar los objetos JWT que se

envían necesariamente entre servicios para transmitir los datos de dicho usuario.

Este servicio ofrece un amplio abanico de posibilidades en la configuración del acceso entre servicios. Se ha elegido frente a otras soluciones de código abierto como STYTCH, Ory, Okta o Auth0 porque Keycloak ofrece un nivel más alto en cuanto a rendimiento, escalabilidad y disponibilidad [22].

Gluu es otra de las tecnologías que tienen prestaciones y ventajas similares a Keycloak. Es un servicio *open source* que soporta SAML, OpenID Connect, SSO y OAuth 2.0. Sin embargo, Gluu es un sistema que requiere de 8 GB de RAM y 40 GB de espacio en disco, mientras que Keycloak solo necesita de 512 Mb de RAM y 1 GB de disco. Por ello la segunda tecnología se ajusta más a los recursos con que cuenta la Universidad de La Habana [31].

Para la experimentación se creará un cliente capaz de interactuar con el sistema implementado para autenticar usuarios. Keycloak permite proteger aplicaciones que se ejecutan en diversas plataformas y tecnologías que usan los protocolos OpenID Connect y SAML [32].

Entre los soportes ofrecidos, se ha escogido la biblioteca de Python **Python Keycloak**, que provee fácil acceso a la API de Keycloak. Python es un lenguaje de programación que facilita trabajar rápido e integrar sistemas eficientemente [33]. Podría decirse que durante la última década Python se ha convertido en uno de los lenguajes de código abierto más utilizados.

Para la visualización de la solución se utiliza Flask, un *micro framework* de Python que proporciona las funcionalidades básicas de *web frameworks*.

## 3.2. Ingestión de Datos

### 3.2.1. Herramientas para el uso de LDAP

A partir del protocolo LDAP se han desarrollado diversas implementaciones por parte de algunas empresas o fundaciones. El cuadro siguiente lista algunas de las implementaciones de este protocolo, así como la empresa o fundación detrás de esta y el tipo de licencia con que es distribuido [34]:

Software	Empresa que lo desarrolla	Tipo de licencia
Novell eDirectory	Novell, Inc.	Privativa
Red Hat Directory Server	Red Hat, Inc.	Libre (GPL)
Active Directory	Microsoft Corporation	Privativa
Open Directory	Apple Inc.	Privativa
Apache Directory Server	Apache Software Foundation	Libre (Apache License)
Oracle Internet Directory	Oracle Corporation	Privativa
OpenDS	Sun Microsystems	Libre (CDDL)
OpenLDAP	OpenLDAP Foundation	Libre (OpenLDAP Public License)
IBM Tivoli Directory Server	IBM	Privativa

Tabla 3.1: Diferentes implementaciones del protocolo LDAP.

Teniendo en cuenta la tabla previa, se puede extraer una serie de opciones candidatas de servicios de directorios que tienen la posibilidad de ser implementados, tomando como base solo dos criterios de elección seleccionados: que sea software libre y que sea gratuito. Realizando la clasificación bajo estos lineamientos, las opciones elegibles son:

- Apache Directory Server
- OpenDS
- OpenLDAP

Además, se comprobó un tercer criterio al realizar las pruebas a las opciones candidatas, el cual fue la capacidad de despliegue de dicha solución en diferentes sistemas operativos. OpenLDAP no se encuentra disponible en el sitio oficial para Windows, sistema operativo utilizado en el Nodo Central, por lo cual no se utilizará.

ApacheDS tiene una documentación amplia y es fácil de desplegar. También cuenta con instaladores fáciles de usar para diversos sistemas operativos, entre los cuales se encuentran Linux, Solaris, Mac OS X y Windows. Además, provee el código fuente en caso que sea necesaria su compilación. Por lo tanto, esta será la herramienta utilizada para acceder a LDAP.

ApacheDS cuenta con una herramienta bastante completa para la gestión del directorio llamada Apache Directory Studio la cual está basada en el entorno de

desarrollo Eclipse. Esta se utilizará para acceder a los dos LDAP con los que se interactúa, la de los trabajadores y la de los estudiantes.

### 3.2.2. Keycloak como herramienta para la ingestión de datos

Para la instalación y configuración de Keycloak se siguen las guías: [Keycloak Get Started](#) y [Configuring Keycloak](#). En esta solución se utiliza [keycloak-20.0.1.zip](#).

Se realizan las siguientes configuraciones:

#### 1. Creación de usuario administrador

Para ello se debe abrir <http://localhost:8080/> y llenar el formulario con el nombre de usuario y contraseña de preferencia.

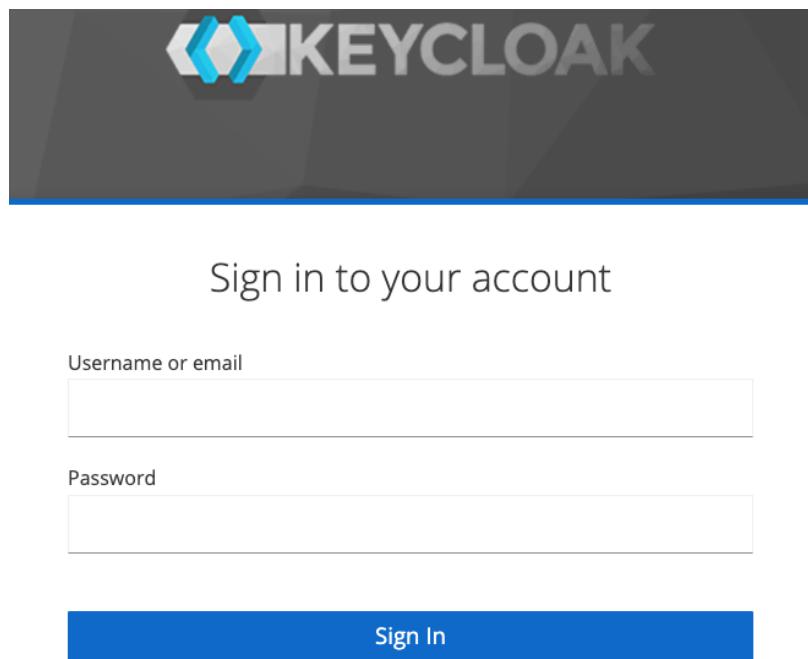


Figura 3.1: Usuario administrador

#### 2. Creación de *REALM*

Keycloak crea *Realms* o reinados separados y no accesibles entre ellos y administra clientes, usuarios y otras entidades dentro de estos. Cada *realm* tiene su propia configuración.

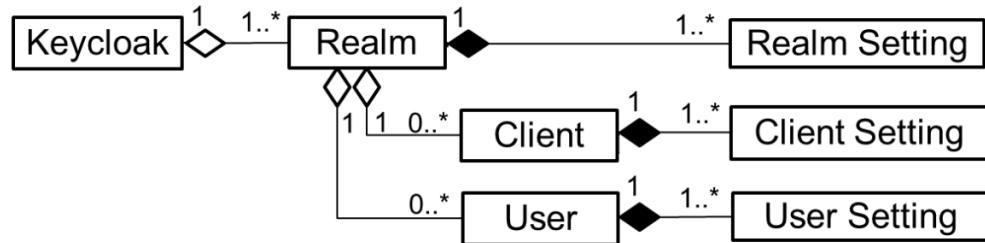


Figura 3.2: Relación de entidades en Keycloak

En este caso se necesita un único sistema para la autenticación de varios clientes, por lo que solo se utilizará un *realm* llamado *Master*.

### 3. Creación de *User Federation*

Keycloak puede almacenar y administrar usuarios. A menudo las instituciones ya tienen los servicios de LDAP o *Active Directory* para guardar los usuarios y las credenciales. Keycloak es capaz de validar credenciales y extraer información de fuentes externas. [35]

La captura de pantalla muestra la interfaz de usuario de Keycloak en la página "User federation". El menú lateral izquierdo muestra las siguientes opciones: master, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, y User federation. La opción "User federation" está resaltada en azul.

En la sección central, el título es "User federation" y se indica que "User federation provides access to external databases and directories, such as LDAP and Active Directory.". Hay dos botones: "Add new provider" y "Manage priorities". Una lista muestra un proveedor llamado "Ldap" que está "Enabled".

Figura 3.3: Keycloak: *User Federation*

En este caso, la información se extraerá de LDAP. Por lo tanto, la configuración será la siguiente:

### LDAP

Settings    Mappers

**General options**

Console display name \* ⓘ Ildap

Vendor \* ⓘ Other

**Connection and authentication settings**

Connection URL \* ⓘ ldap://10.6.240.133:389

Enable StartTLS ⓘ  Off

Use Truststore SPI ⓘ Only for ldaps

Connection pooling ⓘ  On

Connection timeout ⓘ

**Test connection**

Bind type \* ⓘ simple

Bind DN \* ⓘ cn=admin,dc=uh,dc=cu

Bind credentials \* ⓘ .....  
**Test authentication**

Figura 3.4: Configuración de LDAP en Keycloak(1)

**LDAP searching and updating**

Edit mode *	<input type="text" value="READ_ONLY"/>
Users DN *	<input type="text" value="ou=estudiante,dc=uh,dc=cu"/>
Username LDAP attribute *	<input type="text" value="uid"/>
RDN LDAP attribute *	<input type="text" value="uid"/>
UUID LDAP attribute *	<input type="text" value="entryUUID"/>
User object classes *	<input type="text" value="top"/>
User LDAP filter	<input type="text"/>
Search scope	<input type="text" value="One Level"/>
Read timeout	<input type="text"/>
Pagination	<input checked="" type="checkbox"/> Off

**Synchronization settings**

Import users	<input checked="" type="checkbox"/> On
Sync Registrations	<input checked="" type="checkbox"/> On
Batch size	<input type="text"/>
Periodic full sync	<input checked="" type="checkbox"/> Off

Figura 3.5: Configuración de LDAP en Keycloak (2)

Luego de conectar Keycloak con LDAP, se puede comprobar que los usuarios sincronizan. En la siguiente imagen se pueden ver los usuarios en LDAP desde Apache Directory Studio:

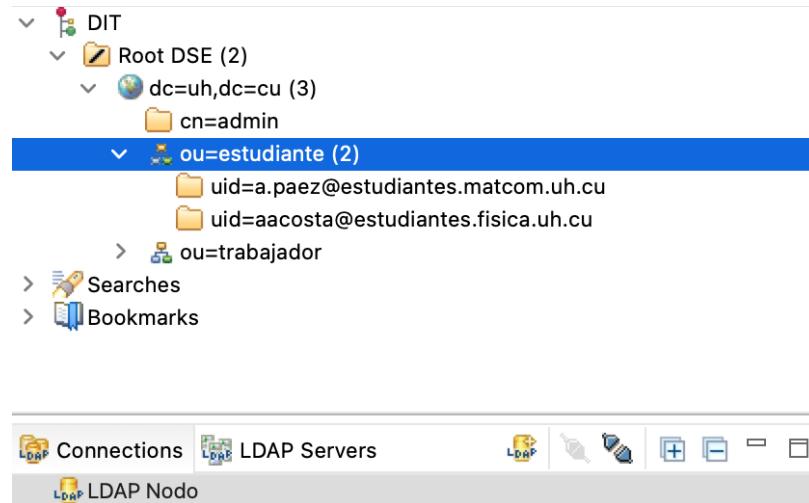


Figura 3.6: Usuarios en LDAP vistos desde Apache Directory Studio

En la siguiente imagen se puede comprobar que Keycloak sincroniza todos los usuarios de LDAP:

Username	Email	Last name	First name
a.paez@estudiantes.matcom.uh.cu	Paez Ruiz	Paez Ruiz	Adrian Tubal
aacosta@estudiantes.fisica.uh.cu	Acosta Martinez	Acosta Martinez	Alberto
admin	-	-	-

Figura 3.7: Lista de usuarios

Los usuarios sincronizan correctamente el *username*, los apellidos y los nombres. Sin embargo, el correo no sincronizó correctamente de forma automática. Para ello Keycloak brinda la opción de mapear atributos de LDAP a los de Keycloak.

User federation > Settings

## LDAP

Name	Type
creation date	user-attribute-ldap-mapper
email	user-attribute-ldap-mapper
first name	user-attribute-ldap-mapper
last name	user-attribute-ldap-mapper
modify date	user-attribute-ldap-mapper
username	user-attribute-ldap-mapper

Figura 3.8: Configuración para mapear

User federation > Settings > Mapper details

### email

---

ID	9fdf636b-9812-4e92-bec4-8b0ef72a1213
Name *	email
Mapper type *	user-attribute-ldap-mapper
User Model Attribute ?	email
LDAP Attribute	uid
Read Only	<input checked="" type="checkbox"/> On
Always Read Value From LDAP	<input checked="" type="checkbox"/> On
Is Mandatory In LDAP ?	<input type="checkbox"/> Off
Attribute default value ?	
Force a Default Value ?	<input checked="" type="checkbox"/> On
Is Binary Attribute	<input type="checkbox"/> Off

Figura 3.9: Configuración para mapear email

**Users**  
Users are the users in the current realm. [Learn more](#)

---

User list		Permissions	
<input type="text"/> Search user	<input type="button" value="→"/>	<input type="button" value="Add user"/>	
<input type="button" value="Delete user"/>			
<input type="checkbox"/> Username	Email	Last name	First name
<input type="checkbox"/> a.paez@estudiantes.matcom.uh.cu	<a href="#">a.paez@estudiantes.matcom.uh.cu</a>	Paez Ruiz	Adrian Tubal
<input type="checkbox"/> aacosta@estudiantes.fisica.uh.cu	<a href="#">aacosta@estudiantes.fisica.uh.cu</a>	Acosta Martinez	Alberto
<input type="checkbox"/> admin	<a href="#">–</a>	–	–

Figura 3.10: Usuarios con email mapeado

### 3.3. Capa de autenticación

Para la autenticación se necesita un nombre de usuario único para cada trabajador y estudiante de la institución. Al utilizarse distintas fuentes de datos, no se puede garantizar que todas las bases de datos tengan las mismas estructuras. Sin embargo, todos los usuarios tienen una cuenta de correo que los identifica únicamente, por lo que utilizar este campo como nombre de usuario garantiza que cada persona es identificada.

En el siguiente diagrama se muestra el flujo de información que se realiza para el correcto funcionamiento de la autenticación:

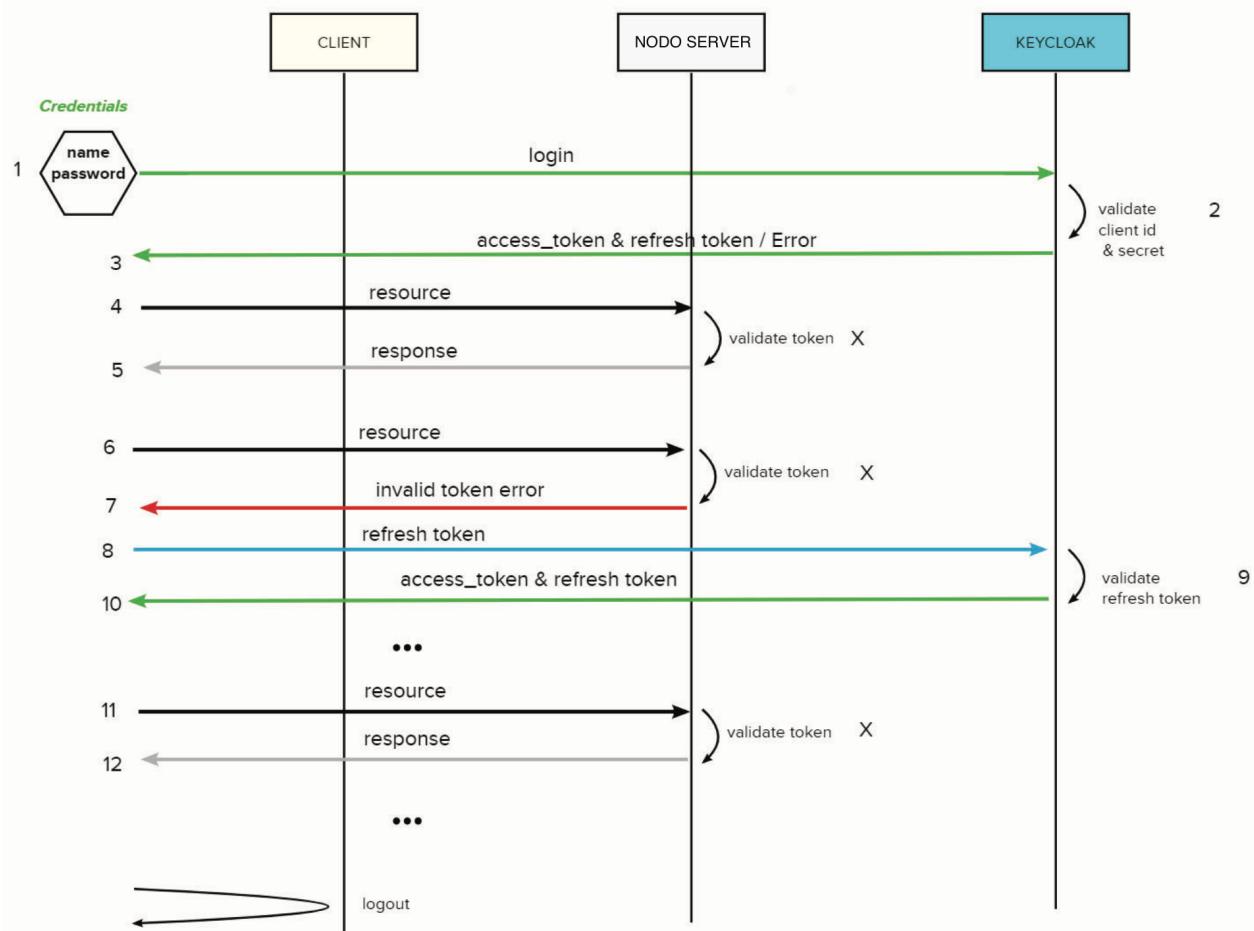


Figura 3.11: Diagrama de flujo de información

Cada paso descrito en el diagrama de flujo se explica detalladamente a continuación:

- La primera acción (1) consiste en que el cliente haga login en el sistema por medio de unas credenciales básicas que son su identificador y contraseña. Esta petición se redirigirá automáticamente al servidor Keycloak, el único que contiene la información almacenada de los clientes del sistema, así como sus credenciales.
- El segundo paso (2), consiste en la validación de dichas credenciales en la aplicación de autenticación Keycloak, y la generación de un *Access Token*.
- El tercer paso (3) es el envío de dicho *token* a través de JWT hacia el usuario o aplicación cliente. Sin embargo, en caso de que las credenciales hayan resultado erróneas, se enviará el mensaje de error correspondiente.
- En el cuarto paso (4), la aplicación cliente ya iniciada sesión, dispone del *Access Token* para realizar consultas, por lo que genera la primera petición de algún recurso cualquiera al servidor del Nodo.
- El paso denominado X es la validación instantánea del *token* que contiene la petición del cliente. Antes de entrar en la lógica de interfaz que ofrece la API del servidor, cabe mencionar que aquí se ha insertado un denominado “Interceptor”. Este, procesa el JWT que le llega en la petición al servidor y valida toda la información del mismo. Este paso se repite en varias ocasiones, para cada petición de recurso ya que cada recurso requiere unos permisos o ámbitos específicos.
- El quinto (5) paso consiste en la generación de la respuesta que envía el servidor a la aplicación cliente en caso de validación de token correcta.
- El sexto paso (6) esta vez corresponde a otra petición de recurso distinta, que será seguida de una errónea validación de token en el paso X.
- El séptimo paso (7) corresponde al envío del mensaje de error a la aplicación cliente, que será tratado como expiración del token, y le pedirá una extensión o actualización de la sesión.
- El octavo paso (8) corresponde al consecuente envío del *refresh token*, que sirve para extender la sesión actual. Es el mecanismo que se lleva a cabo a bajo nivel en cualquier sesión abierta de un servicio web.
- En el noveno paso (9) se produce una validación de este *refresh token*, y la obtención de un nuevo *Access Token* que se obtiene de forma similar pero no igual al mencionado en el segundo paso (2).

- En el décimo paso (10) se envía este último token generado a la aplicación cliente de manera que este podrá realizar nuevas peticiones.
- En los pasos undécimo (11) y duodécimo (12) el escenario se repite de forma iterativa hasta que el cliente decidiese acabar en el último paso:
- El paso N consiste en el cierre de sesión por parte del cliente y por lo tanto la comunicación se cierra en este ciclo.

# Capítulo 4

## Experimentación y resultados

En este capítulo se procede al cumplimiento de los objetivos principales de este trabajo. Se mostrará cómo crear y configurar un cliente en Keycloak y por último, se procede a la experimentación del sistema creado a través de un cliente realizado en *Python*.

### 4.1. Configuración de cliente Keycloak

Para la siguiente experimentación se ha utilizado una base de datos del Nodo Central, a la cual se accede de forma remota a través de una VPN y accediendo a una IP. La máquina en la cual se hospeda dicha base de datos tiene 1 GB de RAM y 2 CPUs lógicos virtualizados. La computadora en la que se experimentó la solución tiene 8 GB de RAM y 2.7 GHz Dual-Core Intel Core i5 de CPU.

Luego de configurado LDAP y sincronizados los usuarios, Keycloak permite añadir clientes de forma sencilla. En las siguientes imágenes se muestra paso a paso cómo se agrega un nuevo cliente al sistema:

Clients > Create client

### Create client

Clients are applications and services that can request authentication of a user.

---

1 General Settings

Client type ⓘ	OpenID Connect
Client ID * ⓘ	nodo
Name ⓘ	nodo
Description ⓘ	Cliente para la autenticación de los usuarios de la UH
Always display in console ⓘ	<input checked="" type="checkbox"/> Off

---

Figura 4.1: Nuevo cliente en Keycloak(1)

Clients > Create client

### Create client

Clients are applications and services that can request authentication of a user.

---

1 General Settings

2 Capability config

Client authentication ⓘ	<input checked="" type="checkbox"/> On	
Authorization ⓘ	<input checked="" type="checkbox"/> On	
Authentication flow	<input checked="" type="checkbox"/> Standard flow ⓘ	<input checked="" type="checkbox"/> Direct access grants ⓘ
	<input type="checkbox"/> Implicit flow ⓘ	<input checked="" type="checkbox"/> Service accounts roles ⓘ
	<input type="checkbox"/> OAuth 2.0 Device Authorization Grant ⓘ	
	<input type="checkbox"/> OIDC CIBA Grant ⓘ	

---

Figura 4.2: Nuevo cliente en Keycloak (2)

Luego se puede ver el nuevo cliente nodo en la lista de clientes:

The screenshot shows the Keycloak administration interface. The left sidebar is titled 'Manage' and includes options like 'Clients', which is currently selected and highlighted in blue. Other options include 'Client scopes', 'Realm roles', 'Users', 'Groups', 'Sessions', 'Events', 'Configure', 'Realm settings', 'Authentication', 'Identity providers', and 'User federation'. The main content area is titled 'Clients' and contains the following text: 'Clients are applications and services that can request authentication of a user.' Below this are two tabs: 'Clients list' (which is active) and 'Initial access token'. A search bar labeled 'Search for client' is followed by a 'Create client' button and an 'Import client' link. A table lists the clients:

Client ID	Type
account	OpenID Connect
account-console	OpenID Connect
admin-cli	OpenID Connect
broker	OpenID Connect
master-realm	OpenID Connect
nodo	OpenID Connect
security-admin-console	OpenID Connect

Figura 4.3: Lista de clientes

Luego de crear el cliente, este se puede conectar a los servicios de Keycloak con la correcta configuración. Para ello será necesario utilizar las credenciales del cliente:

Clients > Client details

**nodo** OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings    Keys    Credentials    Roles    Client scopes    Sessions    Advanced

**General Settings**

Client ID \* ⓘ nodo

Name ⓘ nodo

Description ⓘ

Always display in console ⓘ  Off

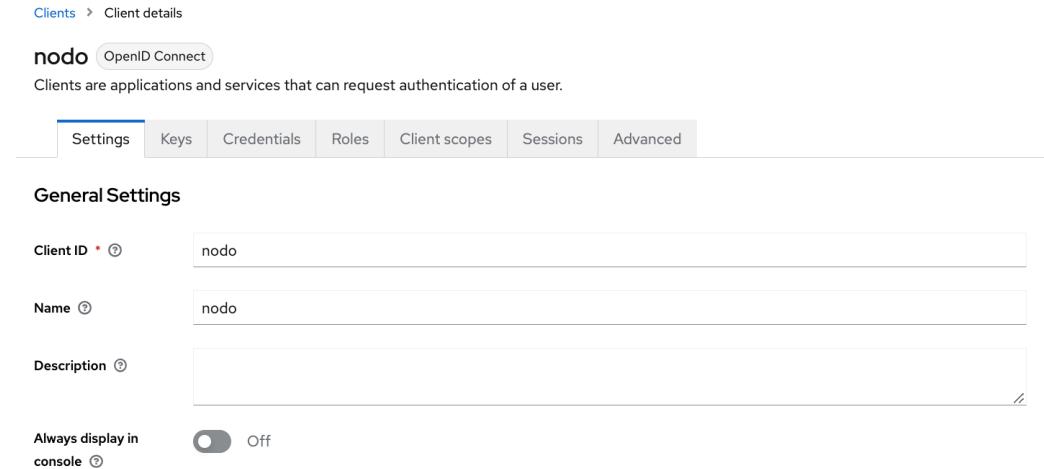


Figura 4.4: Cliente nodo

Clients > Client details

**nodo** OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings    Keys    **Credentials**    Roles    Client scopes    Sessions    Advanced

**Client Authenticator** Client Id and Secret

Save

**Client secret** .....

**Registration access token**

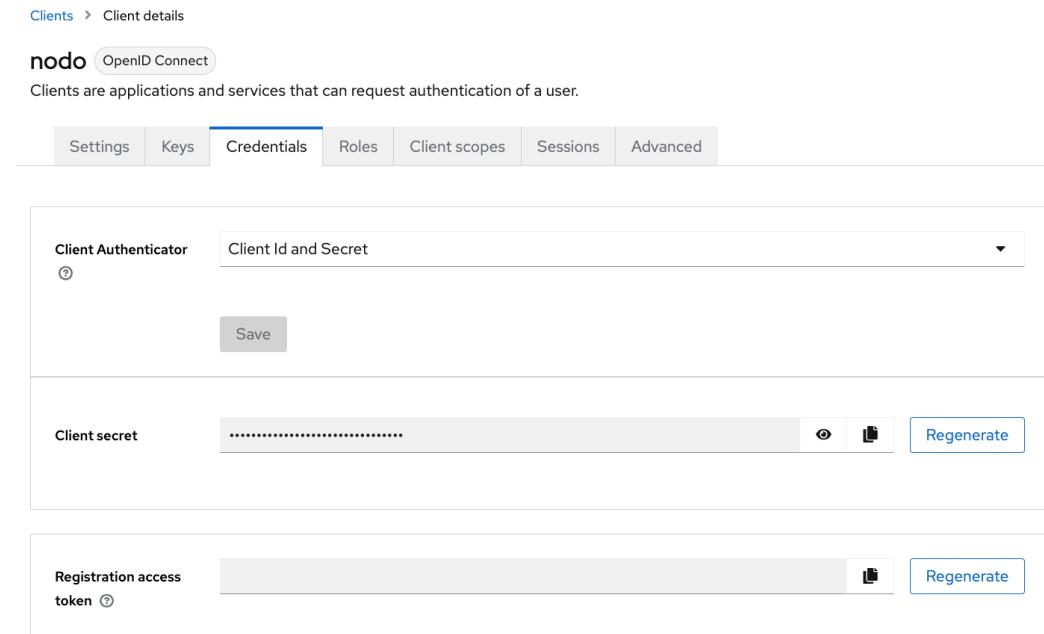


Figura 4.5: Credenciales del cliente nodo

## 4.2. Creación de cliente Keycloak para obtención de *tokens*

En el siguiente código se muestra cómo se puede conectar el cliente a Keycloak a través de la biblioteca **python-keycloak**. También se crea una interfaz básica con Flask para visualizar los resultados.

```
1 from flask import Flask
2 from flask import Flask, render_template, request
3 from keycloak.keycloak_openid import KeycloakOpenID
4 from keycloak.exceptions import KeycloakAuthenticationError,
5     KeycloakGetError
6
7
8 app = Flask(__name__)
9
10 keycloak_open_id = KeycloakOpenID(server_url="http://localhost:8080/",
11     client_id="nodo",
12     realm_name="master",
13     client_secret_key="secret key")
14 keycloak_open_id.well_know()
15
16 @app.route('/login', methods=['POST', 'GET'])
17 def login():
18     error = None
19     if request.method == 'POST':
20         username = request.form['username']
21         success, result = valid_login(username, request.form['password'])
22         if success:
23             return log_the_user_in(username, result["access_token"],
24             result["refresh_token"])
25         else:
26             error = result["error_description"]
27             # The code below is executed if the request method
28             # was GET or the credentials were invalid
29             return render_template('login.html', error=error)
30
31 def valid_login(username, password):
32     # import pdb
33     # pdb.set_trace()
34     try:
35         token = keycloak_open_id.token(username, password)
36     except (KeycloakAuthenticationError, KeycloakGetError) as e:
37         return False, json.loads(e.error_message)
38     return True, token
```

```
38  
39 def log_the_user_in(username, token, refresh_token):  
40     return render_template('success.html', username=username, token=  
        token, refresh_token=refresh_token)
```

Ejemplo de código 4.1: Conexión de cliente a Keycloak

El código anterior genera la siguiente interfaz:

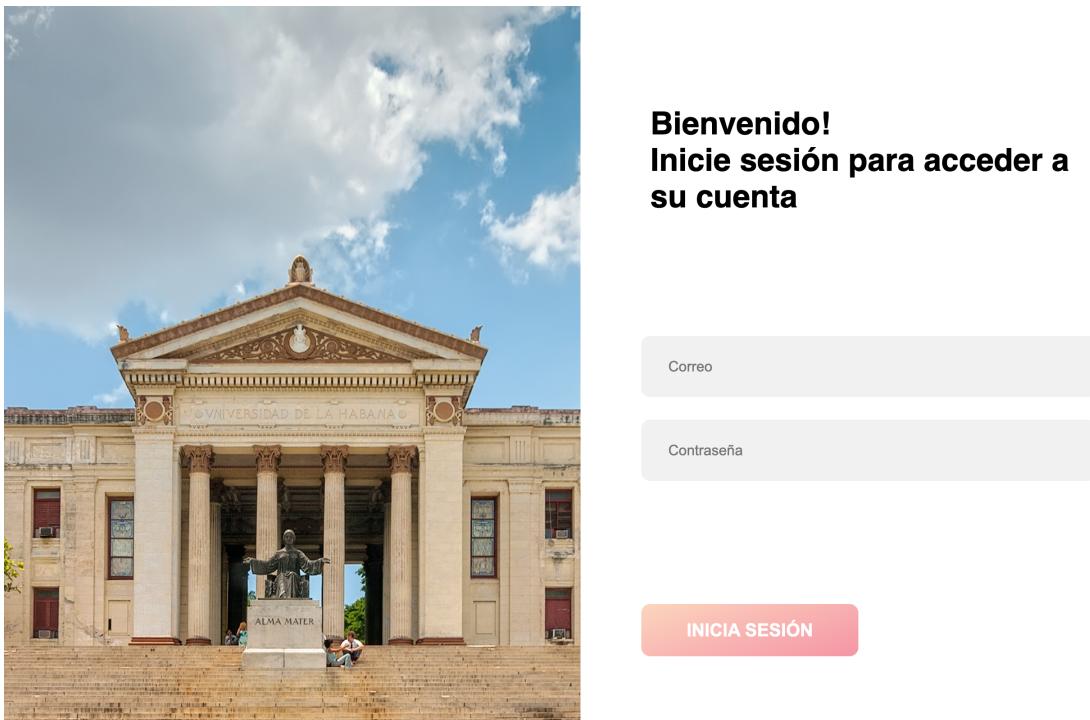


Figura 4.6: Interfaz visual

Luego de introducir un correo y su correspondiente contraseña se obtiene el *token* de autenticación y el *refresh token*:

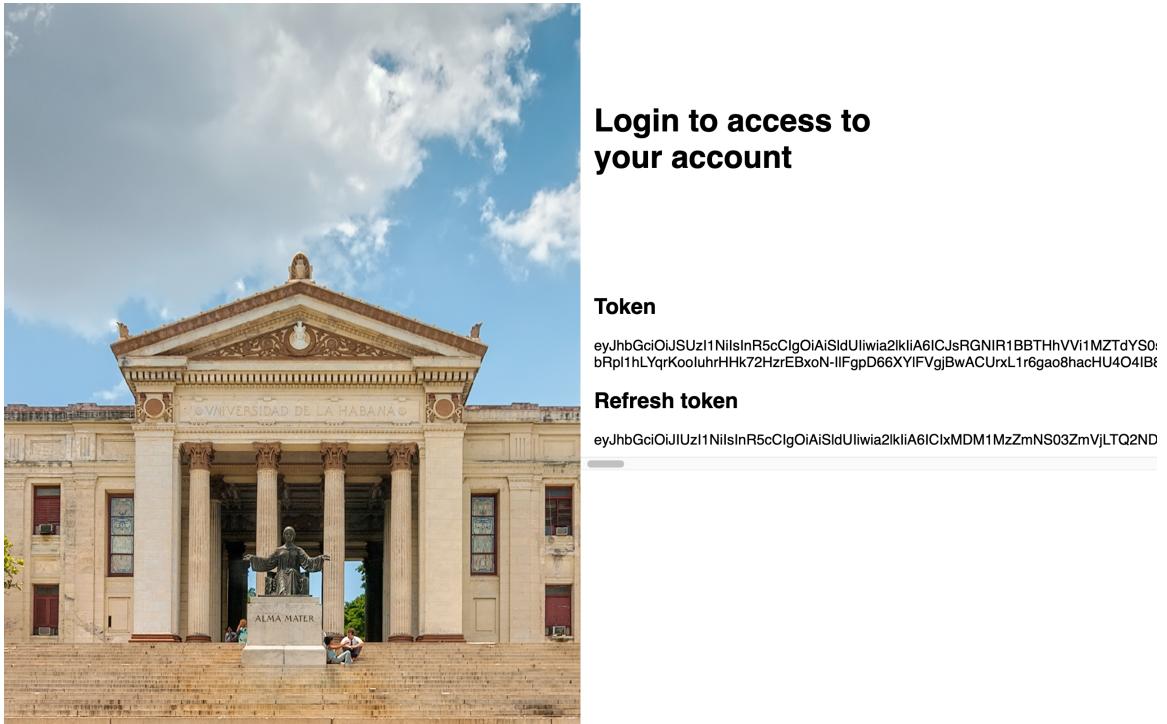


Figura 4.7: Resultado de autenticación exitosa

### 4.3. Evaluación

Debido al alto tráfico de información al que será sometido el sistema de autenticación, se considera necesario evaluar su comportamiento. Considerando que en la universidad se cuenta con 38 678 estudiantes y 600 docentes, se asume un estimado de 50 000 usuarios en total. Para realizar las pruebas se utilizará Locust, una herramienta de código abierto para pruebas de carga. [36]

Locust es un software desarrollado en Python para la evaluación del comportamiento de aplicaciones web con múltiples usuarios concurrentes. Es capaz de generar tráfico que se conecta a la aplicación web con el objetivo de probar el comportamiento con una gran carga. Es una herramienta que define el comportamiento y la carga del tráfico virtual en una aplicación o en un servidor web. [37]

Para evaluar el comportamiento de un sitio web con una concurrencia de 1000 usuarios, son necesarios 1000 usuarios. En la práctica es difícil disponer de 1000 personas para que abran el sitio a la vez y comprueben si este está funcionando. Locust es capaz de generar el tráfico de usuario y los resultados se representan en múltiples parámetros, incluyendo número de pedidos, tamaño de los contenidos, cantidad de intentos fallidos, entre otros.

En el siguiente código se muestra cómo conectar Locust con el sistema de autenticación creado:

```
1 from locust import HttpUser, task
2 from random import choice
3
4 URL_TOKEN = "realms/{realm-name}/protocol/openid-connect/token"
5
6 class KeycloakUser(HttpUser):
7     def on_start(self):
8         self.users = [("user", "password")]
9         self.realm_name = "realm"
10        self.client_id = "client"
11        self.secret_key = "secret key"
12
13    @task
14    def token_load_test(self):
15        username, password = choice(self.users)
16        params_path = {"realm-name": self.realm_name}
17        payload = {
18            "username": username,
19            "password": password,
20            "client_id": self.client_id,
21            "grant_type": ["password"],
22            "code": "",
23            "redirect_uri": "",
24            "client_secret": self.secret_key
25        }
26        self.client.post(URL_TOKEN.format(**params_path), data=payload)
```

Ejemplo de código 4.2: Conexión de Locust a Keycloak

El código anterior genera la siguiente interfaz visual:

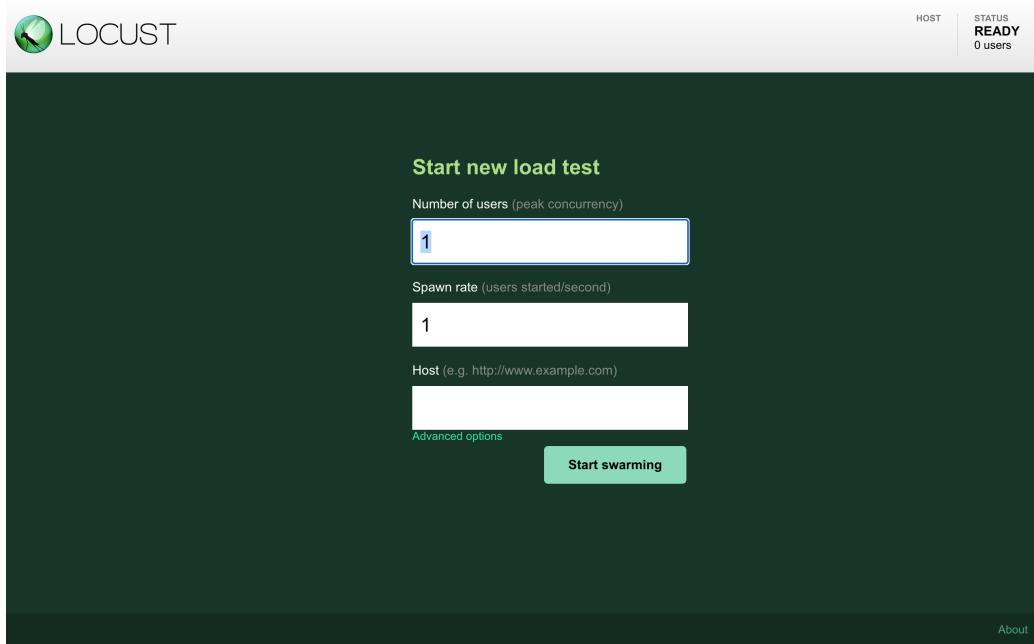


Figura 4.8: Interfaz visual de Locust

Con la siguiente configuración:

A screenshot of the Locust configuration interface. It shows the same 'Start new load test' form as Figura 4.8, but with different values: 'Number of users (peak concurrency)' is set to '100', 'Spawn rate (users started/second)' is set to '10', and 'Host (e.g. http://www.example.com)' is set to 'http://localhost:8080/'. There is also a 'Run time (e.g. 20, 20s, 3m, 2h, 1h20m, 3h30m10s, etc.)' field containing '30s'. The 'Start swarming' button is visible at the bottom.

Figura 4.9: Configuración de Locust

Se obtienen los siguientes resultados:

Request Statistics									
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
POST	/realms/master/protocol/openid-connect/token	4847	0	417	33	2008	2124	160.3	0.0
	<b>Aggregated</b>	<b>4847</b>	<b>0</b>	<b>417</b>	<b>33</b>	<b>2008</b>	<b>2124</b>	<b>160.3</b>	<b>0.0</b>
Response Time Statistics									
Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
POST	/realms/master/protocol/openid-connect/token	420	460	510	550	590	620	910	2000
	<b>Aggregated</b>	<b>420</b>	<b>460</b>	<b>510</b>	<b>550</b>	<b>590</b>	<b>620</b>	<b>910</b>	<b>2000</b>

Figura 4.10: Estadísticas obtenidas de prueba de Locust



Figura 4.11: Total de pedidos por segundo

En el gráfico anterior se muestra la cantidad de pedidos respondidos y fallidos por segundo. Se puede apreciar que el sistema es capaz de responder hasta 170 pedidos por segundo sin respuestas fallidas.

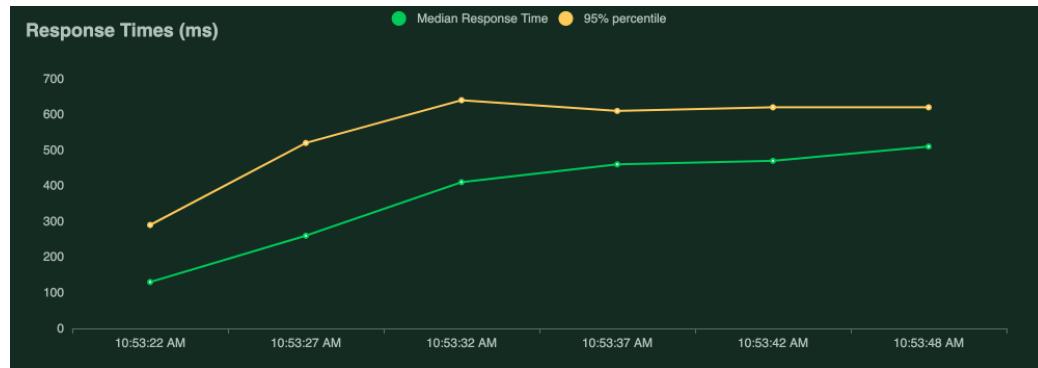


Figura 4.12: Tiempo de respuesta

El gráfico muestra la media del tiempo de respuesta de los pedidos. Se percibe que a medida que aumentan los usuarios el tiempo de respuesta aumenta, sin embargo, este se mantiene por debajo de un segundo.

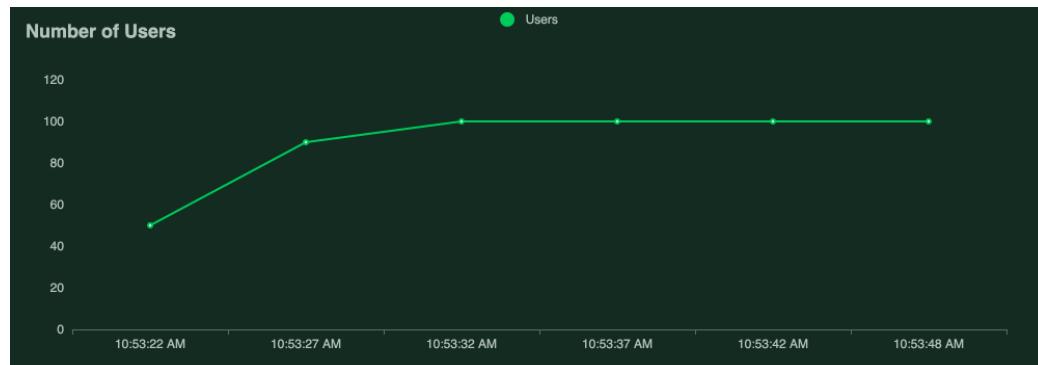


Figura 4.13: Número de usuarios

Este gráfico simula el aumento de los usuarios a lo largo del tiempo.

De la prueba de carga llevada a cabo, se puede concluir que, si el sistema es capaz de responder más de 100 pedidos por segundo, es capaz de responder al menos 360 000 pedidos por hora. Teniendo en cuenta que los *tokens* de autenticación tienen una duración y en la universidad hay un estimado de 50 000 mil usuarios, se puede asumir que el sistema es capaz de interactuar con todos los usuarios de la institución accediendo a sus cuentas por 7 clientes distintos a lo largo de una hora.

tráfico de la Universidad de La Habana.

# Conclusiones

Durante el desarrollo de este trabajo se realizó un estudio del estado del arte acerca de Keycloak y las facilidades que brinda. Además, se estudiaron las mejores soluciones para la autorización y autenticación de usuarios.

Se analizó el sistema que se utiliza actualmente en el Nodo Central para el almacenamiento de los usuarios y la autenticación de los mismos. Además, se procedió al diseño de un sistema más eficiente y con mejor experiencia de usuario. Esto contribuyó al cumplimiento del objetivo principal de este trabajo, la propuesta de una metodología que permite la autenticación de usuarios desde distintas aplicaciones de la institución de una forma eficiente y repetible.

Como parte de la metodología se utilizan las herramientas LDAP para la administración de los datos de los usuarios y Keycloak para la autenticación y autorización. Se emplea el método Single Sign-On, la autenticación basada en *tokens* y el protocolo OpenID Connect para garantizar un inicio de sesión sencillo y seguro.

El sistema resultante es un mecanismo de autenticación que mejora respecto al que se encuentra en uso actualmente mediante el uso de herramientas más modernas y seguras. Dicho mecanismo permite a todos los clientes del Nodo Central autenticar y autorizar de forma fiable a sus usuarios mediante unas pocas líneas de código.

# Recomendaciones

Esta tesis se concentró en la capa de autenticación y solucionó los problemas que existían en esta área. Sin embargo, para un mejor funcionamiento del sistema completo, podría ser mejorado el mecanismo utilizado para el almacenamiento de los usuarios. El sistema de autenticación central trabaja sobre un sistema implementado en LDAP que es el encargado del manejo de las diversas bases de datos que existen en la Universidad de La Habana. Este tiene problemas estructurales y funcionales, por lo que es propenso a errores.

Como solución a este problema se propone la utilización de Apache Kafka, un sistema de almacenamiento publicador/subscriptor distribuido, particionado y replicado [38]. Esta es una herramienta probada y utilizada por grandes empresas como *Netflix*, *Pintrest*, *Adidas* y *aribnb*, capaz de actualizar la información de las bases de datos en tiempo real. [39]

Por otra parte, Keycloak brinda facilidades para crear roles y la administración de permisos de los usuarios. Se propone indagar sobre este tema para que el sistema propuesto, además de autenticar, sea capaz de dar acceso a recursos más específicos según el role del usuario. Por ejemplo, dar un rol a los estudiantes distinto al de los trabajadores de forma automática.

También se sugiere crear un manual de usuario para los administradores de los servicios clientes que utilizarán la implementación presentada. Este documento facilitaría la migración al nuevo servicio y ayudaría a acelerar el proceso. Además, sería una fuente permanente de información sobre el trabajo a ejecutar, lo cual aseguraría la continuidad y coherencia de los procedimientos a través del tiempo y facilitaría el cambio de personal en el Nodo Central.

# Bibliografía

- [1] R. Kutera y W. Gryncewicz, «Single sign on as an effective way of managing user identity in distributed web systems. The ActGo-Gate project case study,» Informatyka Ekonomiczna, n.<sup>o</sup> 2 (40), 2016 (vid. págs. 1, 7).
- [2] P. Deitel, H. Deitel y A. Deitel, Cómo programar Internet & World Wide Web. Pearson Educación, 2014 (vid. pág. 4).
- [3] X. P. Ye, «Diseño e implantación de un sistema de autenticación Cross-platform para React y React Native,» 2022 (vid. pág. 4).
- [4] Auth0, Access Tokens, <https://auth0.com/docs/secure/tokens/access-tokens>, 2022 (vid. pág. 5).
- [5] A. Banerjee y M. Hasan, «Token-Based Authentication Techniques on Open Source Cloud Platforms,» Sistemas y Telemática, vol. 16, n.<sup>o</sup> 47, 2018 (vid. pág. 5).
- [6] S. Cantor, I. J. Kemp, N. R. Philpott y E. Maler, «Assertions and protocols for the oasis security assertion markup language,» OASIS Standard (March 2005), págs. 1-86, 2005 (vid. pág. 5).
- [7] D. Recordon y D. Reed, «OpenID 2.0: a platform for user-centric identity management,» en Proceedings of the second ACM workshop on Digital identity management, 2006, págs. 11-16 (vid. pág. 5).
- [8] C. Mainka, V. Mladenov, J. Schwenk y T. Wich, «Sok: Single sign-on security—an evaluation of openid connect,» en 2017 IEEE European Symposium on Security and Privacy, IEEE, 2017, págs. 251-266 (vid. págs. 5, 7).
- [9] D. Hardt, «The OAuth 2.0 authorization framework,» inf. téc., 2012 (vid. pág. 5).
- [10] J. Richer y A. Sanso, OAuth 2 in action. Simon y Schuster, 2017 (vid. pág. 5).

- [11] Microsoft. (2022). «What is single sign-on in Azure Active Directory?» Dirección: <https://learn.microsoft.com/es-es/azure/active-directory/manage-apps/what-is-single-sign-on> (vid. pág. 6).
- [12] V. Radha y D. H. Reddy, «A survey on single sign-on techniques,» Procedia Technology, vol. 4, págs. 134-139, 2012 (vid. pág. 7).
- [13] Google. (2022). «About SSO,» dirección: <https://support.google.com/> (vid. pág. 7).
- [14] Auth0. (2022). «Single Sign-On,» dirección: <https://auth0.com/docs/authenticate/single-sign-on> (vid. pág. 7).
- [15] OpenID. (2022). «Welcome to OpenID Connect,» dirección: <https://openid.net/connect/> (vid. pág. 7).
- [16] R. Sánchez Guerrero, «Estudio y puesta en marcha de una infraestructura de gestión de identidad federada basada en SAML 2.0,» Tesis de mtría., 2009 (vid. pág. 8).
- [17] N. Naik y P. Jenkins, «Securing digital identities in the cloud by selecting an apposite Federated Identity Management from SAML, OAuth and OpenID Connect,» en 2017 11th RCIS, IEEE, 2017, págs. 163-174 (vid. pág. 8).
- [18] LDAP. (2022). «LDAP,» dirección: <https://ldap.com/> (vid. pág. 9).
- [19] Microsoft. (2022). «Active Directory Domain Services Overview,» dirección: <https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview> (vid. pág. 10).
- [20] Keycloak Documentation, <https://www.keycloak.org/documentation> (vid. págs. 12, 13).
- [21] J. Bradley, N. Sakimura y M. B. Jones, JSON Web Token (JWT), 2015. dirección: <https://www.rfc-editor.org/rfc/rfc7519.txt> (vid. pág. 13).
- [22] D. Lobato Navacerrada, «Regulación de acceso a una aplicación médica mediante Keycloak,» 2022 (vid. págs. 15, 25).
- [23] C. Muyón y F. Montaluisa, «Métodos de seguridad de la información para proteger la comunicación y los datos de servicios web REST en peticiones HTTP utilizando JSON Web Token y Keycloak Red Hat Single Sign On,» Revista Ibérica de Sistemas e Tecnologias de Informação, n.º E29, págs. 198-213, 2020 (vid. págs. 15, 16, 24).
- [24] C. Fernández Pérez, «Aplicación web para la gestión de procesos de ingestión de datos en entorno de BI/Big Data,» 2020 (vid. pág. 20).

- [25] I. S. de Diseño. (2022). «Historia del Instituto,» dirección: <https://www.isdi.co.cu/index.php/site/historia/Historia>. (vid. pág. 20).
- [26] J. L. Teherán Sierra y col., «Mecanismo de autenticación y control de acceso para Software-Defined Networking-SDN,» 2014 (vid. pág. 20).
- [27] O. Rodriguez Valdés, C. M. Legón y R. Socorro Llanes, «Seguridad y usabilidad de los esquemas y técnicas de autenticación gráfica,» Revista Cubana de Ciencias Informáticas, vol. 12, págs. 13-27, 2018 (vid. pág. 21).
- [28] O. Lizama, G. Kindley, J. J. Morales y A. Gonzales, «Redes de Computadores: Arquitectura Cliente-Servidor,» Universidad Tecnica Federico Santa Maria, págs. 1-8, 2016 (vid. pág. 21).
- [29] G. Mok Rodríguez, E. Carmona Fernández, C. A. García Santoya y M. Embarek El-Bah Valdés, Evea: Una puerta hacia otra forma de Estudio, 2022. dirección: <http://www.cubadebate.cu/especiales/2022/04/02/evea-una-puerta-hacia-otra-forma-de-estudio/> (vid. pág. 22).
- [30] R. Ferrer, «Pandemia por COVID-19: el mayor reto de la historia del intensivismo,» Medicina intensiva, vol. 44, n.º 6, pág. 323, 2020 (vid. pág. 23).
- [31] G. Vassallo, S. Chiusano y D. Preuveneers, «Continuous authentication with biometrics on smartphones,» 2017 (vid. pág. 25).
- [32] Securing Applications and Services Guide, 2022. dirección: [https://www.keycloak.org/docs/latest/securing\\_apps/](https://www.keycloak.org/docs/latest/securing_apps/) (vid. pág. 25).
- [33] Welcome to Python.org, 2022. dirección: <https://www.python.org/> (vid. pág. 25).
- [34] S. González Díaz, «Implementación de un sistema unificado de autenticación de usuarios aplicado a los diferentes servicios de la Universidad Tecnológica de Bolívar,» 2010 (vid. pág. 25).
- [35] keycloak, 2022. dirección: [https://www.keycloak.org/docs/latest/server\\_admin/](https://www.keycloak.org/docs/latest/server_admin/) (vid. pág. 28).
- [36] T. F. Düllmann, R. Heinrich, A. van Hoorn, T. Pitakrat, J. Walter y F. Willnecker, «CASPA: A platform for comparability of architecture-based software performance engineering approaches,» en 2017 IEEE International Conference on Software Architecture Workshops, IEEE, 2017, págs. 294-297 (vid. pág. 43).
- [37] S. Pradeep e Y. K. Sharma, «A pragmatic evaluation of stress and performance testing technologies for web based applications,» en 2019 Amity International Conference on Artificial Intelligence (AICAI), IEEE, 2019, págs. 399-403 (vid. pág. 43).

- [38] P. Gallegos Jiménez, «Aplicación de gestión de entrenamiento con notificaciones de eventos mediante cola kafka,» 2015 (vid. pág. 49).
- [39] T. A. S. Foundation, Apache Kafka — [kafka.apache.org](https://kafka.apache.org), <https://kafka.apache.org/power>ed-by, 2022 (vid. pág. 49).