**Departamento de Ciencias de la Computación (DCCO)**

**Carrera de Ingeniería de Software**

**Curso de Aplicaciones Distribuidas**

Diseño de Seguridad con OAuth 2.0

Presentado por: Marlyn Almeida, Sebastian Bolaños, Nicole Lara, Axel Pullaguari

Tutor: Morales, Dario.

Ciudad: Sangolquí, Ecuador

Fecha: 01/02/2025

Contenido

[**1.** **Introducción** 3](#_Toc192427894)

[**2.** **Objetivo** 3](#_Toc192427895)

[**3.** **Configuración de Seguridad con OAuth 2.0** 3](#_Toc192427896)

[**3.1.** **Configuración del Servidor de Autorización** 3](#_Toc192427897)

[**3.2.** **Protección de Endpoints** 4](#_Toc192427898)

[**4.** **Guía de Integración de OAuth 2.0** 4](#_Toc192427899)

[**4.1.** **Flujo de Autenticación y Autorización** 4](#_Toc192427900)

[**4.2.** **Configuración del Cliente OAuth 2.0** 4](#_Toc192427901)

[**4.3.** **Uso de Token en Peticiones** 4](#_Toc192427902)

[**5.** **Conclusión** 5](#_Toc192427903)

**Configuraciones de Seguridad y Guías de Integración de OAuth 2.0**

# **Introducción**

La seguridad es un aspecto crítico en la gestión de acceso dentro de la aplicación de recursos humanos. Para garantizar un acceso seguro y controlado a los microservicios, se ha implementado OAuth 2.0, un estándar de autorización ampliamente utilizado en sistemas distribuidos. Este documento detalla la configuración de seguridad aplicada en el sistema y proporciona una guía detallada para la integración de OAuth 2.0.

# **Objetivo**

Implementar un sistema de autenticación y autorización seguro mediante OAuth 2.0, permitiendo un acceso restringido a los recursos del sistema y garantizando la protección de la información de los empleados y usuarios.

# **Configuración de Seguridad con OAuth 2.0**

Para implementar la seguridad en el sistema, se han configurado dos componentes clave:

* **Servidor de Autorización:** Gestiona la autenticación de los usuarios y la emisión de tokens de acceso.
* **Protección de Endpoints:** Define qué rutas requieren autenticación y cómo se validan los tokens.

## **Configuración del Servidor de Autorización**

|  |
| --- |
| @Configuration  @EnableAuthorizationServer  public class AuthorizationServerConfig extends AuthorizationServerConfigurerAdapter {  @Override  public void configure(ClientDetailsServiceConfigurer clients) throws Exception {  clients.inMemory()  .withClient("client-id")  .secret(new BCryptPasswordEncoder().encode("client-secret"))  .authorizedGrantTypes("authorization\_code", "client\_credentials", "refresh\_token")  .scopes("read", "write")  .accessTokenValiditySeconds(3600)  .refreshTokenValiditySeconds(7200);  }  } |

Este código configura un servidor de autorización en memoria que gestiona clientes y sus permisos. Se definen los tipos de autenticación permitidos (authorization\_code, client\_credentials), junto con la duración de los tokens de acceso y actualización.

## **Protección de Endpoints**

|  |
| --- |
| @Configuration  @EnableResourceServer  public class ResourceServerConfig extends ResourceServerConfigurerAdapter {  @Override  public void configure(HttpSecurity http) throws Exception {  http  .authorizeRequests()  .antMatchers("/api/public/\*\*").permitAll()  .antMatchers("/api/protected/\*\*").authenticated()  .and()  .oauth2ResourceServer().jwt();  }  } |

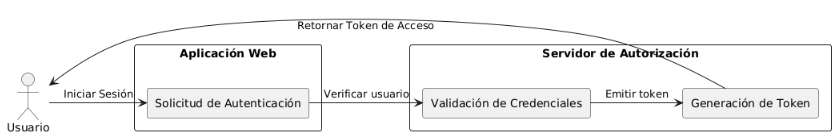
Se define la seguridad de los endpoints, donde las rutas /api/public/\*\* son accesibles sin autenticación, mientras que /api/protected/\*\* requieren un token válido. Se utiliza validación de tokens JWT para la autenticación.

# **Guía de Integración de OAuth 2.0**

Para integrar OAuth 2.0 en el sistema, se deben seguir los siguientes pasos:

## **Flujo de Autenticación y Autorización**

El siguiente diagrama representa el flujo de autenticación y autorización mediante OAuth 2.0:



## **Configuración del Cliente OAuth 2.0**

Los clientes que acceden a los microservicios deben configurarse para usar OAuth 2.0:

|  |
| --- |
| const authConfig = {  clientId: 'client-id',  clientSecret: 'client-secret',  authorizationUrl: 'https://auth.example.com/oauth/authorize',  tokenUrl: 'https://auth.example.com/oauth/token',  scope: 'read write'  }; |

## **Uso de Token en Peticiones**

Para consumir los microservicios protegidos, se debe enviar el token en cada solicitud HTTP:

|  |
| --- |
| const headers = new HttpHeaders().set('Authorization', `Bearer ${accessToken}`);  this.http.get('https://api.example.com/protected/resource', { headers }); |

# **Conclusión**

La implementación de OAuth 2.0 permite un control seguro del acceso a los recursos protegidos dentro del sistema. Se han configurado un Servidor de Autorización y un Recurso Protegido para validar tokens y restringir accesos. Además, se ha detallado la integración con clientes externos para consumir los microservicios de manera segura.