**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](#_Toc192425604)

[**2.** **Objetivos** 3](#_Toc192425605)

[**3.** **Flujos de OAuth 2.0 Implementados** 3](#_Toc192425606)

[**3.1.** **Flujo de Credenciales de Cliente** 3](#_Toc192425607)

[**3.2.** **Flujo de Autorización con Código** 3](#_Toc192425608)

[**4.** **Configuración del Servidor de Autorización** 4](#_Toc192425609)

[**4.1.** **Configuración de Spring Security en el Servidor de Autorización** 4](#_Toc192425610)

[**4.2.** **Configuración del Recurso Protegido** 4](#_Toc192425611)

[**5.** **Conclusión** 5](#_Toc192425612)

**Especificación de Flujos de OAuth 2.0 y Configuración del Servidor de Autorización**

# **Introducción**

OAuth 2.0 es un protocolo de autorización estándar que permite a las aplicaciones obtener acceso seguro a los recursos protegidos sin necesidad de manejar directamente las credenciales de los usuarios. En el sistema de gestión de recursos humanos, OAuth 2.0 se implementa para gestionar el acceso a los microservicios y garantizar que solo los usuarios y servicios autenticados puedan realizar operaciones sensibles. Este documento describe los flujos de autorización utilizados y la configuración del servidor de autorización.

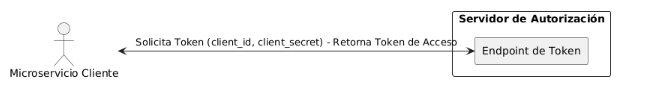
# **Objetivos**

* Implementar OAuth 2.0 para controlar el acceso seguro a los microservicios.
* Definir los flujos de autenticación y autorización según el tipo de usuario y servicio.
* Especificar la configuración del servidor de autorización.
* Garantizar la seguridad y el control de acceso dentro del sistema.

# **Flujos de OAuth 2.0 Implementados**

## **Flujo de Credenciales de Cliente**

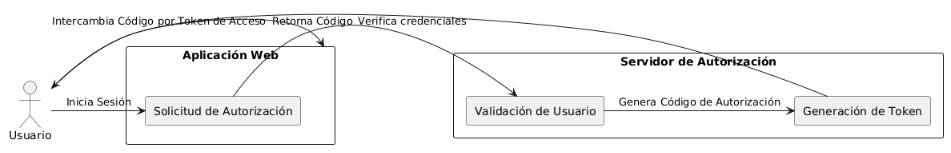
Este flujo se usa cuando un servicio necesita autenticarse con otro servicio sin intervención de un usuario.



* Un microservicio envía una solicitud de token al servidor de autorización.
* El servidor valida las credenciales y responde con un token de acceso.
* El microservicio usa este token para autenticarse en otras API protegidas.

## **Flujo de Autorización con Código**

Este flujo se usa para la autenticación de usuarios en el sistema y garantiza que solo usuarios autorizados accedan a los recursos protegidos.



* El usuario accede a la aplicación y es redirigido al servidor de autorización.
* El servidor valida sus credenciales y emite un código de autorización.
* La aplicación intercambia este código por un token de acceso para realizar solicitudes protegidas.

# **Configuración del Servidor de Autorización**

Para la implementación de OAuth 2.0, se utiliza Spring Security con OAuth2, configurando un servidor de autorización centralizado.

## **Configuración de Spring Security en el 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")  .scopes("read", "write");  }  } |
|  |

Este código configura un servidor de autorización que almacena clientes en memoria. Define los clientes permitidos, los tipos de credenciales soportados y los permisos asociados. Utiliza BCryptPasswordEncoder para almacenar de manera segura la clave secreta del cliente, garantizando protección ante accesos no autorizados.

## **Configuración del Recurso Protegido**

|  |
| --- |
| @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();  }  } |

Esta configuración define qué rutas están protegidas y cuáles son de acceso público. Todas las rutas bajo /api/protected/\*\* requieren autenticación, mientras que las bajo /api/public/\*\* son accesibles sin restricciones. Esto garantiza que solo usuarios autenticados puedan acceder a información sensible dentro del sistema.

# **Conclusión**

La implementación de OAuth 2.0 en este sistema permite un control seguro del acceso a los recursos protegidos, asegurando que solo usuarios y servicios autenticados puedan interactuar con los microservicios. Se han definido dos flujos de autorización clave: el Flujo de Credenciales de Cliente para la comunicación entre microservicios y el Flujo de Autorización con Código para la autenticación de usuarios. La configuración con Spring Security OAuth2 facilita la integración y seguridad del sistema.