

## Especialización en Back End II

# Integrando Keycloak con Spring Cloud para Microservicios de forma embebida mediante Gateway

En este tutorial paso a paso, veremos cómo utilizar Keycloak de forma embebida en un microservicio utilizando un gateway.

## Índice

- [Configuración Previa](#)
- [Creación de Microservicio](#)
- [Configuración de Gateway](#)
- [Pruebas](#)

# Configuración Previa

En Keycloak deberemos configurar los siguientes aspectos.

- Creamos un nuevo reino llamado DH
- Creamos un nuevo cliente **llamado api-gateway-client** y vamos a asignarle como Root URL <http://localhost:9090/>
- En los settings del cliente vamos a llevar a cabo las siguientes configuraciones:

## General Settings

Client ID * ?	<input type="text" value="api-gateway-client"/>
Name ?	<input type="text"/>
Description ?	<input type="text"/>
Always display in UI ?	<input type="checkbox"/> Off

## Access settings

Root URL ?	<input type="text" value="http://localhost:9090"/>
Home URL ?	<input type="text"/>
Valid redirect URIs ?	<input type="text" value="http://localhost:9090/*"/> <a href="#">+ Add valid redirect URIs</a>
Valid post logout redirect URIs ?	<input type="text"/> <a href="#">+ Add valid post logout redirect URIs</a>
Web origins ?	<input type="text" value="http://localhost:9090"/> <a href="#">+ Add web origins</a>
Admin URL ?	<input type="text" value="http://localhost:9090"/>

## Capability config

Client authentication ?	<input checked="" type="checkbox"/> On
Authorization ?	<input type="checkbox"/> Off
Authentication flow	<div><input checked="" type="checkbox"/> Standard flow ?<input 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 ?</div>

- Configuramos también un usuario y una contraseña. Vamos a tomar como prueba **admin** y **contraseña** password123.

## Creación de Microservicio

Como **microservicio** de prueba vamos a utilizar uno pre-armado para no perder tiempo en la configuración y creación desde cero de uno.


Podés descargar el microservicio base en el siguiente link: [Descargar Microservicio Base](#).


En caso de que no lo encuentres en el link, también estará subido en la plataforma para que lo puedas descargar.

Las configuraciones a realizar en keycloak para este microservicio son:







- Creación de un cliente llamado backend
- Dejar encendida la autenticación de cliente y hacer click en save.

Capability config

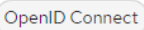
Client authentication  On

Authorization  Off

Authentication flow


<input checked="" type="checkbox"/> Standard flow 	<input 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 	


- Obtener la clave secreta en la nueva pestaña credentials del cliente una vez guardado.

backend 

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



Client secret .....

- Colocar la clave secreta en el **aplicacion.properties**

```
server.port=8086

spring.security.oauth2.client.provider.keycloak.issuer-uri=http://localhost:8080/realms/dh
spring.security.oauth2.client.registration.keycloak.client-id=backend
spring.security.oauth2.client.registration.keycloak.client-secret=ByQtLlHKVyQsmu6EMEPgXQ5CPwLwJEJ2
spring.security.oauth2.client.registration.keycloak.redirect-uri=http://localhost:8086/login/oauth2/code/google
```

## Configuración del Gateway

Para comenzar con la configuración de nuestro gateway vamos a crear un nuevo proyecto en Spring Initializr teniendo en cuenta las siguientes configuraciones:

Un detalle **MUY IMPORTANTE** a tener en cuenta son las versiones que seleccionemos. Recordá que si elegís como versión de **Spring Boot la 3** o cualquiera en adelante, tenés que tener instalado en tu pc y en tu IDE por lo menos el **JDK 17** o una versión superior para asegurarte compatibilidad.

The screenshot shows the Spring Initializr configuration page. On the left, under 'Project', 'Gradle - Groovy' and 'Gradle - Kotlin' are unselected, while 'Gradle - Maven' is selected. Under 'Language', 'Java' is selected, 'Kotlin' and 'Groovy' are unselected. Under 'Spring Boot', '3.1.0 (SNAPSHOT)', '3.1.0 (M2)', and '3.0.7 (SNAPSHOT)' are unselected, while '3.0.6' is selected. '2.7.12 (SNAPSHOT)' and '2.7.11' are also unselected. Under 'Project Metadata', 'Group' is 'com.dh', 'Artifact' is 'apigateway-service', 'Name' is 'apigateway-service', 'Description' is 'Demo project for Spring Boot', and 'Package name' is 'com.dh.apigateway-service'. Under 'Packaging', 'Jar' is selected, 'War' is unselected. Under 'Java', '20' and '11' are unselected, while '17' is selected. On the right, under 'Dependencies', 'Gateway' is selected with 'SPRING CLOUD ROUTING' and 'OAuth2 Client' is selected with 'SECURITY'. A button 'ADD DEPENDENCIES... CTRL + B' is visible.

Podés descargar esta misma configuración desde este link: [Link Configuración](#)

Una vez creado el proyecto, lo descargamos, lo descomprimos y lo abrimos en nuestro IDE.

Una vez levantado nuestro proyecto, procederemos a dividir las configuraciones del gateway en **tres partes**.

En **la primera** configuraremos lo necesario para conectar con **Keycloak**. En **la segunda** configuraremos el **gateway** en sí como microservicio, para asegurarnos que si alguien quiere

acceder al endpoint esté autenticado y que, en caso de que no lo esté, se redirija a la pantalla de login de Keycloak para solicitar la autenticación; y en **la tercera** configuraremos la seguridad del Gateway.

## 1era Parte

Pondremos la siguiente configuración en nuestro archivo `application.properties` del proyecto del gateway

```
spring.security.oauth2.client.provider.api-gateway-service.issuer-uri=http://localhost:8080/realms/dh
spring.security.oauth2.client.registration.api-gateway-service.provider=api-gateway-provider
spring.security.oauth2.client.registration.api-gateway-service.client-id=api-gateway-client
spring.security.oauth2.client.registration.api-gateway-service.client-secret=
spring.security.oauth2.client.registration.api-gateway-service.scope=openid
spring.security.oauth2.client.registration.api-gateway-service.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.api-gateway-service.redirect-uri=http://localhost:9090/login/oauth2/code/keycloak

server.port=9090
```

Como vemos, el campo “client-secret” está vacío. El valor que va en el mismo lo obtendremos de Keycloak en el siguiente apartado:

**api-gateway-client** OpenID Connect

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

Settings Keys **Credentials** Roles Client scopes Service accounts roles Sessions Advanced

Client Authenticator Client Id and Secret

Save

Client secret ..... 👁 📋 Regenerate

Registration access token 📋 Regenerate

Recordá que, si no te aparece la pestaña “Credentials” es porque no activaste la Autenticación del cliente cuando llevaste a cabo su creación. Volvé al primer apartado del tutorial donde se muestra las configuraciones del cliente y asegurate de tener todas igual y darle click a **Save**.

Una vez copiado el secret, vamos a completar el valor que nos otorgue en nuestro archivo **application.properties**.

## 2da Parte

Ahora debemos configurar el ruteo de nuestro gateway al microservicio correspondiente. Lo vamos a hacer agregando la siguiente configuración:

```
spring.cloud.gateway.default-filters[0]=TokenRelay
spring.cloud.gateway.routes[0].id=product-service
spring.cloud.gateway.routes[0].uri=http://localhost:8083
spring.cloud.gateway.routes[0].predicates[0]=Path=/hello/**
```

La opción **Token Relay** es un filtro que toma el token que llega en la request y lo agrega en el ruteo hacia el microservicio al que se dirige la solicitud.

Luego por cada ruta de microservicio que tengamos tenemos que establecer y realizar aquí las configuraciones necesarias. Como por ahora tenemos solo un microservicio, configuraremos solo ese.

En id colocamos un nombre identificador que queramos para nuestro microservicio, en uri colocaremos su correspondiente dirección y puerto y en path colocaremos el correspondiente más una barra con dos asteriscos /\*\*.

## 3ra Parte

Procederemos ahora a configurar la seguridad de nuestro gateway, forzando la autenticación de cada usuario que quiera acceder a nuestros endpoints de nuestros microservicios. Para ello vamos a crear una nueva clase de configuración llamada **SecurityConfig**.

```
@Configuration
public class SecurityConfig {

    @Bean
    public SecurityWebFilterChain springSecurityFilterChain
(ServerHttpSecurity http) {

        http
            .authorizeExchange()
            .anyExchange()
            .authenticated()
            .and()
            .oauth2Login(); // to redirect to oauth2 login page.

        return http.build();
    }
}
```

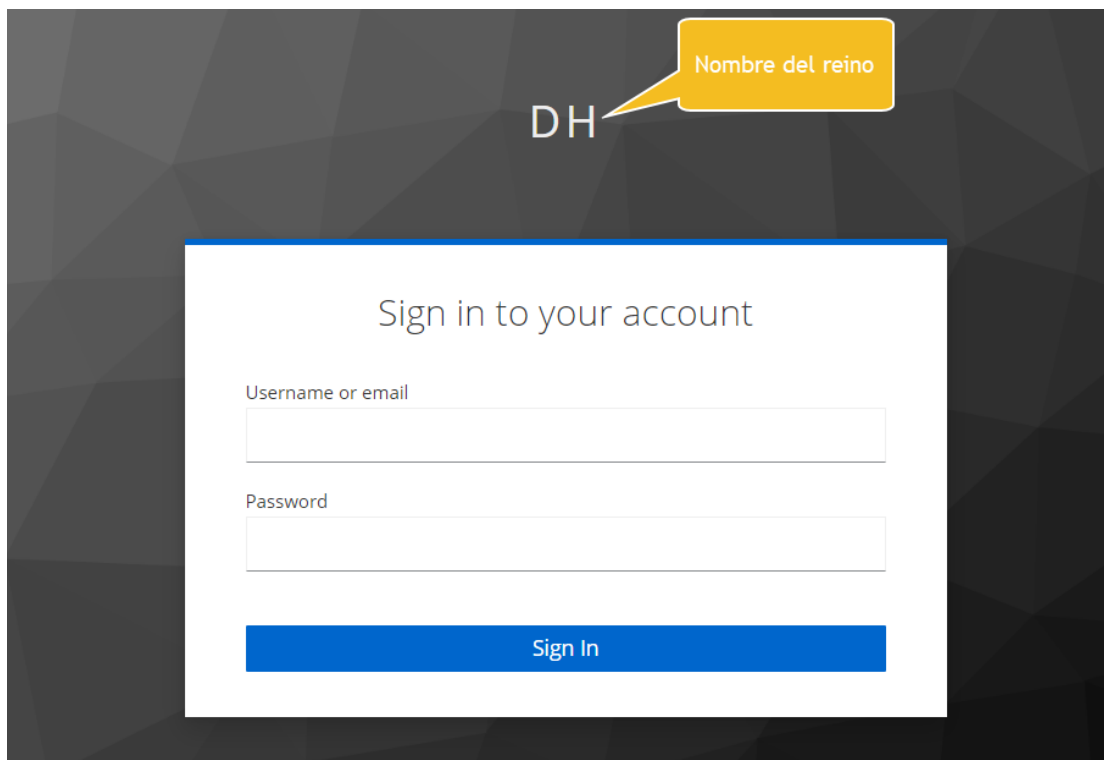
Una vez configurado todo esto, ejecutamos nuestro microservicio base (que vamos a utilizar para las pruebas) y también nuestro gateway.

## Pruebas

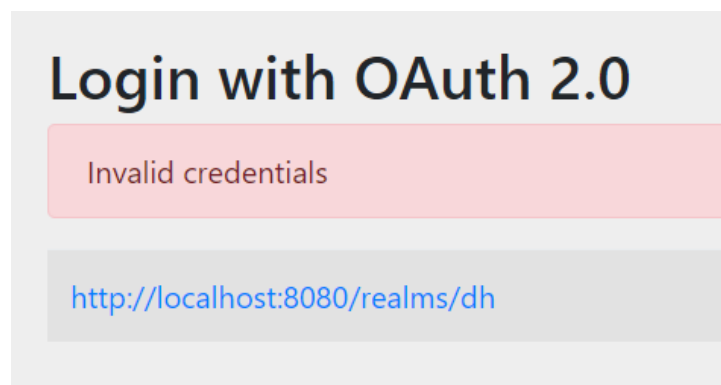
Una vez que se estén ejecutando tanto nuestro microservicio como nuestro gateway, vamos a proceder a probar que el gateway esté redireccionando correctamente a nuestro microservicio y también validar que está haciendo la autenticación

Para ello, vamos a dirigirnos a la url <https://localhost:9090/hello/user> (dato importante acá es recordar que el puerto 909 es el de nuestro gateway pero la dirección que le proporcionamos es el path de nuestro microservicio).

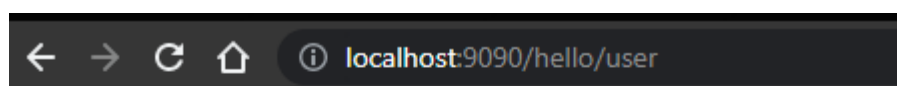
Si todo sale ok, el gateway nos va a redirigir a Keycloak y nos va a solicitar que nos autenquemos (mediante una pantalla de login) para validar que tengamos acceso al correspondiente endpoint:



En caso de que las credenciales no sean correctas, vamos a recibir el siguiente mensaje en la pantalla:



En caso de que coloquemos el usuario y contraseña de forma correcta y el mismo tenga los permisos para acceder al endpoint en cuestión, podremos acceder a la respuesta sin problema alguno.



hello