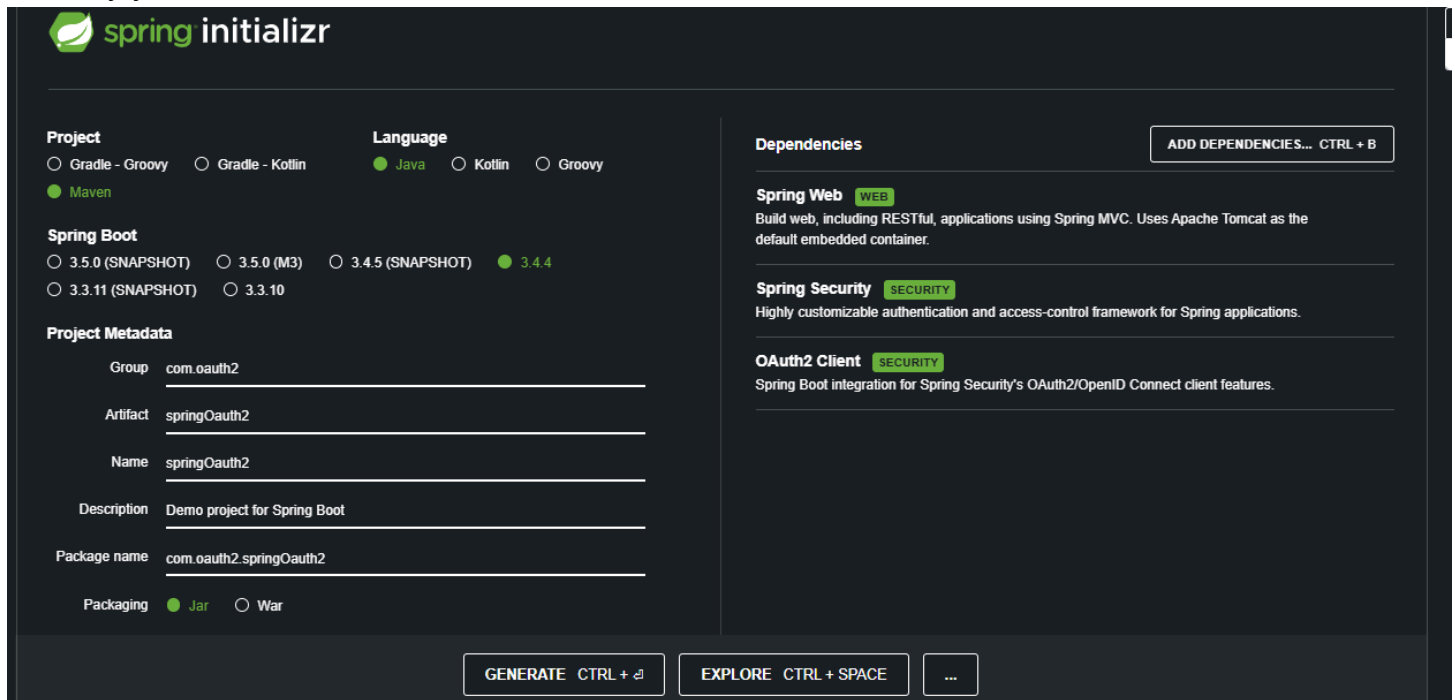


# Integrar Autenticacion con GitHub

## 1. Creamos el proyecto

Voy a trabajar con spring boot, por lo que descargare las dependencias de Spring web, Spring Security y OAuth2 Client



The screenshot shows the Spring Initializr web application. It has a dark theme. On the left, there are sections for 'Project', 'Language', 'Spring Boot', and 'Project Metadata'. The 'Project' section has radio buttons for 'Gradle - Groovy', 'Gradle - Kotlin', 'Java' (selected), 'Kotlin', and 'Groovy'. The 'Language' section has radio buttons for 'Maven' (selected), 'Gradle - Groovy', 'Gradle - Kotlin', 'Java' (selected), 'Kotlin', and 'Groovy'. The 'Spring Boot' section has radio buttons for '3.5.0 (SNAPSHOT)', '3.5.0 (M3)', '3.4.5 (SNAPSHOT)', '3.4.4' (selected), '3.3.11 (SNAPSHOT)', and '3.3.10'. The 'Project Metadata' section has input fields for 'Group' (com.oauth2), 'Artifact' (springOauth2), 'Name' (springOauth2), 'Description' (Demo project for Spring Boot), and 'Package name' (com.oauth2.springOauth2). There are also radio buttons for 'Packaging' (Jar selected, War). On the right, there is a 'Dependencies' section with a button 'ADD DEPENDENCIES... CTRL + B'. It lists 'Spring Web' (WEB), 'Spring Security' (SECURITY), and 'OAuth2 Client' (SECURITY) with their descriptions. At the bottom, there are buttons 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and a three-dot menu.

*Ejecutamos el proyecto para validar que todo este bien*

## 2. Creamos los controlladores para el ejemplo

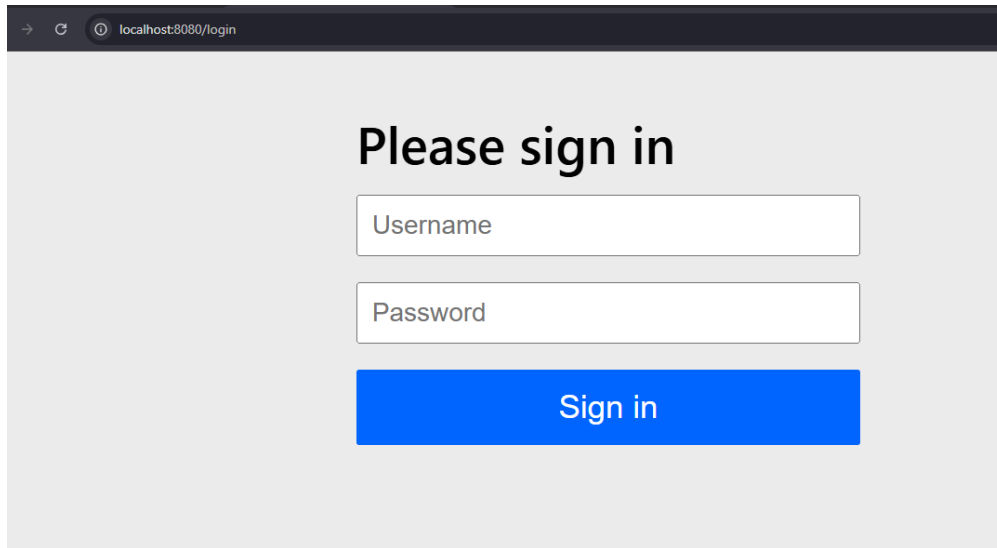
Creare dos rutas, una que hace referencia a una ruta publica y otra a una privada

```
1 package com.oauth2.springOauth2.Controller;  
2  
3 import org.springframework.web.bind.annotation.GetMapping;  
4 import org.springframework.web.bind.annotation.RestController;  
5  
6 no usages  
7 @RestController  
8 public class HomeController {  
9  
10     no usages  
11     @GetMapping("/hello")  
12     public String hello(){  
13         return "Hello, I'm Edwin. This is to test FREE endpoints";  
14     }  
15  
16     no usages  
17     @GetMapping("/helloSecured")  
18     public String helloSecured(){  
19         return "Hello, I'm Edwin. This is to SECURED free endpoints";  
20     }  
21 }
```

Creamos un usuario de prueba en el archivo `application.properties` para ingresar a los endpoints creados anteriormente:

```
1 spring.application.name=springOauth2
2
3 spring.security.user.name=admin
4 spring.security.user.password=1234
5 spring.security.user.roles=ADMIN
6
```

Nos logeamos con las credenciales



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/login'. The main content area has a light gray background. In the center, there is a heading 'Please sign in' in a large, bold, black font. Below the heading are two white input fields with gray borders. The first field is labeled 'Username' and the second is labeled 'Password'. Below these fields is a solid blue button with the text 'Sign in' in white.

Accedemos a la ruta `/hello`



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/hello'. The main content area is white and displays the text 'Hello, I'm Edwin. This is to test FREE endpoints' in a black font.

### 3. Activar la seguridad de OAuth2 y configurar los endpoints a los que se tendran acceso

una vez validado que todo funciona bien, creamos un archivo de configuracion para la seguridad, donde se activa la autenticación mediante proveedores OAuth2 como **Google, GitHub, etc.**, usando la configuración por defecto, además colocamos los permisos que tendran los diferentes endpoints

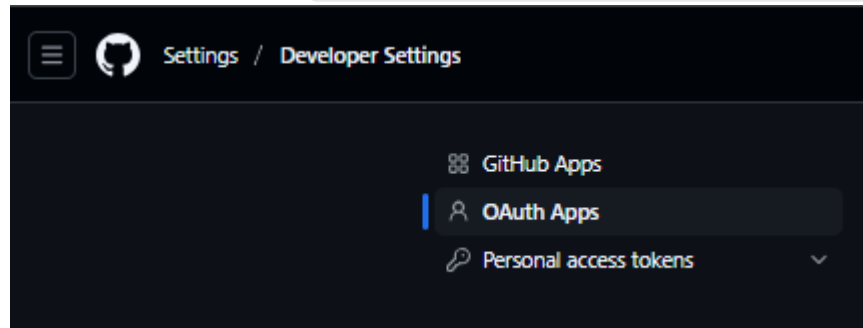
```

7  import org.springframework.security.config.annotation.web.builders.HttpSecurity;
8  import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
9  import org.springframework.security.web.SecurityFilterChain;
10
11  no usages
12  @Configuration
13  @EnableWebSecurity
14  public class SecurityConfig {
15
16      no usages
17      @Bean
18      @
19      public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {
20          return httpSecurity
21              .authorizeHttpRequests(request -> {
22                  request.requestMatchers(HttpMethod.GET, ...patterns: "/", "/hello").permitAll();
23                  request.anyRequest().authenticated();
24              })
25              .formLogin(Customizer.withDefaults())
26              .oauth2Login(Customizer.withDefaults())
27              .build();
28      }
29  }

```

## 4. En gitHub

Nos dirigimos a nuestra cuenta de github `my-account/Setting/Developer-Settings/`



*Una vez alli creamos una nueva aplicacion OAuth2*

Register a new OAuth app

Application name \*

springOauth2

Something users will recognize and trust.

Homepage URL \*

http://localhost:8080

The full URL to your application homepage.

Application description

Application description is optional

This is displayed to all users of your application.

Authorization callback URL \*

http://localhost:8080/login/oauth2/code/github

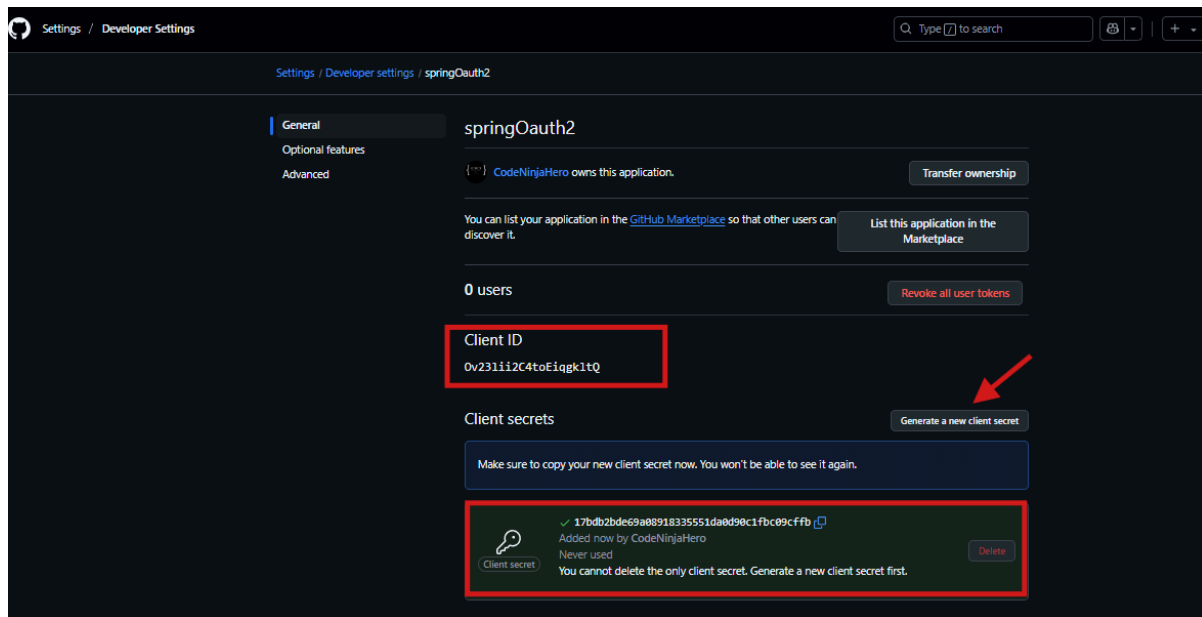
Your application's callback URL. Read our [OAuth documentation](#) for more information.

☒ Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow. Read the [Device Flow documentation](#) for more information.

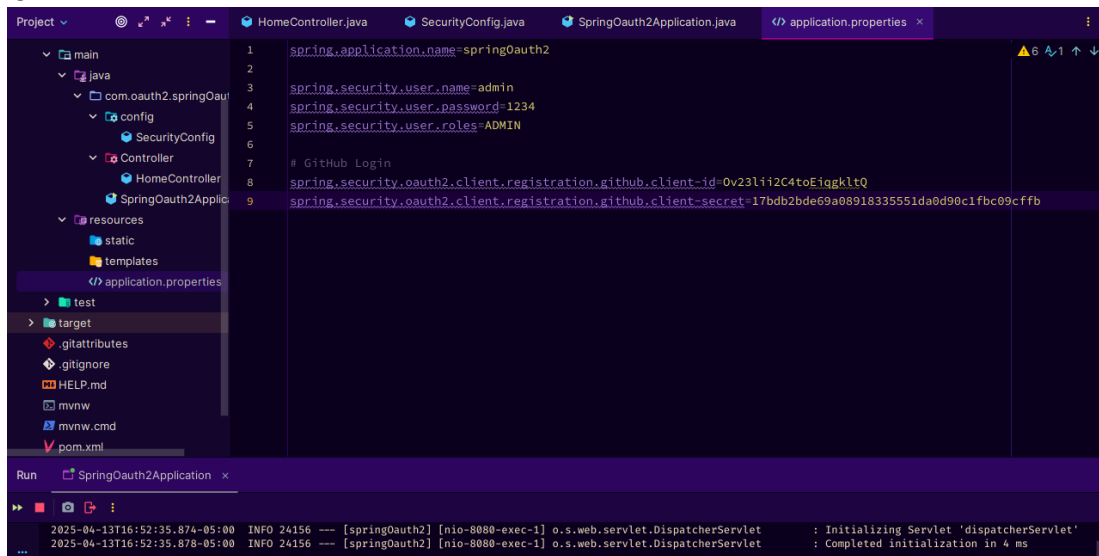
Register application Cancel

*Después de darle en registrar, nos proporcionara el **Client ID**, le damos en generar **Client Secret**, y listo esos son los dos datos que necesitamos*

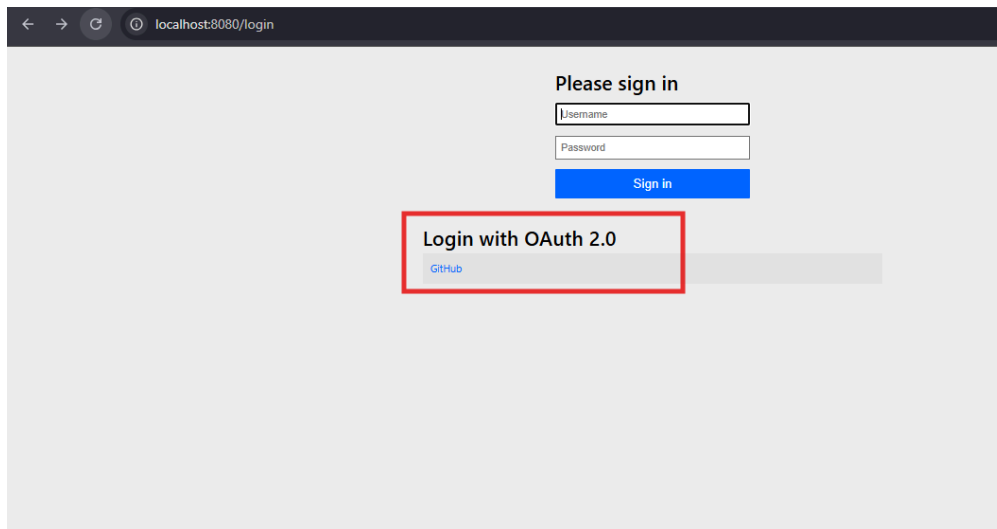


## 5. Agregar la conexion

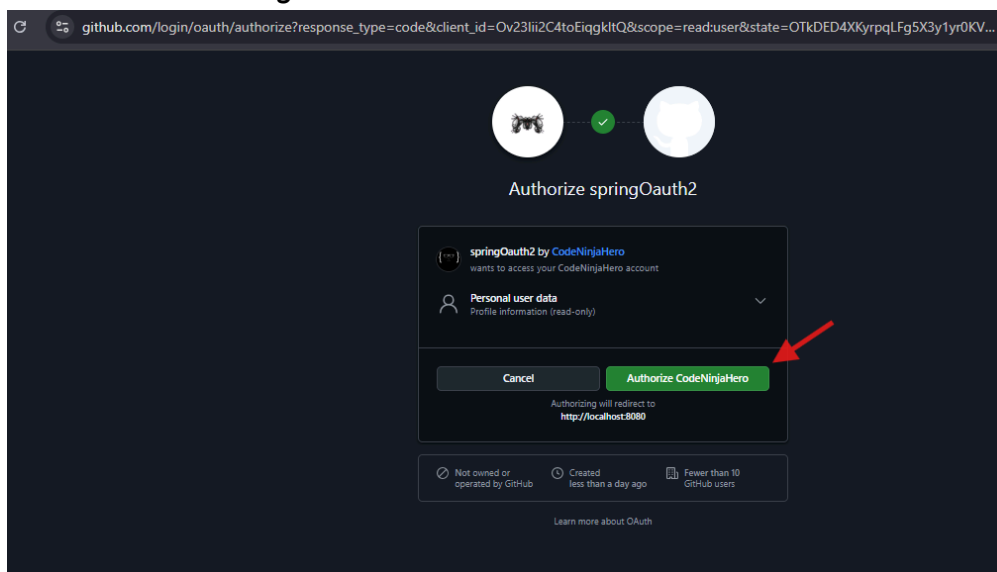
En el archivo `application.properties` añadimos las siguientes dos líneas con los datos que nos proporciona github



Ejecutamos de nuevo



*La integración de OAuth 2 usando gitHub fue finalizada*



Enlace del repositorio:

```
git clone https://github.com/CodeNinjaHero/OAuth.git
cd springoauth2
```