



Acceso a datos

ActUT05_02. Añadir seguridad a
nuestra ApiRest

Francisco José García Cutillas | 2FPGS_DAM



Índice

Añadir seguridad a nuestra ApiRest	3
Bibliografía	8

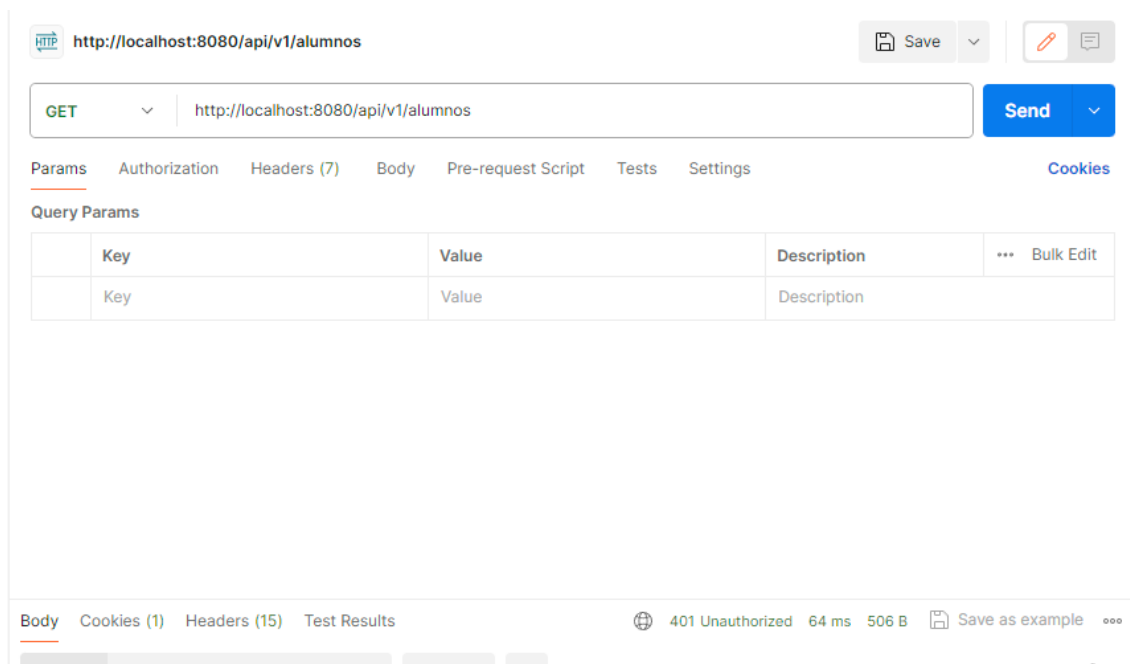
Añadir seguridad a nuestra ApiRest

En el caso de nuestra ApiRest vamos a añadirle una mejora que va a constar de una autenticación por usuario y contraseña para poder realizar las peticiones.

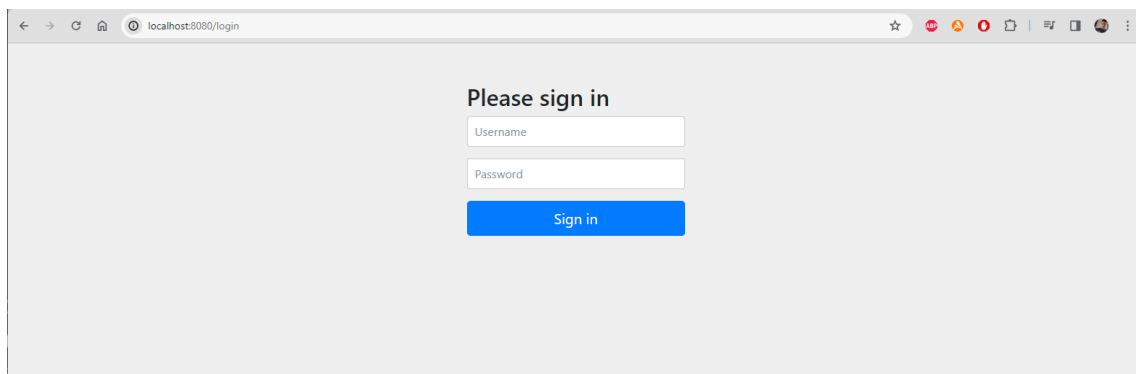
Para comenzar vamos a añadir la siguiente dependencia en nuestro fichero pom.xml.

```
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
</dependencies>
```

Una vez que añadimos la dependencia, si volvemos a realizar una petición a nuestra Api, podemos observar que nos da el error 401, indicando que no estamos autorizados.

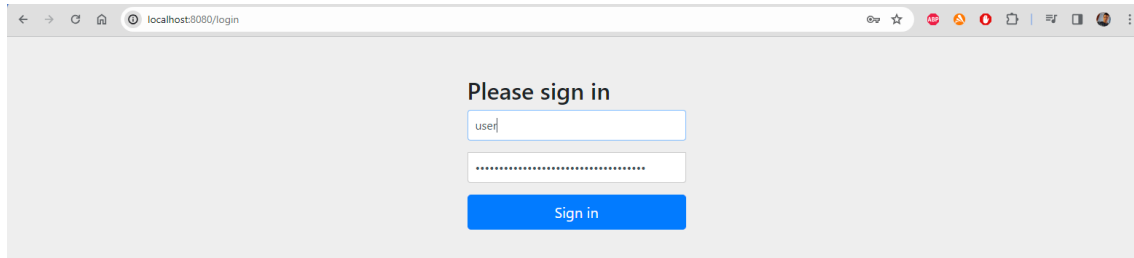


Sin embargo, en nuestro navegador se vería de la siguiente manera:



Ahora mismo no tenemos ningún usuario ni contraseña almacenados. Pero Spring por defecto tiene el usuario “user” y la contraseña siguiente. Ésta nos la muestra por el log del servidor.

```
2024-02-27T16:26:19.411+01:00 WARN 20980 --- [ restartedMain] .s.s.UserDetailsServiceAutoConfiguration :  
  
Using generated security password: 12ec0441-7f11-4795-8b62-8f5099507093  
  
This generated password is for development use only. Your security configuration must be updated before running your application in production.
```



localhost:8080/login

Please sign in

user

.....

Sign in

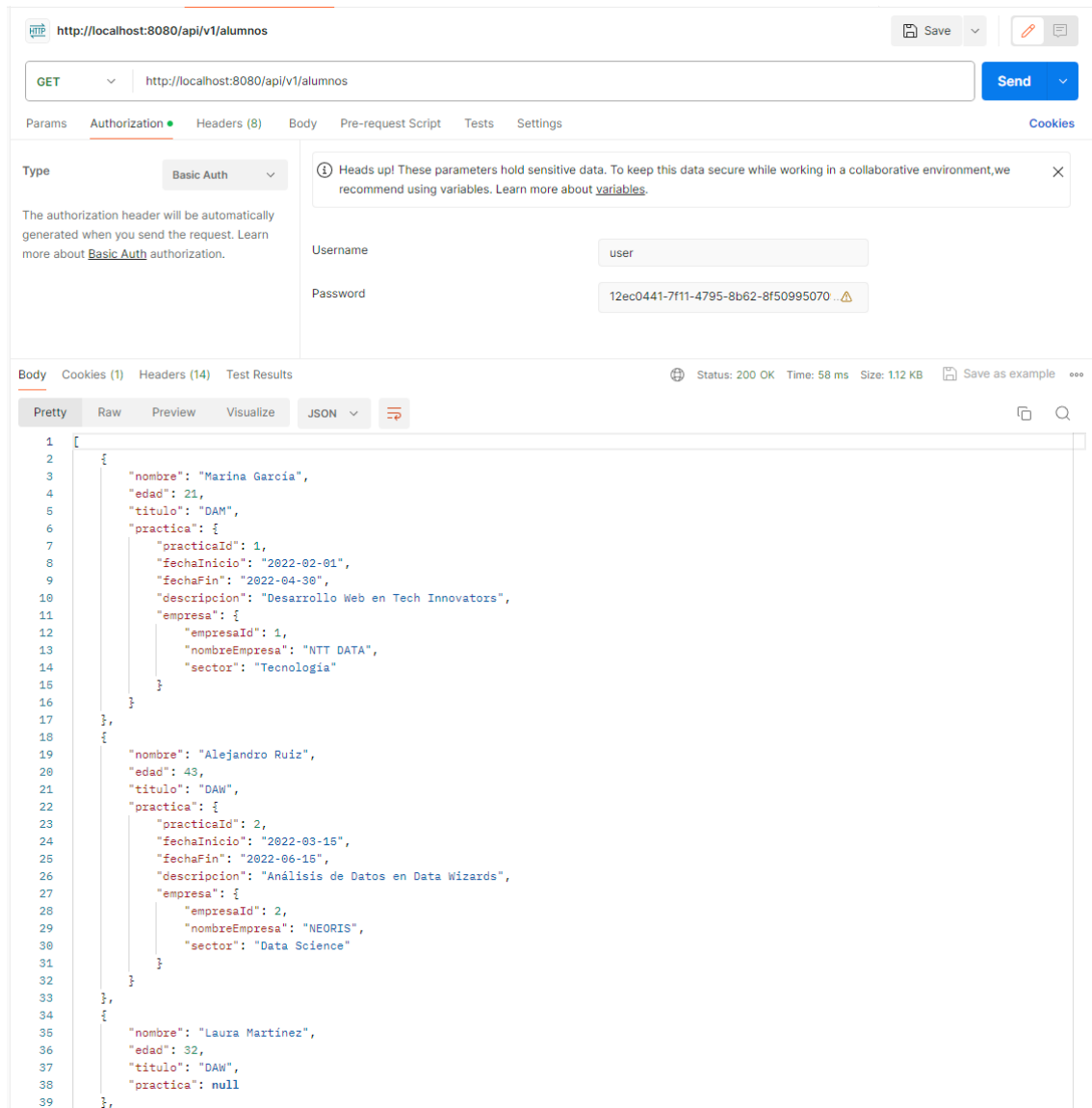
Utilizando dichas credenciales, ya nos permite realizar nuestra consulta.



localhost:8080/api/v1/alumnos?continue

```
[{"nombre":"Marina García","edad":21,"titulo":"DAW","practica":{"practicaId":1,"fechaInicio":"2022-02-01","fechaFin":"2022-04-30","descripcion":"Desarrollo Web en Tech Innovators","empresa":{"empresaId":1,"nombreEmpresa":"NTT DATA","sector":"Tecnología"}}}, {"nombre":"Alejandro Ruiz","edad":43,"titulo":"DAW","practica":{"practicaId":2,"fechaInicio":"2022-03-15","fechaFin":"2022-06-15","descripcion":"Análisis de Datos en Data Wizards","empresa":{"empresaId":2,"nombreEmpresa":"MEGALIS","sector":"Data Science"}}}, {"nombre":"Laura Martinez","edad":32,"titulo":"DAW","practica":null}, {"nombre":"Javier López","edad":24,"titulo":"DAW","practica":null}, {"nombre":"Sara Rodriguez","edad":21,"titulo":"ASIA","practica":null}]
```

En PostMan, debemos ir a la pestaña “Authorization” y en Type especificamos “Basic Auth”. Usando las mismas credenciales que anteriormente, también nos deja acceder.



Ahora nos vamos a ir a nuestro “application.properties” en `src/main/resources` y añadimos lo siguiente:

```

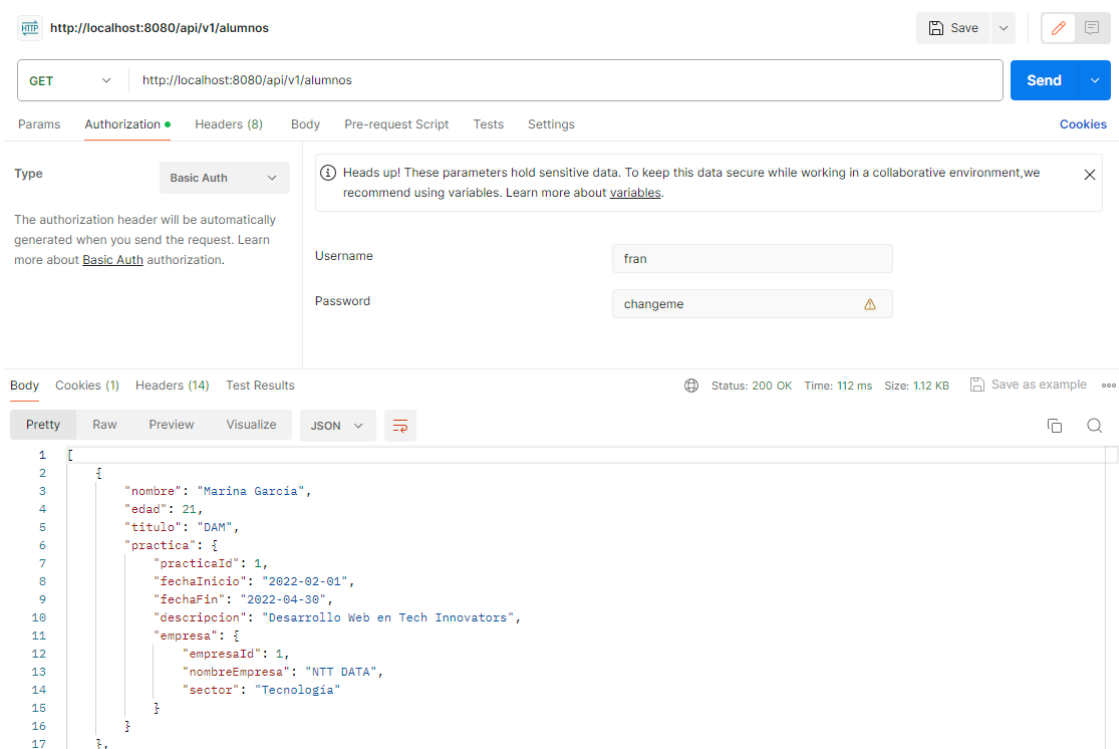
10  spring.security.user.name=fran
11  spring.security.user.password=changeme
12

```

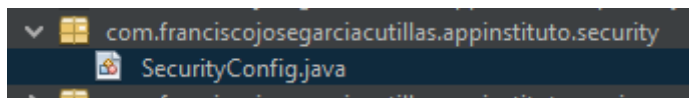
Éstas serán a partir de ahora nuestras credenciales para acceder al sistema.

Cabe destacar que en un entorno de producción la contraseña no puede estar en claro, pero para realizar estas pruebas lo vamos a hacer de esta manera.

Una vez hecho esto, podemos comprobar que con estas nuevas credenciales ya nos dejaría acceder.



Ahora vamos a pasar a configurar la seguridad de nuestra aplicación. Para ello vamos a crear un nuevo paquete que hemos llamado “security” y, dentro del mismo, nuestra clase de configuración “SecurityConfig”.



Antes de empezar con la configuración vamos a crear un end point de tipo GET, que en nuestro caso va a simular como si fuera una página de bienvenida en la que no sería necesario estar logueado para poder verla.

```

//EndPoint de prueba para Spring Security
@GetMapping("/hello")
public String hello(){
    return "Bienvenido";
}

```

En la clase “SecurityConfig” se encuentra toda aquella configuración relacionada con la manera de acceder a nuestra ApiRest.

En nuestro caso, sólo hemos configurado un filtro en el que va a permitir que cualquier usuario que acceda a la ruta “/api/vi/hello”, pueda ver la página. Ésta sería la página de inicio de nuestra api.

Sin embargo, si quisieran acceder a otro endPoint distinto del anterior, ya sería necesario autenticarse en el sistema.

```

1 package com.franciscojosegarciaacutillas.appinstituto.security;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
6 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
7 import org.springframework.security.config.http.SessionCreationPolicy;
8 import org.springframework.security.web.SecurityFilterChain;
9 import org.springframework.security.web.authentication.AuthenticationSuccessHandler;
10
11 /**
12  *
13  * @author Fran
14  */
15 @Configuration
16 @EnableWebSecurity
17 public class SecurityConfig {
18
19     @Bean
20     public SecurityFilterChain filterChain(HttpSecurity httpSecurity) throws Exception {
21
22         return httpSecurity
23             .authorizeHttpRequests(auth -> {
24                 auth.requestMatchers(patterns: "/api/v1/hello").permitAll(); //Permite el acceso sin login a esta ruta
25                 auth.anyRequest().authenticated(); //Todo lo que no sea lo anterior, se necesita autenticación
26             })
27             //Dónde ir si se accede con éxito en el login
28             .formLogin(form -> {
29                 form.successHandler(successHandler: successHandler());
30                 form.permitAll();
31             })
32             //Política de creación de la sesión
33             .sessionManagement(session -> {
34                 session.sessionCreationPolicy(sessionCreationPolicy: SessionCreationPolicy.ALWAYS);
35             })
36             .build();
37
38     }
39
40     //Método para indicar a qué endPoint debe ir cuando se loguee un usuario
41     public AuthenticationSuccessHandler successHandler() {
42         return ((request, response, authentication) -> {
43             response.sendRedirect( string: "/api/v1/inicio");
44         });
45     }
46
47 }

```

Esta página se mostraría sin necesidad de autenticarse.

http://localhost:8080/api/v1/hello

GET http://localhost:8080/api/v1/hello

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies (1) Headers (14) Test Results

Status: 200 OK Time: 5 ms Size: 433 B Save as example

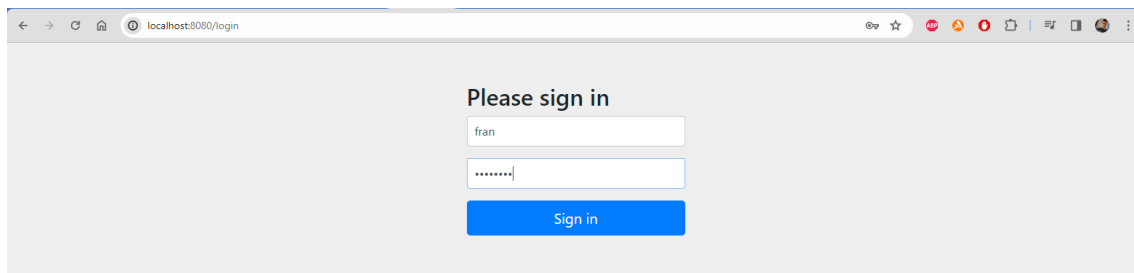
Pretty Raw Preview Visualize Text

1 Bienvenido

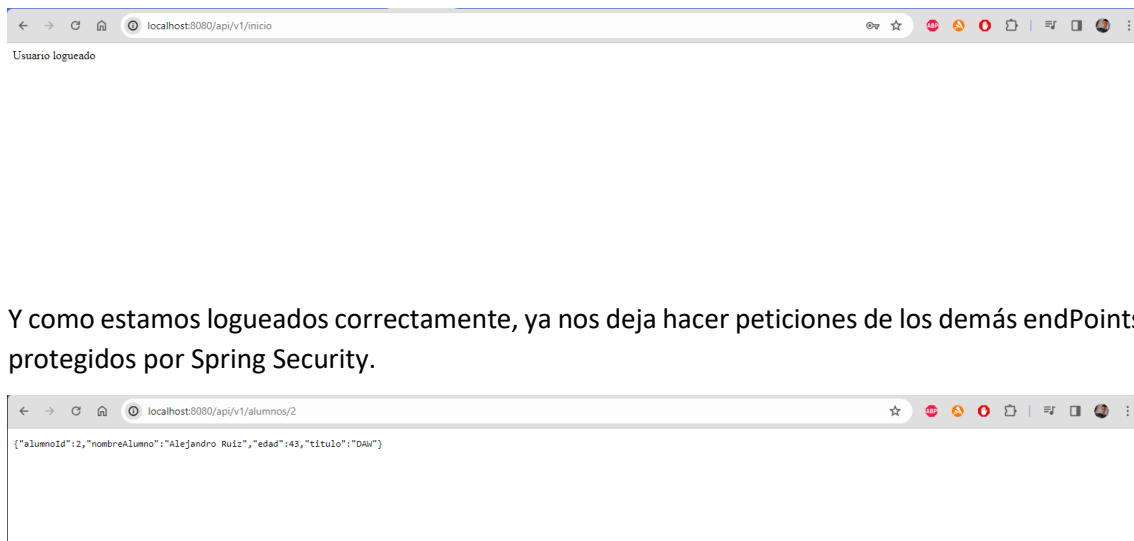
Vamos a crear un endPoint al que se dirigirá nuestra api cuando un usuario se haya logueado correctamente.

```
//EndPoint para usuario logueado
@GetMapping("/inicio")
public String inicio(){
    return "Usuario logueado";
}
```

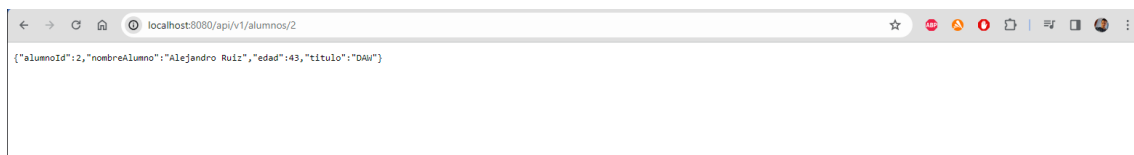
Nos logueamos.



Nos muestra nuestro endPoint que hemos configurado en SecurityConfig.java



Y como estamos logueados correctamente, ya nos deja hacer peticiones de los demás endpoints protegidos por Spring Security.



Bibliografía

- <https://spring.io/guides/gs/securing-web>
- <https://github.com/spring-guides/gs-securing-web.git>
- <https://github.com/LeoOlivaresD/Spring-Security-JWT.git>
- <https://dev.to/noelopez/spring-security-lambda-dsl-1a4m>
- <https://github.com/UnProgramadorNace/Spring-Security-Basic.git>