

Anotaciones comunes

@NotNull: Indica que un campo no puede ser nulo.

@NotEmpty: Indica que una lista no puede estar vacía.

@NotBlank: Indica que un string no puede estar vacío.

@Min and **@Max:** Indica un rango de valores numéricos aceptados.

@Pattern: Indica la estructura de un string con una expresión regular.

@Email: Indica que un campo debe ser una dirección de correo válida.

<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation>

<https://github.com/yoniescobar/cliente-react-sabado.git>

<https://github.com/yoniescobar/api-restfull-sabado.git>

Temas

Introducción a la seguridad de APIs Rest.

Crearemos una API de usuario y roles.


Oauth 2 con JWT para securizar nuestras APIs.

Métodos de seguridad en aplicaciones:


- Uso de sesiones en aplicaciones monolíticas.
- Seguridad basada en token.

¿Qué es un JSON Web Token?

- JWT es un estándar RFC 7519 para transmitir información con la identidad y claims de un usuario de forma segura entre un cliente/servidor.
- En otras palabras; es una cadena de texto que tiene 3 partes codificadas en Base64, separadas por un punto (header.payload.firma) que generamos y entregamos a los clientes de nuestra API
- Es importante aclarar que la cadena/token esta codificado y lo crea nuestra aplicación

 **JWT**

Debugger Libraries Introduction Ask Get a T-shirt!

Crafted by  Auth0

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD: DATA

```
{  "sub": "1234567890",  "name": "John Doe",  "iat": 1516239022}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
) ☐ secret base64 encoded
```

OAuth 2.0 (Autorización abierta)

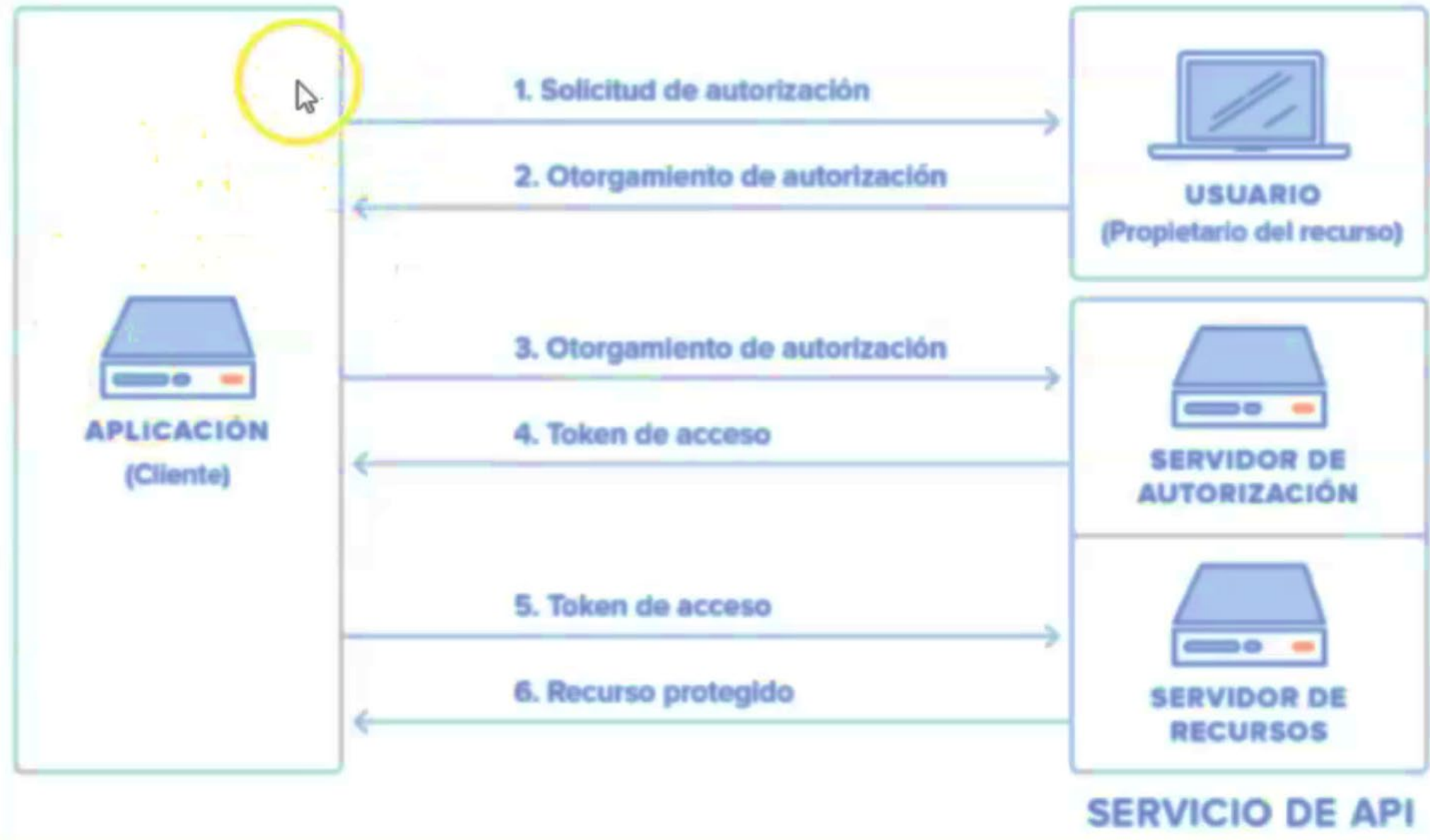
- OAuth 2 es un framework de autorización, que permite a las aplicaciones obtener acceso a recursos.
- OAuth 2 provee un flujo de autorización para aplicaciones web, aplicaciones móviles e incluso programas de escritorio.

Roles presentes en OAuth:

Client.- Es la aplicación cliente que quiere acceder a la cuenta de un usuario, en un servicio determinado. A fin de conseguir ello, debe contar con una autorización del usuario, y esta autorización se debe validar (a través de la API del servicio).

Resource Owner.- El "dueño del recurso" es el usuario que autoriza a una aplicación, para que pueda acceder a su cuenta. El acceso está limitado en función del "scope" que concede el usuario durante la autorización.

Resource & Authorization Server.- Resource server es el servidor que almacena las cuentas de usuarios, y authorization server es el servidor que verifica la identidad de los usuarios y emite access tokens a la aplicación cliente.





Nos provee la posibilidad de usar Oauth 2 y poder realizar nuestros servidores de autenticación y autorización para nuestras apis.

Es este curso realizaremos un sistema para securizar nuestras apis con spring security.


```
<!--      https://mvnrepository.com/artifact/org.springframework.security.oauth/spring-security-oauth2-->
<dependency>
  <groupId>org.springframework.security.oauth</groupId>
  <artifactId>spring-security-oauth2</artifactId>
  <version>2.3.4.RELEASE</version>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-jwt -->
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-jwt</artifactId>
  <version>1.0.9.RELEASE</version>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime -->
<dependency>
  <groupId>org.glassfish.jaxb</groupId>
  <artifactId>jaxb-runtime</artifactId>
</dependency>
```

```
Role.java
5 import lombok.NoArgsConstructor;
6
7 import javax.persistence.*;
8 import java.io.Serializable;
9
10 no usages new *
11 @Entity
12 @Table(name = "roles")
13 @NoArgsConstructor
14 @AllArgsConstructor
15 @Data
16 public class Role implements Serializable {
17     private static final long serialVersionUID = 1L; // serializable
18     @Id
19     @GeneratedValue(strategy = GenerationType.IDENTITY)
20     private Long id;
21     @Column(unique = true, length = 100) //Validaciones
22     private String nombre;
23 }
24

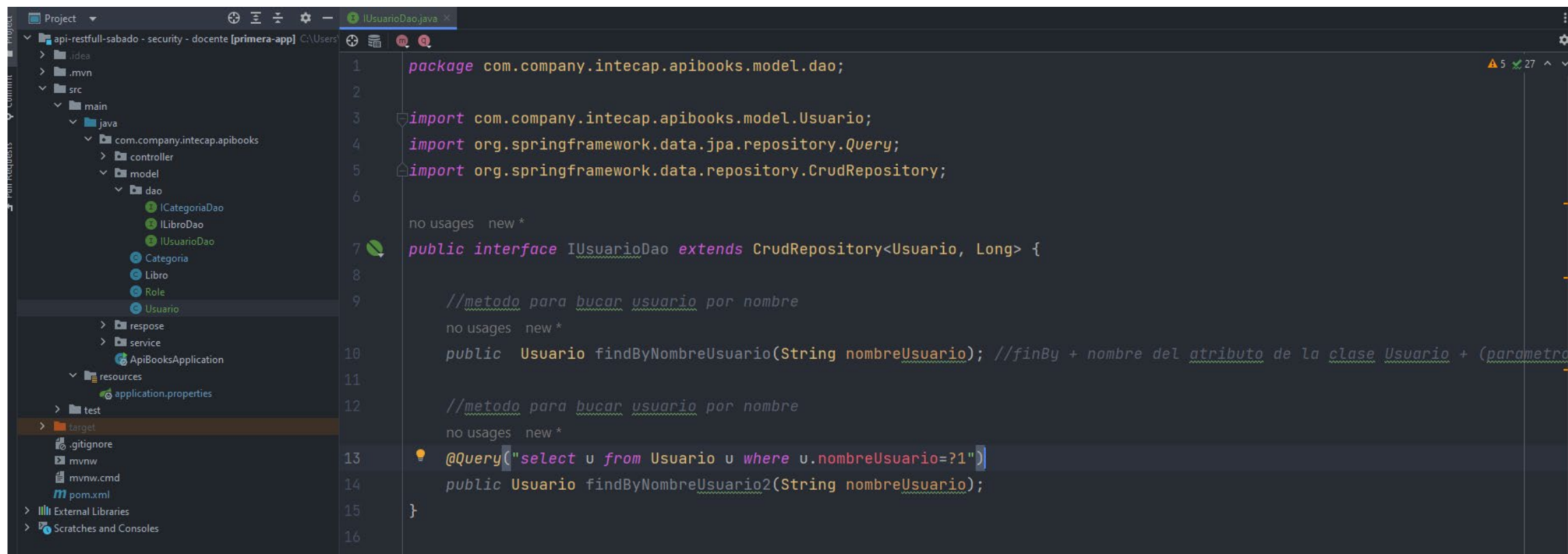
Usuario.java
4 import Lombok.Data;
5 import lombok.NoArgsConstructor;
6 import javax.persistence.*;
7 no usages new *
8 @Entity
9 @Table(name = "usuarios")
10 @NoArgsConstructor
11 @AllArgsConstructor
12 @Data
13 public class Usuario {
14     private static final long serialVersionUID = 1L; // serializable
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Long id;
18     @Column(unique = true, length = 100) //Validaciones
19     private String nombreUsuario;
20
21     @Column(length = 65)
22     private String password;
23
24     private Boolean habilitado;
25 }
```

Tabla Role y Usuario

```
Role.java
1 package com.company.intecap.apibooks.mode;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 import javax.persistence.*;
8 import java.io.Serializable;
9
10 @Entity
11 @Table(name = "roles")
12 @NoArgsConstructor
13 @AllArgsConstructor
14 @Data
15 public class Role implements Serializable {
16     private static final long serialVersionUID = 1L;
17     @Id
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private Long id;
20     @Column(unique = true, length = 100) //Validaciones
21     private String nombre;
22 }
23
24
25 Usuario.java
26 @Entity
27 @Table(name = "usuarios")
28 @NoArgsConstructor
29 @AllArgsConstructor
30 @Data
31 public class Usuario {
32     private static final long serialVersionUID = 1L; // serializable id
33     @Id
34     @GeneratedValue(strategy = GenerationType.IDENTITY)
35     private Long id;
36     @Column(unique = true, length = 100) //Validaciones
37     private String nombreUsuario;
38     @Column(length = 65)
39     private String password;
40     private Boolean habilitado; //para saber si el usuario esta habilitado o no
41
42     //Relacion de muchos a muchos
43     @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
44     @JoinTable(name = "usuarios_roles",
45         joinColumns = @JoinColumn(name = "usuario_id"),
46         inverseJoinColumns = @JoinColumn(name = "role_id"),
47         uniqueConstraints = {@UniqueConstraint(columnNames = {"usuario_id", "role_id"})})
48     private List<Role> roles;
49 }
```

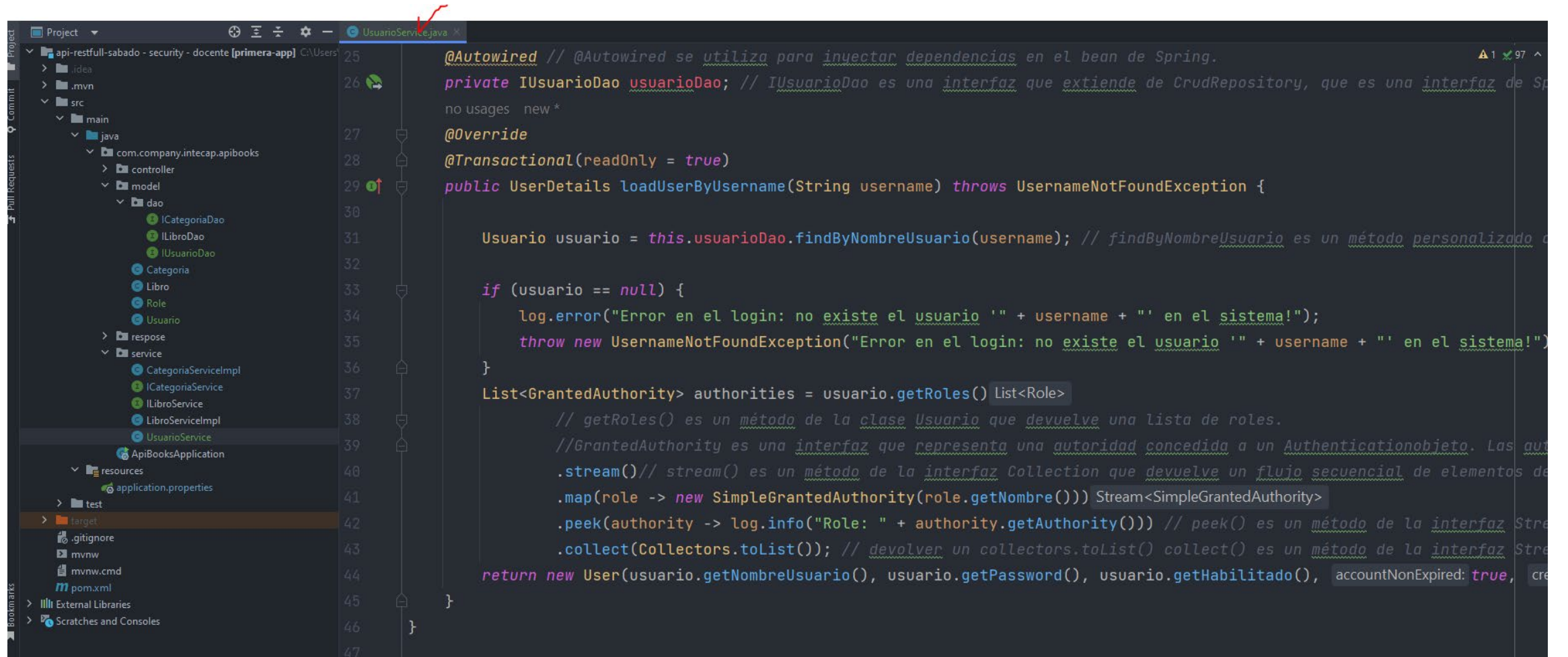
- Relaciones
- Un usuario tiene roles asociados
- Un usuario puede tener muchos roles y
- Un rol puede tener muchos usuarios

Interfaz de UsuarioDao.java



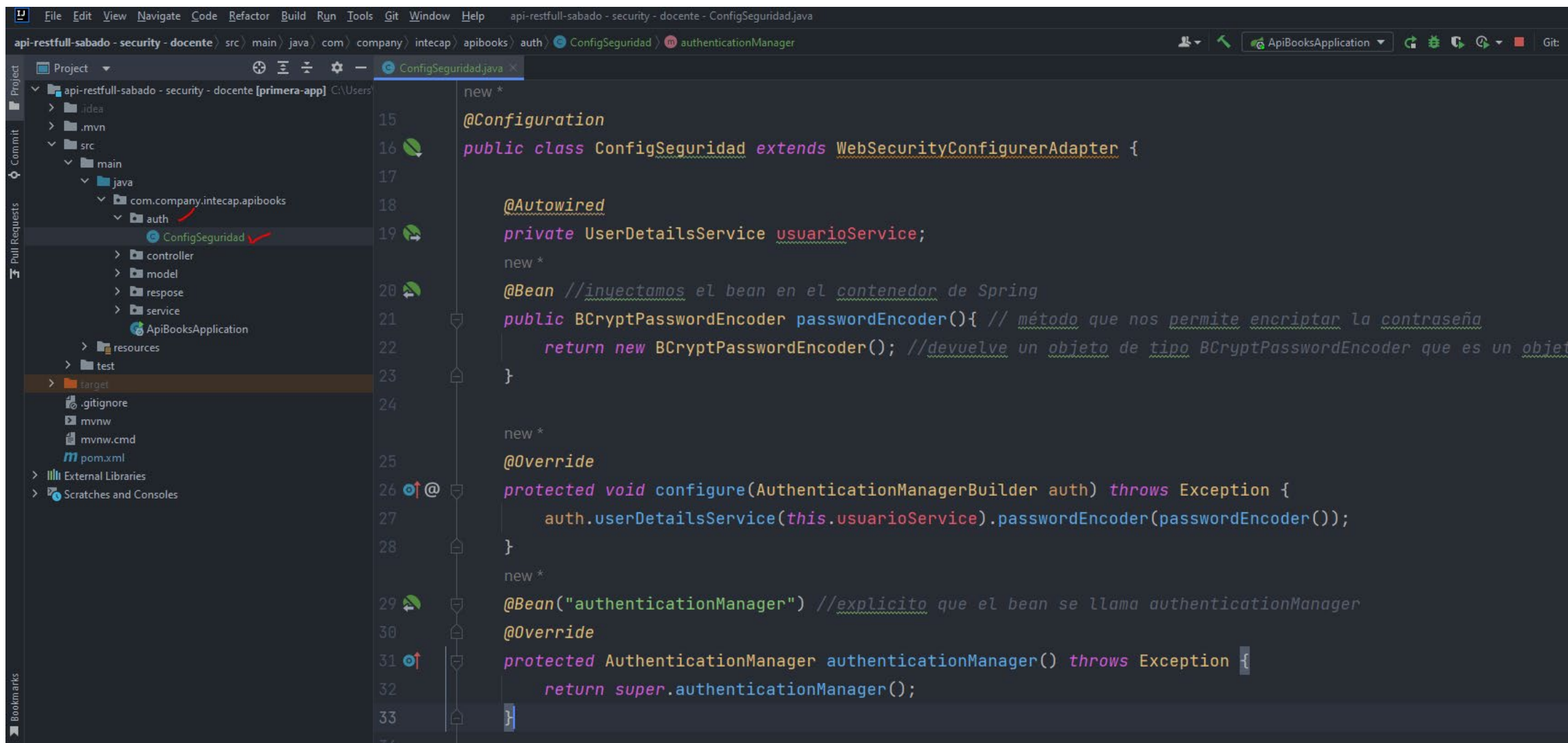
```
1 package com.company.intecap.apibooks.model.dao;
2
3 import com.company.intecap.apibooks.model.Usuario;
4 import org.springframework.data.jpa.repository.Query;
5 import org.springframework.data.repository.CrudRepository;
6
7 no usages new *
8 public interface IUsuarioDao extends CrudRepository<Usuario, Long> {
9
10     //metodo para buscar usuario por nombre
11     no usages new *
12     public Usuario findByNombreUsuario(String nombreUsuario); //finBy + nombre del atributo de la clase Usuario + (parametro
13
14     //metodo para buscar usuario por nombre
15     no usages new *
16     @Query("select u from Usuario u where u.nombreUsuario=?1")
17     public Usuario findByNombreUsuario2(String nombreUsuario);
18 }
```


Creando UsuarioService



```
25 @Autowired // @Autowired se utiliza para inyectar dependencias en el bean de Spring.
26 private IUserdao usuarioDao; // IUserdao es una interfaz que extiende de CrudRepository, que es una interfaz de Sp
no usages new *
27
28 @Override
29 @Transactional(readOnly = true)
30 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
31
32     Usuario usuario = this.usuarioDao.findByNombreUsuario(username); // findByNombreUsuario es un método personalizado d
33
34     if (usuario == null) {
35         log.error("Error en el login: no existe el usuario '" + username + "' en el sistema!");
36         throw new UsernameNotFoundException("Error en el login: no existe el usuario '" + username + "' en el sistema!");
37     }
38
39     List<GrantedAuthority> authorities = usuario.getRoles() List<Role>
40
41     // getRoles() es un método de la clase Usuario que devuelve una lista de roles.
42     // GrantedAuthority es una interfaz que representa una autoridad concedida a un Authenticationobjeto. Las aut
43     .stream()// stream() es un método de la interfaz Collection que devuelve un flujo secuencial de elementos de
44     .map(role -> new SimpleGrantedAuthority(role.getNombre())) Stream<SimpleGrantedAuthority>
45     .peek(authority -> log.info("Role: " + authority.getAuthority())) // peek() es un método de la interfaz Stre
46     .collect(Collectors.toList()); // devolver un collectors.toList() collect() es un método de la interfaz Stre
47     return new User(usuario.getNombreUsuario(), usuario.getPassword(), usuario.getHabilitado(), accountNonExpired: true, cre
```

ConfigSeguridad

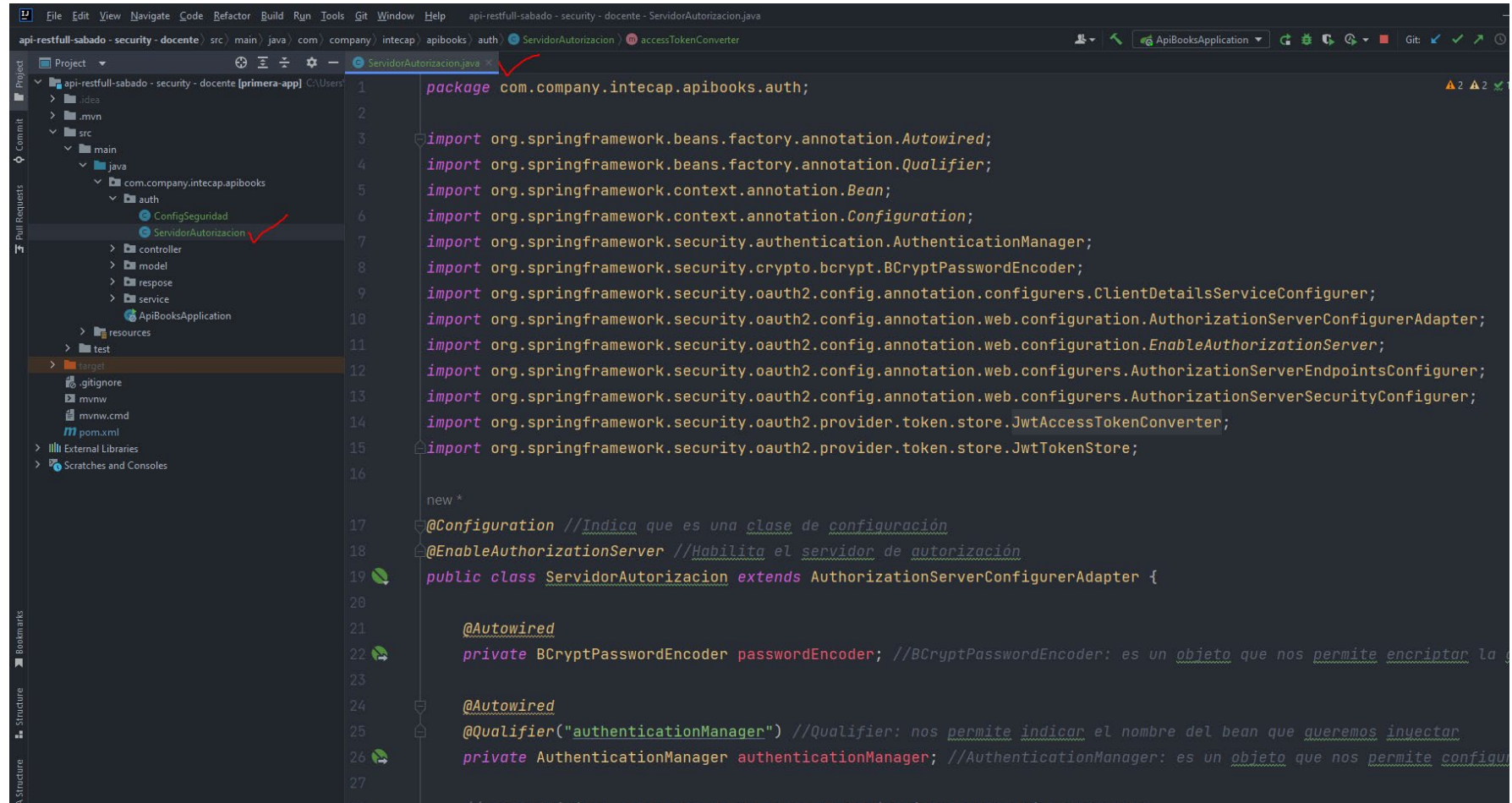


The screenshot shows an IDE with the following components:

- Project Explorer (Left):** Displays the project structure. The path is `api-restfull-sabado - security - docente > src > main > java > com > company > intecap > apibooks > auth > ConfigSeguridad`. The `ConfigSeguridad` file is highlighted with a red checkmark.
- Code Editor (Right):** Shows the content of `ConfigSeguridad.java`. The code is as follows:

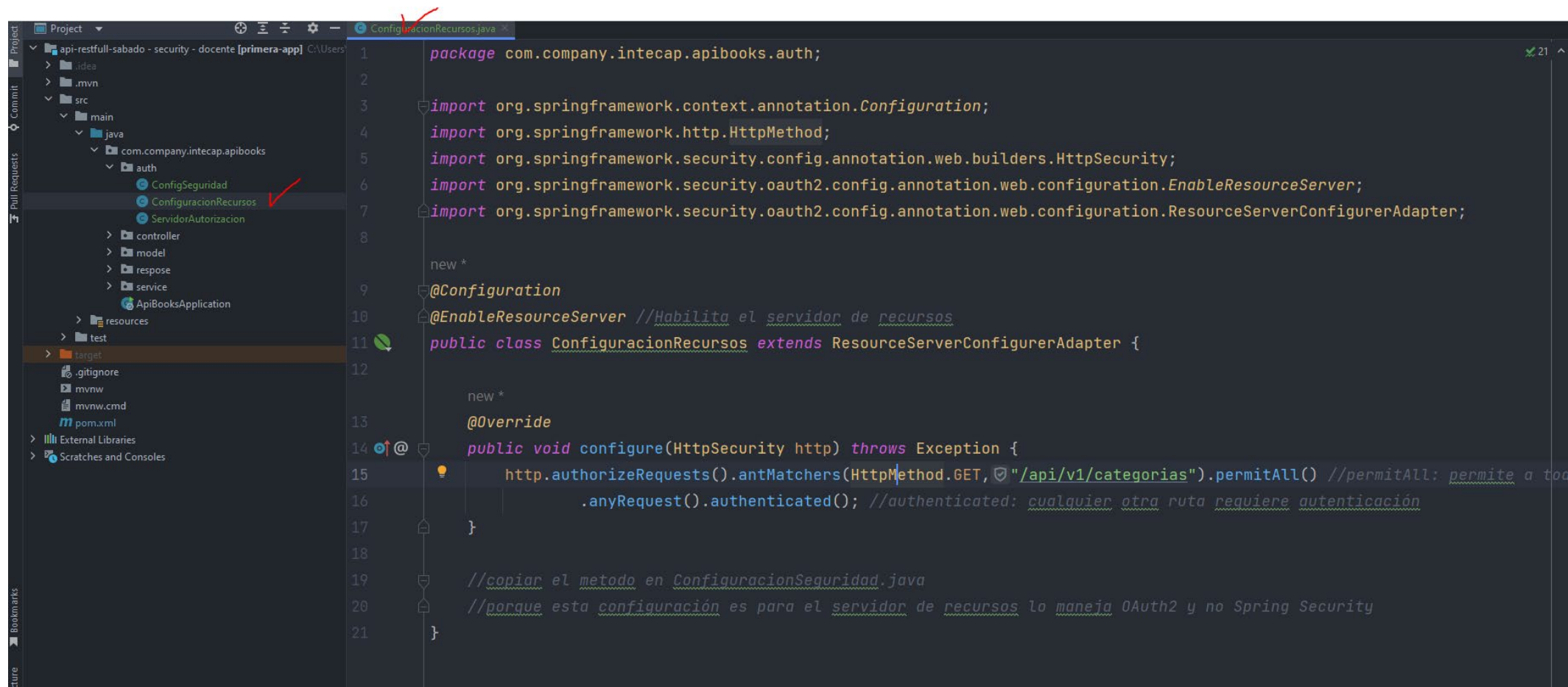
```
15 new *
16 @Configuration
17 public class ConfigSeguridad extends WebSecurityConfigurerAdapter {
18
19     @Autowired
20     private UserDetailsService usuarioService;
21
22     new *
23     @Bean //inyectamos el bean en el contenedor de Spring
24     public BCryptPasswordEncoder passwordEncoder(){ // método que nos permite encriptar la contraseña
25         return new BCryptPasswordEncoder(); //devuelve un objeto de tipo BCryptPasswordEncoder que es un objeto
26     }
27
28     new *
29     @Override
30     protected void configure(AuthenticationManagerBuilder auth) throws Exception {
31         auth.userDetailsService(this.usuarioService).passwordEncoder(passwordEncoder());
32     }
33
34     new *
35     @Bean("authenticationManager") //explicito que el bean se llama authenticationManager
36     @Override
37     protected AuthenticationManager authenticationManager() throws Exception {
38         return super.authenticationManager();
39     }
40 }
```

Configuración de Autorización generación de token y permiso



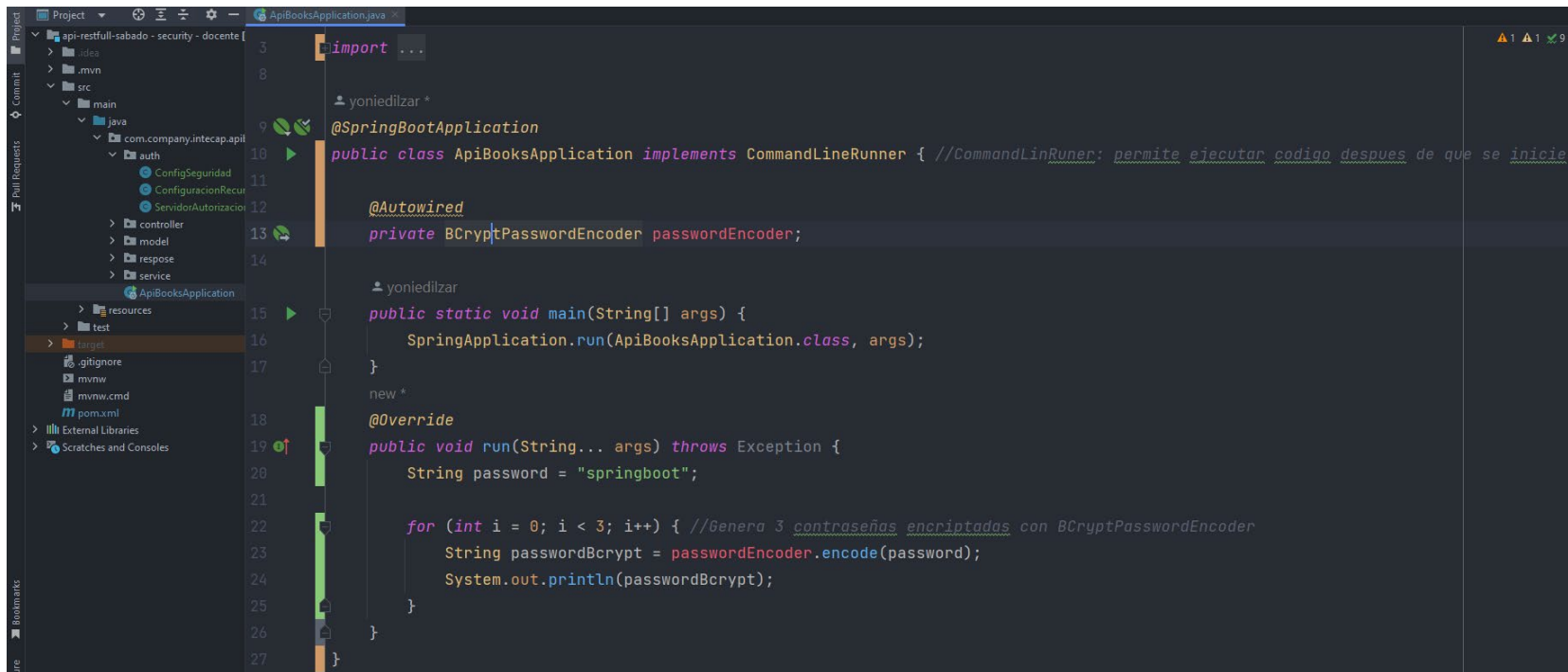
```
1 package com.company.intecap.apibooks.auth;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.beans.factory.annotation.Qualifier;
5 import org.springframework.context.annotation.Bean;
6 import org.springframework.context.annotation.Configuration;
7 import org.springframework.security.authentication.AuthenticationManager;
8 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
9 import org.springframework.security.oauth2.config.annotation.configurers.ClientDetailsServiceConfigurer;
10 import org.springframework.security.oauth2.config.annotation.web.configuration.AuthorizationServerConfigurerAdapter;
11 import org.springframework.security.oauth2.config.annotation.web.configuration.EnableAuthorizationServer;
12 import org.springframework.security.oauth2.config.annotation.web.configurers.AuthorizationServerEndpointsConfigurer;
13 import org.springframework.security.oauth2.config.annotation.web.configurers.AuthorizationServerSecurityConfigurer;
14 import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;
15 import org.springframework.security.oauth2.provider.token.store.JwtTokenStore;
16
17 new *
18 @Configuration //Indica que es una clase de configuración
19 @EnableAuthorizationServer //Habilita el servidor de autorización
20 public class ServidorAutorizacion extends AuthorizationServerConfigurerAdapter {
21
22     @Autowired
23     private BCryptPasswordEncoder passwordEncoder; //BCryptPasswordEncoder: es un objeto que nos permite encriptar la
24
25     @Autowired
26     @Qualifier("authenticationManager") //Qualifier: nos permite indicar el nombre del bean que queremos inyectar
27     private AuthenticationManager authenticationManager; //AuthenticationManager: es un objeto que nos permite configu
```


Configuración de servidor de recursos



```
1 package com.company.intecap.apibooks.auth;
2
3 import org.springframework.context.annotation.Configuration;
4 import org.springframework.http.HttpMethod;
5 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
6 import org.springframework.security.oauth2.config.annotation.web.configuration.EnableResourceServer;
7 import org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfigurerAdapter;
8
9 new *
10 @Configuration
11 @EnableResourceServer //Habilita el servidor de recursos
12 public class ConfiguracionRecursos extends ResourceServerConfigurerAdapter {
13
14     new *
15     @Override
16     public void configure(HttpSecurity http) throws Exception {
17         http.authorizeRequests().antMatchers(HttpMethod.GET, "/api/v1/categorias").permitAll() //permitAll: permite a todos
18         .anyRequest().authenticated(); //authenticated: cualquier otra ruta requiere autenticación
19     }
20
21     //copiar el metodo en ConfiguracionSeguridad.java
22     //porque esta configuración es para el servidor de recursos lo maneja OAuth2 y no Spring Security
23 }
```


Crear usuarios y roles en base de datos



```
import ...

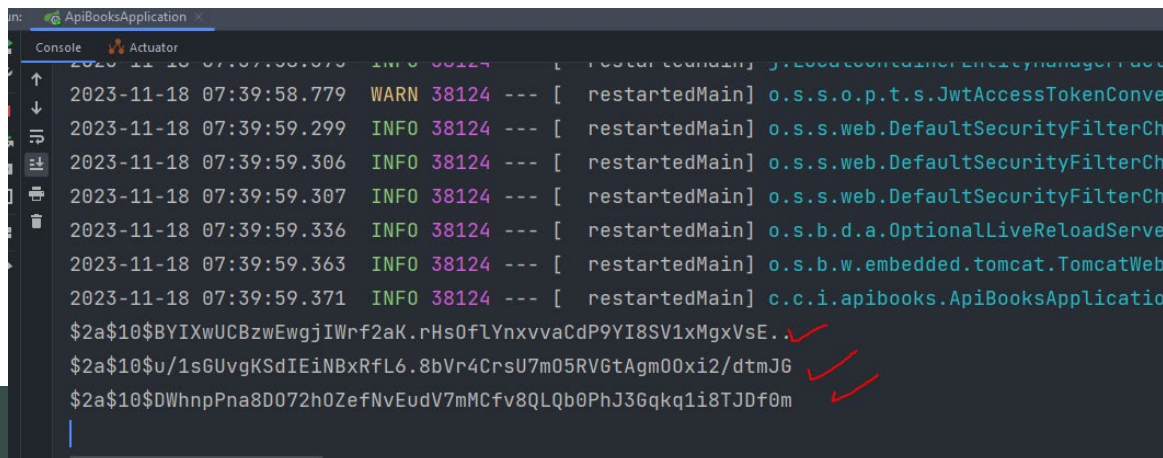
@SpringBootApplication
public class ApiBooksApplication implements CommandLineRunner { //CommandLineRunner: permite ejecutar codigo despues de que se inicie

    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    public static void main(String[] args) {
        SpringApplication.run(ApiBooksApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        String password = "springboot";

        for (int i = 0; i < 3; i++) { //Genera 3 contraseñas encriptadas con BCryptPasswordEncoder
            String passwordBcrypt = passwordEncoder.encode(password);
            System.out.println(passwordBcrypt);
        }
    }
}
```



```
2023-11-18 07:39:58.779 WARN 38124 --- [ restartedMain] o.s.s.o.p.t.s.JwtAccessTokenConve
2023-11-18 07:39:59.299 INFO 38124 --- [ restartedMain] o.s.s.web.DefaultSecurityFilterCh
2023-11-18 07:39:59.306 INFO 38124 --- [ restartedMain] o.s.s.web.DefaultSecurityFilterCh
2023-11-18 07:39:59.307 INFO 38124 --- [ restartedMain] o.s.s.web.DefaultSecurityFilterCh
2023-11-18 07:39:59.336 INFO 38124 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServe
2023-11-18 07:39:59.363 INFO 38124 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWeb
2023-11-18 07:39:59.371 INFO 38124 --- [ restartedMain] c.c.i.apibooks.ApiBooksApplicatio
$2a$10$BYIXwUCBzwEwgjIWrf2aK.rHs0fLYnxvvaCdP9YI8SV1xMgxVsE. ✓
$2a$10$u/1sGUvgKsDIEiNBxRfL6.8bVr4CrS07m05RV6tAgm00xi2/dtmJ6 ✓
$2a$10$DWhnpPna8D072h0ZefNvEudV7mMCfv8QLQb0PhJ3Gqkq1i8TJDf0m ✓
```

\$2a\$10\$NdWC6jYy8JnEmsb4tl.FEeHL97xbUxtsaEv4O4OqihSUD/.ji3xjW
\$2a\$10\$63P5wn0IOcf3OEfIC/rSQed0YjYvBGQ6lqv68yCykxDdf3Fdi0i6S
\$2a\$10\$17tnFtP8Mk4aAVDSx1TxAuvur3CpyhM4vTF9iGeY/rzjFKO1h3sJm

```
SELECT * FROM bdtesting.usuarios;INSERT INTO `bdtesting`.`usuarios`(  
    `habilitado`, `nombre_usuario`,  
    `password`)values(1,"Yoni","$2a$10$NdWC6jYy8JnEmsb4tl.FEeHL97xbU  
xtsaEv4O4OqihSUD/.ji3xjW");INSERT INTO `bdtesting`.`usuarios`(  
    `habilitado`, `nombre_usuario`,  
    `password`)values(1,"Admin","$2a$10$63P5wn0lOCf3OEflC/rSQed0YjYv  
BGQ6lqv68yCykxDdf3Fdi0i6S");
```

test-intecap x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- bd_railway_prueba
- bdtesting**
 - Tables
 - categoria
 - libro
 - roles
 - usuarios**
 - usuarios_roles
 - Views
 - Stored Procedures
 - Functions
- book-db
- db-microservice
- db_agencia_viajes
- db_agencia_viajes2
- db_agenda
- db_book
- db_books
- db_login
- db_tarea4
- db_tasks
- db_tienda
- dbestudiantes
- escuela_final
- inventario
- nevbd
- railway
- sys

usuarios

Limit to 1000 rows

```
1 • SELECT * FROM bdtesting.usuarios;
2
3 • INSERT INTO `bdtesting`.`usuarios`
4   (
5     `habilitado`,
6     `nombre_usuario`,
7     `password`
8   )
9   values
10  (1,
11   "Yoni",
12   "$2a$10$NdWC6jYy8JnEmsb4t1.FEeHL97xbUxtsaEv4040qihSUD/.ji3xjW"
13  );
14
15 • INSERT INTO `bdtesting`.`usuarios`
16   (
17     `habilitado`,
18     `nombre_usuario`,
19     `password`
20   )
21   values
22   (1,
23    "Admin",
24    "$2a$10$63P5wn01OCf30EfIC/rSQed0YjYvBGQ6Iqv68yCykxDdF3Fdi0i6S"
25  );
```

test-intecap x

FileEditViewQueryDatabaseServerToolsScriptingHelp

SQL

SQL

+

+

+

+

+

+

+

+

+

+

Navigator

SCHEMAS

Filter objects

bd_railway_prueba

bdtesting

Tables

categoria

libro

roles

usuarios

usuarios_rols

Views

Stored Procedures

Functions

book-db

db-microservice

db_agencia_viajes

db_agencia_viajes2

db_agenda

db_book

db_books

db_login

db_tarea4

db_tasks

db_tienda

dbestudiantes

escuela_final

inventario

nevbd

railway

usuarios

usuarios

Limit to 1000 rows

1

SELECT * FROM bdtesting.usuarios;

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	habilitado	nombre_usuario	password
	1	1	Yoni	\$2a\$10\$NdWC6jYy8JnEmsb4tl.FEeHL97xbUxtsaEv4O4QqihSUD/.ji3xjW
	2	1	Admin	\$2a\$10\$63P5wn0LOCf3OeflC/rSQed0YjYvBGQ6Iqv68yCykxDdf3Fdi0i6S
*	NULL	NULL	NULL	NULL

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

db_books

Tables

categorias

libros

roles

usuarios

usuarios

Views

Stored F

Function

db_invento

sys

Administration - Startup / Shutdo...

SQL File 6*

usuarios x

Limit to 1000 rows

1 • SELECT * FROM db_books.usuarios;

Select Rows - Limit 1000

Table Inspector

Copy to Clipboard

Table Data Export Wizard

Table Data Import Wizard

Send to SQL Editor

Create Table...

Create Table Like...

Alter Table...

Name (short)

Name (long)

Select All Statement

Insert Statement

io password

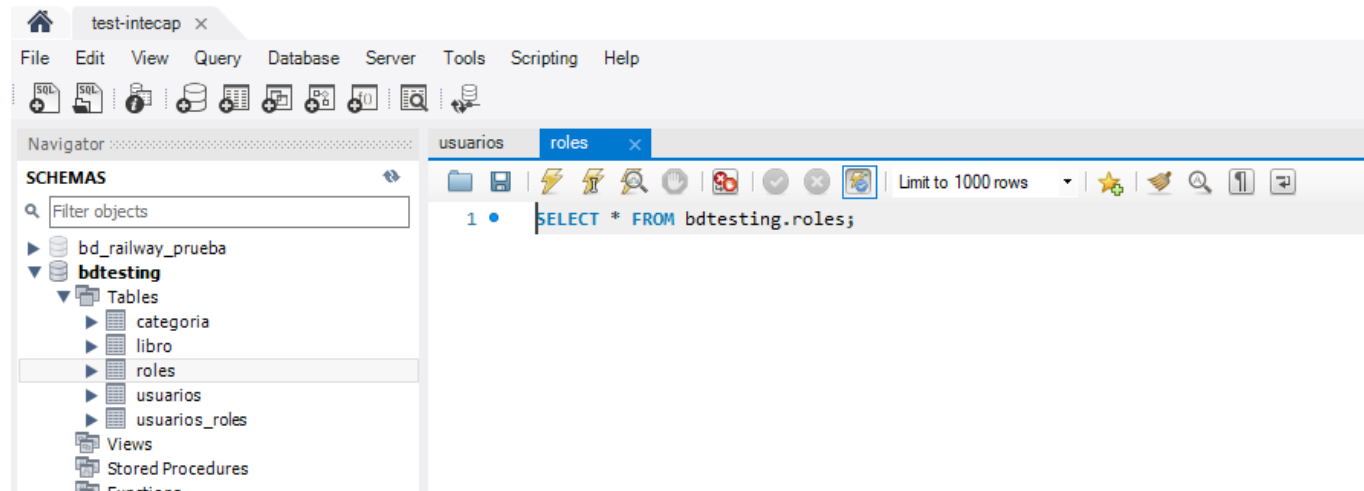
\$2a\$10\$aaUKhtrTyGvfVmvihwqnP.NnUQMEKoh...

\$2a\$10\$faeOSEAK5R7rKi1DHu18sO5s33t9UdA...

NULL

Edit: Export/Import:

```
INSERT INTO `bdtesting`.`roles`(`nombre`)VALUES("ROLE_USER");INSERT INTO  
`bdtesting`.`roles`(`nombre`)VALUES("ROLE_ADMIN");
```



```
INSERT INTO `bdtesting`.`usuarios_roles`(`usuario_id`,`role_id`)VALUES(1,1);INSERT INTO
`bdtesting`.`usuarios_roles`(`usuario_id`,`role_id`)VALUES(2,1);INSERT INTO
`bdtesting`.`usuarios_roles`(`usuario_id`,`role_id`)VALUES(2,2);
```

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases. The 'bdtesting' database is selected, and its 'usuarios_roles' table is highlighted. Below this, the 'Information' pane shows the table's structure: 'usuario_id' (bigint PK) and 'role_id' (bigint PK). The main editor on the right contains a SQL script with the following queries:

```
1 • SELECT * FROM bdtesting.usuarios;
2
3 • INSERT INTO `bdtesting`.`usuarios`
4 (
5     `habilitado`,
6     `nombre_usuario`,
7     `password`
8 )
9 values
10 (1,
11     "Yoni",
12     "$2a$10$gojDKHSvUUI1qdBcCNyhpUXgBhhXfw7nRiD8T81hSqL78i3Us/V0"
13 );
14
15 • INSERT INTO `bdtesting`.`usuarios`
16 (
17     `habilitado`,
18     `nombre_usuario`,
19     `password`
20 )
21 values
22 (1,
23     "Admin",
24     "$2a$10$q3II/qmmJmQFxfv6n1lfqu.xtK0Pd/nrTAVq33CfZDw/jTMJ4EUfcG"
25 );
26
27
28 • INSERT INTO `bdtesting`.`roles`
29 (`nombre`)
30 VALUES
31 ("ROLE_USER");
32
33 • INSERT INTO `bdtesting`.`roles`
34 (`nombre`)
35 VALUES
36 ("ROLE_ADMIN");
37
```

MySQL Workbench

test-intecap x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

bd_railway_prueba

bdtesting

Tables

categoria

libro

roles

Columns

Indexes

Foreign Keys

Triggers

usuarios

usuarios_roles

Columns

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

book-db

db-microservice

db_agencia_viajes

db_agencia_viajes2

db_agenda

db_book

db_books

db_login

db_tarea4

db_tasks

db_tienda

dbestudiantes

Administration Schemas

Information

Table: usuarios_roles

Columns:

usuario_id bigint PK

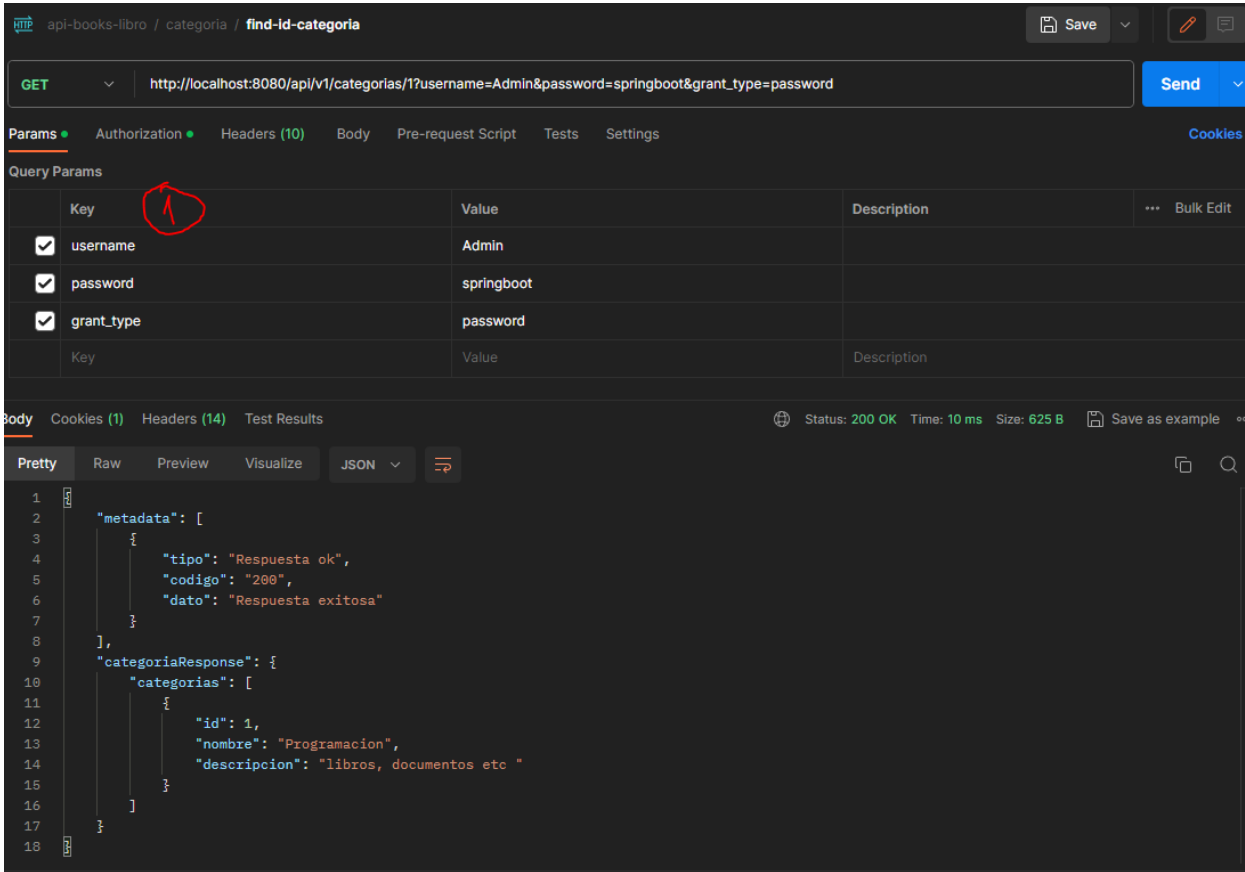
role_id bigint PK

usuarios roles

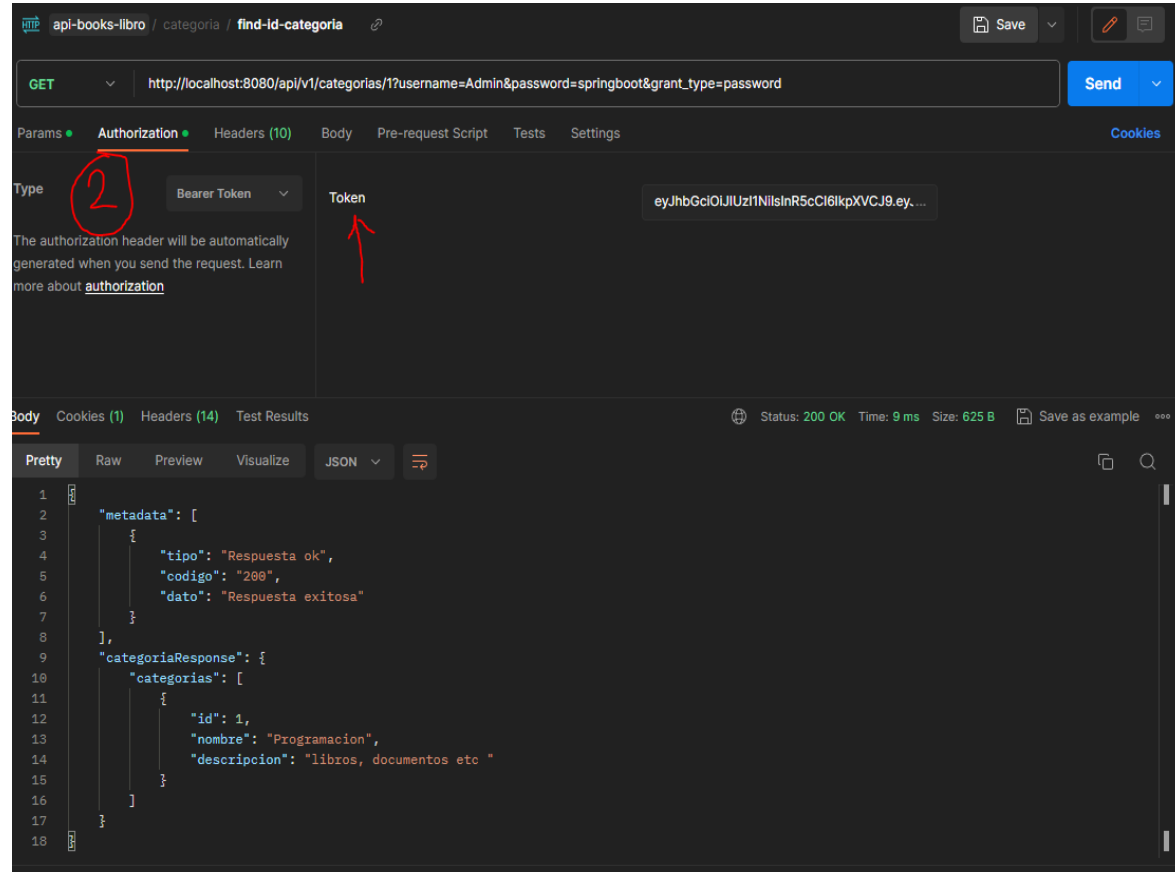
Limit to 1000 rows

```
1 • INSERT INTO `bdtesting`.`usuarios_roles`
2   (`usuario_id`,
3    `role_id`)
4   VALUES
5   (1,
6    1);
7
8 • INSERT INTO `bdtesting`.`usuarios_roles`
9   (`usuario_id`,
10  `role_id`)
11  VALUES
12  (2,
13   1);
14
15 • INSERT INTO `bdtesting`.`usuarios_roles`
16   (`usuario_id`,
17   `role_id`)
18  VALUES
19  (2,
20   2);
21
```


Colocar los parámetros de accesos Login.



Pegar el token generado



```
"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJleHAiOjE3MDAzMzA1NzQsInZjaWFnZWZ0eSUiOiJkfkBwLlUiwiYXV0aG9yaXRpZXMiOiJsUk9MRVBRE1TjIiIsIlJPTFVEfVFNFuIjDlCjQdGk1OiJmEmZMTc5My01MDgyLT  
Q4MTYyOWM2YXNjbGlkcy9nYnltYTA1LjCybnRfaWQiOiJjbGllbnRlcmVudCIsInNjb3JlIjpbInJlYWQlLCJ3cm10ZSIdfQ.  
pcfpqyUWKkkyKsVxxFXIN-ARQW-FAEGPSYajyV-JFYU4"  
"token_type": "bearer",  
"refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
```