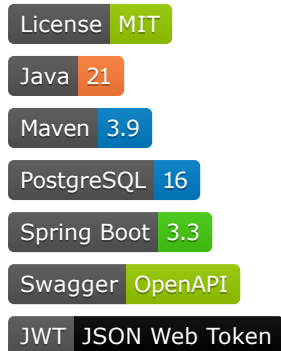


Escuela Colombiana de Ingeniería Julio Garavito

Arquitectura de Software – ARSW



Estudiantes:

- [Alexandra Moreno](#)
- [Alison Valderrama](#)
- [Jeisson Sánchez](#)
- [Valentina Gutierrez](#)

Requisitos

- **JDK 21**
- **Maven 3.9+**
- **Git**

Ejecución del proyecto

1. Clonar o descomprimir el proyecto:

```
git clone https://github.com/arsw-la-ley-del-corazon/Lab5-BluePrintsApi_with_JWT.git
```

ó si el profesor entrega el **.zip**, descomprimirlo y entrar en la carpeta.

2. Ejecutar con Maven:

```
mvn spring-boot:run
```

3. Verificar que la aplicación levante en **http://localhost:8080**.

Actividades propuestas

1. Revisar el código de configuración de seguridad (`SecurityConfig`) e identificar cómo se definen los endpoints públicos y protegidos.

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .csrf(csrf -> csrf.disable())
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/actuator/health", "/auth/login").permitAll()
            .requestMatchers("/v3/api-docs/**", "/swagger-ui/**", "/swagger-ui.html").permitAll()
            .requestMatchers("/api/**").hasAnyAuthority("SCOPE_blueprints.read", "SCOPE_blueprints.write")
            .anyRequest().authenticated()
        )
        .oauth2ResourceServer(oauth2 -> oauth2.jwt(Customizer.withDefaults()));
    return http.build();
}
```

- **PUBLICO**

`"/actuator/health", "/auth/login", "/v3/api-docs/", "/swagger-ui/", "/swagger-ui.html".`

La etiqueta `permitAll()` nos permite declarar excepciones de seguridad.

- **PROTEGIDO**

`/api/.hasAnyAuthority("SCOPE_blueprints.read", "SCOPE_blueprints.write")`: Esto nos indica que esta `/api/` está protegido y requiere permisos, ya que obliga a que el token JWT contenga al menos una de esas autoridades.

El `anyRequest().authenticated()` nos obliga a que haya un usuario autenticado, es decir, que el request traiga un token JWT válido.

2. Explorar el flujo de login y analizar las claims del JWT emitido.

En Swagger UI, ejecutamos el endpoint público POST `/auth/login` con las credenciales de ejemplo:

Username : student

Password : student123

auth-controller

POST /auth/login

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{
  "username": "student",
  "password": "student123"
}
```

Execute

Clear

Esto nos genera un objeto JSON con el token JWT:

Code

Details

200

Response body

```
{
  "access_token": "eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodWwzovL2RlY3Npcy11Y2kvYmx1ZXBhY50cyIsInYiOiJ6InW0dWlbnQlLCJleHAiOiJlZ3NTgXNjcyMjIsIm1hdCT6MTc1ODF2MzY2Miwic2NvcGUiOiJibWV1chJpbmRzLnJlYWQgYmxiZXBhY50cy53cm10ZS79.dNognK3ZhrInfKq1iZfz-10CPKJH-8Ji8i10qbsEFheZ586TRS6qo8r2eRtWQtp9Pm5_71fBMNkXdiZUtl1C4eJXG6-rinVNL CGS28YB601Vd8TphwOn0bxdHRlQErkkA7f637WV4XaQ_F06pbrLVd7Ffu5E3p9_1UM6SVqMLgHfx2Hj0S8ySm-ndSYtfzC-g0H6QnduCNXX29sc59Sw3tEwh7ae1JLKeGpnXkuzej8p54AasbMD83ayjH5mF5APbnbc061AGjaARBRUY8-092d9SAGaiWuomG1QizAuNk_yXrS6Rd7q7QmqndVelo05vLiAt9xa9amAcLpTbg",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Download

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-type: application/json
date: Thu, 18 Sep 2025 02:47:42 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
transfer-encoding: chunked
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
```

Copiamos el valor de access_token en la opción Authorize de Swagger

Available authorizations

X

bearer-jwt (http, Bearer)

Value:

eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodWwzovL2RlY3Npcy11Y2kvYmx1ZXBhY50cyIsInYiOiJ6InW0dWlbnQlLCJleHAiOiJlZ3NTgXNjcyMjIsIm1hdCT6MTc1ODF2MzY2Miwic2NvcGUiOiJibWV1chJpbmRzLnJlYWQgYmxiZXBhY50cy53cm10ZS79.dNognK3ZhrInfKq1iZfz-10CPKJH-8Ji8i10qbsEFheZ586TRS6qo8r2eRtWQtp9Pm5_71fBMNkXdiZUtl1C4eJXG6-rinVNL CGS28YB601Vd8TphwOn0bxdHRlQErkkA7f637WV4XaQ_F06pbrLVd7Ffu5E3p9_1UM6SVqMLgHfx2Hj0S8ySm-ndSYtfzC-g0H6QnduCNXX29sc59Sw3tEwh7ae1JLKeGpnXkuzej8p54AasbMD83ayjH5mF5APbnbc061AGjaARBRUY8-092d9SAGaiWuomG1QizAuNk_yXrS6Rd7q7QmqndVelo05vLiAt9xa9amAcLpTbg

Authorize

Close

Una vez logrado, podemos invocar los endpoints protegidos por la API.

PROFESSEUR : M.DA ROS

◆ 4 / 8 ◆

BTS SIO BORDEAUX - LYCÉE GUSTAVE EIFFEL

Parameters

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/api/blueprints' \
-H 'accept: */*' \
-H 'Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczovL2RlY3Npcy11Y2kvYmx1ZXByaW50cyIsInN1YiI6InN0dWRLbnRqIlC1eHAiojE3NTgyNDQ0MzIsImh0dCI6MTc1ODI0MDgzMiwiY2NvcGUiOiJibHV1chJpbmRzLnJlYmQyYmx1ZX'

```

Request URL

```
http://localhost:8080/api/blueprints
```

Server response

Code

Details

200

Response body

```
[
  {
    "name": "Casa de campo",
    "id": "b1"
  },
  {
    "name": "Edificio urbano",
    "id": "b2"
  }
]
```

Download

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-type: application/json
date: Fri, 19 Sep 2025 00:20:29 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
transfer-encoding: chunked
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
```

Para analizar los claims del JWT emitido, hacemos uso de jwt.io, el cual descodifica el `access_token`. Nos otorga las siguientes claims:

JSON CLAIMS TABLE		HIDE DETAILS ↗
iss	https://decsis-eci/blueprints	The issuer of the JWT. Learn more
sub	student	The subject of the JWT (the user). Learn more
exp	1758244627 (Thu Sep 18 2025 20:17:07 GMT-0500 (hora estándar de Colombia))	The expiration time on or after which the JWT MUST NOT be accepted for processing. Learn more
This value must be a NumericDate type, representing seconds.		
iat	1758241027 (Thu Sep 18 2025 19:17:07 GMT-0500 (hora estándar de Colombia))	The time at which the JWT was issued. Learn more
This value must be a NumericDate type, representing seconds.		
scope	blueprints.read blueprints.write	

- **Iss:** Identifica el emisor del token, en este caso nuestra API https://decsis-eci/blueprints.
- **Sub:** Identifica el usuario autenticado que solicito el token, en este caso student.
- **Exp:** Es el tiempo de expiracion del token, en este caso Thu Sep 18 2025 20:17:07 GMT-0500.
- **Iat:** Se refiere al momento exacto en el que el token fue emitido: Thu Sep 18 2025 19:17:07 GMT-0500.
- **Scope:** Define los permisos del usuario dentro del sistema, en este caso SCOPE_blueprints.read y SCOPE_blueprints.write.

3. Extender los scopes (`blueprints.read`, `blueprints.write`) para controlar otros endpoints de la API, del laboratorio P1 trabajado.

Para este punto importamos el proyecto trabajado en el lab P1 y le aplicamos la seguridad a los endpoints, para eso usamos `@PreAuthorize("hasAuthority('SCOPE_xxx')")`, esto nos dice que el token JWT solo funciona si incluye el scope, ya sea read o write dependiendo de la solicitud del endpoint.

```
// GET /blueprints/{author}
@GetMapping("/{author}")
@PreAuthorize("hasAuthority('SCOPE_blueprints.read')")
@Operation(
    summary = "Listar blueprints por autor",
    description = "Obtiene todos los blueprints creados por un autor específico. Requiere permisos de lectura.",
    security = @SecurityRequirement(name = "bearer-jwt"),
    responses = {
        @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "200", description = "Consulta exitosa",
            content = @Content(schema = @Schema(implementation = ApiResponse.class))),
        @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "404", description = "Autor sin blueprints",
            content = @Content(schema = @Schema(implementation = ApiResponse.class))),
        @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "401", description = "No autorizado - Token JWT requerido",
            content = @Content(schema = @Schema(implementation = ApiResponse.class))),
        @io.swagger.v3.oas.annotations.responses.ApiResponse(responseCode = "403", description = "Prohibido - Permisos insuficientes",
            content = @Content(schema = @Schema(implementation = ApiResponse.class)))
    }
))
```

4. Modificar el tiempo de expiración del token y observar el efecto.

RTA

En el código en el archivo `AuthController.java` se modificó la línea 37 de:

```
Instant exp = now.plusSeconds(60);
```

a:

```
Instant exp = now.plusSeconds(ttl);
```

Para poder comprobar cómo se comporta el sistema al expirar el token, no se validaba si el token expiraba, así que se desarrolló las debidas clases y métodos que permitieran validar el token en cada petición, para esto se creó la clase `JwtValidationFilter.java` y se configuró en el archivo `SecurityConfig.java` para que se ejecutara antes de cada petición.:

 Test Jwt Validations

5. Documentar en Swagger los endpoints de autenticación y de negocio.

 Swagger

 Swagger

Se puede evidenciar todo los endpoint documentados con autenticación:

Swagger 

Haz clic en el badge para abrir la documentación.

