LAB 14: QUARKUS SECURE JWT

Autor: José Díaz

Github Repo: https://github.com/joedayz/quarkus-bcp-2025.git

1. Abre el proyecto **11-secure-jwt-start**.
2. Revisa la clase **JwtResource**. El endpoint **/jwt/{username}** genera un JWT para un usuario dado.

   IMPORTANTE:

   Este endpoint no requiere un password por simplicidad. En producción, no uses JWTs sin autenticación.

3. Revisa la clase **UserResource**. El endpoint **/user/expenses** retorna los expenses del usuario autenticado.
4. Revisa la clase **AdminResource**. El endpoint **/admin/expenses** lista todos los expenses. Solo usuarios con el role **ADMIN** deberían tener permisos para usar este endpoint.
5. Revisa el archivo **/src/main/resources/application.properties**. El archivo configura las propiedades requeridas para construir JWTs usando el SmallRye JWT generation API.

Windows:

```
mp.jwt.verify.issuer = https://example.com/redhattraining

smallrye.jwt.sign.key.location = C:/Users/Amadeo/DO378/secure-jwt/privateKey.pem
mp.jwt.verify.publickey.location = C:/Users/Amadeo/DO378/secure-jwt/publicKey.pem
```

Linux o MacOSX:

```
mp.jwt.verify.issuer = https://example.com/redhattraining

smallrye.jwt.sign.key.location = ${HOME}/DO378/secure-jwt/privateKey.pem
mp.jwt.verify.publickey.location = ${HOME}/DO378/secure-jwt/publicKey.pem
```

IMPORTANTE: Ejecuta la clase **GenerateKeys.java** y verifica que dichos archivos existen en la ruta indicada. Asegurate que ambos archivos .pem existan, sino, no podrás continuar este laboratorio.

6. Revisa la clase **JWTGeneratorTest**. Este test verifica los grupos asignados a los JWTs de usuarios regulares y administradores. **Usuarios regulares** deben pertenecer al grupo **USER**. Los **administradores** deben pertenecer a los grupos **USER** y **ADMIN**. Esta clase también verifica que los JWTs para usuarios regulares incluyen los claims: **iss, sub, upn, aud**, y **locale**.
7. Revisa la clase **UserResourceTest**. Esta clase test verifica que el endpoint **/user/expenses** es seguro. Usuarios no autenticados no deben poder acceder al endpoint **/user/expenses**. Los usuarios regulares autenticados deben poder listar sus propios expenses.
8. Revisa la clase **AdminResourceTest**. Esta clase test verifica que el endpoint **/admin/expenses** es seguro. Usuarios no autenticados y usuarios regulares no deben poder acceder al endpoint **/admin/expenses**. Los usuarios administradores autenticados deben poder listar todos los expenses.
9. Ejecutar todos los tests y verificar que nueve de ellos fallan.

```
[student@workstation secure-jwt]$ mvn test
...output omitted...
[ERROR]  AdminResourceTest.guestsCannotListExpenses:26 1 expectation failed.
Expected status code <401> but was <200>.

[ERROR]  AdminResourceTest.regularUsersCannotListExpenses:39 1 expectation failed.
Expected status code <403> but was <200>.

[ERROR]  UserResourceTest.guestsCannotListExpenses:26 1 expectation failed.
Expected status code <401> but was <200>.

[ERROR]  JwtGeneratorTest.adminJwtBelongsToAdminGroup:83 JWT groups for admin do
 not contain ADMIN ...
[ERROR]  JwtGeneratorTest.adminJwtBelongsToUserGroup:74 JWT groups for admin do
 not contain USER ...
[ERROR]  JwtGeneratorTest.userJwtBelongsToUserGroup:29 JWT groups for regular user
 do not contain USER ...
[ERROR]  JwtGeneratorTest.userJwtContainsAudienceClaim:74 JWT 'aud' claim not set
 as expected ...
[ERROR]  JwtGeneratorTest.userJwtContainsLocaleClaim:65 JWT 'locale' claim not set
 as expected ...
[ERROR]  JwtGeneratorTest.userJwtContainsSubjectClaim:38 JWT 'sub' claim not set
 as expected ...

[ERROR] Tests run: 13, Failures: 9, Errors: 0, Skipped: 0
...output omitted...
```

10. Completar el código para crear JWTs que contengan los claims requeridos.
    a. Abre la clase **JwtGenerator** y modifica el método **generateJwtForRegularUser**. Este método genera JWTs para

usuarios regulares. Agrega los claims **sub, aud, y locale**, de la siguiente manera:

```
public static String generateJwtForRegularUser( String username ) {
    return Jwt.issuer( ISSUER )
            .upn( username + "@example.com" )
            .subject(username)
            .audience("expenses.example.com")
            .claim("locale", "en_US")
            .sign();
}
```

b. Completa el método **generateJwtForRegularUser** agregando un grupo para los usuarios regulares. Establece el claim groups para que contenga el grupo USER.

```
public static String generateJwtForRegularUser( String username ) {
    return Jwt.issuer( ISSUER )
            .upn( username + "@example.com" )
            .subject(username)
            .audience("expenses.example.com")
            .claim("locale", "en_US")
            .groups(new HashSet<>(List.of("USER")))
            .sign();
}
```

c. En la misma clase completa el método **generateJwtForAdmin**. Este método genera JWT para administradores. Establece el claim groups para que contenga los grupos **USER** y **ADMIN**.

```
public static String generateJwtForAdmin( String username ) {
    return Jwt.issuer( ISSUER )
            .upn( username + "@example.com" )
            .subject( username )
            .claim( "locale", "en_US" )
            .groups(new HashSet<>(List.of("USER","ADMIN")))
            .sign();
}
```

d. Ejecuta los tests en la clase **JwtGeneratorTest** y verifica que 8 tests han pasado

mvn test -Dtest=JwtGeneratorTest

```
[student@workstation secure-jwt]$ mvn test -Dtest="JwtGeneratorTest"
...output omitted...
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
...output omitted...
```

11. Asegura el endpoint **/user/expenses**. Permitir el acceso sólo a los que tengan el role USER.

a. Anotar la clase **UserResource** para asegurar sus endpoints. Restringir el acceso sólo al role **USER**.

```
@Path( "/user" )
@RolesAllowed({"USER"})
public class UserResource {…}
```

b. Ejecuta los tests en la clase UserResourceTest y verifica que los dos tests pasen.

mvn test -Dtest=UserResourceTest

```
[student@workstation secure-jwt]$ mvn test -Dtest="UserResourceTest"
...output omitted...
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
...output omitted...
```

12. Asegura el endpoint **/admin/expenses**. Permite el acceso sólo al role **ADMIN**.

    a. Anota la clase **AdminResource** y asegurar sus endpoints. Restringe el acceso al role **ADMIN**.

```
@Path( "/admin" )
@RolesAllowed("ADMIN")
public class AdminResource {…}
```

    b. Ejecuta los tests en la clase **AdminResourceTest** y verifica que los tres tests pasen.

    mvn test -Dtest=AdminResourceTest

```
[student@workstation secure-jwt]$ mvn test -Dtest="AdminResourceTest"
...output omitted...
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
...output omitted...
```

13. Verifica que todos los tests pasen.

    mvn test

```
[student@workstation secure-jwt]$ mvn test
...output omitted...
[INFO] Tests run: 13, Failures: 0, Errors: 0, Skipped: 0
...output omitted...
```

Felicitaciones, hemos terminado el laboratorio.

Enjoy!

José