OAuth2: Tokens en base de datos

Implementa la seguridad de tu API Rest con Spring Boot

Hasta ahora

- Nuestros clientes se han almacenado en memoria

```
public void configure(ClientDetailsServiceConfigurer clients) {
        clients
                .inMemory()
```

- También nuestros tokens
  - Por defecto, se configura en memoria (*InMemoryTokenStore*)

¿Y si queremos almacenar en base de datos?

- Necesitamos proporcionar un esquema para
  - clientes
  - tokens
  - códigos de autorización
  - ...
- Tenemos uno en el repositorio oficial en github:

  https://github.com/spring-projects/spring-security-oauth/blob/master/spring-security-oauth2/src/test/resources/schema.sql

import.sql

- Si queremos crear algunas tablas a través de un script DDL, y mantener la creación automática de las tablas asociadas a entidades y asociaciones, podemos crear un fichero llamado **import.sql**.
- En producción
  - Posiblemente utilicemos ddl-auto=none
  - También algún sistema de migración de versiones de bases de datos, como *Liquibase* o *Flyway*

Configuración de H2

- Para tener una base de datos persistente.
- También para poder consultar a través de la consola.
    - Puntualmente, configuramos la seguridad para poder acceder a la consola.

Servidor de autorización

- Inyectamos el datasource (configurado vía *properties*).
- Configuramos los clientes a través de JDBC

```
public void configure(ClientDetailsServiceConfigurer clients) {
                clients.jdbc(dataSource) ...
```

- Creamos un TokenStore almacenado a través de JDBC, y lo configuramos.

```
@Bean
public TokenStore tokenStore() {
     return new JdbcTokenStore(dataSource);
}
```

Ejecución

- Podemos probar a solicitar un token con POSTMAN.
- Si el mismo usuario vuelve a loguearse, le devuelve el mismo token.
- Podemos ejecutar la consola de H2 y verificar que los tokens están allí.