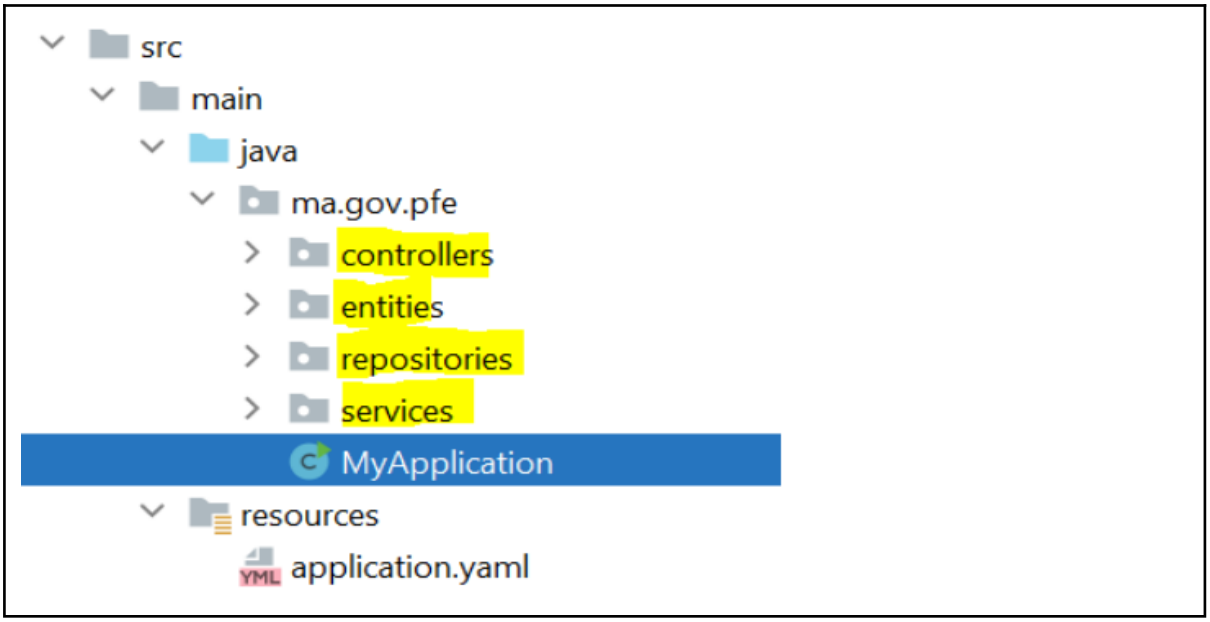
Travaux Pratiques Pour les séances

du 01/12/2024 => 19/01/2025

A. Création composants User Entity

1. Vérifier la structure de votre projet et la présence des packages : controllers, services, repositories, entities



2. Vérifier le pom.xml de votre projet, s'il contient toutes les dépendances

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>ma.gov.pfe</groupId>
    <artifactId>pfe</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
```

```
        <version>2.7.18</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.26</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5.1</version>
            </plugin>
        </plugins>
    </build>

    <repositories>
        <repository>
            <id>central</id>
            <url>https://repo1.maven.org/maven2/</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
```

```
</project>
```

3. Vérifier que le fichier de configuration application.yml de votre projet contient :

```yaml
server:
 port: 9090

spring:
 datasource:
   driver-class-name: org.h2.Driver
   url: jdbc:h2:file:~/pfe1
   serverName: localhost
   username: sa
   password: password

 jpa:
   show_sql: true
   generate-ddl: true
   hibernate:
     ddl-auto: create

 h2:
   console:
     enabled: true
     path: /h2-console
```

4. Dans la package “entities”, créer l’entité User Entity suivante:

```java
package ma.gov.pfe.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "T_USERS")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class UserEntity {
   @Id
   private Long id;
   private String name;
   private String login;
```

```
    private String password;
}
```

5. Dans la package “repositories”, créer le repository pour User Entity:

```
package ma.gov.pfe.repositories;

import ma.gov.pfe.entities.UserEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends
JpaRepository<UserEntity, Long> {
}
```

Noter que l’interface Repository doit :

- Étendre l’interface **JpaRepository** de Spring data.
- Être annoté par **@Repository**

6. Démarrer L’application à partir de la classe de démarrage MyApplication.java

```
package ma.gov.pfe;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MyApplication {
    public static void main(String[] args) {
        SpringApplication.run(MyApplication.class);
    }
}
```

Noter que la classe de démarrage:

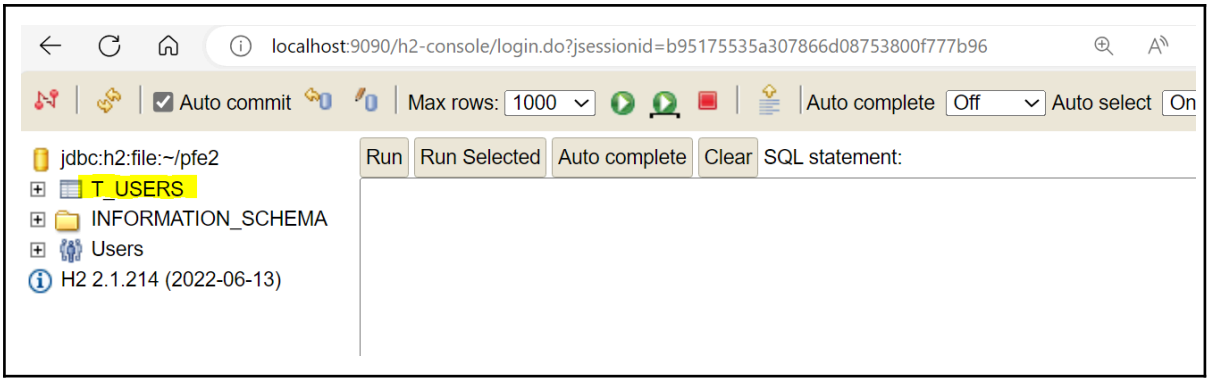
- Contenir la méthode main
- Être annoté par **@SpringBootApplication**

7. Vérifier les logs de demmarage, surtout les parties en jaune ci-dessous.

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.18)

2025-01-18 17:34:15.439  INFO 18000 --- [           main] ma.gov.pfe.MyApplication                 : Starting MyApplication using Jav
2025-01-18 17:34:15.444  INFO 18000 --- [           main] ma.gov.pfe.MyApplication                 : No active profile set, falling b
2025-01-18 17:34:16.417  INFO 18000 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA re
2025-01-18 17:34:16.488  INFO 18000 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository
2025-01-18 17:34:18.242  INFO 18000 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s):
2025-01-18 17:34:18.257  INFO 18000 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-01-18 17:34:18.257  INFO 18000 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache
2025-01-18 17:34:18.419  INFO 18000 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded Web
2025-01-18 17:34:18.419  INFO 18000 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: init
2025-01-18 17:34:16.417  INFO 18000 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA rep
2025-01-18 17:34:16.488  INFO 18000 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository s
2025-01-18 17:34:18.242  INFO 18000 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s):
2025-01-18 17:34:18.257  INFO 18000 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-01-18 17:34:18.257  INFO 18000 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache
2025-01-18 17:34:18.419  INFO 18000 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebA
2025-01-18 17:34:18.419  INFO 18000 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initi
2025-01-18 17:34:18.461  INFO 18000 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-01-18 17:34:19.051  INFO 18000 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-01-18 17:34:19.077  INFO 18000 --- [           main] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-cons
2025-01-18 17:34:19.329  INFO 18000 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing Persistence
2025-01-18 17:34:19.424  INFO 18000 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core ver
2025-01-18 17:34:19.720  INFO 18000 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons An
2025-01-18 17:34:19.910  INFO 18000 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hib
Hibernate: drop table if exists t_users CASCADE
Hibernate: create table t_users (id bigint not null, login varchar(255), name varchar(255), password varchar(255), primary key (id))
2025-01-18 17:34:20.887  INFO 18000 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform impl
2025-01-18 17:34:20.902  INFO 18000 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFact
2025-01-18 17:34:20.964  WARN 18000 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enable
2025-01-18 17:34:22.036  INFO 18000 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9090 (
2025-01-18 17:34:22.052  INFO 18000 --- [           main] ma.gov.pfe.MyApplication                 : Started MyApplication in 7.316 se
```

8. Vérifier La création de la table Users au niveau de la base H2

Accéder à l'URL : http://localhost:9090/h2-console



9. Créer un nouveau package "dtos", dans votre projet. Dans ce package créer la classe "UserDto" suivante:

```java
package ma.gov.pfe.dtos;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
```

```java
@AllArgsConstructor
@NoArgsConstructor
public class UserDto {
   private Long id;
   private String name;
   private String login;
   private String password;
}
```

10. Créer un nouveau package “mappers” pour faire les conversions Dto vers Entiry ou Entity vers Dto.

Nous allons utiliser la dependency MapStruct pour faire ces conversions. Ajouter la dépendance sur le pom.xml.

```xml
<dependency>
 <groupId>org.mapstruct</groupId>
 <artifactId>mapstruct</artifactId>
 <version>1.5.3.Final</version>
</dependency>
```

Modifier `maven-compiler-plugin` dans votre pom.xml pour faire fonctionner mapstruct et lombok.

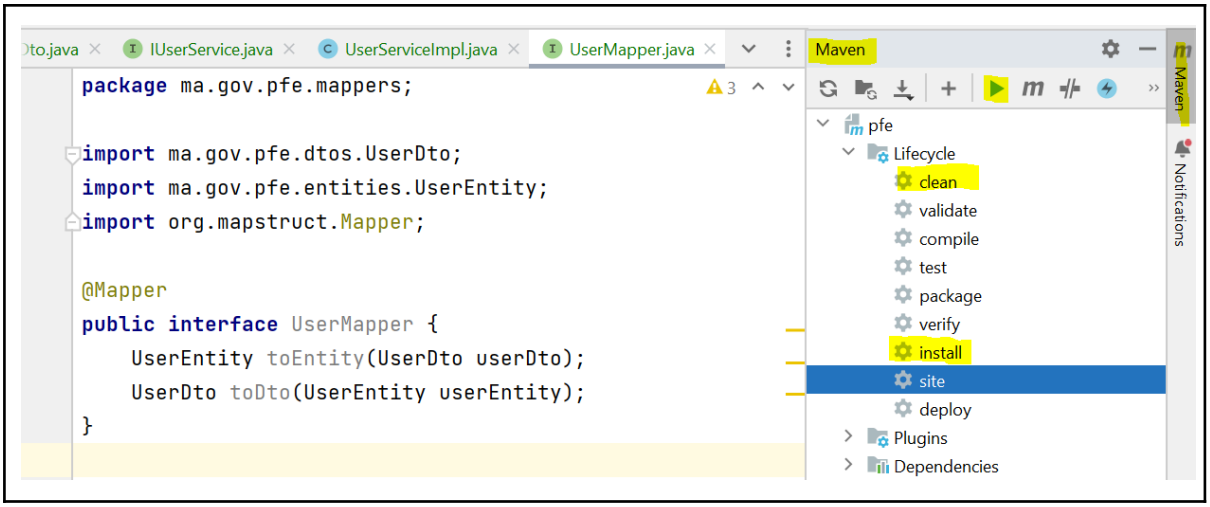
```xml
<plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <version>3.5.1</version>
 <configuration>
   <source>1.8</source>
   <target>1.8</target>
   <annotationProcessorPaths>
     <path>
       <groupId>org.mapstruct</groupId>
       <artifactId>mapstruct-processor</artifactId>
       <version>1.5.3.Final</version>
     </path>
     <path>
       <groupId>org.projectlombok</groupId>
       <artifactId>lombok</artifactId>
       <version>1.18.26</version>
     </path>
     <!-- This is needed when using Lombok 1.18.16 and above
-->
```

```
      <path>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok-mapstruct-binding</artifactId>
        <version>0.2.0</version>
      </path>
    </annotationProcessorPaths>
  </configuration>
</plugin>
```

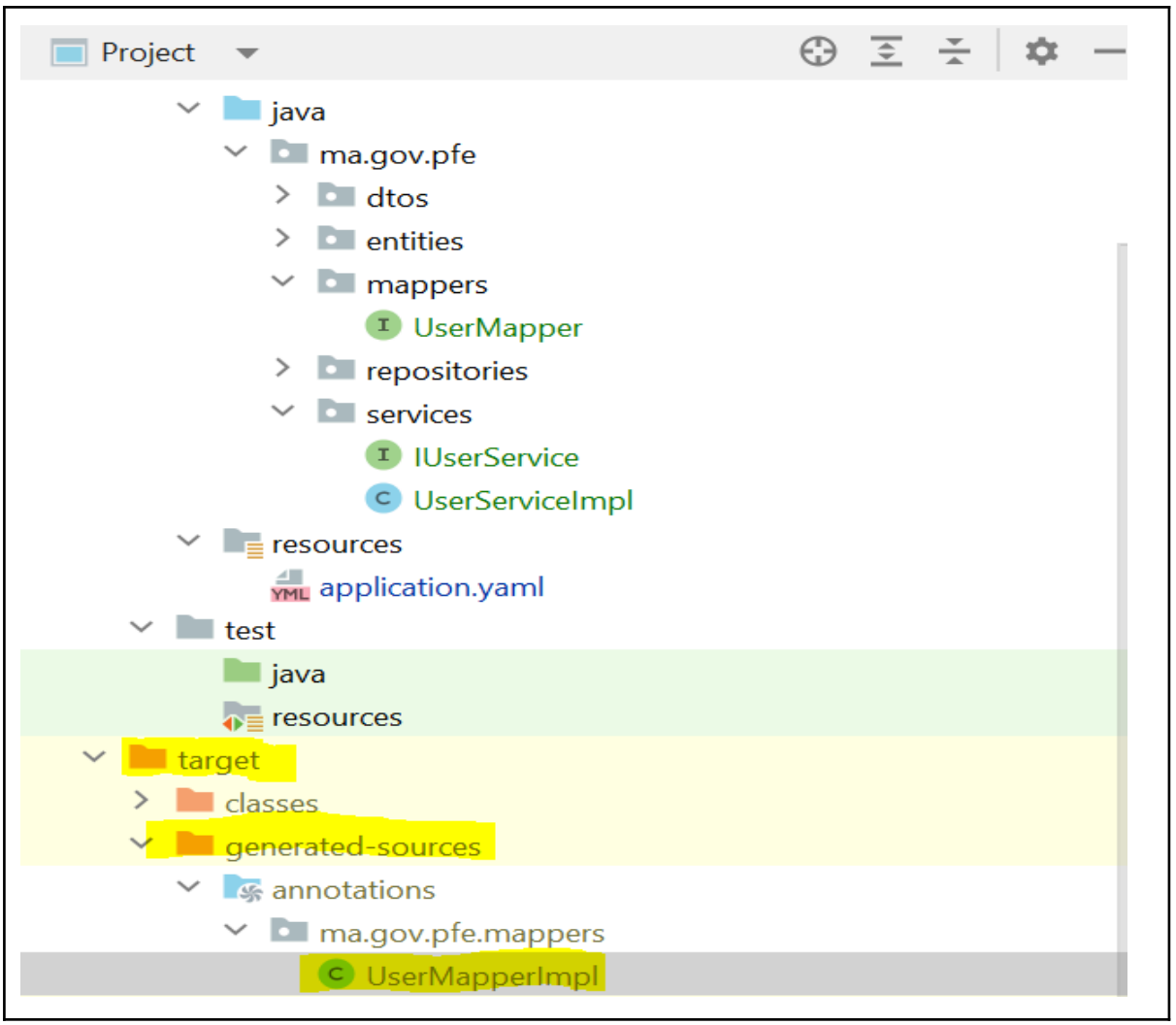
11. Dans le package “mappers”, Créer L'interface “UserMapper” suivante:

```
package ma.gov.pfe.mappers;

import ma.gov.pfe.dtos.UserDto;
import ma.gov.pfe.entities.UserEntity;
import org.mapstruct.Mapper;

@Mapper(componentModel = "spring")
public interface UserMapper {
    UserEntity toEntity(UserDto userDto);
    UserDto toDto(UserEntity userEntity);
}
```

12. Faire un clean -> install



13. Vérifier le dossier target de votre projet si l'implémentation de l’interface UserMappper:



14. Dans le package “services”, créer l’interface suivante IUserService suivante:

```java
package ma.gov.pfe.services;

import ma.gov.pfe.dtos.UserDto;

import java.util.List;

public interface IUserService {
    UserDto addUser(UserDto userDto);
    UserDto updateUser(UserDto userDto);
    void deleteUser(Long id);
    List<UserDto> selectUsers();
}
```

15. Dans le package “services”, créer La classe UserServiceImpl qui implémente IUserService:

La classe UserServiceImpl dépend de : UserRepository et UserMapper il faut les créer comme attributs et les injecter par constructeur.

```
package ma.gov.pfe.services;

import ma.gov.pfe.dtos.UserDto;
import ma.gov.pfe.mappers.UserMapper;
import ma.gov.pfe.repositories.UserRepository;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

@Service
public class UserServiceImpl implements IUserService {
    private final UserRepository userRepository;
    private final UserMapper userMapper;

    public UserServiceImpl(UserRepository userRepository,
UserMapper userMapper) {
        this.userRepository = userRepository;
        this.userMapper = userMapper;
    }

    @Override
    public UserDto addUser(UserDto userDto) {
        return
userMapper.toDto(userRepository.save(userMapper.toEntity(user
Dto)));
    }

    @Override
    public UserDto updateUser(UserDto userDto) {
        return
userMapper.toDto(userRepository.save(userMapper.toEntity(user
Dto)));
    }

    @Override
    public void deleteUser(Long id) {
        userRepository.deleteById(id);
    }

    @Override
    public List<UserDto> selectUsers() {
        return
userRepository.findAll().stream().map(userEntity ->
userMapper.toDto(userEntity)).collect(Collectors.toList());
    }
```

```
}
```

16. Dans le package “controllers”, créer la classe UserController.
Cette classe dépend de UserService, il faut l’injecter par constructeur.

```
package ma.gov.pfe.controllers;

import ma.gov.pfe.dtos.UserDto;
import ma.gov.pfe.services.IUserService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/users")
public class UserController {
    private final IUserService userService;

    public UserController(IUserService userService) {
        this.userService = userService;
    }
    @PostMapping
    public UserDto addUser(@RequestBody UserDto userDto) {
        return userService.addUser(userDto);
    }
    @PutMapping
    public UserDto updateUser(@RequestBody UserDto userDto) {
        return userService.updateUser(userDto);
    }
    @DeleteMapping("/{id}")
    public void deleteUser(@PathVariable("id") Long id) {
        userService.deleteUser(id);
    }
    @GetMapping
    public List<UserDto> selectUsers() {
        return userService.selectUsers();
    }
}
```

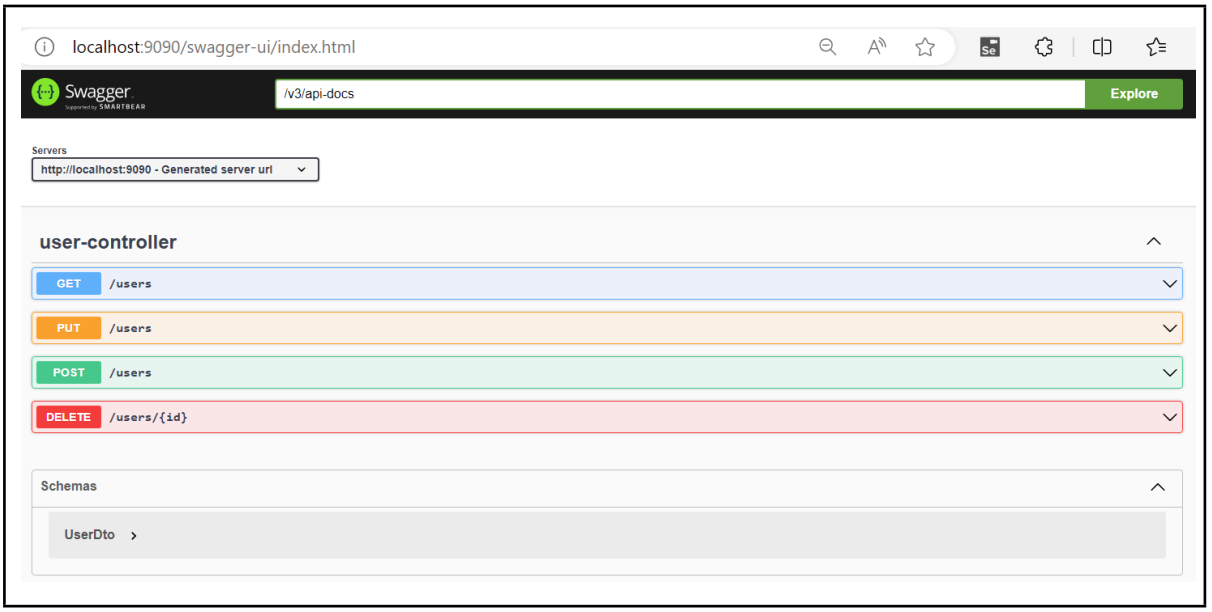
17. Ajoutez la dépendance Swagger pour tester votre application.
Ajouter la dependency suivante a votre pom.xml:

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.7.0</version>
</dependency>
```

18. Ajoutez la configuration Swagger pour tester votre application. Créer d'abord un package "config.swagger" puis la classe de configuration suivante

```
package ma.gov.pfe.config.swagger;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SwaggerConf {

    @Bean
    public OpenAPI getOpenApi() {
        return new OpenAPI().info(new Info());
    }
}
```

19. Démarrer l'application et Vérifier l'accès au swagger via le lien :
    http://localhost:9090/swagger-ui/index.html



20. Modifier la classe de demmarage MyApplication.java pour afficher le lien du swagger et le lien de la base H2

```
package ma.gov.pfe;


import lombok.extern.slf4j.Slf4j;
```

```java
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.core.env.Environment;

import java.net.InetAddress;
import java.net.UnknownHostException;

@SpringBootApplication
@Slf4j
public class MyApplication {

   public static void main(String[] args) throws
UnknownHostException {
       SpringApplication app = new
SpringApplication(MyApplication.class);
       Environment env = app.run(args).getEnvironment();
       log.info("Access
URLs:\n----------------------------------------------------------
----\n" +
                       "Url Swagger:
\thttp://127.0.0.1:{}/swagger-ui/index.html\n" +
                       "Url H2 Base:
\thttp://127.0.0.1:{}/h2-console\n" +

"------------------------------------------------------------",
               env.getProperty("server.port"),
               env.getProperty("server.port"));
   }
}
```

21. Démarrer et vérifier les URLs sur la console 👍

```
Hibernate: drop table if exists t_users CASCADE
Hibernate: create table t_users (id bigint not null, login varchar(255), name varchar(255), password varchar(255), primary key (id))
2025-01-18 18:54:20.923  INFO 18100 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform imp
2025-01-18 18:54:20.964  INFO 18100 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFac
2025-01-18 18:54:21.733  WARN 18100 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabl
2025-01-18 18:54:22.664  INFO 18100 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9090
2025-01-18 18:54:22.681  INFO 18100 --- [           main] ma.gov.pfe.MyApplication                 : Started MyApplication in 9.599 s
2025-01-18 18:54:22.685  INFO 18100 --- [           main] ma.gov.pfe.MyApplication                 : Access URLs:
------------------------------------------------------------
Url Swagger:    http://127.0.0.1:9090/swagger-ui/index.html
Url H2 Base:    http://127.0.0.1:9090/h2-console
------------------------------------------------------------
```

Exercice A : Création des composants Profile Entity

En se basant sur les étapes précédente, et en supposant que Profile Entity est définie par id(Long) et label (String), creer :

1. Créer ProfileEntity dans le package “entities”
2. Créer ProfileRepository dans le package “repositories”
3. Créer ProfileDto dans le package “dtos”
4. Créer ProfileMapper dans le package “mappers”
5. Créer IProfileService dans le package “services” avec les méthodes crud
6. Créer ProfileServiceImpl dans le package “services”.
    a. Implementer les methodes crud de l’interface IProfileService.
    b. Injecter ProfileMapeer et ProfileRepository
7. Créer ProfileController dans la package “controllers” avec les méthodes http: Get, Post, Put et Delete
8. Lancer l’application et tester par le swagger la mise à jour de la table des profiles

Exercice B : Création des composants Course Entity

En se basant sur les étapes précédente, et en supposant que Course Entity est définie par id(Long) , label(String) et hours(Integer), créer :

9. Créer CourseEntity dans le package “entities”
10. Créer CourseRepository dans le package “repositories”
11. Créer CourseDto dans le package “dtos”
12. Créer CourseMapper dans le package “mappers”
13. Créer ICourseService dans le package “services” avec les méthodes crud
14. Créer CourseServiceImpl dans le package “services”.
    a. Implementer les methodes crud de l’interface ICourseService.
    b. Injecter CourseMapeer et CourseRepository
15. Créer CourseController dans la package “controllers” avec les méthodes http: Get, Post, Put et Delete
16. Lancer l’application et tester par le swagger la mise à jour de la table des Courses

---