**+Etudiant 1** : Nom : Benkaba Prénom : Marwa

**Rapport TP N°02 :**

**Spring :**

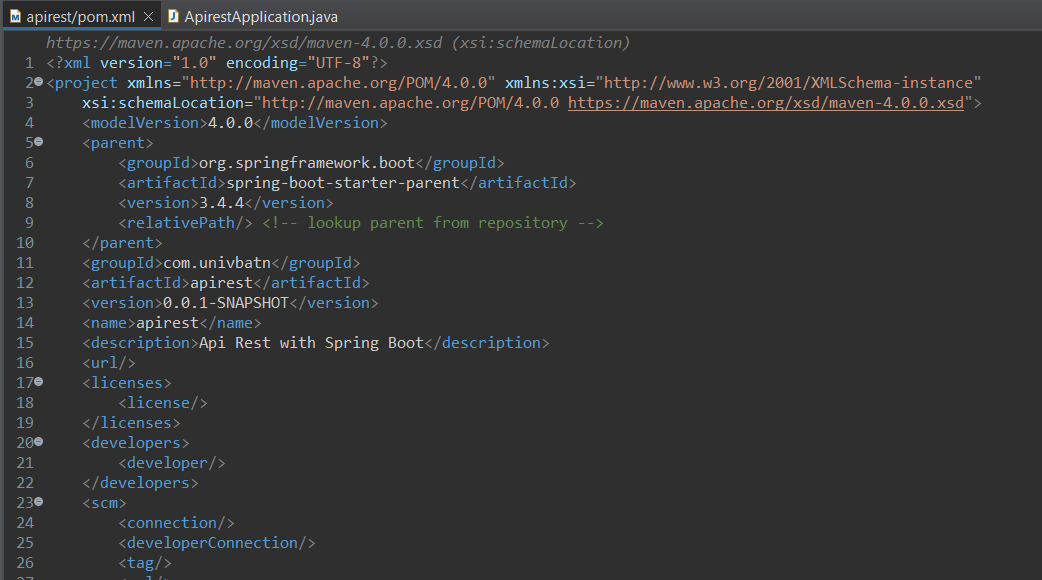
**A/ Environnement :**

 Spring tool suit

Jdk-21.

 Spring initializr : utilisé pour configurer le projet Spring Boot (connexion à la base de données, dépendances, etc.)

**B/ Dans un projet Spring, nous avons une structure minimale en trois étapes:**

**Étape 1 : Créer la structure minimale :**

1. Le fichier pom.xml contient les dépendances suivantes :

-springFramework.boot.

-JPA .

-web.

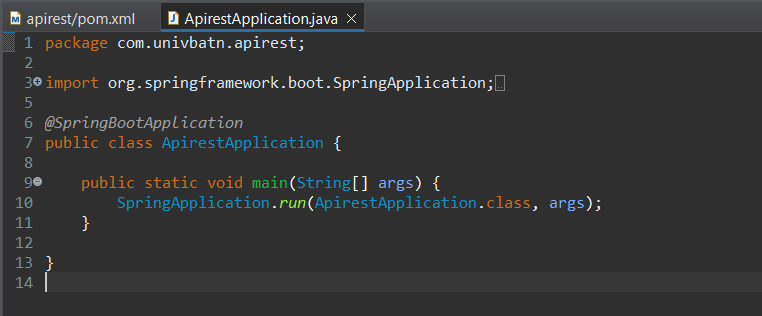
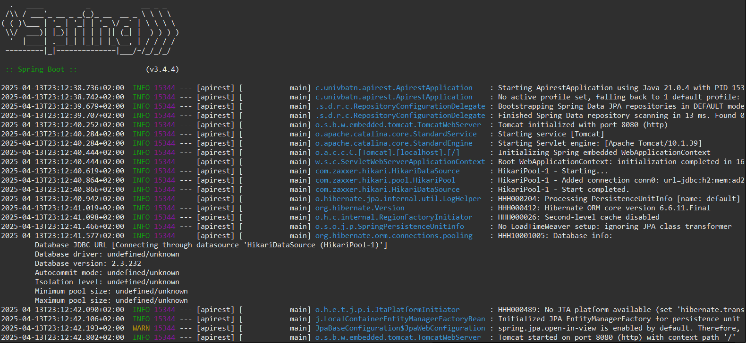
-Base de données "h2".

-Bibliothèque " Lombok ".

-framework "test" pour tester l’application.

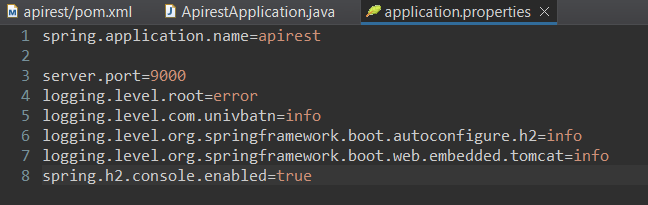
- Plugin Maven

1. Class ApirestApplication :

Lancer l'application : clic droit sur le projet 🡪 Run As 🡪Spring Boot App.

**Étape 2 : Configuration :**

1. Aller dans : apirest => src/main/resources => application.properties .

- Le fichier de configuration (application.properties) contient :

 spring.application.name=apirest  
→ Définit le nom de l'application : **apirest**.

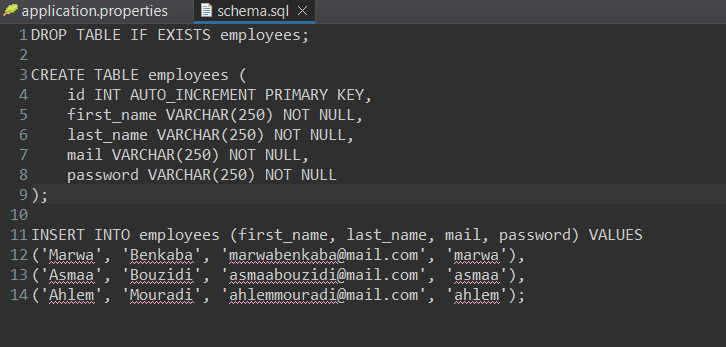
 server.port=9000  
→ Spécifie le port du serveur **Tomcat** (ici : 9000).

 logging.level.root=error  
→ Permet d'afficher uniquement les **erreurs** dans les logs (niveau de log : error).

 logging.level.org.springframework.boot.autoconfigure.h2=info  
→ Affiche les informations liées à **H2** lors du démarrage de l'application, notamment l'URL d'accès.

 logging.level.org.apache.catalina.core=info  
→ Affiche des informations sur le démarrage de **Tomcat** (utile pour confirmer qu’il s’exécute sur le port 9000).

 spring.h2.console.enabled=true  
→ Active la **console H2** pour accéder visuellement à la base de données via un navigateur.

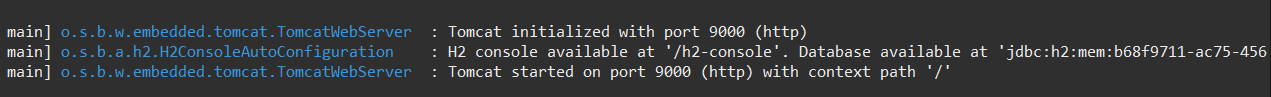
1. Ajouter un fichier de type SQL :

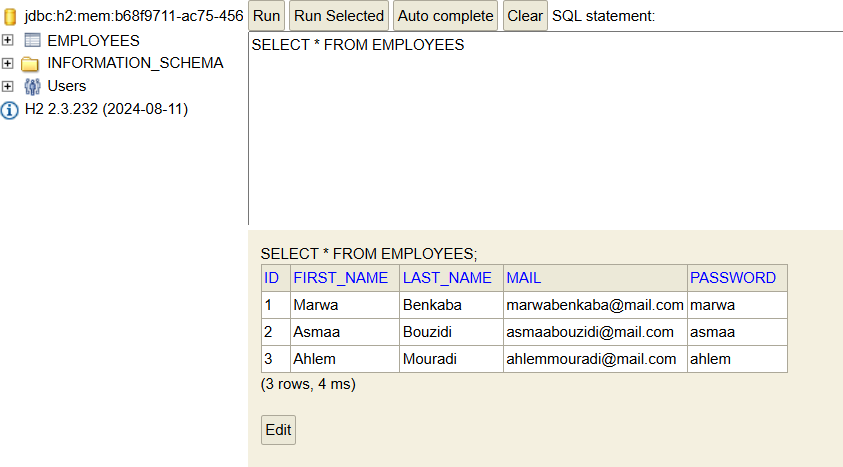
src/main/resources🡪 clic droit 🡪 New 🡪 General 🡪 File 🡪 Next 🡪 File name : "schema.sql" 🡪 Finish

Ce fichier permet de définir la structure de la base de données (table « employees », colonnes  « id,first\_name.. », etc.) qui sera automatiquement exécutée au démarrage de l’application.

1. Exécuté l'application :

Copier le lien dans le navigateur : <http://localhost:9000/h2-console> 🡪 Modifier l'**JDBC URL** avec le lien de la base de donnees.





**Étape 3 : Le développement :**

**1-**Créer 4 packages pour organiser l’application :

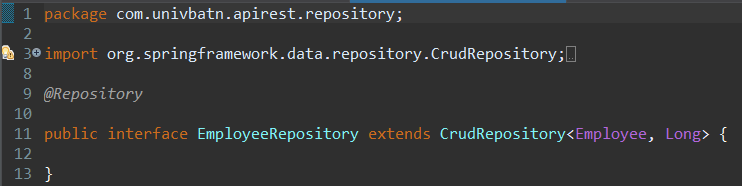
**🙠 La couche controller :** apirest 🡪 src (java) 🡪 New 🡪 Package com.univbatan.apirest.controller.

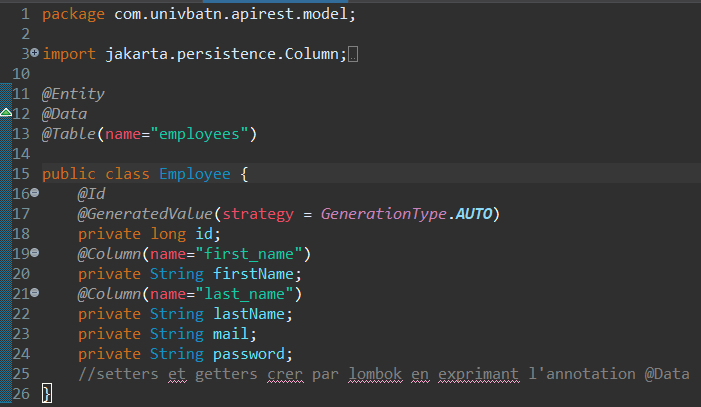
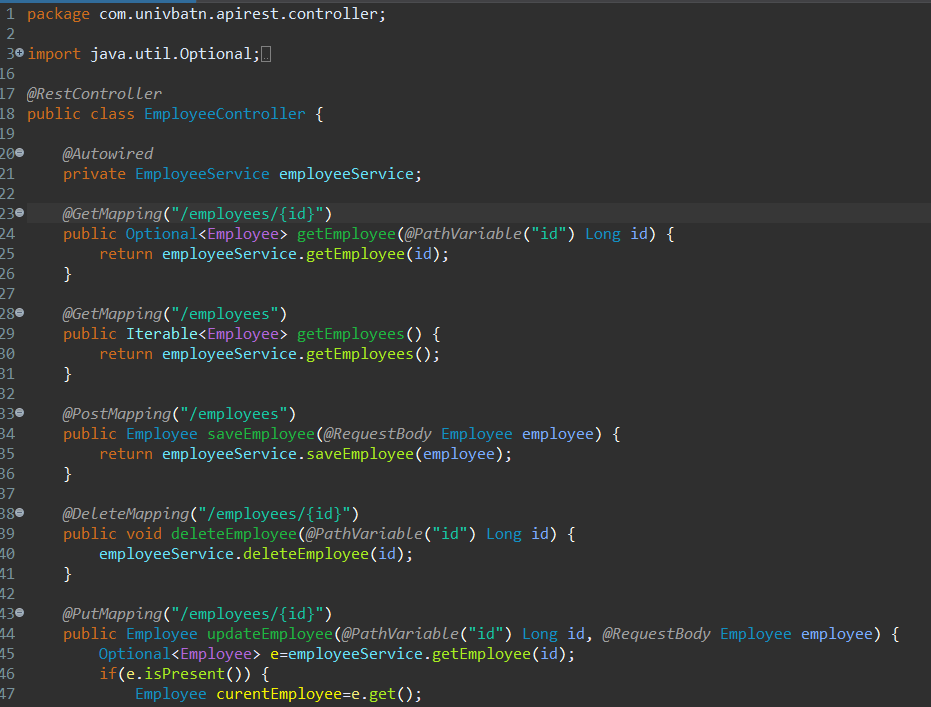
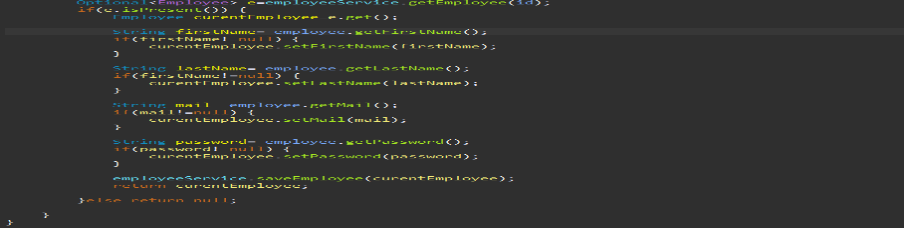
Le rôle de cette couche est de récupérer les requêtes du client et de les passer à la couche modèle.

**🙠La couche service :** apirest🡪src(java)🡪new🡪package "com.univbatan.apirest.service " .

**🙠La couche model :**apirest🡪src(java)🡪new🡪package " com.univbatan.apirest.model" .

**🙠** **La couche repository :** apirest🡪src(java)🡪new🡪package " com.univbatan.apirest.repository" .

C’est la couche responsable de la communication avec la base de données



**2-** L’implémentation des entité dans la couche model :

- Créé un class com.univbatan.apirest.modele=>new=>class"Employée".

- Ajouter les annotations :

@entity :Utilise pour indiquer qui cette class elle va être mapping sur le Db .

**@data :** C'est le bibliothèque qui va ajouter automatiquement les setters/getters pour tous les attributs.

**@ID :** represent ID de la Table ou clé primaire .

**@GeneratedValue**: Pour gérer les clés primaires.

**@table :** Pour changer le nom de la table.

**@Column :** Pour changer le nom de column.

**3-** Redémarrer l'application RunAs =>Spring Boot App.

**4-** L'implémentation de l'interface Cette interface fournis par le spring pour commencer avec le BD et exécuté les opérations CRUD , et l'implémentation de cette interface qui génèrer ou moment d'exécution

4.1/ créé interface com.univbatan.repository=>new=>interface=>"EmployeeRepository" =>finish

4.2/ Ajouter les Annotations \_@Repository Cette class represent un interface de communication avec le BD

**5-** couche service C'est une couche qui va implémenter la logique métier d'une autre application 5.1/ créé un class com.univbatan.service=>new=>class=>"EmployeeService" =>finish

5.2/ Ajouter les annotations

**@Autowired** qui permet d'injection le code "EmployeeRepository" dans class

**@service** Permet injection d'instances

**@data**

**6-** Exposé les opérations On utilise un service web de type Rest

6.1/ Créé un API Rest \_ API qui va guider la requête de client vers la methode la plus propriétaire com.univbatan.apirest.controller=>new=>class "EmployeeController" =>finish \_ ajouter un annotation @RestController

6.2/ Créé un objet de type service \_ utilise le annotation @Autowired pour injecté cette objet dans cette class

6.3/ Créé les methods source => Generate delegate mathode => sélectionnez les methods nécessaires=>Generate

6.4/ noté cette méthodes :

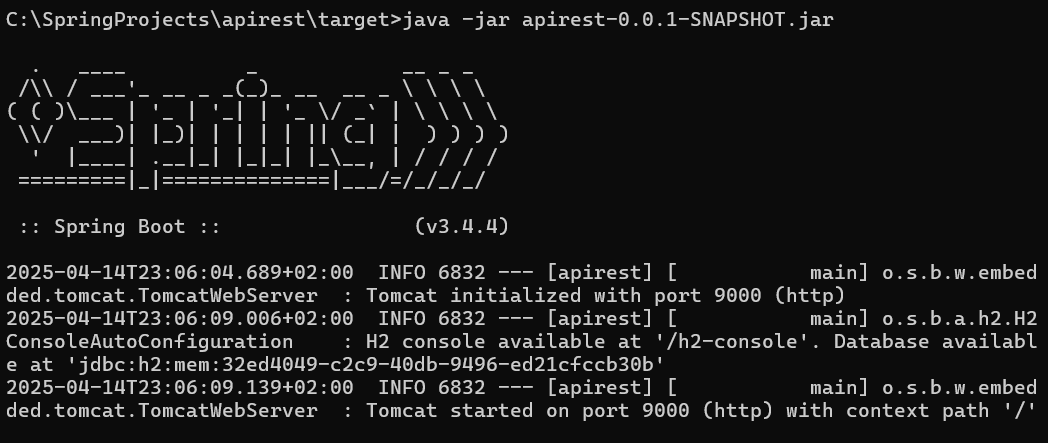
\_@GetMapping \_@PostMapping \_@DeleteMapping \_@PutMapping \_@PathVariable Pour les variables dans le lien \_@RequestBody Pour les variables dans le cœur de la requête http.

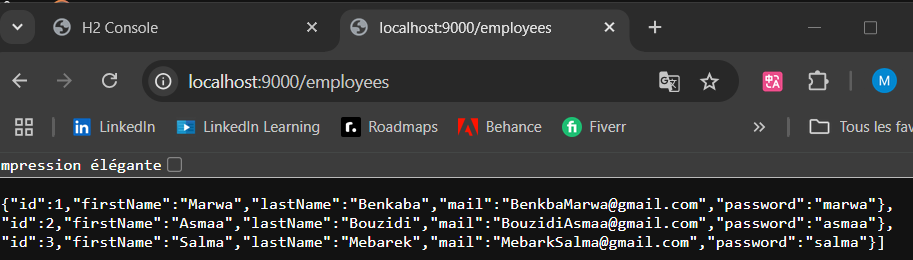
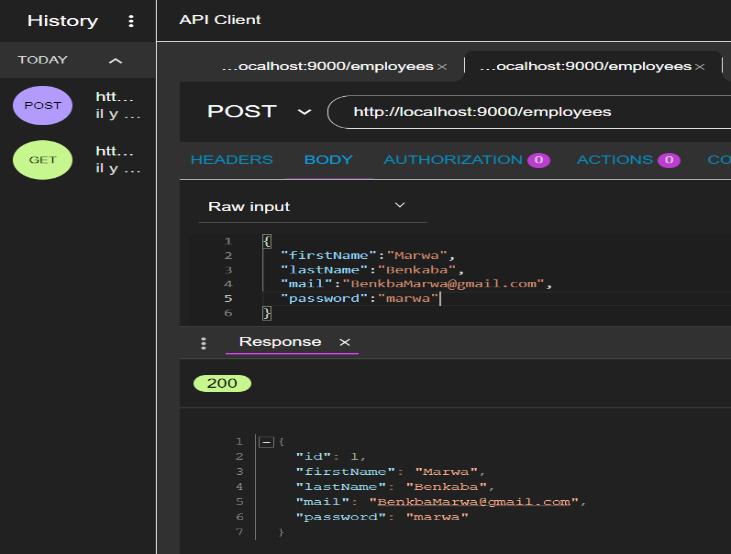
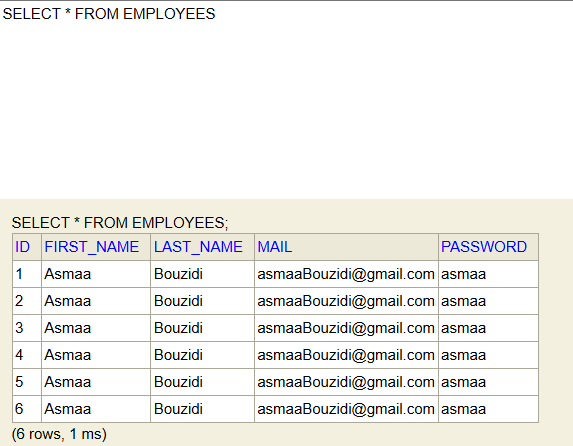
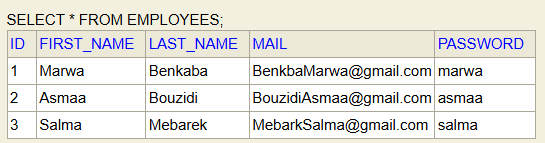
**7-** Les étapes de tester l'application "RestAPI" :

⚫ Lancé l'application Run us => SpringBoot app \_ généré un URL coller cette URL Dans navigateur pour afficher le contenaire de BD.  
⚫ utilisant le client ARC pour ajouter des éléments dans BD

⚫ Impacté le serveur dans application RunUs => MovenBuild => Goals " clean package " => Run Pour Générer le JAR cette fichier contient tous nécessaires de fonctionnement de serveur pour exécuter l'application

⚫Dans CMD : java -jar apirest-0.0.1-SNAPSHOT.jar





**NB :**  
**SNAPSHOT** : C’est une version en cours de développement, ce n’est pas la version finale. C’est une version qui accepte les mises à jour.

1- **Avantages de Spring Boot pour une API REST** :

* Configuration automatique
* Démarrage rapide (embedded Tomcat)
* Intégration facile avec JPA, sécurité, etc.
* Moins de configuration XML

2- **Inversion de Contrôle (IoC)** :

* Principe où le framework gère la création et l’injection des dépendances.
* Dans Spring, implémenté via le conteneur IoC et les annotations comme @Autowired.

3- **Différence entre les annotations** :

* @Component : Composant générique.
* @Repository : Couche d’accès aux données (DAO).
* @Service : Contient la logique métier.
* @Controller : Gère les requêtes web (MVC).