Ingénieur 2ème année
Spécialité Informatique - Apprentissage

QUIZON Rosselle

TP n°1 :
API Rest et Micro-services

Du développement au déploiement d'applications web

1. Objectifs du TP

Ce TP avait pour objectif de découvrir les bases du développement d'une API REST avec Spring Boot, puis d'étendre cette API avec une vraie couche de persistance via JPA/Hibernate, d'abord sur une base H2 en mémoire, puis sur MySQL.

Le TP était divisé en deux grandes parties :
- Création d'une API simple sans base de données (liste en mémoire).
- Passage à une API professionnelle connectée à une base de données (H2 puis MySQL).

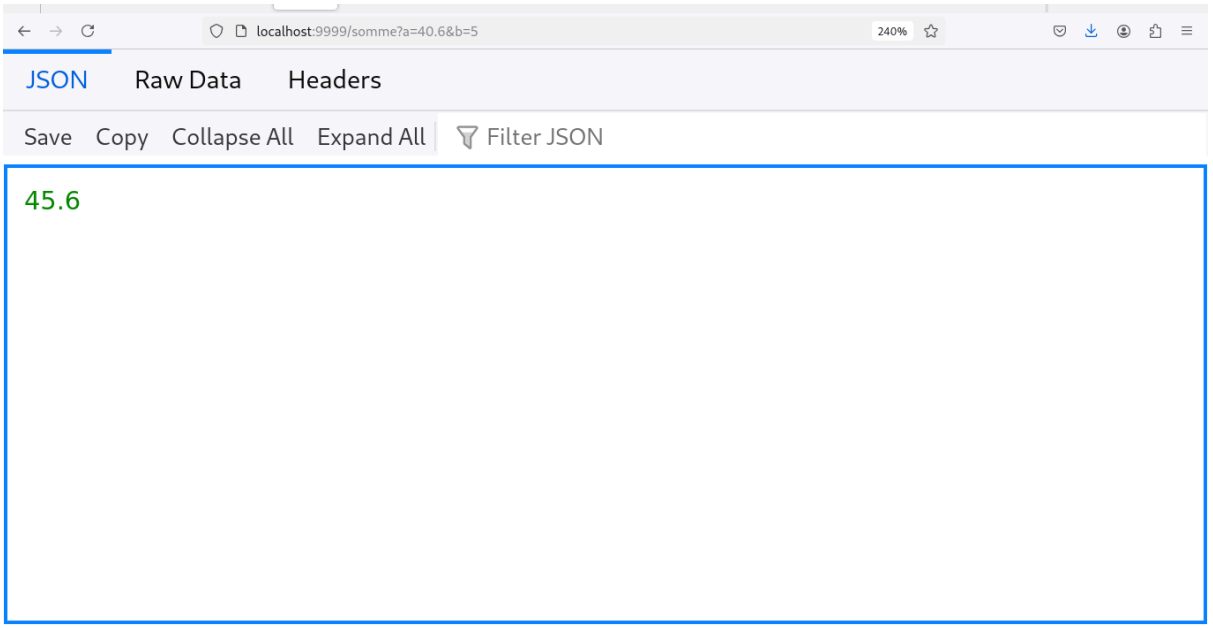
2. Lancement de l'application et initialisation

Le projet a été créé avec Spring Initializr via le site web générateur de projet Spring Boot. J'ai dû utiliser la version Java 17 pour des questions de compatibilité avec IntelliJ Community.

Il contient :
- une classe principale DemoApplication,
- un contrôleur REST MyApi,
- un modèle simple Etudiant.

La classe DemoApplication démarre Spring Boot et lance un serveur web (Tomcat embarqué) sur le port configuré dans application.properties.

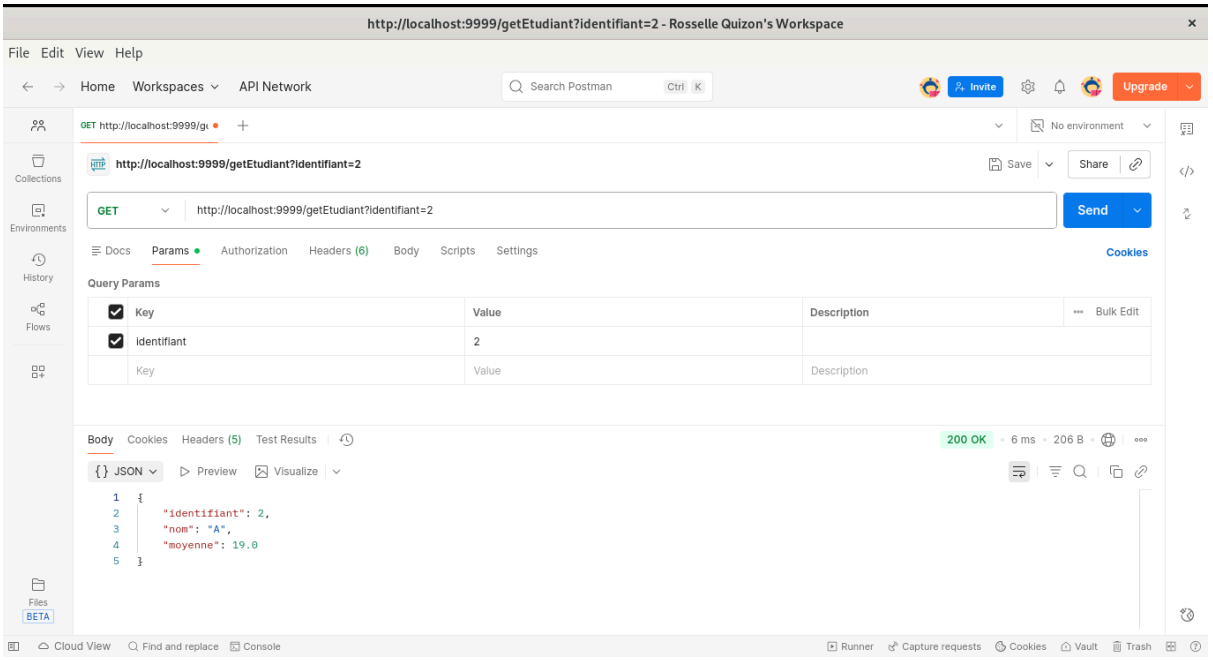
L'API tourne sur http://localhost:9999.



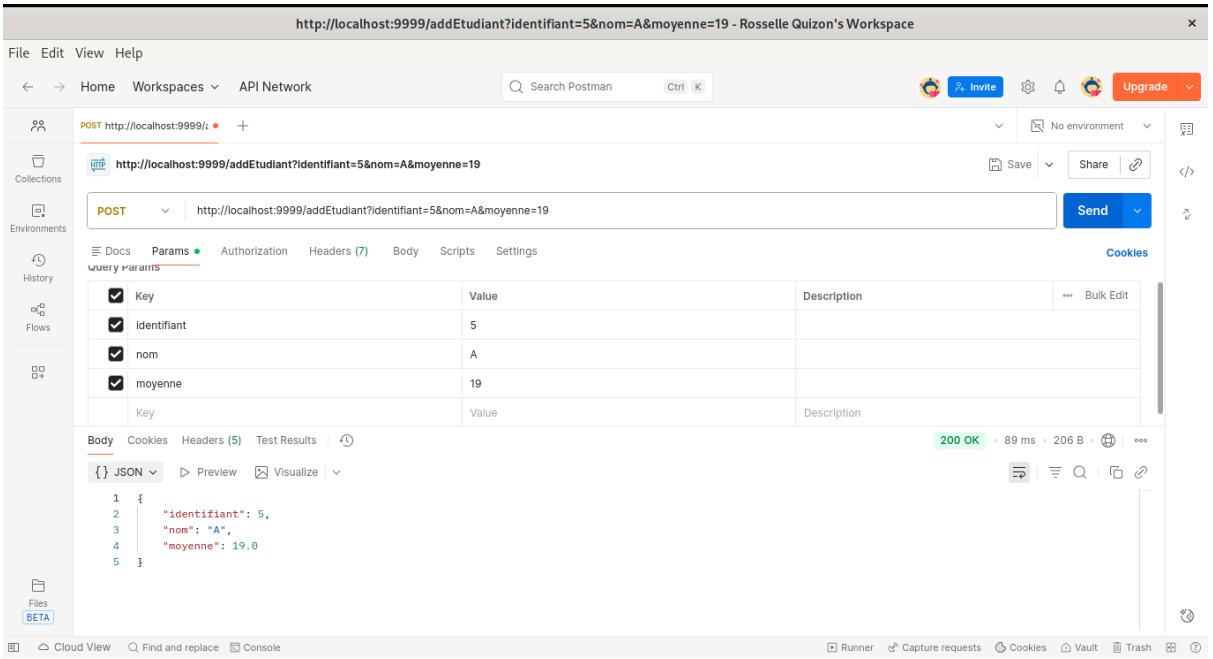
Dans la classe MyApi, une liste static sert de fausse base de données. Elle est remplie dès le chargement de la classe.

On crée ensuite les 4 méthodes importantes (Routes REST) :

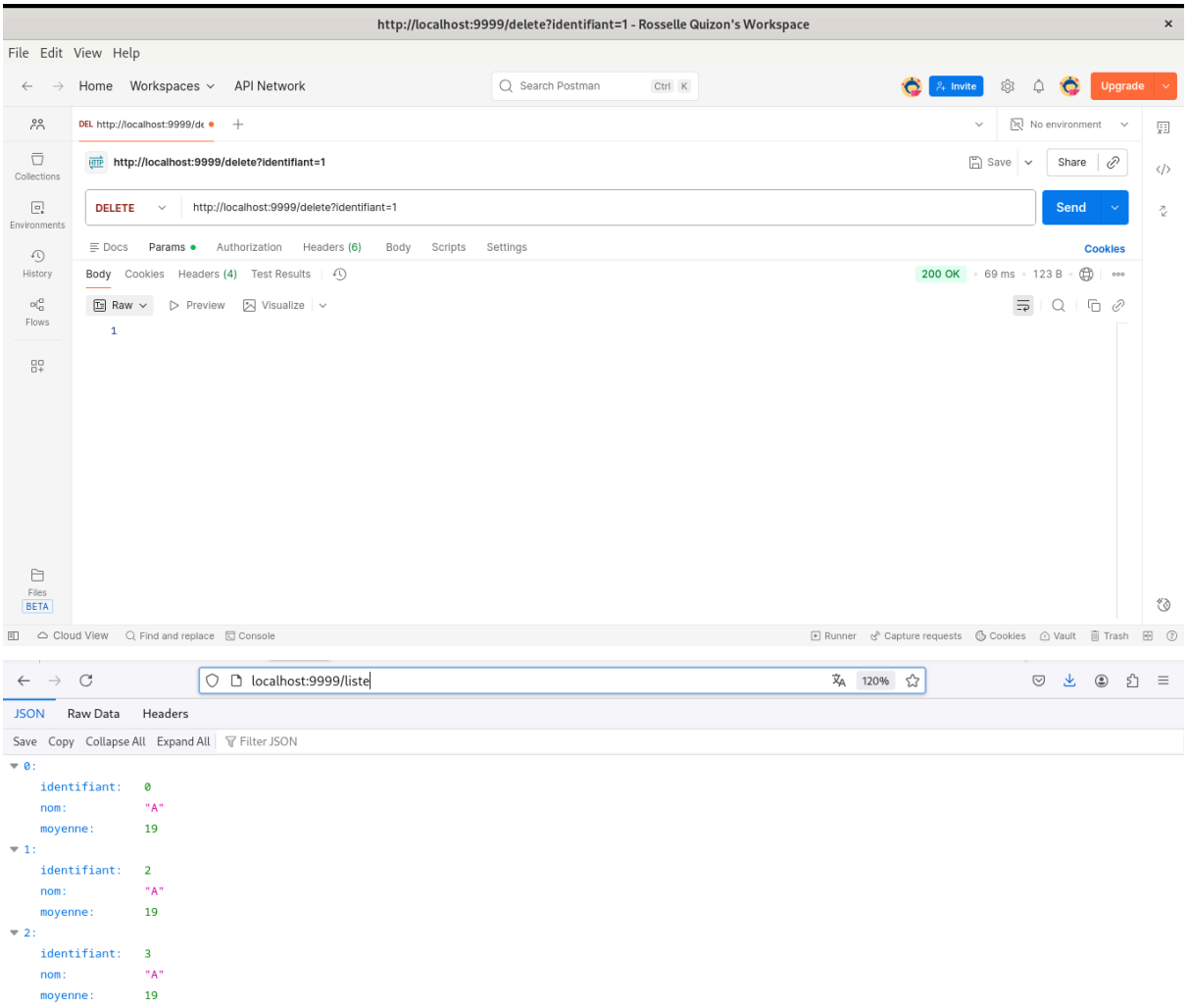
- GET :



- POST :

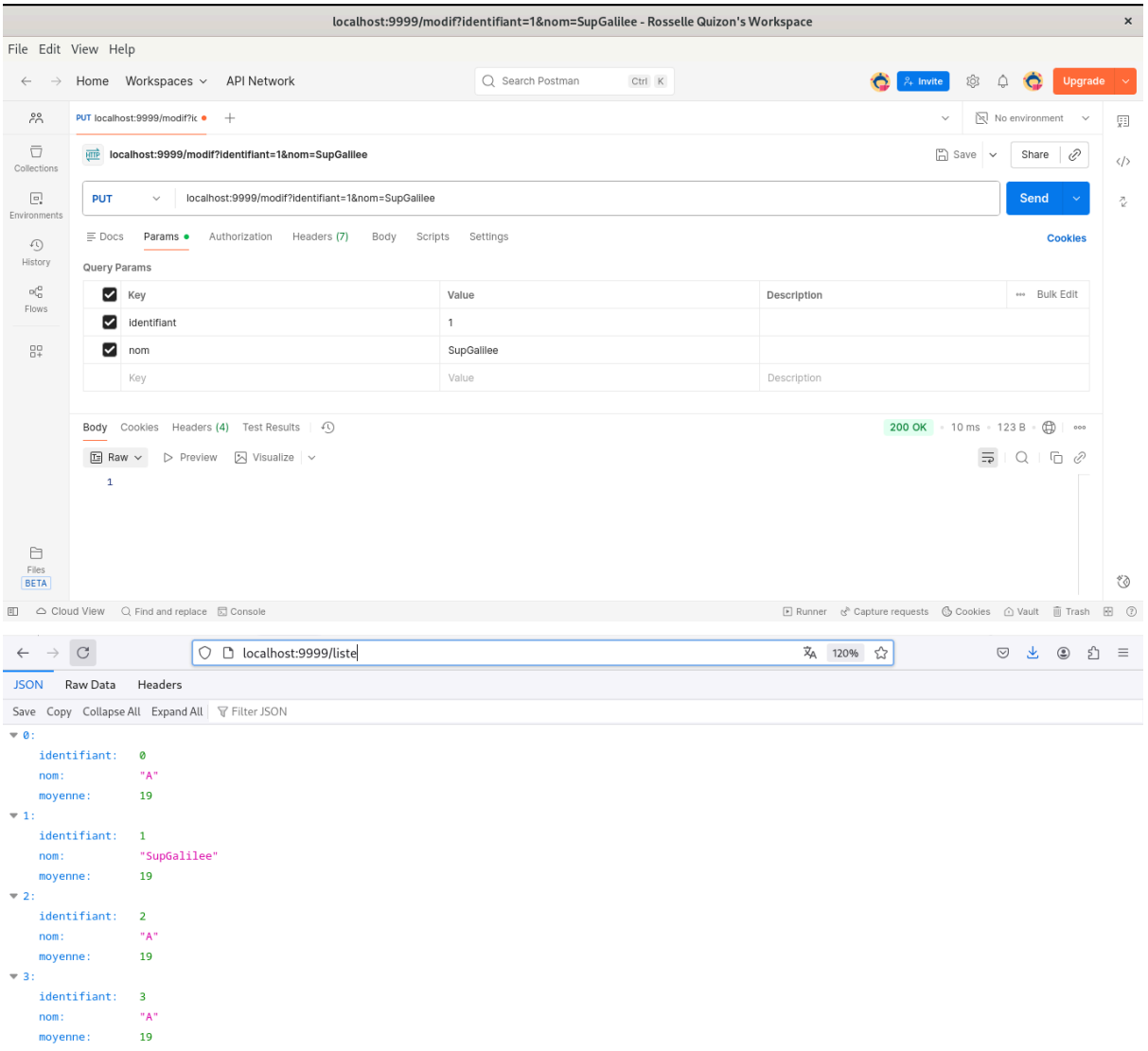


- DELETE :


http://localhost:9999/delete?identifiant=1 - Rosselle Quizon's Workspace
File Edit View Help
Home Workspaces API Network
Search Postman
Invite
Upgrade
No environment
Collections
Environments
History
Flows
Files BETA
http://localhost:9999/delete?identifiant=1
Save
Share
DELETE
http://localhost:9999/delete?identifiant=1
Send
Docs Params Authorization Headers (6) Body Scripts Settings
Cookies
Body Cookies Headers (4) Test Results
200 OK 69 ms 123 B
Raw Preview Visualize
1
Cloud View Find and replace Console
Runner Capture requests Cookies Vault Trash

localhost:9999/liste
120%
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
0:
identifiant: 0
nom: "A"
moyenne: 19
1:
identifiant: 2
nom: "A"
moyenne: 19
2:
identifiant: 3
nom: "A"
moyenne: 19

- PUT :



3. Modélisation des données : l'entité Adherent

Après la partie "liste en mémoire", on est passé sur une vraie base de données avec Hibernate qui génère les tables et Spring Data JPA pour communiquer avec la base.

La classe Adherent représente une entité JPA.

Les points importants :
- @Entity indique que cette classe doit être mappée sur une table relationnelle
- @Id désigne l'attribut qui sert de clé primaire en base.
- @GeneratedValue(strategy = GenerationType.AUTO) demande à Hibernate de générer automatiquement la valeur de l'id (auto-incrément).

4. AdherentRepository

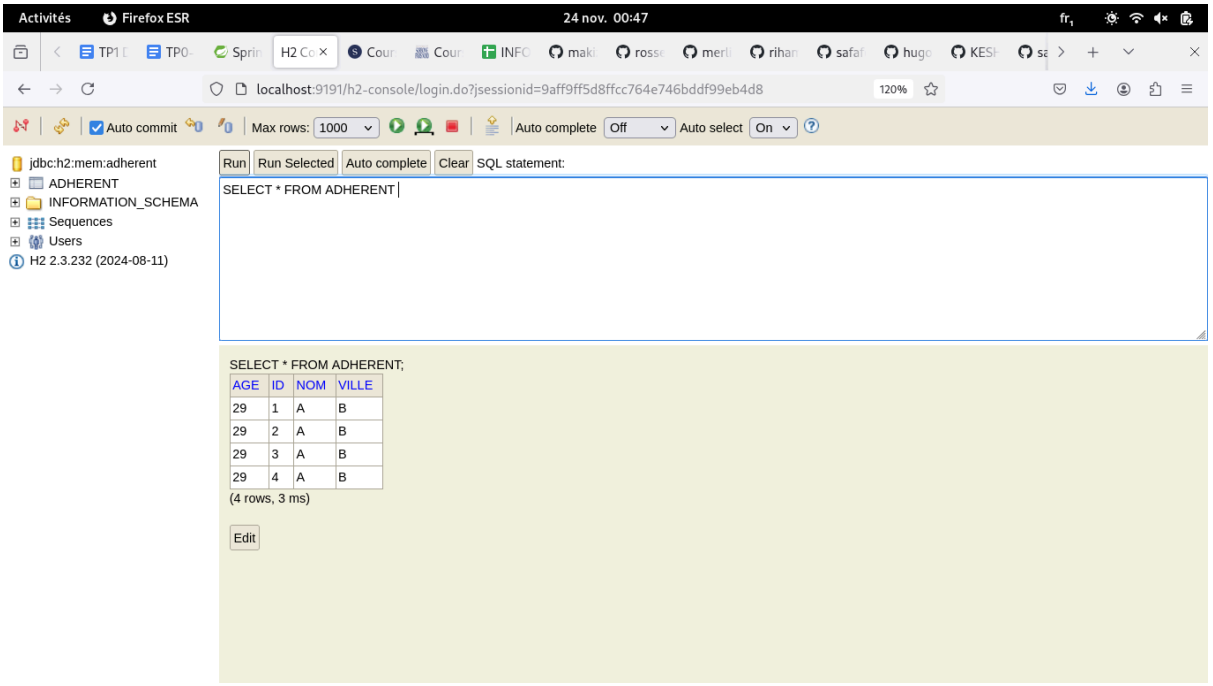
La couche d'accès aux données est assurée par cet interface :
public interface AdherentRepository extends JpaRepository<Adherent, Long> { }
JpaRepository<Adherent, Long> fournit automatiquement toutes les opérations CRUD standard (findAll, findById, save, deleteById, etc.) sans écrire de SQL à la main.
Le premier paramètre générique est le type de l'entité (Adherent), le second le type de la clé primaire (Long).
Grâce à Spring Data JPA, on n'a pas besoin d'implémenter l'interface : Spring génère l'implémentation à l'exécution et l'injecte là où on en a besoin (par exemple dans le CommandLineRunner ou dans un futur contrôleur REST).

5. Configuration de l'application : application.properties

- server.port=9191 modifie le port d'écoute du serveur Tomcat (par défaut c'est 8080)
- spring.datasource.url pointe vers la base MySQL adherent hébergée sur XAMPP (port MySQL par défaut : 3306).
- spring.datasource.username et spring.datasource.password contiennent les identifiants de connexion à MySQL.
- spring.jpa.hibernate.ddl-auto=create indique à Hibernate de créer le schéma à chaque démarrage (suppression et recréation des tables).

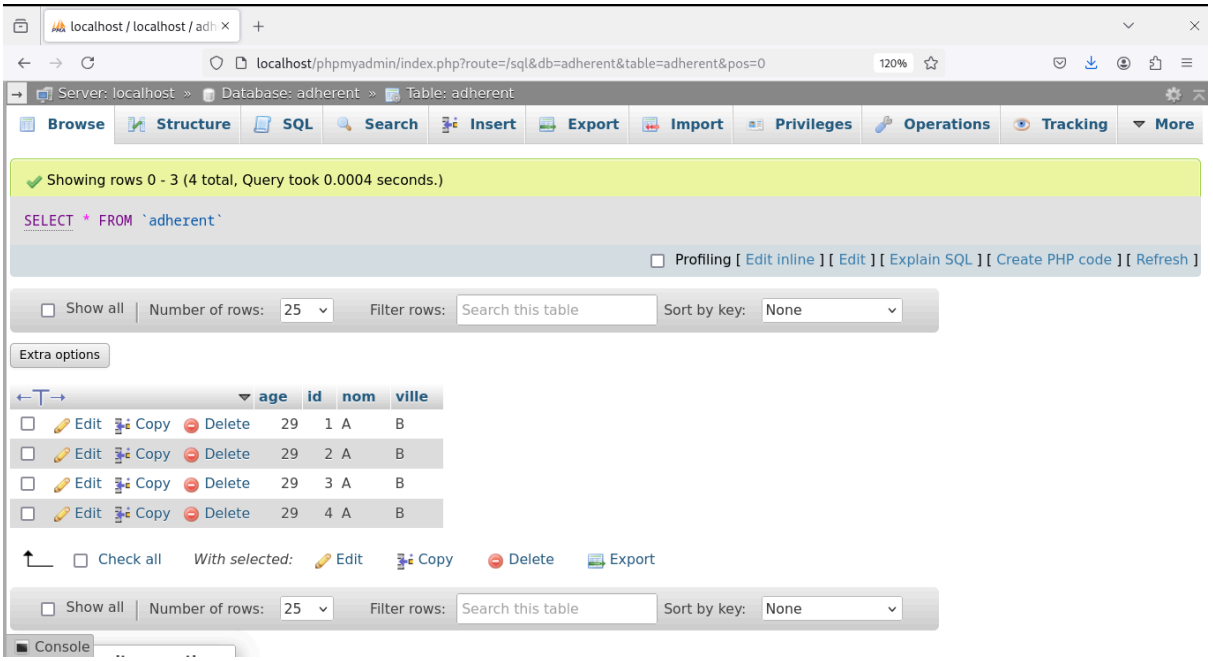
6. Test avec H2

Quand H2 est activé, Spring crée automatiquement une base en mémoire.



7. Passage à MySQL

Pour les Tests avec MySQL, la base de données que j'ai utilisée a été accessible via XAMPP.



Bilan

Ce TP t’a permis de :
- comprendre comment créer un projet Spring Boot avec Spring Initializr et Maven, et comment Spring Boot intègre automatiquement un serveur Tomcat embarqué ;
- mettre en œuvre les notions de JPA / Hibernate à travers la création d’une entité Adherent
- utiliser Spring Data JPA via l’interface AdherentRepository qui simplifie énormément la couche DAO ;
- configurer la connexion à une base de données à l’aide de quelques propriétés dans application.properties ;
- initialiser la base de données au démarrage grâce à un CommandLineRunner, pratique pour insérer des données de test ;
- tester concrètement les micro-services REST à l’aide de Postman et du navigateur, en observant les réponses JSON.