



ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR
Membre de

HONORIS UNITED UNIVERSITIES

COMPTE RENDU J2EE TP2

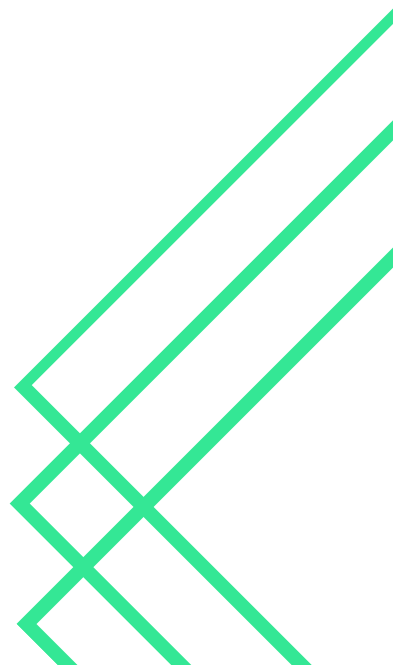
Réalisé par:
AZRHILIL Jihane

Objectif

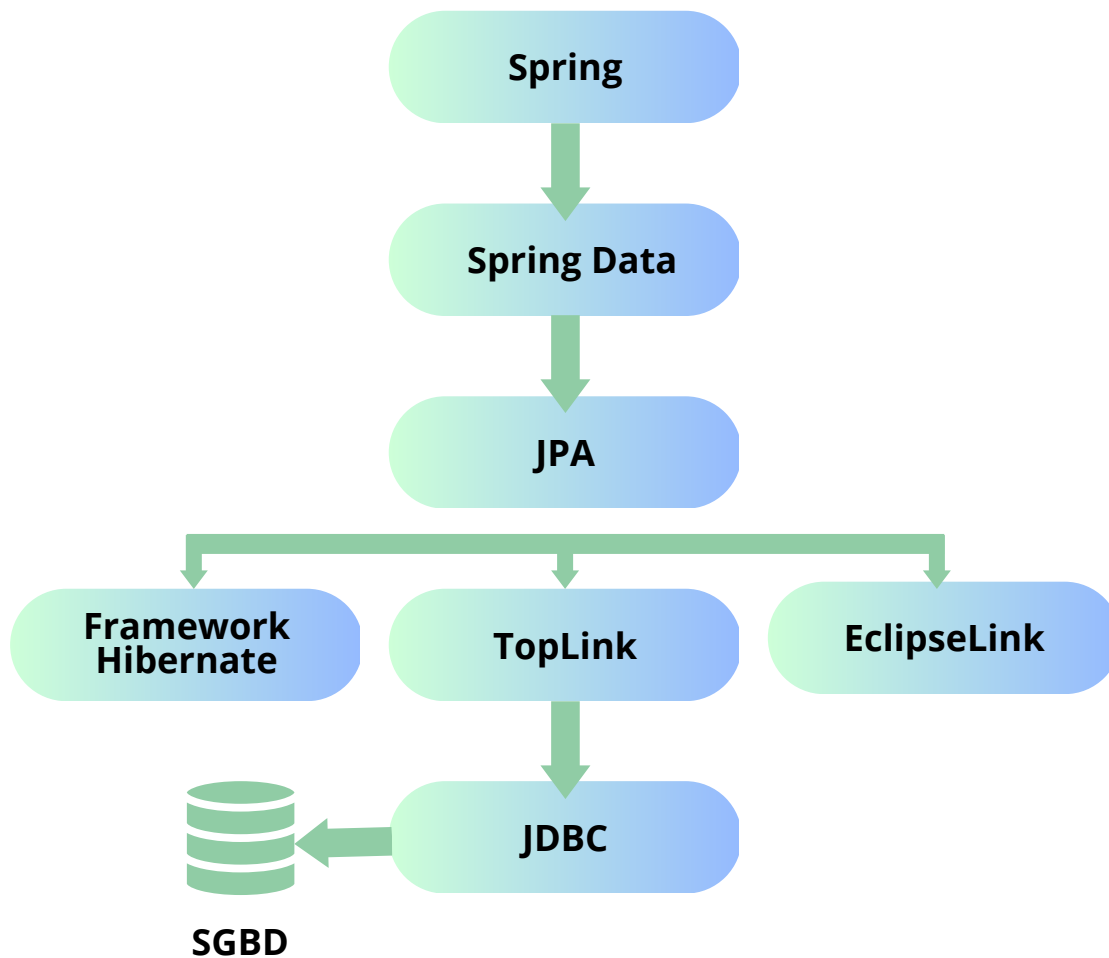
L'objectif de ce TP est de créer un projet Spring boot avec les différents opérations **CRUD** afin de le tester avec la base de données **H2** puis avec **mySQL**.

Etapes

- Créer un projet spring boot avec les dépendances: **spring data jpa, lombok, spring web, spring web dev tools, et h2 database.**
- Créer l'entité Jpa **Etudiant**.
- Configurer le data source ([application.properties](#))
- Créer l'**interface Etudiant Repository** basé sur spring data.
- Tester l'application avec des opération d'ajout de consultation de mise a jour et de suppression des étudiants.
- Utiliser une base de donnée **Mysql** au lieu de h2.



Architecture du projet:



Spring Data: un module de spring qui donne le concept de mapping.

JPA: une interface qui peut être implémenter par plusieurs classes.

- **EMF:** EntityManagerFactory (crée les 2 entities au dessous)
- **EM:** EntityManger (offre les différents gestion CRUD)
- **ET:** EntityTransaction (pour tous ce qui est transaction)

Hibernate: un framework qui présente le mapping (ORM)

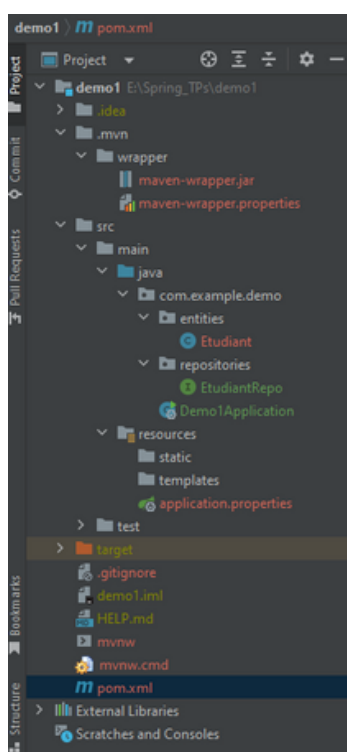
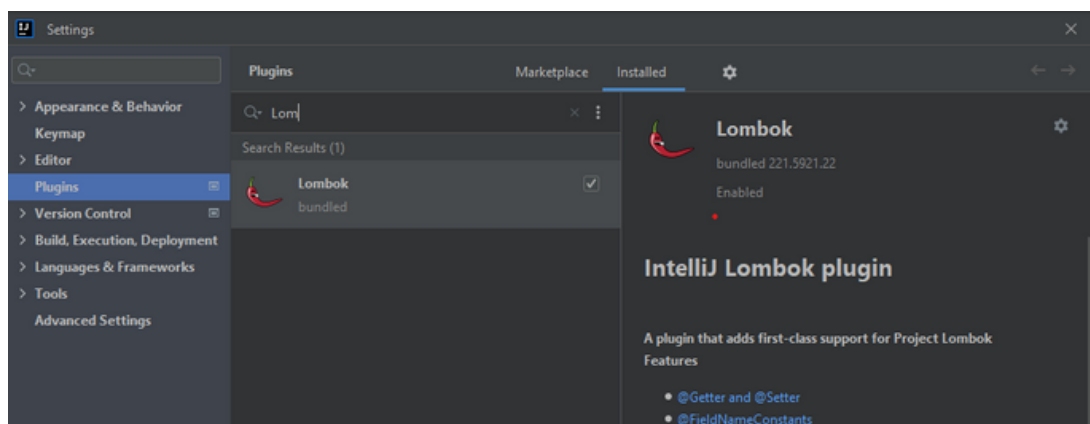
JDBC: un API java qui permet l'accès et l'interaction avec la base de données.

Création du projet Spring:

On va créer un projet Spring qui fait automatiquement appel à **maven** et qui va nous permettre gérer les différents dépendances:

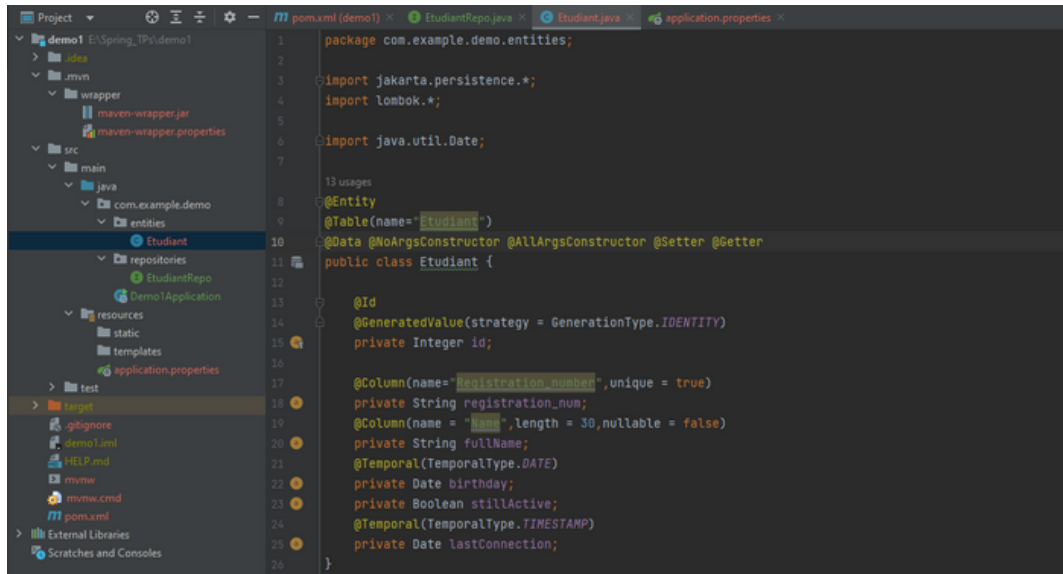
- **H2 Database:** un type de base de données du test en localhost
- **Spring Data JPA**
- **Lombok:** qui permet de gérer les constructeurs, les setters, les getters...
- **Spring web**
- **Spring Boot DevTools:** permet de compiler automatiquement et directement après la mise à jour

Après la création du projet on doit vérifier si le Lombok est vraiment installé. Allons vers **Settings-> Plugins-> cherchant Lombok**



On va créer un package **entities** qui va regrouper toutes les classes créées. Puis un autre package **repositories** qui contient notre interfaces qui hérite de la classe **JpaRepository**.

La classe Etudiant



```
1 package com.example.demo.entities;
2
3 import jakarta.persistence.*;
4 import lombok.*;
5
6 import java.util.Date;
7
8 @Entity
9 @Table(name="Etudiant")
10 @Data @NoArgsConstructor @AllArgsConstructor @Setter @Getter
11 public class Etudiant {
12
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Integer id;
16
17     @Column(name="Registration_number", unique = true)
18     private String registration_num;
19     @Column(name = "Fullname", length = 30, nullable = false)
20     private String fullName;
21     @Temporal(TemporalType.DATE)
22     private Date birthday;
23     private Boolean stillActive;
24     @Temporal(TemporalType.TIMESTAMP)
25     private Date lastConnection;
26 }
```

Les annotations:

@Entity: permet de créer une table qui correspond à notre classe ainsi de mapper sur toute la classe.

@Table: permet de créer la table sous le nom donné en paramètre

@AllArgsConstructor: constructeur avec paramètres

@NoArgsConstructor: constructeur sans arguments

Remarque: les deux constructeurs sont gérés par le Lombok

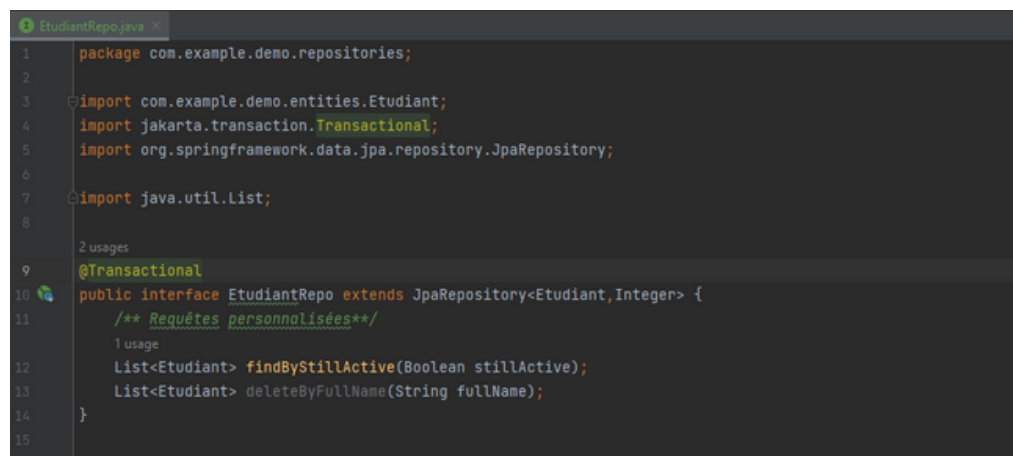
@Id: permet de définir la clé primaire

@GeneratedValue: permet d'incrémenter l'id

@Temporal avec TimeStamp: permet de donner la date avec l'heure

@Temporal avec le type Date: permet de donner seulement la date.

Interface Etudiant



```
1 package com.example.demo.repositories;
2
3 import com.example.demo.entities.Etudiant;
4 import jakarta.transaction.Transactional;
5 import org.springframework.data.jpa.repository.JpaRepository;
6
7 import java.util.List;
8
9 @Transactional
10 public interface EtudiantRepo extends JpaRepository<Etudiant,Integer> {
11     /** Requetes personnalisées**/
12     List<Etudiant> findByStillActive(Boolean stillActive);
13     List<Etudiant> deleteByFullName(String fullName);
14 }
15
```

Chaque interface créée dans la couche **DAO** doit hériter de la classe **JpaRepository** qui prends en paramètres le nom de la classe et le type de la clé primaire.

Au sein de cette interface on peut implémenter des fonctions qui ne se trouvent pas dans JpaRepository, en respectant la forme de cette classe.

On peut aussi définir des requêtes personnalisées qui ne respectent pas la forme de JpaRepository.

Classe main

```
1 usage
@SpringBootApplication
public class DemoApplication implements CommandLineRunner {
    14 usages
    @Autowired
    private EtudiantRepo etdRepo;
    public static void main(String[] args) { SpringApplication.run(DemoApplication.class, args); }

    @Override
    public void run(String... args) throws Exception {
```

La classe implémente l'interface **CommandLineRunner** qui permet à l'application de s'exécuter systématiquement sans besoin de redemander l'exécution.

Les annotations:

@SpringApplication permet d'exécuter l'application.

@Autowired permet d'activer l'injection des dépendances sur un objet d'une manière automatique.

1. Insertion

```
System.out.println("***** Insertion *****");
etdRepo.save(new Etudiant( id: null, registration_num: "A1", fullName: "Alya Dine", new Date( year: 2000, month: 8, date: 4), stillActive: true, new Date()));
etdRepo.save(new Etudiant( id: null, registration_num: "A2", fullName: "Rayan Berrada", new Date( year: 1997, month: 10, date: 12), stillActive: true, new Date()));
etdRepo.save(new Etudiant( id: null, registration_num: "A3", fullName: "Lilya Bennis", new Date( year: 2001, month: 2, date: 5), stillActive: false, new Date()));
etdRepo.save(new Etudiant( id: null, registration_num: "A4", fullName: "Lina Azel", new Date( year: 1997, month: 3, date: 11), stillActive: true, new Date()));
etdRepo.save(new Etudiant( id: null, registration_num: "A5", fullName: "Anwar Idrici", new Date( year: 1995, month: 8, date: 2), stillActive: false, new Date()));
System.out.println("***** Inserted rows *****");
System.out.println("Count:" + etdRepo.count()); // count pour calculer le nombre d'enregistrement dans la BD
```

- la méthode **save** permet d'enregistrer la ligne insérée dans la base de données.
- La méthode **count** permet de calculer le nombre des enregistrements.

```
Run: DemoApplication x
Console
***** Inserted rows *****
Count:5
```

2. Consultation

```
System.out.println("\n***** Display Rows *****");
List<Etudiant> etudiantList=etdRepo.findAll(); // récupérer tous les lignes
etudiantList.forEach(etudiant->{System.out.println(etudiant.toString());});
```

```
***** Display Rows *****
Etudiant(id=1, registration_num=A1, fullName=Alya Diman, birthday=3908-06-04, stillActive=true, lastConnection=2023-04-08 01:00:32.434)
Etudiant(id=2, registration_num=A2, fullName=Rayan Berrada, birthday=3897-11-12, stillActive=true, lastConnection=2023-04-08 01:00:32.521)
Etudiant(id=3, registration_num=A3, fullName=Lilya Bennis, birthday=3901-03-05, stillActive=false, lastConnection=2023-04-08 01:00:32.523)
Etudiant(id=4, registration_num=A4, fullName=Lina Azel, birthday=3897-04-11, stillActive=true, lastConnection=2023-04-08 01:00:32.524)
Etudiant(id=5, registration_num=A5, fullName=Anwar Idrissi, birthday=3895-07-02, stillActive=false, lastConnection=2023-04-08 01:00:32.526)
```

La méthode **findAll** c'est une méthode prédéfini qui permet de récupérer tous les enregistrements.

3. Modification et recherche par Id:

```
System.out.println("\n***** Get Element By Id *****");
Etudiant etd= etdRepo.findById(2).orElse( other: null); // chercher l'étudiant à partir de son Id
System.out.println(etd.toString());
System.out.println("\n***** Update Element *****");
etd.setRegistration_num("S2");
etdRepo.save(etd); // sauvegarder la ligne modifiée
System.out.println(etd.toString());
```

```
***** Get Element By Id *****
Etudiant(id=2, registration_num=A2, fullName=Rayan Berrada, birthday=3897-11-12, stillActive=true, lastConnection=2023-04-08 01:00:32.521)

***** Update Element *****
Etudiant(id=2, registration_num=S2, fullName=Rayan Berrada, birthday=3897-11-12, stillActive=true, lastConnection=2023-04-08 01:00:32.521)
```

4. Suppression et sélection:

```
System.out.println("\n***** Delete Element *****");
etdRepo.delete(etd);
System.out.println("Count:"+etdRepo.count());
etdRepo.deleteById(3);
System.out.println("Count:"+etdRepo.count());
System.out.println("\n***** Select active students *****");
List<Etudiant> activeEtds=etdRepo.findByStillActive(true);
activeEtds.forEach(aEtd->{System.out.println(aEtd.toString());});
```

```
***** Delete Element *****
Count:4
Count:3

***** Select active students *****
Etudiant(id=1, registration_num=A1, fullName=Alya Diman, birthday=3908-06-04, stillActive=true, lastConnection=2023-04-08 01:00:32.434)
Etudiant(id=4, registration_num=A4, fullName=Lina Azel, birthday=3897-04-11, stillActive=true, lastConnection=2023-04-08 01:00:32.524)
```

application.properties

```
application.properties
1 spring.datasource.url=jdbc:h2:mem:students
2 spring.h2.console.enabled=true
3 spring.datasource.username=admin
4 server.port=8080
5
```

Exécution

English Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded)

Setting Name: Generic H2 (Embedded) Save Remove

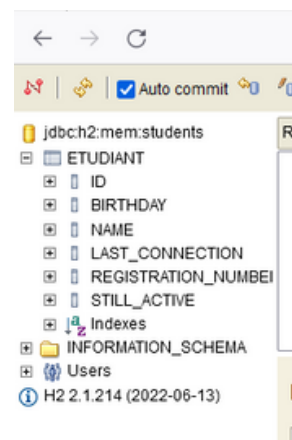
Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:students

User Name: admin

Password:

Connect Test Connection



jdbc:h2:mem:students

ETUDIANT

INFORMATION_SCHEMA

Users

H2 2.1.214 (2022-06-13)

Run Run Selected Auto complete Clear SQL statement

SQL statement

select * from Etudiant

ID	BIRTHDAY	NAME	LAST_CONNECTION	REGISTRATION_NUMBER	STILL_ACTIVE
1	3900-06-04	Alya Diman	2023-04-08 01:00:32.434	A1	TRUE
4	3897-04-11	Lina Azel	2023-04-08 01:00:32.524	A4	TRUE
5	3895-07-02	Anwar Idriiss	2023-04-08 01:00:32.526	A5	FALSE

(3 rows, 5 ms)

Edit

Utilisation d'une base de donnée Mysql au lieu de h2.

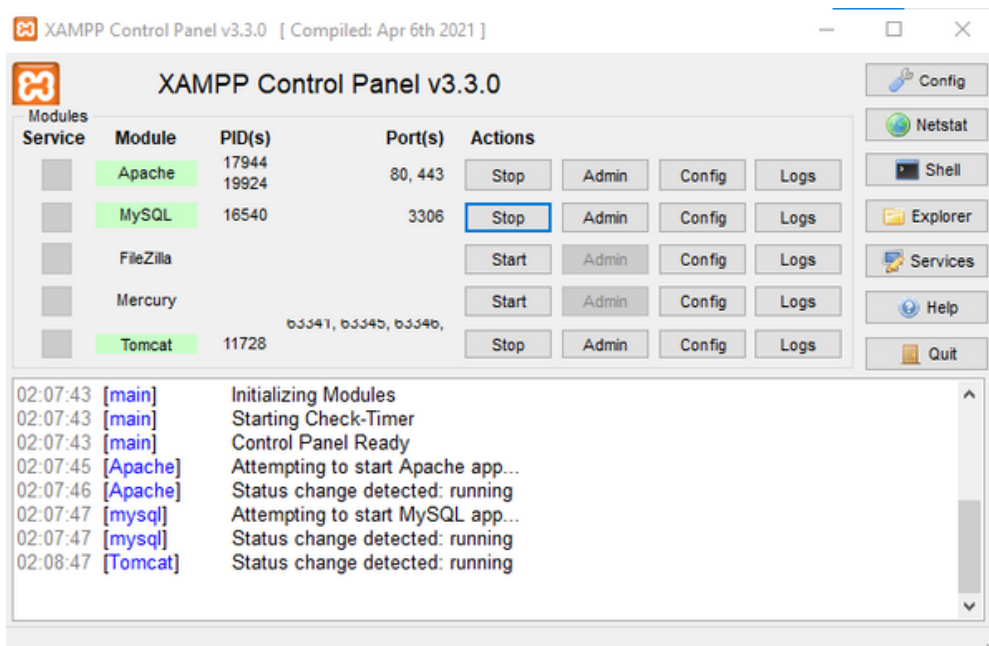
- pom.xml

```
<!--<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency-->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.24</version>
</dependency>
```

- application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/Etudiant?createDatabaseIfNotExist=true
spring.datasource.username=root
server.port=8080
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.show-sql=true
```

- Exécution



localhost/phpmyadmin/index.php?route=/sql&pos=0&db=etudiant&table=etudiant

phpMyAdmin

Récentes Préférences

Nouvelle base de données

- etudiant
 - Nouvelle table
 - etudiant
 - Colonnes
 - Nouvelle colonne
 - birthday (date, NULL, nullable)
 - id (PRI, int)
 - last_connection (datetime, NULL, nullable)
 - name (varchar)
 - registration_number (UNI, varchar, NULL)
 - still_active (bit, NULL, nullable)
 - Index
 - information_schema
 - mysql
 - performance_schema
 - phpmyadmin
 - test

Seigneur : 127.0.0.1 > Base de données : etudiant > Table : etudiant

Parcourir Structure SQL Rechercher Insérer Exporter Importer Privileges Opérations Plus

Affichage des lignes 0 - 2 (total de 3, traitement en 0.0005 seconde(s).)

SELECT * FROM `etudiant`

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher Nombre de lignes : 25 Filtrer les lignes : Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

				id	birthday	name	last_connection	registration_number	still_active
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	3900-06-04	Alya Diman	2023-04-08 02:08:46	A1	1
<input type="checkbox"/>	Éditer	Copier	Supprimer	4	3897-04-11	Lina Azel	2023-04-08 02:08:46	A4	1
<input type="checkbox"/>	Éditer	Copier	Supprimer	5	3895-07-02	Anwar Idrissi	2023-04-08 02:08:46	A5	0

Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

Tout afficher Nombre de lignes : 25 Filtrer les lignes : Chercher dans cette table Trier par clé : Aucun(e)

Opérations sur les résultats de la requête

Imprimer Copier dans le presse-papiers Exporter Afficher le graphique Créer une vue

Console de requêtes SQL