



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**

Membre de

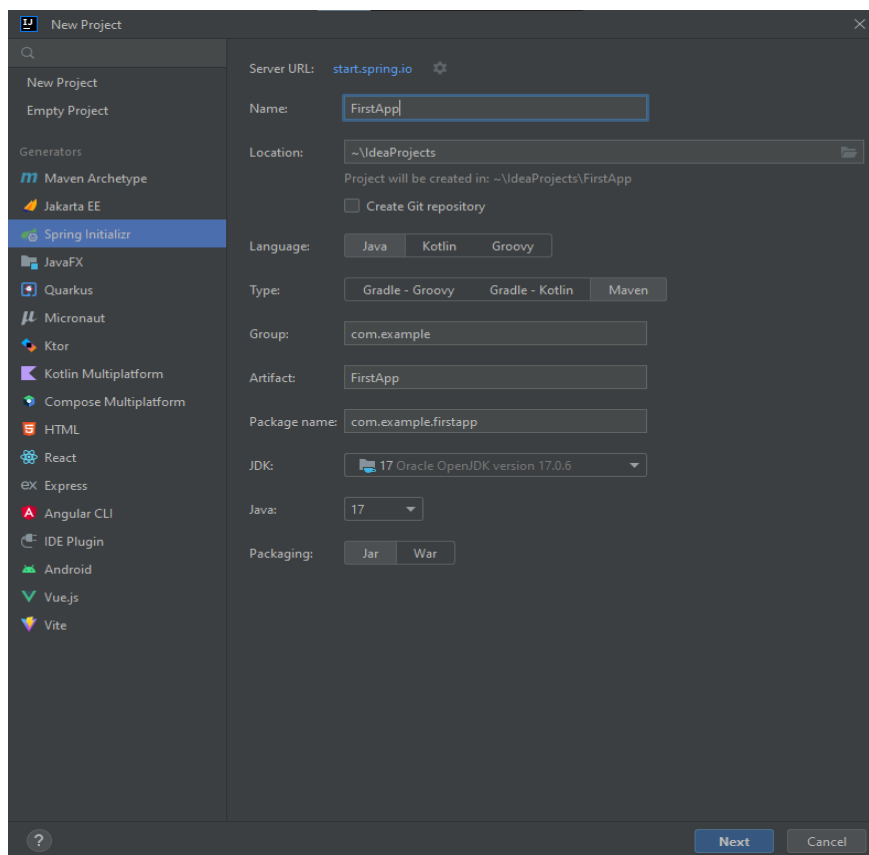
HONORIS UNITED UNIVERSITIES

Compte Rendu TP Jpa, Hibernate, Spring data

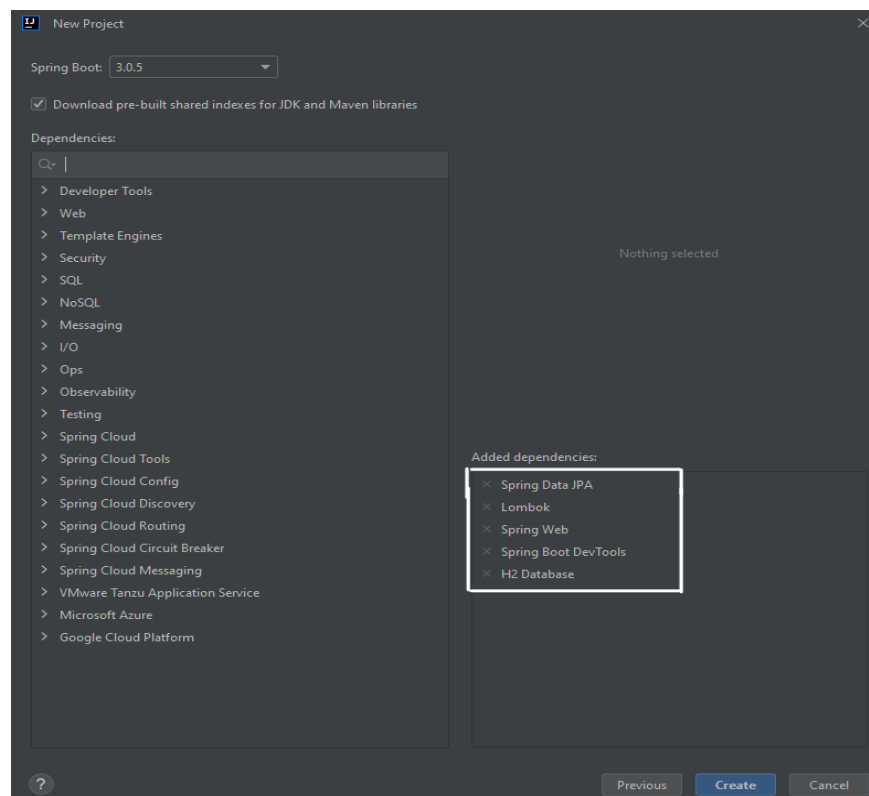
4IIR-EMSI Centre -G5

Réalisé par : Faidi Zahira

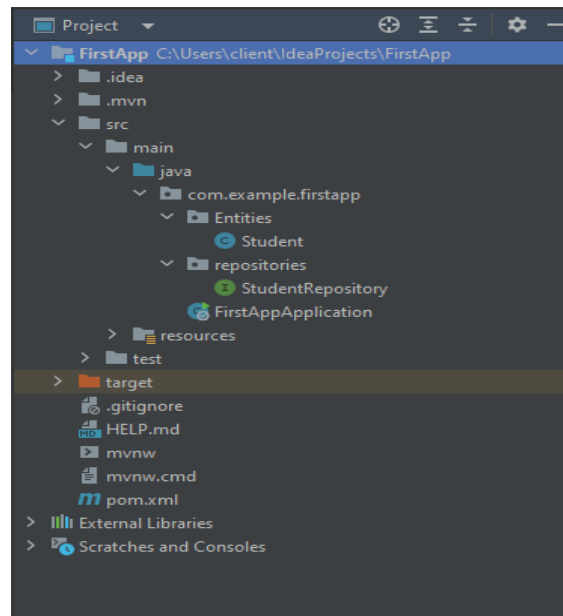
Création de projet Spring



L'ajout des dépendances nécessaire : spring data jpa, lombok, spring web, spring web dev tools, et h2 BD



Structure Globale du projet



Code de l'entité JPA Student

```
package com.example.firstapp.Entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.CreationTimestamp;

import java.util.Date;

@Entity @Table(name="EMSI_STUDENTS")
@Data @AllArgsConstructor @NoArgsConstructor
public class Student {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column(name = "REGISTRATION N", unique = true)
    private String registrationNumber;
    @Column(name = "Name", length = 30, nullable = false)
    private String fullName;
    @Temporal(TemporalType.DATE)
    private Date birthday;
    private Boolean stillActive;
    @Temporal(TemporalType.TIMESTAMP) @CreationTimestamp
    private Date lastConnection;
}
```

Configuration de la data source (application.properties)

```
spring.datasource.url=jdbc:h2:mem:students
spring.datasource.username=admin
server.port=8080
```

Création de l'interface Etudiant Repository basé sur spring data

```
package com.example.firstapp.repositories;

import com.example.firstapp.Entities.Student;
import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student, Integer> {

}
```

Implementation des Opérations CRUD

```
package com.example.firstapp;

import com.example.firstapp.Entities.Student;
import com.example.firstapp.repositories.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import java.util.Date;
import java.util.List;

@SpringBootApplication
public class FirstAppApplication implements CommandLineRunner {
    @Autowired
    private StudentRepository studentRepository;

    public static void main(String[] args) {
        SpringApplication.run(FirstAppApplication.class, args);
    }

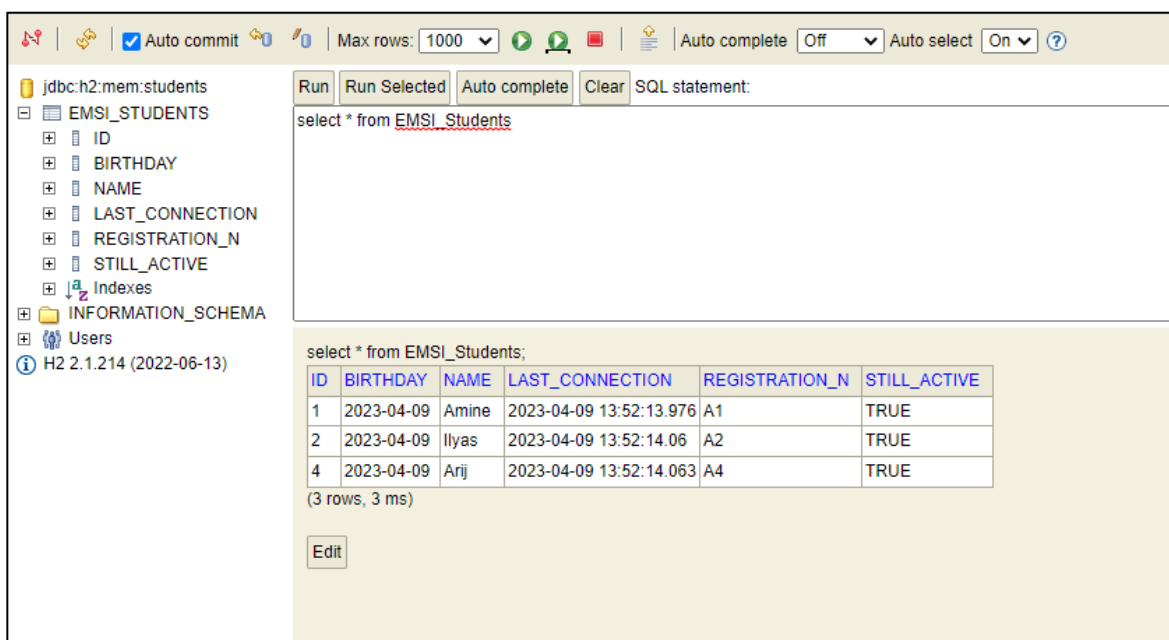
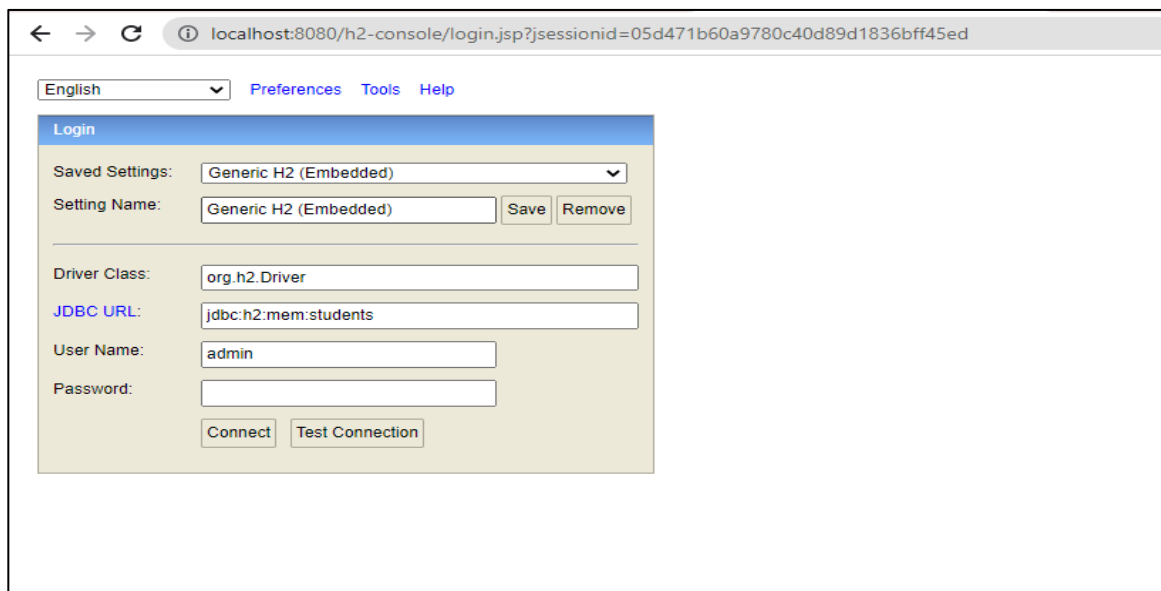
    @Override
    public void run(String... args) throws Exception {
        System.out.println("***** Insertion *****");
        studentRepository.save(new Student(null, "A1", "Amine", new Date(), true, null));
        studentRepository.save(new Student(null, "A2", "Ilyas", new Date(), true, null));
        studentRepository.save(new Student(null, "A3", "Saad", new Date(), true, null));
        studentRepository.save(new Student(null, "A4", "Arij", new Date(), true, null));
        studentRepository.save(new Student(null, "A5", "Lina", new Date(), true, null));
        System.out.println("***** Inserted rows *****");
        System.out.println("Count : "+studentRepository.count());
        System.out.println("***** Display rows *****");
        List<Student> students = studentRepository.findAll();
        students.forEach(student -> {
            System.out.println(student.toString());
        });
        System.out.println("***** Get Element By ID *****");
        Student student = studentRepository.findById(3).orElse(null);
        System.out.println(student.toString());
        System.out.println("***** Update an Element *****");
        student.setRegistrationNumber("S3");
        studentRepository.save(student);
        System.out.println("***** Delete an Element *****");
        studentRepository.delete(student);
        System.out.println("Count : "+studentRepository.count());

        studentRepository.deleteById(5);
        System.out.println("Count : "+studentRepository.count());
    }
}
```

Résultat de l'exécution

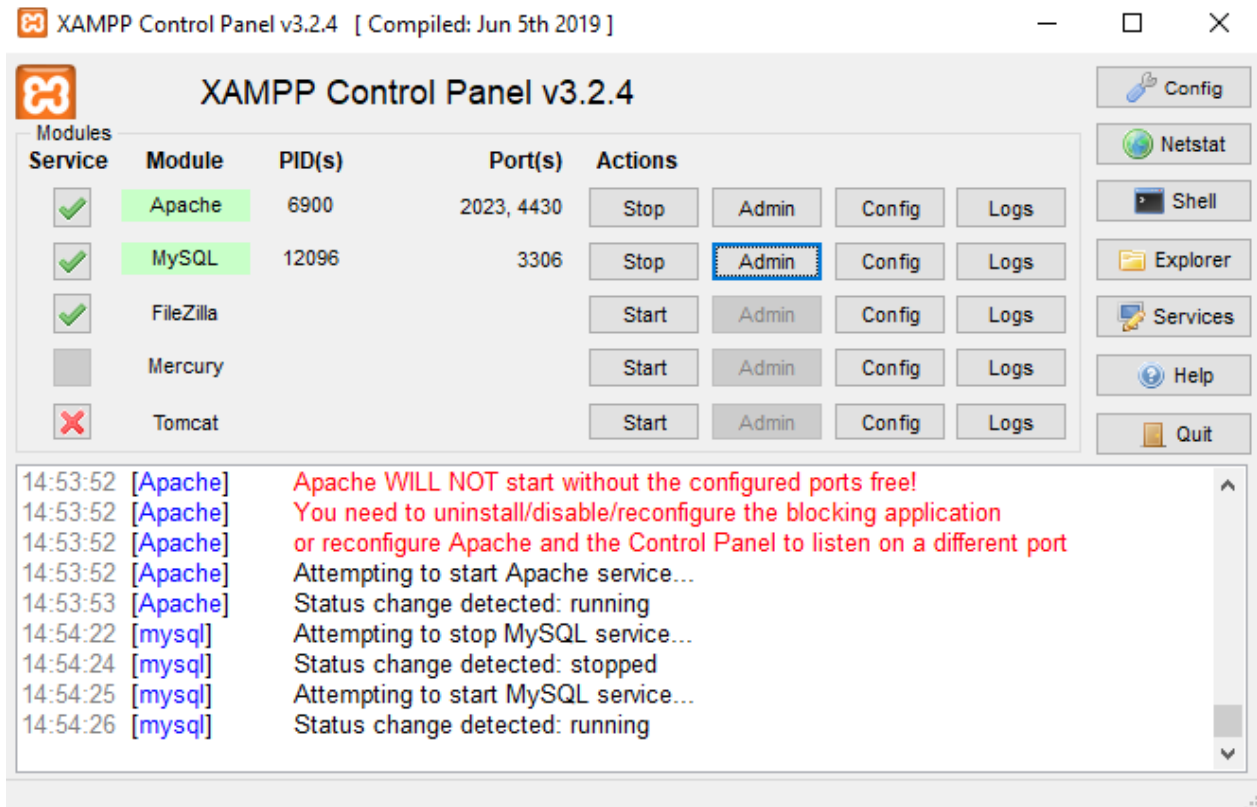
```
***** Insertion *****
***** Inserted rows *****
Count : 5
***** Display rows *****
Student(id=1, registrationNumber=A1, fullName=Amine, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.88)
Student(id=2, registrationNumber=A2, fullName=Ilyas, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.947)
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.949)
Student(id=4, registrationNumber=A4, fullName=Arij, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.95)
Student(id=5, registrationNumber=A5, fullName=Lina, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.952)
***** Get Element By ID *****
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 13:45:17.949)
***** Update an Element *****
***** Delete an Element *****
Count : 4
Count : 3
```

Visualiser les données à partir de la base de données H2



ID	BIRTHDAY	NAME	LAST_CONNECTION	REGISTRATION_N	STILL_ACTIVE
1	2023-04-09	Amine	2023-04-09 13:52:13.976	A1	TRUE
2	2023-04-09	Ilyas	2023-04-09 13:52:14.06	A2	TRUE
4	2023-04-09	Arij	2023-04-09 13:52:14.063	A4	TRUE

Changer la base de données H2 en MySql



Lancement de service MySQL avec XAMPP ensuite en modifier la configuration du fichier application.properties avec la configuration suivante :

```
spring.datasource.url=jdbc:mysql://localhost:3306/Etudiant?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
server.port=2020
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=true
```

L'ajout de la dépendance JDBC MySQL dans le fichier pom.xml

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.32</version>
</dependency>
```

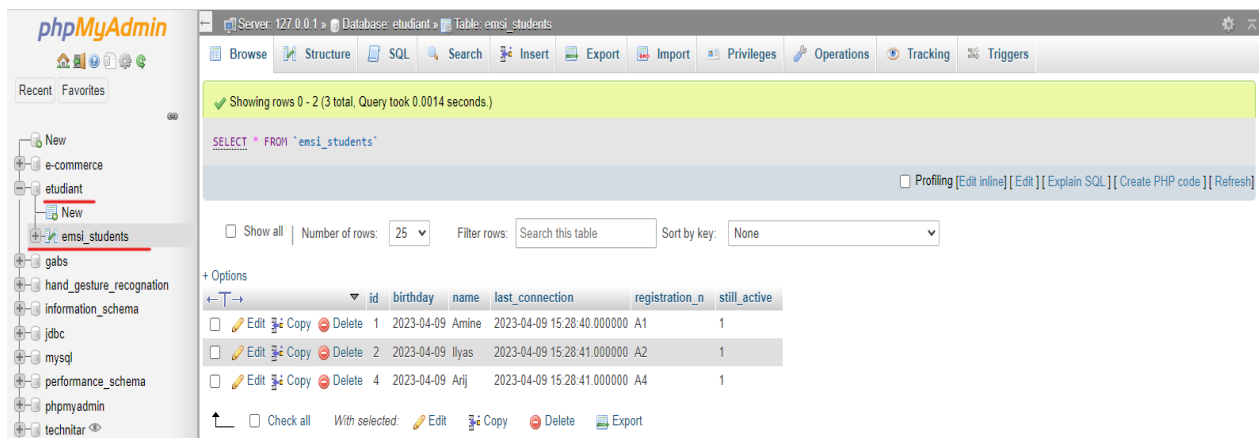
La base de données Etudiant est créée s'il n'existe pas ainsi que la table « emsi_students »

Résultat de compilation des opérations CRUD

```
***** Insertion *****
Hibernate: insert into emsi_students (birthday, name, last_connection, registration_n, still_active) values (?, ?, ?, ?, ?)
Hibernate: insert into emsi_students (birthday, name, last_connection, registration_n, still_active) values (?, ?, ?, ?, ?)
Hibernate: insert into emsi_students (birthday, name, last_connection, registration_n, still_active) values (?, ?, ?, ?, ?)
Hibernate: insert into emsi_students (birthday, name, last_connection, registration_n, still_active) values (?, ?, ?, ?, ?)
***** Inserted rows *****
Hibernate: select count(*) from emsi_students s1_0
Count : 5
***** Display rows *****
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0
Student(id=1, registrationNumber=A1, fullName=Amine, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:40.0)
Student(id=2, registrationNumber=A2, fullName=Ilyas, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
Student(id=4, registrationNumber=A4, fullName=Arif, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
Student(id=5, registrationNumber=A5, fullName=Lina, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
***** Get Element By ID *****
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0 where s1_0.id=?
Student(id=3, registrationNumber=A3, fullName=Saad, birthday=2023-04-09, stillActive=true, lastConnection=2023-04-09 15:28:41.0)
***** Update an Element *****
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0 where s1_0.id=?
Hibernate: update emsi_students set birthday=?, name=?, last_connection=?, registration_n=?, still_active=? where id=?

***** Delete an Element *****
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0 where s1_0.id=?
Hibernate: delete from emsi_students where id=?
Hibernate: select count(*) from emsi_students s1_0
Count : 4
Hibernate: select s1_0.id,s1_0.birthday,s1_0.name,s1_0.last_connection,s1_0.registration_n,s1_0.still_active from emsi_students s1_0 where s1_0.id=?
Hibernate: delete from emsi_students where id=?
Hibernate: select count(*) from emsi_students s1_0
Count : 3
```

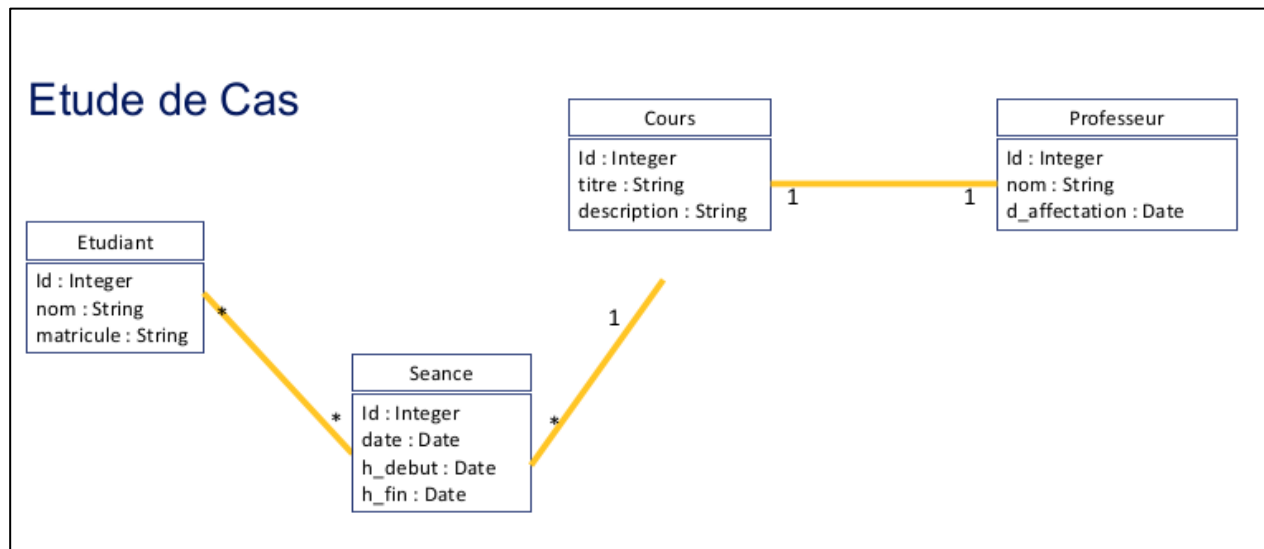
Résultat d'exécution dans phpMyAdmin



The screenshot shows the phpMyAdmin interface for the 'etudiant' database. The 'emsi_students' table is selected, and the SQL query 'SELECT * FROM `emsi_students`' is executed. The table contains 3 rows of data. The interface includes a sidebar with a tree view of databases and tables, and a main area with tabs for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, Tracking, and Triggers. The 'Browse' tab is active, showing the table structure and data.

	id	birthday	name	last_connection	registration_n	still_active
<input type="checkbox"/>	1	2023-04-09	Amine	2023-04-09 15:28:40.000000	A1	1
<input type="checkbox"/>	2	2023-04-09	Ilyas	2023-04-09 15:28:41.000000	A2	1
<input type="checkbox"/>	4	2023-04-09	Arif	2023-04-09 15:28:41.000000	A4	1

Création des Tables en suivant le diagramme de classe ci-dessous



Le code de l'entité "Course" ci-dessous illustre une relation One-to-One avec l'entité "Professor" et One-To-Many avec l'entité Session

```
package com.example.firstapp.Entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.Collection;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Course {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column(length = 25, nullable = false, unique = false)
    private String title;
    private String description;
    private int timing;
    @OneToOne(mappedBy = "course")
    private Professor professor;
    @OneToMany
    private Collection<Session> sessions;
}
```

Code de Repo de l'entité Course

```
package com.example.firstapp.repositories;

import com.example.firstapp.Entities.Course;
import org.springframework.data.jpa.repository.JpaRepository;
```



```
public interface CourseRepository extends JpaRepository<Course, Integer> {
}
```

Code de l'entité Professor

```
package com.example.firstapp.Entities;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.CreationTimestamp;

import java.util.Date;

@Entity @Data @AllArgsConstructor @NoArgsConstructor
public class Professor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column(length = 30, nullable = false)
    private String fullName;
    @Temporal(TemporalType.DATE) @CreationTimestamp
    private Date assignmentDate;
    @OneToOne
    private Course course;
}
```

Repository de l'entité "Professor"

```
package com.example.firstapp.repositories;

import com.example.firstapp.Entities.Professor;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProfessorRepository extends JpaRepository<Professor, Integer>
{
}
```

Le code de l'entité "Session" définit une relation "ManyToOne" avec les entités "Student" et "Course". Cette relation signifie qu'une session est associée à un seul étudiant et à un seul cours, mais qu'un étudiant peut être associé à plusieurs sessions pour différents cours.

```
package com.example.firstapp.Entities;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.Date;

@Entity @Data @AllArgsConstructor @NoArgsConstructor
public class Session {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Temporal(TemporalType.DATE)
    private Date date;
    @Temporal(TemporalType.TIME)
    private Date start_time;
    @Temporal(TemporalType.TIME)
    private Date end_time;
}
```

```

    @ManyToOne
    private Student student;
    @ManyToOne
    private Course course;
}

```

Repository de l'entité "Session"

```

package com.example.firstapp.repositories;

import com.example.firstapp.Entities.Session;
import org.springframework.data.jpa.repository.JpaRepository;

public interface SessionRepository extends JpaRepository<Session, Integer> {
}

```

Pour exprimer la relation OneToMany dans la classe "Student", il suffit d'ajouter un attribut de type collection nommé "sessions". Cette collection représente l'ensemble des sessions liées à l'étudiant. La relation OneToMany indique que chaque étudiant peut avoir plusieurs sessions, mais chaque session est liée à un seul étudiant. En ajoutant cet attribut, nous pouvons facilement accéder aux sessions d'un étudiant et effectuer des opérations telles que l'ajout ou la suppression de sessions.

```

@OneToMany
private Collection<Session> sessions;

```

Résultat d'exécution la création des tables à réussi

```

Hibernate: create table course (id integer not null auto_increment, description varchar(255), timing integer not null, title varchar(25) not null, primary key (id)) engine=InnoDB
Hibernate: create table course_sessions (course_id integer not null, sessions_id integer not null) engine=InnoDB
Hibernate: create table emsi_students (id integer not null auto_increment, birthday date, name varchar(30) not null, last_connection datetime(6), registration_n varchar(255), still_acti
Hibernate: create table emsi_students_sessions (student_id integer not null, sessions_id integer not null) engine=InnoDB
Hibernate: create table professor (id integer not null auto_increment, assignment_date date, full_name varchar(30) not null, course_id integer, primary key (id)) engine=InnoDB
Hibernate: create table session (id integer not null auto_increment, date date, end_time time, start_time time, course_id integer, student_id integer, primary key (id)) engine=InnoDB
Hibernate: alter table course_sessions drop index UK_7130ux8p1pouldavghco81b6p
Hibernate: alter table course_sessions add constraint UK_7130ux8p1pouldavghco81b6p unique (sessions_id)
Hibernate: alter table emsi_students drop index UK_a3vjvtvLbany4513mbr04y7hsu
Hibernate: alter table emsi_students add constraint UK_a3vjvtvLbany4513mbr04y7hsu unique (registration_n)
Hibernate: alter table emsi_students_sessions drop index UK_kucri9b21ophrct9ktj92580
Hibernate: alter table emsi_students_sessions add constraint UK_kucri9b21ophrct9ktj92580 unique (sessions_id)
Hibernate: alter table course_sessions add constraint FKediong3hi0ata64gp5p9n1ueu foreign key (sessions_id) references session (id)
Hibernate: alter table course_sessions add constraint FKawiofwq6gt04l0upj2h5y62er foreign key (course_id) references course (id)
Hibernate: alter table emsi_students_sessions add constraint FK7817b2thyj930mwer97In0v8l foreign key (sessions_id) references session (id)
Hibernate: alter table emsi_students_sessions add constraint FKgvoL9b1x8c2w0nt4l929ultc2 foreign key (student_id) references emsi_students (id)
Hibernate: alter table professor add constraint FKtn8tj1sjq1m5h52qofiq1jwsq foreign key (course_id) references course (id)
Hibernate: alter table session add constraint FK5ibwg70x5r9hl56pge5yg2vzb foreign key (course_id) references course (id)
Hibernate: alter table session add constraint FKrc7q3m27otttd43qgghk90p foreign key (student_id) references emsi_students (id)

```

Les tables créées dans la base de données MySQL avec les différentes relations

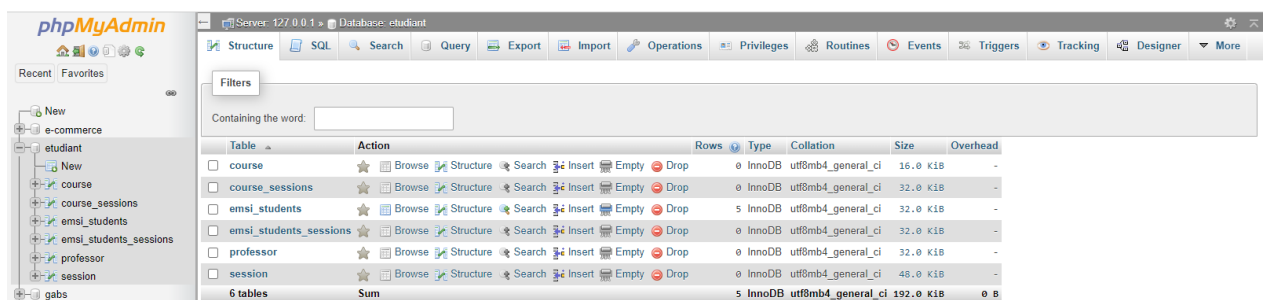


Table	Action	Rows	Type	Collation	Size	Overhead
course	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 K	-
course_sessions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 K	-
emsi_students	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	32.0 K	-
emsi_students_sessions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 K	-
professor	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 K	-
session	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 K	-
6 tables	Sum	5	InnoDB	utf8mb4_general_ci	192.0 K	0 B

