

# Projet Java : Gestion des études

Présenté par :  
EL-ASMI BAKR  
EL MKADMI AYMANE  
BOUATTANE AYMAN

2024/2025

# Connexion à la base de données mysql

**Aiven** : une interface web fournie par la plateforme **Aiven**, une solution cloud qui permet de gérer et d'héberger différents services de bases de données

## Pourquoi ?

- **Gestion simplifiée des services** : Vous pouvez créer, configurer et gérer des instances de bases de données (MySQL) ou d'autres services avec quelques clics.
- **Console en ligne** : Elle permet d'accéder directement à votre base de données via une console en ligne, sans avoir besoin d'installer de client local.

# Création du Compte :

 **aiven**  
CONSOLE

Home Projects Tools Billing Support Admin My Org

fatielasmi-6389  
mysql-3f41b7a8

← Back to project

Overview

Integrations

Metrics

AI insights

Logs

Current queries

Users

Databases

Backups

Service settings

## Connect to mysql-3f41b7a8

Connect with:  
mysql

Connect to the Aiven for MySQL® database using mysql. For more information see our [documentation](#).

1. Install the [mysql client](#).
2. From your terminal, run the following code:

```
$ mysql --user avnadmin --password=***** --host mysql-3f41b7a8-fatielasmi-6389.g.aivencloud.com --port 16923 defaultdb
```
3. To confirm that the connection is working, issue the following query:

```
$ select 1 + 2 as three;
```
4. The output should look like the following if the connection was successful:

```
+-----+  
| three |  
+-----+  
| 3 |  
+-----+  
1 row in set (0.0539 sec)
```

Done

# Création des TABLES :

Ma base de données contient les tables suivantes qui peuvent être affichées en utilisant la commande

**SHOW TABLES;**

```
mysql> SHOW TABLES;
```

+-----+-----+	
Tables_in_defaultdb	
+-----+-----+	
Employe	
Etudiant	
Examen	
Filiere	
InscriptionAdministrative	
Module	
Notes	
Personne	
+-----+-----+	

```
8 rows in set (0.09 sec)
```

# Connexion du projet avec la base de données

```
public class DatabaseConnection { no usages
    private static final String URL = "jdbc:mysql://mysql-3f41b7a8-fatielasmi-6389.g.aivencloud.com:16923/defaultdb?ssl-mod
    private static final String USER = "avnadmin"; 1 usage
    private static final String PASSWORD = "AVNS_JuwWtOPbKidNzgzxenz"; 1 usage

    public static Connection connect() { no usages
        Connection connection = null;
        try {
            // Charger le pilote JDBC
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Établir la connexion
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
```

Ce code définit une méthode pour se connecter à la base de données MySQL distante (hébergée sur le cloud via Aiven), en utilisant un url, un mot de passe et un utilisateur spécifiques. l'url contient les informations Le type de base de données (jdbc:mysql://)., L'hôte et le port où la base de données MySQL est accessible et le nom de la base de données que j'ai utilisé (defaultdb).

# Inscriptions

La logique de l'inscription que j'ai adopté est la suivante :

- Interface Inscription
- La classe InscriptionAdministrative
- La classe InscriptionPédagogique

# Inscriptions : Interface Inscription

```
package Inscriptions;
```



```
public interface Inscription <Object> { 2 usages 2 implementations  
    abstract public Boolean inscriptionValidee(Object o); 1 usage 2 implementations  
}
```

La méthode abstraite inscriptionValidee pour retourner si l'inscription est validée ou pas et va être redéfinie dans les autres classes

# Inscriptions : La classe Inscription Administrative

Les attributs de la classe sont :



```
public class InscriptionAdministrative implements Inscription<Float> { 6 usages
    protected static float montantApayer; 2 usages
    protected Boolean reussirBac; 3 usages
    protected float montantPaye; 3 usages
    protected Personne personne=new Personne(); 2 usages
    protected int age; 3 usages
    protected Boolean validation; 4 usages
```

**MontantApayer** est le montant fixé par l'école, tandis que **montantPayé** correspond à la somme versée par l'étudiant.



## Inscriptions : La classe Inscription Administrative

```
public Boolean inscriptionValidee(Float montantApayer) { 1 usage
    if(age<22 && reussirBac==true && montantPaye==montantApayer) {
        //age<22
        validation=true;
        return true;
    }
    validation=false;
    return false;
}
```

Plus le constructeur, les getters et les setters , j'ai redéfini la méthode inscriptionValidee. Pour effectuer une inscription administrative, il faut avoir moins de 22 ans, avoir obtenu son Bac et avoir réglé les frais de scolarité

# Inscriptions : La classe Inscription Pédagogique

```
package Inscriptions;
import java.util.ArrayList;
import Filieres.Etudiant;
import Filieres.Filiere;
import EquipePeda.Personne;

public class InscriptionPeda implements Inscription<InscriptionAdministrative> { 16 usages
    protected Filiere choixFiliere; 7 usages
    protected Boolean validationAdmin; 4 usages
    protected Boolean validation; 4 usages
    protected Personne personne=new Personne(); 6 usages
```

Après avoir validé l'inscription administrative, il faut choisir la filière à y accéder pour valider l'inscription pédagogique, raison pour laquelle j'ai choisi ces attributs

# Inscriptions : La classe Inscription Pédagogique

```
public Boolean inscriptionValidee(InscriptionAdministrative i) { 10 usages
    ArrayList<Etudiant> etud=new ArrayList<>();
    etud=choixFiliere.getEtudiants();
    validationAdmin=i.inscriptionValidee(InscriptionAdministrative.getMontantApayer());
    int capacite=choixFiliere.getCapacite();
    if(etud.size()<capacite && validationAdmin) {
        personne=new Etudiant(i.getPersonne().getNom(),i.getPersonne().getPrenom(),i.getPersonne().getId());
        etud.add((Etudiant)personne);
        ((Etudiant) personne).setFiliere(choixFiliere);
        validation=true;
    }
    return true;
}

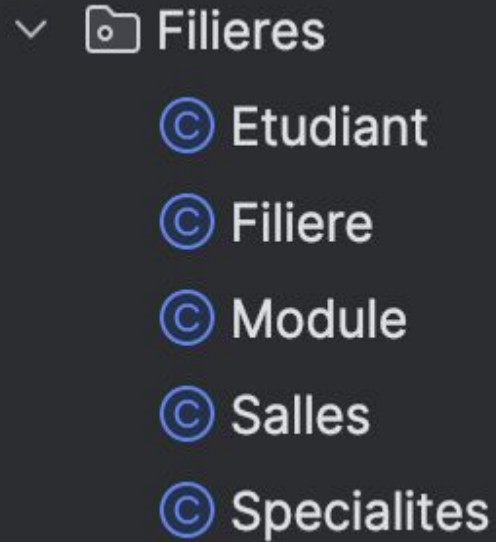
validation=false;
return false;
}
```

La méthode inscriptionValidee est définie de manière différente. En effet, pour accéder à une filière, il doit y avoir une place disponible, c'est-à-dire que le nombre d'étudiants doit être inférieur à la capacité de celle-ci. Si cette condition est remplie, l'inscription pédagogique est validée et l'étudiant est ajouté à la liste des étudiants de la filière.

# Filières

Dans ce package, j'ai créé plusieurs classes :

- Etudiant
- Filière
- Module
- Salles
- Specialites




# Filières : La classe Etudiant

Les attributs de la classe Filière sont :

```
protected String sexe; 2 usages
protected String filiereBac; 2 usages
protected Filiere filiere; 3 usages
protected String cne; 3 usages
protected HashMap<String, Float> notesMatières = new HashMap<>(); 4 usages
protected HashMap<Integer, HashMap<Module, Float>> noteModules = new HashMap<>(); 10
```

Cette classe permet également de gérer les informations des étudiants en déterminant leurs notes maximale, minimale et moyenne. Elle prend en charge l'enregistrement et la manipulation des notes par matière et par module, ainsi que l'interaction avec une base de données MySQL pour récupérer les données des étudiants et leurs notes.

# Filières : La classe Filière



```
public class Filiere { 10 usages
    protected String nomFiliere; // GI 4 usages
    protected ArrayList<Employe> equipe = new ArrayList<>(); // HashMap 4 usages
    protected HashMap<Integer, LinkedHashMap<String, String>> emplois = new HashMap<>();
    protected int anneeOuverture; 4 usages
    protected int idPromo; // 15 2 usages
    protected Specialites specialite; 2 usages
    protected int capacite; 2 usages
    protected ArrayList<Etudiant> etudiants = new ArrayList<>(); 7 usages
```

La classe Filiere est conçue pour gérer tous les aspects liés à une filière dans une école d'ingénieurs, tels que :

- La gestion des informations générales (nom, spécialité, capacité, année d'ouverture, etc.).
- La gestion des étudiants (ajout, suppression, tri par nom, etc.).
- La gestion des emplois du temps (ajout et affichage des plannings hebdomadaires).

# Filières : La classe Module

```
public class Module { 67 usages
    protected String nomModule; 4 usages
    protected LinkedHashMap<String,String[][]> matieres =new LinkedHashMap<>(); 4 usages
    protected int nombreElement; 3 usages
```

La classe Module est conçue pour gérer les informations associées à un module académique dans un programme d'étude, tels que :

- Le nom du module.
- Le nombre d'éléments pédagogiques associés au module.
- La liste des matières associées à ce module, avec leur volume horaire pour chaque type d'activité pédagogique.

# Filières : La classe Salles

```
public class Salles { 8 usages
    protected static HashMap<Integer, Boolean> num= new HashMap<>(); 10 usages
    protected static String [] activite= {"CM", "TD", "TP", "AP"}; 2 usages
    protected static HashMap<String, ArrayList<Integer>> salle=new HashMap<>(); 10 usages
```

La classe Salles permet de gérer les salles. Elle fournit des fonctionnalités pour :

- Organiser les salles par type d'activité pédagogique ("cour magistral, travaux pratiques, travaux dirigés, activités pratiques")
- Suivre l'état des salles (disponible ou occupée).
- Affecter des salles pour des activités spécifiques.



# Filières : La classe Specialite

```
public class Specialites { 3 usages
    private static final long serialVersionUID = 1L; no usages
    protected String domaine; 3 usages
    protected String nomSpecialite; 2 usages
    protected HashSet<Module> modules=new HashSet<>(); 3 usages
```

La classe Specialites représente une spécialité académique dans une filière, comprenant :

- Un domaine (exemple : "Informatique").
- Un nom (exemple : "Développement full-stack").
- Une collection de modules liés à cette spécialité.

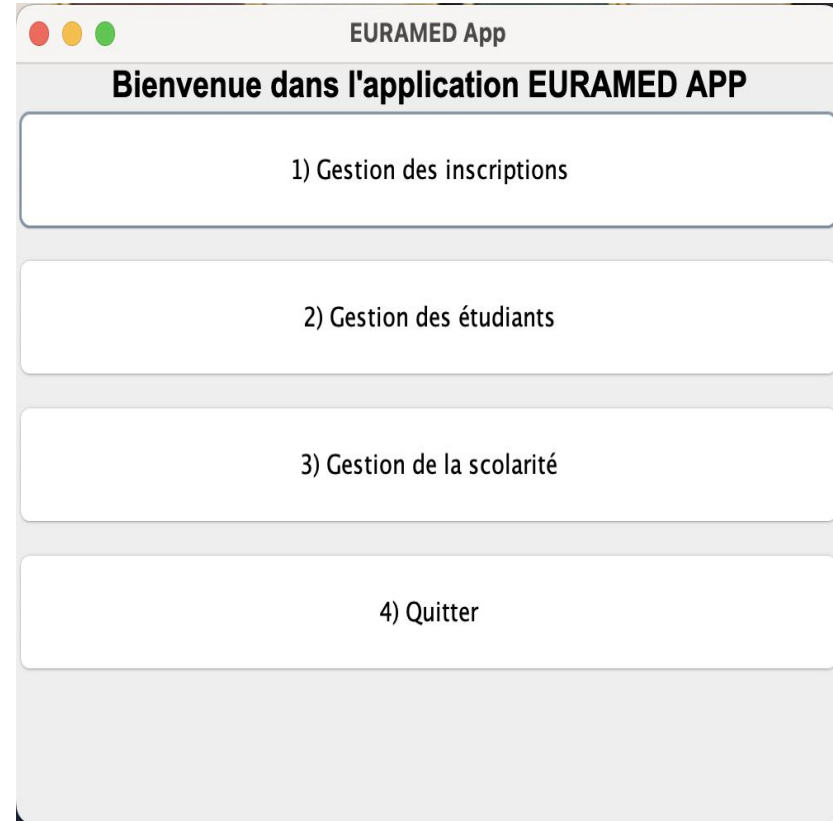
Cette classe permet de regrouper et gérer les modules qui font partie d'une spécialité donnée, tout en définissant son domaine d'étude.

# Interface utilisateur

Voici une version corrigée de votre texte :

Une fois l'application lancée, un menu s'affiche pour permettre de choisir entre les quatre fonctionnalités suivantes :

- **\*\*Gestion des inscriptions\*\*** : permet d'effectuer l'inscription administrative et pédagogique.
- **\*\*Gestion des étudiants\*\*** : permet de récupérer les informations des étudiants, telles que la note maximale, la note minimale, la moyenne de l'année, etc.
- **\*\*Gestion de la scolarité\*\*** : permet d'afficher les emplois du temps et les salles pour chaque semaine.
- **\*\*Quitter l'application\*\*** : pour fermer le programme.



# Interface utilisateur

Voici une version corrigée et reformulée de votre texte :

En choisissant le premier volet, Gestion des inscriptions, un formulaire s'affiche pour permettre la saisie des informations suivantes concernant l'étudiant, nécessaires à l'inscription administrative :

- Nom
- Prénom
- Réussir son Bac
- Date de naissance
- Montant payé cad frais de scolarité

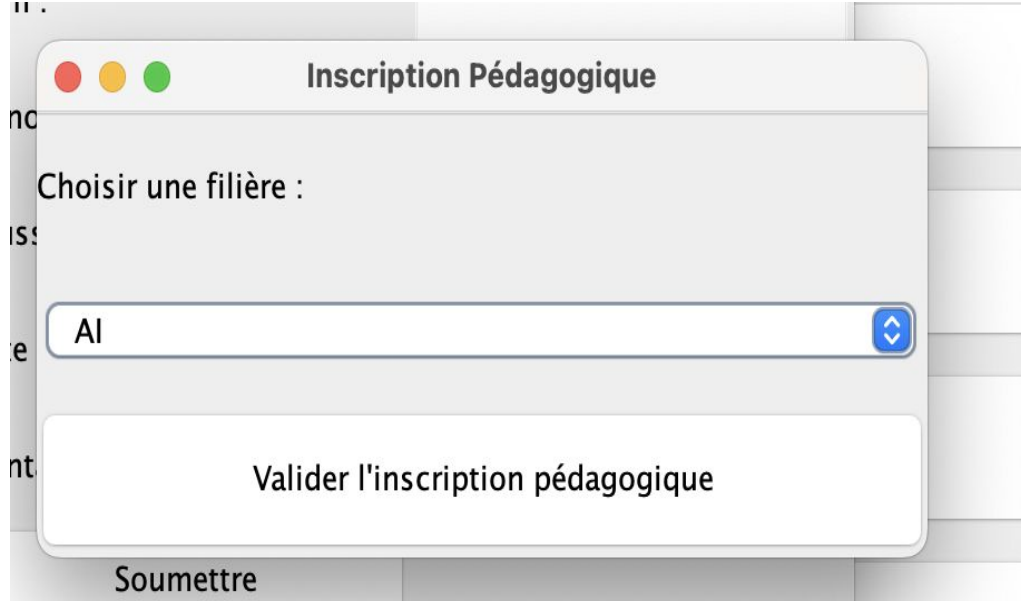
Une fois les informations saisies, elles sont enregistrées dans la base de données pour compléter l'inscription pédagogique.

The image shows a screenshot of a mobile application window titled "EURAMED App" with a subtitle "Inscription Administrative". The form contains the following fields and controls:

- Nom :** A text input field.
- Prénom :** A text input field.
- Réussir BAC (oui/non) :** A dropdown menu with "oui" selected and a blue arrow icon on the right.
- Date de Naissance (année) :** A text input field.
- Montant Payé :** A text input field.
- Soumettre** : A large white button with a black border at the bottom center.

# Interface utilisateur

L'inscription pédagogique permet à l'étudiant de choisir la filière à laquelle il souhaite être affecté. Dans cette interface, l'étudiant a le choix entre trois filières : AI, Data et Cloud.. Ces choix sont présentés dans une liste déroulante, et l'étudiant sélectionne la filière qui lui convient, puis valide l'inscription.



The image shows a web application window titled "Inscription Pédagogique". Inside the window, there is a label "Choisir une filière :". Below this label is a dropdown menu with "AI" selected. To the right of the dropdown is a blue button with a downward arrow. Below the dropdown is a large white button with the text "Valider l'inscription pédagogique". At the bottom of the window, there is a "Soumettre" button.

# Interface utilisateur : formulaire d'inscription Pédagogique

```
[mysql> SELECT * FROM Filiere;
```

id	nom
1	AI
2	Data
3	Cloud

```
3 rows in set (0.09 sec)
```

```
[mysql> SELECT * FROM Etudiant;
```

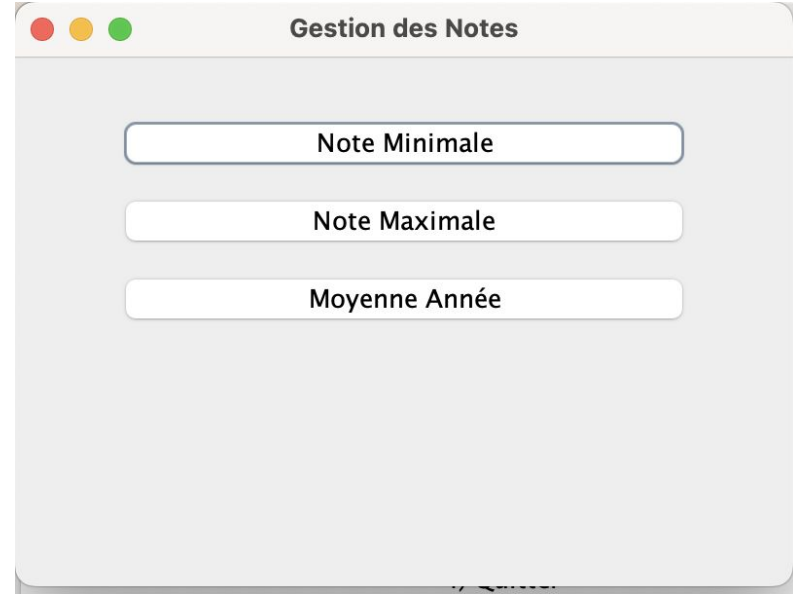
id	filiereId
1	3

```
1 row in set (0.07 sec)
```

Une fois l'inscription pédagogique validée, un étudiant est ajouté dans la liste des étudiants ayant comme filière celle sélectionnée dans la barre déroulante de l'inscription.

# Interface utilisateur : Gestion des étudiant

Pour le deuxième volet, Gestion des étudiants, il permet de choisir entre trois options : rechercher la note minimale d'un étudiant, la note maximale, ou calculer la moyenne de l'année.



# Interface utilisateur : Gestion des étudiant

En sélectionnant l'option souhaitée, il faut saisir le nom et le prénom de l'étudiant afin d'obtenir le résultat recherché.

Voici un exemple du cas de la note minimale :  
Lorsqu'on choisit l'option "Note minimale", l'utilisateur doit entrer le nom et le prénom de l'étudiant. Ensuite, l'application recherche dans la base de données les différentes notes de l'étudiant pour chaque matière et affiche la note minimale obtenue pendant l'année.

