2018 - 2019



**gESTION D’EMPLOI DU TEMPS**

Joris MURATET

**Rapport de Projet BDD2**

**JORIS MURATET EVA LAMESA ALEXANDRE OUDNI**

Sommaire

Table des matières

[Sommaire 1](#_Toc533269581)

[Création d’une base de données 2](#_Toc533269582)

[Dictionnaire de Données 3](#_Toc533269583)

[Connexion à la base de données 6](#_Toc533269584)

[Recherche des élèves par promotion ou groupe de TD 7](#_Toc533269585)

[Contraintes et mise en place des Triggers 11](#_Toc533269586)

[Emploi du temps 14](#_Toc533269587)

[Connexion aux Profils 15](#_Toc533269588)

[Ajouter un cours 17](#_Toc533269589)

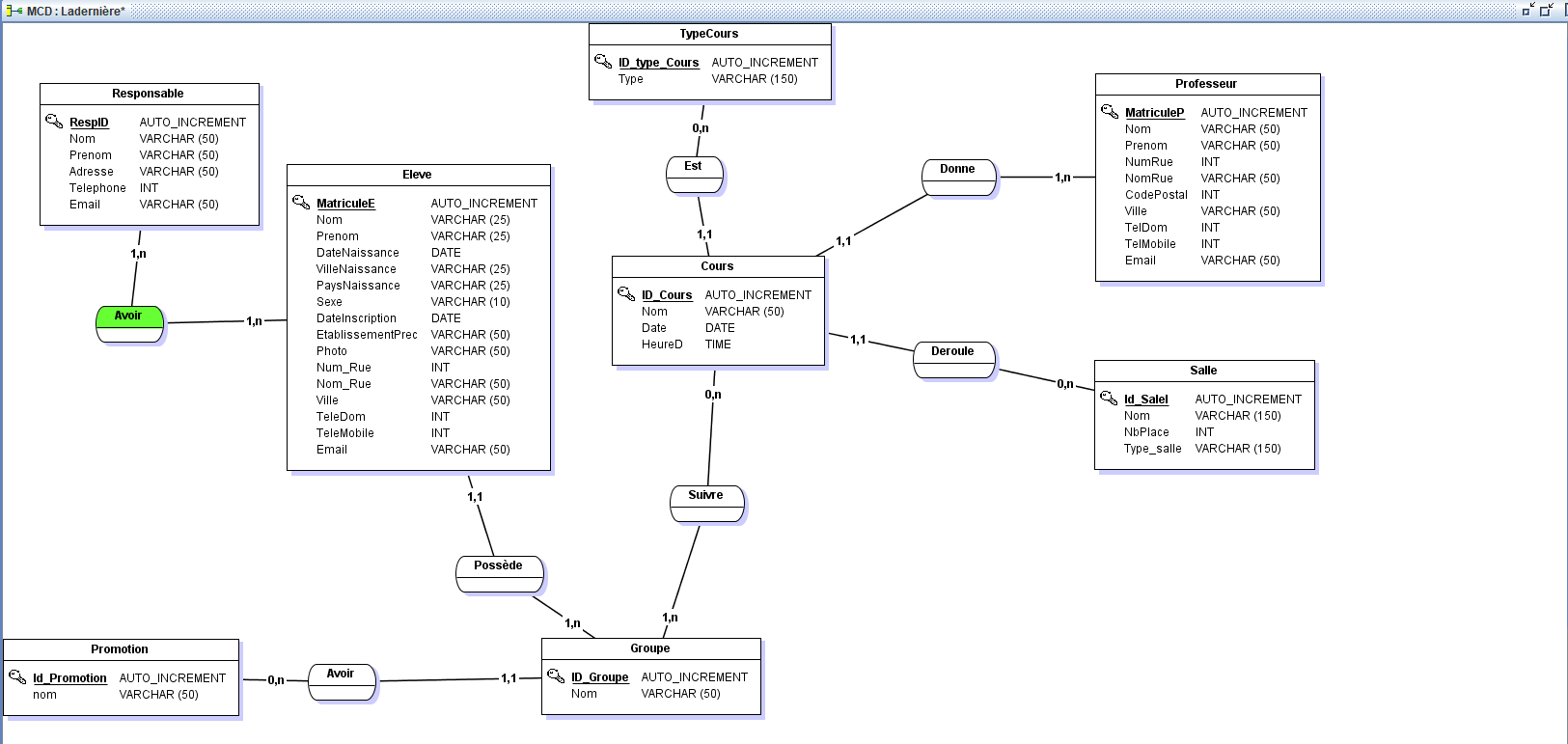
[Modifier la salle 20](#_Toc533269590)

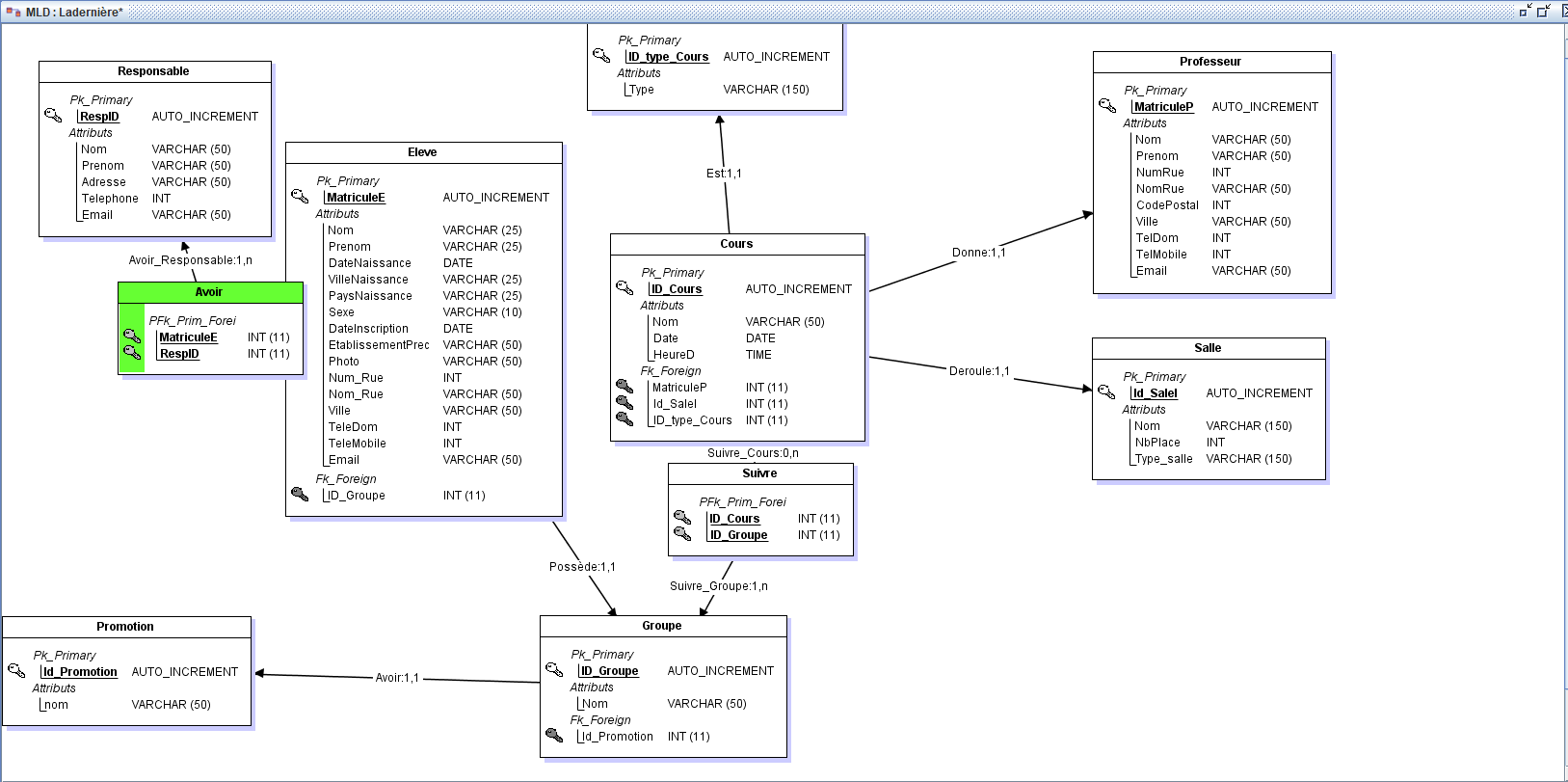
[Confirmation avant de quitter 21](#_Toc533269591)

*Le but de ce projet était de réaliser un outil de gestion d’emploi du temps pour une administration afin de gérer la planification des cours au sein d’une école.*

Création d’une base de données

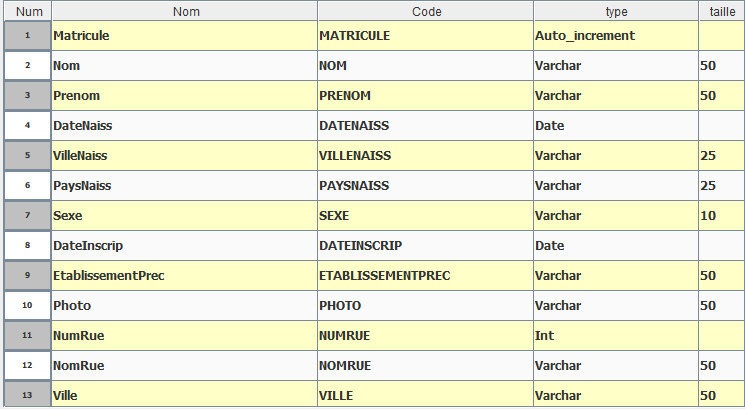
Afin d’interagir avec les différentes entités de notre projet, nous avons réalisé la MCD grâce à Jmerise.

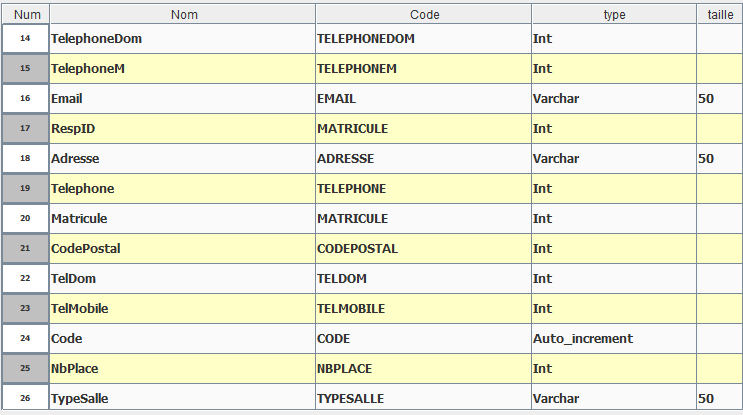
**MCD :**

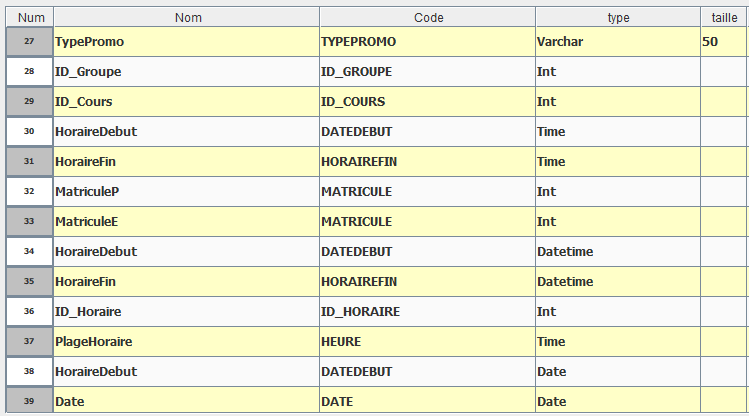
**MDL :**

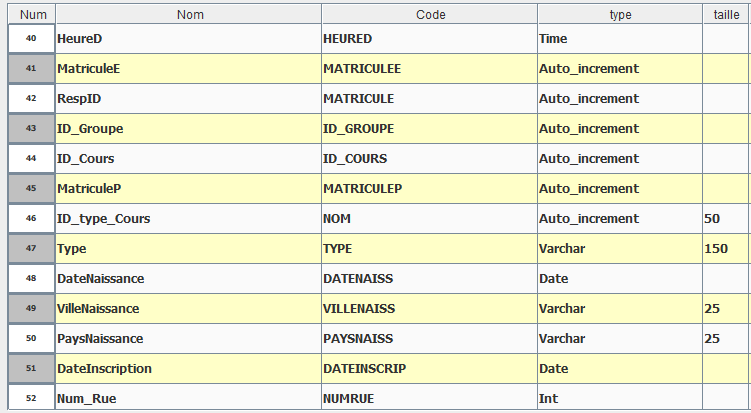
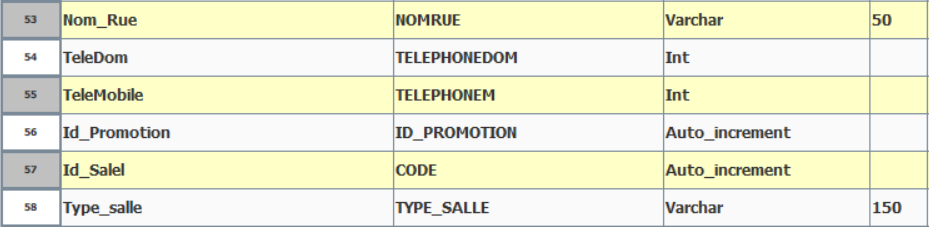
Dictionnaire de Données

Afin de voir tous les attributs de nos tables de manière plus précise, voici le dictionnaire de données :

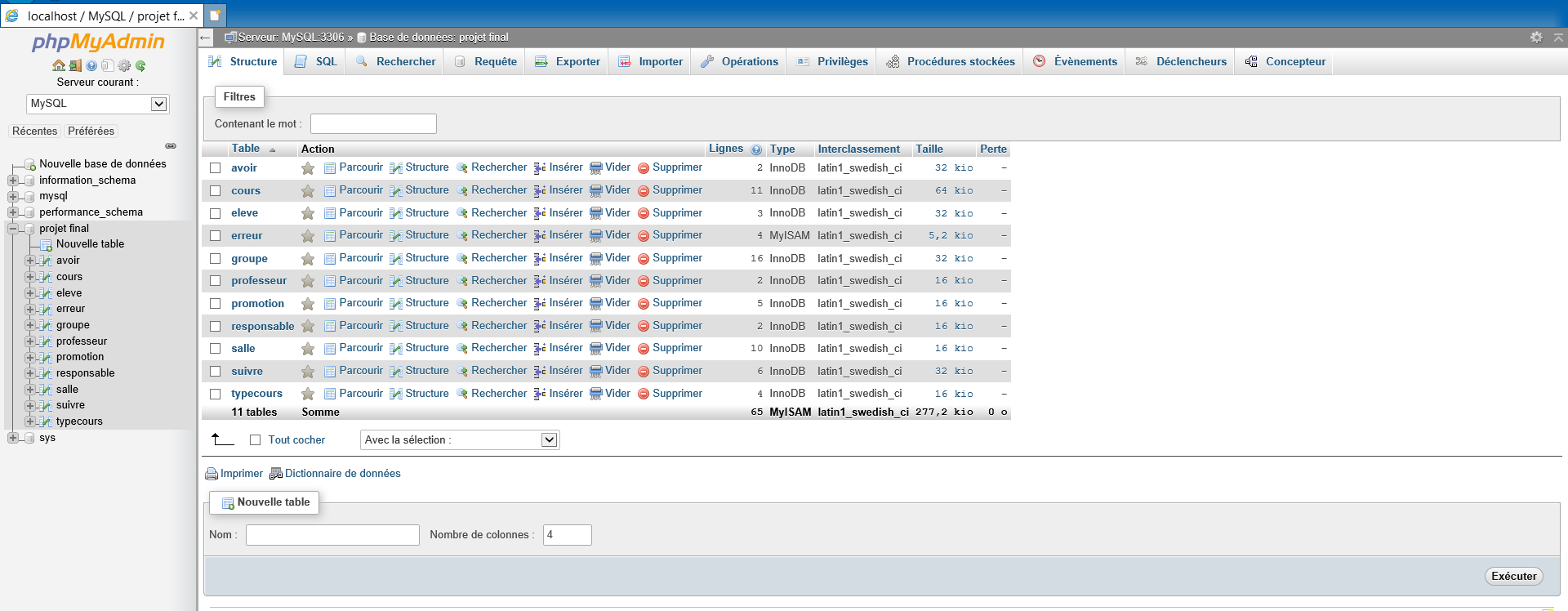








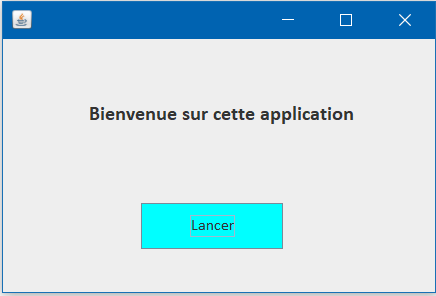
Nous avons ensuite généré le code SQL puis nous l’avons exécuté dans phpMyAdmin en utilisant Wampserver64 pour nous connecter de manière locale à notre base de données.



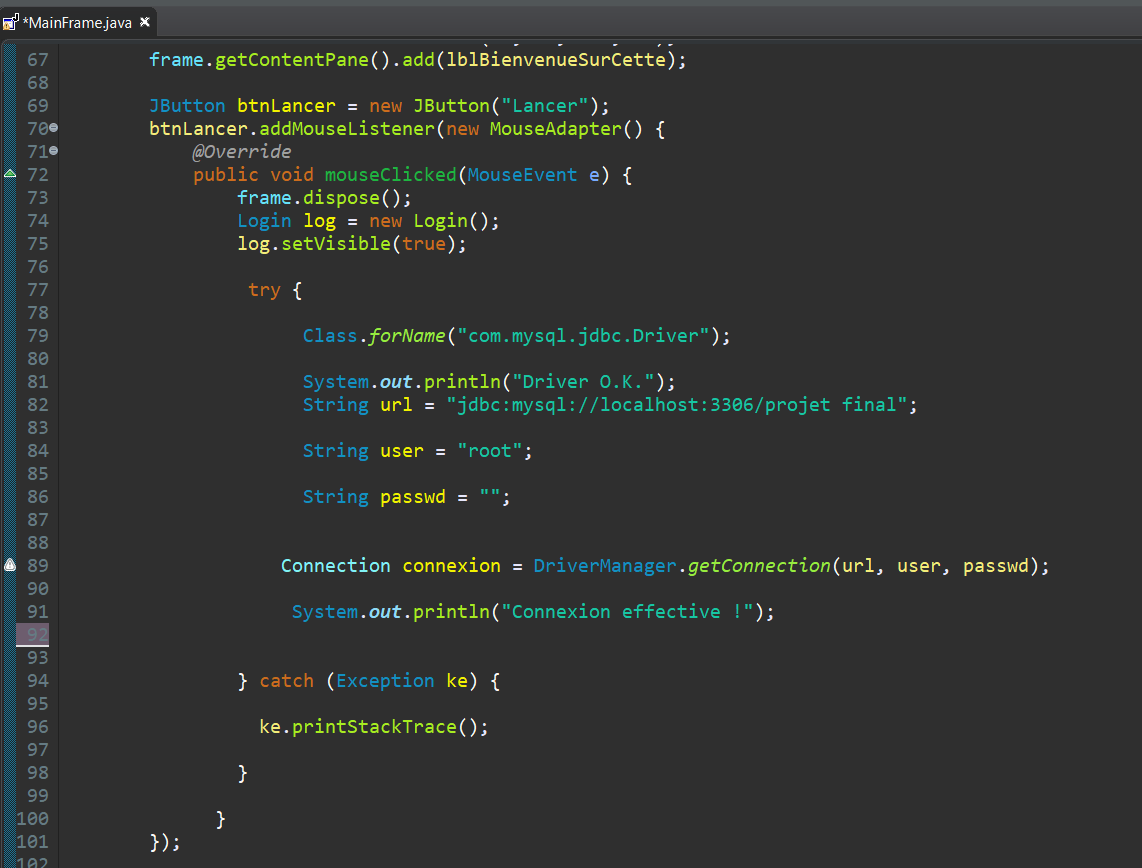
Connexion à la base de données

Après avoir réalisé notre base de données, le but a été de réaliser une application permettant d’interagir avec la base de données afin d’en récupérer des informations et/ou d’en ajouter. Il était donc nécessaire de se connecter à cette dernière.

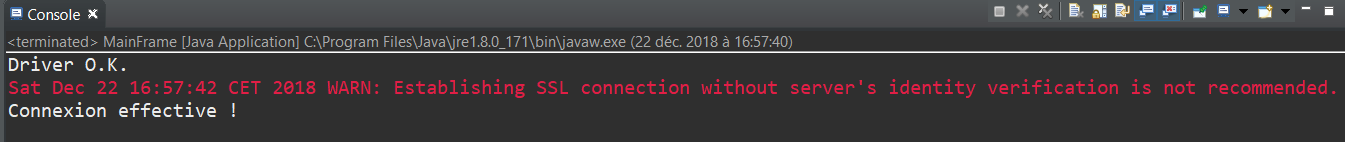
Nous avons préalablement ajouté à notre projet le Driver JDBC qui nous permet la connexion à la base de données. Nous avons réalisé une classe MainFrame qui est donc notre classe principale qui permet de lancer l’application finale. Nous avons réalisé les interfaces graphiques à l’aide de WindowBuilder.

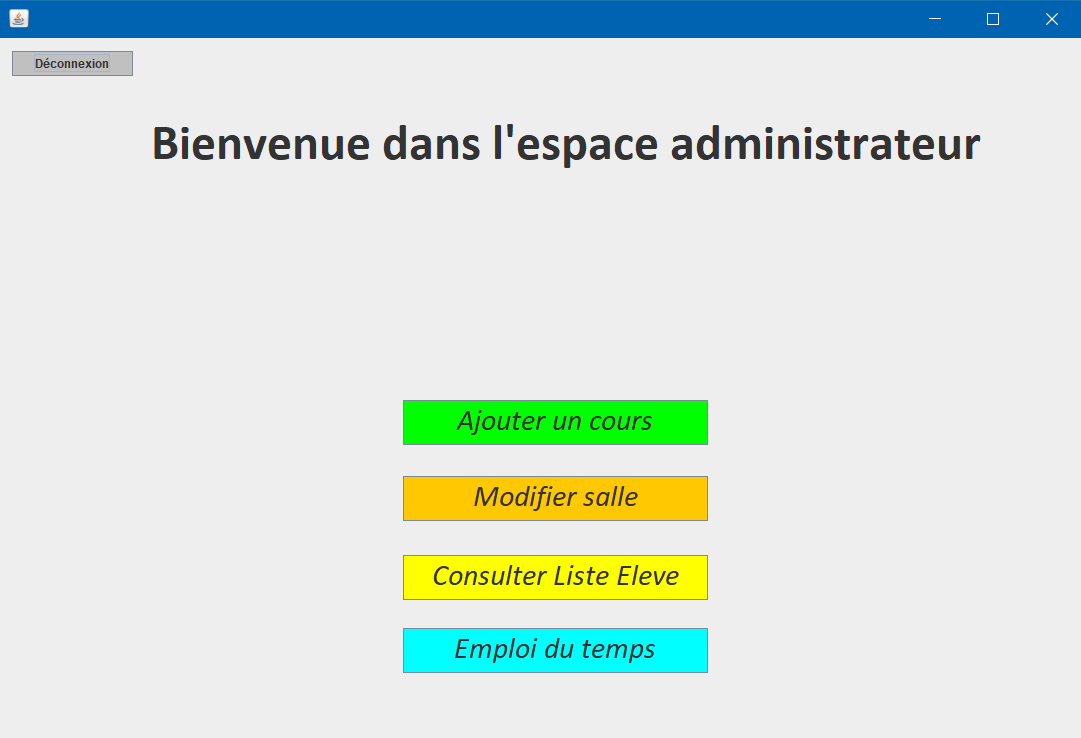


Le bouton « Lancer » permet de se connecter à la base de données car il exécute le code suivant :



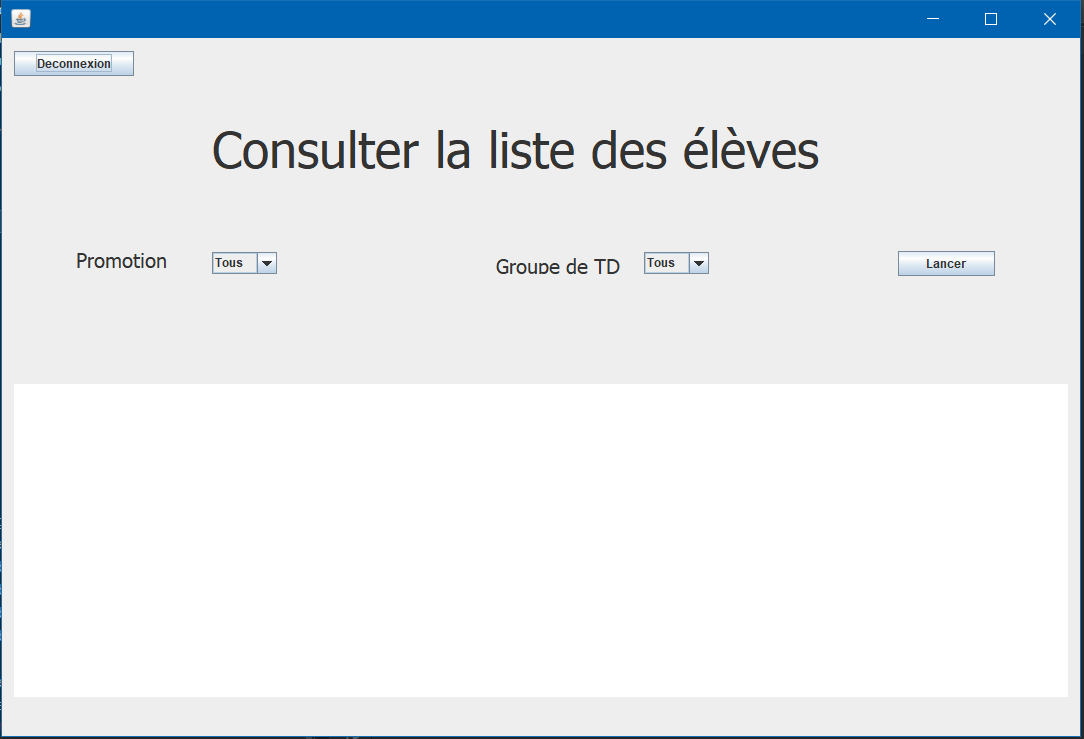
Les informations contenues dans le try/catch ci-dessus permettent dans un premier temps de prendre en compte le driver pour JAVA, un message s’affiche sur la console signifiant que le Driver est opérationnel. Dans un second temps nous nous connectons à la base de données par le biais de PHPmyadmin puis nous testons une connexion et « Connexion effective » s’affiche pour signifier que la connexion à bien eu lieu. Voici ce que nous renvoie la console :



Recherche des élèves par promotion ou groupe de TD

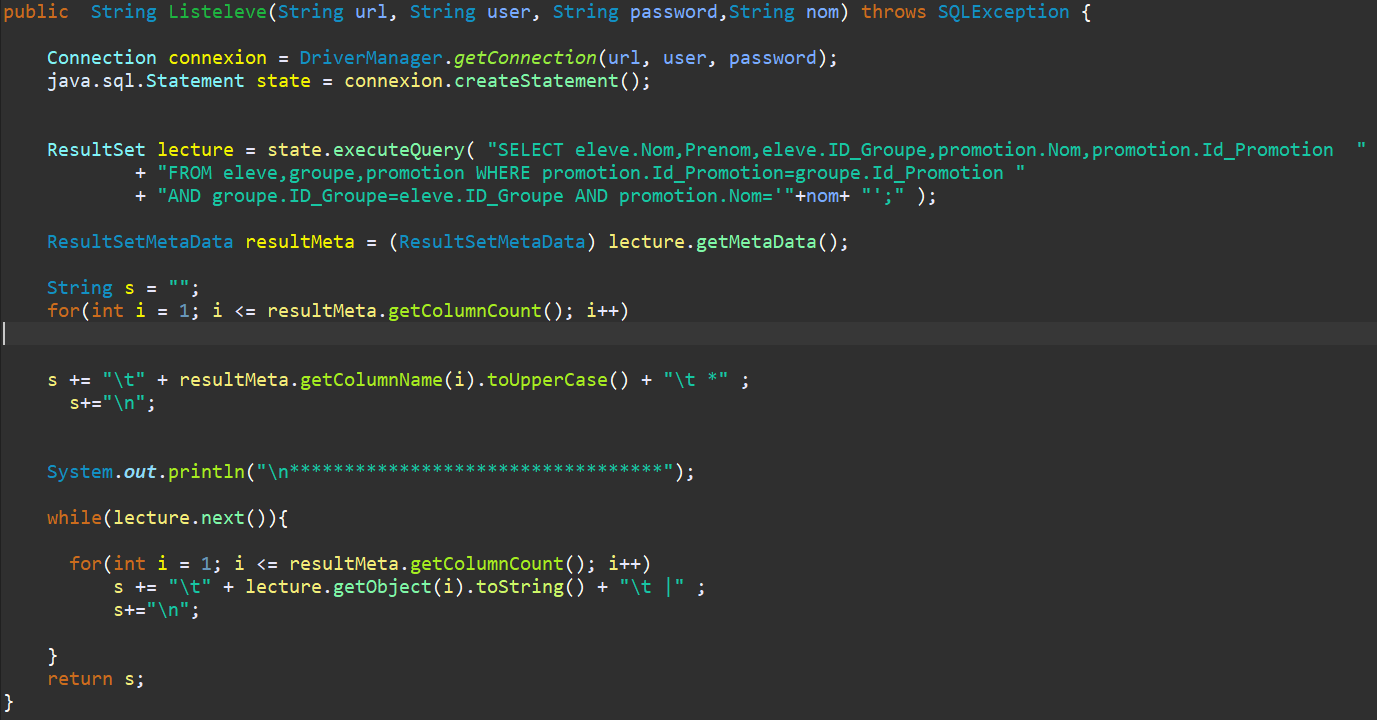
Une des premières fonctionnalités demandées était l’affichage des différentes listes d’élèves par promotion ou groupe de TD. Pour cela nous avons réalisé deux classes, une pour l’interface graphique et une pour l’exécution des requêtes.

La classe de l’interface graphique est Consulter.



On peut donc choisir la promotion et le groupe de TD. La partie en blanc correspond à l’affichage.

Afin d’afficher ce que l’on souhaite on utilise les méthodes de la classe Trieleve.

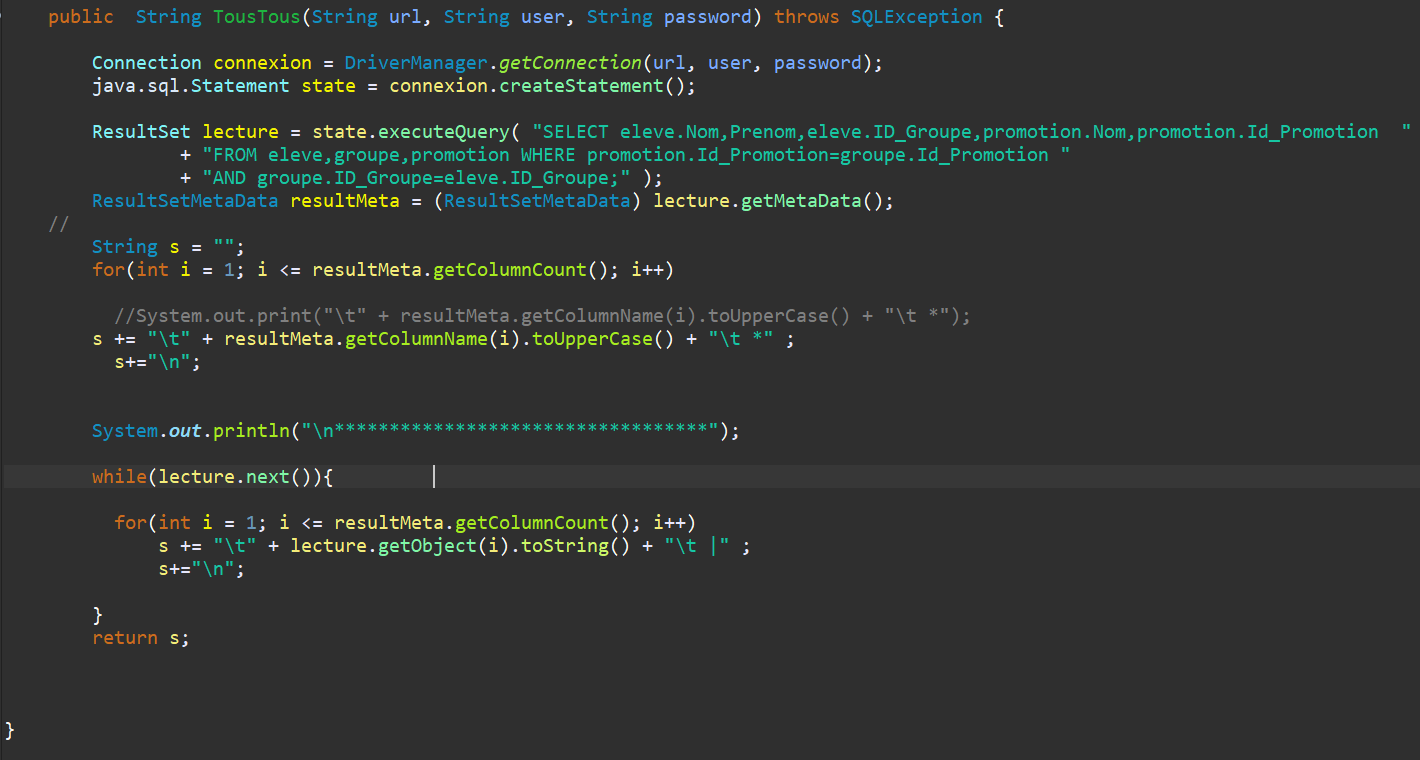


Dans cette méthode on se connecte à la Base et on exécute la requête qui nous permet de récupérer les informations relatives à chaque élève dans le cas où l’on choisit d’afficher tous les élèves d’une promotion. (Groupe est donc égal à Tous). Ensuite, la fin du code permet d’afficher toutes les lignes et les colonnes ainsi récupérées de la table.



Cette deuxième méthode fait sensiblement la même chose mais elle permet d’exécuter la requête en tenant compte d’une promotion en particulier et d’un groupe en particulier.

Enfin, la dernière méthode de cette classe est :

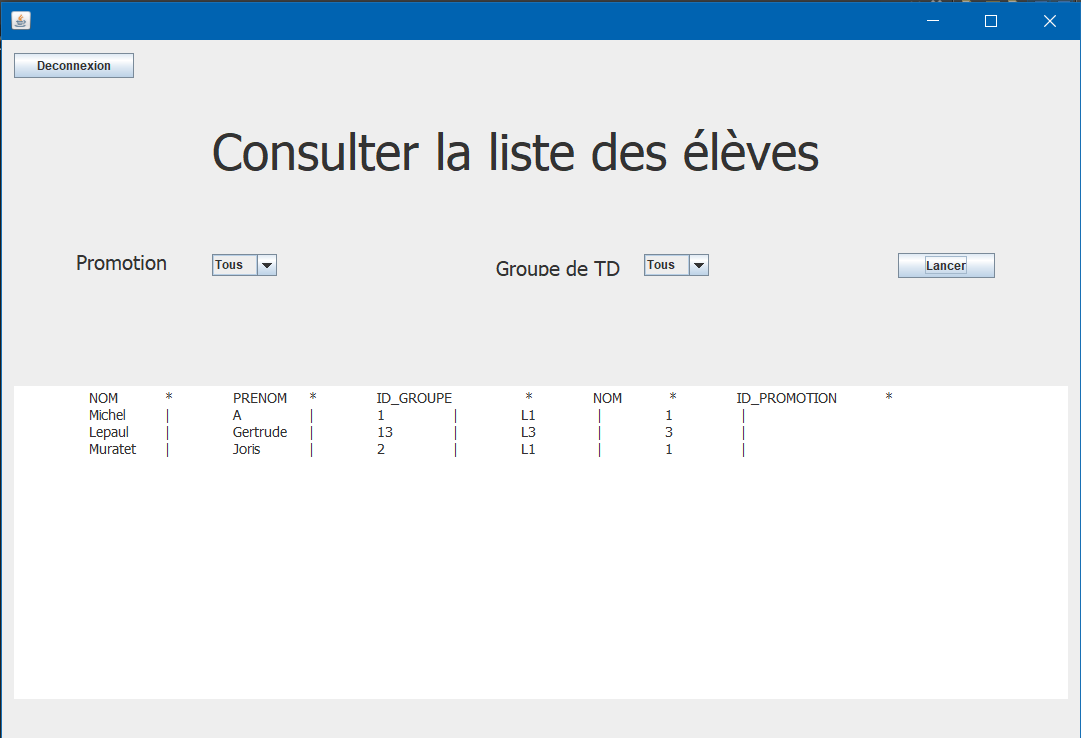


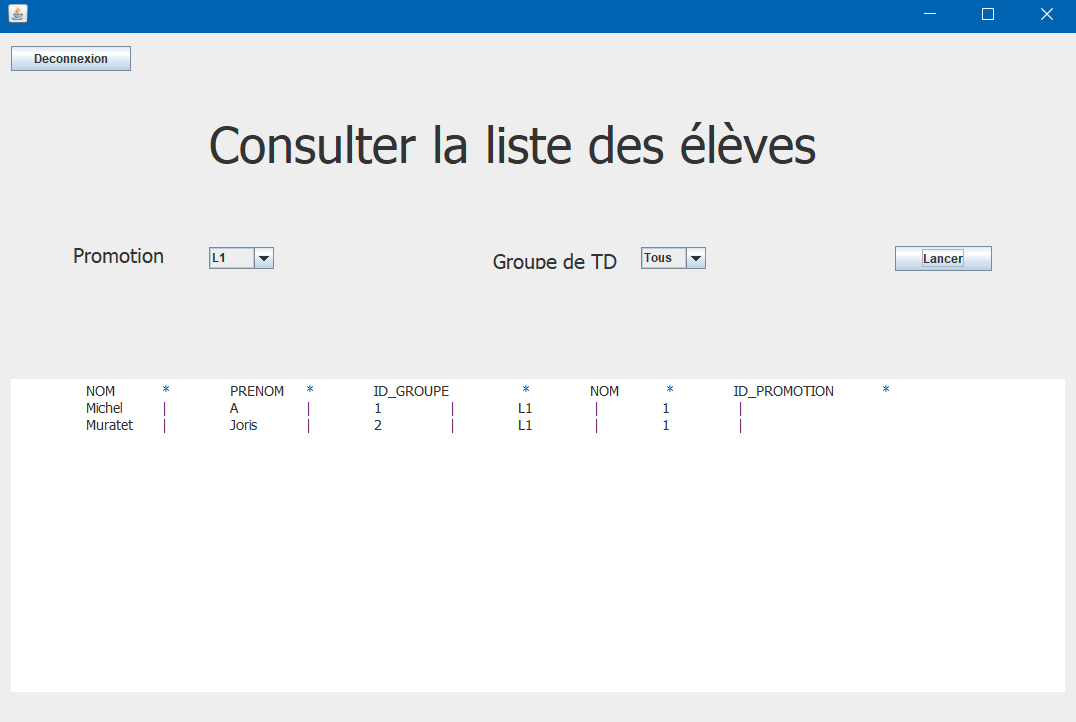
Elle permet d’exécuter la requête pour récupérer tous les élèves présents dans la table eleve.

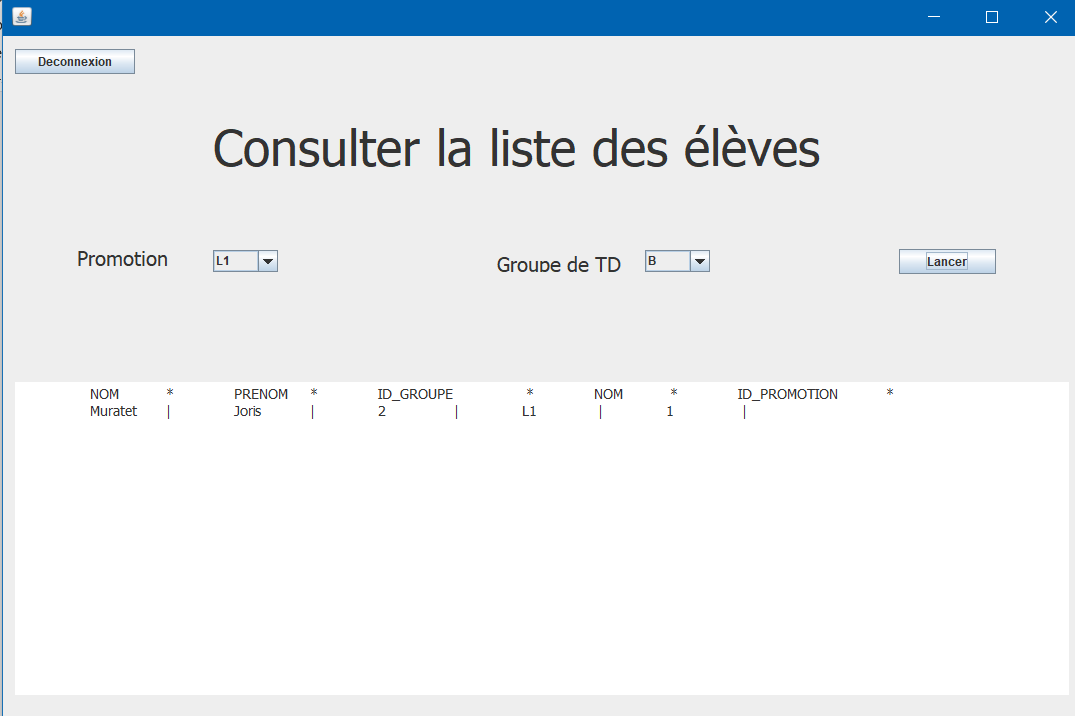
Concernant l’interface graphique on tient compte du choix de l’utilisateur pour appeler une des 3 méthodes lors du clic sur le bouton.



Voici quelques exemples de Test : (notre base n’est constituée que de 3 élèves).

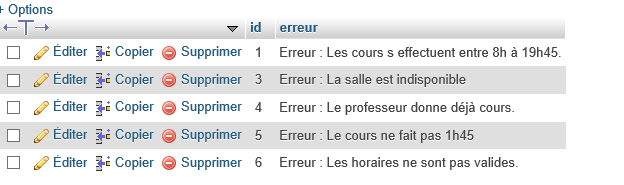




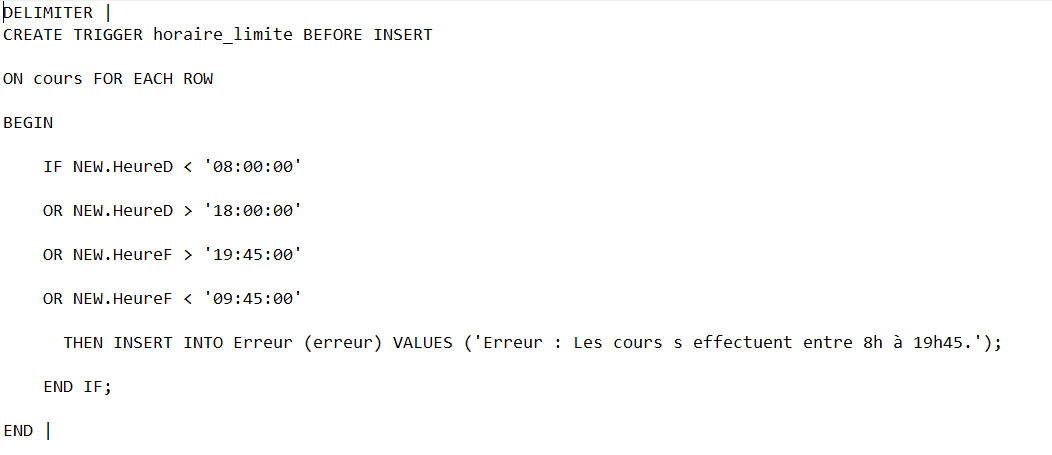


Contraintes et mise en place des Triggers

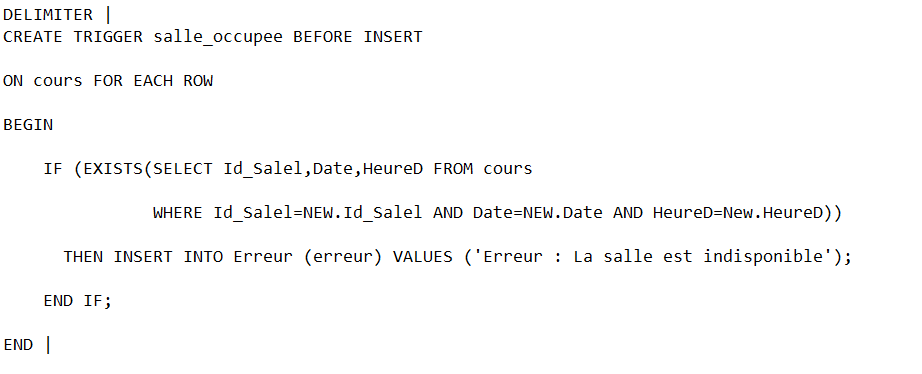
Pour pouvoir gérer notre planification d’emploi du temps, il est nécessaire d’avoir des contraintes. Nous avons donc utilisé des Triggers dans le code SQL. Afin de gérer les erreurs et empêcher la modification des données de la base nous avons créé une table erreur dans la base qui contient les erreurs de chacune des contraintes que nous créons, mais les tuples de cette table sont **uniques**. Lorsqu’on décide d’ajouter un tuple qui ne respecte pas une contrainte, notre code va essayer d’ajouter un élément correspondant à l’erreur associée, et comme l’erreur existe déjà dans la base il est impossible de la modifier.



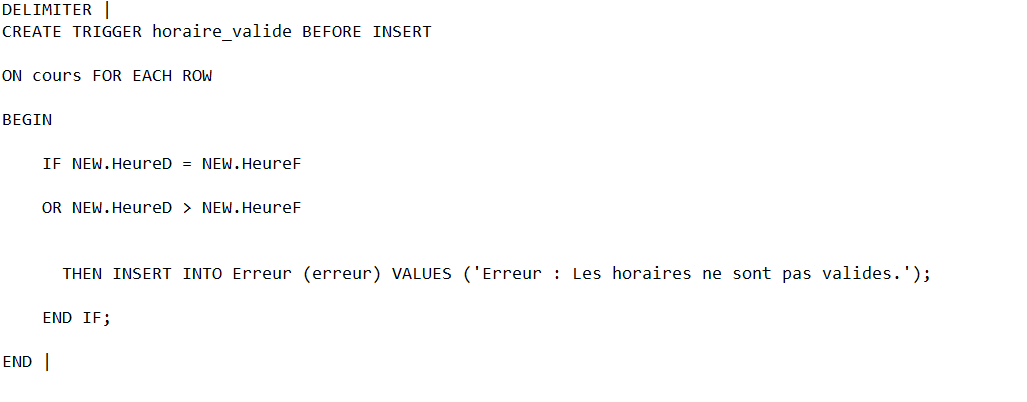
**Contrainte : « Les cours sont prévus du lundi au samedi de 8h00 à 20h00 »**



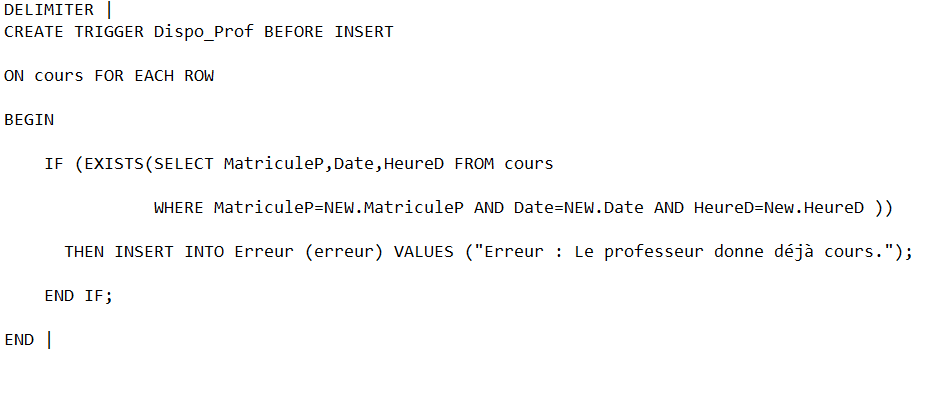
**Contrainte : « Pour ajouter un cours la salle doit être disponible »**



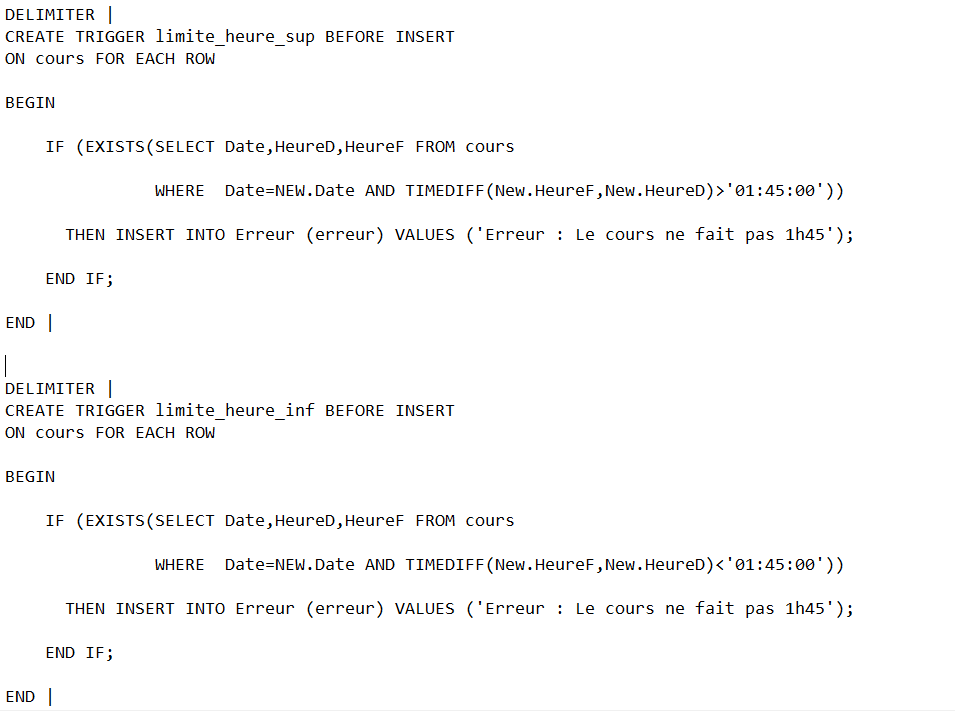
**Contrainte : « Pour pouvoir ajouter un cours, il faut que l’heure de début de cours soit inférieure à celle de fin »**



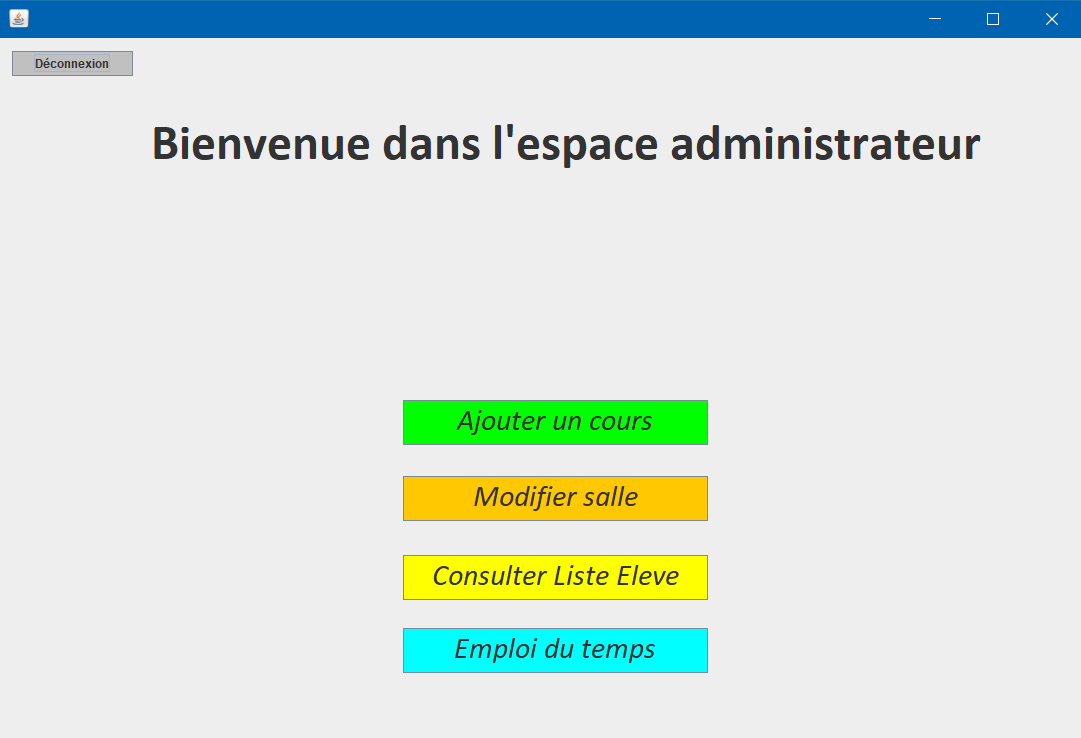
**Contrainte : « Pour pouvoir ajouter un cours, il faut que le professeur soit disponible »**



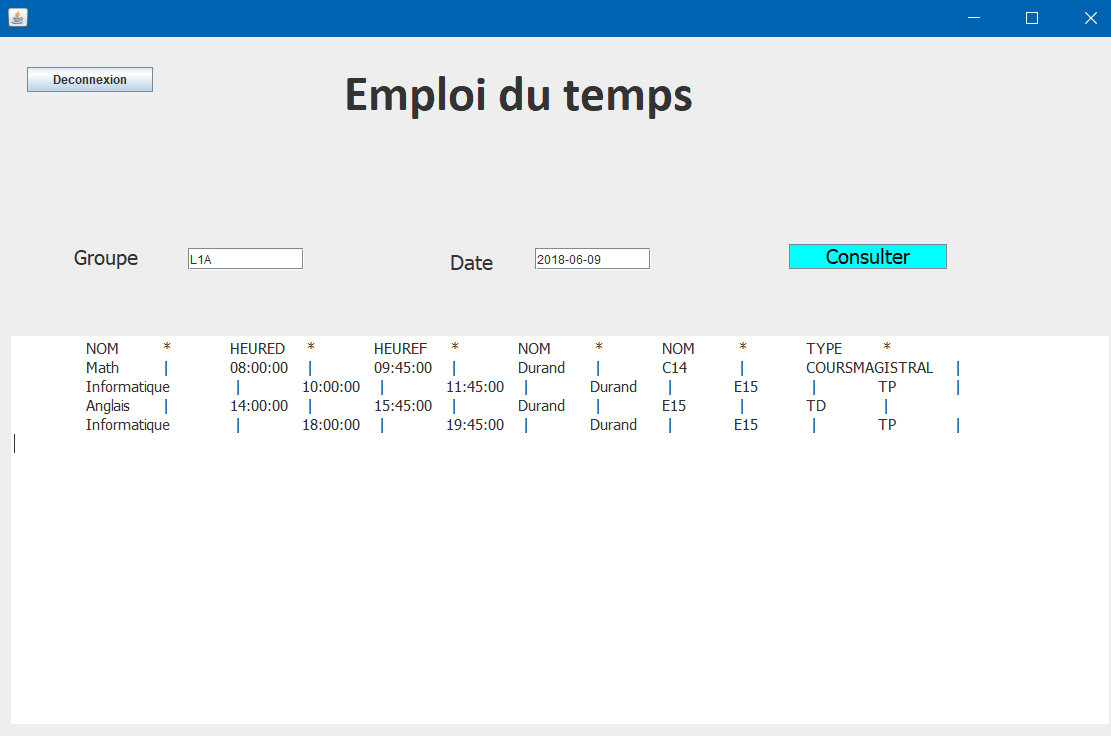
**Contrainte : « Pour pouvoir ajouter un cours, il faut que la durée soit impérativement de 1h45 »**



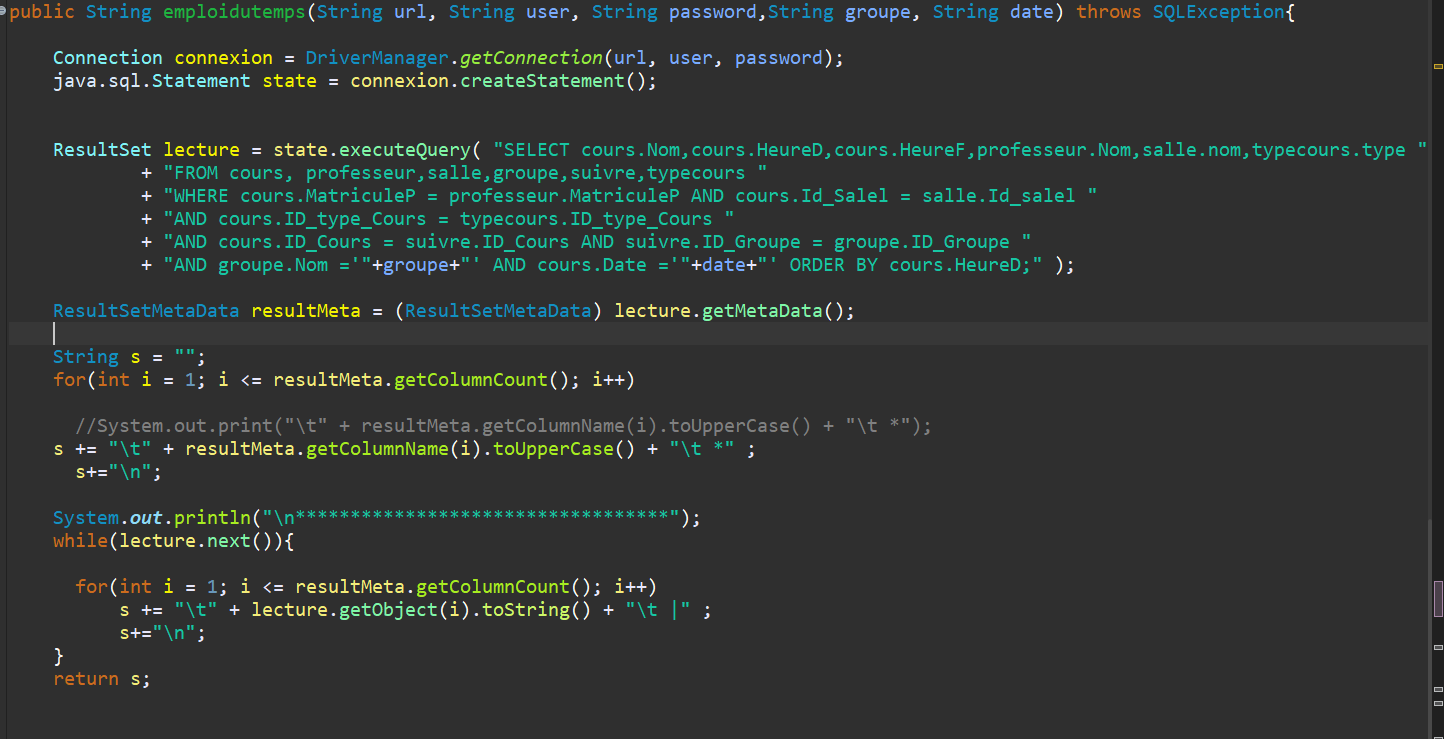
Nous n’avons pas fait la contrainte « Une salle ne peut pas recevoir plus d’élèves que sa capacité maximale » car notre base de données ne contient pas assez d’éléments pour pouvoir avoir des tests concluants.

Emploi du temps

Par manque de temps et difficultés nous n’avons pas réussi à faire l’affichage de l’emploi du temps sur une semaine entière et entre 2 dates. Par contre, nous avons fait l’affichage de l’emploi du temps pour une date donnée. Donc l’élève peut consulter les cours qu’il a sur une journée.

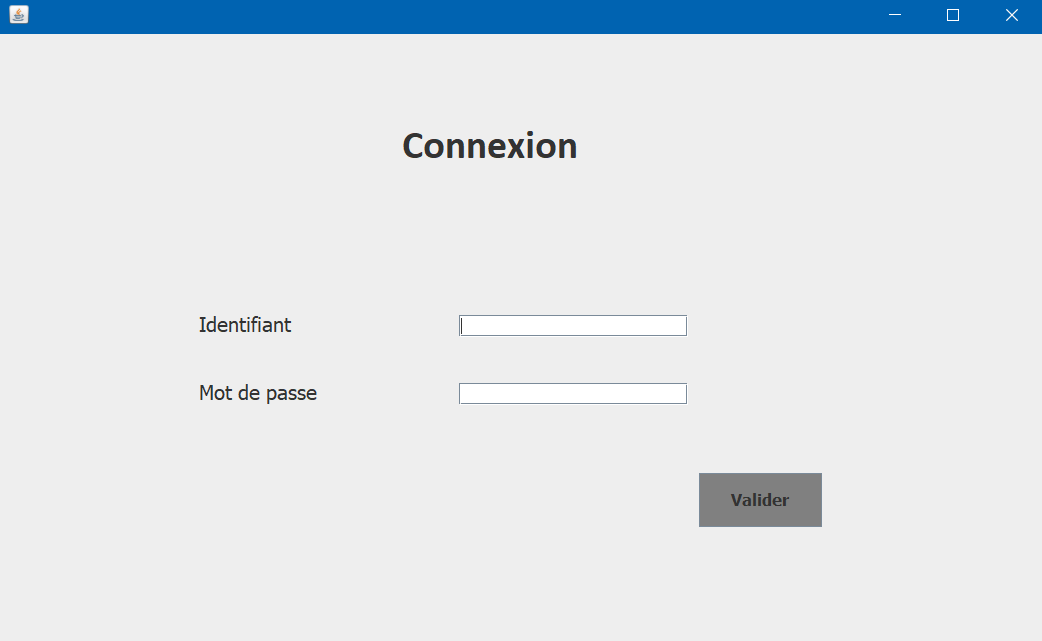


Cet affichage est réalisé grâce à la méthode emploidutemps de la classe cours.

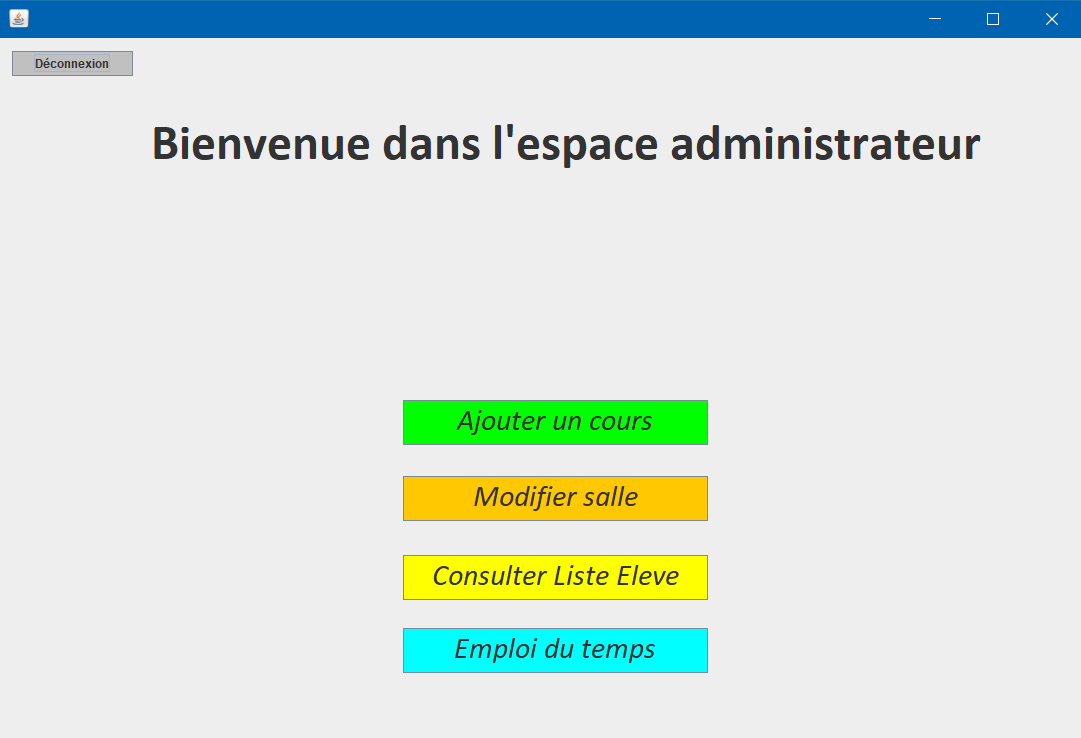


Connexion aux Profils

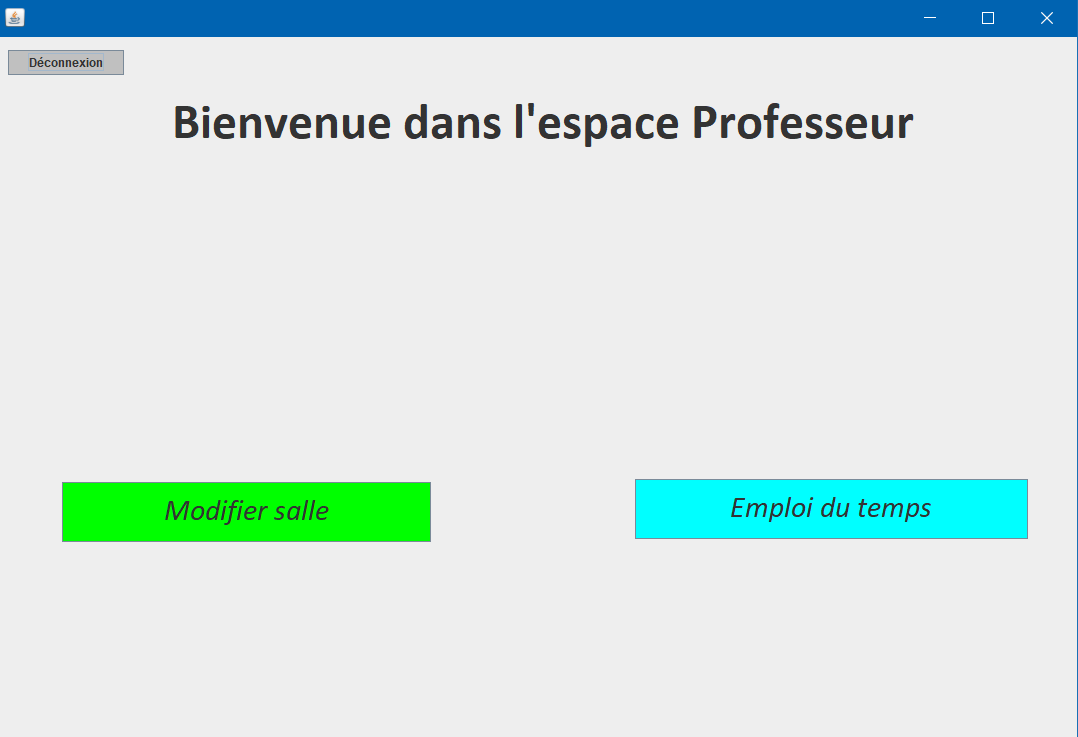
Notre application permet de choisir le profil : **admin, professeur ou élève**. Chacun de ces profils ont des fonctionnalités différentes :



Pour accéder au profil de l’administrateur il faut écrire : l’identifiant admin et le mot de passe admin

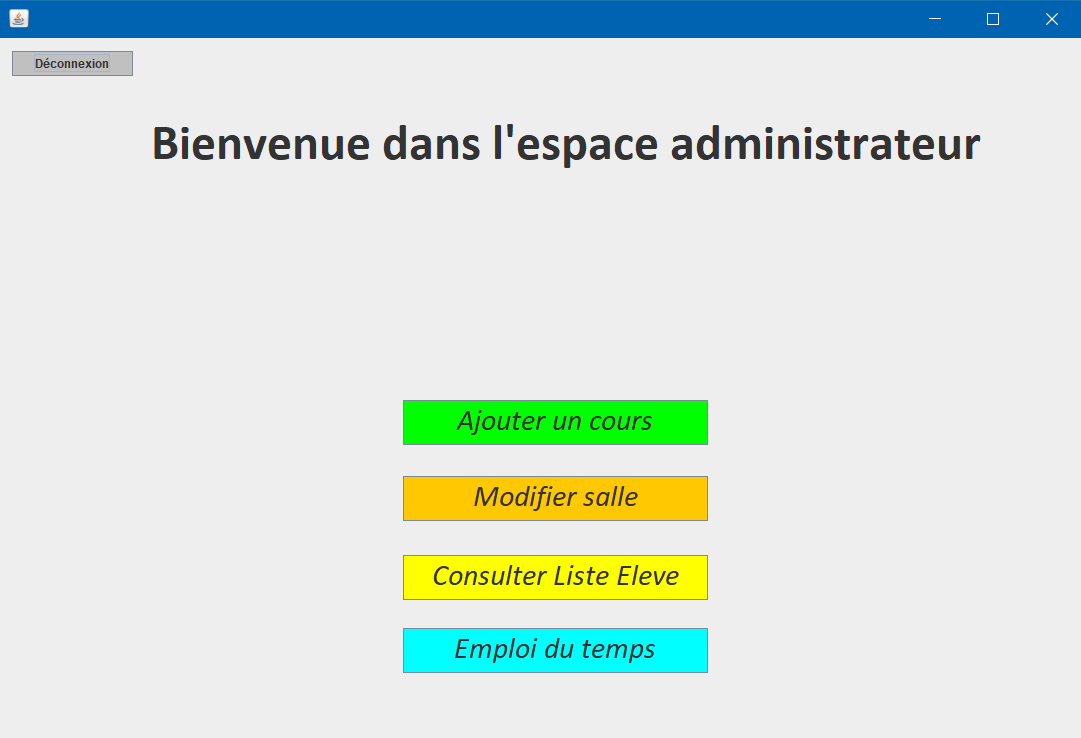


Pour accéder au profil du professeur, il faut écrire : l’identifiant prof et le mot de passe prof

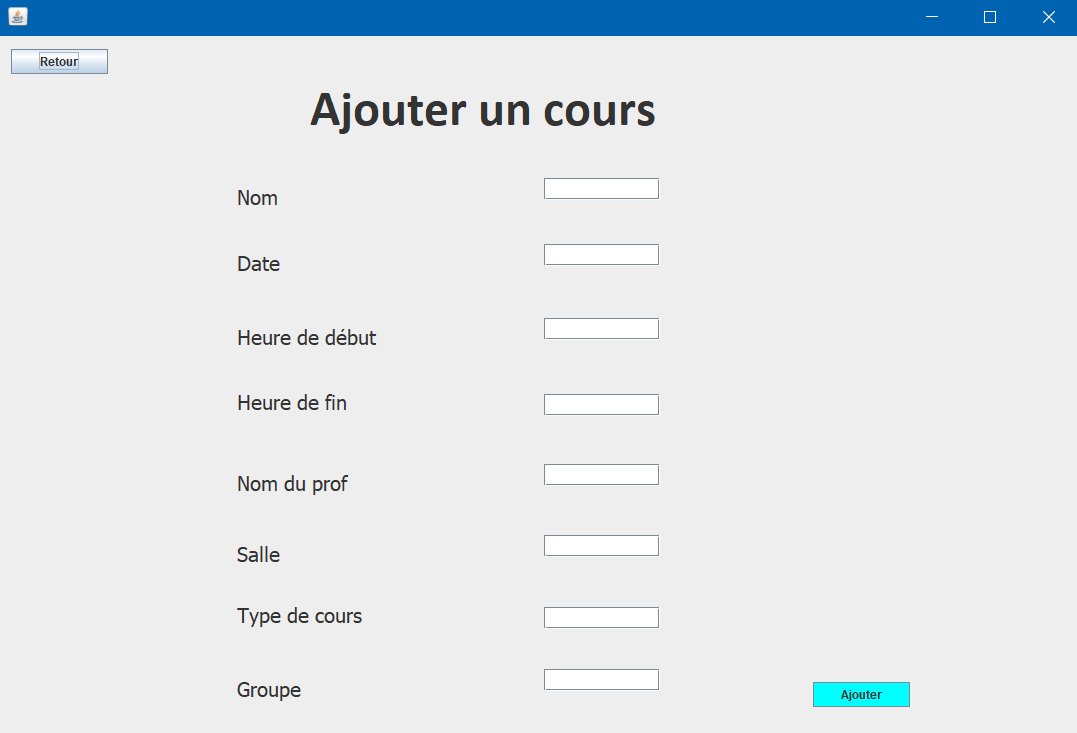


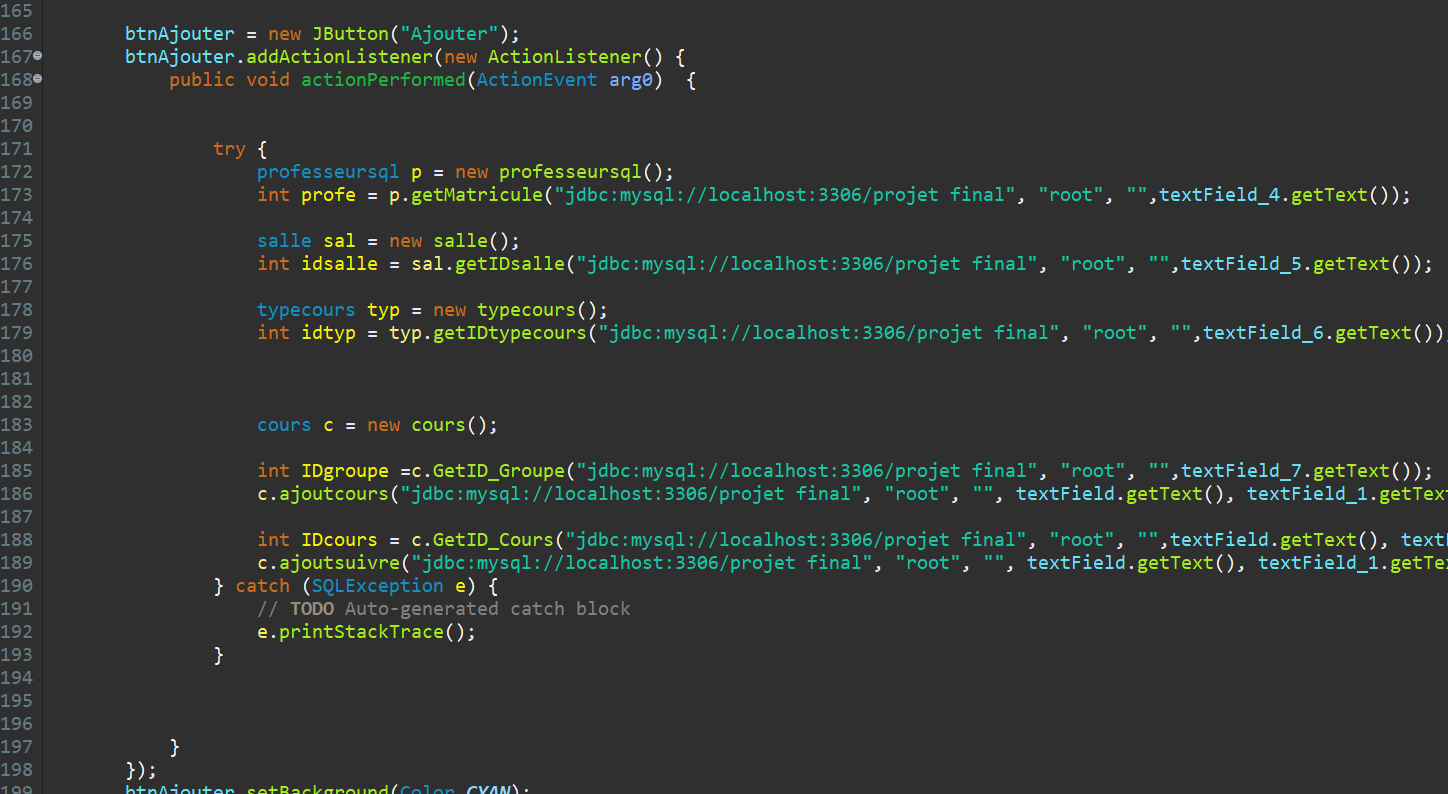
Pour accéder au profil de l’élève, il faut écrire : l’identifiant eleve et le mot de passe eleve



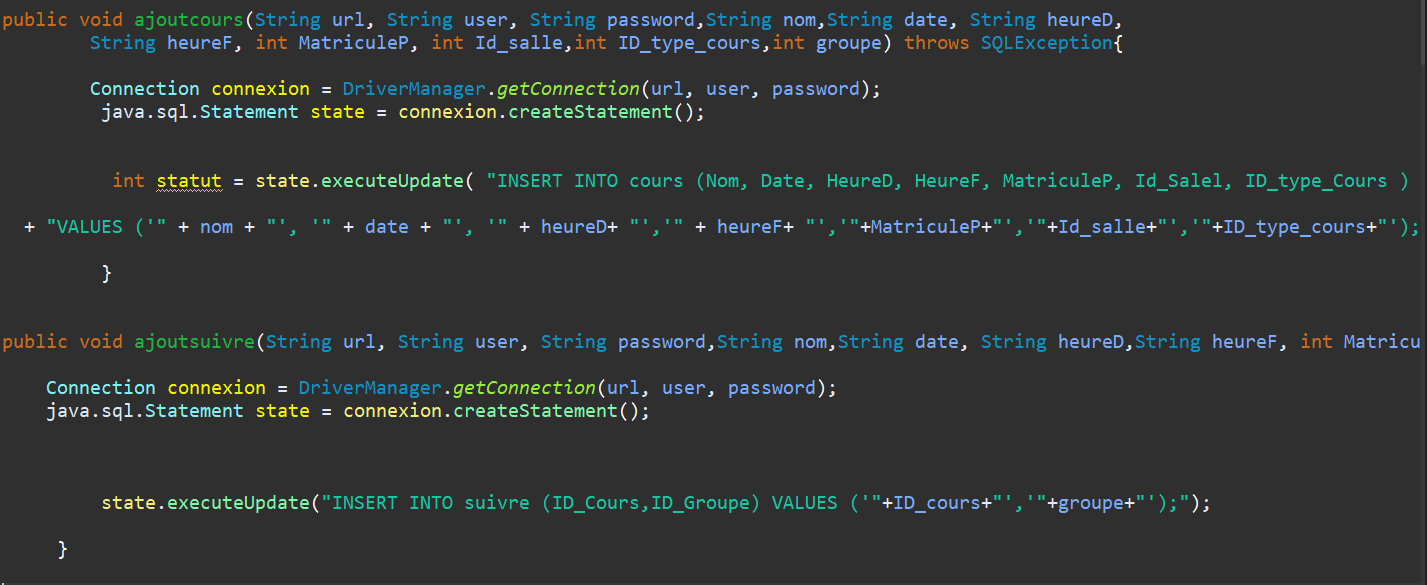
Ajouter un cours

Pour pouvoir ajouter un cours nous avons réalisé une interface graphique :



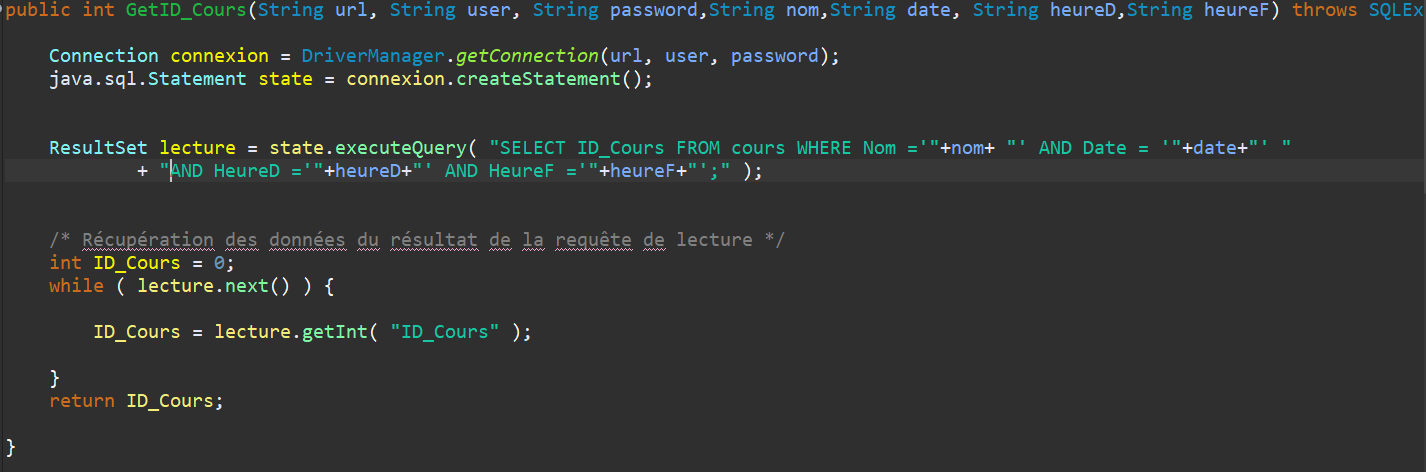


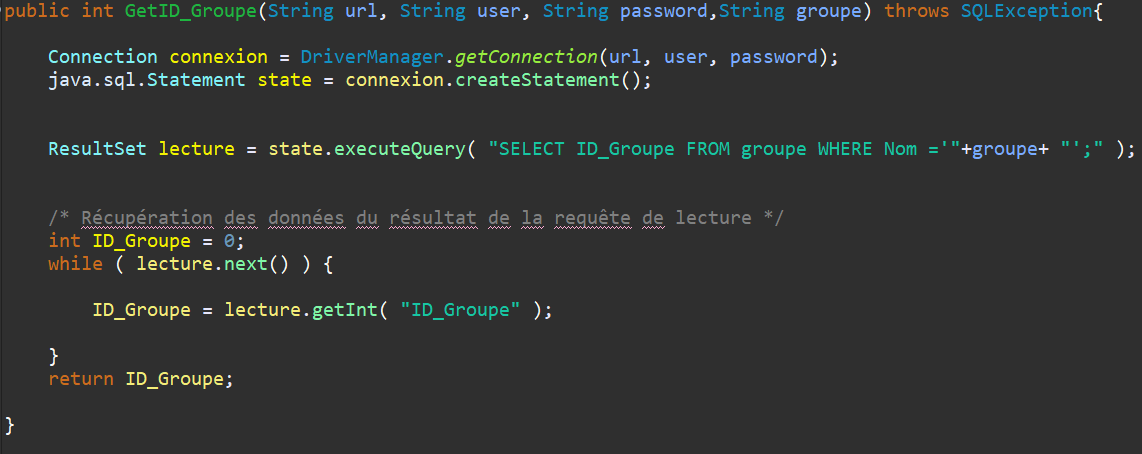
Lorsque l’on remplit tous les champs et que l’on clique sur ajouter, on fait appel aux méthodes ajoutcours et ajoutsuivre de la classe cours qui permet d’ajouter un cours dans la table cours et de remplir dans la table suivre le groupe et le cours suivit par ce groupe.



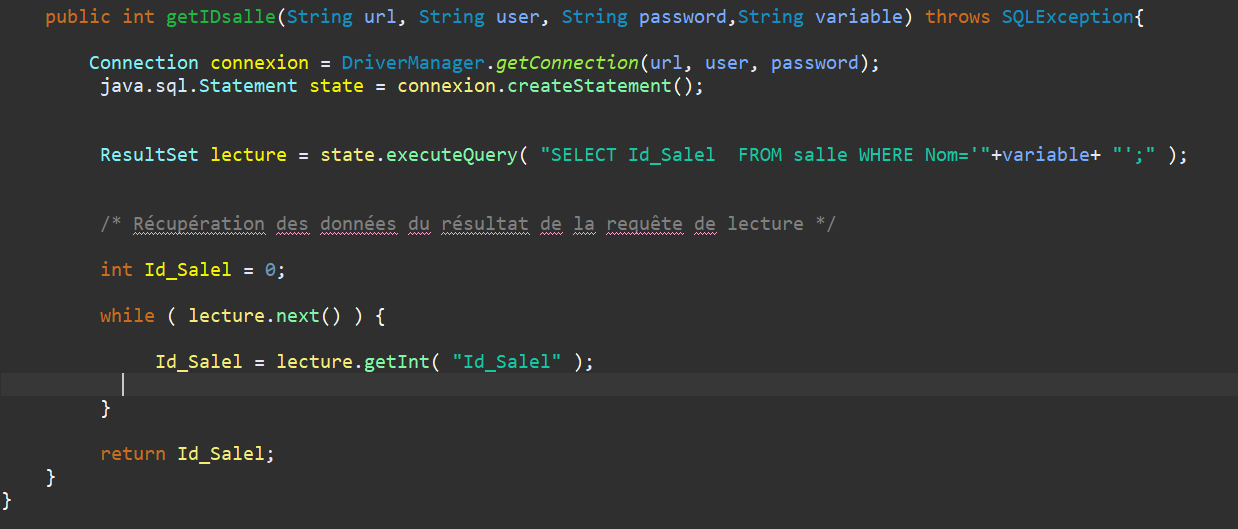
Nous avons également réalisé des méthodes afin de récupérer les différents ID de plusieurs tables pour s’en servir en argument de certaines méthodes comme ajoutcours et ajoutsuivre.

Dans la classe cours :

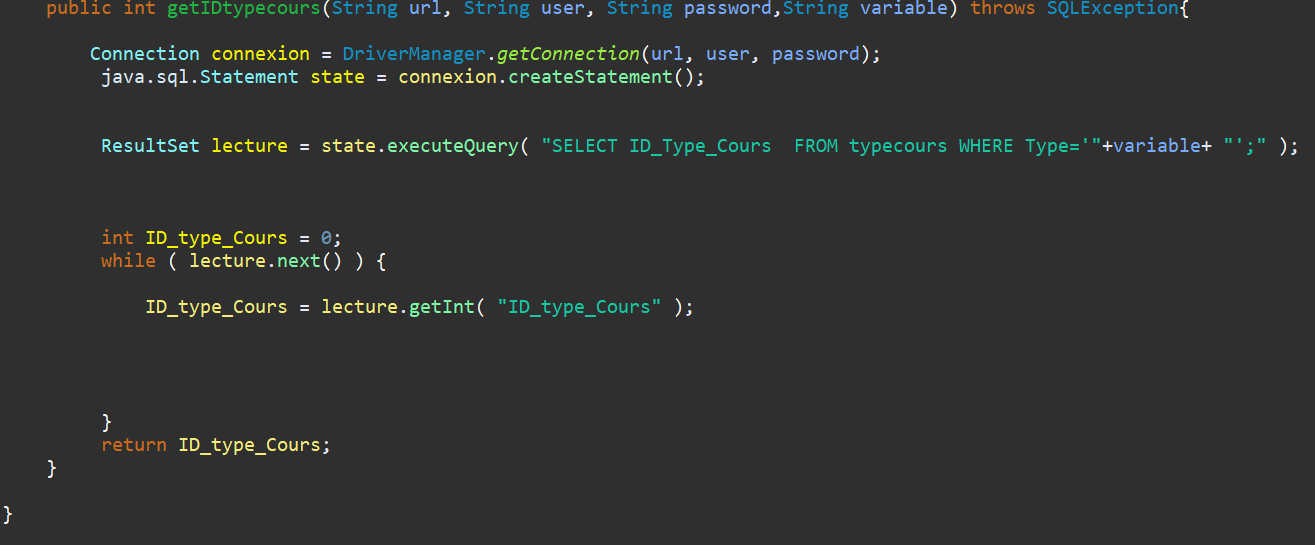


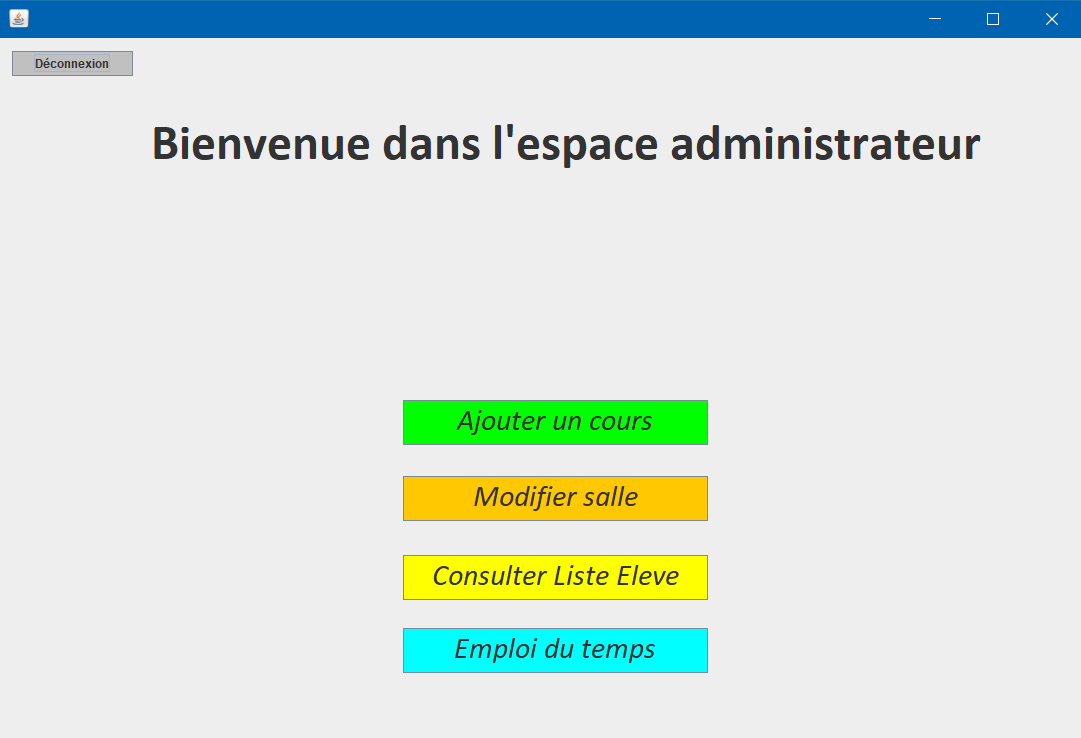


Dans la classe salle :

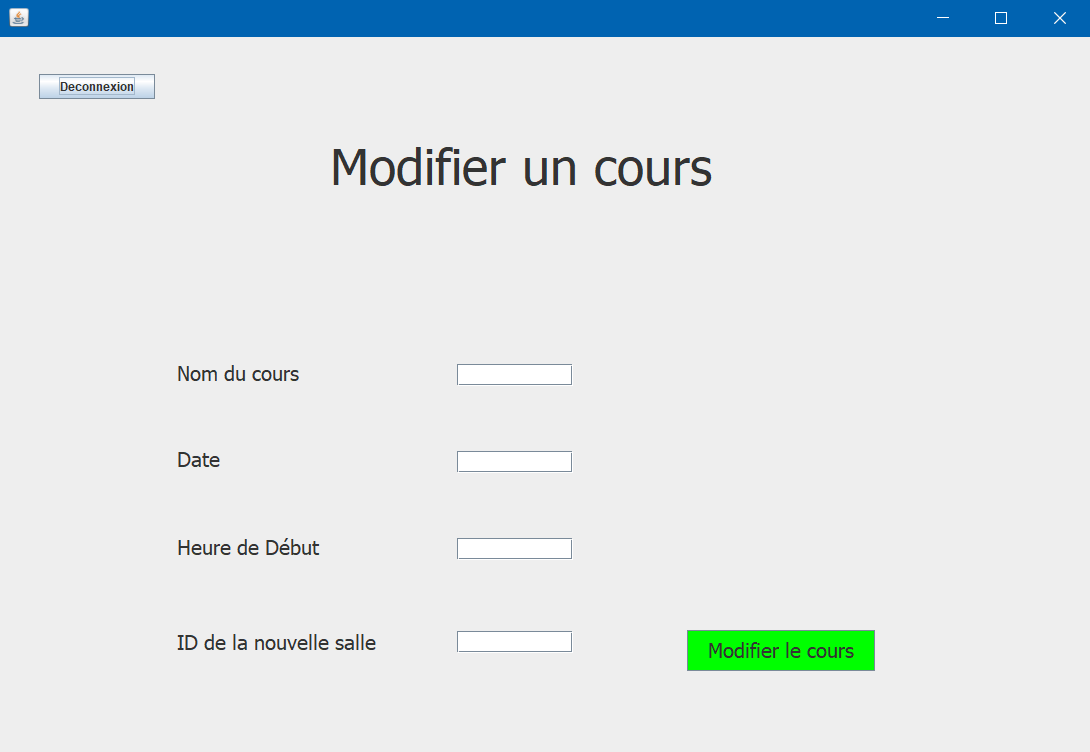


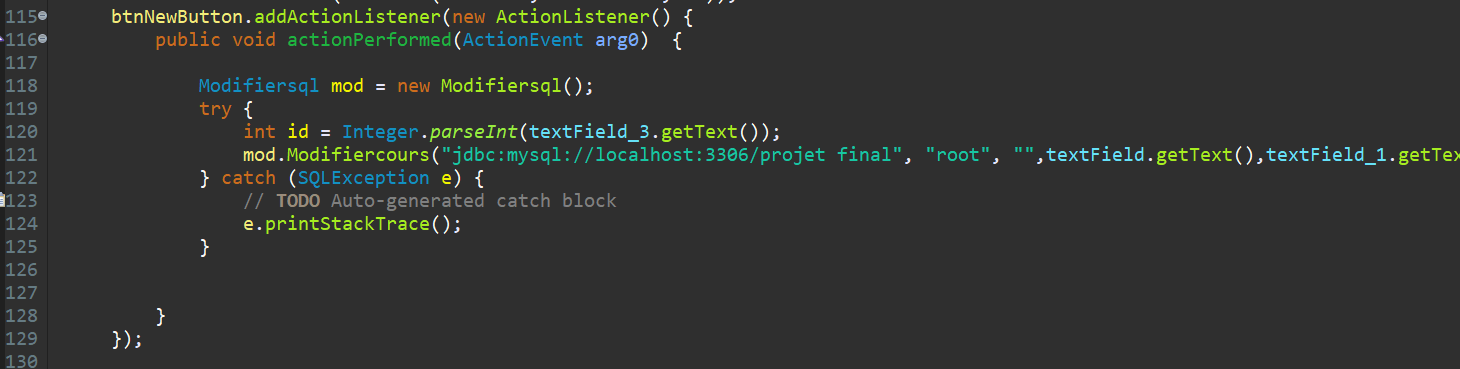
Dans la classe typecours :



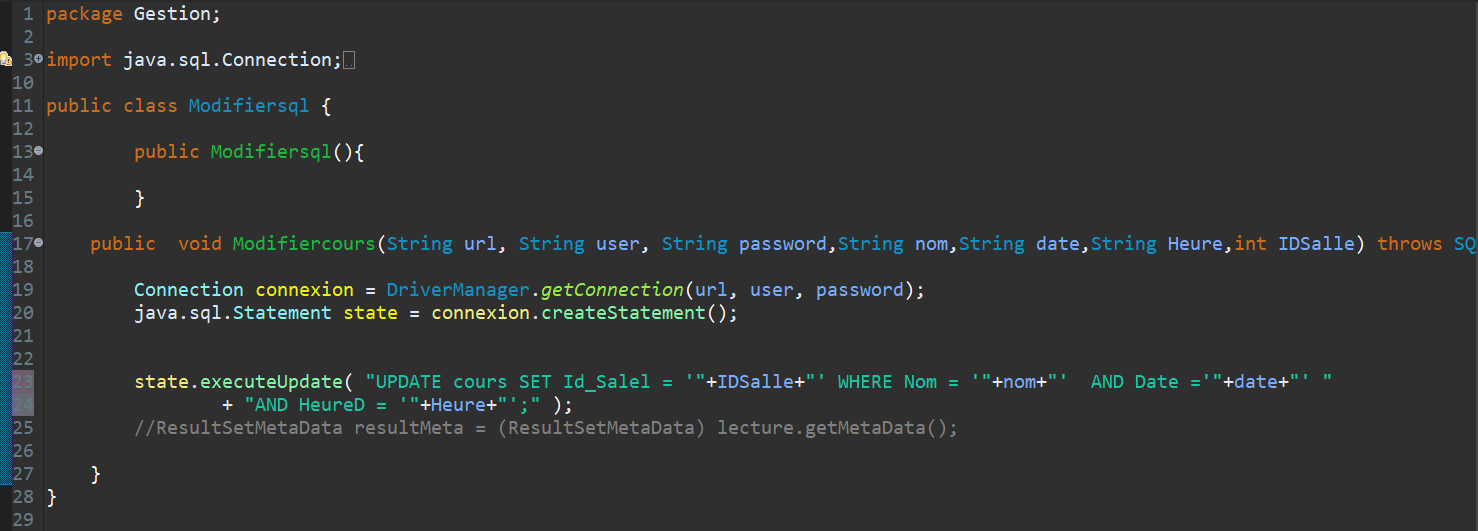
Modifier la salle

Une autre fonctionnalité est, pour un professeur, de pouvoir modifier la salle de son cours. Nous avons donc réalisé l’interface graphique suivante :



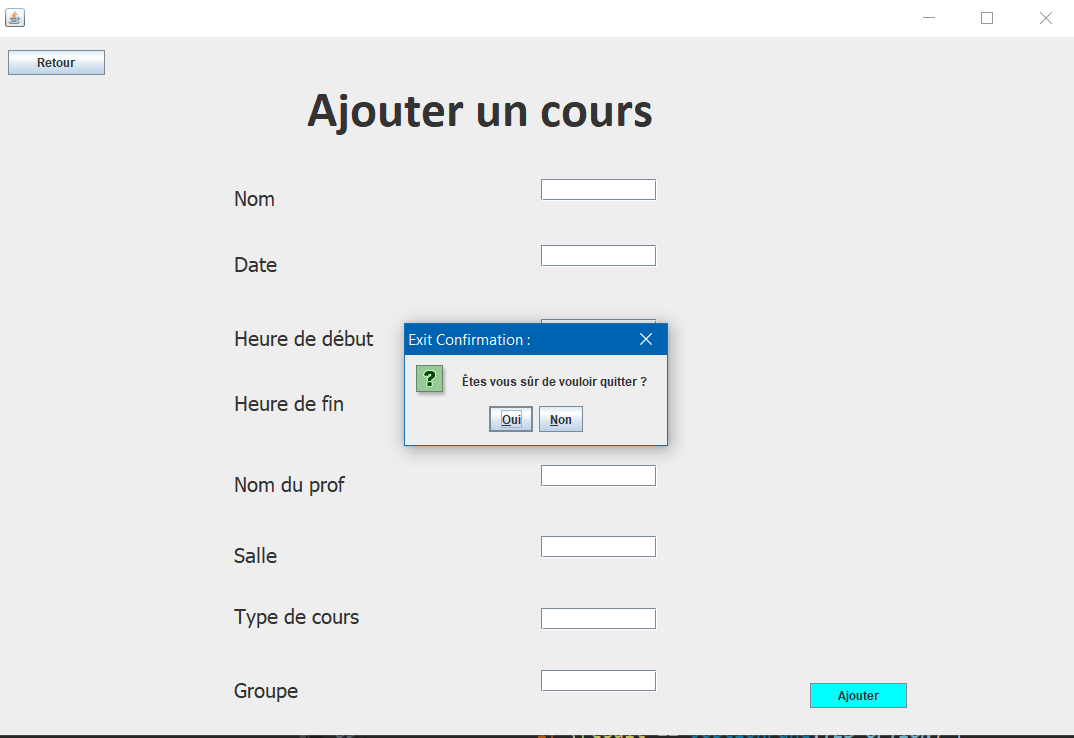


Nous avons codé la méthode modifier de la classe Modifiersql pour pouvoir changer la salle en exécutant une requête SQL.

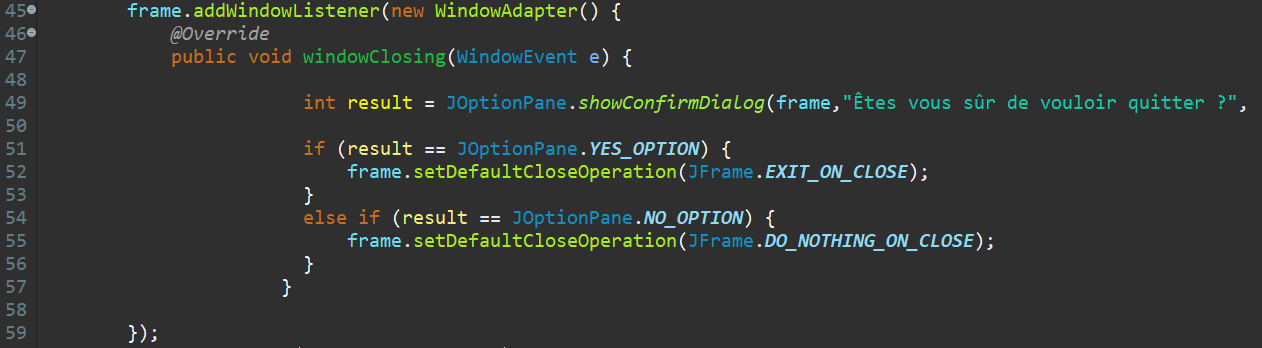


Confirmation avant de quitter

Il a été demandé qu’un message de confirmation apparaisse avant de pouvoir quitter l’application comme par exemple ci-dessous :

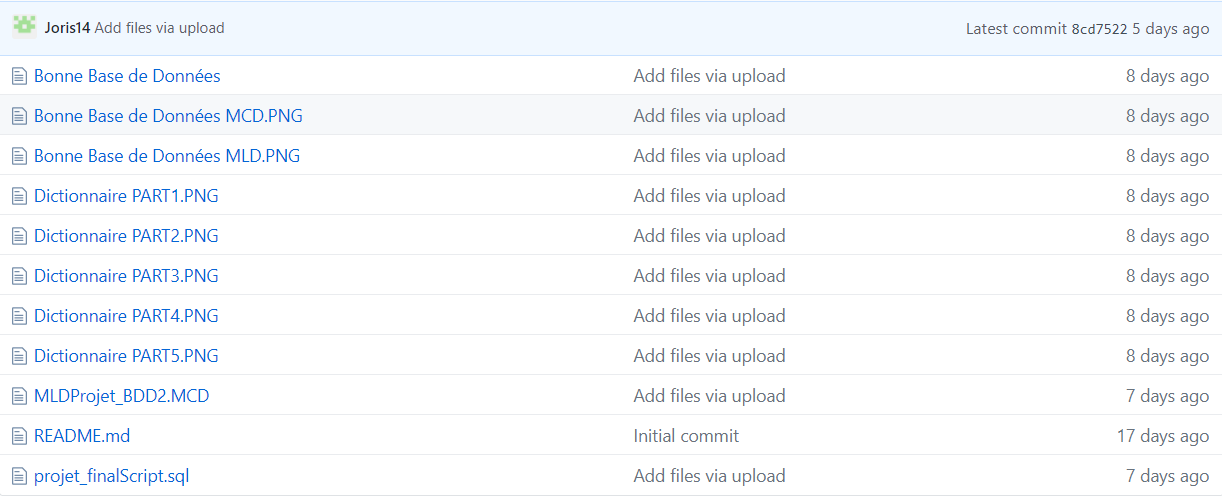


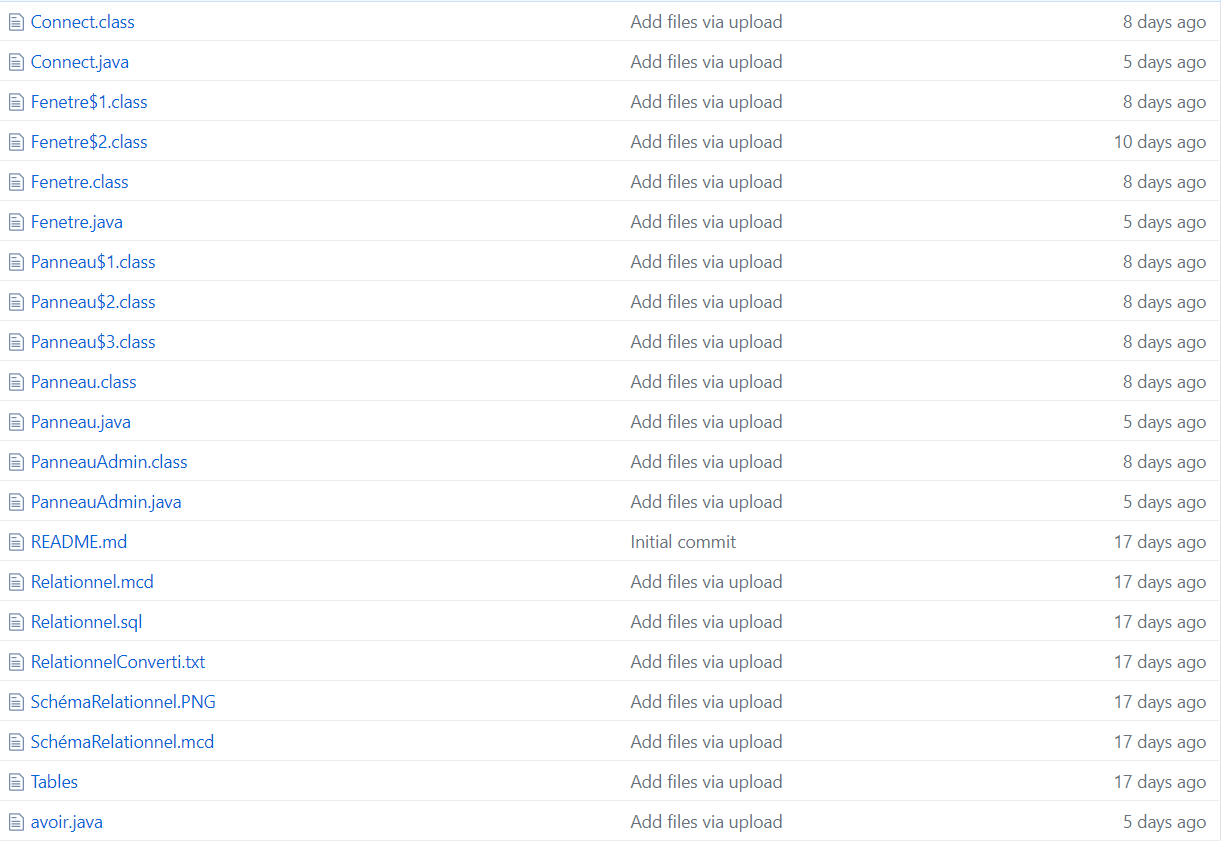
C’est pourquoi chacune de nos interfaces graphiques possèdent cette partie de code :



Outils de communication

Nous nous sommes aidés de github afin de partager les fichiers plus facilement durant notre projet.





Conclusion

Ce projet nous à permis de mettre en pratique et en application les notions de base de données apprise tout au long de ce module et ainsi nous confronter à un cas concret. Nous nous sommes heurtés à certaines difficultés liées au langage mais également au temps. De plus, nous aurions aimé approfondir l’application avec plus de fonctionnalités et une meilleure ergonomie.