



Projet J2EE

Application web de gestion de scolarité

Mohamed Haddache

02/12/24

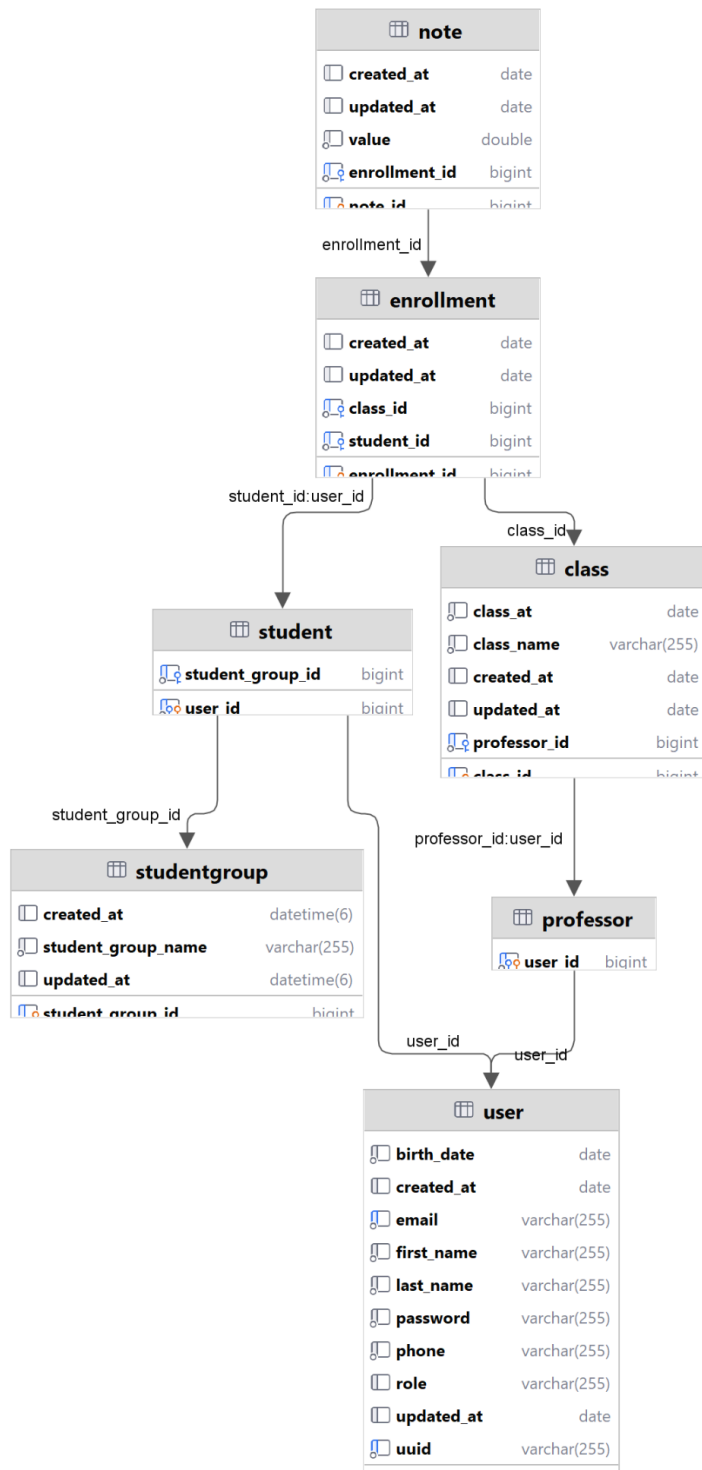
Jules Belletre, Théo Belliere, Pauline Maceiras, Clement Zhang ING2
GSI3 et Donovan Cardenas Temiquel ING2 GSI1

Sommaire

Conception	2
Schéma de conception de la base de données :	2
Architecture du projet	3
Bonnes pratiques utilisées	3
Spring Boot	5
Conception	5
Implémentation	6
Models	6
Repositories	6
Controllers	6
Views	7
Services	7
Récapitulatif des fonctionnalités	7
Annexe	9
Images du projet	9
Documentation	17

Conception

Schéma de conception de la base de données :



Explication du schéma :

Au coeur de l'application, **Cours**(*Class*) sont enseignés par des **Professeurs** (*Professor*), qui hérite des propriétés d'une même classe commune **Utilisateur** (*User*), regroupant des informations personnelles telles que le nom, l'email et le rôle (*PROFESSOR*, *STUDENT* ou *ADMIN*). Chaque **Cours** est associé à une liste d'**Inscriptions** (*Enrollments*), qui représente la participation des étudiants à un cours donné. Ces **Inscriptions** sont liées à des **Étudiants** (*Student* eux même héritent de la classe *User*) et permettent de suivre leurs **Notes**, reflétant leurs résultats académiques. De plus, les **Étudiants** sont regroupés dans des **Groupe d'étudiants** (*StudentGroup*), facilitant la gestion des classes.

Architecture du projet

Le projet utilise une architecture Model View Controller (MVC). Ici, le modèle correspond aux classes dans le package `models`. Ces classes ont chacune une table correspondante dans la base de données. Les contrôleurs sont dans le package `views`, divisées en trois packages principaux : un pour les étudiants, un pour les professeurs et un pour les administrateurs. Les vues sont toutes dans le dossier `webapp`. Les pages affichées sont dans le dossier `WEB-INF`. Il y a plusieurs sous-parties de ces vues qui sont similaires. Nous avons choisi de factoriser le code en créant des fichiers pour les parties communes. Ces fichiers sont dans le dossier `webapp/assets`. Il y a aussi à l'intérieur les feuilles de styles des pages `jsp` correspondantes.

Bonnes pratiques utilisées

Au niveau de la déclaration des entités hibernate l'application impose plusieurs contraintes pour garantir l'intégrité et la cohérence des données :

- **Relations strictes** : Les dépendances sont définies avec des relations `@OneToMany`, `@ManyToOne`, et `@JoinColumn`, avec des suppressions en cascade (`@OnDelete, CascadeType.ALL`) pour éviter les données orphelines. Par exemple, la suppression d'un **Cours** entraîne celle des **Inscriptions** et **Notes** associées.
- **Contraintes d'unicité** : Les colonnes comme `email` et `uuid` dans **Utilisateur** sont uniques pour éviter les doublons, tandis que des relations obligatoires (`nullable = false`) garantissent la présence des données essentielles, comme le lien entre un étudiant et son groupe.
- **Traçabilité** : Les champs `created_at` et `updated_at` dans chaque entité assurent le suivi des modifications.
- **Héritage normalisé** : La stratégie `@Inheritance(JOINED)` pour la classe **Utilisateur** unifie la gestion des rôles (professeurs, étudiants) tout en normalisant la base de données.

En dehors de la conception de la base de données nous avons aussi suivi de bonnes pratiques lors du développement de l'application :

Utilisation de Design Patterns :

- **Pattern Command** : Ce pattern a été utilisé pour encapsuler les actions exécutées par Hibernate, permettant une séparation claire entre les requêtes clients et leur exécution. Le **Client** (ex. un contrôleur ou servlet) crée des commandes spécifiques (CRUD sur les entités), transmises à un **Invoker**, qui délègue leur exécution au **Receiver** (implémenté par les services Hibernate).
- **Pattern Facade** : Une couche d'abstraction a été ajoutée pour simplifier l'accès aux opérations complexes sur les entités Hibernate, centralisant les interactions avec la base de données. Cela réduit le couplage entre les composants.

- **Pattern Singleton** : Ce pattern a été appliqué à la configuration Hibernate pour garantir qu'une seule instance de la session factory soit créée et partagée dans toute l'application. Cela optimise les ressources et évite les duplications coûteuses.

Intégration d'API tierces :

Nous avons intégré Gmail API afin de gérer l'envoi de notifications par email. Cela garantit une communication fiable avec les utilisateurs tout en s'appuyant sur une solution éprouvée et sécurisée pour les services de messagerie. Le compte utilisé est celui-ci cyschoolmanager@gmail.com.

Pour le bon fonctionnement de cette partie il faut penser à ajouter le fichier des clefs API : `clients_secret.json`

Contrôle des accès avec filtres :

Nous avons mis en place un filtre d'URL pour restreindre l'accès aux sections sensibles de l'application, telles que les parties administratives. Seuls les utilisateurs authentifiés en tant qu'**admin** peuvent accéder aux chemins (/admin/*).

Utilisation de variables d'environnement :

Pour rendre notre application aussi sécurisée que possible tout en la rendant adaptable à différents environnements, nous avons utilisé des variables d'environnement dans la configuration du serveur Tomcat.

APP_MAIL	cyschoolmanager@gmail.com
DB_PASSWORD	cytech0001
DB_URL	jdbc:mysql://localhost:3306/cy_school_manager
DB_USERNAME	root

Spring Boot

On a fait en parallèle une application avec les mêmes fonctionnalités utilisant le Framework Spring Boot.

Les principaux avantages de cette technologie sont :

Spring Boot JPA	Intégration et utilisation seamless d'hibernate ORM.
Annotations	Simplification dans le développement et dans la compréhension du code. Exemple : @Autowired
Repositories	Architecture héritant des fonctions de CRUD qui simplifie et économise du temps de développement.
Controllers	Améliorations dans les méthodes nécessaires pour afficher une page avec les bons paramètres. Améliore le développement et la repartition de la logique dans l'application.

Le seul désavantage trouvé est l'utilisation de fichiers JSP avec Spring Boot.

La technologie JSP/JSTL étant aujourd'hui obsolète n'est plus prise en compte par ce framework. Il est donc déconseillé de fusionner l'utilisation de ces deux technologies.

Néanmoins, pour respecter la deadline du projet, nous avons tout de même décidé d'intégrer les pages JSP de l'application sans Spring Boot dans cette deuxième itération du projet demandé.

Pour ce faire, nous avons dû créer une run configuration compilant et créant un fichier WAR avec les dépendances requises.

Ce WAR est ensuite exécuté par notre environnement.

Cette configuration garantit l'accès à nos vues JSP par l'application Spring Boot.

Conception

La conception de la base de données reste la même. Cependant l'interaction avec celle-ci est différente.

Grâce au fichier `application.properties` on peut centraliser l'ensemble des variables nécessaires pour la connexion à MySQL ainsi que celle pour lancer l'application.

On a la possibilité de faire des `@Queries` en dialecte MySQL dans les repositories des objets dans notre schéma. On peut aussi bien utiliser la classe `CrudRepository` pour déléguer pour garder une codebase DRY et correctement organisée.

Implémentation

`application.properties`

```
spring.application.name=CY School Manager Spring
spring.datasource.url=jdbc:mysql://localhost:3306/cy_school_manage
r?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update
spring.sql.init.mode=always
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

Models

On réutilise les classes annotées `@Entity` créées dans la première itération du projet. Ces classes nous servent de DAO afin de créer les tables nécessaires lors de l'initialisation de l'application.

Repositories

Ces fichiers nous permettent de définir les méthodes d'accès à la database.

Exemple :

```
@Repository
public interface ProfessorRepository extends
CrudRepository<Professor, Integer> {
    Professor getProfessorByUserId(Long userId);
}
```

Controllers

Ces contrôleurs nous permettent de définir les routes de notre application.

Exemple :

```
@Controller
@RequestMapping(path="/student")
public class StudentController {
```

```

@GetMapping(path="/index")
public String Index(Model model, HttpServletRequest request) {
    request.getSession().getAttribute("user");
    return "/student/index";
}
}

```

Views

On retrouve ici tous les fichiers JSP déjà utilisés dans l'application sans Spring Boot.

Services

On a configuré ici les services externes pratiques pour des features comme l'export des notes ou l'envoi d'emails.

```

@Service
public class PdfGenerator {

    private static final String REPORTS_DIRECTORY =
"C:/tomcat/reports";

    public static String generateReportForStudent(Student student,
List<Enrollment> enrollments) {}
}

```

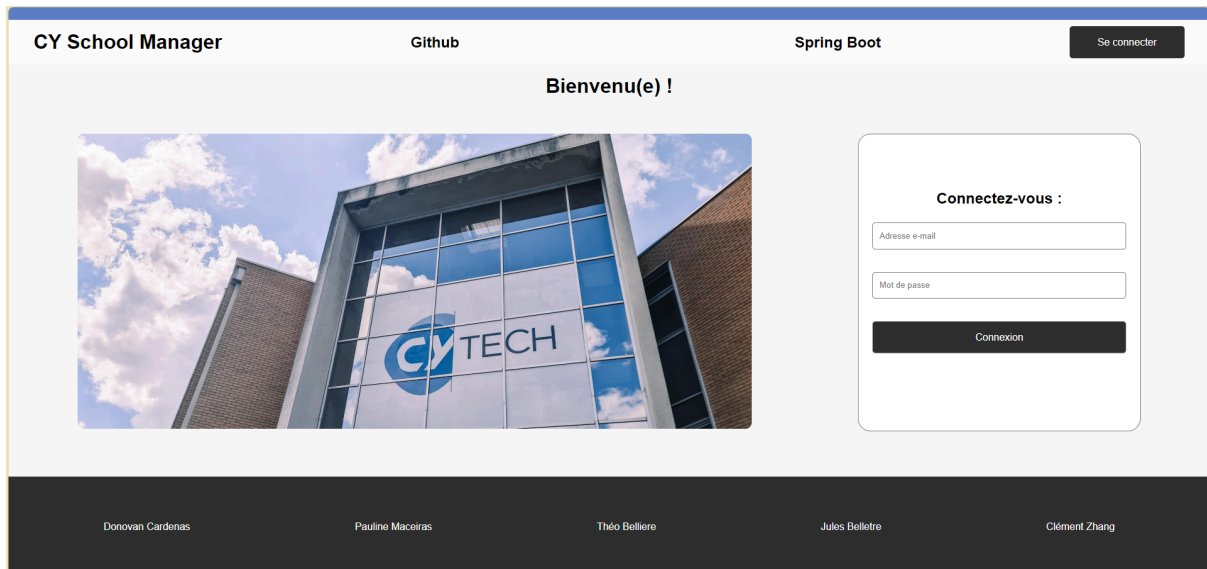
Récapitulatif des fonctionnalités

Intitulé	Description	Fichiers correspondants
Authentification	Identifiant et mot de passe pour se connecter	package views→ Login.java
Autorisation	Filtrage des pages /admin/* à l'aide d'un @WebFilter	
Mailing	Lors des modifications des inscriptions d'un user par l'admin et lors des modifications de ses infos personnelles. On utilise Gmail API.	package mailing → Gmailer.java (annexe 15, 16, 17)
Génération de pdf	Remplissage de la template avec les bonnes données en utilisant Thymeleaf puis génération du pdf avec IText	package pdfgenerator→ PdfGenerator.java resources/templates → rapport_notes.html

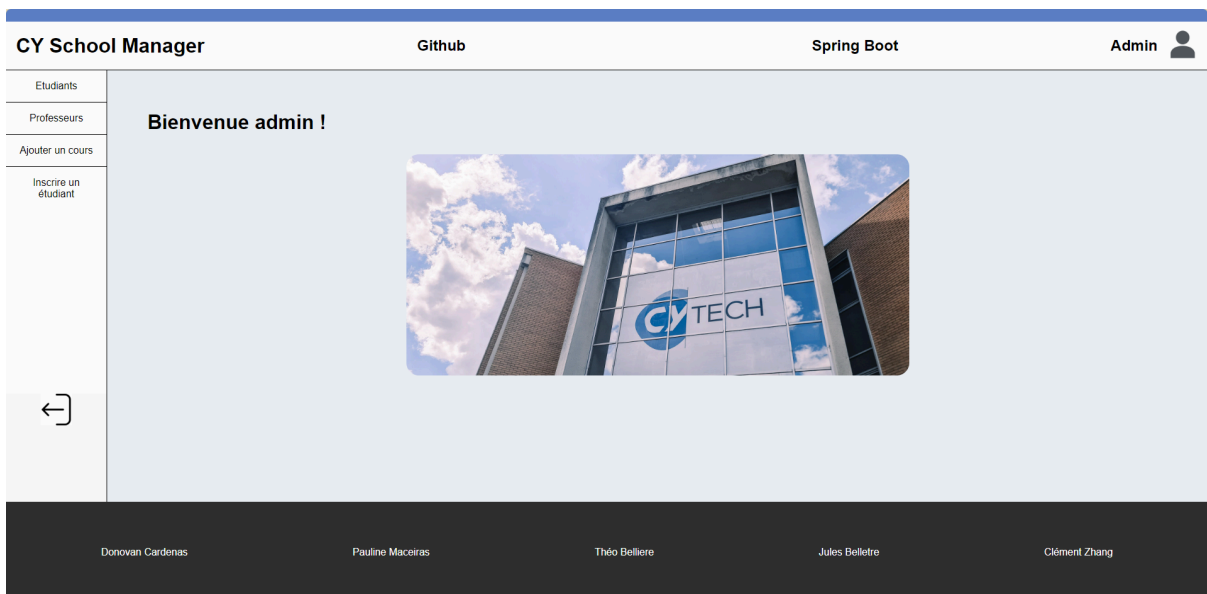
CRUD d'utilisateurs	Les admins peuvent ajouter, modifier et supprimer des utilisateurs (élèves, professeurs)	package views.admin→ AddUser.java, UpdateUser.java, RemoveUser.java
Attribution de cours	Les admins peuvent affecter des cours à des élèves et des professeurs	package views.admin→ AddUser.java, UpdateUser.java
Ajout de cours	Les admins peuvent ajouter des cours en sélectionnant un professeur	package views.admin→ AdminAddClass.java
Remise des notes	Les professeurs peuvent ajouter des notes aux élèves en choisissant la matière correspondante	package views.professor→ ProfessorAddNote.java
Affichage des matières enseignées/suivies	Les professeurs et élèves peuvent voir leurs cours attribués	package views→ professor.Professor Schedule.java student.StudentSchedule.java

Annexe

Images du projet



Annexe 1 : page de login



Annexe 2 : page d'accueil admin

CY School Manager

Github

Spring Boot

Admin

Etudiants

Professeurs

Ajouter un cours

Inscrire un étudiant

Etudiants

Ajouter un étudiant

Email	Nom	Prenom	Classe	Action			
jane.doe@example.com	Doe	Jane	GSI-1	Afficher	Modifier	Supprimer	
john.smith@example.com	Smith	John	GSI-1	Afficher	Modifier	Supprimer	
alice.brown@example.com	Brown	Alice	GSI-2	Afficher	Modifier	Supprimer	
michael.jones@example.com	Jones	Michael	GSI-2	Afficher	Modifier	Supprimer	
emily.wilson@example.com	Wilson	Emily	GSI-3	Afficher	Modifier	Supprimer	
david.johnson@example.com	Johnson	David	GSI-3	Afficher	Modifier	Supprimer	
pauline.maceiras@gmail.com	Maceiras	Pauline	ING-2	Afficher	Modifier	Supprimer	

←

Donovan Cardenas

Pauline Maceiras

Théo Belliere

Jules Belletr

Clément Zhang


Annexe 3 : page liste étudiants

CY School Manager

Github

Spring Boot

Admin



Etudiants

Professeurs

Ajouter un cours

Inscrire un étudiant

Professeurs

Ajouter un professeur

Email	Nom	Prenom	Cours	Action		
cdumont@school.com	Dumont	Clara	Introduction to Mathematics , Cours nul	Afficher	Modifier	Supprimer
mlefebvre@school.com	Lefebvre	Marc	Advanced Physics	Afficher	Modifier	Supprimer
smartins@school.com	Martins	Sofia	Biology for Beginners	Afficher	Modifier	Supprimer
pnguyen@school.com	Nguyen	Paul	Computer Science Basics	Afficher	Modifier	Supprimer
lcaron@school.com	Caron	Lucie	French Literature	Afficher	Modifier	Supprimer

←

Donovan Cardenas

Pauline Macoiras

Théo Belliere

Jules Belletr

Clément Zhang

Annexe 4 : page liste professeurs

CY School Manager
Github
Spring Boot
Admin

Etudiants
Professeurs
Ajouter un cours
Inscrire un étudiant

Ajouter une Classe

Nom de la classe :

Rechercher un professeur

Nom : Prénom : Email :

Résultats :

- ☐ Clara Dumont - cdumont@school.com
- ☐ Marc Lefebvre - mlefevre@school.com
- ☐ Sofia Martins - smartins@school.com
- ☐ Paul Nguyen - pnguyen@school.com
- ☐ Lucie Caron - lcaron@school.com

Donovan Cardenas
Pauline Maceiras
Théo Belliere
Jules Belleire
Clément Zhang

Annexe 5 : page d'ajout de cours

CY School Manager
Github
Spring Boot
Admin

Etudiants
Professeurs
Ajouter un cours
Inscrire un étudiant

Inscrire un élève

Recherche d'Étudiants

Nom : Prénom : Classe :

Résultats :

- ☐ Jane Doe - GSI-1
- ☐ John Smith - GSI-1
- ☐ Alice Brown - GSI-2
- ☐ Michael Jones - GSI-2
- ☐ Emily Wilson - GSI-3
- ☐ David Johnson - GSI-3
- ☐ Pauline Maceiras - ING-2

Sélectionnez un cours

Donovan Cardenas
Pauline Maceiras
Théo Belliere
Jules Belleire
Clément Zhang

Annexe 6 : page d'inscription d'un étudiant

CY School Manager
Github
Spring Boot
Admin

Etudiants
Professeurs
Ajouter un cours
Inscrire un étudiant

Ajouter un utilisateur

Nom
Prénom
Email
Numéro de téléphone
Date de naissance
Statut
Numéro de groupe

Ajouter des cours

Confirmer

Donovan Cardenas
Pauline Maceiras
Théo Belliere
Jules Belletre
Clément Zhang

Annexe 7 : page ajout d'un utilisateur

CY School Manager
Github
Spring Boot
Admin

Etudiants
Professeurs
Ajouter un cours
Inscrire un étudiant

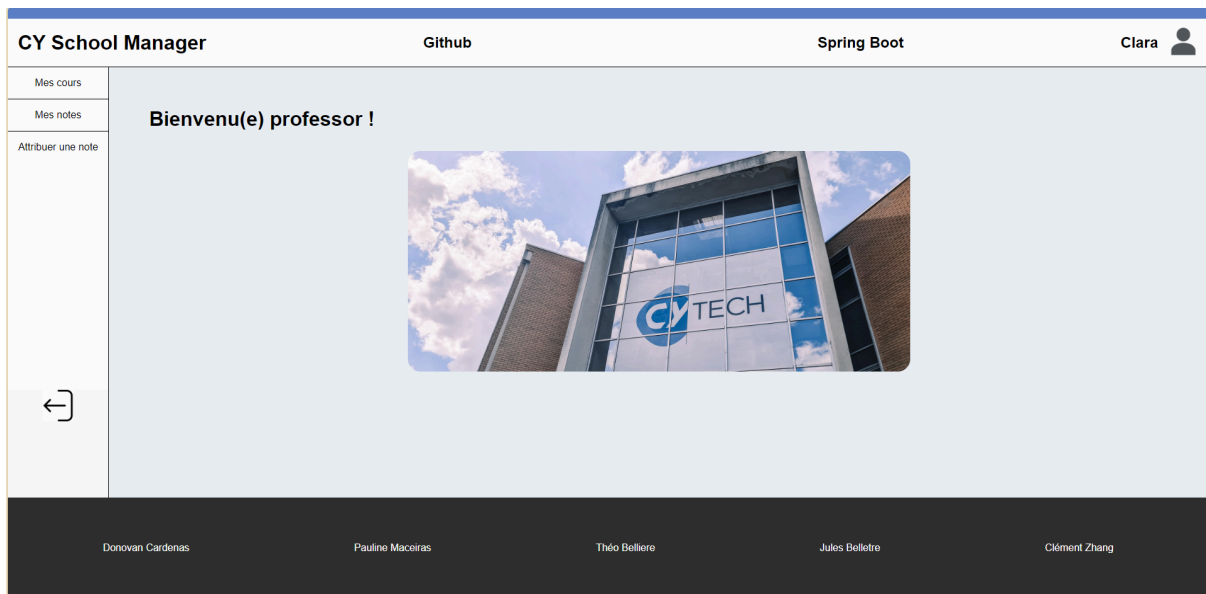
Mes coordonnées

Nom
Prénom
Email
Numéro de téléphone

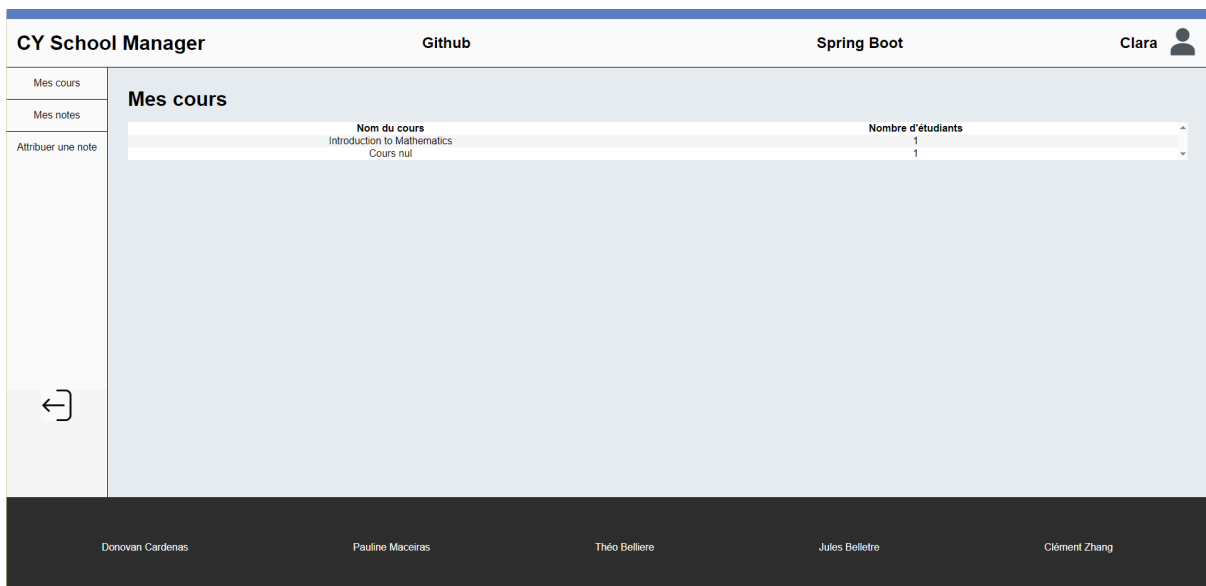
Mettre à jour

Donovan Cardenas
Pauline Maceiras
Théo Belliere
Jules Belletre
Clément Zhang

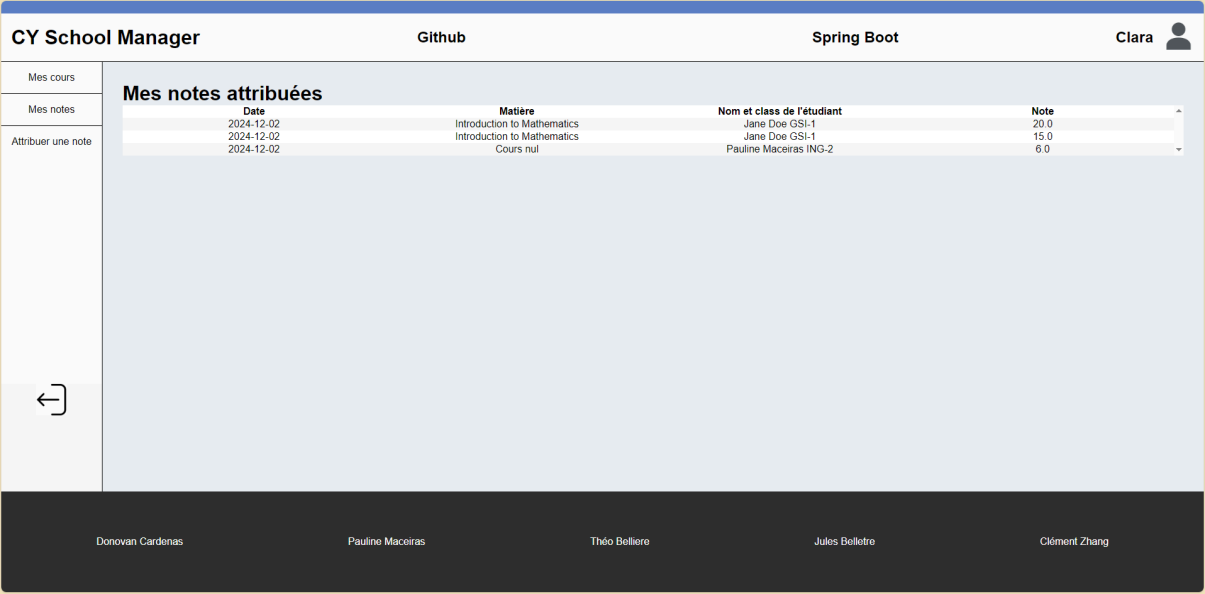
Annexe 8 : page profil



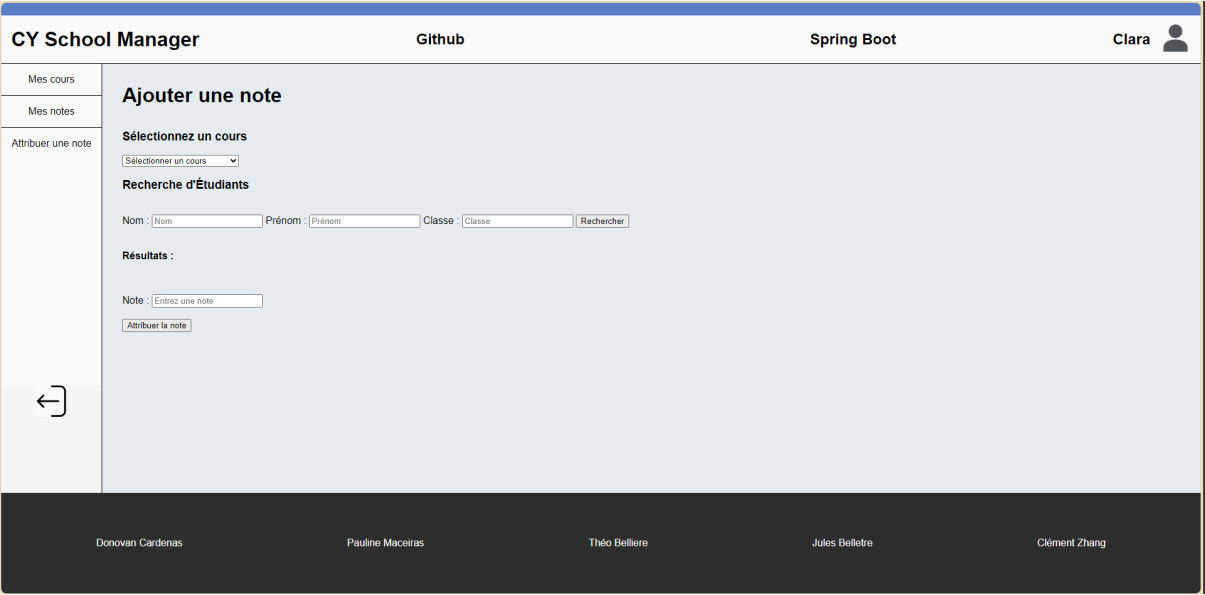
Annexe 9 : page d'accueil professeur




Annexe 10 : page mes cours professeur




Annexe 11 : page mes notes professeur



Annexe 12 : page ajout note professeur

CY School Manager							Jane 
Emploi de temps	Emploi du temps						
Mes notes	Heure	Lundi	Mardi	Mercredi	Jeudi	Vendredi	
	8:00	Introduction to Mathematics					
	9:00	Introduction to Mathematics					
	10:00		Advanced Physics				
	11:00		Advanced Physics				
	12:00						
	13:00						
	14:00						
	15:00						
	16:00						
	17:00						
	18:00						

Annexe 13 : Page planning étudiant

CY School Manager							Jane 
Emploi de temps	Mes notes						
Mes notes	Télécharger mes Notes PDF						
	Date	Matière	Libellé	Note			
	2024-12-02	Introduction to Mathematics	Partiel session 1	20.00			
	2024-12-02	Introduction to Mathematics	Partiel session 2	15.00			
	2024-12-02	Advanced Physics	Partiel session 1	15.00			

Annexe 14 : page notes étudiant

Validation de l'inscription Externes Boîte de réception x



cyschoolmanager@gmail.com

À moi ▼

Bonjour theo be,
Votre inscription scolaire a été validée pour le cours : French
Literature avec le ou la professeur Caron

← Répondre

→ Transférer

Annexe 15 : Mail de confirmation d'inscription à un cours

Modification de vos informations personnelles Externes Boîte de réception x



cyschoolmanager@gmail.com

À moi ▼

Bonjour theo be,
Vos informations personnelles ont bien été modifiées avec succès.

Voici un récapitulatif de vos informations :

Prénom : theo

Nom : be

Email : belliereth@cy-tech.fr

Date de naissance : 2005-05-05

Téléphone : 1234

Annexe 16 : Mail de confirmation de changement des données personnelles

Nouvelles notes disponibles Externes Boîte de réception x



cyschoolmanager@gmail.com

À moi ▼

Bonjour theo be,
De nouvelles notes ont été ajoutées ou modifiées à votre dossier

Annexe 17 : Mail d'information d'ajout de nouvelles notes

Documentation

[JPA Criteria Queries | Baeldung](#)

[Envoyer des e-mails | Gmail | Google for Developers](#)

[Guide de démarrage rapide pour Java | Gmail | Google for Developers](#)