Gaétan Retel
Julien Guyot
Mateo Lopez
Guillaume Androny
Ulrich Sautreuil
ING2 GSI2

30 novembre 2024

RAPPORT DE PROJET

JEE 2024-2025

HADDACHE MOHAMED

Sommaire

I Conception du projet

1 Méthodologie

Nous avons utilisé github avec un nommage propre des commits, selon conventionalcommits.org, et l'utilisation de branches et de pull request. Les pulls request permettent également de faire de la revue de code, et de faire valider son code par quelqu'un d'autre, afin d'éviter les erreurs. Les issues GitHub nous ont permis de séparer le projet en plein de petites étapes définies par des labels et permettre ainsi d'attribuer ces tâches aux différents membres du groupe afin d'avoir une gestion du temps et une organisation efficace. Nous avons aussi push sur des branches pour éviter les conflits, préserver la stabilité du code de la branche principale main en ne fusionnant les branches avec le main qu'après des vérifications et des tests poussés afin d'éviter les erreurs.

Le projet à rendre était divisé en 2. Cependant, ces 2 exercices sont identiques dans la conception mais diffèrent dans leur implémentation. Nous nous sommes alors dits que si nous réussissions à réaliser le 1er exercice, il nous serait alors plus facile d'implémenter le 2nd.

2 Organisation d'équipe et répartition des tâches

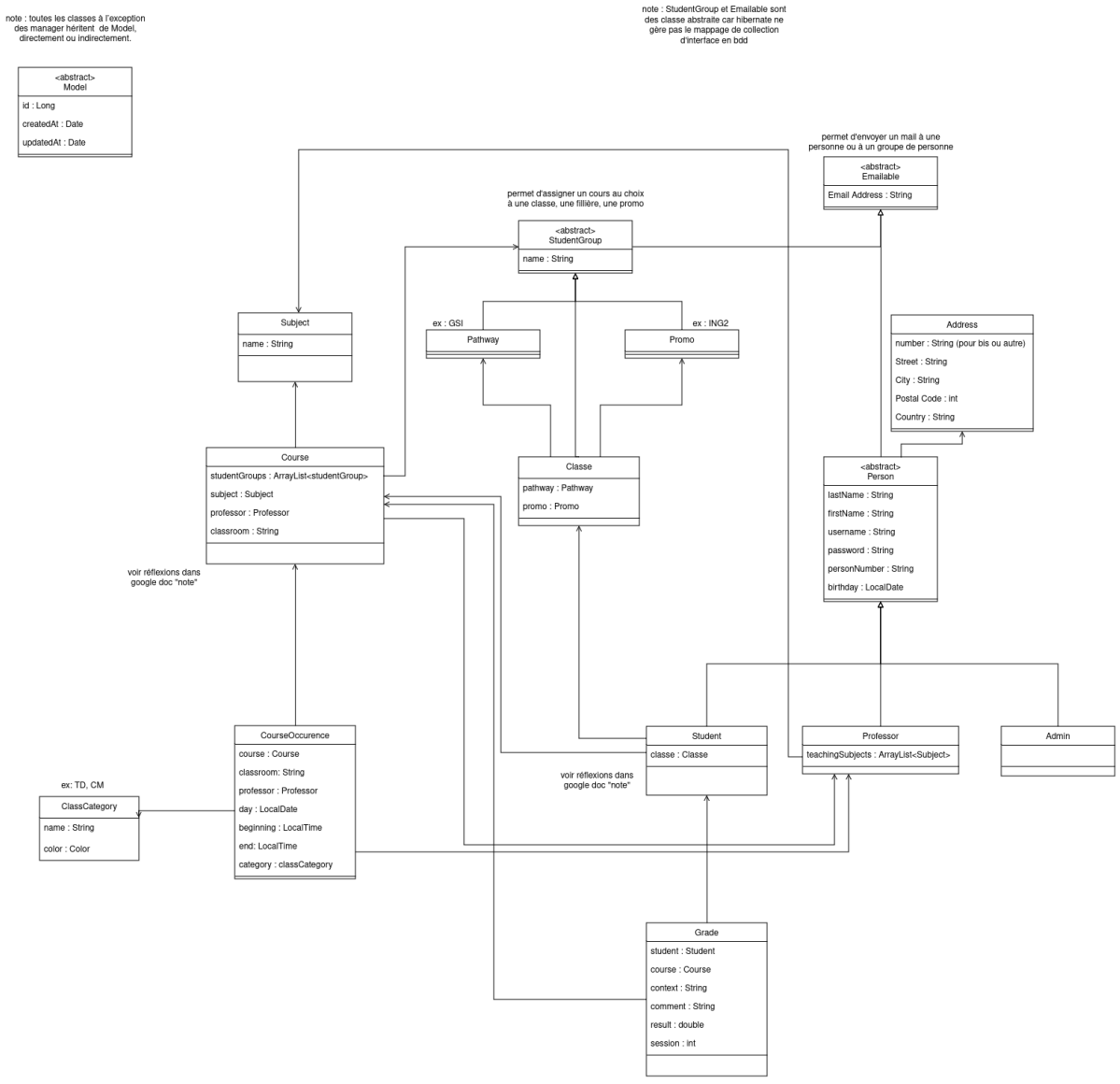
Gaétan a configuré le projet jakarta EE tandis que Julien a configuré le projet Spring Boot. De plus, ils ont tous les deux créés des issues GitHub en laissant les différents membres du groupe se les répartir entre eux. Pour le projet jakarta EE, Gaétan a travaillé sur une bonne partie des pages admin et la page de login ainsi que l'emploi du temps, Julien a quant à lui géré toute la partie gestion de notes c'est-à-dire la consultation et l'affichage ainsi que la génération en pdf du relevé de notes et le formulaire de saisie des notes pour les professeurs. Mateo a travaillé sur les entités cours et occurrences de cours, tandis que Guillaume a fait quelques pages admin. Pour le projet Spring Boot, la répartition des tâches était similaire aux différences près que Julien a travaillé en plus sur la page de login et la gestion des classes et Mateo s'est occupé en plus de la gestion des cours. Ulrich s'est quant à lui occupé d'une page admin et du rapport.

II Architecture du projet

1 Diagramme de classe

Nous avons commencé par la conception en réalisant un diagramme de classe. Ce diagramme a été notre support pendant la durée du projet. Il nous est même arrivé de le modifier en cours car l'implémentation était impossible.

Un exemple est la suppression des interfaces car nous avons trouvé que Hibernate ne gère pas les annotations pour mapper l'héritage avec des interfaces en base de données, comme précisé dans la documentation. Ainsi nous avons utilisé l'héritage et les classes abstraites pour pallier le problème.



pour consulter le diagramme de classes, vous pouvez cliquer ici (à visualiser avec draw.io)

Pour la partie Modèle, nous avons créé tous les objets tels qu'ils héritent de la classe abstraite Model. Cette classe Model a comme attribut la date de création et de dernière modification de l'objet, ainsi qu'un identifiant unique.

Parmi les objets, plusieurs avaient un e-mail comme attribut comme les personnes (admin, enseignant, étudiant, classes, promotions, filières). Nous avons donc créé une classe abstraite Emailable, qui a un champ email. Les objets ayant ce champ héritent donc de Emailable, qui hérite elle-même de Model.

De la même manière, plusieurs objets représentent des groupes d'élèves, nous avons donc créé une autre classe abstraite StudentGroup, dont héritent les classes Classe Promo et Pathway. La classe StudentGroup hérite directement de Emailable, car le projet est conçu de tel sorte qu'il soit possible d'envoyer un mail à des groupes de personnes.

Les autres objets (note, cours, adresse …) héritent directement de la classe Model.

Dans les différentes classes, nous avons implémenté leurs attributs, les getters et les setters. De plus, nous avons spécifié leur comportement dans la base de données comme la réaction de la base de données face à la suppression d'une instance comportant une clé étrangère ou encore les liaisons ayant des cardinalités supérieur à 1.

2 Généralités

Pour chacun des projets nous avons utilisé Maven, le framework Hibernate pour pouvoir gérer les objets dans la base de données, le serveur d'application Tomcat et une séparation du code dite en couche dite MVC (Model View-Controller), avec un système de SSR, Server-Side Rendering, qui signifie que le contenu HTML est généré dynamiquement sur le serveur avant d'être envoyé au client. Dans les deux projets, l'architecture est assez ouverte, elle est donc facile à faire évoluer et aisément maintenable sur le long terme.

Les views en JSP servent à l'affichage de l'interface graphique. On y retrouve le code html statique, ainsi que la partie chargée de le modifier dynamiquement.

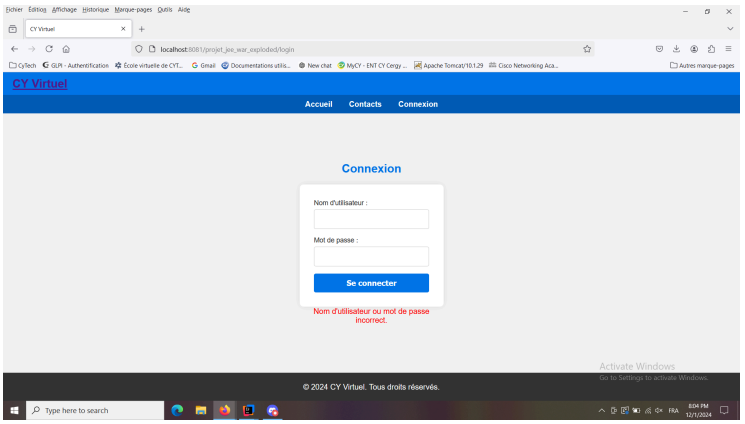
Un controller à un rôle similaire à un servlet, c'est la partie de l'application qui interagit directement avec les requêtes de l'utilisateur. Il traite les requêtes HTTP, et fait appel à la bonne fonction dans une autre couche du code.

La liste des fonctions qui sont appelées par les controllers, qui utilisent les objets de base du projet, et qui interagissent avec la base de données, se trouvent dans des classes appelées "services" qui implémentent la logique métier correspondant à une certaine requête de l'utilisateur.

Les repository de Spring Boot servent quant à eux à définir uniquement les fonctions de base pour interagir avec la base de données, ce sont des fonctions dites CRUD.

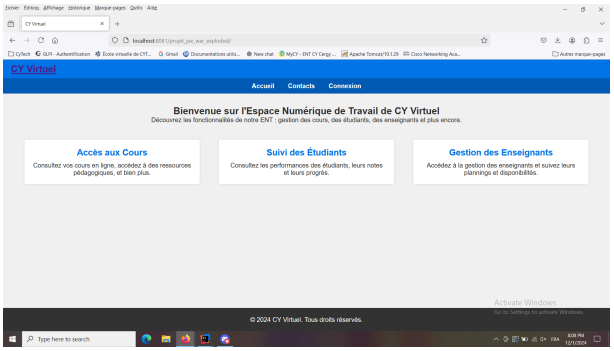
III Fonctionnalitées de l’application

La première fonctionnalité de cette application est l’authentification de l’utilisateur.

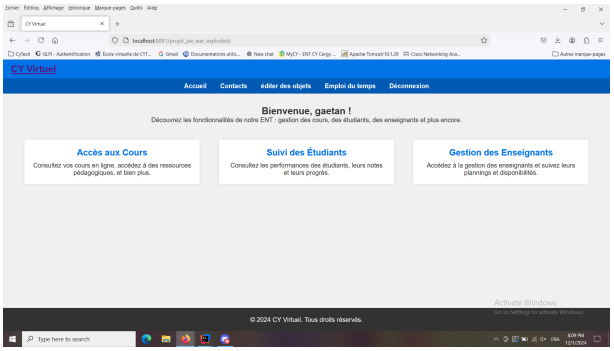


Cette image représente l’application lorsque l’identifiant ou le mot de passe est incorrect.

Cette page login fait la transition entre la page d’accueil avant et après l’authentification.

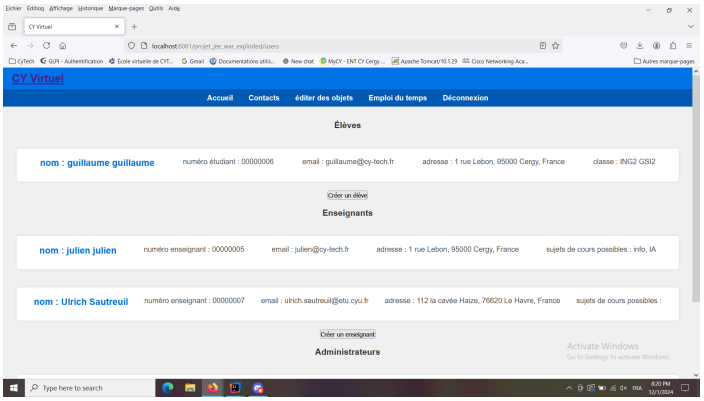


Avant l’authentification



Après l’authentification

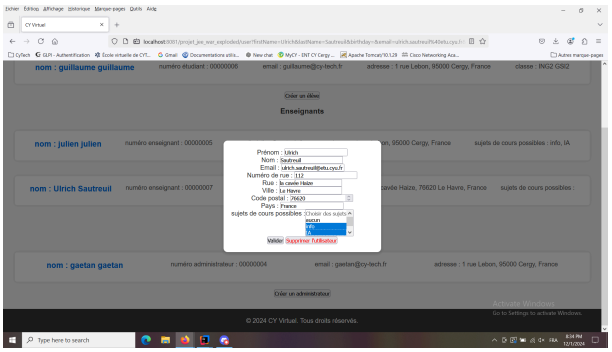
Ici l’utilisateur s’est authentifié en tant qu’administrateur. Un administrateur peut éditer les différentes entités en cliquant sur l’onglet “éditer des objets”. Les fonctionnalités pour éditer les objets sont similaires. Prenons le cas de la gestion des utilisateurs. Pour cela, il faut sélectionner la section “gérer les utilisateurs". La page qui en résulte est de cette forme.

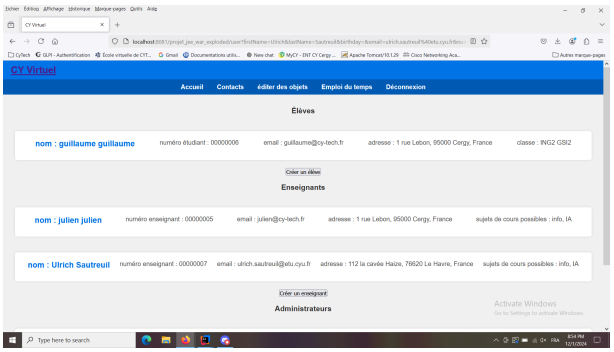


L’application nous affiche la liste des utilisateurs qui sont répertoriés dans la base de données et les classe selon leur rôle. Pour modifier ou supprimer un utilisateur déjà existant, il faut sélectionner cet utilisateur. Un popup va alors s’afficher avec les attributs remplis et 2 boutons. L’un pour valider les modifications et l’autre pour supprimer l’utilisateur. Le bouton situé sous chaque liste d’utilisateur permet de créer ce type d’utilisateur. La popup résultant est identique sauf pour la présence du 2ème bouton et le remplissage des champs.

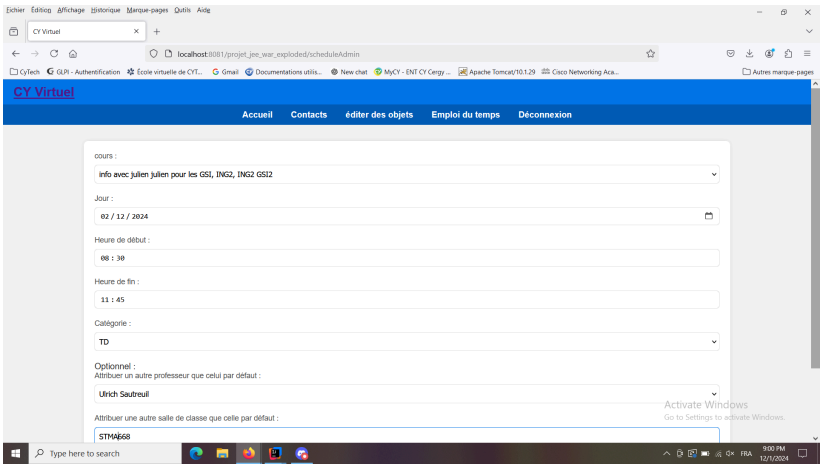
Dans le cas spécifique des enseignants, il est possible d’affecter un même enseignant à plusieurs cours.Par exemple, on peut créer un cours pour la classe ING2 GSI2, avec un professeur particulier, et créer un autre cours destiné à toute la filière GSI avec ce même professeur. De cette manière, on peut facilement assigner des cours à des groupes différents d'élèves. Il est également possible d’associer plusieurs groupes d'élèves à un seul cours. Un seul cours peut donc être associé aux ING2 GSI1, aux ING2 GSI2, et aux ING2 GSI3.

De cette manière, on peut facilement associer beaucoup d'élèves à un cours. Il serait par exemple possible de mettre un cours “discours du directeur” à la totalité des élèves de l’école en choisissant seulement les promos PREING1, PREING2, ING1, ING2 et ING3.



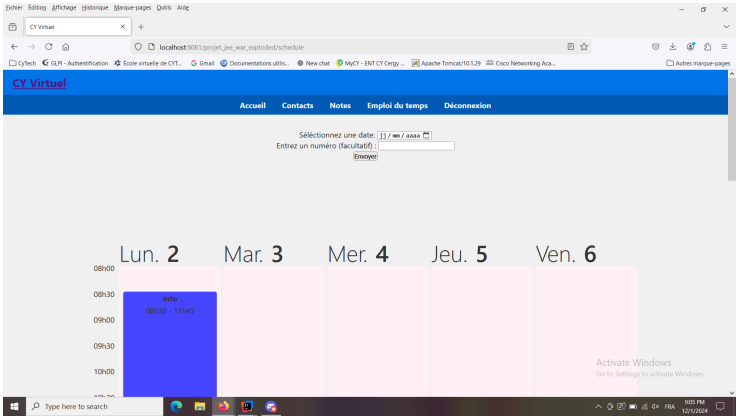


Un autre exemple d’action possible est la planification des cours, c'est-à-dire la gestion des occurrence d’un cours. Il faut au préalable avoir créé un cours grâce à la section “Gérer les cours”. De la même manière, on remplit les champs avant de valider.

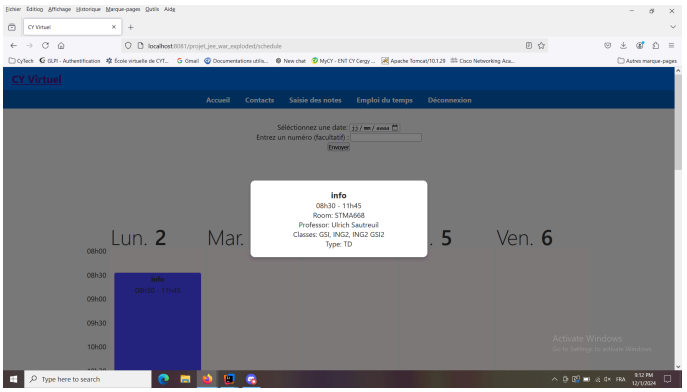


Sur l’application, un étudiant de cette classe a déjà été configuré. En se connectant sur le compte de l’étudiant et en cliquant sur l’onglet “Emploi du temps”, on obtient le résultat suivant.

On peut également consulter l’emploi du temps d’une personne donnée en étant connecté avec un compte administrateur, en spécifiant le numéro d’étudiant/de professeur/ de la personne. Ce numéro peut être retrouvé dans la page “gestion des utilisateurs”.

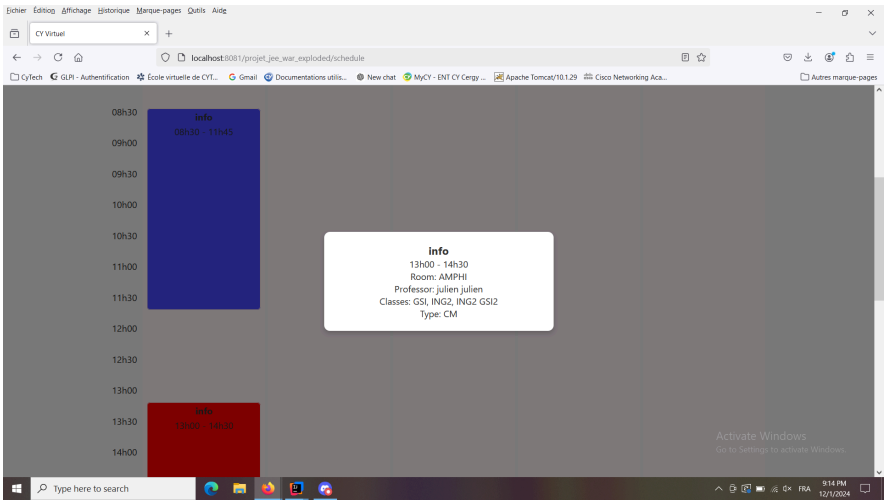


En cliquant sur le cours qui est affiché en bleu (car c’est un TD), une popup apparaît avec les informations du cours.



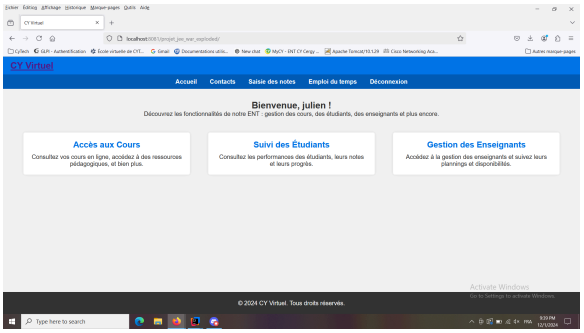
Ici, le professeur par défaut du cours “info” est le professeur Julien. Cependant il est possible de préciser qu’une occurrence du cours “info” aura un professeur particulier, en cas de remplacement par exemple. Dans notre cas, le professeur Ulrich est noté comme enseignant pour cette occurrence là du cours.

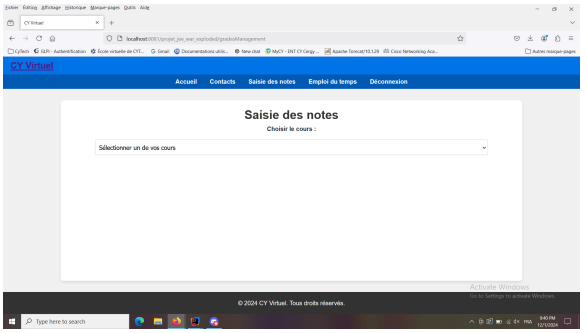
Si aucun professeur remplaçant n’avait été précisé, l’application aurait affiché le professeur par défaut du cours, c'est-à-dire Julien.



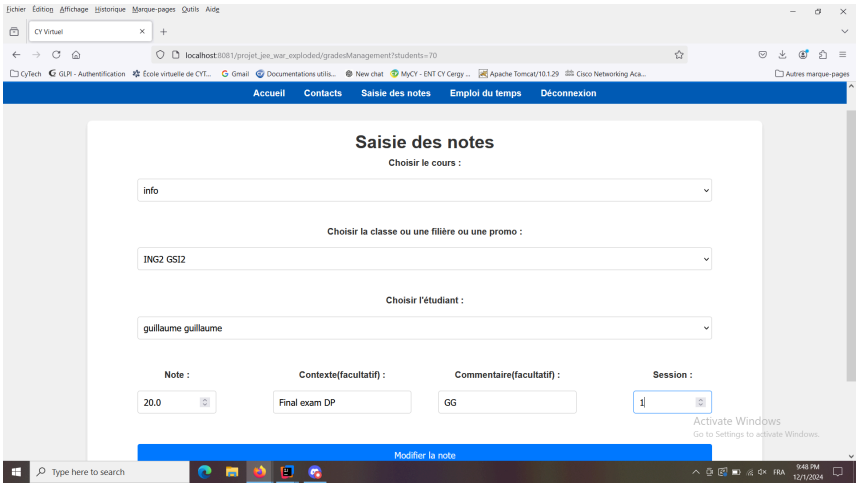
Cet emploi du temps est communiqué pour les étudiants et les professeurs concernés. La gestion des autres objets suit le même schéma.

La dernière fonctionnalité majeure de ce projet est la gestion des notes qui comprend la saisie des notes par les professeurs et leur visualisation par les étudiants. Pour commencer, il faut saisir une note par l'intermédiaire du compte d'un professeur.

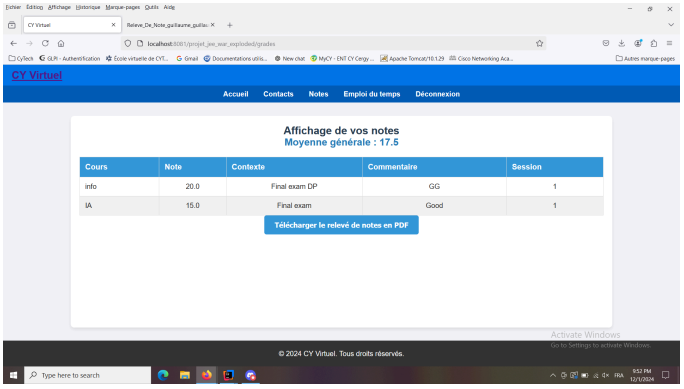


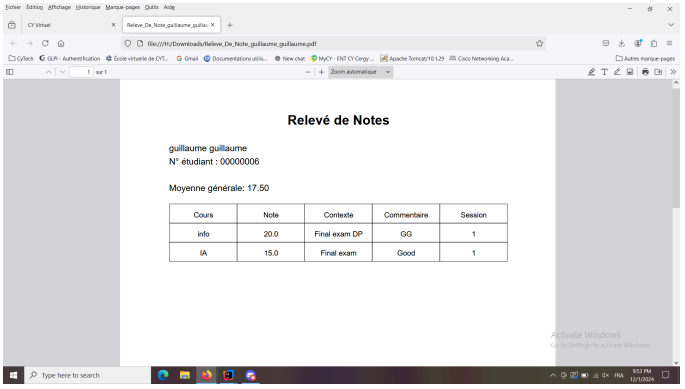


Les options pour saisir une note apparaissent graduellement car le cours limite le choix des groupes d'étudiants qui eux-mêmes limitent le choix de l'étudiant noté.



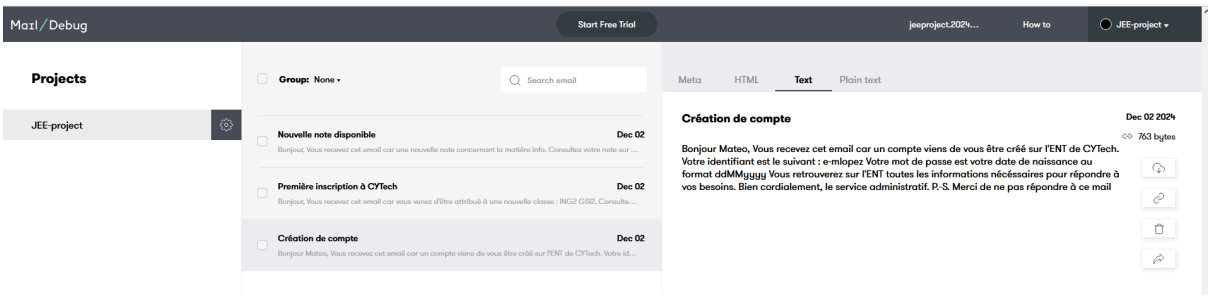
L'étudiant quant à lui, accède à ses notes en cliquant sur l'onglet "Note". Sur la page qui en résulte, l'étudiant peut voir chacune de ses notes avec leur détail. Il peut ensuite les imprimer en pdf.



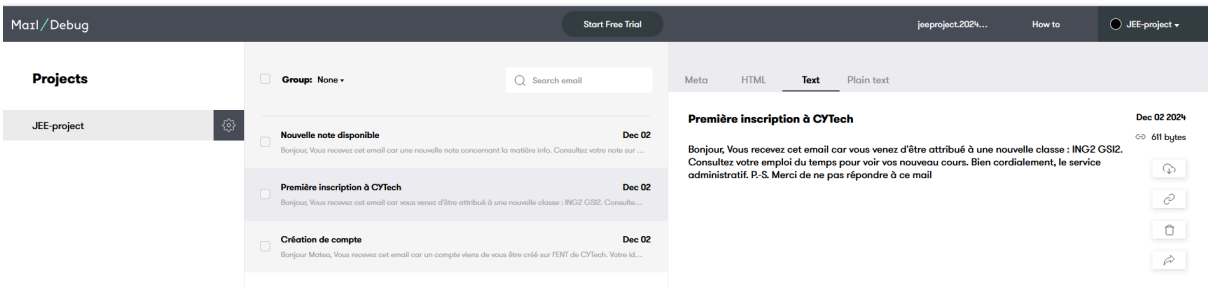


Nous utilisons debugmail.io pour avoir un système de mail fonctionnel. Voici le username : jeeproject.2024@gmail.com et le password : MyP@ssw0rd123 pour s'y connecter. Un mail est envoyé quand un compte est créé pour nous, avec les informations nécessaires pour se connecter. Un mail est aussi envoyé quand une classe ou une note est attribuée à un élève, respectivement par l'administrateur ou un professeur. Précisons que le mail est envoyé avec un minimum de formatage, mais que ce n'est pas restitué correctement par le site, et le texte apparaît donc sans aucun saut de ligne ou autre espacement. On peut voir le formatage en cliquant sur "Plain text", mais dans ce cas les caractères spéciaux ne sont plus supportés.

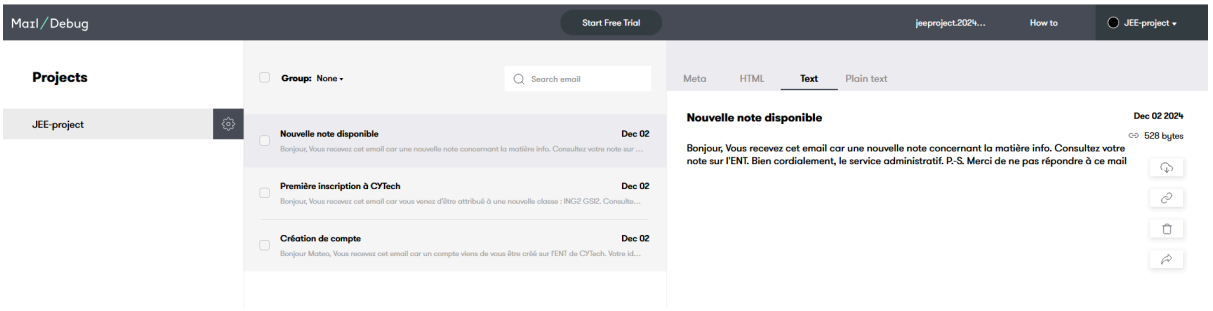
Exemple du mail de création de compte :



Exemple du mail d'attribution de classe :



Exemple du mail d’attribution de note :



IV Particularité des projets

1 Projet jakarta EE

Comme dit précédemment, nous avons utilisé dans ce projet la méthode MVC. Dans le projet Jakarta, les controllers sont appelés Servlet, et les services sont qualifiés de Manager.

Pour la partie vue, la page login permet de différencier les utilisateurs selon leurs rôles. La page index correspond à la page d'accueil avant et après la connexion de l’utilisateur. La page contact est la dernière page commune à tous les rôles (admin, enseignant, étudiant) et elle correspond à une page fixe qui affiche les données de la personne à contacter si l’application rencontre un problème. Enfin, il y a la page permettant de visualiser l’emploi du temps.

Les autres pages diffèrent selon le rôle de l’utilisateur. La plupart sont des pages réservées exclusivement à l’administrateur. En effet, ces pages permettent de créer, modifier et supprimer les différentes entités telles que les cours, les utilisateurs ou encore les classes. Ces pages renvoient vers des servlets qui eux-mêmes renvoient sur la page qui l’a appelée. La liste des instances présentes dans la base de données est affichée sur ces pages.

Les 2 dernières pages sont les pages liées aux notes. L’affichage de ses notes est exclusif à un étudiant et la saisie des notes est exclusive à l’enseignant.

La partie contrôle est quant à elle constituée des servlets qui font le lien entre les entités de la partie modèle et les pages de la partie vue. Elle est aussi constituée, dans notre projet, de classes qui possèdent le suffixe “Manager”. Il s’agit de classes regroupant les méthodes utilisées par les servlets.

Quelques servlets sont utilisés uniquement pour rediriger l’utilisateur vers une page définie comme ceux des pages “Contact” ou encore “Index”. Les servlets qui régissent les demandes de création, de modification et de suppression des entités gérées par l’administrateur utilisent 2 urls distinctes, ou bien la même url en différenciant les requêtes get ou post. L’une des requêtes indique à la servlet qu’il doit faire une modification dans la base de données, après avoir fait la modification demandée. Le servlet redirige alors vers lui-même avec l’autre url, qui sert à générer la page html à partir de la view jstl correspondante.

Le servlet pour gérer les notes est le même si l’utilisateur est un étudiant souhaitant aller sur l’onglet “Note” ou si l’utilisateur est un enseignant souhaitant aller sur l’onglet “Saisie des notes”. Le comportement de ce servlet est configuré de la même façon que les servlets liés à l’administrateur.

Il y a pour finir le servlet pour le login qui permet de différencier la partie vue selon le rôle de l’utilisateur.

2 Projet Spring Boot

Ce projet ayant une finalité identique au premier projet, l’architecture de ce projet est très similaire à celui-ci. En effet les parties modèles et vue sont identiques. La différence réside donc dans la partie contrôle.

Dans le projet Jakarta, nous avons eu recours à des servlets. Dans ce projet, ils ont été remplacés par des classes Java appelées controllers (voir la partie d’explication de la logique en couche dans la première partie du rapport).

Les managers du projet avec Jakarta ont eux été remplacés par la couche repository et service.