Mémoire de projet

Application Informatique – « Prise de rendez-vous »

Encadré par **Stéphane HÉRAUVILLE**

Intervenants :

Kevin BIOJOUT	Lucas FICKER
Anass EL GHARBAOUI	Abdoulaye FOFANA
Thomas HAMELIN	Angelo FOTI

I. Avant-propos

Ce document résume l'ensemble du travail que nous avons effectué afin de produire une application permettant la prise de rendez-vous lors d'événements organisés au sein de l'UFR de Sciences et Techniques de l'Université de Rouen Normandie.

Cette application repose sur le framework Spring et donc, par conséquent, le langage Java. Elle a été développée en six mois et peut être considérée comme prête à l'emploi avec l'ensemble des fonctionnalités demandées dans le cahier des charges.

Elle reste bien sûr perfectionnable, notamment sur la question de la cohérence de la charte graphique, ou encore de la sûreté des opérations (retours clairs à l'utilisateur), mais dans l'éventualité où l'application est utilisée en respectant les consignes, elle remplit parfaitement les attendus.

Le déploiement de ce projet se veut simple : une simple base de données MySQL est nécessaire afin de mettre en œuvre l'application. Ensuite, il suffit de définir les coordonnées de la base de données, puis de démarrer l'application.

Vous pourrez découvrir, tout au long de ce mémoire, l'ensemble des concepts à retenir pour comprendre l'architecture globale du projet ainsi que les subtilités.

II. Remerciements

Nous adressons nos remerciements à monsieur **Stéphane HÉRAUVILLE** pour son accompagnement et son encadrement durant toute la durée de ce projet.

Nous remercions également l'ensemble de **l'équipe pédagogique** de la 3e année de licence Informatique pour la qualité de ses formations.

Table des matières

I. Avant-propos 2
II. Remerciements 3
III. Contexte 6
IV. Analyse du besoin 7
A. Cahier des charges 7
B. Diagramme des cas d'utilisation 7
C. Importation et exportation de données 8
D. Impression des créneaux d'un événement 8
V. Organisation du travail 9
VI. Présentation de la stack technique 10
A. Git – Gestion de code source 10
B. Gantt – Réalisation de diagrammes 10
C. Spring – Framework au cœur de l'application 10
D. Thymeleaf – Moteur de rendu HTML 11
E. Lombok – Génération automatique de code source 11
F. MySQL – Système de gestion de bases de données relationnelles 11
G. IntelliJ IDEA – Environnement de développement intégré Java 11
H. Bootstrap – Framework CSS & JS pour la conception du front-end 12
I. Lucidchart – Conception de diagrammes 12
VII. Utilisation de l'écosystème Spring 13
A. Spring Initializr 13
B. Spring Web 13
C. Spring Validation 13
D. Spring Security 13
E. Spring Data JPA 13
VIII. Conception du frontend 14
A. Maquettage du produit 14
B. Intégration des templates 14
IX. Conception du backend 16
A. Conception du schéma de la base de données 16

B. Arborescence des modules 16
C. La représentation d'une donnée en Java : les entités 17
D. L'accès aux données : les Data Access Object avec Spring JPA 17
E. Incorporation de la logique métier : l'importance des services 18
F. Interception des requêtes web : le rôle des contrôleurs 18
X. Conclusion 19

III. Contexte

Le projet « Prise de rendez-vous » consiste en la réalisation d'une plateforme de prise de rendez-vous afin de faciliter l'organisation des diverses soutenances ayant lieu au cours de la licence Informatique. L'objectif est de réduire la charge de travail imputée aux enseignants dans l'organisation de ces soutenances en offrant un outil prêt-en-main leur permettant de publier un calendrier d'événements en quelques clics, de pouvoir gérer les étudiants inscrits pour ces événements et de pouvoir facilement exporter cette liste sous un format imprimable sur les portes de salles notamment. Le projet est composé d'un monolithe Spring Boot portant toute la charge technique de gérer les événements, les créneaux, les comptes ainsi que l'interface utilisateur. Le système de gestion de bases de données retenu est MySQL pour sa fiabilité au long terme ainsi que sa simplicité d'utilisation.

The project « Prise de rendez-vous » consists in the realization of a platform of appointment taking in order to facilitate the organization of the presentations taking place during the Computer Science bachelor. The objective is to reduce the workload of the teachers in the organization of these presentations by offering a ready-to-use tool allowing them to publish a calendar of events in a few clicks, to be able to manage the students registered for these events and to be able to easily export this list under a printable format on the doors of classrooms in particular. The project is composed of a Spring Boot monolith carrying all the technical stack of managing events, slots, accounts and the user interface. The database management system chosen is MySQL for its long-term reliability and ease of use.

IV. Analyse du besoin

A. Cahier des charges

Pour commencer le projet, nous avons commencé par définir le cahier des charges en définissant les besoins du client et les cas d'utilisations de l'application. Cela nous permet donc d'estimer nos différentes tâches au sein du projet. Cela nous permet également de se mettre d'accord avec le client sur ce qu'il attend de l'application, et de comprendre le fonctionnement de celle-ci. Après l'élaboration du cahier et sa validation par l'encadrant, nous avons pu commencer la réalisation des premières maquettes, le schéma initial de notre base de données, ainsi que la répartition des différentes tâches futures après concertation avec l'ensemble de l'équipe. Nous avons par la suite conçu un diagramme des cas d'utilisation afin d'y voir plus clair dans les objectifs à atteindre.

B. Diagramme des cas d'utilisation

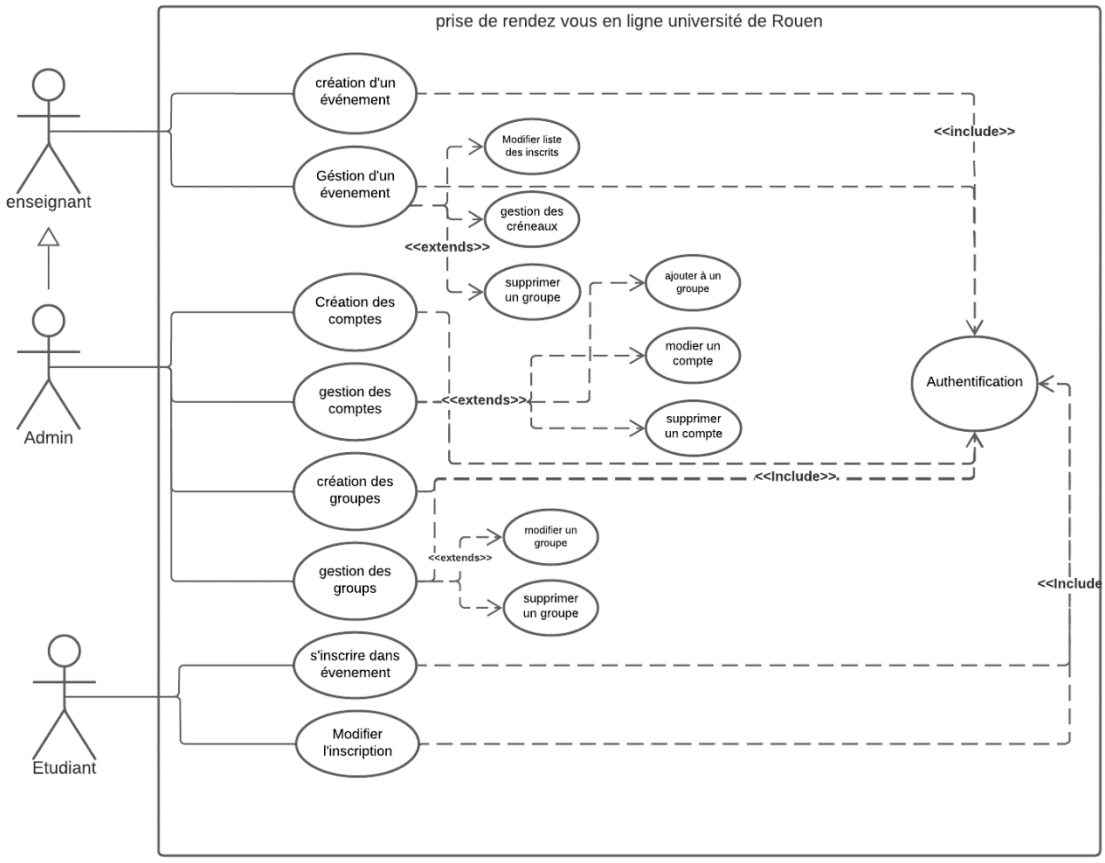

Figure 1 : Diagramme des cas d'utilisation.

Le diagramme des cas d'utilisation permet de donner une vision claire des interactions possibles de l'utilisateur sur l'application. Elles donnent une vision sur les fonctionnalités d'un système, d'un sous-système ou d'un composant plus particulièrement. On capture le comportement souhaité d'un utilisateur qui est acteur dans ce diagramme. Ainsi, on représente clairement l'ensemble des besoins de l'utilisateur à couvrir.

Dans le cadre de notre application, nous disposons de trois acteurs : l'administrateur, l'enseignant et l'étudiant. L'administrateur peut créer des comptes utilisateur, et gérer l'ensemble des événements de l'application. L'enseignant peut créer des événements et gérer les siens. L'étudiant peut quant à lui réserver un créneau disponible à un événement pour lequel il a accès, il est également en mesure de se rétracter si son créneau ne lui convient plus.

C. Importation et exportation de données

Une des fonctionnalités disponibles dans ce projet est l'importation de comptes étudiants depuis une liste CSV. L'idée derrière cette fonctionnalité est de rendre très accessible la création d'une grande quantité de comptes étudiants en une seule action.

Pour effectuer cette action, il vous faudra vous rendre sur la page de gestion des comptes, accessible aux administrateurs du système. Sur cette page vous trouverez un formulaire dans lequel vous pourrez insérer votre liste au format CSV. Une fois votre liste insérée, vous pourrez cliquer sur le bouton "Envoyer". Vous serez redirigé vers la page de gestion des comptes actualisée avec les nouveaux comptes, ou une erreur en cas d'erreur dans la liste CSV.

Cette fonctionnalité a été créée à l'aide de la bibliothèque de manipulation des fichiers CSV développée par la fondation Apache.

D. Impression des créneaux d'un événement

Le principal objectif de cette application est de pouvoir fournir une liste des créneaux d'un événement dans un support affichable par les professeurs. Dans ce but, une fonctionnalité d'impression a été développée. Vous retrouverez l'accès à cette fonctionnalité dans la page de gestion d'un événement, disponible pour les comptes de professeur, à travers un bouton nommé "Impression". Lorsque vous cliquez sur ce bouton, vous êtes alors redirigé vers l'impression d'un tableau contenant toutes les informations essentielles de l'évènement.

V. Organisation du travail

Nous avons réalisé un diagramme de Gantt afin de clarifier la répartition des tâches tout au long du projet. De la prise en main des technologies à la finalisation des dernières fonctionnalités, nous avons daté l'ensemble des étapes dans ce diagramme afin de pouvoir retracer le cours du projet facilement.

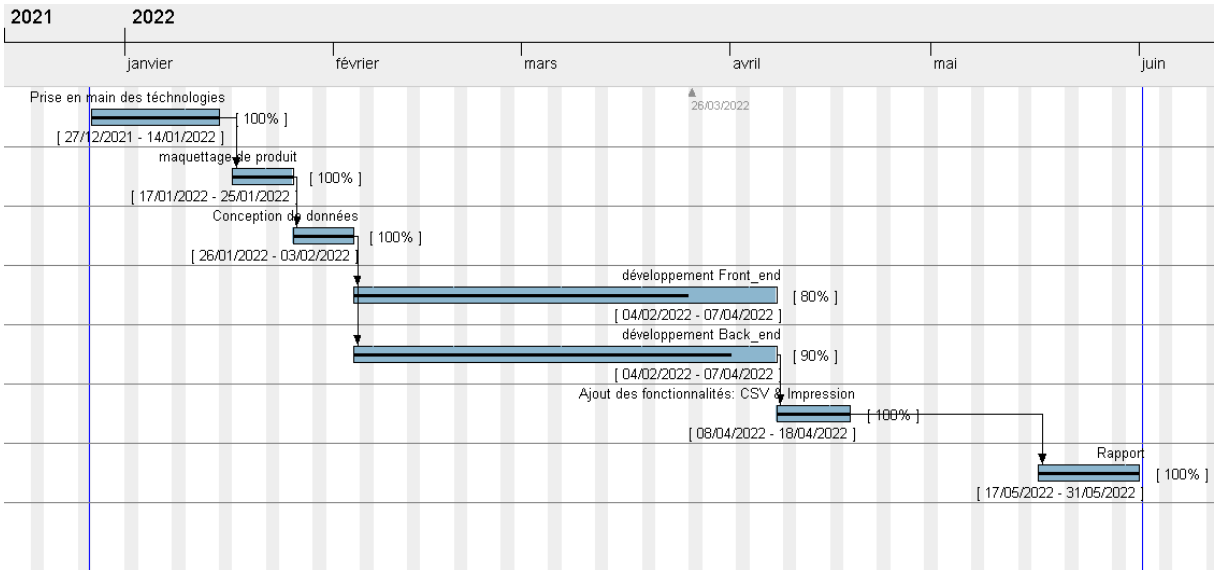

Figure 2 : Diagramme de Gantt.

VI. Présentation de la stack technique

L'application est composée d'un monolithe Spring Boot qui gère l'ensemble des fonctionnalités et des responsabilités du projet. Nous avons décidé d'utiliser les technologies suivantes pour mener à bien son développement :

A. Git – Gestion de code source

Git est un logiciel open-source de gestion de version décentralisée, il utilise un système de connexion pair à pair (modèle d'échange en réseau où chaque entité est à la fois client et serveur). Le code informatique développé est stocké sur l'ordinateur de chaque contributeur du projet ou sur un serveur dédié. C'est un outil de bas niveau qui se veut simple et performant, dont la principale tâche est de gérer l'évolution du contenu d'une arborescence.

B. Gantt – Réalisation de diagrammes

Le diagramme de Gantt est un outil très populaire pour planifier les différentes actions d'un projet. Il permet d'avoir une vision claire des objectifs et des attendus à des dates données.

C. Spring – Framework au cœur de l'application

Spring Boot est un framework Java permettant de faciliter l'architecture et l'infrastructure d'un projet. Il est doté de principes très populaires tels que l'inversion de contrôle, la programmation orientée aspect, ainsi qu'une couche d'abstraction pour ne pas se soucier des détails techniques et donc de se focaliser sur l'essentiel. Il nous a été extrêmement utile dans le sens où il a très certainement grandement réduit le temps nécessaire à la réalisation de ce projet.

D. Thymeleaf – Moteur de rendu HTML

Thymeleaf est un moteur de rendu HTML, il permet à partir de templates de gérer des pages HTML dynamiquement avec des données fournies par l'application Java. Ainsi nous pouvons rendre du contenu personnalisé à l'utilisateur, ce qui est essentiel dans une application de ce type.

E. Lombok – Génération automatique de code source

Lombok est une librairie Java très appréciée des développeurs puisqu'elle permet d'éviter le « boilerplate » Java. Nous n'avons plus besoin de rédiger les accesseurs en lecture et en écriture, une simple annotation de classe suffit à les obtenir. La magie de Lombok s'exécute à la compilation du code source Java, mais les IDE fournissent une auto complétion très agréable pour développer.

F. MySQL – Système de gestion de bases de données relationnelles

MySQL est un système de gestion de bases de données relationnelles développé par Oracle. Nous l'utilisons dans le cadre du projet pour la persistance des données. Ainsi, nous sommes en mesure de stocker des données de manière pérenne.

G. IntelliJ IDEA – Environnement de développement intégré Java

IntelliJ IDEA est un environnement de développement intégré spécialisé dans l'univers Java. Il fournit tout un éventail d'outils et d'intégrations qui permet de faciliter la vie du développeur. Ainsi, nous pouvons nous focaliser sur l'essentiel qui est la conception de notre produit au lieu de perdre du temps avec les configurations entre autres.

H. Bootstrap – Framework CSS & JS pour la conception du front-end

Bootstrap est un framework de Design System CSS et JavaScript permettant de réaliser une charte graphique très rapidement et facilement. Il nous a facilité le développement des pages web responsive par l'utilisation de systèmes de grille.

I. Lucidchart – Conception de diagrammes

Lucidchart est une plateforme de conception, elle permet aux développeurs de créer et partager des diagrammes et des différents schémas conceptuels, cet outil nous a aidé à avoir une vision profonde sur l'architecture et les fonctionnalités de notre application.

VII. Utilisation de l'écosystème Spring

Nous avons utilisé un certain nombre de librairies faisant partie de l'écosystème Spring afin de compléter les fonctionnalités offertes par le framework.

A. Spring Initializr

Tout d'abord, pour initialiser le projet, nous nous sommes rendus sur l'outil *Spring Initializr* qui nous permet de générer un projet vierge avec les différentes dépendances que nous avons choisies, mais aussi la version de Java que nous utilisons qui est Java 17.

B. Spring Web

Spring Web est une dépendance pour notre projet qui permet de développer des applications Web, qui nous permet d'utiliser Spring MVC (Model-View-Controller), et nous permet aussi de connecter notre application à un serveur web comme Tomcat ou Jetty. Dans le cadre de notre application, notre choix s'est porté sur Jetty.

C. Spring Validation

Cette dépendance nous est utile dans l'application afin de vérifier que l'utilisateur remplit correctement les champs des différents formulaires. Par exemple, lorsque l'utilisateur remplira un formulaire demandant une adresse mail, une erreur sera envoyée si ce qui est entré n'est pas au format d'une adresse mail.

D. Spring Security

Spring Security nous permet de gérer toute la partie authentification de l'application. Cela nous aide donc pour la gestion de la connexion, de la déconnexion, ainsi que la gestion des sessions utilisateurs.

E. Spring Data JPA

Spring Data JPA permet d'implémenter l'API JPA *'Java Persistance' API* basé sur des références en fournissant une couche d'abstraction à l'application, cela fait partie des modules qui facilitent la création et la gestion des base de données et des tables de la base de données, ainsi que l'accès aux données.

VIII. Conception du frontend

Le frontend est la partie de l'application avec lequel l'utilisateur interagit directement. Dans notre application, il s'agit de tout ce qui compose les templates Thymeleaf. Ainsi, nous retrouvons l'expérience utilisateur et l'interface utilisateur.

A. Maquettage du produit

La première étape dans la réalisation du projet a été la réalisation de maquettes de rendu du site web représentant un aperçu de ce à quoi doit ressembler le projet terminé.

La maquette que nous avons choisi pour l'application est une interface graphique sous la forme d'un tableau de bord, où l'utilisateur peut se connecter et rediriger vers son espace où il peut effectuer toutes les opérations autorisées.

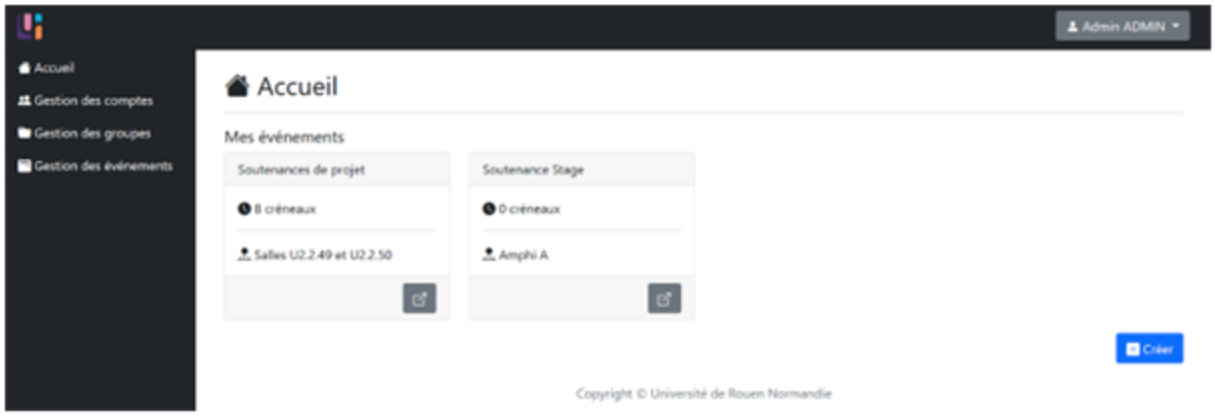

Figure 3 : Extrait de l'interface utilisateur.

B. Intégration des templates

Thymeleaf est le moteur de rendu HTML que nous avons utilisé pour gérer l'affichage des vues de l'application. Il est extrêmement simple d'utilisation puisqu'il ne requiert aucune configuration additionnelle lorsqu'il est utilisé avec Spring Boot, ce qui est notre cas, et aussi parce que ses templates sont une extension de HTML : cela signifie que tout code source HTML valide, est également un template Thymeleaf valide.

Il est alors très simple d'intégrer nos maquettes statiques à Thymeleaf en remplaçant au fur et à mesure les données statiques par des instructions SpEL (pour Spring Expression Language) nous permettant d'y insérer des données personnalisées.

La personnalisation de ces données se fait au niveau du contrôleur, dans lequel nous pouvons utiliser l'objet Model nous offrant une interface nous permettant d'ajouter

des variables qui seront ensuite disponibles au moment de la construction de la page finale.

À l'aide de cette technologie, nous sommes en mesure de tirer pleinement partie de Spring MVC en renvoyant du contenu personnalisé aux utilisateurs de l'application.

IX. Conception du backend

A. Conception du schéma de la base de données

Nous avons commencé par la conception du modèle de la base de données avant de passer à la programmation du backend de l'application. Nous avons pour cela utilisé le cahier des charges, qui nous permet de définir les différentes fonctionnalités de l'application et ainsi d'adapter notre modèle pour qu'il réponde efficacement aux demandes de l'application. Avoir une base de données aboutie permet de faciliter l'implantation des différentes fonctionnalités.

Pour notre base de données, nous avons opté pour MySQL car nous avions des bases solides dans cette technologie. De plus, MySQL est l'une des technologies les plus simples à mettre en place pour l'installation de l'application.

Notre base de données a donc été créée dans le but d'associer un créneau d'événement à un étudiant. De même, cela nous permet de définir des comptes étudiants et enseignants qui sont utilisés dans le système de connexion à l'application.

B. Arborescence des modules

Le backend de l'application est donc composé de contrôleurs, de services, de répertoires, d'entités ou d'objets de transfert de données. Ces différents composants sont architecturés selon l'architecture du *Domain Driven Design* : une arborescence de modules correspondant aux périmètres de chaque partie. Ainsi, nous retrouvons par exemple un module pour la gestion des utilisateurs, un autre pour celle des événements ou encore pour les créneaux.

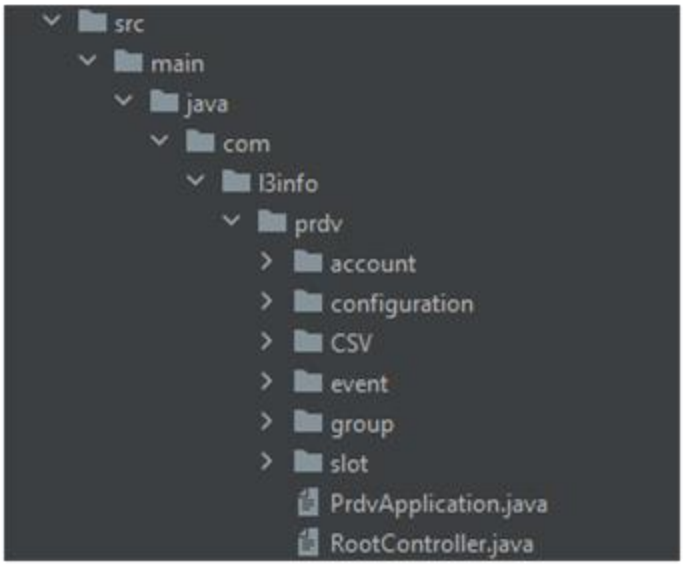

Figure 4 : Arborescence du projet.

C. La représentation d'une donnée en Java : les entités

Les entités sont des POJO (pour Plain Old Java Object) qui représentent une entrée de table dans une base de données. Elles sont dotées d'attributs qui représentent chacun une colonne dans la base (sauf si elles sont marquées @Transient), et possèdent des accesseurs en lecture et en écriture afin de pouvoir muter les données comme bon nous semblent.

Elles sont une pièce maîtresse dans la réalisation d'une application ayant à charge le stockage de données. Il est donc nécessaire de bien les penser dès le début afin d'éviter des modifications trop importantes en cours de réalisation du projet.

Nous disposons donc de trois entités : Account, afin de gérer les comptes utilisateur ; Event, afin de gérer les événements ; ainsi de Slot, pour gérer les créneaux. Certaines de ces entités ont des relations avec d'autres : elles peuvent être de type Many-to-Many, One-to-Many ou encore Many-to-One. Spring Data JPA s'occupe de gérer les relations, ainsi nous avons juste à manipuler des listes ou des attributs comme si nous gérions de simples classes Java.

D. L'accès aux données : les Data Access Object avec Spring JPA

Les répertoires sont un composant clé lorsqu'il s'agit de gérer la persistance des données. Ce sont eux qui effectuent le lien entre le système de gestion de bases de données (en l'occurrence MySQL pour ce projet), ainsi qu'un service ou tout autre code manipulant ces données. Il s'agit du dernier maillon dans notre application avant la reprise en main par le framework, puis la base de données.

La particularité des répertoires Spring Data JPA est qu'ils sont facilement extensibles : nous pouvons rajouter des requêtes à notre guise en étendant l'interface. Par exemple, nous pouvons rajouter la méthode Optional<Account> findAccountByEmail(String email) : sans aucune autre action de notre part, Spring Data JPA génère la requête MySQL correspondante.

Bien sûr, comme l'automatisation n'est jamais parfaite, il se peut que l'on ait à écrire les requêtes par nous-même afin d'optimiser ces requêtes, ou encore pour réaliser des fonctionnalités un peu trop exotiques pour le framework. Par exemple, nous avons eu à réaliser une requête nous permettant de récupérer la liste des événements pour lesquels un utilisateur cible a un créneau à son actif. Comme nous n'avons pas pu réaliser une telle fonctionnalité "automatiquement", nous avons donc écrit celle-ci à la main grâce à l'annotation @Query qui nous permet de personnaliser le comportement d'une méthode de l'interface.

E. Incorporation de la logique métier : l'importance des services

Les services portent l'ensemble de la logique métier dans une application, de quelconque nature qu'elle soit. Ils regroupent l'ensemble des fonctionnalités susceptibles de modifier des données dans l'application. En général, un service s'intercale entre le contrôleur et le répertoire connectant une entité dans la base de données. En bref, c'est celui-ci qui réceptionne les données de l'utilisateur, effectue les modifications adéquates en base de données, puis renvoie une réponse cohérente à ce dernier.

Dans une démarche de réaliser une application respectant le principe du Domain Driven Design : c'est-à-dire qu'on sépare les services en fonction de leur domaine d'expertise. Ainsi, on retrouve un service pour gérer les comptes utilisateur, un autre service pour la gestion des événements, ainsi qu'un dernier pour la gestion des créneaux. Accessoirement, nous pouvons rajouter un service contenant du code métier lié à l'import de listes d'étudiants sous la forme d'un fichier CSV.

F. Interception des requêtes web : le rôle des contrôleurs

Le contrôleur est un composant essentiel dans une application web. Ce sont eux qui interceptent les requêtes réalisées vers le serveur. Ils portent la responsabilité de fournir une réponse fiable et interprétable par le client.

Dans notre application, nous renvoyons principalement du contenu HTML : étant basé sur l'architecture MVC, notre application fournit directement la vue à l'utilisateur, contrairement à des applications avec une distinction entre frontend et backend.

Cependant, nous avons retenu l'architecture des routes typique du principe REST (pour REpresentational State Transfer). Ainsi, nos routes peuvent prendre cette apparence : */events/12/slots/14* pour désigner le créneau d'identifiant 14 de l'événement d'identifiant 12. Cela nous permet d'avoir une vue claire des routes à rendre, et nous sert également dans le cadre de débogages pour situer les problèmes plus facilement.

X. Conclusion

Ce projet nous a permis de découvrir une grande diversité de technologies et de façons de travailler. Premièrement, le travail en équipe et notamment les outils collaboratifs tel que *git* qui nous a permis de garder notre code en sécurité tout en collaborant à plusieurs dessus. Deuxièmement, nous avons découvert le framework *Spring* qui simplifie énormément le développement d'applications en Java : celui-ci fournit un grand nombre d'intégrations nous permettant d'éviter de réinventer la roue.

Bien que nous ayons rempli l'intégralité du cahier des charges initial, afin d'obtenir une plateforme utilisable aisément, il nous semblerait judicieux d'ajouter certaines fonctionnalités. Parmi elles, nous pouvons noter l'envoi de messages électroniques de rappel aux étudiants n'ayant pas réservé de créneau à un événement, ou encore la synchronisation de la gestion des comptes avec le *Central Authentication Service* de l'Université de Rouen Normandie, ou encore la synchronisation des créneaux réservés par un étudiant sur *ADECampus* afin qu'ils soient facilement visibles par ce dernier.