



**UNIVERSITÉ
DE LORRAINE**



nancy

Charlemagne
Informatique

IUT Nancy Charlemagne
Université de Lorraine
2 ter boulevard Charlemagne
BP 55227
54052 Nancy Cedex
Département informatique

Conception et Développement d'une Plateforme Interactive pour la Planification des Emplois du Temps

Rapport de Fin de Semestre
Association : Well Tennis Club

Raouf Achraf
Pedron Matheo
Ketzinger Tom
Micheli Thomas
Année universitaire 2024–2025



**UNIVERSITÉ
DE LORRAINE**



IUT nancy **Charlemagne**
Informatique

Projet tutoré

Présenté à
IUT Nancy Charlemagne

Filière :
BUT INFORMATIQUE

Conception et Développement d'une Plateforme Interactive pour la Planification des Emplois du Temps

Réalisé par :

- Raouf Achraf
- Pedron Matheo
- Ketzinger Tom
- Micheli Thomas

Encadré par :

- Mr. Borne Michaël
- Mr. Perrin Olivier

Année Universitaire : 2024-2025

Responsable de parcours : Debled-Rennesson Isabelle

REMERCIEMENT

Cher(e)s membres du jury,

C'est avec une profonde gratitude que nous prenons la plume aujourd'hui pour vous présenter notre rapport de fin de semestre, qui retrace l'avancement du projet tutoré que nous avons réalisé jusqu'à présent au sein de notre école.

Tout d'abord, nous tenons à exprimer notre reconnaissance envers **notre établissement**, qui nous offre l'opportunité de développer ce projet dans un cadre académique enrichissant. Nous souhaitons exprimer notre sincère gratitude à **Monsieur Borne Michaël** et **Monsieur Perrin Olivier**, nos professeurs référents, pour leur accompagnement, leur disponibilité et leurs précieux conseils tout au long de cette expérience.

Nous adressons également nos remerciements à **Madame Debled-Rennesson Isabelle**, responsable de notre parcours et membre du jury, pour son implication et son soutien tout au long de notre formation.

Enfin, nous remercions l'ensemble des enseignants et encadrants de notre formation pour leur engagement et leur investissement, qui nous permettent d'acquérir les compétences essentielles à la poursuite de ce projet et à notre développement académique.

TABLE DES MATIÈRES

Introduction générale.....	6
Analyse.....	7
Travail Réalisé.....	12
TimeFold.....	12
1. Création du domaine.....	12
2. Mise en place des contraintes.....	12
3. Mise en place du solveur et résolution.....	13
4. Liaison avec le serveur Spring.....	14
Back-End.....	15
1. Gestion de la base de données.....	15
2. Lien entre Spring boot et la base de données.....	16
3. Lien entre Spring boot et le front-end.....	16
4. Mise en place de la sécurité.....	17
5. Tests de validation.....	18
Front-End.....	18
1. Affichage de la plateforme.....	18
2. Fonctionnalités développées.....	20
2.1 Affichage des données Joueurs / Entraîneurs.....	20
2.2 Affichage des séances.....	20
2.3 Afficher les contraintes des terrains / séances.....	21
2.4 Importer / Exporter des données.....	21
2.5 Page d'inscription.....	22
2.6 Afficher / Valider les inscrits.....	22
2.7 Affichage Mobile.....	23
2.8 Filtrage des données.....	24
2.9 Possibilité de modifier les informations des joueurs et entraîneurs.....	24
2.10 Connexion sécurisé.....	24
2.11 Sécurisation des url.....	24
3. Tests de validation.....	24
4. Difficultés rencontrées.....	25
Planning.....	26
Itération 1 : Premières mises en place.....	26
Itération 2 : Intégration des fonctionnalités et améliorations.....	26
Itération 3 : Consolidation et optimisation des fonctionnalités.....	27
Itération 4 : Finalisation et sécurisation du projet.....	27
Répartition du travail.....	29
Réflexion sur l'organisation et ajustements nécessaires.....	29
Pour les prochaines itérations.....	30
Les éléments originaux.....	31
Mathéo.....	31
Thomas.....	31
Achraf.....	31
Tom.....	32
Objectifs à atteindre.....	33
Itération 5 : Optimisation des fonctionnalités et corrections.....	33
Itération 6 : Finalisation des fonctionnalités et automatisation.....	33
Itération 7 : Déploiement et documentation.....	34
Conclusion.....	35

Introduction générale

Ce document constitue notre **rapport de fin de semestre** et retrace l'ensemble du travail accompli dans le cadre de notre **projet tutoré**. Il présente les différentes étapes de conception, de développement et de gestion du projet, ainsi que les défis rencontrés et les solutions mises en place.

Au début de l'année, nous avons dû choisir un projet encadré par les professeurs et prendre en charge l'intégralité de son développement, depuis l'analyse jusqu'au déploiement. Parmi la liste des projets proposés, nous avons sélectionné celui qui nous semblait le plus intéressant : la création d'une application permettant la génération d'un emploi du temps annuel, en intégrant plusieurs contraintes transmises par le client.

L'objectif principal de ce projet est de concevoir un système de gestion des emplois du temps hebdomadaires pour les entraînements du **Well Tennis Club**, en tenant compte des spécificités du club. L'outil vise à automatiser jusqu'à 90 % de la planification, en s'adaptant aux niveaux, aux âges et aux disponibilités des joueurs, tout en respectant des contraintes fixes telles que la disponibilité des terrains, des entraîneurs et des infrastructures.

Afin de mener à bien cette mission, nous avons bénéficié de plusieurs supports fournis par le client, notamment un diaporama détaillant les objectifs du projet, un tableau regroupant les données des joueurs et un exemple d'emploi du temps. Ces documents nous ont servi de base de travail et nous ont permis d'orienter notre réflexion et notre méthodologie.

Dans ce rapport, nous présenterons l'ensemble des étapes que nous avons suivies jusqu'à présent, en détaillant les choix techniques effectués, les défis rencontrés ainsi que les solutions mises en place pour assurer une gestion efficace et optimisée des séances d'entraînement au sein du club.

Analyse

Pour ce premier diagramme, nous retrouvons le diagramme de classe du domaine Timefold. Nous y trouvons une grande similarité avec la base de données puisqu'en effet, le domaine s'en inspire pour éviter une conversion trop problématique d'un type à l'autre. Le diagramme de classe fourni en itération 1 (après avoir étudié le fonctionnement de Timefold) décrivait déjà dans l'idée cette structure. Les principales modifications se font voir sur les noms qui sont devenus des UUID et l'ajout de nouvelles méthodes de traitement. L'élément le plus notable est l'ajout de l'attribut "sessionConstraint" du joueur qui représente son type de groupe.

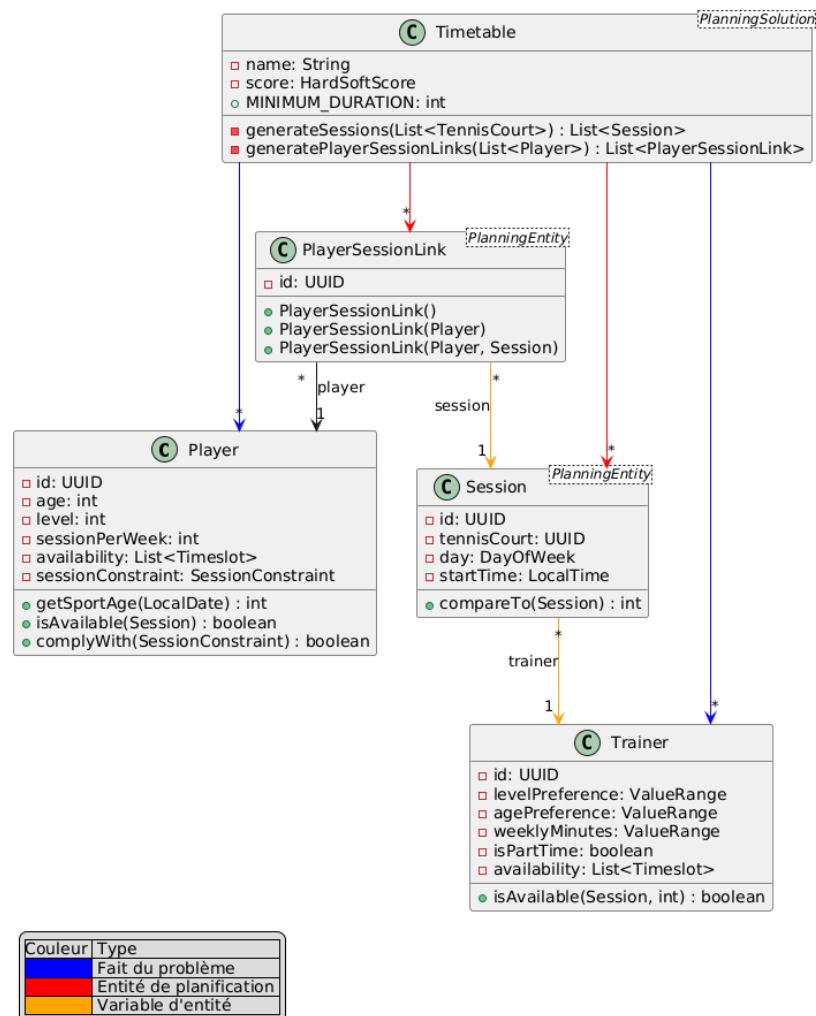


Figure 1 : Diagramme de classe du domaine Timefold

Ce second diagramme, permet ici de décrire le fonctionnement de la communication entre le client et Timefold. Il est à noter que ce diagramme présente quatre méthodes : le lancement ; le statut ; la sauvegarde ; l'arrêt, en effet la méthode pour obtenir les sessions générées par Timefold n'est pas gérée par le contrôleur de Timefold, mais bien celui des sessions, Timefold s'occupera des générer et des sauvegarder uniquement. Ce diagramme de séquence présente donc le déroulement des quatre méthodes à la suite, nous n'y présentons pas les cas d'erreur pour ne pas surcharger inutilement le diagramme. Effectivement, si une requête n'est pas tolérée, Spring ne déroule pas l'action et retourne simplement le code d'erreur adéquat.

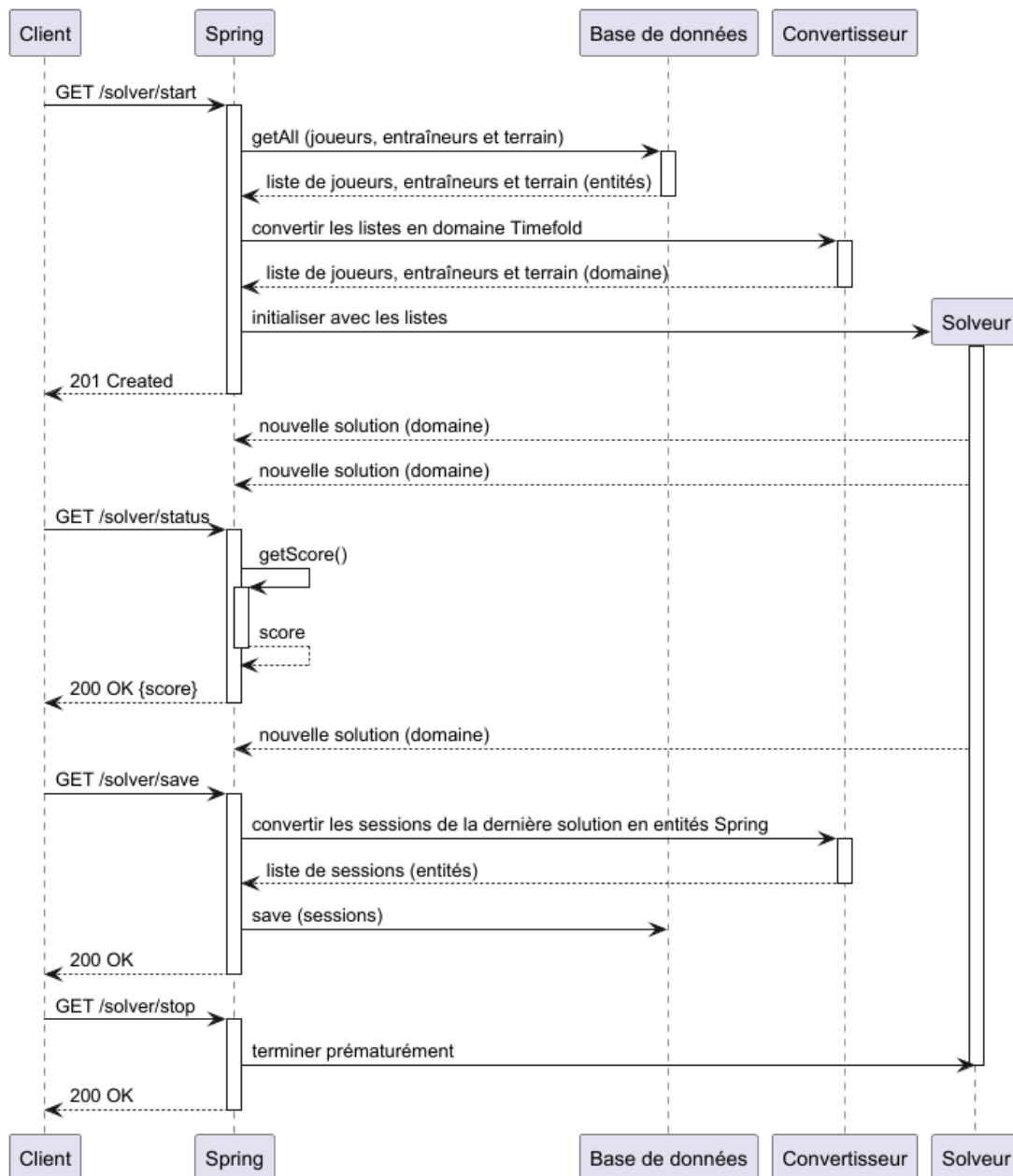


Figure 2 : Diagramme de séquence des quatre requêtes du contrôleur de Timefold

Pour ce qui est de la base de données, nous avons prévu d'utiliser une base de données MongoDB pour sa flexibilité. Mais lors de la première itération nous avons décidé de changer et de partir sur une base de données SQL qui a l'avantage de mieux gérer les clés primaires, les clés étrangères et la duplication de données. Voici donc ci-dessous le schéma relationnel de la base de données. La table Session_constraint n'est reliée à aucune table puisque cette table ne sert qu'à stocker les contraintes qui seront utilisées par Timefold

1

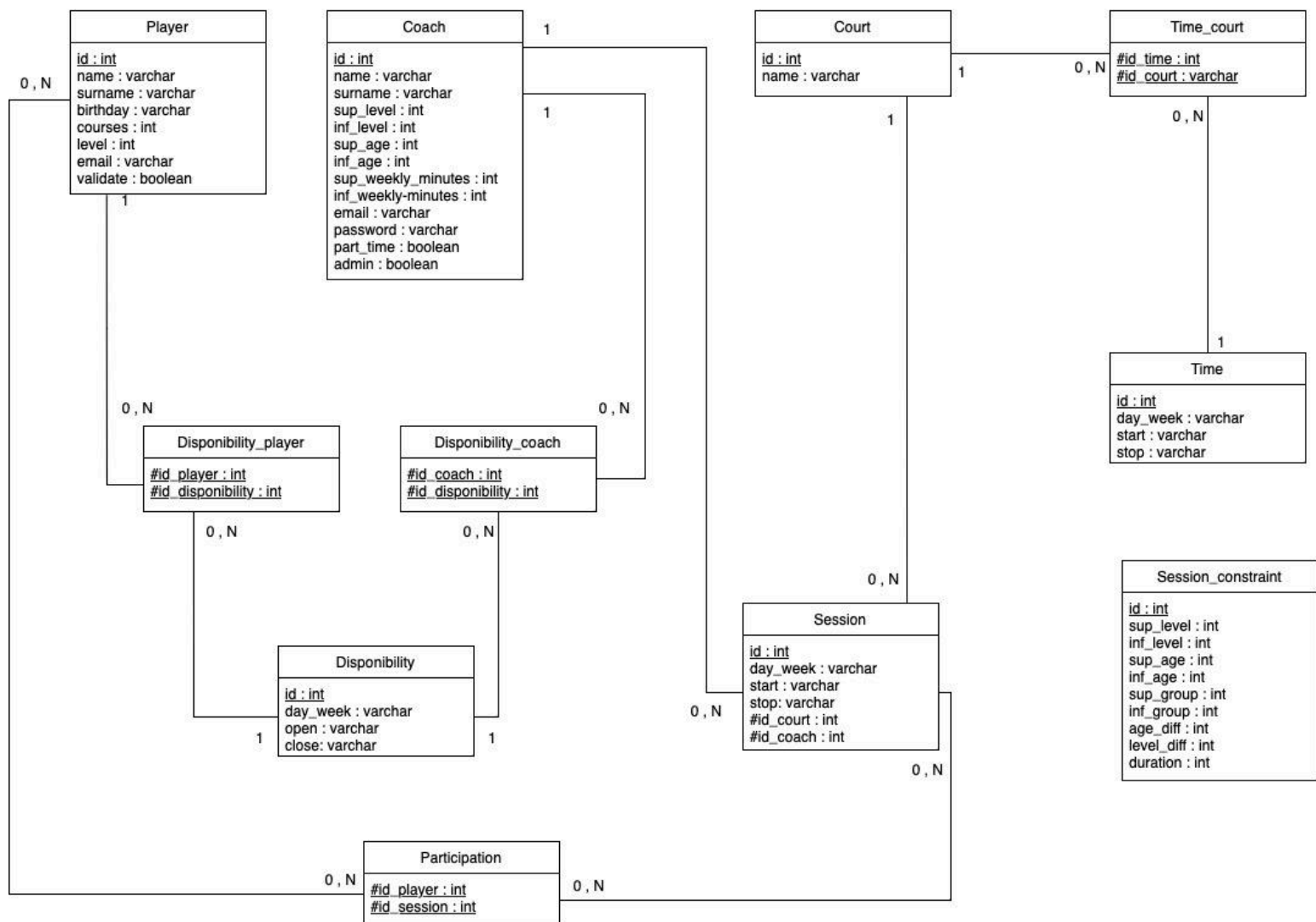


Figure 3 : Schéma relationnel de la base de données

Pour le backend nous avons également décidé de faire une API Rest grâce à Spring boot. Ce choix a été fait car on utilise Java, un langage que nous connaissons bien car on l'a beaucoup

étudié en cours, parce que Thomas avait déjà utilisé cette technologie lors de son stage et aussi parce que c'était compatible avec Timefold. Voici ci-dessous un diagramme de séquence d'une quelconque requête venant du frontend vers le backend.

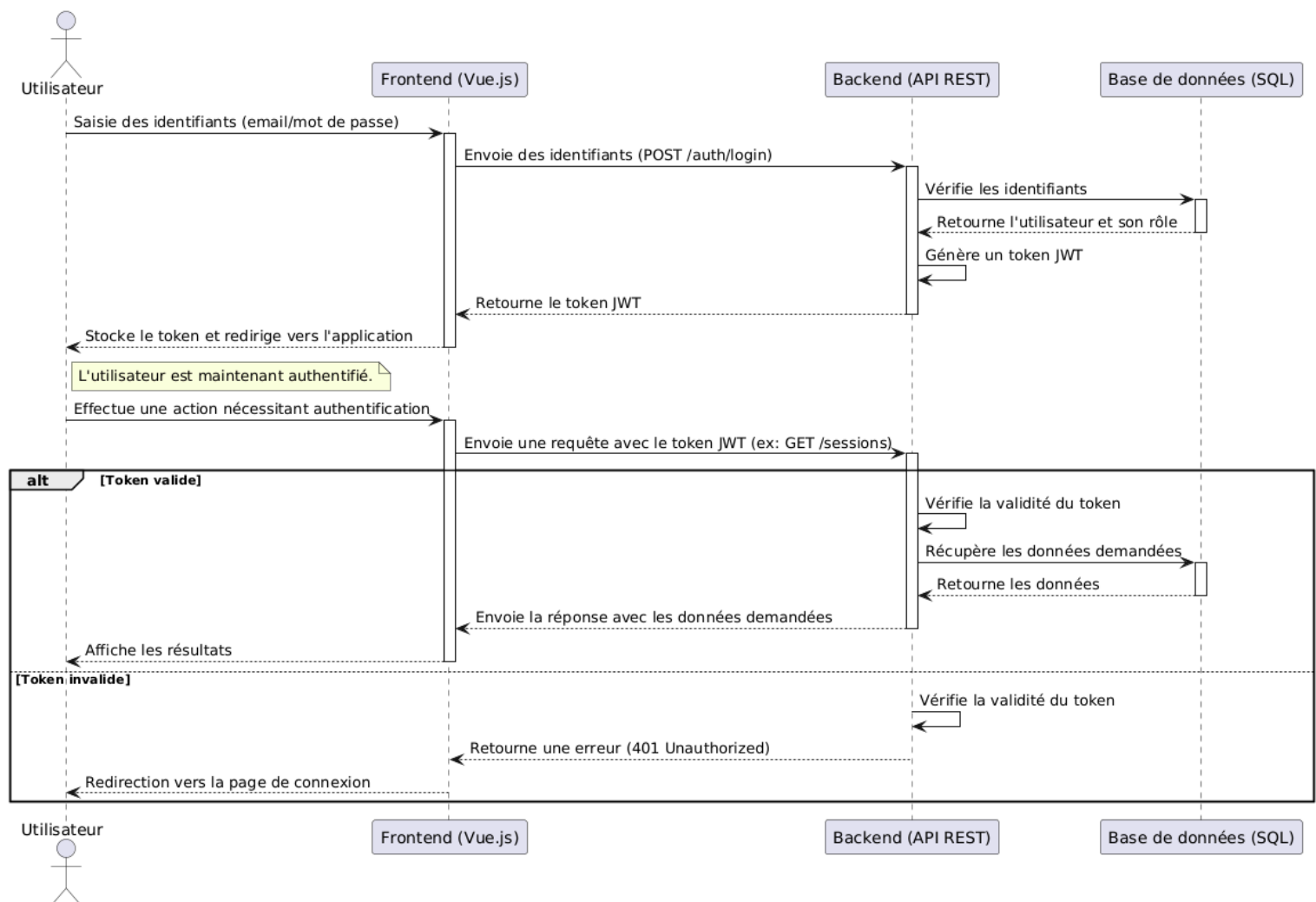


Figure 4 : Diagramme de séquence du fonctionnement de l'authentification avec JWT

Ce diagramme explique, en outre, le fonctionnement du token de sécurité, mais aussi comment se déroulent les échanges entre le front-end et le back-end.

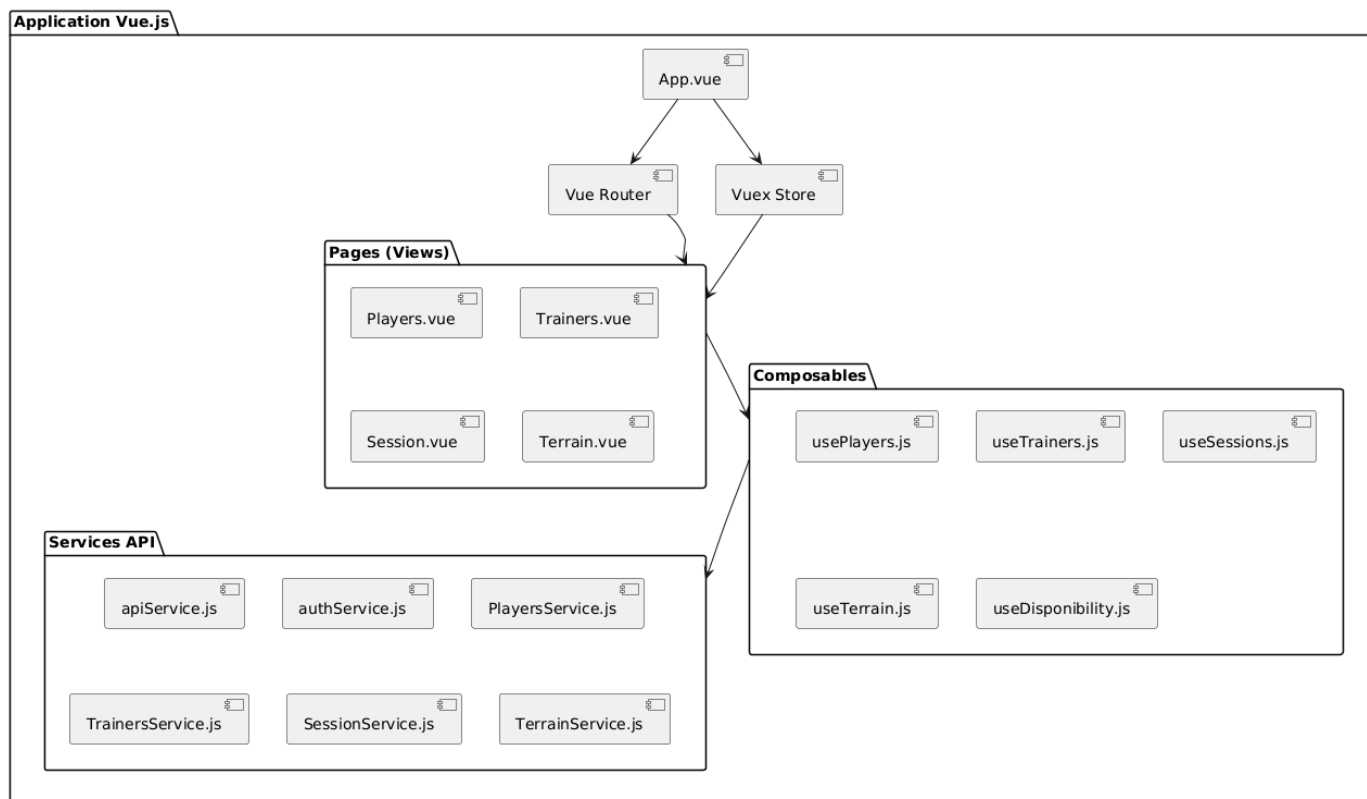


Figure 5 : Architecture simplifiée du Front-end Vue.js

App.vue est le composant principal, il inclut :

- Vue Router pour la gestion des routes.
- Vuex Store (si utilisé) pour la gestion des données globales.

Les Pages (Players.vue, Trainers.vue, etc.)

- Correspondent aux différentes vues de l'application.
- Communiquent avec les composables pour récupérer les données.

Les Composables (usePlayers.js, useSessions.js, etc.)

- Centralisent la logique métier et les appels aux services API.
- Permettent de réutiliser facilement la logique à plusieurs endroits.

Les Services API (apiService.js, PlayersService.js, etc.)

- Gèrent les requêtes HTTP pour récupérer et envoyer des données.
- Utilisent Axios et apiService.js pour une gestion propre des requêtes

Travail Réalisé

TimeFold

Timefold est un outil principalement axé sur les problèmes de planification. En effet, Timefold permet de rechercher une solution (pas forcément optimale) à notre problème de planification des joueurs et entraîneurs sur les différents créneaux des terrains de tennis. Le problème se transpose dans Timefold grâce à différentes composantes et demande une configuration du solveur spécifique à nos besoins.

1. Création du domaine

Le domaine du problème, permet de définir deux types d'éléments, les faits du problème et les entités de planification. Dans notre cas, les faits, c'est-à-dire les objets qui ne vont pas changer au cours de la résolution, sont : les terrains, les entraîneurs et les joueurs. À l'inverse, les entités de planification, c'est-à-dire les objets dont une variable doit être planifiée, sont : les sessions avec leur entraîneur et leurs joueurs. Pour assembler ces éléments, il existe un objet qui permet de regrouper les faits et les entités ensemble. Ce groupement correspond alors à une solution de planification, qui est dans notre cas l'emploi du temps.

Afin de nous assurer que le domaine est correctement mis en place nous avons intégré quelques tests sur les fonctions complexes notamment :

- la méthode permettant de calculer l'âge sportif d'un joueur ;
- les méthodes précisant si une personne (joueur ou entraîneur) est disponible pour une séance donnée ;
- les méthodes sur les différents minimaux et maximaux de valeur (préférence d'âge ou de niveau, taille de groupe, etc.).

2. Mise en place des contraintes

Les contraintes, interviennent une fois le domaine formalisé, elles permettent de fournir un score à chaque solution de planification (chaque emploi du temps). Ce score permet alors d'indiquer si une solution est plus ou moins bonne qu'une autre solution. Les contraintes se découpent en trois principaux groupes :

- **les contraintes des joueurs** vérifient qu'un joueur est disponible et n'a pas déjà de session dans la journée. De plus, des joueurs peuvent être regroupés si leurs contraintes de groupe¹ sont les mêmes et que cette même contrainte est respectée ;

¹ *contrainte de groupe*: représente le type de joueur toléré dans un groupe en vérifiant : l'âge (min-max), le niveau (min-max), la taille du groupe (min-max), la différence d'âge, la différence de niveau. De plus, elle définit la durée de la session des joueurs du groupe.

- **les contraintes de session** permettent de définir qu'un entraîneur doit être présent si des joueurs sont planifiés dessus (et inversement). Également, deux séances ne peuvent pas se chevaucher ;
- **les contraintes d'entraîneur** traitent les préférences d'âge et de niveau ainsi que la disponibilité de celui-ci. De plus, une contrainte permet de vérifier qu'un entraîneur n'a pas deux séances simultanément.

La mise en place de ces contraintes s'accompagne de justification, elles permettront dans le futur d'indiquer à l'administrateur précisément ce qu'un emploi du temps à comme problème plutôt que de simplement indiquer le score de la planification.

Il est à noter que durant la mise en place des contraintes, un problème a été rencontré lors de la validation du nombre d'heures des entraîneurs. En effet, le nombre d'heures à planifier étant très proche du nombre d'heures de travail total, Timefold rencontre des difficultés dans la planification. Pour pallier ce problème, il a été décidé que les contraintes mettant en place cette validation seront retirées et dans le futur remplacées par une aide visuelle côté client web. Cette aide prendra la forme d'une indication du nombre total d'heures effectué par l'entraîneur.

De la même manière que pour le domaine, nous avons testé le bon fonctionnement des contraintes en implémentant plusieurs tests². En moyenne, chaque contrainte possède une demi-douzaine de tests, vérifiant chaque cas possible et en essayant au maximum de penser aux configurations précises pouvant poser problème. Dans le futur, de nouveaux tests seront ajoutés pour vérifier les contraintes dans des configurations plus complexes et également tester les contraintes ensembles plutôt que de manière individuelle.

3. Mise en place du solveur et résolution

La dernière étape de la création d'un projet Timefold est la configuration du solveur. En effet, il s'agit du cœur de l'application, cette configuration permettra de préciser le déroulement des différentes étapes de la résolution.

Premièrement, les variables des entités de planification ne sont au début pas initialisées. Cette étape permettra donc de rapidement placer des variables dans les différents emplacements possibles. Il s'agit donc de l'étape de construction heuristique, ici nous n'allons pas chercher à faire le placement le plus proche de l'optimal mais simplement de prendre la première variable venue qui ne pénalise pas trop le score. Cette étape est faite deux fois, une fois pour le placement des entraîneurs et une seconde pour le placement des joueurs.

Deuxièmement, une fois les variables initialisées, l'objectif est d'améliorer le score au maximum. Pour ce faire, nous utilisons un premier algorithme de recherche locale nommé recherche par tabou. Celui-ci va conserver une liste des configurations trop pénalisantes pour

² il est à noter que certains ne fonctionnent plus dû à des modifications dans le domaine pour correspondre avec les entités Spring, cela sera corrigé dans la prochaine itération

les éviter dans le futur, cette liste est de taille variable en fonction du domaine courant. Grâce à ces tabous, nous permettons à l'algorithme d'éviter de retourner en arrière sur des mauvaises combinaisons et donc d'accélérer la découverte de meilleurs scores.

Dernièrement, il est en réalité possible d'utiliser autant d'algorithmes de recherche locale que voulu. Timefold en implémente plusieurs qui sont plus ou moins efficaces selon les cas de figure. C'est pourquoi, pour notre problème nous allons utiliser, en plus de la recherche par tabou, un algorithme dit d'acceptation tardive. Il permettra de parcourir rapidement les branches en observant un nombre restreint de nœuds qu'il comparera avec les meilleurs scores que l'algorithme a pu rencontrer jusqu'à maintenant.

Les explications des algorithmes choisis n'entrent pas dans les détails de chacun, mais fournissent les points importants qui ont amené à leurs sélections. De plus, chacun de ces algorithmes sont davantage configurables avec différentes variables (la taille de la liste tabou, la mémoire des meilleurs scores, etc.), cette configuration entraîne le besoin de mettre en place un test de performance ("benchmark") sur différentes combinaisons de variables. Timefold rend cela possible en fournissant des outils de comparaison et de lancement simultanés de plusieurs solveurs et ainsi d'obtenir des résultats chiffrés de la résolution selon différents paramètres.

Finalement, après avoir trouvé une combinaison efficace de paramètres pour notre problème et avoir correctement intégré les contraintes, il ne reste alors plus qu'à lancer le solveur. Nous obtenons finalement un placement de 85% des joueurs et entraîneurs pour le jeu de données fourni par l'administrateur du club de tennis, lors des prochaines itérations, nous créerons plusieurs ensemble de données fictifs pour obtenir un réel pourcentage d'efficacité du solveur.

Cette partie fut plus complexe qu'attendue, en effet, comprendre la manière de configurer le solveur a été difficile, beaucoup de paramètres sont à prendre en compte. Notamment, les différentes variables, la méthode pour planifier un problème avec plusieurs entités de planification ou simplement la manière d'effectuer le test de performance. Cependant, cet apprentissage a été intéressant notamment sur la découverte des différents algorithmes qu'offre Timefold.

4. Liaison avec le serveur Spring

La partie Timefold étant en grande partie terminée, il ne reste alors plus que la liaison avec le reste du projet à réaliser. Pour ce faire, il était initialement prévu de combiner les entités Spring avec le domaine de Timefold. Cependant, nous avons constaté que faire cette fusion aurait été plus complexe que prévu et surtout aurait inutilement fait perdre du temps.

En remplacement, des méthodes de conversion ont été mises en place, elles permettent actuellement de fournir à Timefold les entraîneurs, les joueurs et les terrains de tennis et inversement de fournir à la base de données les sessions générées. Il est prévu en itération 5 de

pouvoir fournir des sessions à Timefold pour que la génération reprenne à partir d'une génération précédente.

Finalement, l'accès à Timefold se fait en passant par l'API Spring grâce à quatre points d'entrée : le lancement ; le statut ; la sauvegarde ; l'arrêt. Ils permettent respectivement de lancer le solveur, de connaître le statut du solveur (arrêté ou le meilleur score actuel), de sauvegarder la meilleure planification actuellement trouvée et finalement d'arrêter le solveur.

Pour conclure, nous avons grandement sous-estimé la difficulté d'implémentation du problème. Nous pensions avoir fini l'implémentation en itération 3 pour au final consacrer une itération supplémentaire à Timefold. Cela nous a causé des retards sur le reste du projet, mais nous pensons tout de même réussir à rattraper cela, notamment en nous réorganisant comme nous le verrons plus bas dans ce document. Timefold bien que son implémentation ne demande pas une connaissance approfondie des différents algorithmes, nous a tout de même poussés à apprendre et à comprendre les différents avantages et inconvénients de ces algorithmes. Ce fut une expérience enrichissante qui nous a permis de comprendre différents aspects de la résolution des problèmes de planification.

Back-End

Le backend est constitué d'une API Rest Spring boot. Spring boot est un framework Java open source permettant de programmer des applications Spring autonomes. Spring boot est basé sur des outils de gestion de dépendance comme Maven et Gradle. Dans notre cas, on a choisi Maven. On utilise de nombreuses dépendances pour nous faire économiser un maximum de temps et pour clarifier notre code. On peut notamment citer Lombok qui nous permet de générer automatiquement les Getters, les Setters et les constructeurs lors de la compilation en utilisant simplement des annotations dans notre code java. On utilise également une dépendance pour Timefold qui vous a été expliquée précédemment. D'autres dépendances seront également expliquées par la suite. L'application est constituée de plusieurs packages, un pour chaque métier. Chaque métier est généralement lié à une table de la base de données. Lors de cette partie Back-end, je ne préciserai pas mais je vais parler pour un seul package car le fonctionnement est similaire pour tous les packages.

1. Gestion de la base de données

La base de données est donc en SQL grâce à l'utilisation de Postgres. Pour sa gestion nous avons également utilisé des dépendances maven. Il y a dans un premier temps Liquibase qui est un outil de gestion de bases de données. On peut ainsi gérer notre base de données grâce à différents scripts au format XML. Ces scripts permettent de créer, modifier et supprimer les différentes tables mais également d'insérer des données ou de modifier les contraintes des différentes tables. Lorsque la base de données est créée, les scripts se lancent automatiquement lors du lancement de l'application.

2. Lien entre Spring boot et la base de données

Pour lier Spring boot et la base de données il faut dans un premier temps connecter la base de données à l'application. Pour ce faire, nous mettons notre base de données dans un docker et nous utilisons les informations de connexion à cette base de données dans un fichier nommé "application-local.yml". Ce fichier correspond au profil local. Il y aura également par la suite un profil "intégration".

Une fois la base de données connectée au serveur, il faut pouvoir l'utiliser. Pour ce faire nous utilisons la dépendance Jakarta Persistence (JPA). Pour la gestion de la base de données nous utilisons 3 objets java : les entités, les repository et les services.

Pour commencer, l'entité est l'élément le plus important pour lier la base de données et Spring boot car c'est dans cet élément que les données sont converties. Chaque attribut de cet objet correspond aux différentes colonnes de la table. Ces objets ne contiennent aucune méthode et grâce à notre dépendance Lombok, nous mettons uniquement les annotations permettant de générer automatiquement le boilerplate. Ainsi nous avons des objets très lisibles et clairs représentant parfaitement la table. Les annotations nous permettent également de préciser à quelle table l'entité est liée ainsi que de préciser quelles sont les clés primaires, de dire à quelle colonne l'attribut fait référence et nous permet également de faire des jointures entre les différentes tables.

Nous avons ensuite le repository. C'est une interface java qui nous permet de faire les différentes requêtes à la base de données. Cet objet ne contient généralement pas beaucoup de lignes de code car les requêtes principales sont automatiques, il n'y a donc pas à écrire les méthodes permettant de récupérer tous les éléments, d'ajouter des éléments, de les modifier et de les supprimer. Il nous faut écrire uniquement les méthodes impliquant une colonne en particulier comme par exemple lorsque nous voulons récupérer un élément grâce à sa clé primaire et si on veut récupérer une liste d'éléments dont une colonne correspond à une valeur particulière.

Enfin, il y a le service. Cet objet a pour attribut un repository et permet de faire faire les requêtes. Cet objet permet de faire le lien entre le repository et la communication avec le front-end. Comme il y a un repository par table de la base de données et que chaque méthode d'insertion de données possède le même nom, "save" dans notre exemple et que plusieurs bases de données peuvent être utilisées par la suite, les services permettent de "renommer" les méthodes et donc d'avoir un code plus lisible et compréhensible.

3. Lien entre Spring boot et le front-end

Pour permettre au front-end de communiquer avec l'application Spring boot, n'ayant actuellement toujours pas encore mis notre application sur un docker, il faut lancer le docker de la base de données puis l'application grâce à notre IDE ou aux lignes de commandes pour pouvoir faire nos requêtes via le front-end. Mais pour lier le front-end et le back-end il fallait rajouter des éléments, il y a ainsi trois nouveaux objets : les controllers, les dtos et les mappers.

Le plus important est le controller. Cet objet met en place différents endpoints http pour permettre au client front-end de communiquer au serveur. Cet objet permet les différentes

opérations CRUD. C'est ici qu'on utilise les services parlés précédemment. Cet objet est le point d'entrée de notre application car lorsque le front-end fait une requête au back-end c'est ici que le serveur sait ce qu'il doit faire en fonction de l'appel qu'il reçoit et ce qu'il doit renvoyer. Chaque méthode de cet objet correspond à un endpoint. Cet objet permet de récupérer les paramètres de chemin comme lorsque nous voulons accéder à un seul élément. Cet objet permet aussi de récupérer le body de chaque requête.

Le prochain objet est donc le dto qui suit parfaitement le controller puisque c'est cet objet qui est récupéré dans le body de la requête et qui peut être envoyé comme réponse de requêtes. Le dto est notre entité au format JSON. Il ne contient aucune méthode et n'a aucune autre utilité que d'être récupéré dans le body d'une requête et d'être renvoyé dans les réponses aux requêtes.

Cela nous permet de rebondir sur le dernier objet qui est le mapper. Cet objet est une interface dans laquelle nous n'écrivons que les définitions des méthodes qui sont par la suite générées automatiquement à la compilation. Cet objet permet de convertir les entités en dto et inversement, ainsi que les listes de dtos/entités. Cet objet est utilisé dans le controller car ce dernier va appeler les services qui ont besoin d'entités pour communiquer avec la base de données.

Pour les controllers et les dtos il y a de nombreuses annotations Swagger permettant aux développeurs front-end de comprendre notre api. Swagger est une dépendance maven permettant de documenter notre api de manière standardisée. Lors du lancement de l'application, on peut accéder à swagger sur navigateur. On peut y voir le format JSON des objets retournés par les différentes requêtes des controllers ainsi que le format JSON du body que la requête attend. On y voit également les différents endpoints, une courte description de chaque méthode des controllers ainsi que les différents codes réponses (200, 201, 404, etc.). Swagger est également une documentation interactive, car il est possible de tester les différentes requêtes sur le navigateur à l'URI swagger correspondant à l'application. En mettant en place cet outil, les développeurs front-end savent directement quelle requête ils doivent faire, ils peuvent traiter chaque cas de réponse et connaître le bon format de l'objet à renvoyer ainsi que de celui à recevoir.

4. Mise en place de la sécurité

La sécurité de l'application n'est pas finie, mais la mise en place d'un token permet tout de même de la sécuriser un peu. Nous utilisons un token JWT qui est unique pour chaque utilisateur en fonction de l'heure à laquelle il est créé. Un JWT (JSON Web Token) est composé de trois parties : il y a d'abord le header contenant type de token ainsi que l'algorithme de signature utilisé, il y a ensuite le payload avec les différentes informations à transmettre puis il y a la signature qui permet l'intégrité du token. Dans notre cas on stock dans le payload l'utilisateur connecté ainsi que l'heure d'expiration du token.

Le token est nécessaire pour accéder à l'entièreté de l'application et des différents endpoints à quelques exceptions près. Il n'est pas nécessaire pour aller sur swagger. Le token n'est également pas nécessaire pour inscrire un nouveau joueur via le formulaire. Ce formulaire sera accessible aux nouveaux joueurs et n'ayant pas de compte, ils ne peuvent pas se connecter. Ce token n'est logiquement pas nécessaire pour se connecter puisqu'on ne peut logiquement pas

être connecté pour se connecter. L'entièreté des autres endpoints nécessite le token sans quoi un code erreur sera renvoyé au client lui faisant comprendre qu'il doit être authentifié pour pouvoir exécuter cette commande.

5. Tests de validation

Le back-end étant nécessaire pour le développement du front-end, l'entièreté des tests n'a pas encore été réalisée. Nous avons pour l'instant prioriser les tests concernant les objets liés à la base de données en testant les ajouts, les suppressions et les modifications d'objets dans la base de données dans différentes conditions nous permettant d'avoir un total de plus de 100 tests. Ce sont des tests unitaires grâce à JUnit. Pour les tests des controllers de la prochaine itération, nous utiliserons mockito pour pouvoir simuler le client.

Front-End

1. Affichage de la plateforme

Nous avons créé une **maquette Figma**, qui nous a été **d'une grande aide** pour **visualiser la structure de la plateforme** et éviter de perdre du temps à effectuer des modifications de structure en cours de développement. Celle-ci a été conçue de la manière suivante :



Figure 6 : Maquette Figma

Le résultat actuel de la plateforme est celui ci :

Données

Contraintes

Filtrer les données...

Entraîneurs

Nom	Prénom	Niveau Min-Max	Âge Min-Max
Brandt	Pierre	1 - 1	1 - 1
BRANDT	Pierre	14 - 19	1 - 99
HOUTMANN	Bastien	1 - 19	1 - 99
DESSOY	Pierre	1 - 15	1 - 99
HOMBOURGER	Malo	1 - 14	1 - 99
BODOT	Lucie	1 - 13	1 - 99
MONTAGNE	Nicolas	1 - 14	1 - 99
HOMBOURGER	Solal	1 - 12	1 - 18
DOYEN	Manon	1 - 12	1 - 18
MARIE	Maxence	1 - 12	1 - 18
ADLINE	Thomas	1 - 12	1 - 18

Joueurs

Nom	Prénom	Âge	Niveau
CONART	Gabriel	4 ans	0
OGBALET	Salomon	4 ans	0
RAPENNE	Léon	4 ans	0
BADJI	Maylis	5 ans	1
BLUZAT	Luke	5 ans	1
KREMPF	Marius	5 ans	3
LEGER	Céleste	5 ans	1
OKOLI	Uchechukwu	5 ans	2

Terrain 1

Terrain 2

Terrain 3

Filtrer

Lundi

20:00

21:30

Entraîneur : BRANDT Pierre

Âge : 1 - 99 ans

Niveau : 14 - 19

Joueurs :

• CHANDECLERC Aloïs

• GRENOT Martin

Supprimer

17:00

18:30

Entraîneur : BRANDT Pierre

Âge : 1 - 99 ans

Niveau : 14 - 19

Joueurs :

• GUERIN Auguste

• L'HUILIER Anatole

• NGUEMO Aïss

• VOYDEVILLE Margot

Supprimer

18:30

20:00

Entraîneur : Aucun entraîneur

Âge : N/A ans

Niveau : N/A

Joueurs :

• BRUNSTEIN HUGUENIN Noam

• DEREINE-CHARRON Luce

• MARCHAND Ilona

• SCHAACK Roman

• TARTE Zélie

• HOMBOURGER Paloma

• LEDROIT Hector

• FINANCE Sacha

Supprimer

Mardi

Mercredi

20:30

22:00

Entraîneur : HOUTMANN Bastien

Âge : 1 - 99 ans

Niveau : 1 - 19

Joueurs :

• BELLINA Léa

• BRUSCHI Julien

• DIAS Carlos

• KIRCHER Nicolas

• LAUMONT Olivier

• PIETTE Cléa

• TOUTAIN Anne-Cécile

• WEBERT David

Supprimer

Entraîneur : HOUTMANN Bastien

Joueurs :

Lancer

Envoyer les modifications

Placement réalisé à 95%

Envoyer l'emploi du temps

Figure 7 : Page d'admin

Données

Contraintes

Terrains

Terrain 1

Jour	Ouverture	Fermeture	Actions
Lundi	17:00	22:30	
Mercredi	13:00	22:30	
Jeudi	17:00	22:30	
Vendredi	17:00	22:30	
Samedi	09:00	14:00	

Terrain 2

Jour	Ouverture	Fermeture	Actions
Lundi	17:00	22:30	
Mardi	17:00	22:30	
Mercredi	09:00	22:30	
Jeudi	17:00	22:30	
Vendredi	17:00	22:30	
Samedi	09:00	16:00	

Terrain 3

Séances

3 à 4 ans

5 à 7 ans

Données

Contraintes

Importer vos données

Consulter les inscrits

Importer Données et Contraintes - format CSV

Télécharger vos données

Planning - format XLS

Données et contraintes - format CSV

Nouvelle année

Envoyer le mail de réinscription

Supprimer l'ensemble des joueurs

Inscriptions en attente

Achraf Rao

Tom Ketzinger

Pedron Matheo

Micheli Thomas

Figure 8 : Onglets Données / Contraintes / Paramètres

19

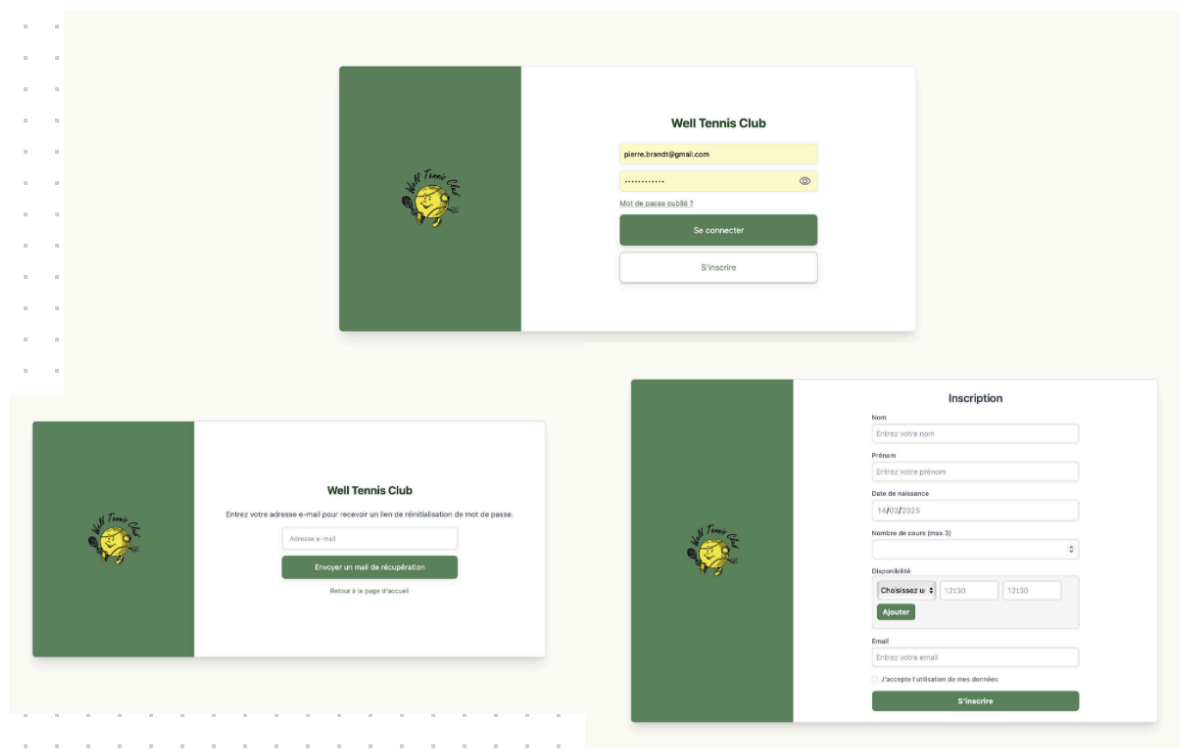


Figure 9 : Page de connexion / Inscription / Mot de passe oublié

2. Fonctionnalités développées

Au cours du projet, nous avons mis en place plusieurs fonctionnalités essentielles pour assurer une gestion efficace des emplois du temps et améliorer l'expérience utilisateur. Voici un récapitulatif des principales fonctionnalités développées :

2.1 Affichage des données Joueurs / Entraîneurs

Lorsque l'administrateur se connecte, il peut retrouver, sur le côté gauche de la page, les données de tous les joueurs, incluant leur nom, prénom, âge et niveau. Il a également accès aux informations sur les entraîneurs, telles que leur nom, prénom, niveau minimum et maximum qu'ils peuvent entraîner, ainsi que l'âge minimum et maximum des joueurs qu'ils peuvent encadrer.

2.2 Affichage des séances

Sur la plateforme, les séances sont affichées à droite, organisées en fonction des terrains. Chaque carte de séance comprend l'heure de début et de fin, les informations générales de la séance, le nom de l'entraîneur, l'âge et le niveau du groupe. La liste des joueurs est divisée en deux colonnes de quatre joueurs, et un bouton permet de supprimer la séance si nécessaire.



17:00 18:30	Entraîneur : BRANDT Pierre Âge : 1 - 99 ans Niveau : 14 - 19	Joueurs : <ul style="list-style-type: none"> • GUERIN Auguste • L'HUILLIER Anatole • NGUEMO Ava • VOYDEVILLE Margot 	 Supprimer
18:30 20:00	Entraîneur : Aucun entraîneur Âge : N/A ans Niveau : N/A	Joueurs : <ul style="list-style-type: none"> • BRUNSTEIN HUGUENIN Noam • DEREINE-CHARRON Luce • MARCHAND Ilona • SCHAACK Roman • TARTE Zélie • HOMBOURGER Paloma • LEDROIT Hector • FINANCE Sacha 	 Supprimer

Figure 10 : Exemple de séance

2.3 Afficher les contraintes des terrains / séances

Les contraintes sont affichées dans l'onglet gauche de la plateforme. L'administrateur a accès aux contraintes des terrains, notamment les jours d'ouverture et les horaires de chaque terrain. Prochainement, il pourra modifier ces informations directement afin qu'elles soient prises en compte lors de la génération de l'emploi du temps.

2.4 Importer / Exporter des données

Dans l'onglet Paramètres, l'administrateur a accès à plusieurs boutons, dont deux permettent d'importer et d'exporter les données. Pour importer des données, il doit cliquer sur le bouton correspondant et sélectionner un fichier au format XLSX. Ce fichier doit contenir plusieurs colonnes avec des noms spécifiques, que la plateforme analysera afin d'extraire et de stocker les informations dans la base de données.

Concernant l'export des données, l'administrateur pourra récupérer un fichier XLSX contenant les données des joueurs, entraîneurs, contraintes des terrains et séances. Ces informations seront organisées en différentes feuilles au sein du fichier Excel pour une meilleure lisibilité.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Nom	Prénom	Date de nais.	Âge	Email	Niveau	Cours par se	Disponibilités					
2	Raouf	Achraf	03/09/2004	21	achraf.raoufi	6	1	17:30-19:00		Mardi	Mercredi		
3	Micheli	Thomas	13/04/2004	20	micheli.thom	5	1	17:30-19:01				17:30-19:00, 20:30-21:00	09:00-12:00
4	Ketzinger	Tom	15/05/2004	20	ketzinger.to	6	2	17:30-19:02					9-12
5	Pedron	Matheo	06/10/2004	21	pedron.math	7	3	17:30-19:03					9-13
6													9-14

Figure 11 : Exemple de fichier à importer

	A	B	C	D	E	F	G	H	I	J	K	L	M
1								Disponibilités					
2	Nom	Prénom	Date de nais.	Âge	Email	Niveau	Cours par se	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi
3	Raouf	Achraf	2004-09-03	21	achraf.raoufi	6	1	17:30 - 19:00		09:00 - 12:00		17:30 - 19:00	09:00 - 12:00
4	Micheli	Thomas	2004-04-13	21	micheli.thom	5	1	17:30 - 19:01	17:30 - 19:02	09:00 - 12:00			09:00 - 12:00
5	Ketzinger	Tom	2004-05-15	21	default_Ketz	6	2	17:30 - 19:02		09:00 - 13:00	09:00 - 14:00	09:00 - 15:00	09:00 - 13:00
6	Pedron	Matheo	2004-10-06	21	pedron.math	7	3	17:30 - 19:03		09:00 - 14:00			09:00 - 14:00

	A	B	C	D	E	F	G	H	I	J	K
1	Nom	Prénom	Niveau Min	Niveau Max	Âge Min	Âge Max	Minutes Heb	Minutes Heb	Email	Temps partie	Admin
2	Brandt	Pierre	1	1	1	1	1	1	pierre.brandt	Oui	Oui
3											
4											
5											
6											

Figure 12 : Exemple de fichier exporté

2.5 Page d'inscription

Mise en place de la page d'inscription pour les joueurs. L'endroit de sa mise en place n'est pas encore fixé (pour l'instant présent sur l'écran d'authentification). Mais ce dernier va permettre aux nouveaux joueurs de renseigner les différentes informations personnelles pour pouvoir s'inscrire définitivement au well tennis club.

2.6 Afficher / Valider les inscrits

L'administrateur peut afficher la liste des inscrits en cliquant sur le bouton "consulter les inscrits" dans l'onglet Paramètres. Une fois la liste affichée, il lui suffit de sélectionner un inscrit pour consulter ses informations et définir son niveau avant de le valider. Une fois validé, l'inscrit est stocké dans la base de données et sera affiché dans la liste des joueurs de l'onglet Données.



Figure 13 : Affichage des inscrits

Détails de l'inscription

Nom : Achraf Rao

Email : achraf@gmail.com

Date de naissance : 2004-09-03

Nombre de cours : 1

Niveau :

Disponibilités :

- Mardi : 12:00 - 14:00

Fermer Valider

Figure 14 : Détails de l'inscription

2.7 Affichage Mobile

Afin de permettre aux utilisateurs d'accéder à la plateforme depuis un mobile, nous avons rendu celle-ci responsive. Cependant, nous avons réduit certaines fonctionnalités en mode mobile pour garantir une expérience plus fluide et adaptée à l'écran réduit.

L'administrateur aura uniquement accès à l'emploi du temps et à la visualisation des données et des contraintes. Dans l'onglet Données, il pourra uniquement effectuer une recherche, mais ne pourra pas ajouter ou modifier des joueurs ou des entraîneurs, contrairement à l'affichage sur ordinateur. Dans l'onglet Contraintes, il pourra uniquement consulter les contraintes des terrains et des séances, sans possibilité de modification.

Concernant l'affichage des sessions, nous l'avons simplifié tout en conservant les informations essentielles. La carte de la session affiche les noms des joueurs, le nom de l'entraîneur et l'heure de la séance. En appuyant sur l'heure, l'affichage de l'heure est remplacé par le niveau des joueurs de la session et l'âge du groupe.

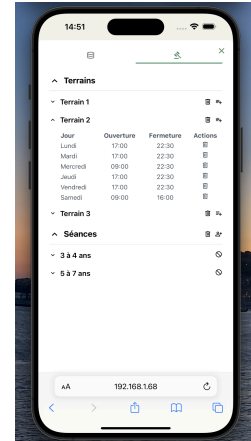
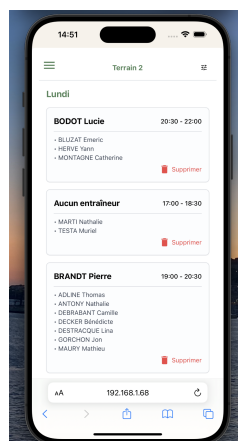
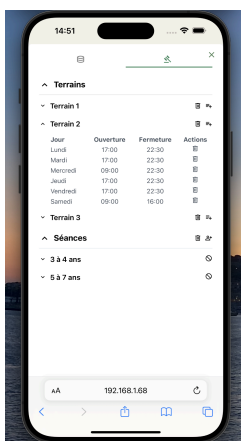


Figure 15 : Affichage Mobile

2.8 Filtrage des données

Il est possible, grâce à une barre de recherche de filtrer les joueurs et les entraîneurs par leurs noms, pour plus vite les retrouver dans les nombreuses données. Il est prévu d'augmenter les filtres possibles.

2.9 Possibilité de modifier les informations des joueurs et entraîneurs

Il est possible à l'administrateur de modifier l'entièreté des données personnelles des entraîneurs et des joueurs. Mais aussi de rajouter manuellement des joueurs, et supprimer joueurs/entraîneurs

2.10 Connexion sécurisé

A partir de l'onglet de connexion, les entraîneurs peuvent avoir accès à leur compte coach, qui leur permet de consulter leur emploi du temps. Ils possèdent des identifiants, préalablement créés par un administrateur. Pour leur mot de passe, un mail de configuration de mot de passe leur sera envoyé.

2.11 Sécurisation des url

Il est impossible d'accéder au site internet s'il n'y a pas eu de connexion préalable. L'utilisation de token jwt permet de sécuriser les pages et les requêtes envoyées. En effet, à part la demande de connexion, aucun utilisateur ne peut envoyer de requête à la base de données s'il ne possède pas le bon token d'authentification.

3. Tests de validation

Nous avons effectué plusieurs tests afin de vérifier que les fonctionnalités implémentées fonctionnent correctement. Concernant l'affichage des données, nous avons vérifié que les données affichées correspondent bien aux données stockées dans la base de données. Nous avons également vérifié que la mise à jour des données s'effectue correctement dans la base de données. Swagger et Postman sont des outils qui nous ont bien aidé.

Pour les fonctionnalités d'import/export, les mêmes tests ont été effectués afin de vérifier que toutes les données importées sont bien stockées et affichées sur la plateforme.

Concernant l’affichage mobile, celui-ci a été testé grâce au navigateur Firefox et au simulateur Apple dans le logiciel Xcode. Nous avons testé plusieurs tailles d’écran afin de nous assurer que la plateforme est bien responsive sur mobile.

4. Difficultés rencontrées

Quelques difficultés pour régler l’affichage sur ordinateur, notamment des soucis lorsque la taille de l’écran est plus grande, ainsi que pour l’affichage des sessions.

Un des défis majeurs rencontrés concernait la compréhension et l’implémentation des tokens JWT (JSON Web Token). L’apprentissage du fonctionnement des JWT a pris du temps, notamment leur rôle dans l’authentification et la sécurisation des requêtes API. Un problème initial a également été identifié avec l’intégration des tokens, car le bon type de token n’avait pas été mis en place dès le début. Le token choisi manquait d’informations essentielles, ce qui a entraîné des erreurs lors de l’authentification et des requêtes.

Un autre point critique concernait les nombreux éléments à ne pas oublier lors de la mise en place des JWT :

- La gestion du stockage du token (sessionStorage, localStorage, cookies) et ses implications en termes de sécurité.

- Le rafraîchissement du token (token expiration et refresh token).

- L’inclusion correcte du token dans les en-têtes des requêtes API pour garantir l’accès sécurisé aux ressources protégées.

- Les vérifications côté serveur pour s’assurer que le token est valide et bien formé.

Cette phase d’apprentissage a nécessité un investissement important, mais elle a permis d’améliorer les bonnes pratiques de sécurisation et d’optimiser significativement l’implémentation du système d’authentification. Avec le recul, ces difficultés ont été des défis formateurs qui ont renforcé la compréhension des mécanismes de sécurité web et de gestion des accès utilisateurs.

Un autre problème qui a fait perdre du temps concernait la gestion des CORS. Pensant que la configuration était correcte côté backend, de nombreux ajustements ont été faits en boucle sur le front, alors que le problème venait en réalité d’une mauvaise configuration du serveur. Cette erreur a entraîné une perte de temps considérable avant de pouvoir identifier la source réelle du problème et appliquer la solution appropriée. Cette phase d’apprentissage a nécessité un investissement important, mais elle a permis d’améliorer les bonnes pratiques et d’optimiser significativement l’implémentation du système d’authentification.

Planning

Itération 1 : Premières mises en place

Lors de cette première itération, nous avons travaillé sur la mise en place des éléments fondamentaux du projet :

- Affichage de la **page administrateur** en utilisant uniquement des **données statiques**, sans interaction.
- Création du **serveur Spring Boot** et configuration des accès extérieurs aux données.
- Vérification de la **structure de la base de données**, implémentation en **Java** avec **PostgreSQL**, puis transfert des données.
- Découverte de **TimeFold** et compréhension de son fonctionnement.
- Tests de récupération des données pour s'assurer de leur intégrité.
- Tests d'accès au serveur via **HTTPS** pour garantir une connexion sécurisée.
- Connexion du **serveur Spring Boot** avec **PostgreSQL** pour assurer l'interaction entre la base de données et l'application.

Itération 2 : Intégration des fonctionnalités et améliorations

Au cours de cette deuxième itération, nous avons progressé sur l'intégration des fonctionnalités et la structuration du projet :

- **Préparation des requêtes API** pour faciliter la communication entre le serveur et l'interface.
- **Tests approfondis de TimeFold** pour vérifier son bon fonctionnement et son intégration future.
- **Mise en place de TimeFold dans le projet** et début de l'implémentation des contraintes liées à la génération des emplois du temps.
- **Migration du front-end vers Vue.js**, remplaçant l'ancienne architecture **HTML**.
- **Ajout de la fonctionnalité de modification des données**, avec mise à jour automatique dans la base de données.
- **Correction d'un retard** sur la création du **serveur Spring Boot** et la configuration des accès extérieurs aux données.
- **Amélioration de l'interface web en Vue.js** : correction de bugs, amélioration de l'affichage et ajout de nouvelles pages (formulaire d'inscription, réinitialisation de mot de passe, onglet d'importation des données).
- **Début des liaisons entre l'API et l'interface web** pour afficher et traiter les données dynamiquement.
- **Tests de connexion et de restriction d'accès** aux différentes pages pour assurer la sécurité des utilisateurs.

Itération 3 : Consolidation et optimisation des fonctionnalités

Lors de cette troisième itération, nous avons avancé sur plusieurs aspects techniques et fonctionnels du projet :

- **Tests approfondis de TimeFold** pour assurer son bon fonctionnement et son intégration complète.
- **Liaison de l'API avec l'emploi du temps** de l'interface web pour afficher dynamiquement les plannings.
- **Intégration des contraintes liées aux terrains** dans l'API afin de garantir une planification réaliste.
- **Création de fichiers XML de données** permettant de stocker et manipuler plusieurs ensembles d'informations en base de données.
- **Mise en place des outils d'import et d'export des données** aux formats CSV et XLSX pour faciliter la gestion et la récupération des informations.
- **Correction de bugs liés à la duplication des indices** dans la base de données lors de l'ajout de nouvelles données.
- **Vérification de la transmission et de la sauvegarde des données**, assurant leur intégrité et leur cohérence.
- **Modification de la structure de la base de données**, en passant de **ID simple** à **UUID**, pour renforcer l'unicité des enregistrements.
- **Finalisation du formulaire d'inscription des joueurs.**
- **Mise en place d'une gestion dynamique des mises à jour de l'affichage parent**, afin de refléter en temps réel les modifications effectuées dans le formulaire d'inscription.
- **Affichage des entraîneurs à partir de la base de données**, avec possibilité d'**ajout et de modification**.
- **Ajout d'un filtrage basique des joueurs par nom et prénom** pour faciliter la recherche.
- **Tentative de mise en place d'un token unique** pour renforcer la sécurité des connexions et des requêtes API.
- **Gestion des UUID lors de la création et de la modification des différents acteurs**, permettant une meilleure communication avec l'API.
- **Correction des problèmes d'affichage** liés à l'ouverture du formulaire d'inscription.

Itération 4 : Finalisation et sécurisation du projet

Lors de cette quatrième itération, nous avons travaillé sur l'amélioration des fonctionnalités, l'intégration des dernières contraintes et la sécurisation de l'application :

- **Mise en place de la page d'inscription**, avec enregistrement des utilisateurs dans la base de données.

- **Vérification des ajouts et modifications** pour s'assurer qu'ils sont bien pris en compte dans la base de données.
- **Résolution du problème de CORS**, encore en cours de configuration.
- **Connexion entre TimeFold et le serveur Spring**, permettant une communication fluide entre les deux.
- **Affichage des inscriptions en attente de validation** pour permettre aux administrateurs de gérer les nouveaux utilisateurs.
- **Correction de la redirection de l'erreur 401** pour garantir une bonne gestion des échecs de token.
- **Fusion des entités Spring et TimeFold** afin de centraliser les traitements et optimiser le fonctionnement du système.
- **Tests et finalisation des contraintes TimeFold**, assurant une gestion efficace des règles de planification.
- **Renommage de la table "Coach" en "Trainer"** pour une meilleure cohérence dans la base de données.
- **Amélioration de la gestion des erreurs de token**, en évitant d'afficher des messages trop détaillés pour renforcer la sécurité.
- **Ajout de Spring Security** et connexion avec la page d'authentification de l'interface web.
- **Création d'une table "Contrainte"** pour centraliser la gestion des règles de planification.
- **Amélioration du traitement des listes vides**, renvoyant désormais une liste vide au lieu d'une exception.
- **Modifications de l'import des données** pour mieux gérer les formats et optimiser l'intégration des informations.
- **Amélioration de l'affichage mobile**, pour garantir une meilleure expérience utilisateur sur tous les supports.

Répartition du travail

Durant ces quatre itérations, nous avons décidé d'assigner des rôles spécifiques à chaque membre de l'équipe afin d'optimiser notre efficacité et d'assurer une bonne progression du projet :

- **Gestion de TimeFold** : Une personne était dédiée exclusivement à l'intégration et à la gestion de **TimeFold**, en s'occupant de l'implémentation des contraintes, des tests et de son intégration avec le serveur Spring.
- **Gestion de la base de données** : Une personne a été chargée de la **conception, de la mise en place et de la gestion** de la base de données en **PostgreSQL**, ainsi que des migrations et de la gestion des UUID.
- **Développement du Back-End** : Une personne s'est concentrée sur le **développement de l'application Spring boot**. La personne chargée du développement du back-end est la même personne qui a fait la gestion de la base de données.
- **Développement du Front-End** : Deux personnes se sont concentrées sur le **développement de l'interface utilisateur en Vue.js**, en travaillant sur l'affichage, la connexion avec l'API, la gestion des formulaires et l'adaptation aux différents écrans (desktop et mobile).

Réflexion sur l'organisation et ajustements nécessaires

À la fin de l'**itération 4**, nous avons constaté que notre organisation initiale n'était pas totalement **optimale**. Le manque de communication entre nous sur le travail dans le **front-end** et la gestion de la **base de données** a entraîné plusieurs problèmes, notamment des erreurs dans la gestion des requêtes.

Par exemple, lors de l'inscription d'un utilisateur, le **front-end envoyait les données à la base de données**, mais la requête était rejetée car la **base exigeait un token**. Or, à ce stade, **l'utilisateur est simplement en visite sur le site, il n'a pas de token et n'a pas besoin d'en générer un** pour soumettre son inscription. Ainsi, lorsque le formulaire était complété et que la requête **POST** était envoyée, la **base de données refusait la requête**, car **seul l'administrateur est autorisé à ajouter des joueurs**.

Face à ces difficultés, nous avons réalisé qu'une **meilleure répartition des tâches** aurait été plus efficace :

- **Deux personnes pour la gestion de la base de données**, afin d'anticiper les conflits entre le front et la BD et de mieux structurer les règles d'accès.
- **Une personne dédiée à la gestion de TimeFold**, en se concentrant sur l'optimisation et l'intégration des contraintes.

- **Une personne chargée du développement visuel**, pour garantir une interface fluide et ergonomique.

Par la suite, **une personne aurait rejoint l'équipe front-end** afin de **poursuivre le travail sur cette partie**. Cette réorganisation aurait permis **d'optimiser dès le début du projet la gestion des échanges de données** entre le front-end et le back-end, évitant ainsi certaines pertes de temps liées aux ajustements des requêtes et aux modifications dans la base de données lors de l'itération 4.

Pour les prochaines itérations

Pour les prochaines itérations, **la personne en charge de TimeFold**, étant en phase de finalisation, rejoindra probablement **l'équipe back-end** afin de **renforcer cette partie**. Une fois le **back-end totalement finalisé**, nous nous concentrerons **tous sur le front-end** pour **l'améliorer et le finaliser**, en optimisant l'interface et l'expérience utilisateur. Cette approche nous permettra d'assurer une **intégration fluide** entre toutes les composantes du projet et de garantir un produit final abouti.

Les éléments originaux

Mathéo

Ma participation au projet dont je suis le plus satisfait est Timefold, il s'agit de l'élément où j'ai consacré le plus de temps et qui permettra, je l'espère, de faire gagner un grand nombre d'heures à l'administrateur du site. Savoir que le temps passé à mettre en place Timefold ne sera pas inutile me rend particulièrement fier. Il s'agit de la première création de ce type que j'ai pu réaliser, et elle restera dans mes créations dont je suis fier que ce soit de par son utilité, mais également son implémentation.

De plus, bien qu'une grande partie de mon temps ait été consacrée à Timefold, j'ai eu l'occasion de participer à d'autres parties du projet. Notamment, la création de la maquette de l'interface cliente, il s'agit d'un élément dont je suis particulièrement fier et qui a été totalement respecté lors de l'implémentation au cours des itérations.

Thomas

Je pense pour moi que l'élément dont je suis le plus fier est le token JWT mis en place dans le back-end. Ayant déjà utilisé Spring boot lors de mon stage de deuxième année, je connaissais déjà le framework, mais je n'avais pas mis en place de token de sécurité comme ici. Je suis fière de cette fonctionnalité, car j'ai tout appris de moi-même en cherchant sur internet et en demandant des conseils à Monsieur Perrin. J'ai passé une itération entière dessus avec une journée entière de documentation.

Cette itération était un peu frustrante de mon point de vue. J'avais l'impression de ne pas avancer dans le projet, je voyais mes collègues avancer dans leurs différentes fonctionnalités tandis que moi, je me renseignais et j'apprenais le fonctionnement du token. Mais cette frustration n'a fait qu'augmenter ma joie lorsque j'ai réussi son implémentation et que mes différents essais sur Postman étaient concluants.

J'ai donc logiquement choisi cet élément, car c'est celui qui m'a le plus appris et c'est également celui qui m'a rendu le plus heureux lorsque je l'ai fini.

Achraf

Concernant ma contribution, l'élément que j'ai développé et dont je suis fier est l'affichage mobile. Celui-ci est entièrement responsive et correspond parfaitement à ce que j'avais imaginé. Cette fonctionnalité n'était pas initialement demandée, mais en y réfléchissant, j'ai réalisé qu'aujourd'hui, la majorité des utilisateurs naviguent sur mobile. Ainsi, offrir la possibilité d'accéder à la plateforme directement depuis un smartphone, sans avoir besoin d'un ordinateur, apporte un véritable confort d'utilisation.

Un autre élément dont je suis particulièrement fier est l’affichage des inscrits et la possibilité de les valider directement via l’interface. De manière générale, je suis très satisfait du rendu visuel de la plateforme, qui offre une expérience fluide et agréable pour les utilisateurs.

Tom

Gestion des Disponibilités des Joueurs et Entraîneurs

Concernant ma contribution, l’élément que j’ai développé et dont je suis particulièrement fier est la gestion des disponibilités des joueurs et des entraîneurs. Cette fonctionnalité permet d'optimiser l'organisation des entraînements et de minimiser les conflits d'horaires, garantissant ainsi une meilleure planification. L'interface utilisateur que j’ai mise en place est simple et intuitive. J’ai intégré un formulaire interactif permettant aux utilisateurs d'ajouter et de modifier leurs créneaux horaires tout en visualisant en temps réel les disponibilités existantes. De plus, j’ai développé un système de validation automatique qui garantit la cohérence des créneaux saisis et réduit considérablement les erreurs.

Ce système constitue un élément important du projet. Son architecture modulaire permet une évolutivité et une maintenance simplifiée, tout en offrant une expérience fluide et intuitive pour la gestion des disponibilités.

Objectifs à atteindre

Afin de finaliser le projet dans les meilleures conditions, nous avons décidé d'ajuster nos prévisions pour les prochaines itérations. Globalement, nous allons consacrer les deux prochaines itérations à la finalisation des **fonctionnalités de base**, ainsi qu'à l'intégration des **fonctionnalités bonus** que nous avons proposé de développer au début du projet.

La **dernière itération**, quant à elle, sera entièrement dédiée à **l'installation de notre plateforme chez le client (serveur, affichage en ligne)** et à la **création de documents explicatifs**, afin d'accompagner le client dans la prise en main du système et d'assurer une transition fluide vers son utilisation.

Suite à l'ajustement de nos prévisions, la planification des prochaines itérations sera organisée de la manière suivante :

Itération 5 : Optimisation des fonctionnalités et corrections

L'objectif de cette itération est d'améliorer l'expérience utilisateur en intégrant de nouvelles fonctionnalités et en corrigeant les derniers problèmes d'affichage.

- Implémentation du **reset du mot de passe par email**.
- Ajout d'un système de **Drag & Drop** pour faciliter l'organisation des sessions.
- **Lancement de TimeFold** pour générer automatiquement les plannings.
- **Modification et sauvegarde des sessions** directement depuis l'interface du site.
- **Correction des problèmes visuels** pour améliorer l'ergonomie.
- Affichage et gestion des **contraintes de séance**.
- **Modification des contraintes** pour permettre une personnalisation avancée.
- Liaison des **modifications front-end avec TimeFold** pour assurer une mise à jour en temps réel des données.
- **Création d'un affichage spécifique pour les entraîneurs**, afin qu'ils puissent consulter leurs sessions facilement.

Itération 6 : Finalisation des fonctionnalités et automatisation

Cette itération vise à automatiser certaines tâches et à ajouter des outils facilitant la gestion des plannings.

- Mise en place d'un **outil de passage à une nouvelle année**, permettant de réinitialiser et préparer les données pour une nouvelle saison.
- **Envoi automatique des plannings par email** aux joueurs et entraîneurs.
- **Optimisation des requêtes** pour améliorer les performances et récupérer les données plus efficacement.

- **Ajout de filtres avancés** pour trier et rechercher les données plus facilement (emplois du temps, sessions, joueurs, etc.).
- Intégration d'une fonctionnalité permettant aux entraîneurs de **signaler leurs absences** via l'interface.
- Finalisation du **responsive design** pour assurer une expérience optimale sur mobile et tablette.
- Mise en place d'un **affichage PDF**, permettant de générer et imprimer les plannings.

Itération 7 : Déploiement et documentation

L'ultime itération sera consacrée à la mise en production du projet et à la documentation.

- **Création des dockers d'installation**, pour simplifier le déploiement sur différentes machines.
- **Mise en place du projet sur le serveur de M. Borne**, pour assurer son hébergement et son accessibilité.
- **Rédaction du document final**, récapitulant l'ensemble du projet, de sa conception à sa mise en production.

À l'issue de ces dernières itérations, nous espérons avoir finalisé l'ensemble des fonctionnalités demandées par le client, ainsi que les fonctionnalités bonus que nous avons proposées. **L'objectif** est de livrer une plateforme complète, fonctionnelle et optimisée, répondant aux besoins du Well Tennis Club tout en offrant une expérience utilisateur fluide et intuitive.

Conclusion

À la fin de l'écriture de ce rapport, nous pouvons considérer que le projet est en bonne voie de réussite. En effet, nous avons, dans l'ensemble, suivi notre étude préalable, progressé dans les laps de temps imposés et fourni un livrable correct à la fin de l'itération 4.

La principale complexité du projet résidait dans l'algorithme de planification de l'emploi du temps, et nous pouvons affirmer que nous en sommes à la fin de son implémentation. La demande principale du client est ainsi respectée et fonctionnelle.

Cependant, malgré l'application qui gravite autour de l'algorithme et qui est déjà très bien avancée, nous ne devons pas nous relâcher. Il est essentiel de poursuivre sur notre bonne dynamique, car il reste encore de nombreuses tâches à mettre en place, finaliser ou même optimiser.