



UNIVERSITÉ
DE LORRAINE

IUT 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 projet
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 le projet tutoré que nous avons réalisé 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 ont permis d'acquérir les compétences essentielles au développement de ce projet et à notre progression académique.

TABLE DES MATIÈRES

Introduction générale.....	7
Organisations dans le groupe et répartition du travail.....	8
Itération 1 : Premières mises en place.....	10
Itération 2 : Intégration des fonctionnalités et améliorations.....	10
Itération 3 : Consolidation et optimisation des fonctionnalités.....	11
Itération 4 : Finalisation et sécurisation du projet.....	12
Itération 5 : Optimisation des fonctionnalités et corrections.....	13
Itération 6 : Finalisation des fonctionnalités et automatisation.....	14
Itération 7 : Déploiement et documentation.....	14
Analyse.....	15
Travail Réalisé.....	20
TimeFold.....	20
1. Création du domaine.....	20
2. Mise en place des contraintes.....	20
3. Mise en place du solveur et résolution.....	21
4. Liaison avec le serveur Spring.....	22
Back-End.....	23
1. Gestion de la base de donnés.....	23
2. Lien entre Spring boot et la base de données.....	24
3. Lien entre Spring boot et le front-end.....	25
4. Mise en place de la sécurité.....	28
5. Requête utilisable.....	29
6. Tests de validation.....	32
Front-End.....	33
1. Documentation à destination du client.....	33
1.1 Description des vidéos.....	33
1.2 Conseils d'utilisation.....	35
2. Affichage de la plateforme.....	36
3. Fonctionnalités développées.....	38
3.1 Génération de l'emploi du temps.....	38
3.2 Affichage des données Joueurs / Entraîneurs.....	40
3.3 Affichage des séances.....	41
3.4 Créer une session.....	42
3.5 Filtrer les sessions.....	43
3.6 Afficher les contraintes des terrains.....	44
3.7 Afficher les contraintes des sessions.....	44
3.8 Importer / Exporter des données.....	45
3.9 Télécharger l'emploi du temps en pdf.....	45
3.10 Télécharger l'emploi du temps en xlsx.....	46

3.11 Envoi de l'emploi du temps a tout les joueurs.....	46
3.12 Supprimer tous les joueurs.....	47
3.13 Page d'inscription.....	47
3.14 Afficher / Valider les inscrits.....	48
3.15 Affichage Mobile.....	49
3.16 Filtrage des données.....	51
3.17 Possibilité de modifier les informations des joueurs et entraîneurs.....	51
3.18 Connexion sécurisé.....	53
3.19 Sécurisation des url.....	54
3.20 Possibilité de personnalisation des sessions grâce au Drag and Drop....	54
4. Tests de validation.....	55
5. Difficultés rencontrées.....	55
Déploiement sur le serveur.....	57
Conclusion.....	58

Introduction générale

Ce document constitue notre **rapport de fin de projet** 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, 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.

Organisations dans le groupe et répartition du travail

Durant toute la durée du projet, 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 exclusivement dédiée à l'intégration et à la gestion de **TimeFold**, en prenant en charge l'implémentation des contraintes, les tests ainsi que 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 **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 avec Spring Boot**. Il s'agit de la même personne en charge de la gestion de la base de données.
- **Développement du Front-End** : deux personnes se sont consacrées au **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 formats d'écran (desktop et mobile).

Ceci était l'organisation initiale de notre projet ; cependant, elle a naturellement évolué au cours des six mois de travail.

À la fin de l'itération 4, nous avons constaté que notre organisation initiale n'était pas totalement optimale. Un manque de communication, notamment concernant le développement du front-end et la gestion de la base de données, a entraîné plusieurs problèmes, dont des erreurs dans le traitement des requêtes.

Par exemple, lors de l'inscription d'un utilisateur, le front-end transmettait les données à la base de données, mais la requête était rejetée car cette dernière exigeait un token. Or, à ce stade, l'utilisateur n'était qu'un simple visiteur du site : il ne possédait pas de token et n'avait pas à en générer un pour soumettre son inscription. Ainsi, lorsque le formulaire était complété et que la requête POST était envoyée, la base refusait l'opération, car seul un administrateur avait les droits nécessaires pour ajouter des joueurs.

Ce cas illustre les conséquences de certaines lacunes de communication, qui nous ont fait perdre un temps précieux.

Face à ces difficultés, nous avons procédé à une réorganisation de la répartition des tâches :

- **Deux personnes** ont été affectées à la gestion de la base de données, afin d'anticiper les conflits entre le front-end et la base, et de mieux structurer les règles d'accès.
- **Une personne** est restée dédiée à la gestion de TimeFold, en se concentrant sur l'optimisation des performances et l'intégration des contraintes.
- **Une personne** s'est chargée du développement visuel, dans le but de garantir une interface fluide, intuitive et ergonomique.

Puis, à un moment du projet, le développement de la partie TimeFold s'est terminé. La personne qui y était initialement affectée a donc commencé à aider sur le back-end. Nous sommes ainsi complètement passés à deux personnes sur le front-end et deux sur le back-end.

Plus concrètement, au fil des itérations, la répartition stricte au sein du groupe s'est estompée. Les tâches restaient attribuées en fonction des compétences de chacun, mais étant tous assez pressés, il était nécessaire que chaque membre s'intéresse à toutes les parties du projet afin de pouvoir avancer sur ses propres tâches sans devoir solliciter en permanence une autre personne spécialisée. Ainsi, à la fin du projet, nous connaissons tous, plus ou moins, l'intégralité du système.

Nous allons maintenant vous présenter notre planning d'organisation qui a été mis en place pour les 7 itérations :

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.

Itération 5 : Optimisation des fonctionnalités et corrections

L'objectif de cette itération était 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 nous a permis d'automatiser certaines tâches et d'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.
- 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.
- Amélioration des justifications Timefold pour les retours utilisateurs.
- Mise en place d'une sécurité entre les différentes sessions d'entraîneur et d'administrateur.
- Finitions du drag and drop et optimisations de ce dernier.
- Possibilité de créer une nouvelle session vide.
- Filtrage des différentes données et sessions.

Itération 7 : Déploiement et documentation

L'ultime itération a été consacrée à la mise en production du projet et à la documentation. Il y a aussi eu un travail de finalisation.

- **Ajout des dernières demandes faites par M. Brandt**, c'est-à-dire l'ajout de différents champs supplémentaires pour le formulaire d'inscription, et correction des derniers bugs dans le front-End
- **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.

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.

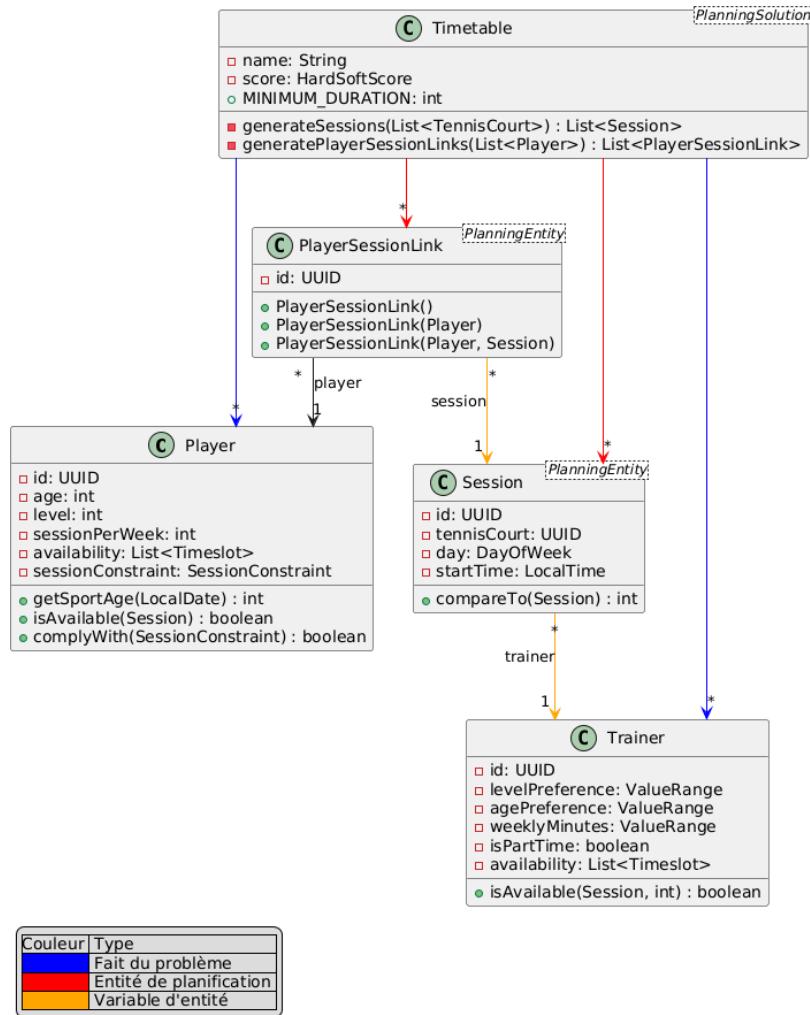


Figure : Diagramme de classe du domaine Timefold

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.

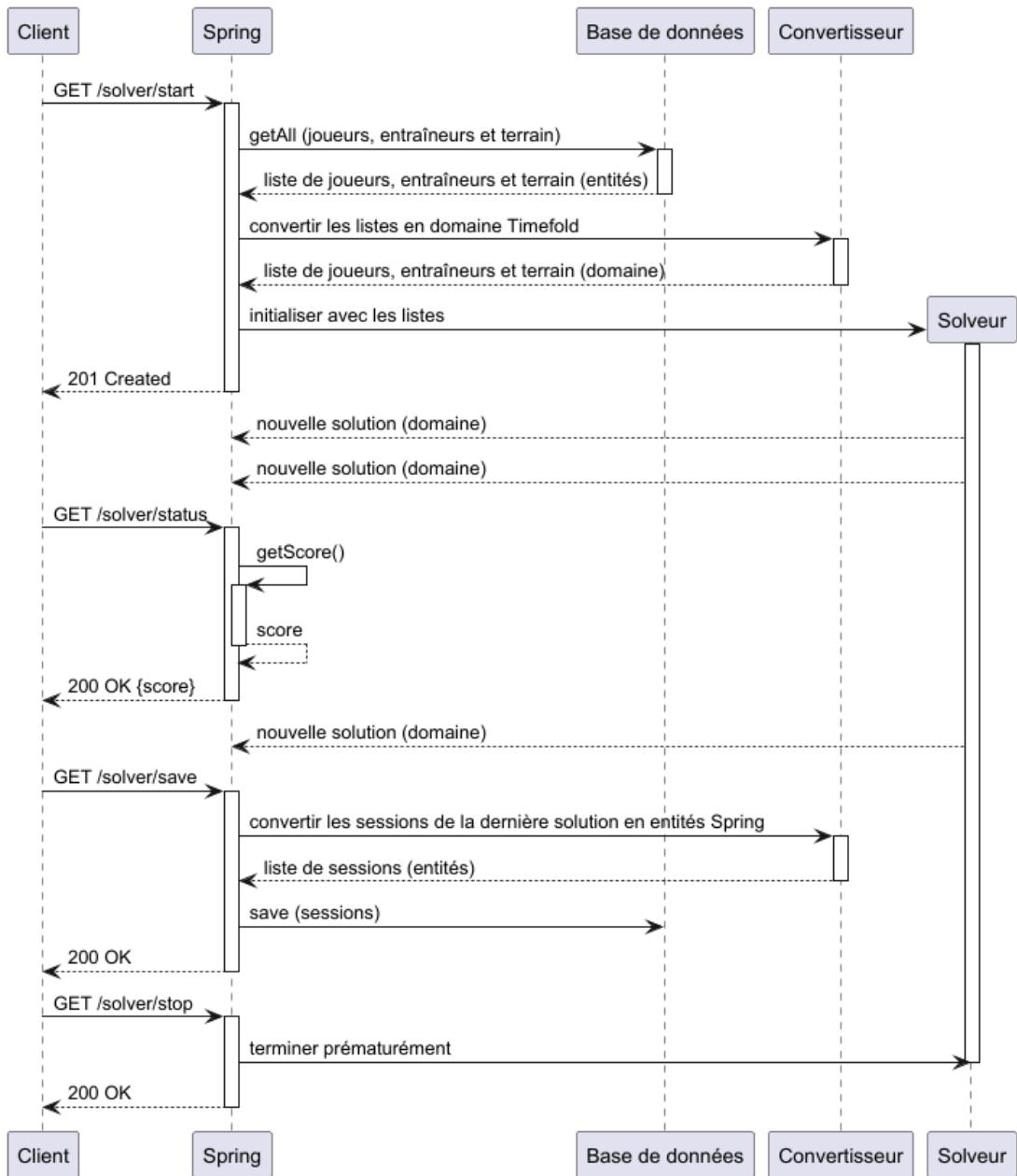


Figure : Diagramme de séquence des quatres requêtes du contrôleur de Timefold

Pour ce qui est de la base de données, nous avions 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

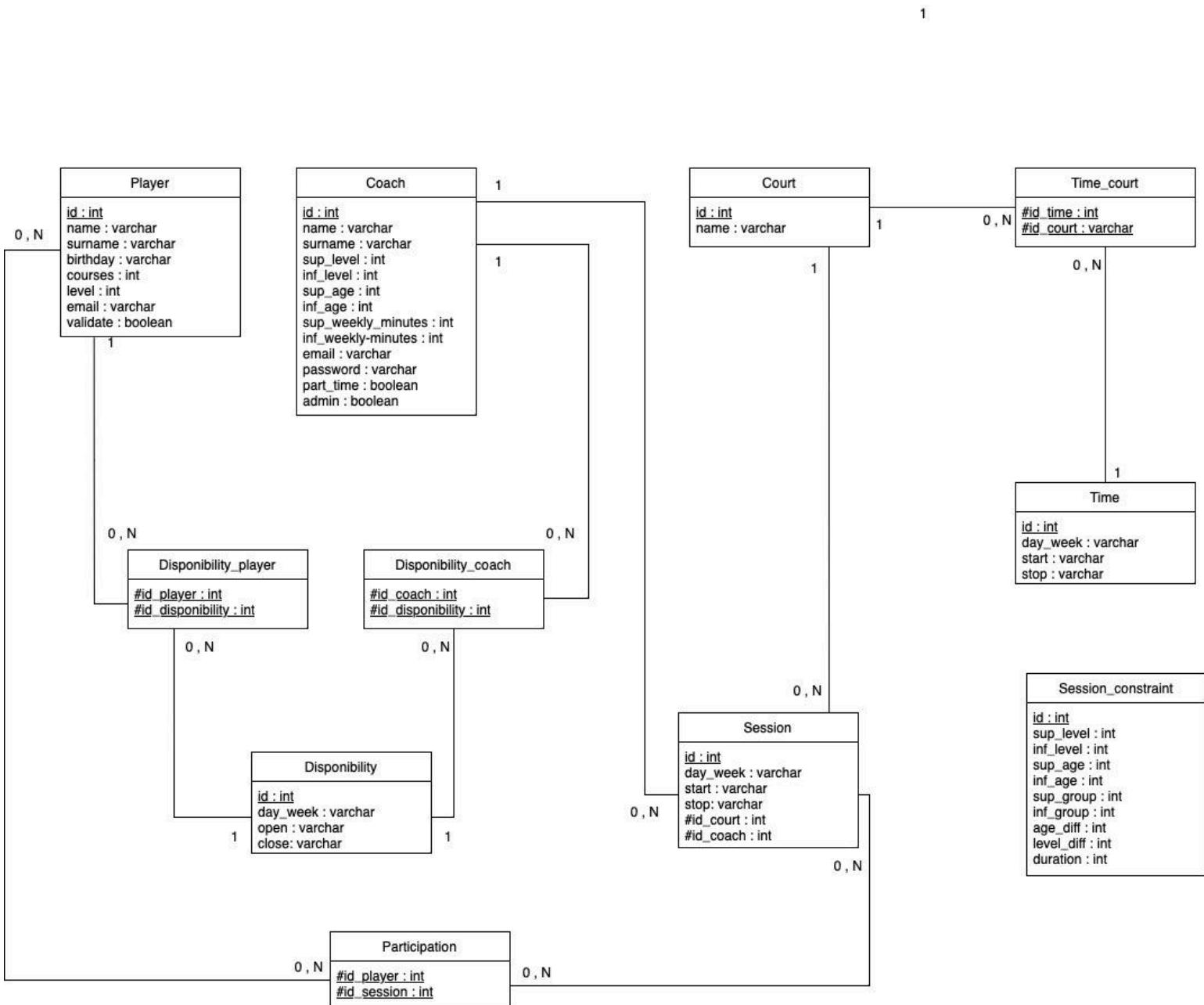


Figure : Schéma relationnel de la base de données

Pour le backend nous avions 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.

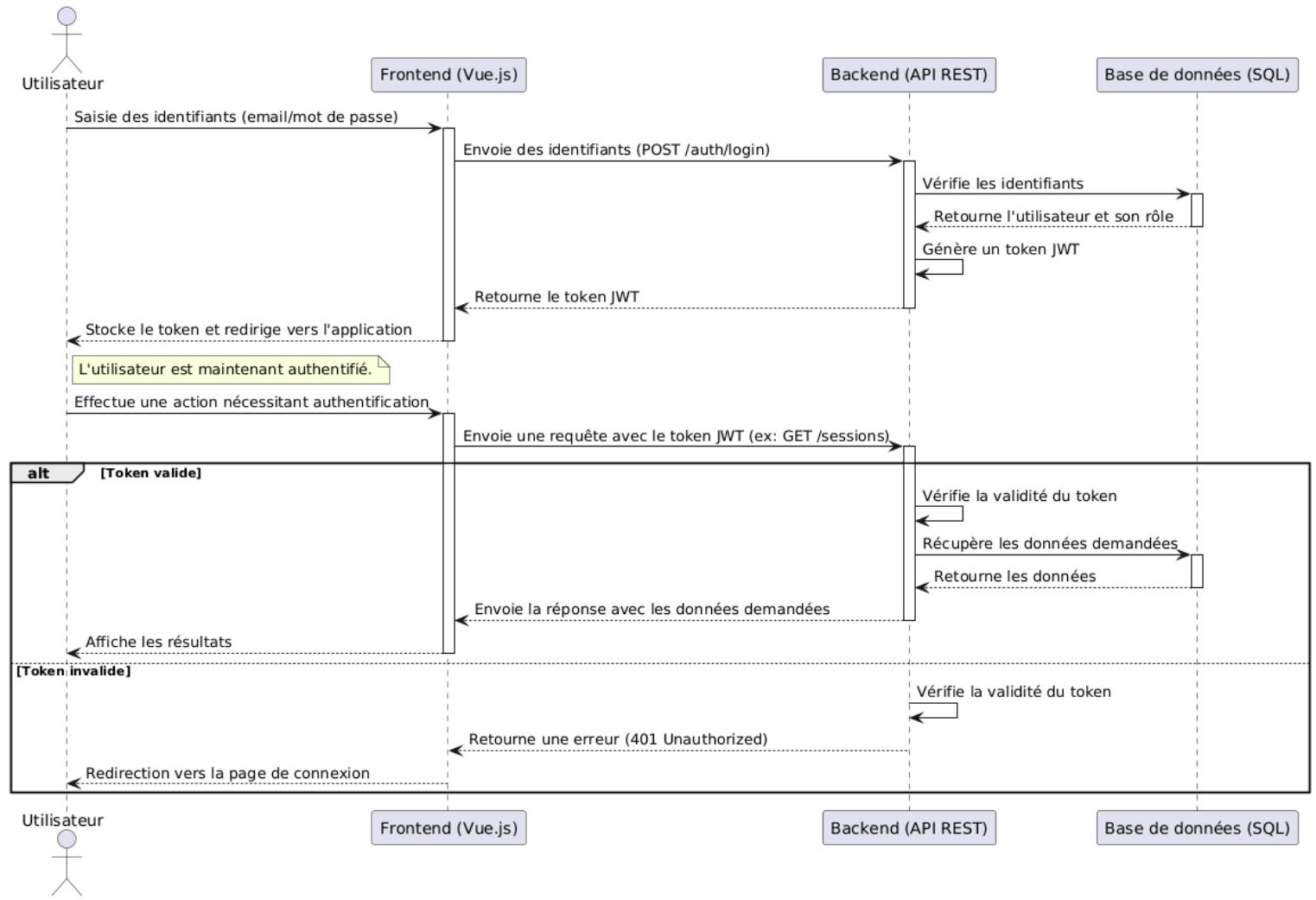


Figure : Diagramme de séquence du fonctionnement de l'authentification avec JWT

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.

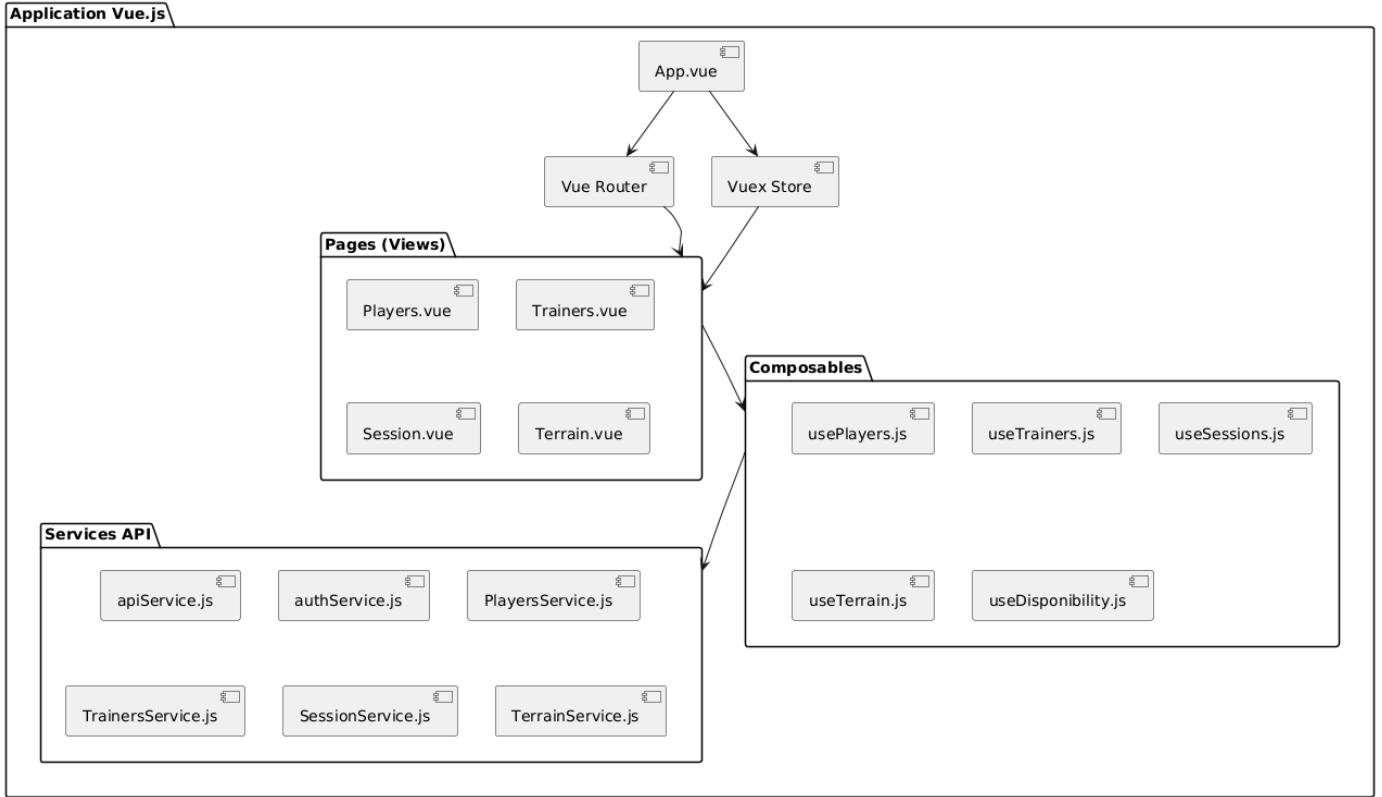


Figure : Architecture simplifiée du Front-end Vue.js

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 tests. 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 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, il y a d'abord "data_structure" qui contient les différents éléments utilisés par Timefold. Il y a ensuite "timefold" qui est utilisé tout ce qui concerne Timefold et qui a été expliqué précédemment. Le troisième package est "data_converter" qui est celui qui permet de convertir les objets de Timefold en objets de l'API et inversement. Le dernier package est composé de sous package, il s'agit de l'API et contient un package pour la configuration du serveur avec tout ce qui est utilisé pour la sécurité de l'API, un second avec toutes les exceptions personnalisées, un troisième avec tous les outils comme les constructeurs de mail et le dernier est le cœur même de l'application avec de nouveaux des sous packages pour chaque métier.

1. Gestion de la base de données

La base de données est 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 sont exécutés automatiquement lors du lancement de l'application.

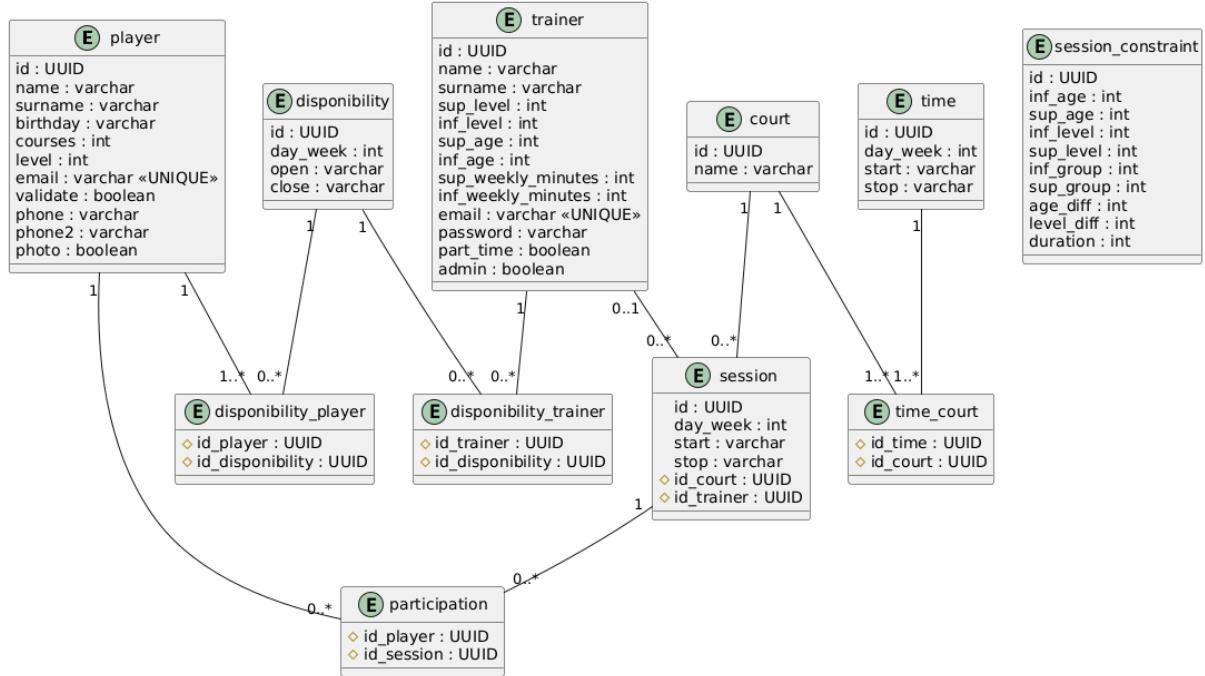


Figure : Diagramme de classe de la base de données

2. Lien entre Spring boot et la base de données

Pour lier Spring boot et la base de données pour un développement local et pouvoir avoir les logs du serveur 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.

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. Chaque domaine métier possède une entité, un repository et un service.

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 nous avons plusieurs manière de faire, on peut dans un premier temps 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 nous pouvons aussi lancer directement le docker compose contenant la base de données, le back-end 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. Certains métiers contiennent plusieurs dto, il y en a plusieurs car certaines requêtes ne nécessitent pas ou ne renvoient chaque élément d'un objet.

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. Les entités sont convertis en dto contenant tous les attributs de l'objet mais lorsqu'un dto ne contenant pas tous les attributs, on le modifie en un dto "complet" pour que la conversion se passe sans problème.

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 standardisé. 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.

```
Trainer ▾ {  
    id                      string($uuid)  
    name                    string  
    surname                string  
    agePreference          ValueRange ▾ {  
        min                  integer($int32)  
        max                  integer($int32)  
    }  
    levelPreference         ValueRange > {...}  
    weeklyMinutes           ValueRange > {...}  
    availability           > [...]  
    partTime                boolean  
}
```

Figure : Schéma d'un entraîneur sur Swagger

Example Value | Schema

```
{  
  "name": "Philippe Chatrier",  
  "times": [  
    {  
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",  
      "dayWeek": 5,  
      "start": "8:00",  
      "stop": "18:00"  
    }  
  ]  
}
```

Figure : Schéma d'une réponse d'une requête sur Swagger

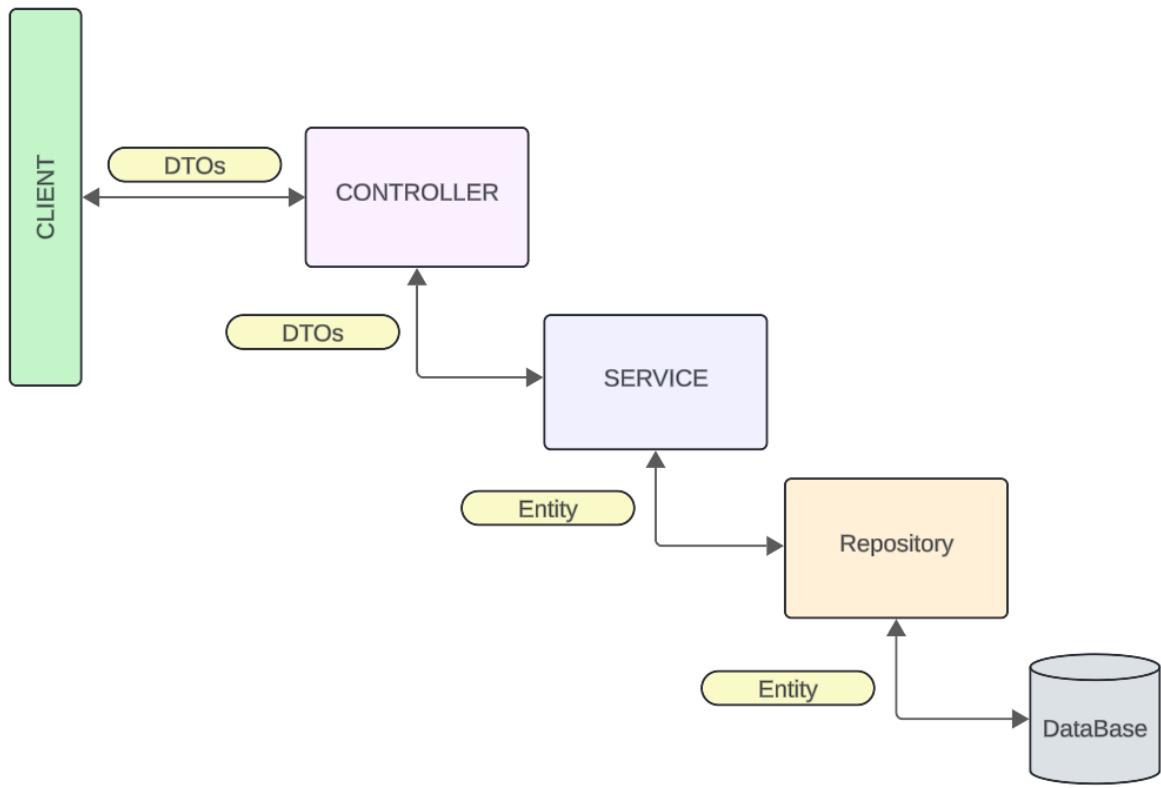


Figure : Schéma d'un package métier allant de la requête du client jusqu'à la base de données

4. Mise en place de la sécurité

Dans un premier temps, nous avons mis en place un token. 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.

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0aG9tYX
MubWljaGVsaTZAZXR1LnVuaXYtbG9ycmFpbmUuZ
nIiLCJpYXQiOjE3NDA2NDIzMjAsImV4cCI6MTc0
ODUzMTcyMH0.VBJ7f9DXLQEG4HGuq-
9iQ4nRtP6oMAxUCBkOv1Ayn-E
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE
{ "alg": "HS256" }
PAYOUT: DATA
{ "sub": "thomas.micheli6@etu.univ-lorraine.fr", "iat": 1740642328, "exp": 1748531728 }
VERIFY SIGNATURE
HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) <input type="checkbox"/> secret base64 encoded

Figure : Exemple d'un token JWT

Nous avons séparé les entraîneurs en deux parties, il y a les administrateurs et les simples entraîneurs. Nous avons filtré certaines requêtes en fonction du rôle de l'entraîneur, ainsi, un administrateur pourra accéder à chaque élément de l'application et n'aura aucune restriction sur les requêtes alors qu'un simple entraîneur ne pourra faire de requêtes que sur les méthodes “get”, il ne pourra donc uniquement visualiser les différentes données de l'application sans jamais pouvoir les modifier. Nous avons également configuré les CORS, uniquement l'adresse ip ou le docker contenant le front-end pourra faire des requêtes à notre API.

5. Requête utilisable

Les requêtes POST et PUT contiennent les nouveaux objets dans le body de la requête. Certaine requête nécessite un “{id}” qui est l’UUID de l’objet à récupérer, modifier ou supprimer. Voici différents tableaux permettant de visualiser les différentes requêtes possibles.

Terrains :

Type de méthode	Path	Variable du path	Body de la requête	Body de la réponse	Utilisation de la requête
GET	/courts	X	X	La liste des terrains	Récupère tous les terrains
PUT	/courts/{id}	id = UUID du terrain à modifier	Le nouvel objet remplaçant l'ancien associé à l'id	L'objet avec les nouveaux attributs	Modifie un terrain

			donné		
POST	/courts	X	Le nouvel objet à ajouter dans la base de données	Le nouvel objet	Ajoute un terrain
DELETE	/courts/{id}	id = UUID du terrain à supprimer	X	X	Supprime un terrain

Sessions :

Type de méthode	Path	Variable du path	Body	Body de la réponse	Utilisation de la requête
GET	/sessions	X	X	La liste des sessions	Récupère toutes les sessions
GET	/sessions/send Mail	X	X	Un booléen indiquant si l'envoie s'est bien effectué ou non	Envoie un mail à chaque joueur contenant leur planning de la semaine
PUT	/sessions/{id}	id = UUID de la session à modifier	Le nouvel objet remplaçant l'ancien associé à l'id donné	L'objet avec les nouveaux attributs	Modifie une session
POST	/sessions	X	Le nouvel objet à ajouter dans la base de données	Le nouvel objet	Ajoute une session
DELETE	/sessions/{id}	id = UUID de la session à supprimer	X	X	Supprime une session

Contraintes de sessions :

Type de méthode	Path	Variable du path	Body	Body de la réponse	Utilisation de la requête
GET	/constraints	X	X	La liste des contraintes de sessions	Récupère toutes les contraintes de sessions
PUT	/constraint/{id}	id = UUID	Le nouvel	L'objet avec	Modifie un

		du terrain à modifier	objet remplaçant l'ancien associé à l'id donné	les nouveaux attributs	terrain
POST	/constraints	X	Le nouvel objet à ajouter dans la base de données	Le nouvel objet	Ajoute une contrainte de session
DELETE	/constraints/{id}	id = UUID de la contrainte de session à supprimer	X	X	Supprime une contrainte de session

Joueurs :

Type de méthode	Path	Variable du path	Body	Body de la réponse	Utilisation de la requête
GET	/players?validate={boolean}	boolean = booléen indiquant si l'inscription a été validé ou non	X	La liste de tous les joueurs associé au statut de l'inscription	Récupère les joueurs associé au statut de l'inscription
GET	/players/{id}	id = UUID du joueur à récupérer	X	Le joueur associé à l'id	Récupère un joueur
PUT	/players/{id}	id = UUID du joueur à modifier	Le nouvel objet remplaçant l'ancien associé à l'id donné	L'objet avec les nouveaux attributs	Modifie un joueur
POST	/players	X	Le nouvel objet à ajouter dans la base de données	Le nouvel objet	Ajoute un joueur
DELETE	/players/{id}	id = UUID du joueur à supprimer	X	X	Supprime un joueur
DELETE	/players/confirm	X	X	X	Supprime tous les joueurs

Entraîneurs :

Type de méthode	Path	Variable du path	Body de la requête	Body de la réponse	Utilisation de la requête
GET	/trainers	X	X	La liste des entraîneurs	Récupère tous les entraîneurs
GET	/trainers/{id}	id = UUID de l'entraîneur à modifier	X	L'entraîneur associé à l'id	Récupère l'id de l'entraîneur
PUT	/trainers/{id}	id = UUID de l'entraîneur à modifier	Le nouvel objet remplaçant l'ancien associé à l'id donné	L'objet avec les nouveaux attributs	Modifie un entraîneur
PATCH	/trainers/change-password	X	Le nouveau mot de passe	X	Change le mot de passe d'un entraîneur
POST	/trainers	X	Le nouvel objet à ajouter dans la base de données	Le nouvel objet	Ajoute un entraîneur
POST	/trainers/reset-password	X	Le mail de l'entraîneur	X	Envoie un mail pour réinitialiser le mot de passe de l'entraîneur
DELETE	/trainers/{id}	id = UUID de l'entraîneur à supprimer	X	X	Supprime un entraîneur

Authentification :

Type de méthode	Path	Variable du path	Body de la requête	Body de la réponse	Utilisation de la requête
POST	/auth/login	X	Les données de connexion	Le token lié à l'entraîneur connecté	Permet à un entraîneur de se connecter

6. Tests de validation

Nous avons dans un premier temps, sur github, les test de notre API mais la partie base de données. Ce sont les tests des dtos, repositorys, services et entités. Nous avons

également sur Postman des séries de tests permettant de tester les controller avec les différents codes réponses.

The screenshot shows a Postman collection structure for testing a Player table. It includes various requests (GET, POST, PUT, DELETE) with their expected status codes and error messages.

```

    {
      "name": "Player",
      "item": [
        {
          "name": "GetPlayerValidate",
          "item": [
            {
              "method": "GET",
              "url": "/api/player/validate",
              "status": "200"
            },
            {
              "method": "GET",
              "url": "/api/player/validate",
              "status": "400 - No validate"
            },
            {
              "method": "GET",
              "url": "/api/player/validate",
              "status": "400 - Wrong value"
            }
          ]
        },
        {
          "name": "GetPlayer",
          "item": [
            {
              "method": "GET",
              "url": "/api/player",
              "status": "200"
            },
            {
              "method": "GET",
              "url": "/api/player",
              "status": "400 - Not UUID"
            },
            {
              "method": "GET",
              "url": "/api/player",
              "status": "404"
            }
          ]
        },
        {
          "name": "CreatePlayer",
          "item": [
            {
              "method": "POST",
              "url": "/api/player",
              "status": "201"
            },
            {
              "method": "POST",
              "url": "/api/player",
              "status": "400 - Wrong format"
            },
            {
              "method": "POST",
              "url": "/api/player",
              "status": "400 - No name"
            }
          ]
        }
      ],
      "request": [
        {
          "method": "POST",
          "url": "/api/player",
          "status": "409"
        },
        {
          "method": "PUT",
          "url": "/api/player",
          "status": "200 - Level"
        },
        {
          "method": "PUT",
          "url": "/api/player",
          "status": "200 - Disponibility"
        },
        {
          "method": "PUT",
          "url": "/api/player",
          "status": "400 - No name"
        },
        {
          "method": "PUT",
          "url": "/api/player",
          "status": "400 - Wrong format"
        },
        {
          "method": "PUT",
          "url": "/api/player",
          "status": "404"
        },
        {
          "method": "PUT",
          "url": "/api/player",
          "status": "409"
        },
        {
          "method": "DELETE",
          "url": "/api/player",
          "status": "204"
        },
        {
          "method": "DELETE",
          "url": "/api/player",
          "status": "400"
        },
        {
          "method": "DELETE",
          "url": "/api/player",
          "status": "404"
        }
      ]
    }
  
```

Figure : Exemple de tous les tests Postman pour tester les requêtes de la table Player

Front-End

1. Documentation à destination du client

1.1 Description des vidéos

Cette partie décrit l'ensemble des fonctionnalités du projet. Elle fournira, pour chaque fonctionnalité, le nom de la vidéo et le timecode (si nécessaire) du moment présentant précisément cette fonction. Il existe également la vidéo *exemple_complet_de_traitement.mp4*, qui montre un déroulement complet de l'application, de l'inscription des joueurs à l'envoi des emplois du temps. Les vidéos sont disponibles dans le dépôt GitHub.

editions_des_contraintes.mp4 :

- créer une contrainte de séance ;
- modifier une contrainte de séance ;

- supprimer une contrainte de séance.

editions_des_entraineurs.mp4 :

- créer un entraîneur ;
- modifier un entraîneur ;
- supprimer un entraîneur.

editions_des_horaires_des_terrains.mp4 :

- créer un horaire de terrain ;
- modifier un horaire de terrain ;
- supprimer un horaire de terrain.

editions_des_joueurs.mp4

- créer un joueur ;
- modifier un joueur ;
- supprimer un joueur.

editions_des_sessions.mp4

- ajouter un entraîneur à une session ;
- changer l'entraîneur d'une session ;
- changer un joueur de session ;
- ajouter un joueur à une session ;
- supprimer un joueur d'une session ;
- changer l'heure d'une session ;
- supprimer une sessions
- créer une session.

filtrage_des_donnees.mp4

- rechercher un joueur ;
- rechercher un entraîneur ;
- rechercher les sessions d'un entraîneur ;
- rechercher les sessions d'un joueur ;
- rechercher les sessions avec un âge et un niveau spécifique.

generation_des_emplois_du_temps.mp4

- lancer la génération ;
- télécharger une sauvegarde ;
- valider les éléments un par un ;
- tout valider ;
- importer une sauvegarde.

gestion_de_la_nouvelle_annee.mp4

- supprimer tous les joueurs ;
- consulter les inscriptions ;
- envoyer les créneaux de chaque joueur à leur email.

gestion_des_donnees.mp4

- télécharger les données (entraîneurs, joueurs, horaires des terrains, contraintes des sessions) ;
- insérer les données (précédemment téléchargé ou créé manuellement) ;
- télécharger les sessions planifiées ;
- télécharger l'emploi du temps en pdf.

inscription_des_joueurs.mp4

- inscription grâce au formulaire ;
- validation de l'adresse email.

reinitialiser_un_mot_de_passe.mp4

- accéder au formulaire de demande ;
- réaliser une demande de réinitialisation ;
- réception du mail et accès au formulaire de changement de mot de passe ;
- changement du mot de passe.

1.2 Conseils d'utilisation

1. Utilisation du glisser-déposer

Lors de la gestion des sessions, vous serez amenés à déplacer des joueurs et des entraîneurs. Pour cela, ne soyez **pas trop rapides** et attendez que les différents chargements soient terminés, notamment dans la partie des données à gauche. Si vous pensez avoir rencontré un bug, n'hésitez pas à **recharger la page du site**.

2. Sauvegarde de la génération

Après avoir lancé le générateur d'emploi du temps, n'oubliez pas d'enregistrer l'ensemble. Cela vous permettra d'annuler toutes vos modifications en important le fichier précédemment téléchargé. Deux éléments sont à noter :

- Le téléchargement fournit la version non modifiée de la génération, même si vous éditez des sessions entre-temps.
- La suppression d'un joueur, d'un terrain ou d'un entraîneur rendra impossible l'importation des générations précédemment téléchargées.

3. Conservation des joueurs pour la nouvelle année

Si vous souhaitez conserver des joueurs lors du passage à la nouvelle année (suppression de tous les joueurs), nous vous conseillons d'utiliser la fonction d'import et d'export des données. En effet, vous pouvez :

- télécharger les données ;
- dans le fichier :
 - supprimer les entraîneurs, les terrains et les contraintes ;
 - faire le tris parmi les joueurs à conserver ou non ;
- utiliser le bouton ‘Supprimer tous les joueurs’ ;
- réimporter le fichier modifié.

4. Faire un visuel d'emploi du temps personnalisé

Si l'emploi du temps pdf fourni ne convient pas à vos besoins, vous pouvez utiliser le bouton ‘Télécharger les sessions par terrain’ pour obtenir la liste des sessions. Ainsi, vous pourrez les trier et les ordonner comme bon vous semble.

5. Création d'un entraîneur

Un entraîneur possède un rôle qu'il est possible de changer avec la case 'Administrateur' de son profil. Si la case est cochée, il aura les mêmes accès que vous, autrement, il aura un accès lecture au site lui permettant de voir à distance :

- la composition des séances ;
- le niveau et l'âge des joueurs ;
- les horaires des terrains.

Après la création d'un entraîneur, aucun mot de passe ne lui est attribué, le seul moyen d'en obtenir un est la réinitialisation de mot de passe. Afin d'obtenir le lien de réinitialisation, il faudra donc indiquer l'email de l'entraîneur.

6. Création des contraintes de séance

Les contraintes des séances ne sont utiles que lors de la génération automatique de l'emploi du temps. Il faut impérativement que chaque joueur puisse intégrer l'une des contraintes existantes. Un joueur peut être intégré si :

- son âge est dans la fourchette d'âge de la contrainte ;
- son niveau est dans la fourchette de niveau de la contrainte ;
- la différence de niveau, la différence d'âge, la durée et l'effectif n'ont pas d'incidence.

2. 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 :

Page d'administration

Page de connexion

constraint

availability

Figure : Maquette Figma

Le résultat actuel de la plateforme est celui ci :

The screenshot shows the main dashboard of the platform. At the top, there are tabs for "Données" (Data), "Contraintes" (Constraints), and "Paramètres" (Parameters). Below these are sections for "Entraîneurs" (Coaches) and "Joueurs" (Players).

Entraîneurs:

Nom	Prénom	Niveau	Âge	Nb. Min-Max	Min-Max	Nb. heures
Brandt	Pierre	1 - 1	1 - 1	0 / 0		
BRANDT	Pierre	14 - 19	1 - 99	3 / 27		
HOUTMANN	Bastien	1 - 19	1 - 99	37 / 28		
DESSOY	Pierre	1 - 15	1 - 99	3 / 10		
HOMBOURGER	Malo	1 - 14	1 - 99	0 / 0		
BODOT	Lucie	1 - 13	1 - 99	1 / 5		
MONTAGNE	Nicolas	1 - 14	1 - 99	0 / 0		
HOMBOURGER	Solal	1 - 12	1 - 18	3 / 0		
DOYEN	Manon	1 - 12	1 - 18	0 / 5		
MARIE	Maxence	1 - 12	1 - 18	0 / 0		
ADLINE	Thomas	1 - 12	1 - 18	3 / 5		
Achraf	Raouf	1 - 12	12 - 19	1.5 / 0		

Joueurs:

Nom	Prénom	Âge	Niveau	Sessions
BRANDENB...	Arthur	14 ans	11	5 / 2
FIXE-MARTZ	Arthur	14 ans	10	4 / 2
VIGNERON	Théo	17 ans	14	4 / 2
achrafaaa...	lpb	21 ans	10	2 / 1
BARBARY	Louise	10 ans	7	2 / 1
BENNEOUA...	Liam	9 ans	6	2 / 1
BENNEOUA...	Neal	7 ans	5	2 / 1
BRUMBT	Anais	99 ans	14	2 / 1

Planning (Lundi à Mercredi):

- Lundi:** Sessions de 17:00 à 21:30 avec entraîneurs HOUTMANN Bastien, BODOT Lucie, et BRANDT Pierre.
- Mardi:** Session de 13:30 à 14:30 avec entraîneur HOMBOURGER Solal.
- Mercredi:** Pas de session programmée.

At the bottom right, there are buttons for "Lancer" (Launch) and "Importer" (Import).

Figure : Page d'admin

The screenshot shows the "Contraintes" (Constraints) and "Paramètres" (Parameters) sections of the platform.

Terrains:

- Terrain 1:** Ouverture (Opening): 17:00; Fermeture (Closing): 22:30. Days: Lundi, Mercredi, Jeudi, Vendredi, Samedi.
- Terrain 2:** Ouverture: 13:00; Fermeture: 22:30.
- Terrain 3:** Ouverture: 17:00; Fermeture: 22:30.

Gestion des données:

- Insérer les données (.xlsx)
- Télécharger les données (.xlsx)
- Télécharger les sessions par terrain (.xlsx)
- Télécharger le planning (.pdf)

Gestion des joueurs:

- Consulter les inscrits
- Envoyer le planning aux joueurs
- Supprimer tous les joueurs

Figure : Onglets Contraintes / Paramètres

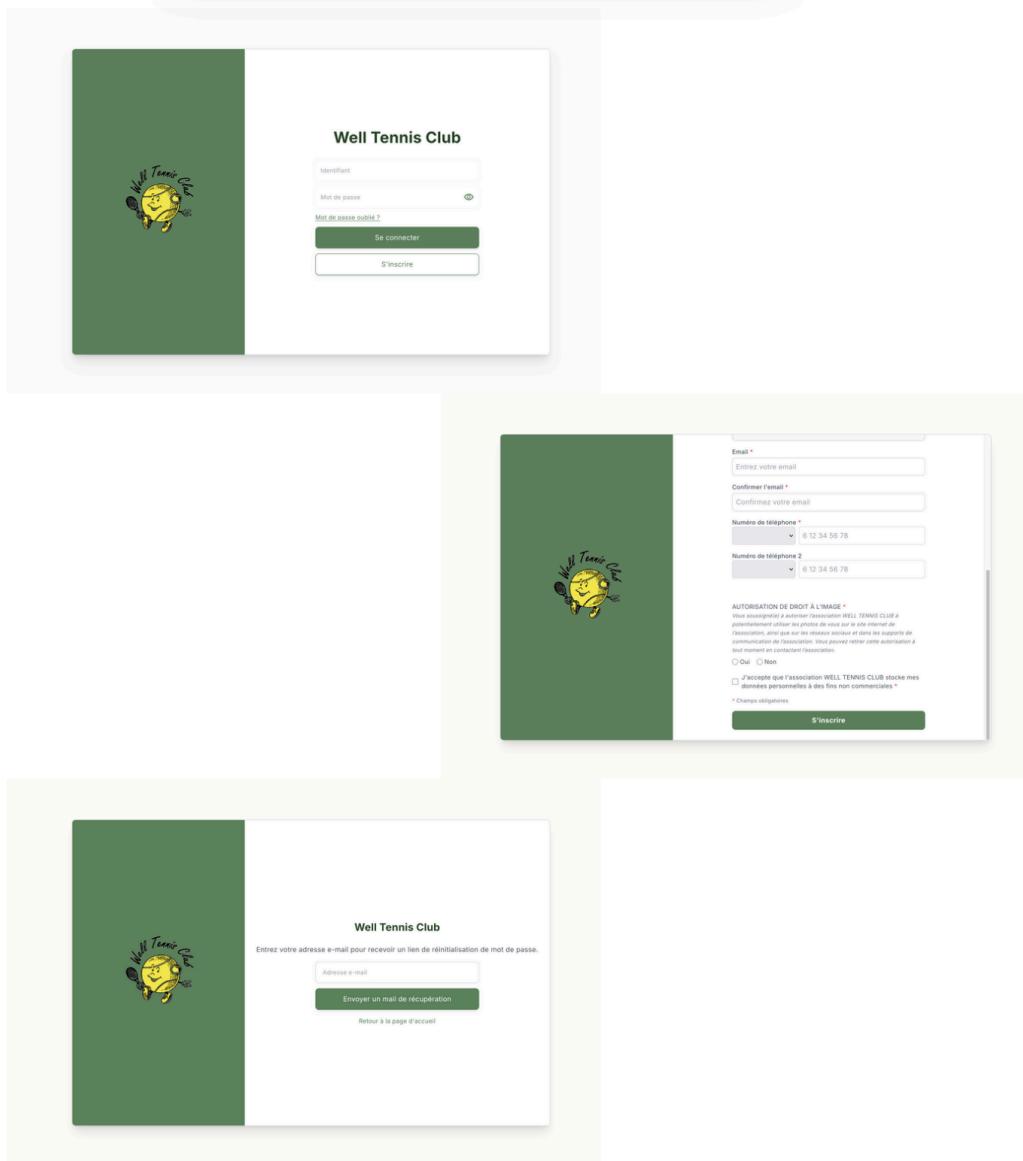


Figure : Page de connexion / Inscription / Mot de passe oublié

3. 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 :

3.1 Génération de l'emploi du temps

Une fois connecté et après avoir validé les inscriptions, l'administrateur définit les contraintes liées aux terrains et aux sessions. Il peut ensuite lancer la génération de l'emploi du temps en cliquant sur le bouton « **Lancer** ». Ce bouton déclenche l'exécution du **Solver**,

qui récupère toutes les informations depuis la base de données et commence la génération automatique de l'emploi du temps.

Le Solver peut effectuer plusieurs itérations jusqu'à trouver la meilleure solution possible. Une fois terminé, il enregistre les sessions générées dans la base de données et affiche un bouton « **Valider la génération** ». Ce bouton permet à l'administrateur de gérer manuellement les sessions qui n'ont pas respecté certaines contraintes — par exemple, une session contenant un joueur de trop ou une session avec un seul joueur inscrit. Il peut personnaliser les contraintes en retirant des joueurs ou l'entraîneur, puis valider ou non chaque session selon ses préférences. Si une session est validée, elle disparaît de l'onglet de validation. Si elle est supprimée, elle est retirée de la base de données ainsi que de l'affichage.

De plus, lorsque le Solver a terminé, l'administrateur peut télécharger les sessions dans un fichier chiffré avec l'extension **.rmpk** (*RaoufMicheliPedronKetzinger*). Ce fichier représente une sauvegarde de la génération effectuée par **Timefold**. Il peut être utilisé pour restaurer l'état initial de l'emploi du temps si l'administrateur effectue des modifications mais souhaite finalement revenir à la version originale. Il lui suffit alors d'importer à nouveau le fichier **emploidutemps.rmpk**, et la plateforme rechargera la génération d'origine.

The screenshot shows the 'BottomPanel' interface. On the left, there are two tabs: 'Données' (selected) and 'Contraintes'. The 'Données' tab contains two tables: one for 'Entraîneurs' (Coaches) and one for 'Joueurs' (Players). The 'Entraîneurs' table includes columns for Nom, Prénom, Niveau, Âge, and Nb. heures. The 'Joueurs' table includes columns for Nom, Prénom, Âge, Niveau, and Sessions. On the right, the 'Contraintes' tab displays a weekly calendar with days from Lundi to Samedi, each with a green circular icon. At the bottom center, there is a red-bordered button labeled 'Lancer' (Launch) and an 'Importer' (Import) button to its right.

Figure : BottomPanel où sont affichés les boutons de génération

The screenshot shows a software interface for managing training sessions. On the left, there are two tables: one for 'Entraîneurs' (Coaches) and one for 'Joueurs' (Players). The 'Entraîneurs' table includes columns for Nom (Name), Prénom (First Name), Niveau (Level), Âge (Age), and Nb. heures (Number of hours). The 'Joueurs' table includes columns for Nom, Prénom, Âge, Niveau, and Sessions. On the right, a calendar view shows days from Monday to Saturday. At the bottom, there are three buttons: 'Télécharger' (Download), 'Valider la génération' (Validate generation, highlighted with a red box), and 'Importer' (Import).

Figure : Bouton affiché une fois la génération terminée, permettant la gestion des sessions non conformes

The screenshots show validation pages for different entities:

- Validation de la session du lundi de 08:00 à 09:00 sur le terrain 2:** Shows a session from Monday 08:00 to 09:00 on field 2. It notes a superposition with another session at 08:30. It includes fields for 'Entraîneur' (Athena Garrett checked) and 'Préférences' (1-18 • 1-12). Buttons: 'Valider la session' (highlighted), 'Supprimer la session'.
- Validation du joueur Alexandra Vance:** Shows player details (Age: 15, Niveau: 10) and availability checkboxes for fields 1 and 2 on specific dates. Buttons: 'Valider le joueur' (highlighted), 'Supprimer la session'.
- Validation de l'entraîneur Athena Garrett:** Shows coach details (Préf. âge: 1 à 18 ans, Préf. niveau: 0 à 12) and availability checkboxes for fields 1 and 2 on specific dates. Buttons: 'Valider le joueur' (highlighted), 'Supprimer la session'.

Figure : Page de validation des sessions générées hors contraintes (joueurs en trop, sessions incomplètes, etc.)

3.2 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 leurs nombres de cours qui

leurs sont attribués. Il a également accès aux informations sur les entraîneurs, telles que leur nom, prénom, niveau minimum et maximum qu'ils peuvent entraîner, l'âge minimum et maximum des joueurs qu'ils peuvent encadrer. Il est aussi affiché le nombre d'heures que l'entraîneur peut effectuer et le nombre d'heures qui lui est déjà attribué.

3.3 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.

The screenshot displays the RightPanel component with a header containing three tabs: "Terrain 1" (highlighted in red), "Terrain 2", and "Terrain 3". A "Filtrer" button is located on the right. The main area is divided by day: "Lundi", "Mardi", and "Mercredi". Each day section contains one or more session cards. Each card shows the following information:

- Entraîneur :** Name of the coach.
- Heure :** Session start time (e.g., 17:00, 18:30).
- Age :** Age group (e.g., 14 - 17, 13 - 18).
- Niveau :** Skill level (e.g., 7, 13 - 18).
- Joueurs :** A list of four players per session, divided into two columns.
- Suppression bouton :** A small red trash can icon used to delete the session.

For Lundi:

- Session 1: Coach HOUTMANN Bastien, 17:00, 14 - 17, 13 - 18. Players: JACQUES Léo, L'HUILLIER Pacôme, MARCHAND Ilona, LEDROIT Hector; BLIN Louison, EVE Grégoire, FINANCE Sacha.
- Session 2: Coach BODOT Lucie, 18:30, 11. Players: GREGOIRE THOMAS Elias, JEANNOT Tom.
- Session 3: Coach BRANDT Pierre, 20:00, 18. Players: CHANDECLERC Alois, GRENOT Martin.

For Mardi:

- Session 1: Coach HOMBOURGER Solal, 13:30, 9 - 13, 5 - 7. Players: ATLANI Tom, MAGISSON-KLEIN Amos, MANTE Gabin, MASSEHIAN Léonard; GANSMÜLLER Théodore, ZEYER Matthieu, BRIMONT Alice.

For Mercredi:

- Session 1: Coach BODOT Lucie, 18:30, 11. Players: GREGOIRE THOMAS Elias, JEANNOT Tom.

Figure : Composant RightPanel où sont affiché les sessions

This screenshot shows a detailed view of two sessions from Monday's list. Each session card includes the following information:

- Entraîneur :** Name of the coach.
- Heure :** Session start time (e.g., 17:00, 18:30).
- Age :** Age group (e.g., 14 - 17, 11).
- Niveau :** Skill level (e.g., 13 - 18, 7).
- Joueurs :** A list of four players per session, divided into two columns.
- Suppression bouton :** A small red trash can icon used to delete the session.

Session 1 (17:00): Coach HOUTMANN Bastien, 14 - 17, 13 - 18. Players: JACQUES Léo, L'HUILLIER Pacôme, MARCHAND Ilona, LEDROIT Hector; BLIN Louison, EVE Grégoire, FINANCE Sacha.

Session 2 (18:30): Coach BODOT Lucie, 11. Players: GREGOIRE THOMAS Elias, JEANNOT Tom.

Figure : Exemple de séance

3.4 Créer une session

L'administrateur peut, depuis sa page de gestion, créer une session vide. Pour chaque jour et chaque terrain, un bouton « + » est disponible. En cliquant dessus, une session par défaut est automatiquement créée pour le jour sélectionné. Par défaut, cette session a une durée d'une heure et commence à minuit. L'administrateur peut ensuite modifier l'horaire et y ajouter les joueurs ainsi que l'entraîneur grâce à une interface de glisser-déposer (drag and drop).

The screenshot shows a web-based application for managing sessions. On the left, there are two tables: one for 'Entraîneurs' (Coaches) and one for 'Joueurs' (Players). Both tables include columns for Nom (Name), Prénom (First Name), Niveau (Level), Âge (Age), and Nb. Min+Max (Nb. Min+Max). Below these tables are buttons for 'Télécharger' (Download), 'Valider la génération' (Validate generation), and 'Importer' (Import). On the right, a schedule is displayed for 'Terrain 1', 'Terrain 2', and 'Terrain 3'. The days of the week are listed as 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi', and 'Dimanche'. The 'Lundi' row is highlighted with a red box around the 'Lundi' cell.

Figure : Exemple de séance

Il y a la possibilité de modifier à la main l'heure d'une séance

This screenshot shows a detailed view for Monday ('Lundi'). It includes a sidebar with time slots from 00:00 to 01:00 and a green 'Enregistrer' (Register) button. To the right, there are four fields: 'Entraîneur:' (Coach: Aucun entraîneur), 'Âge:' (Age: Pas de joueurs), 'Niveau:' (Level: Pas de joueurs), and a small red trash can icon. Below this, the days of the week are listed: 'Mardi', 'Mercredi', and 'Jeudi', each with a '+' sign to add more sessions.

Figure : Exemple de séance

3.5 Filtrer les sessions

Les utilisateurs peuvent filtrer les sessions en fonction de plusieurs critères : l'entraîneur, les joueurs, l'âge et le niveau de chaque session. Il est possible de combiner plusieurs filtres simultanément. Par exemple, un utilisateur peut rechercher les sessions encadrées par l'entraîneur « Toto » tout en filtrant celles où le joueur « Roto » est également présent. Le système permet donc de croiser l'ensemble de ces critères pour affiner la recherche de manière précise et efficace.

Figure : Filtrage des sessions par entraîneur, joueur, âge et niveau

Figure : Filtrage des sessions par entraîneur

3.6 Afficher les contraintes des terrains

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. Il peut également modifier ces informations directement, afin qu'elles soient prises en compte lors de la génération de l'emploi du temps, ou créer des nouveaux terrains.

Jour	Ouverture	Fermeture
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

Figure : Affichage et création d'un terrain et ses contraintes

3.7 Afficher les contraintes des sessions

Les contraintes sont affichées dans l'onglet gauche de la plateforme. L'administrateur a accès aux contraintes des sessions, notamment l'âge, l'effectif, la durée, la différence de niveau et d'âge. Il peut également modifier ces informations directement, afin qu'elles soient prises en compte lors de la génération de l'emploi du temps, ou créer une nouvelle contrainte de session.

Age	Effectif	Durée
3 - 4	4 - 6	1.0h
Diff. niveau		30
Diff. âge		100
Niveau		0 - 19

Figure : Affichage et création d'une contrainte de sessions

3.8 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	N	O	P	
1	Nom	Prénom	Date de naissance	Âge	Email	Numero 1	Numero 2	Niveau		Cours par semaine	Photo	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi
2	achraf	ra	15/05/2004	20	achraf.raouf12@mail.com	789328989	789328989	5			oui	17:30-19:00					
3	achraf	ra	15/05/2004	20	achraf.ra@mail.com	789328989	789328989	6			non	17:30-19:00					
4																	

Figure : Exemple de fichier à importer

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
1	Nom	Prénom	Date de naissance	Âge	Email	Numero 1	Numero 2	Niveau	Cours par semaine	Photo	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	
2	achraf	ra	15/05/2004	20	achraf.raouf12@mail.com	789328989	789328989	5			17:30-19:00						
3	achraf	ra	15/05/2004	20	achraf.ra@mail.com	789328989	789328989	6			17:30-19:00						
4																	

	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	Achraf	RaoufEZ	1	12	12	19	20	25	AchrafEFZ@	Non	Non
3	Achrafdze	ezedZZE	1	12	12	19	20	25	AchraFZEazd	Non	Non
4											
5											
6											

Figure : Exemple de fichier exporté

3.9 Télécharger l'emploi du temps en pdf

L'administrateur a la possibilité de télécharger l'emploi du temps au format PDF. Pour cela, il lui suffit de se rendre dans l'onglet « Paramètres » et de cliquer sur le bouton « Télécharger le planning ». La plateforme récupère alors toutes les sessions et génère automatiquement un PDF indiquant les jours de la semaine ainsi que les terrains. Les sessions y sont affichées par terrain. Lors de la génération, une couleur est automatiquement attribuée à chaque entraîneur afin de faciliter leur repérage ainsi que celui de leurs sessions sur le document. Concernant les sessions, le document affiche le nom de l'entraîneur, l'heure de la session, le niveau du groupe, l'âge du groupe, ainsi que la liste des joueurs.

Emploi du Temps - Terrain 1												
	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi						
1 HOUTMANN Bastien	17:00 - 19:30						10 HOUTMANN Bastien	17:00 - 18:00	HOUTMANN Bastien	17:00 - 18:30	HOUTMANN Bastien	
14 - 17 ans, Niveau: 13 - 18							15 - 18 ans, Niveau: 15 - 17				10 - 15 ans, Niveau: 9 - 13	
JACQUES Léo, LHULLIER Paoline, MARCHAND Iona, LEDROT Hector, GREGORI Grégoire, FINACHE Sacha, BOUDOT Lucie, GREGORI THOMAS Elias, JEANNOT Tom, BRANDT Pierre, CHANDECLERC Alois, GRENOT Marion							ATLANI Tom, MAGISSON-KLEIN Amos, MANTE Gabin, MARCHAND Léonard, GANSMÜLLER Théodore, ZEYER Matthieu, BRIMONT Alice, HOMBURGER Solal	TATOPoulos Achille, VERBRUGHE Anna, DEDENHGS Achille, DESSOY Pierre, HOUTMANN Bastien, 14 - 18 ans, Niveau: 7	FIEVET-LESSER Hector, HERVE Clémence, LAUMONT KNAUF Andreas, MICHAUX Paul, RICHARD André, ROQUES CARMES Louis, DEDENHGS Achille, MASSEHAN Louis, MC CALLA Raphael, DUBOIS Camille, PIERLOT Achille, HAUSBERGER Hugo, DIAS Rafael			
11 ans, Niveau: 7							Aucun entraîneur	17:00 - 18:00	Tom, Niveau: 3			
									KARIM DELAUNAY Eric, MORTIER Victor, SIMER Margot, HSSINE Khayene, HOUTMANN Bastien, 14 - 18 ans, Niveau: 10 - 15			
									CUNY Jules, DEURENNE CHARON Lucas, MARCHAND Iona, MORTIER Axel, SCHAACK Romuald, SKALLI Karim, DELAUNAY Eric, MORTIER Victor, SIMER Margot, HSSINE Khayene, HOUTMANN Bastien, 14 - 18 ans, Niveau: 10 - 15			
									GRIN Jean-Yves, GURRAUD Thomas, GORCHON Jon, HERVE Yann, LECLERE Justine, MAYER Maxime, RODERMAN Maxime, RODERMAN Jean, VIAL Florence, WEBER Quentin, HOUTMANN Bastien, 17:00 - 19:30			
									99 ans, Niveau: 12 - 16			
									GRIN Jean-Yves, GURRAUD Thomas, GORCHON Jon, HERVE Yann, LECLERE Justine, MAYER Maxime, RODERMAN Maxime, RODERMAN Jean, VIAL Florence, WEBER Quentin, HOUTMANN Bastien, 17:00 - 19:30			
									99 ans, Niveau: 15 - 17			
									LANGLOIS Jordan, LOUIS Benjamin			

Figure : Exemple d'emploi du temps PDF

3.10 Télécharger l'emploi du temps en xlsx

Suite à la demande du client, nous avons développé la possibilité de télécharger les sessions dans un fichier XLSX. Cela permet à l'administrateur de conserver une sauvegarde locale. Le fichier est divisé en plusieurs feuilles selon le nombre de terrains. Dans chaque feuille correspondant à un terrain, on retrouve le jour, l'heure de début, l'heure de fin, l'entraîneur ainsi que les joueurs de chaque session.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1 Terrain	Jour	Début	Fin	Entraîneur	Joueurs															
2 Terrain 1	Lundi	17:00	18:30	Bastien HOU Léo JACQUES, Fabrice LHULLIER, Iona MARCHAND, Hector LEDROT, Louison BLIN, Grégoire EVE, Sacha FINACHE, BOUDOT Lucie, GREGORI THOMAS Elias, JEANNOT Tom, BRANDT Pierre, CHANDECLERC Alois, GRENOT Marion																
3 Terrain 1	Lundi	18:30	19:30																	
4 Terrain 1	Mardi	20:00	21:00																	
5 Terrain 1	Mardi	13:30	14:30	Solal HOMB Tom ATLANI, Amos MAGISSON-KLEIN, Gabin MANTE, Léonard MASSEHAN, Théodore GANSMÜLLER, Matthieu ZEYER, Alice BRIMONT																
6 Terrain 1	Mardi	15:00	16:00	Solal HOMB Edouard SIMER																
7 Terrain 1	Mardi	15:00	16:00																	
8 Terrain 1	Mardi	18:00	19:30	Bastien HOU Adeline LHULLIER, Charles KOCH, Victor DORY, Estebane DROUILLY WENGER, Victor MORTIER, Alice DELESTABLE, Maxime REITER, Georges RODERMAN, Jules VIAL, Alois CHANDECLERC, Ruben PIA																
9 Terrain 1	Jeudi	17:00	18:00	Bastien HOU Adeline LHULLIER, Charles KOCH, Victor DORY, Estebane DROUILLY WENGER, Victor MORTIER, Alice DELESTABLE, Maxime REITER, Georges RODERMAN, Jules VIAL, Alois CHANDECLERC, Ruben PIA																
10 Terrain 1	Jeudi	18:00	19:30	Bastien HOU Jules CUNY, Luce DERENE-CHARON, Iona MARCHAND, Axel MORTIER, Roman SCHAACK, Naïf SKALLI, Paloma HOMBURGER, Emry KARIM DELAUNAY, Victor MORTIER, Margot SIMER, Victoire BIADAUD-LAR																
11 Terrain 1	Jeudi	19:30	21:00																	
12 Terrain 1	Jeudi	21:00	22:30	Bastien HOU Jordan LANGLOIS, Benjamin LOUIS																
13 Terrain 1	Vendredi	17:00	18:30	Bastien HOU Hector FIEVET-LESSER, Clémence HERVE																
14 Terrain 1	Vendredi	19:30	20:30																	
15 Terrain 1	Samedi	09:00	10:30	Bastien HOU Hugo HAUDOUIN, Rafael DAS																
16 Terrain 1	Samedi	12:30	13:30	Bastien HOU André LAUMONT KNAUF, Paul MICHAUX, Achille RICHARD, Daniel CZESTOCHOWSKI, Ninon RICHARD, Louis ROQUES CARMES, Vadim DEDERICH, Louis GABRIEL MASSEHAN, Raphael MC CALLA, Camille DU																
17																				
18																				
19																				
20																				
21																				
22																				
23																				
24																				
25																				
26																				
27																				
28																				
29																				
30																				
31																				
32																				
33																				
34																				
35																				
36																				
37																				
38																				
39																				
40																				
41																				
42																				
43																				
44																				
45																				
46																				
47																				

Figure : Exemple d'emploi du temps XLSX

3.11 Envoi de l'emploi du temps a tout les joueurs

Une fois que l'administrateur a terminé les modifications des sessions, il peut envoyer les emplois du temps à chaque joueur par e-mail. Pour cela, il lui suffit de cliquer sur le bouton « Envoyer le planning aux joueurs ». La plateforme effectue alors une requête via le lien de l'API créé dans le backend, qui se charge ensuite d'envoyer à chaque joueur son emploi du temps par e-mail.



Figure : Mail regroupant l'emploi du temps du joueur

3.12 Supprimer tous les joueurs

En fin d'année, l'administrateur peut, s'il le souhaite, supprimer tous les joueurs, ce qui entraînera également la suppression de l'ensemble des sessions. Il lui suffit pour cela de cliquer sur le bouton « Supprimer tous les joueurs ». Une pop-up de confirmation s'ouvrira pour valider son choix. Une fois confirmé, tous les joueurs ainsi que leurs sessions seront définitivement supprimés, permettant ainsi de démarrer une nouvelle année.

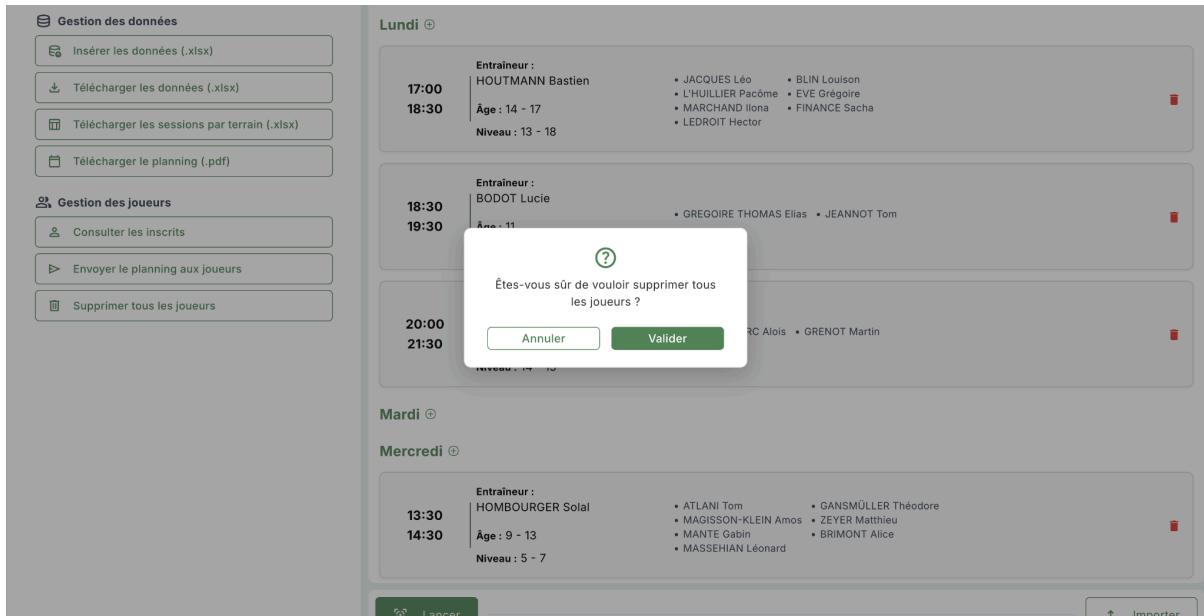


Figure : Pop-up de confirmation “Suppression des joueurs”

3.13 Page d'inscription

Mise en place de la page d'inscription pour les joueurs. L'emplacement de celle-ci n'est pas encore fixé (elle est actuellement présente sur l'écran d'authentification). Cette page permettra aux nouveaux joueurs de renseigner leurs informations personnelles afin de s'inscrire définitivement au Well Tennis Club. Le formulaire comporte plusieurs champs, vérifiés côté front-end, notamment la disponibilité minimale d'une heure, ainsi que la

condition selon laquelle les disponibilités doivent être supérieures au nombre de sessions souhaitées. Nous avons également défini plusieurs champs obligatoires, tels que le numéro, l'e-mail, le nom et le prénom. À la demande du client, nous avons ajouté une question concernant l'autorisation de prise de photo. Concernant les champs nom et prénom, il n'y a pas de vérification spécifique. Pour la date de naissance, l'âge possible est calculé selon l'année actuelle moins 120 ans ; par conséquent, une personne indiquant par exemple JJ:MM:1800 sera bloquée. Le nombre maximal de cours est fixé à 3. Pour le champ « disponibilité », à la suite des différents retours, nous avons décidé de mettre en place une liste déroulante plutôt que d'utiliser la roulette horaire par défaut, qui variait selon le téléphone utilisé. Cela permet d'avoir des horaires précis du type HH:00 ou HH:30 et facilite ainsi le choix pour l'utilisateur. Pour le champ e-mail, l'utilisateur doit respecter le format chainedecaractere@chainedecaractere.chainedecaractere. Le client nous a également demandé d'ajouter un second champ e-mail pour réduire le risque d'erreur, en empêchant le copier-coller entre les deux champs. Enfin, nous avons ajouté une question concernant l'autorisation de prise de photo. Nous avons indiqué par une étoile tous les champs obligatoires.

Figure : Page d'inscription

3.14 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.

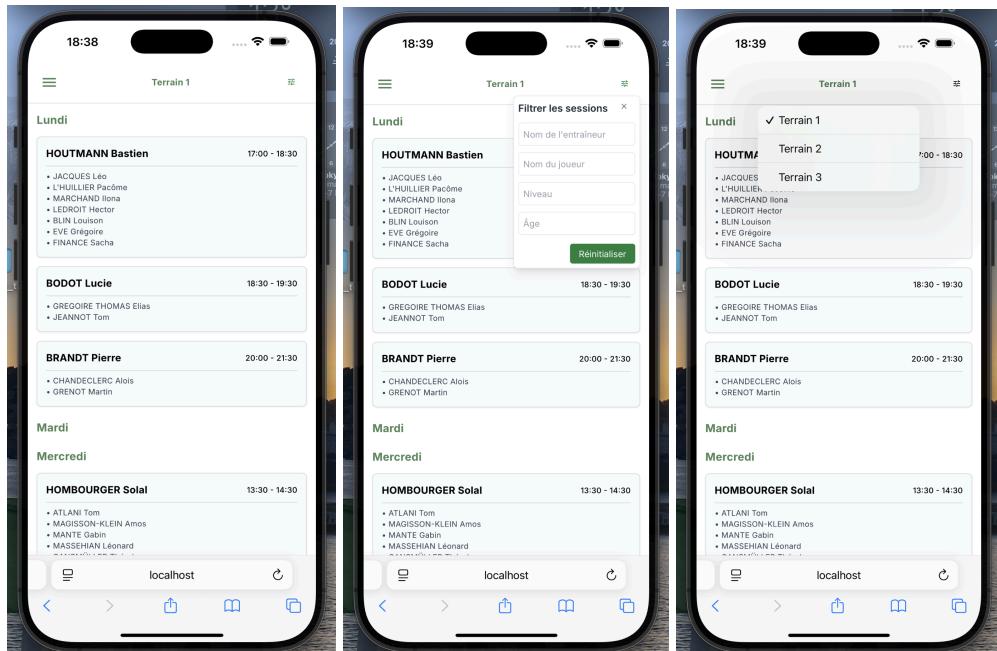
Figure : Affichage des inscrits / Détails de l'inscription

3.15 Affichage Mobile

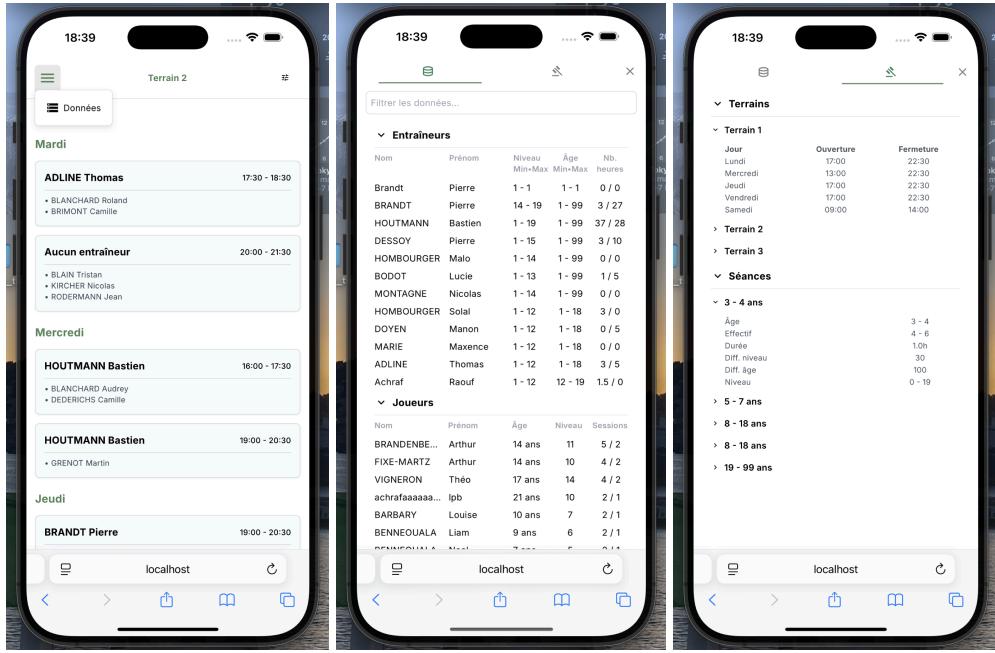
Afin de permettre aux utilisateurs d'accéder à la plateforme depuis un mobile, nous avons rendu celle-ci responsive. Cependant, nous avons réduit certaines fonctionnalités en mode mobile pour garantir une expérience plus fluide et adaptée à l'écran réduit.

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.



Page d'administrateur / entraîneur



Page d'administrateur / entraîneur

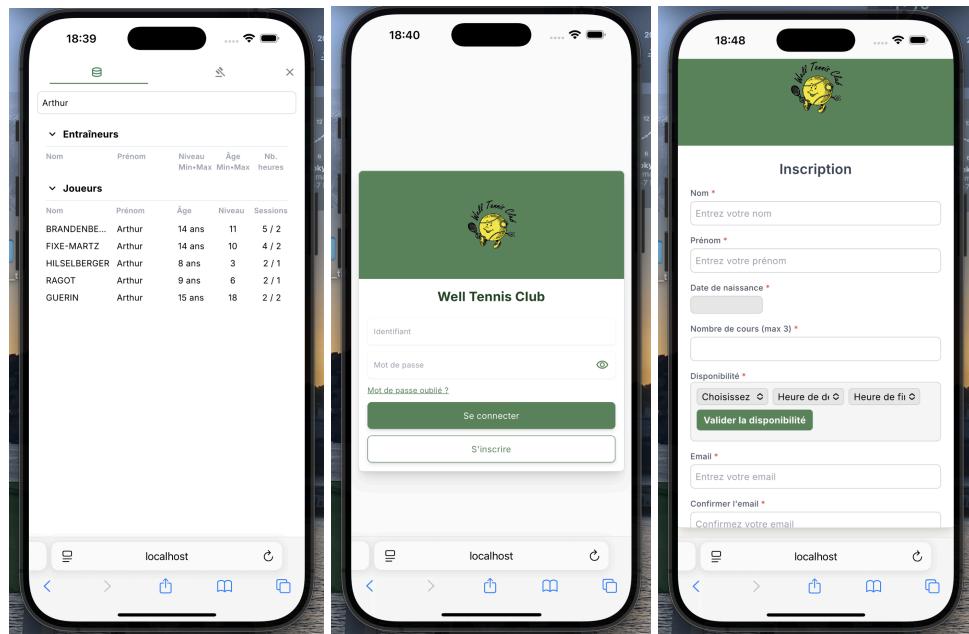


Figure : Affichage Mobile

3.16 Filtrage des données

Il est possible, grâce à une barre de recherche, de filtrer les joueurs et les entraîneurs par leur nom ou prénom, afin de les retrouver plus rapidement parmi les nombreuses données. Par défaut, la liste des joueurs est également filtrée selon le nombre de sessions : si un joueur a plus de sessions que prévu, celui-ci est affiché en haut de la liste afin que l'administrateur puisse gérer son cas.

Entraîneurs		Niveau	Âge	Nb. Min+Max	Min+Max	heures
Brandt	Pierre	1 - 1	1 - 1	0 / 0		
BRANDT	Pierre	14 - 19	1 - 99	3 / 27		
HOUTMANN	Bastien	1 - 19	1 - 99	37 / 28		
DESSOY	Pierre	1 - 15	1 - 99	3 / 10		
HOMBOURGER	Malo	1 - 14	1 - 99	0 / 0		
BODOT	Lucie	1 - 13	1 - 99	1 / 5		
MONTAGNE	Nicolas	1 - 14	1 - 99	0 / 0		
HOMBOURGER	Solal	1 - 12	1 - 18	3 / 0		
DOYEN	Manon	1 - 12	1 - 18	0 / 5		
MARIE	Maxence	1 - 12	1 - 18	0 / 0		
ADLINE	Thomas	1 - 12	1 - 18	3 / 5		
Achraf	Raouf	1 - 12	12 - 19	1.5 / 0		

Joueurs		Âge	Niveau	Sessions
BRANDENB...	Arthur	14 ans	11	5 / 2
FIXE-MARTZ	Arthur	14 ans	10	4 / 2
VIGNERON	Théo	17 ans	14	4 / 2
achrafaaaa...	Ipb	21 ans	10	2 / 1
BARBARY	Louise	10 ans	7	2 / 1
BENNEOUA...	Liam	9 ans	6	2 / 1
BENNEOUA...	Neal	7 ans	5	2 / 1
BRUMBT	Anais	99 ans	14	2 / 1

Figure : Filtrage des données

3.17 Possibilité de modifier les informations des joueurs et entraîneurs

L'administrateur peut modifier l'ensemble des données personnelles des entraîneurs et des joueurs. Il peut également ajouter manuellement des joueurs ou des entraîneurs, ainsi qu'en supprimer. De plus, l'administrateur peut créer des comptes administrateurs temporaires : pour cela, il crée un compte entraîneur auquel il attribue des droits d'administration, puis peut ensuite le supprimer. Concernant le mot de passe du compte entraîneur, une fois que l'administrateur a créé ce compte, l'entraîneur doit simplement utiliser la fonction « mot de passe oublié » ; il recevra alors un e-mail lui permettant de créer un nouveau mot de passe.

Modification du joueur

Nom	FIXE-MARTZ	Disponibilités (uniquement en période de 30 minutes)		
Prénom	Arthur	-	Mercredi	de 13:30 à 19:00
Email	9d6efcc2-b649-4186-9242-032e577801af@mail.fr	-	Samedi	de 09:00 à 13:30
Date de naissance	01/01/2011 soit 14 ans	[+]		
Temps de travail par semaine (en minutes)	N°1 0123456789 N°2 0123456789			
Niveau (0-30)	10			
Nombre de cours	2			
Autorise à être pris en photo	<input checked="" type="checkbox"/>			

BRUMBT Analys 99 ans 14 2 / 1 Lancer Importer

Figure : Pop-up modification joueur

Modifier l'Entraîneur

Nom	HOUTMANN	Disponibilités (uniquement en période de 30 minutes)		
Prénom	Bastien	-	Mardi	de 09:00 à 22:30
Email	5944496e-c048-45ea-8bd1-7075ae5d2754@mail.fr	-	Mercredi	de 09:00 à 22:30
Préférences de niveau	de 1 à 19	-	Jeudi	de 09:00 à 22:30
Préférences d'âge	de 1 à 99	-	Vendredi	de 09:00 à 22:30
Temps de travail par semaine (en minutes)	de 1680 à 1680	-	Samedi	de 09:00 à 16:00
Vacataire	<input checked="" type="checkbox"/>	Administrateur		

BRUMBT Analys 99 ans 14 2 / 1 Lancer Importer

Figure : Pop-up modification entraîneur

The screenshot shows a player creation form with the following fields:

- Disponibilités (uniquement en période de 30 minutes)**: A section with a '+' button.
- Nom**: Bran
- Prénom**: BRAI
- Email**: HOU
- Date de naissance**: DES: jj/mm/aaaa
- Temps de travail par semaine (en minutes)**: N°1 Téléphone 1, N°2 Téléphone 2
- Niveau (0-30)**: 1
- Nombre de cours**: 1
- Autorisé à être pris en photo**: Unchecked checkbox

At the bottom, there are buttons for **Sauvegarder**, **Lancer**, and **Importer**.

Figure : Pop-up création joueur

The screenshot shows a coach creation form with the following fields:

- Disponibilités (uniquement en période de 30 minutes)**: A section with a '+' button.
- Nom**: BRAI
- Prénom**: BRAI
- Email**: BRAI
- Préférences de niveau**: de 0 à 0
- Préférences d'âge**: de 0 à 0
- Temps de travail par semaine (en minutes)**: de minimum à maximum
- Vacataire**: Checked checkbox
- Administrateur**: Unchecked checkbox

At the bottom, there are buttons for **Sauvegarder**, **Lancer**, and **Importer**.

Figure : Pop-up création entraîneur

3.18 Connexion sécurisé

Depuis l'onglet de connexion, les entraîneurs peuvent accéder à leur compte coach afin de consulter leur emploi du temps. Ils disposent d'identifiants préalablement créés par un administrateur. Concernant leur mot de passe, il leur suffit d'effectuer une demande de « mot de passe oublié » ; ils recevront alors un e-mail leur permettant de le réinitialiser.

3.19 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.

3.20 Possibilité de personnalisation des sessions grâce au Drag and Drop

En effet, les sessions sont entièrement personnalisables. Comme mentionné précédemment, il est possible de créer de nouvelles sessions. Il est également possible de les supprimer à l'aide de l'icône de poubelle rouge présente sur chaque session. Chaque session peut être remplie ou modifiée grâce au système de **drag and drop** (glisser-déposer), en déplaçant un joueur ou un entraîneur depuis une autre session, ou en les sélectionnant depuis l'onglet d'affichage global des entraîneurs ou des joueurs. De la même manière, un joueur ou un entraîneur peut être retiré d'une session : lors du clic prolongé sur l'un de ces éléments, une icône de poubelle apparaît à l'écran. Il suffit alors de faire glisser l'élément dessus pour le supprimer de la session.

4. 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.

5. Difficultés rencontrées

Quelques difficultés ont été rencontrées concernant l'affichage sur ordinateur, notamment des problèmes lorsque la taille de l'écran est plus grande, ainsi que pour l'affichage des sessions. Cependant, ces problèmes ont été résolus lors de la dernière itération.

Nous avons également rencontré des problèmes lors de la création des emplois du temps en PDF. Nous avons dû tester plusieurs combinaisons afin que l'affichage soit le plus visuel possible, car dans certains cas, la liste des joueurs était trop longue et cassait l'affichage global de l'emploi du temps. Nous avons aussi rencontré des difficultés lors de la création des méthodes d'import et d'export des données.

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.

Nous avons également rencontré des problèmes concernant la transmission des mises à jour entre les composants Vue. En effet, certains conteneurs avaient été mal configurés, ce qui empêchait la mise à jour en temps réel des modifications effectuées via le **drag and drop**, ainsi que des ajouts et suppressions d'éléments.

Pour conclure, nous avons tout de même réussi à concevoir une interface visuellement agréable, claire et facile à prendre en main. Elle offre une expérience utilisateur fluide, ce qui était un objectif important pour nous. Sur ce point, nous avons même dépassé nos attentes initiales.

Sur le plan technique, l'interface repose principalement sur l'utilisation des liens API fournis par le back-end. Elle interagit efficacement avec celui-ci pour récupérer ou envoyer les données nécessaires. Toutefois, certaines vérifications et validations sont effectuées directement côté front-end, avant la transmission des informations au back-end — par exemple, lors de l'import d'un fichier Excel, les données sont d'abord récupérées et contrôlées afin de s'assurer de leur validité avant de les envoyer au serveur. Cela permet d'éviter des requêtes inutiles, mais aussi d'améliorer la réactivité de l'application tout en renforçant la cohérence et la fiabilité des données.

Déploiement sur le serveur

Pour le déploiement de l'application on utilise Docker. On a un docker compose contenant 3 containers. Le premier est le plus simple, il s'agit de la base de données, on utilise l'image PostgreSQL officiel. On a ensuite un container pour le backend, l'API Rest. Ce dernier container a bien évolué au cours des dernières itérations. On crée notre propre image grâce à un Dockerfile. Initialement cette image faisait 300 MB, mais on a diminué sa taille en ne mettant dans le container uniquement ce qu'on utilise afin de réduire les failles de sécurité. Lors de notre dernière version, cette image du backend fait 200 MB. Pour le troisième et dernier container, on a de nouveau créé notre propre image du frontend grâce à un Dockerfile. Pour les mêmes raisons de sécurité on a diminué sa taille initiale. La version finale de cette dernière image est de 330 MB alors qu'elle faisait 1.5 GB initialement. Une fois ces trois containers lancés on enregistre les images en ".tar" grâce à la commande "docker save". On envoie ensuite les images sur le serveur grâce à la commande "scp". On se connecte au serveur en ssh. Une fois les images docker envoyées sur le serveur en ".tar" il nous faut les extraire, on utilise donc "docker load". Maintenant l'application est presque déployée, il nous faut adapter le docker compose. On reprend celui qu'on lance pour faire les différentes images en modifiant les références aux images. On lui dit d'utiliser les images que le serveur a reçu de nos machines. Il faut ensuite faire un fichier ".env" dans le dossier "./serveur/.env". On peut enfin faire "docker compose up --build -d" qui nous permet de lancer l'application sur Docker et les 3 containers sur le serveur.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ra-il2-2024-ketzinger-micheli-pedron-raouf-frontend	latest	2b106b14608d	3 days ago	332MB
ra-il2-2024-ketzinger-micheli-pedron-raouf-app	latest	8a8e8a541678	3 days ago	236MB
postgres	13	863651115628	5 weeks ago	444MB

Figure : Images après envoi sur le serveur.

Conclusion

À la fin de la rédaction de ce rapport, nous pouvons considérer que le projet est terminé et abouti. En effet, nous avons globalement suivi notre étude préalable. Il y a eu, bien sûr, quelques ajustements, mais nous avons respecté les délais et livré une application concrète et finalisée à notre client.

Durant la seconde partie de l'année, nous avons régulièrement présenté un rendu fonctionnel accompagné d'une présentation de qualité. Nous pouvons être fiers de ce résultat.

La principale complexité du projet résidait dans l'algorithme de planification de l'emploi du temps. Nous pouvons affirmer qu'il est fonctionnel et correctement implémenté dans l'application. De plus, il est modulable par le client selon ses besoins. La demande principale a donc été respectée et est pleinement opérationnelle. Le seul bémol reste le nombre limité de cas sur lesquels l'algorithme a été testé.

Malgré cela, l'application gravitant autour de cet algorithme est achevée. Elle répond entièrement aux exigences du client et reste simple d'utilisation.

À terme, des optimisations auraient pu être envisagées, notamment au niveau des interactions avec la base de données ainsi que de la gestion des vues, qui aurait pu être implantée de manière plus efficace. Certaines parties du code mériteraient également d'être refactorisées.

Le manque de temps nous a empêchés d'apporter ces améliorations. Elles pourront peut-être être mises en œuvre ultérieurement par le groupe, à titre personnel, si le projet devait être réutilisé pour d'autres clients.

Nous ne pensons pas que le projet puisse être repris par un groupe d'étudiants l'année prochaine. Bien qu'il soit perfectible, les améliorations restantes ne présentent pas un défi technique suffisant pour en faire un projet formateur et intéressant pour des étudiants. Il s'agit principalement d'optimiser les requêtes, et éventuellement d'améliorer le solveur. En résumé, ces tâches sont relativement simples mais chronophages.