RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE

BP 526 Cotonou Tel : +229 21 14 19 88
http://www.ifri-uac.net Courriel : contact@ifri.uac.bj

MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Génie Logiciel

Présenté par :
Ayékochikpa Landry Elvis **ETEKA**

Conception d'une application web de gestion et de suivi des abonnements des clients dans une salle de gym : Cas Russian Gym Fitness Club

Sous la supervision :
Fidéle **HOUNSOUNON**

Membres du jury :

SANDA MAHAMA Tidjani A. (MA)	Docteur	IFRI/UAC	Président
GNONLONFOUN Miranda	Ingénieur	IFRI/UAC	Examinateur
HOUNSOUNON Fidéle	Ingénieur	IFRI/UAC	Rapporteur

Année Académique : **2024-2025**

Sommaire

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

List of Figures vi

List of Tables vii

Liste des acronymes viii

Introduction 1

1 Revue de la littérature 3

2 Analyse, Conception et Choix technique 9

3 Résultats de l'application 21

Conclusion 34

Bibliographie 35

Webographie 36

Table des matières 37

Dédicace

Je dédie ce travail à :

- Mes chers parents, pour leur amour inconditionnel, leur soutien indéfectible et leurs encouragements tout au long de mon parcours académique ;
- Mes frères et sœurs, pour leur présence et leur affection ;
- Tous ceux qui me sont chers et proches.

Remerciements

Je tiens à exprimer ma profonde gratitude à l'Éternel, Dieu tout-puissant, pour sa présence, son amour infini, sa protection constante et l'inspiration qu'Il ne cesse d'insuffler dans ma vie.

Je remercie particulièrement :

- Mon maître de mémoire, **M. Fidèle Hounsounoun**, pour son accompagnement, sa disponibilité et ses précieux conseils tout au long de ce travail ;
- Mon maître de stage, **M. Nourou Dine**, pour ses orientations et ses recommandations avisées lors de mon stage académique ;
- Tous les professeurs de **l'Institut de Formation et de Recherche en Informatique (IFRI)**, pour leur encadrement, leur dévouement et leur expertise, qui ont été essentiels dans mon parcours académique ;
- Mes camarades de classe, pour leur soutien, leur collaboration et les échanges enrichissants qui ont marqué notre expérience d'apprentissage commune.

Résumé

De nombreuses salles de sport, comme le Russian Gym Fitness Club, rencontrent des difficultés dans la gestion efficace des abonnements de leurs clients en raison de l'utilisation de méthodes traditionnelles, telles que l'enregistrement manuel sur papier ou l'usage de tableurs Excel. Bien que fonctionnelles à court terme, ces pratiques engendrent des inefficacités et augmentent le risque d'erreurs humaines. Dans ce projet, nous avons conçu et développé une application web dédiée à la gestion et au suivi des abonnements des clients du Russian Gym Fitness Club. Cette application repose sur le langage de modélisation UML pour sa conception, ainsi que sur les frameworks Spring Boot, Tailwind CSS et React JS pour son implémentation. Afin d'assurer la persistance des données et optimiser la gestion du système, une base de données MySQL a été mise en place.

Mots-clés : Gestion des abonnements, Russian Gym Fitness Club, Méthodes traditionnelles, Application web, UML, Spring Boot, React JS, MySQL.

Abstract

Many gyms, such as the Russian Gym Fitness Club, struggle to efficiently manage their clients' subscriptions due to the use of traditional methods, such as manual record-keeping on paper or via Excel spreadsheets. These practices, although effective in the short term, lead to inefficiencies and can result in human errors. In this project, we proposed the design of a web application for managing and tracking gym subscriptions, using the case of the Russian Gym Fitness Club. The implementation and development of this web application were made possible through the UML modeling language, as well as the Spring Boot, Tailwind CSS and React JS frameworks. To enhance the system and ensure data persistence, a database was designed using MySQL.

Key words: Subscription management, Russian Gym Fitness Club, Traditional methods, Web Application, UML, Spring Boot, React JS, MySQL.

List of Figures

1.1 Resamania 4
1.2 Fliip 5
1.3 Virtuagym 5

2.1 Logo du club de fitness Russian Gym 9
2.2 Diagramme des cas d'utilisation du système de gestion du Russian Gym 12
2.3 Diagramme de classes 13
2.4 Diagramme de séquence pour l'authentification 14
2.5 Diagramme de séquence pour la souscription à un abonnement 15

3.1 Notre page d'accueil 21
3.2 Page de connexion de l'application 22
3.3 Page d'inscription de l'application 23
3.4 Interface adhérent – Tableau de bord 24
3.5 Interface de souscription aux abonnements 24
3.6 Page de confirmation après souscription 25
3.7 Interface permettant à l'adhérent de choisir son objectif 25
3.8 Interface de consultation des plannings créés par le coach, accessibles en fonction de la disponibilité de l'adhérent 26
3.9 Interface du chat de discussion entre coachs et adhérents 26
3.10 Interface secrétaire - Tableau de bord 27
3.11 Interface Coach - Tableau de bord 27
3.12 Interface de création d'une séance par un coach 28
3.13 Interface d'affichage des séances créées par un coach 28
3.14 Interface de création d'une activité par un coach 29
3.15 Interface d'affichage des activités créées par un coach 29
3.16 Interface administrateur - Tableau de bord des statistiques 30

List of Tables

1.1 Tableau récapulatif des solutions existantes pour une gestion de salle de sport 7

3.1 Résumé des tests unitaires effectués 31

Liste des acronymes

ACID :

Atomicité, Cohérence, Isolation, Durabilité(ACID) . 18

API :

Application Programming Interface(API) . 31

FNP :

Fichiers Numériques Personnalisés (FNP). 33

IDE :

Integrated Development Environment(IDE) . 19, 20

JSX :

JavaScript XML (JSX) . 17

JWT :

JSON Web Token(JWT) . 20

MD5 :

Message-Digest Algorithm 5(MD5) . 20

MERISE :

Méthode d'Étude et de Réalisation Informatique par Sous Ensemble(MERISE). 16

MVC :

Modèle-Vue-Contrôleur (MVC) . 16

REST :

Representational State Transfer(REST) . 18

RFID :

Radio Frequency Identification (RFID). 33

SGBDR :

Système de Gestion de Base de Données Relationnelle(SGBDR) . 18

SQL :

Système de Gestion de Base de Données Relationnelle(SQL) . 18

UML :

Unified Modeling Language, ou Langage de Modélisation Unifié (UML). 9, 11, 13, 16, 20, 38

XML :

eXtensible Markup Language (XML) . 17

Introduction Générale

Contexte

Dans un contexte où la transformation numérique est un levier stratégique pour l'optimisation des processus organisationnels, les établissements sportifs doivent moderniser leurs outils de gestion afin d'améliorer leur efficacité et la satisfaction de leurs adhérents. Le Russian Gym Fitness Club rencontre une difficulté majeure dans la gestion des abonnements de ses membres, notamment pour distinguer les abonnements expirés et l'accès à la salle de sport. Le système manuel actuellement utilisé pour suivre la validité des abonnements présente des lacunes, entraînant des erreurs humaines et une gestion inefficace. Les membres dont l'abonnement est expiré ou en cours de renouvellement peuvent rencontrer des complications pour accéder aux installations, ce qui génère de la frustration et impacte négativement l'expérience client.

Cette gestion inefficace complique également les opérations internes du club. Les employés doivent constamment vérifier les informations des membres, ce qui engendre une perte de temps et augmente les risques d'erreurs. De plus, l'absence d'un système automatisé empêche le club d'avoir une vue d'ensemble claire sur les membres actifs, compliquant ainsi la prise de décisions stratégiques.

Pour résoudre ces problèmes, le Russian Gym envisage de mettre en place un système informatisé afin de rationaliser le suivi des abonnements, d'alerter automatiquement sur les expirations et d'optimiser la gestion interne, garantissant ainsi une meilleure expérience pour les membres et le personnel.

Problématique

Dans l'industrie dynamique de la remise en forme, la gestion des abonnements est un enjeu crucial pour garantir une expérience fluide et efficace aux membres. Actuellement, le Russian Gym Fitness Club rencontre plusieurs difficultés, notamment dans le suivi des inscriptions et des renouvellements d'abonnement de sa clientèle.

C'est dans ce contexte qu'est née l'idée de concevoir une application web de gestion et de suivi des abonnements, destinée à faciliter le suivi des abonnés et à optimiser l'administration du centre.

Objectifs

L'objectif principal de ce projet est de concevoir une application web pour la gestion et le suivi des abonnements du Russian Gym Fitness Club. Plus précisément, notre solution permettra :

- de choisir un type d'abonnement ;
- de gérer les plannings ;
- la communication entre les abonnés et les coachs.

Organisation du mémoire

Le présent document est structuré en trois chapitres. Le premier chapitre est une revue de la littérature où les concepts clés liés à notre projet seront présentés. Dans le deuxième chapitre, la conception de la solution ainsi que les choix techniques adoptés pour sa mise en œuvre seront détaillés. Enfin, le troisième chapitre présentera un prototype de notre application et proposera une discussion sur les résultats obtenus.

Chapitre 1

Revue de la littérature

Introduction

Avant de faire la lumière sur les solutions existantes, nous allons aborder la partie clarification conceptuelle en définissant quelques notions qui sont rentrées en ligne de compte dans la réalisation du travail.

1.1 Clarification Conceptuelle

1.1.1 Une application web

Une application web, ou **web application** en anglais, est une plateforme accessible via un **navigateur** sans installation locale, contrairement aux **applications mobiles**. Elle ne nécessite donc pas d'installation sur les **machines clientes**. À l'instar des **sites web**, une application web est généralement hébergée sur un serveur et s'utilise à l'aide d'un navigateur web, en manipulant des widgets via un réseau informatique (Internet, intranet, réseau local, etc.).[1]

1.1.2 Gestion des abonnements

La gestion des abonnements est un système complet qui permet aux entreprises de suivre et de gérer toutes les activités liées à leurs offres d'abonnement. Elle englobe l'ensemble des processus nécessaires pour administrer efficacement les relations avec les clients abonnés, depuis la création des plans jusqu'à la facturation et au suivi de l'utilisation [2].

1.1.3 Suivi des abonnements

Le suivi des abonnements est un processus systématique qui permet aux entreprises de surveiller et d'analyser toutes les activités liées à leurs clients abonnés, depuis leur inscription jusqu'à la fin de leur relation commerciale. Ce système assure une gestion efficace de la relation client tout en maintenant une visibilité complète sur l'utilisation des services et le chiffre d'affaires généré [3].

1.1.4 Salle de gym

Une salle de gym est un établissement sportif spécialement équipé pour la pratique d'exercices physiques et le développement musculaire. Elle offre un espace dédié où les membres peuvent utiliser divers appareils et équipements pour améliorer leur condition physique et leur santé globale [4].

1.2 Étude des solutions existantes

Dans cette section, nous analysons plusieurs solutions existantes pour la gestion des abonnements des membres dans une salle de gym. Après une étude approfondie, nous avons sélectionné trois applications web : Resamania, FLiiP et Virtuagym .

1.2.1 Resamania



FIGURE 1.1 : Resamania

Resamania, accessible via l'URL https://www.resamania.fr/, est une application web de gestion pour les salles de sport et les clubs de fitness. Elle permet de gérer les abonnements, les paiements, les réservations de cours et le contrôle d'accès des membres. Elle propose également des outils de communication pour fidéliser les adhérents ainsi qu'un module de reporting pour analyser les performances de la salle. Actuellement, elle ne dispose pas d'une application mobile dédiée pour les adhérents. Resamania est principalement disponible en France, où elle est bien implantée et utilisée par de nombreuses salles de sport.

1.2.2 FLiiP



FIGURE 1.2 : Fliip

FLiiP, accessible via l'URL https://myfliip.com/fr/software/gyms-traditionnels/, est une application web conçue pour les studios de fitness, avec une application mobile permettant aux membres et aux coachs d'accéder aux fonctionnalités principales. Elle automatise la gestion des abonnements, des paiements et des réservations de cours, tout en intégrant des outils de marketing et de notifications pour améliorer la fidélisation. Son interface intuitive facilite la gestion des activités sans effort administratif important. FLiiP est disponible en France et se positionne comme une solution adaptée aux studios de fitness français.

1.2.3 Virtuagym

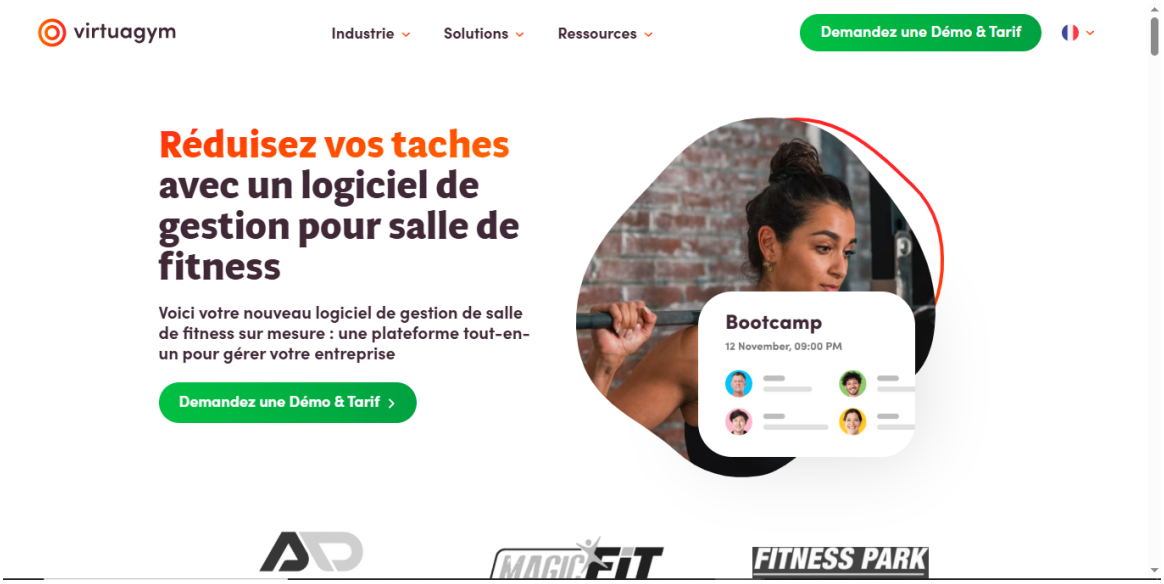


FIGURE 1.3 : Virtuagym

Virtuagym, accessible via l'URL https://business.virtuagym.com/fr/logiciel-gestion-salle-de-fitness/, est une application web complète pour la gestion des clubs de fitness, des entraîneurs personnels et des studios, avec une application mobile dédiée aux membres et aux coachs. Elle permet de créer des programmes d'entraînement et de nutrition personnalisés, de gérer les abonnements et le suivi des membres. Son application mobile renforce l'engagement des utilisateurs en leur offrant un suivi détaillé et une communication fluide avec les coachs. Virtuagym, d'origine néerlandaise, est disponible en France et a même établi un partenariat avec Resamania pour proposer une solution intégrée aux salles de sport françaises.

1.2.4 Tableau récapitulatif

Ce tableau nous permet d'avoir une vue globale sur les fonctionnalités et les insuffisances de chaque solution existante.

TABLE 1.1 : Tableau récapulatif des solutions existantes pour une gestion de salle de sport

Application	**Fonctionnalités**	**Insuffisances**
Resamania	- Gestion des abonnements et des paiements - Réservations de cours et gestion des plannings - Système de contrôle d'accès intégré - Outils de communication pour fidélisation des membres - Reporting et statistiques sur la fréquentation et les revenus - Intégration avec des terminaux de paiement	- Pas d'application mobile dédiée - Manque d'outils de création de programmes d'entraînement et de nutrition - Interface utilisateur peut sembler complexe pour les débutants - Peu de personnalisation possible pour les rapports - Pas d'intégration avec des systèmes tiers (comme des outils de CRM externes)
FLiiP	- Automatisation des abonnements, paiements, et réservations de cours - Outils de marketing intégrés pour campagnes de fidélisation et notifications - Application mobile pour les membres et les coachs - Suivi des performances des membres - Outils de gestion des ressources humaines (coachings, horaires) - Interface simple et intuitive	- Manque de fonctionnalités avancées pour les programmes d'entraînement personnalisés - Pas de système de contrôle d'accès - Pas de reporting avancé sur les finances et la fréquentation - Pas d'intégration avec des outils externes comme des systèmes de paiement tiers
Virtuagym	- Création de programmes d'entraînement et de nutrition personnalisés - Gestion complète des membres et abonnements - Outils de suivi des progrès (poids, objectifs, performance) - Application mobile pour les membres et les coachs - Réservation de cours et gestion des plannings - Reporting avancé et statistiques détaillées - Intégration avec des dispositifs externes (smartwatches, appareils de suivi)	- Pas de système de contrôle d'accès intégré - Outils de communication et de marketing moins développés comparé à d'autres solutions - Interface utilisateur pas toujours intuitive - Manque d'outils de gestion de la comptabilité et des finances - Pas de support direct pour les petites salles sans équipe dédiée

Conclusion

Ce chapitre a permis d'introduire les concepts clés liés à notre projet, notamment les applications web, la gestion et le suivi des abonnements, ainsi que le fonctionnement des salles de gym. Nous avons également analysé les solutions existantes sur le marché, en mettant en évidence leurs fonctionnalités et leurs limites à travers une comparaison détaillée. Cette étude nous a permis de mieux comprendre les besoins spécifiques des salles de sport en matière de gestion des abonnements et de planification. Elle servira de base pour la conception d'une solution adaptée aux attentes du Russian Gym Fitness Club, en tenant compte des forces et des faiblesses des outils existants. Le chapitre suivant abordera notre méthodologie de conception ainsi que les choix techniques effectués.

Chapitre 2

Analyse, Conception et Choix technique

Introduction

Ce chapitre présente l'approche méthodologique adoptée pour la réalisation de la solution. Il est structuré en trois grandes parties : l'analyse du problème, la conception du système avec une modélisation en Unified Modeling Language, ou Langage de Modélisation Unifié (UML), et le choix des outils et technologies. L'objectif est d'aboutir à une solution répondant aux besoins identifiés, tout en garantissant une mise en œuvre efficace et adaptée.



FIGURE 2.1 : Logo du club de fitness Russian Gym

2.1 Analyse des besoins

La solution que nous devons développer doit répondre à des besoins spécifiques identifiés afin de simplifier le suivi des abonnements des clients dans la salle de gym. Cette section présente les besoins

fonctionnels et non fonctionnels du projet.

2.1.1 Les besoins fonctionnels

Les besoins fonctionnels décrivent les fonctionnalités que le système doit offrir pour accomplir les tâches attendues.

2.1.1.1 Suivi des abonnements des clients

Le système permet d'assurer un suivi général des abonnements des clients dans la salle de gym.

2.1.1.2 Fonctionnalités offertes aux clients

Le client pourra :

- Créer un compte
- Modifier son profil
- Choisir son type d'abonnement
- Consulter les plannings
- Effectuer un réabonnement
- Communiquer avec un coach
- Consulter les notifications

2.1.1.3 Fonctionnalités réservées à la secrétaire

La secrétaire pourra :

- Gérer les notifications
- Gérer les paiements
- Gérer les abonnements

2.1.1.4 Fonctionnalités réservées au coach

Le coach pourra :

- Gérer les plannings
- Communiquer avec les clients
- Consulter les notifications
- Modifier son profil

2.1.1.5 Fonctionnalités réservées à l'administrateur

L'administrateur pourra :

- Gérer les utilisateurs
- Consulter les statistiques

2.1.2 Les besoins non fonctionnels

Les besoins non fonctionnels définissent les critères de qualité et les contraintes techniques que le système doit respecter. Notre solution devra ainsi garantir :

- **Optimisation responsive :** Interface fluide et intuitive pour les utilisateurs.
- **Sécurisation des données :** Garantir la confidentialité et l'intégrité des informations.
- **Performance et évolutivité :** Assurer une réponse rapide de l'application, même avec un volume important de données.
- **Préparation pour intégrer facilement d'autres modules :** Permettre l'ajout de nouvelles fonctionnalités sans impact majeur sur le système existant.
- **Support multilingue (anglais/français) :** Interface disponible en plusieurs langues pour s'adapter à différents utilisateurs.

2.2 Conception

Pour modéliser efficacement le fonctionnement global de l'application, nous avons opté pour une approche UML. Cette méthode nous permet de mieux appréhender les interactions et les structures du système à travers différents diagrammes, notamment le diagramme des cas d'utilisation, le diagramme de classes et le diagramme de séquence.

2.2.1 Modélisation avec UML

Notre application web intègre un large éventail de fonctionnalités complexes, nécessitant une structuration et une modélisation rigoureuses. UMLs'est imposé comme le choix idéal grâce à sa capacité à modéliser clairement les fonctionnalités et interactions du système, à soutenir une architecture modulaire via des diagrammes de classes, et à gérer des processus dynamiques avec les diagrammes de séquence. Il facilite également la collaboration en équipe en offrant un langage commun et produit une documentation utile pour la maintenance et l'évolution du projet. UML a ainsi permis de lier efficacement les besoins métiers aux solutions techniques, tout en respectant les objectifs d'ergonomie et de robustesse.

2.2.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation constitue la première étape de l'analyse en UML, résumant la manière dont les utilisateurs interagissent avec le système. Il permet de recueillir, d'analyser et d'organiser les besoins, tout en recensant les grandes fonctionnalités du système. Grâce au diagramme de cas d'utilisation, nous obtenons une vue globale du comportement fonctionnel du système. Le diagramme de cas d'utilisation relatif à notre domaine d'étude se présente comme suit :

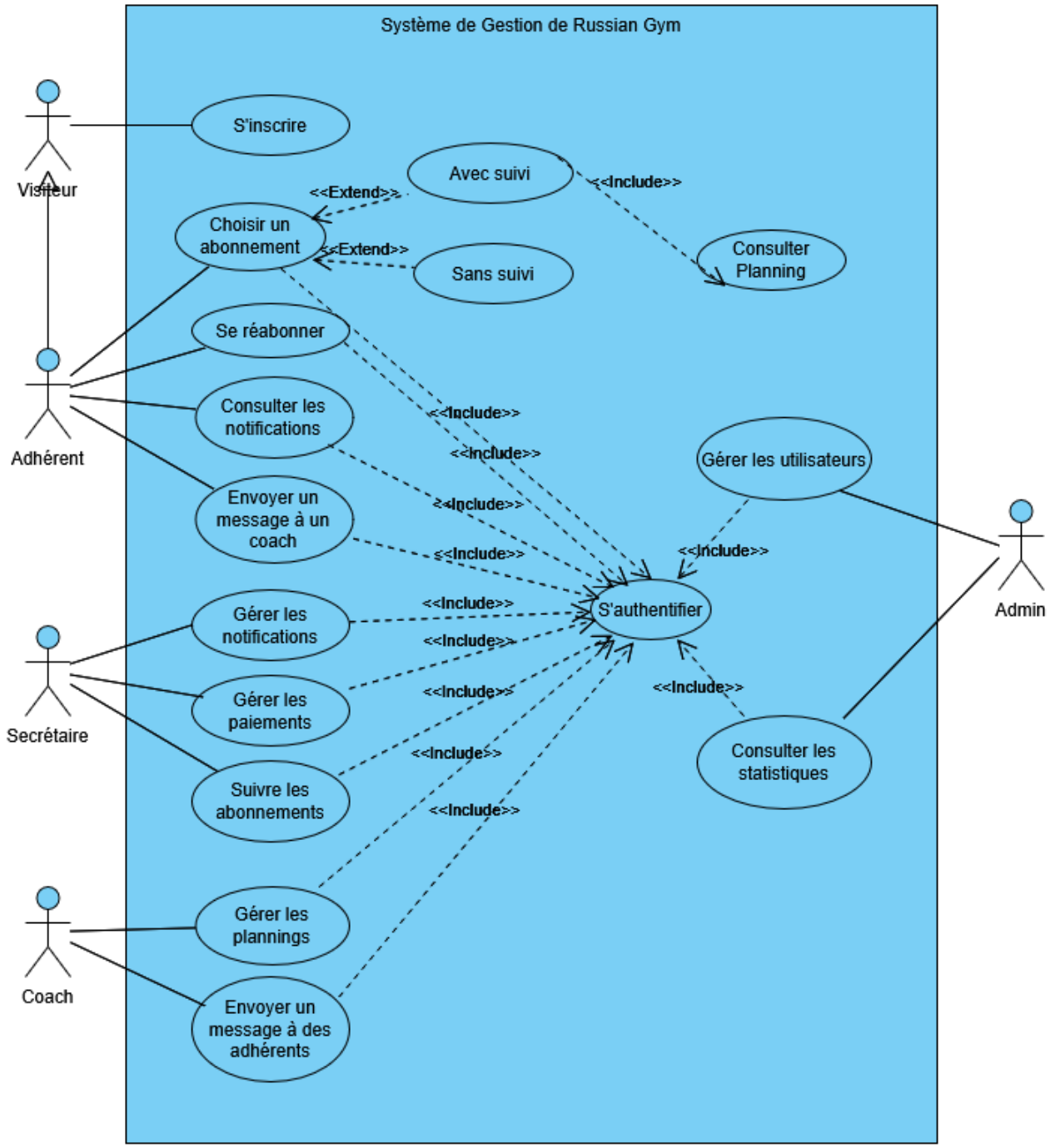


FIGURE 2.2 : Diagramme des cas d'utilisation du système de gestion du Russian Gym

Ce diagramme de cas d'utilisation illustre les interactions entre les différents acteurs du système de gestion Russian Gym : le Visiteur, l'Adhérent, la Secrétaire, le Coach et l'Administrateur. Un Visiteur peut s'inscrire en créant un compte et en choisissant un abonnement, avec ou sans suivi. Une fois inscrit, l'Adhérent peut s'authentifier pour accéder à son espace personnel, consulter son planning, se réabonner, gérer ses notifications et échanger avec un coach. Le Secrétaire est chargé de gérer les abonnements, les paiements et les notifications des adhérents. Le Coach peut, quant à lui, gérer les plannings et échanger avec les adhérents. Enfin, l'Administrateur a pour mission de gérer les utilisateurs et de consulter les statistiques du système. L'authentification est obligatoire pour accéder aux fonctionnalités du système. Ce modèle assure une gestion efficace des abonnements et une communication fluide entre les différents acteurs.

2.2.3 Diagramme de classes

Le diagramme de classes est un élément clé de la modélisation UML, représentant la structure statique d'un système. Il décrit les différentes classes qui composent l'application, leurs attributs et leurs méthodes, ainsi que les relations entre elles.

- **Classe** : Il s'agit d'une description abstraite d'un ensemble d'objets partageant des attributs et des comportements similaires.
- **Attributs** : Ce sont les propriétés ou caractéristiques d'une classe. Ils définissent l'état de chaque instance de la classe.
- **Méthodes** : Ce sont les fonctions ou opérations définies au sein d'une classe. Elles représentent les comportements que les objets de cette classe peuvent effectuer.

Notre diagramme de classes est présenté ci-dessous, illustrant la structure des principales entités de l'application ainsi que leurs attributs et leurs méthodes :

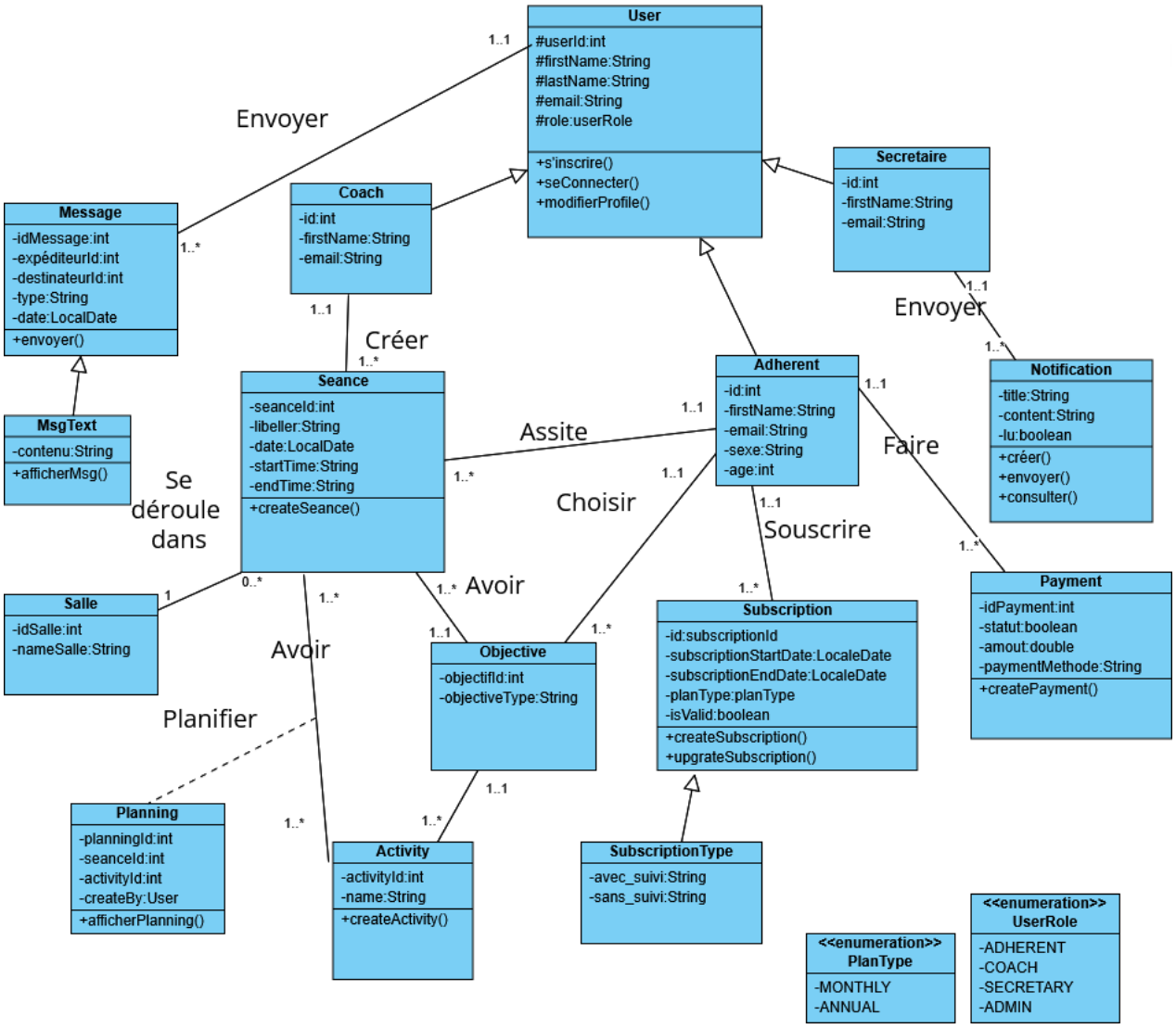


FIGURE 2.3 : Diagramme de classes

2.2.4 Diagramme de séquence

Les diagrammes de séquence sont une représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique.

2.2.5 Diagramme de séquence pour l'authentification

La figure ci-dessous présente le diagramme de séquence pour l'authentification sur l'application.

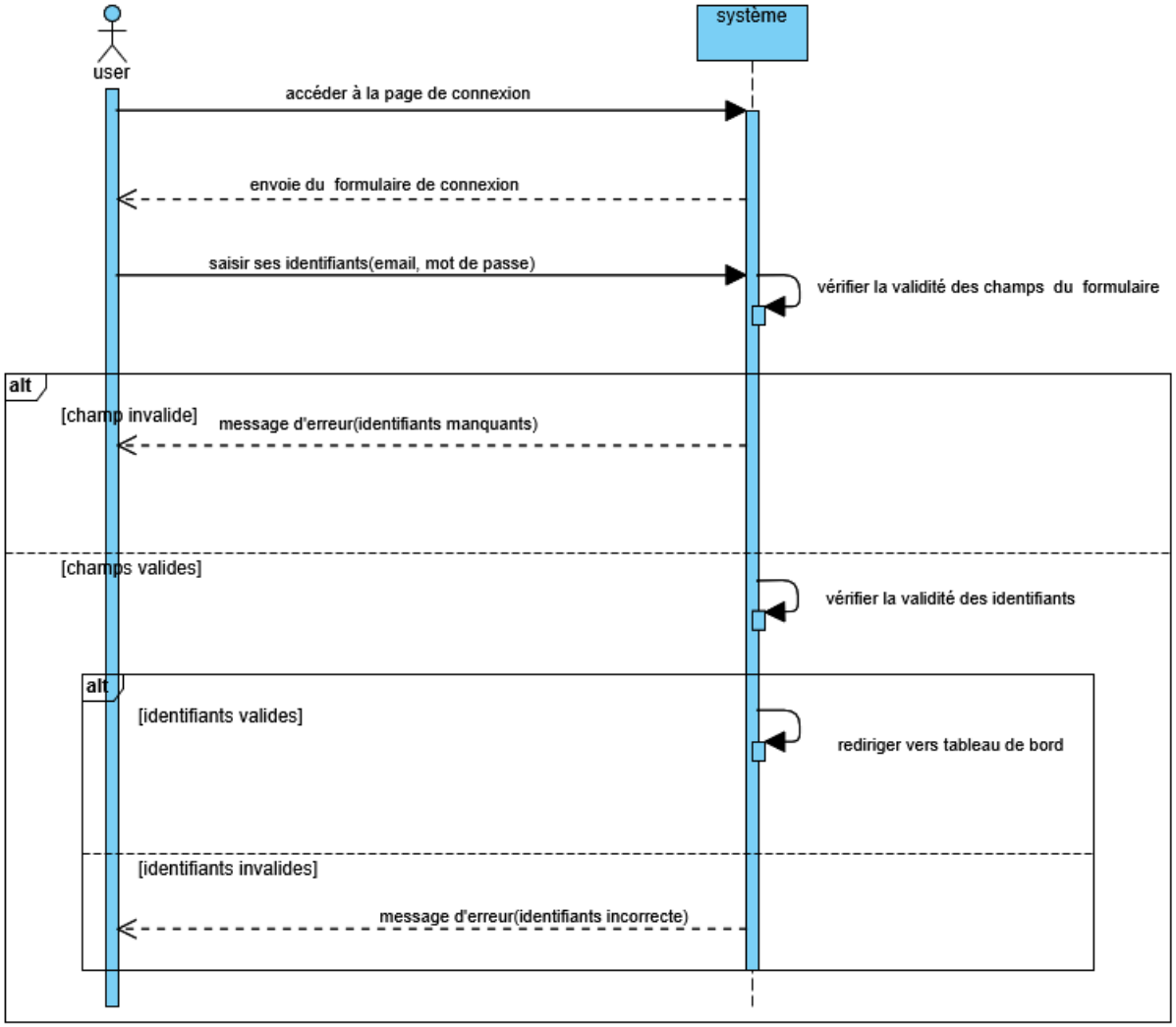


FIGURE 2.4 : Diagramme de séquence pour l'authentification

2.2.6 Diagramme de séquence pour la souscription à un abonnement

À travers ce diagramme de séquence, nous illustrons comment un utilisateur peut souscrire à un abonnement.

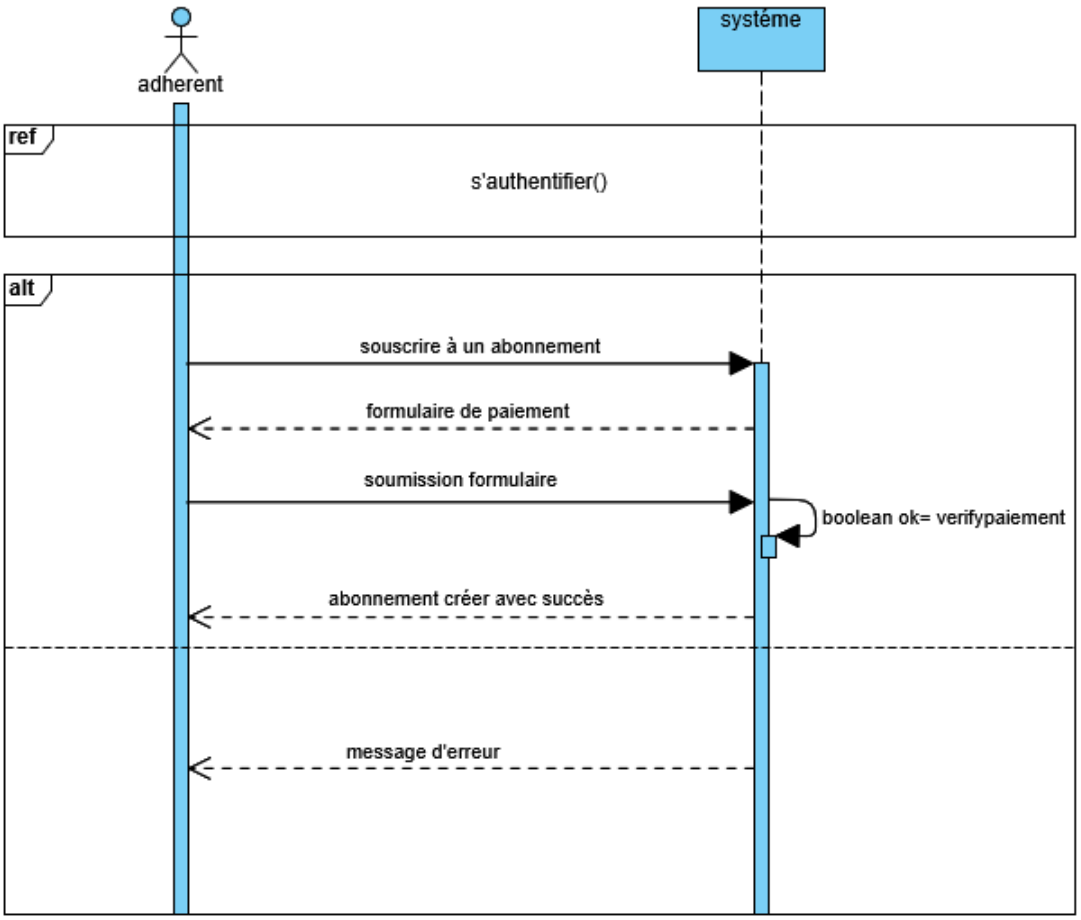


FIGURE 2.5 : Diagramme de séquence pour la souscription à un abonnement

2.3 Choix techniques

2.3.1 Choix du langage de modélisation

La modélisation des données est une documentation des processus et des événements qui se produisent lors de la conception et du développement des applications. Les langages de modélisation des données permettent de représenter des systèmes complexes sous une forme simplifiée, en décrivant les processus et les flux de données, afin de créer un modèle facilitant la construction et la réingénierie. Ils sont définis par un ensemble cohérent de règles utilisées pour interpréter la signification des différents composants. Il existe plusieurs langages de modélisation des données.

MERISE (Méthode d'Étude et de Réalisation Informatique par Sous-Ensemble) est une méthode de conception et de développement de systèmes d'information. Son objectif est de concevoir un système d'information efficace et structuré. Elle repose sur la séparation des données et des traitements en plusieurs modèles conceptuels et physiques. Cette séparation garantit la pérennité du modèle : l'organisation des données reste stable à long terme, tandis que les traitements peuvent être modifiés plus fréquemment[5].

UML (Unified Modeling Language) est un langage de description orienté objet permettant de modéliser une application selon une approche objet. Un objet est défini par ses attributs et les trai-

tements, appelés méthodes, qui peuvent lui être appliqués. UML repose sur un ensemble d'annotations graphiques basées sur des diagrammes, permettant ainsi de spécifier, visualiser et documenter les systèmes logiciels orientés objet[1].

Notre choix s'est porté sur UML, car il s'adapte particulièrement aux projets de développement logiciel modernes. Contrairement à MERISE, principalement utilisée pour la conception des bases de données et la modélisation des systèmes d'information traditionnels, UML offre une approche plus flexible et évolutive.

Grâce à ses différents types de diagrammes (cas d'utilisation, classes, séquences, etc.), UML fournit une vision globale et détaillée du système en cours de développement. Cette flexibilité est essentielle pour notre projet, qui nécessite une structuration claire des interactions entre les différents composants logiciels ainsi qu'une bonne visualisation des flux de données. De plus, UML facilite la communication entre les développeurs et les autres parties prenantes du projet, assurant ainsi une meilleure compréhension des spécifications et des exigences fonctionnelles.

2.3.2 Architecture du Système : Modèle MVC

Notre application web adopte une architecture MVC (Model-View-Controller), un choix stratégique permettant de structurer ses composants de manière modulaire et efficace. Cette approche a été retenue car elle facilite la séparation des préoccupations, améliorant ainsi la lisibilité, la maintenabilité et l'évolutivité du code. En optant pour MVC, nous pouvons également assurer une meilleure réutilisation des composants et une gestion optimisée des interactions entre le front-end et le back-end.

L'architecture se compose de trois couches distinctes, chacune ayant un rôle spécifique dans le bon fonctionnement de l'application :

- La couche Modèle (Model) gère les données et la logique métier de l'application. Nous avons choisi MySQL comme base de données pour sa robustesse et sa capacité à gérer des transactions complexes. Cette couche garantit l'intégrité des données et fournit des interfaces facilitant les interactions avec le back-end.
- La couche Vue (View), développée avec React.js, a été choisie pour sa flexibilité et sa capacité à créer des interfaces réactives et performantes. React.js permet un rendu dynamique et optimise l'expérience utilisateur grâce à son approche basée sur le Virtual DOM.
- Le Contrôleur (Controller), implémenté avec Spring Boot et des API RESTful, a été retenu pour sa puissance et sa facilité d'intégration avec des bases de données et des services externes. Il assure une communication fluide entre le front-end et le back-end, garantissant une transmission efficace et sécurisée des données.

Nous avons opté pour cette architecture MVC, car elle s'adapte parfaitement aux besoins de notre projet, qui implique une forte interaction entre l'interface utilisateur et la base de données. Elle facilite l'évolutivité, optimise la maintenance et améliore la gestion des mises à jour, tout en assurant une séparation claire des responsabilités. Grâce à cette approche, notre application bénéficie d'une structuration robuste et flexible, garantissant une performance optimale et une expérience utilisateur fluide.

2.3.3 Front-End

2.3.3.1 React.js

React est une bibliothèque JavaScript libre et open source développée par Meta (anciennement Facebook) pour créer des interfaces utilisateur. Elle permet de concevoir des applications web en structurant l'interface sous forme de composants réutilisables. React utilise JSX, une combinaison de JavaScript et de XML, pour définir des éléments d'interface utilisateur. Les composants React peuvent être assemblés de manière flexible afin de concevoir des applications de complexité variable. Maintenue activement par Meta et la communauté open source, cette bibliothèque évolue continuellement. Elle est particulièrement adaptée au développement d'interfaces utilisateur dynamiques et interactives.

Nous avons choisi **React.js** pour le développement du front-end de notre application en raison de plusieurs avantages majeurs :

- **Modularité et réutilisabilité** : React permet d'organiser l'interface en composants réutilisables, simplifiant ainsi la maintenance et l'évolution de l'application.
- **Performance optimisée** : Grâce à son *Virtual DOM*, React réduit le nombre de modifications directes du DOM réel, ce qui améliore la réactivité et la fluidité de l'application.
- **Écosystème riche et support communautaire** : React bénéficie d'une vaste communauté et d'un écosystème complet (Redux, React Router, etc.), offrant des solutions adaptées aux besoins des applications modernes.
- **Compatibilité avec les API REST et GraphQL** : React facilite la gestion des requêtes asynchrones et s'intègre aisément avec un backend basé sur **Spring Boot**.

2.3.3.2 Tailwind CSS

Tailwind CSS est un framework CSS utilitaire qui permet de créer des interfaces utilisateur personnalisées rapidement. Contrairement aux frameworks CSS traditionnels qui fournissent des composants prédéfinis, Tailwind CSS se concentre sur l'utilisation de classes utilitaires qui permettent de contrôler directement les propriétés CSS d'un élément dans le code HTML. Cette approche rend le développement plus flexible et permet de concevoir des designs de manière plus précise, sans avoir à écrire de CSS personnalisé.

2.3.3.3 Axios

Axios est une bibliothèque JavaScript pour effectuer des requêtes HTTP, basée sur des promesses. Elle est utilisée pour faire des appels API (GET, POST, PUT, DELETE, etc.) dans le navigateur. Elle permet également de gérer les erreurs de manière simple, de configurer des en-têtes HTTP et d'annuler des requêtes.

2.3.4 Back-End

2.3.4.1 Spring Boot

Spring Boot est un framework Java moderne qui simplifie le développement d'applications web et de microservices. Développé par Pivotal Software, il automatise la configuration des applications Spring

grâce à ses fonctionnalités d'auto-configuration. Spring Boot permet de créer rapidement des applications autonomes et exécutables avec une configuration minimale. Il adopte une approche conventionnelle plutôt que configurationnelle, ce qui facilite le démarrage rapide des projets. Le framework intègre nativement des fonctionnalités essentielles comme la gestion des dépendances et le monitoring, offrant ainsi une solution robuste et performante.

Nous avons choisi **Spring Boot** pour le développement du back-end de notre application en raison de ses nombreux avantages :

- **Simplicité et rapidité de développement** : Grâce à son auto-configuration et à ses dépendances intégrées, Spring Boot réduit le temps de mise en place du projet.
- **Architecture modulaire et extensible** : Il permet de structurer le code en couches bien définies, facilitant ainsi la maintenance et l'évolution de l'application.
- **Intégration facile avec les bases de données** : Spring Boot prend en charge divers SGBD (MySQL, PostgreSQL, MongoDB, etc.) et simplifie l'accès aux données grâce à *Spring Data JPA*.
- **Compatibilité avec les API REST** : Il offre des outils puissants pour exposer des services RESTful, assurant une communication fluide avec le front-end développé en **React.js**.
- **Adapté aux microservices et aux applications cloud-native** : Spring Boot est idéal pour concevoir des architectures évolutives et déployables dans des environnements cloud.

2.3.5 Base de données

2.3.5.1 MySQL

MySQL est un système de gestion de bases de données relationnelles SGBDR open source, largement utilisé pour le stockage et la gestion de données dans un format structuré. Développé initialement par MySQL AB et acquis par Oracle Corporation, il repose sur l'utilisation du SQL (Structured Query Language) comme langage de manipulation des données. Reconnue pour sa **fiabilité**, sa **simplicité d'utilisation** et sa **compatibilité multi-plateformes**, MySQL se distingue par ses performances élevées et son évolutivité. Il prend en charge les transactions ACID, assurant ainsi l'intégrité des données, et propose des fonctionnalités avancées telles que la **réplication**, la gestion des **vues** et des mécanismes d'optimisation des requêtes.

Le choix de **MySQL** pour la gestion de la base de données dans le cadre de notre projet repose sur plusieurs critères fondamentaux :

- **Performance et rapidité** : MySQL est optimisé pour le traitement rapide des requêtes, ce qui améliore l'efficacité globale de l'application, notamment dans les environnements à fort trafic.
- **Support des transactions ACID** : Cette fonctionnalité garantit la cohérence des données en assurant une gestion rigoureuse des transactions, un atout majeur pour les systèmes nécessitant une haute fiabilité.
- **Interopérabilité avec Spring Boot** : Grâce à *Spring Data JPA*, MySQL s'intègre parfaitement avec le framework **Spring Boot**, offrant ainsi une solution cohérente pour la gestion du backend.

- **Large communauté et support étendu** : MySQL bénéficie d'une vaste communauté de développeurs et d'une documentation complète, ce qui facilite la résolution de problèmes et l'optimisation des performances.

2.3.5.2 Comparaison avec d'autres SGBD

Afin d'évaluer la pertinence de **MySQL** dans le cadre de notre projet, nous avons procédé à une comparaison avec d'autres systèmes de gestion de bases de données populaires, à savoir **NoSQL** (MongoDB) et **MariaDB**.

- **NoSQL (MongoDB)** : Contrairement à MySQL, MongoDB est une base de données orientée document, utilisant des structures JSON. Ce modèle est particulièrement adapté aux applications nécessitant une grande flexibilité dans la structure des données, telles que les applications de *big data* ou de stockage de documents non structurés. Toutefois, bien que MongoDB excelle dans ces domaines, **MySQL** demeure préférable pour les applications nécessitant des transactions complexes, des relations de données bien définies et une gestion stricte de la cohérence des données.
- **MariaDB** : MariaDB est un fork de MySQL qui propose des optimisations spécifiques et des performances accrues dans certains cas. Cependant, **MySQL** reste la base de données la plus utilisée dans l'industrie, bénéficiant d'un support plus large et d'une plus grande compatibilité avec des outils comme **Spring Boot**, ce qui en fait une solution privilégiée pour les applications orientées entreprise.

Ainsi, bien que d'autres systèmes de gestion de bases de données existent, nous avons opté pour **MySQL** en raison de sa robustesse éprouvée, de sa facilité d'intégration avec notre architecture, et de son large éventail de fonctionnalités adaptées à notre projet.

2.4 Outils de développement utilisés

2.4.1 Git & Github

Git est un système de contrôle de version décentralisé qui permet de gérer efficacement les modifications du code source au fil du temps. GitHub, une plateforme en ligne fondée sur Git, facilite l'hébergement, la collaboration et le suivi des versions des projets de développement.

2.4.2 IntelliJ IDEA Community Edition

IntelliJ IDEA Community Edition est une version gratuite et open source de l'environnement de développement intégré IDE IntelliJ IDEA. Elle offre des fonctionnalités avancées pour le développement dans des langages tels que Java, Groovy, Kotlin, et les technologies associées. Cette édition inclut notamment un débogueur intégré, un système de contrôle de versions Git, et des outils d'analyse de code sophistiqués. Elle est particulièrement adaptée au développement Java pur et aux frameworks populaires comme Spring Boot. Contrairement à l'édition Ultimate payante, la Community Edition se concentre sur le développement Java et ne comprend pas les fonctionnalités de développement web complet ou mobile. Elle est maintenue par JetBrains et bénéficie des mises à jour régulières pour assurer son efficacité dans le développement logiciel moderne.

2.4.3 VS Code

VS Code (Visual Studio Code) est un éditeur de code léger et open source développé par Microsoft. Il offre une expérience de développement moderne avec une interface intuitive et une grande extensibilité grâce à ses extensions. VS Code prend en charge de nombreux langages de programmation et propose des fonctionnalités telles que le débogage intégré, la gestion de Git, et l'intelligence de code avancée. Contrairement aux IDE complets comme IntelliJ IDEA, VS Code se positionne comme un éditeur de code polyvalent qui peut évoluer vers un IDE complet selon les besoins grâce à ses extensions. Sa légèreté et sa flexibilité en font un outil populaire auprès des développeurs, particulièrement adapté au développement web et cloud moderne.

2.5 Authentification et Sécurité

- **Notre application utilise un JSON Web Token (JWT) pour l'authentification** : Ce système permet de transmettre des informations de manière sécurisée entre le client et le serveur grâce à des jetons signés. Le processus d'authentification se fait en deux étapes : d'abord, l'utilisateur se connecte et le serveur génère un jeton JWT qu'il renvoie au client. Ensuite, pour chaque requête, le client envoie ce jeton, et le serveur le vérifie pour garantir que l'utilisateur est authentifié. JWT présente plusieurs avantages : il ne nécessite pas de stockage de session côté serveur (réduisant ainsi la charge et améliorant la performance), il est sécurisé grâce à sa signature et il offre une grande flexibilité, notamment pour les systèmes distribués.
- **Bcrypt pour sécuriser les mots de passe dans notre application** : bcrypt est un algorithme de hachage qui rend les mots de passe illisibles. Contrairement aux méthodes classiques comme MD5, bcrypt utilise un processus de salage (ajout d'une chaîne unique à chaque mot de passe) et un facteur de coût pour ralentir le hachage, le rendant plus résistant aux attaques par force brute. Même si les données sont compromises, le mot de passe d'origine reste pratiquement introuvable, assurant ainsi une protection optimale des utilisateurs.

Conclusion

Dans ce chapitre, nous avons présenté l'analyse des besoins du projet ainsi que la modélisation UML utilisée pour concevoir le système. Les choix techniques ont également été détaillés, incluant les outils et technologies adaptés à la mise en œuvre de la solution. Cette approche méthodique garantit que l'application répondra efficacement aux attentes des utilisateurs tout en étant flexible et évolutive.

Chapitre 3

Résultats de l'application

Introduction

Ce chapitre présente le résultat de notre travail à travers des captures d'écran ainsi que ses limites. Il s'agit d'une application web de gestion et de suivi des abonnements des clients dans une salle de gym.

3.1 Présentation des interfaces de l'application Web

3.1.1 Page d'accueil

Il s'agit de la première page que l'utilisateur découvre en accédant à notre application. Elle lui permet d'explorer les services que nous proposons avant de créer un compte.

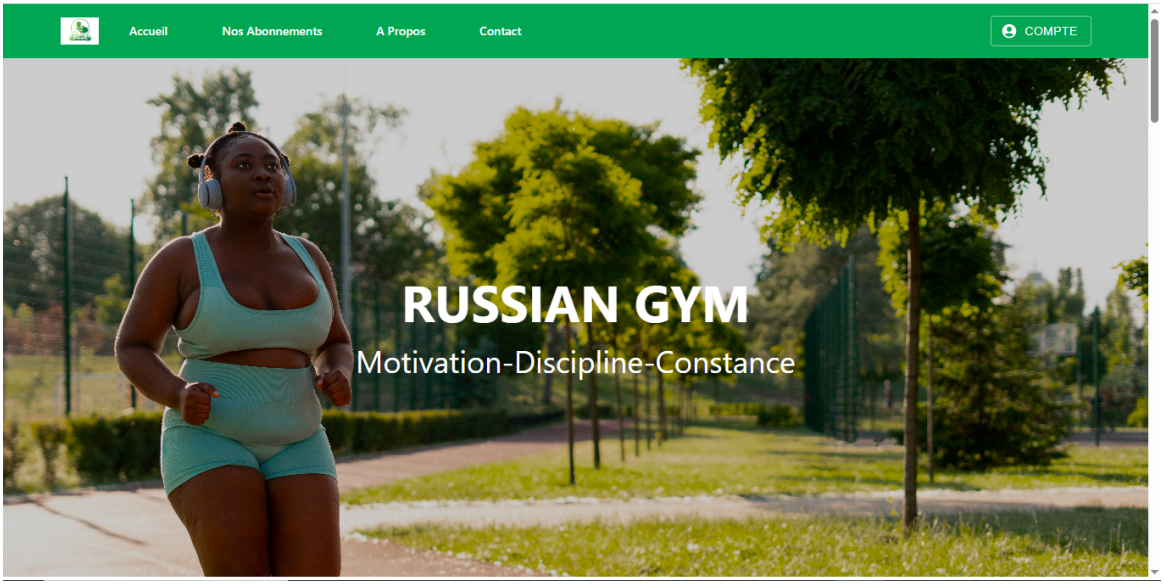


FIGURE 3.1 : Notre page d'accueil

3.1.2 Page d'authentification

La page d'authentification permet aux utilisateurs d'accéder à l'application via une connexion sécurisée ou de créer un nouveau compte si nécessaire.

- Connexion utilisateur

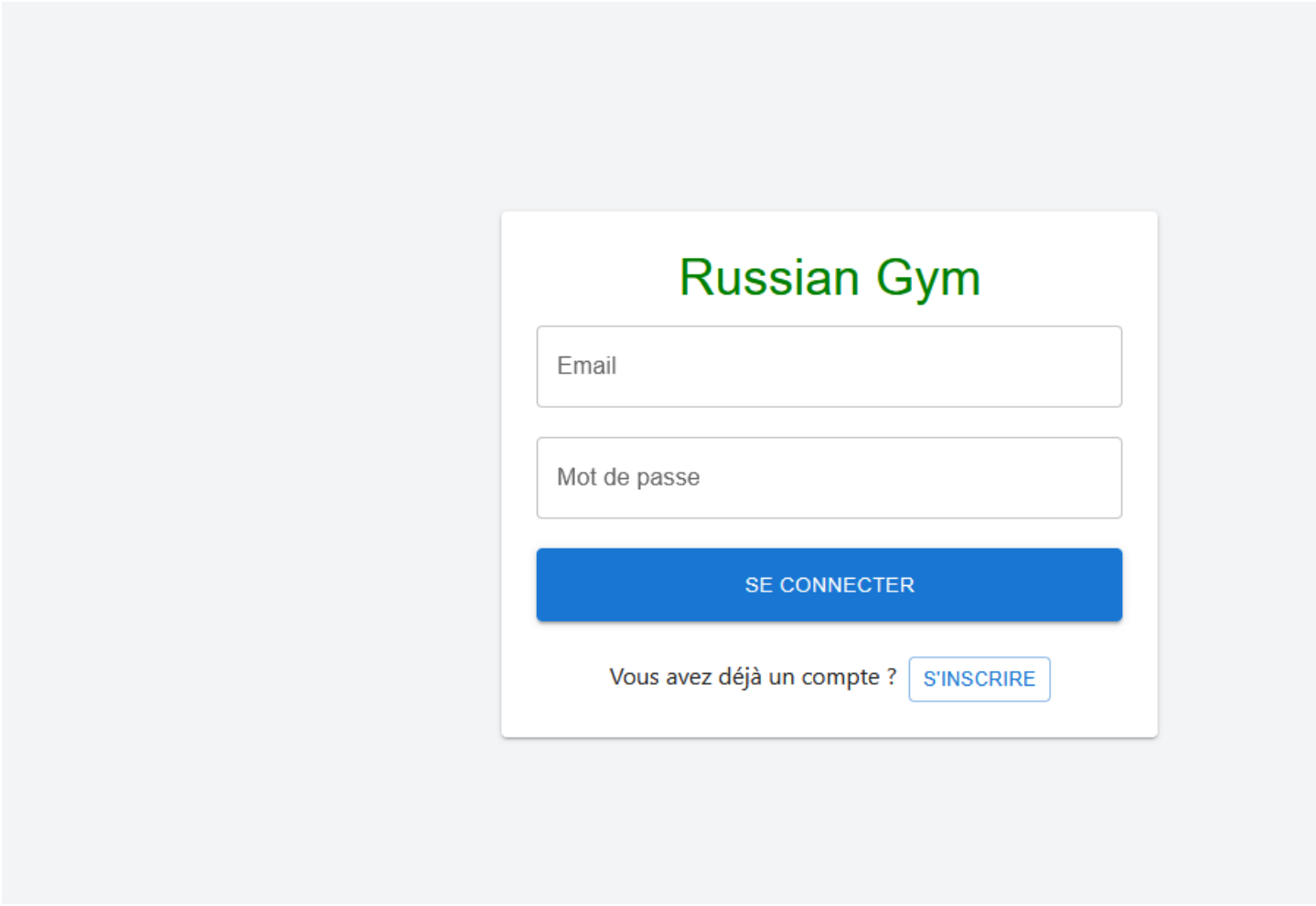


FIGURE 3.2 : Page de connexion de l'application

Cette interface offre aux utilisateurs enregistrés une manière simple de se connecter à leur compte. En saisissant leur adresse e-mail et leur mot de passe, ils accèdent à l'application. Pour une expérience utilisateur fluide, des fonctionnalités telles que la validation en temps réel des champs.

- Page d'inscription

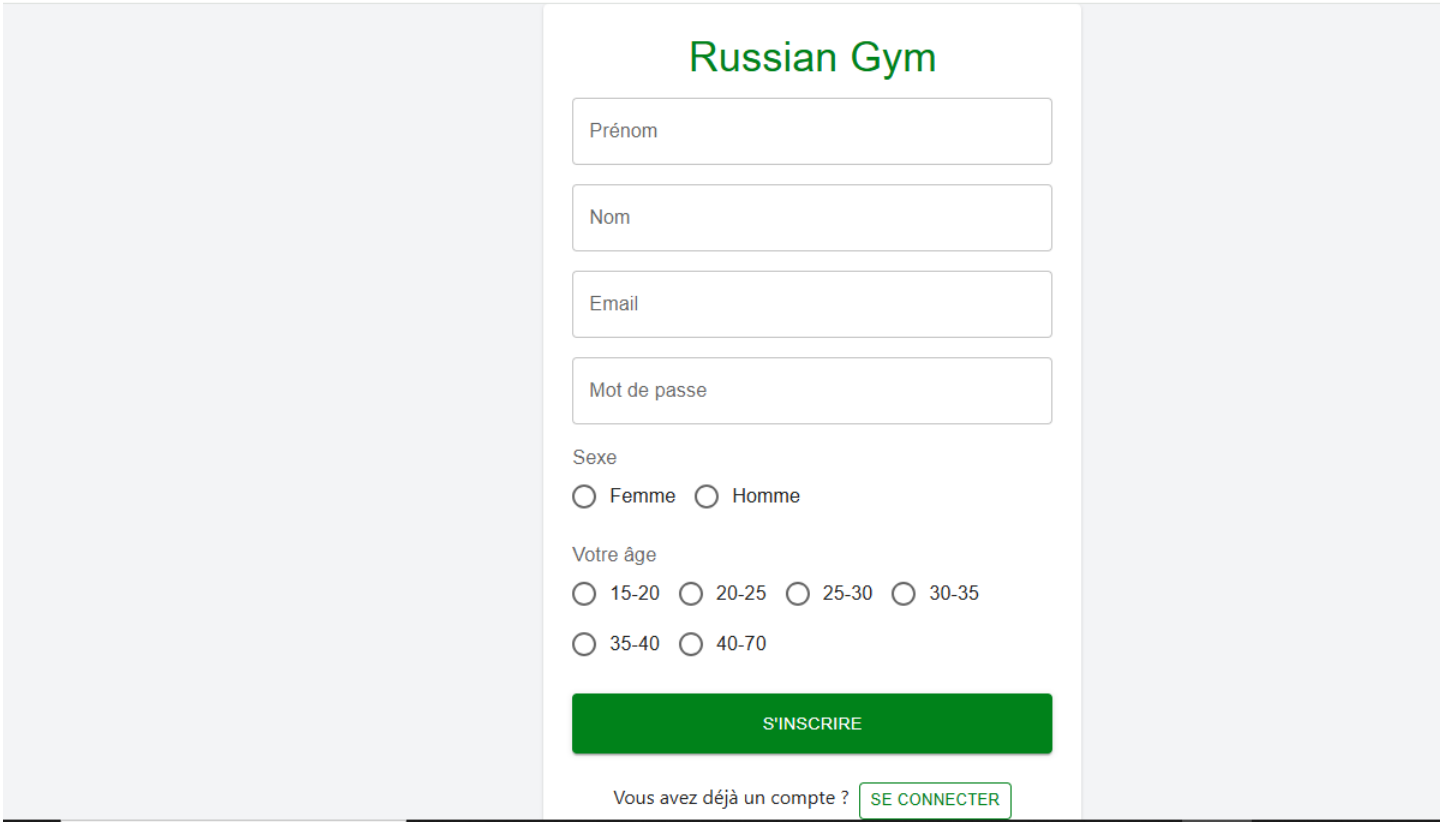


FIGURE 3.3 : Page d'inscription de l'application

Cette interface permet aux nouveaux utilisateurs de créer un compte en quelques étapes simples. Ils doivent remplir un formulaire comprenant leur nom complet, leur adresse e-mail et un mot de passe sécurisé. Ensuite, ils sélectionnent leur sexe et indiquent leur tranche d'âge. Un système de validation des données, incluant la vérification de l'unicité de l'adresse e-mail et la robustesse du mot de passe, assure une inscription fiable et sécurisée.

3.1.3 Tableau de bord de l'adhérent

Après sa connexion, l'utilisateur accède à son tableau de bord, un espace centralisé où il peut consulter les informations essentielles de son abonnement : son statut, son type, sa durée de validité et la possibilité de le renouveler après expiration. Un menu d'accès rapide simplifie la navigation vers les fonctionnalités principales, offrant une expérience utilisateur fluide et intuitive grâce à une interface moderne.

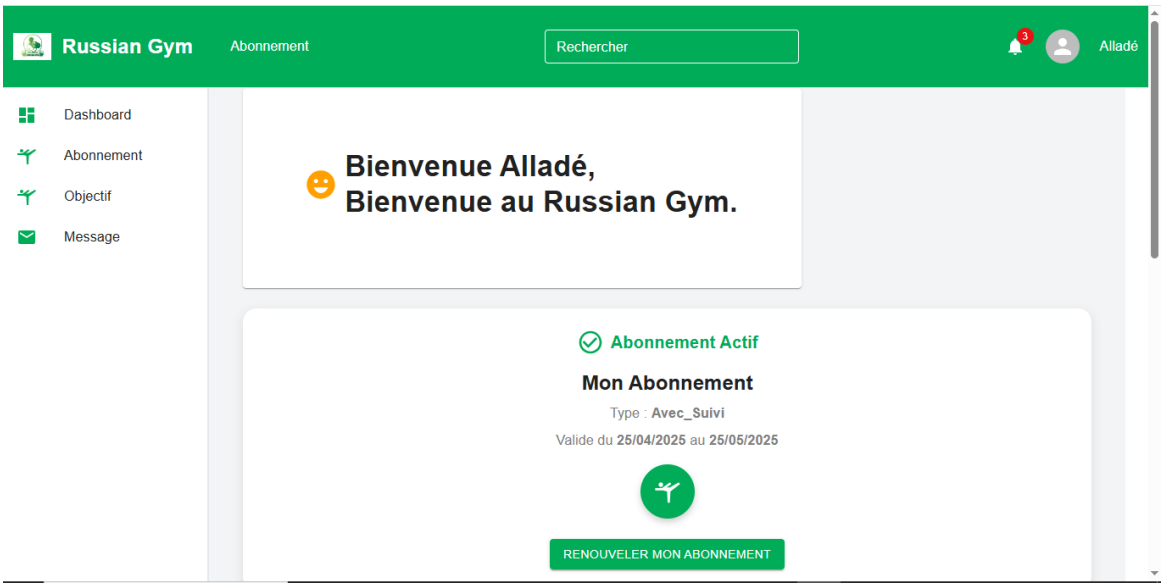


FIGURE 3.4 : Interface adhérent – Tableau de bord

3.1.4 Souscription à un abonnement

Cette section présente les différents types d'abonnements proposés par le centre, parmi lesquels l'utilisateur pourra choisir celui qui lui convient pour souscrire.

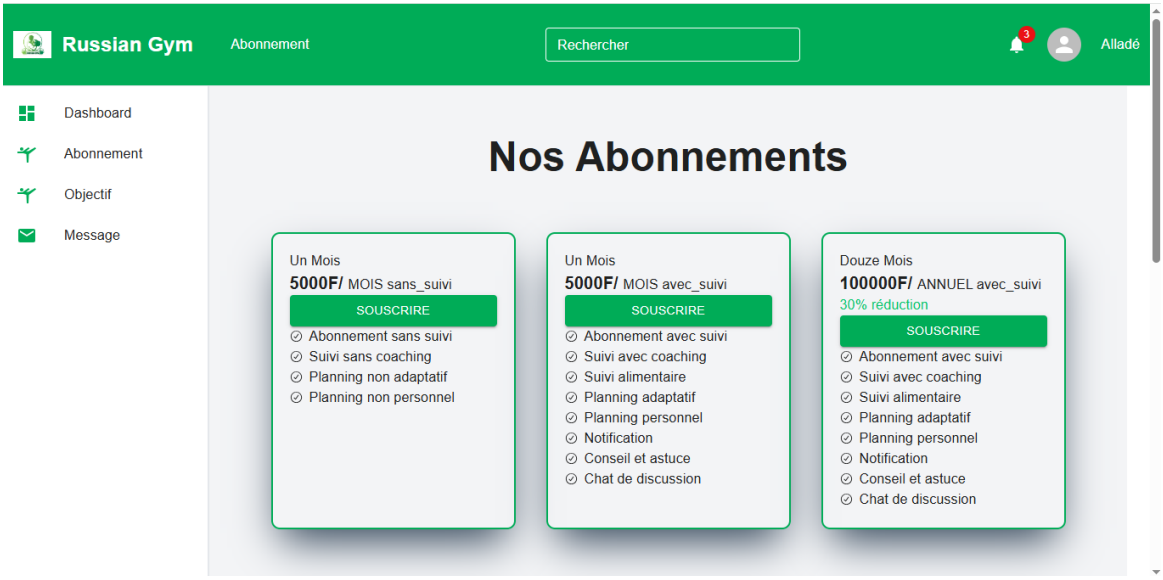


FIGURE 3.5 : Interface de souscription aux abonnements

3.1.5 Message de succès après abonnement

Cette section présente la page vers laquelle le client est redirigé après la réussite du paiement de son abonnement, lorsqu'il a souscrit à une offre avec suivi.

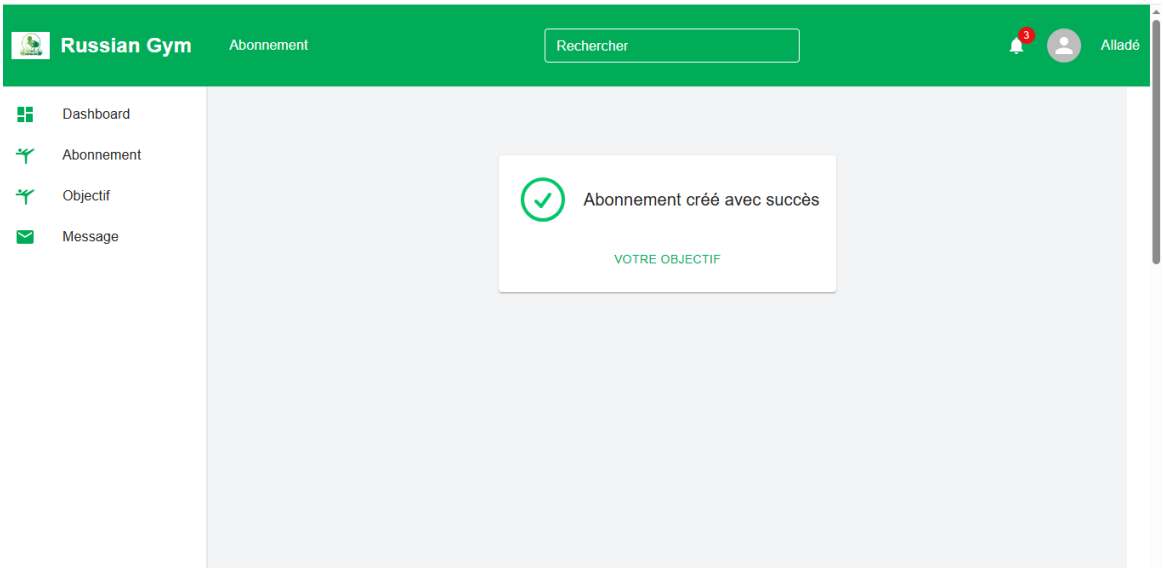


FIGURE 3.6 : Page de confirmation après souscription

3.1.6 Sélection d'objectif par un adhérent

Cette section présente les différents types d'objectifs que l'abonné peut choisir pour atteindre son but, et qui lui permettront de recevoir le planning correspondant à cet objectif.

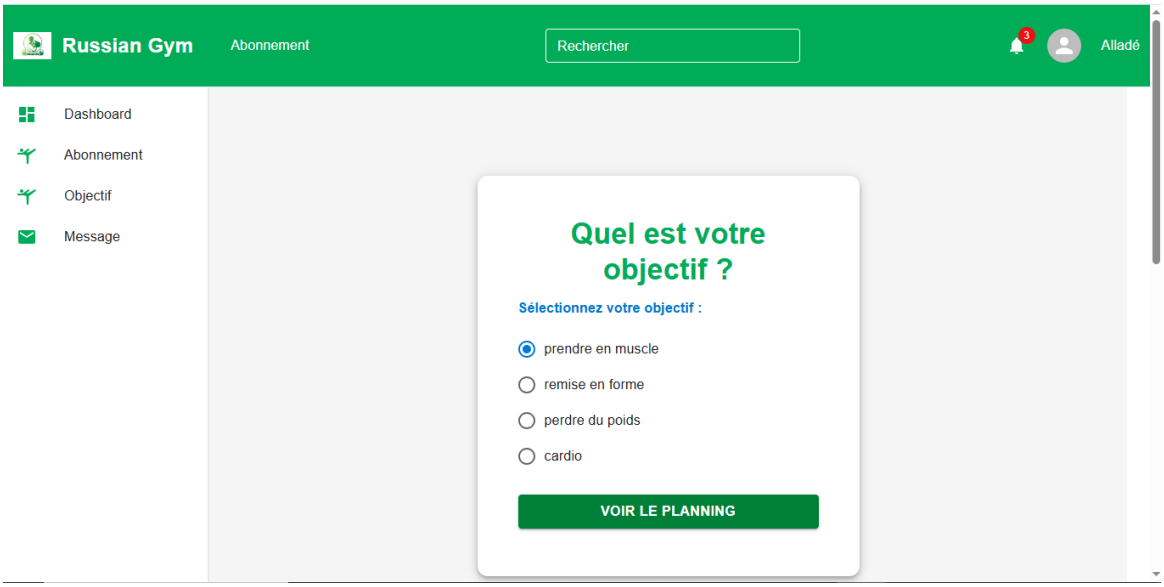


FIGURE 3.7 : Interface permettant à l'adhérent de choisir son objectif

3.1.7 Page de visualisation des plannings par l'adhérent

Cette section présente la consultation des plannings créés par le coach, auxquels l'adhérent pourra assister en fonction de sa disponibilité.



FIGURE 3.8 : Interface de consultation des plannings créés par le coach, accessibles en fonction de la disponibilité de l'adhérent

3.1.8 Chat de discussion entre coachs et adhérents

Cette section présente l'interface de chat de discussion permettant la communication entre les coachs et les adhérents.



FIGURE 3.9 : Interface du chat de discussion entre coachs et adhérents

3.1.9 Tableau de bord de la secrétaire

Cette section présente l'affichage des statuts des abonnements des adhérents dans le centre, que la secrétaire pourra consulter pour suivre l'évolution des abonnements.

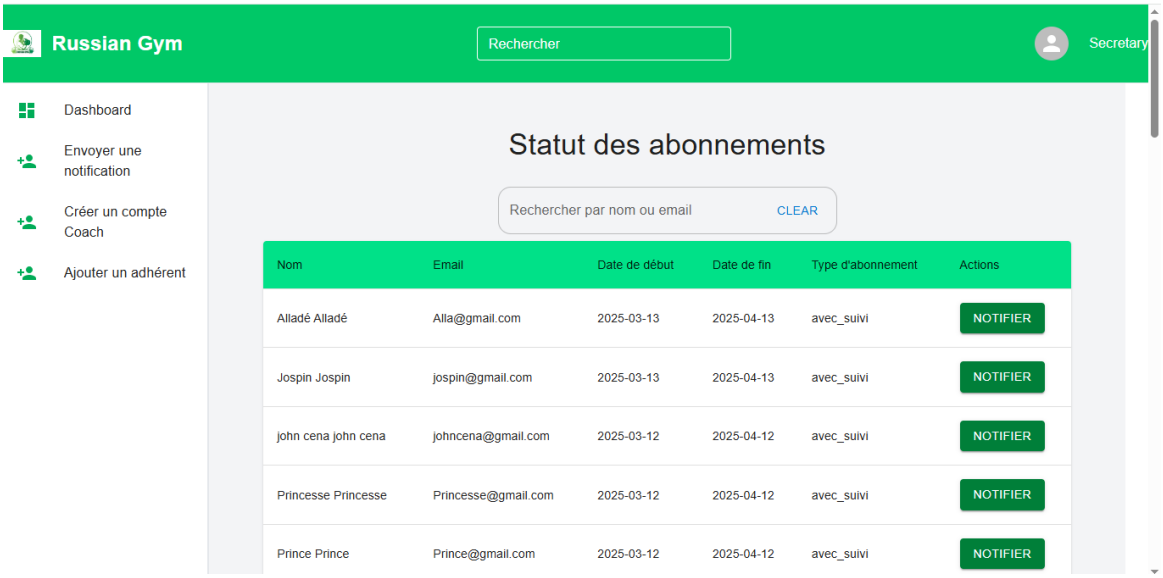


FIGURE 3.10 : Interface secrétaire - Tableau de bord

3.1.10 Tableau de bord du coach

Cette section présente le tableau de bord du coach, où il pourra voir le nombre de clients qu'il suit dans sa spécialité en fonction des objectifs.

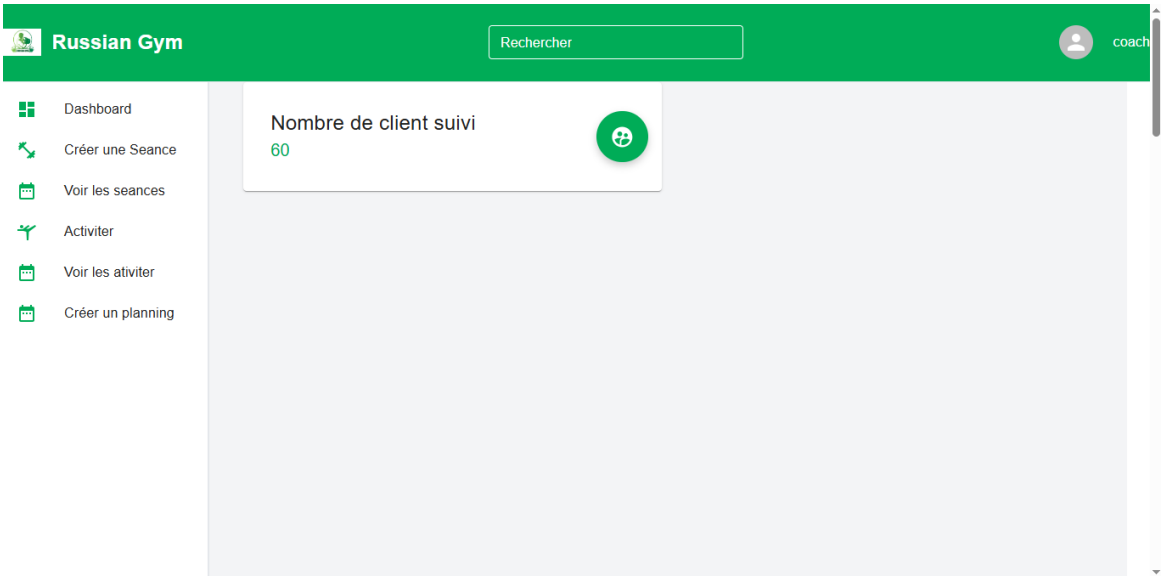


FIGURE 3.11 : Interface Coach - Tableau de bord

3.1.11 Gestion des plannings

3.1.11.1 Création d'une séance par le coach

Cette section présente l'interface permettant à un coach de créer une séance pour la semaine.

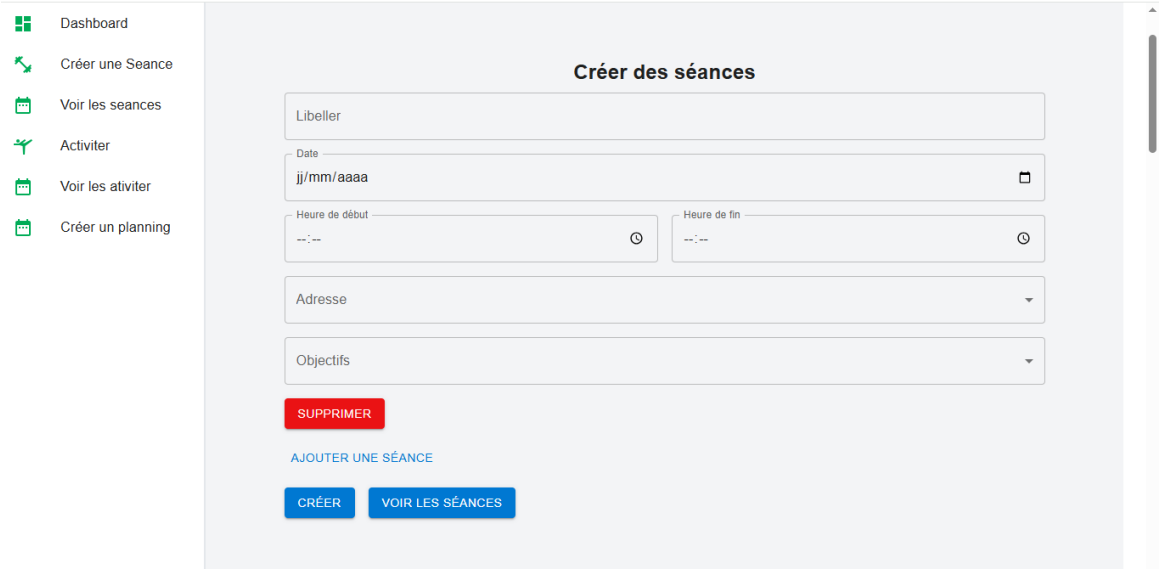


FIGURE 3.12 : Interface de création d'une séance par un coach

3.1.11.2 Visualisation des séances créées par un coach

Cette section permet au coach de consulter les séances qu'il a créées afin de les modifier ou de les supprimer.

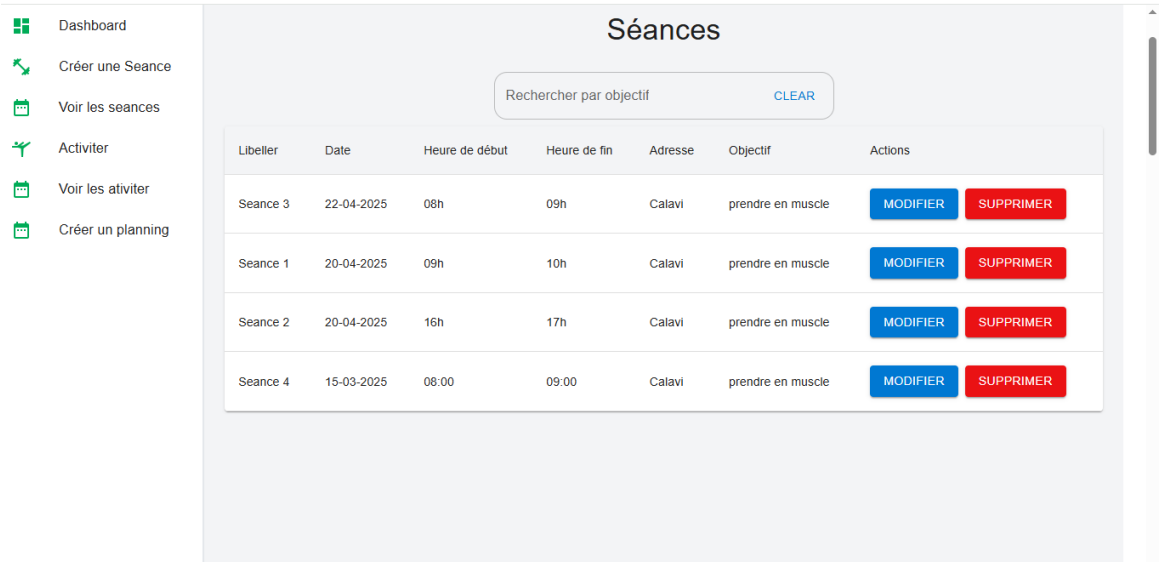


FIGURE 3.13 : Interface d'affichage des séances créées par un coach

3.1.11.3 Création d'une activité par un coach

Cette section permet au coach de créer une activité en tenant compte des séances et des objectifs.

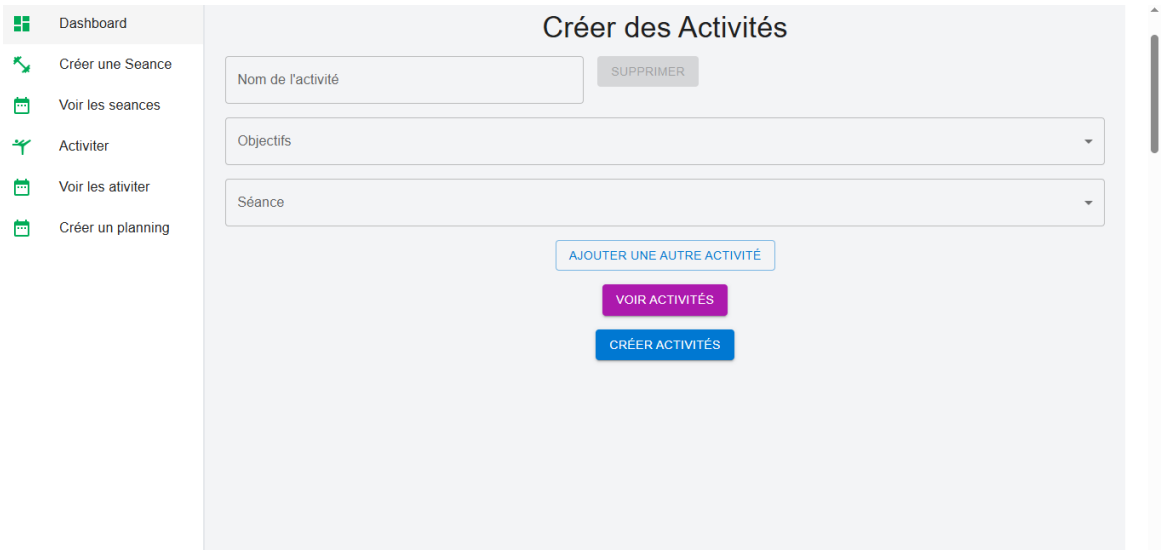


FIGURE 3.14 : Interface de création d'une activité par un coach

3.1.11.4 Interface de visualisation des activités créées par un coach

Cette section permet d'afficher les activités créées par le coach afin de les modifier ou de les supprimer.

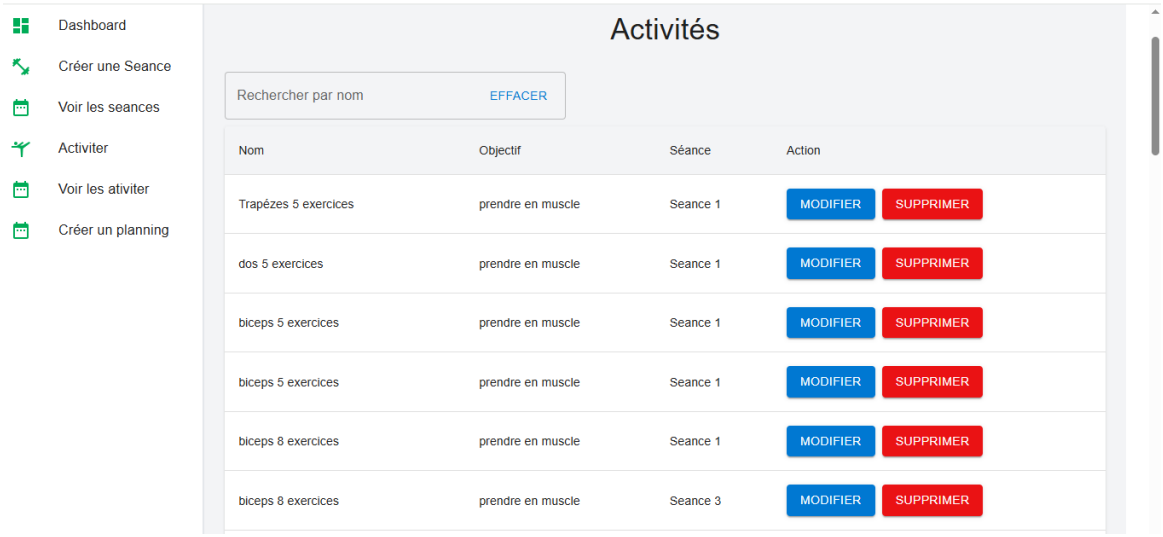


FIGURE 3.15 : Interface d'affichage des activités créées par un coach

3.1.12 Tableau de bord de l'administrateur

Cette section permet à l'administrateur de consulter les statistiques complètes des interactions ayant lieu avec les utilisateurs.

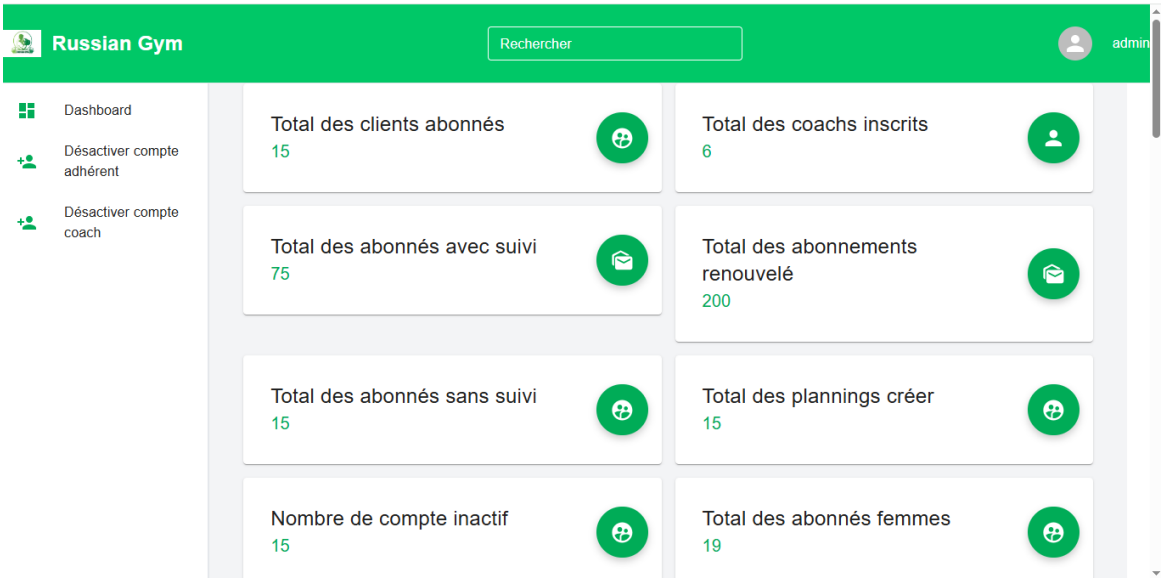


FIGURE 3.16 : Interface administrateur - Tableau de bord des statistiques

3.2 Tests et Validation de l'Application

3.2.1 Tests Unitaires

Les tests unitaires ont été réalisés principalement pour la fonctionnalité de création de compte. Ces tests visent à vérifier que chaque composant de cette fonctionnalité fonctionne correctement de manière isolée.

3.2.1.1 Objectif des Tests Unitaires

- Garantir que l'utilisateur peut créer un compte si les informations fournies sont correctes ;
- Empêcher la duplication de comptes avec un email déjà enregistré ;
- Retourner les statuts et messages appropriés en cas de succès ou d'erreur.

3.2.1.2 Méthodologie

Pour effectuer ces tests, nous avons réalisé des simulations en isolant la base de données à l'aide de mocks, tant au niveau du back-end que du front-end. Les réponses attendues de l'API ont été comparées aux résultats réels pour valider les comportements suivants :

- Vérification si un utilisateur existe déjà dans la base de données avec l'email fourni ;
- Ajouter un nouvel utilisateur en cas de non-existence de compte avec l'email fourni ;
- Réception du retour avec les statuts appropriés (201 pour succès, 400 pour utilisateur déjà existant).

3.2.1.3 Résultats

Tous les scénarios prévus ont été validés avec succès, garantissant la robustesse du back-end et du front-end pour cette fonctionnalité.

TABLE 3.1 : Résumé des tests unitaires effectués

Test	Statut	Description
Création de compte réussie (Back-End)	✓	Un utilisateur est créé avec succès dans la base de données.
Utilisateur existant (Back-End)	✓	Retourne une erreur 400 et un message approprié.
Création de compte réussie (Front-End)	✓	L'API retourne un message de succès dans l'interface React js.
Utilisateur existant (Front-End)	✓	L'API retourne une erreur et affiche un message d'erreur.

3.3 Tests Manuels

En complément des tests unitaires, des tests manuels ont été effectués pour évaluer des fonctionnalités complexes et interactives. Ces tests permettent de valider l'intégration et l'expérience utilisateur, en s'assurant de la fluidité et de l'interactivité des différentes fonctionnalités.

3.3.1 Souscription à un abonnement

Description : Cette fonctionnalité permet aux nouveaux adhérents de choisir l'abonnement qui leur convient et d'accéder aux fonctionnalités associées.

Tests Réalisés :

- **Création d'un compte** : Vérification que l'adhérent peut s'inscrire correctement.
- **Choix du type d'abonnement avec suivi** :
  - Redirection vers une page de succès après la création de l'abonnement ;
  - Sélection du type d'objectif (exemple : prendre en muscle) ;
  - Affichage automatique du planning hebdomadaire généré.

Résultats : Tous les scénarios prévus ont été validés avec succès. L'inscription et la souscription à un abonnement fonctionnent correctement, avec une redirection fluide et un affichage dynamique du planning.

3.3.2 Gestion des plannings

Description : Cette fonctionnalité permet de gérer efficacement les plannings des séances et des activités, en prenant en compte les disponibilités des salles et les objectifs des adhérents.

Tests Réalisés :

- Création d'une séance avec les informations suivantes :
  - Saisie de la date, de l'heure de début et de fin de la séance ;
  - Association d'une salle disponible ;
  - Définition d'un objectif pour la séance.
- Création d'une activité :

  - Ajout du nom de l'activité ;
  - Association de l'activité à une séance en fonction de l'objectif défini.

Résultats : Les tests réalisés ont validé avec succès la création et la gestion des plannings, assurant une prise en compte correcte des informations saisies, la bonne affectation des salles et des activités aux séances, ainsi que la cohérence entre les objectifs définis et les activités associées.

3.3.3 Suivi des abonnements

Description : Cette fonctionnalité permet au centre de suivre efficacement les abonnements des adhérents afin de gérer leur statut et leur renouvellement.

Tests Réalisés :

- Connexion de la secrétaire au système ;
- Affichage des abonnements dans un tableau contenant les informations suivantes :
  - Nom et email de l'adhérent ;
  - Date de début et de fin de l'abonnement.

Résultats : Les tests ont été validés avec succès. La secrétaire peut se connecter sans problème et consulter les abonnements des adhérents dans un tableau structuré, garantissant une gestion fluide et efficace.

3.4 Conclusion des Tests

Les tests unitaires et manuels ont permis de garantir la stabilité et la fiabilité des fonctionnalités essentielles de l'application. Les résultats obtenus valident que l'application répond aux besoins des utilisateurs, tant en termes de précision que d'expérience utilisateur.

3.5 Discussion

La conception de l'application web de gestion et de suivi des abonnements des clients pour Russian Gym Fitness est guidée par plusieurs objectifs clés : suivre les abonnements des clients, permettre aux nouveaux adhérents de s'abonner en choisissant le type d'abonnement qui leur convient, renouveler leur abonnement après expiration, recevoir un planning adapté à leur objectif, gérer les plannings et mettre en place un chat de discussion pour faciliter la communication entre les abonnés et les coachs. Actuellement, le système permet d'effectuer les abonnements en choisissant un type d'abonnement, de recevoir un planning après la sélection de l'objectif, de gérer les plannings, de renouveler les abonnements après expiration et de suivre efficacement le statut des abonnés.

3.5.1 Difficultés rencontrées

Le développement de notre application a rencontré plusieurs défis majeurs, notamment des bugs complexes qui ont retardé certaines fonctionnalités clés. Ces problèmes ont nécessité des ajustements

pour assurer la stabilité de l'application et une gestion rigoureuse des ressources pour minimiser les retards. Certaines interfaces ont dû être modifiées pour améliorer l'expérience utilisateur, rendant l'application plus intuitive et fluide. L'intégration de la fonctionnalité hors ligne, bien que prévue, n'a pas pu être réalisée en raison de contraintes techniques et temporelles. Malgré ces obstacles, l'équipe a travaillé pour garantir une expérience optimale et renforcer la performance de l'application.

3.5.2 Perspectives

À l'avenir, plusieurs améliorations sont envisagées pour enrichir l'application. L'intégration de l'intelligence artificielle optimisera les recommandations d'entraînement en proposant des conseils personnalisés. L'ajout de fonctionnalités hors ligne permettra l'accès aux données en cas de connectivité réduite. Le développement d'une version mobile offrira une flexibilité accrue, accessible depuis n'importe quel endroit. La personnalisation des thèmes et des rapports renforcera l'engagement des abonnés. Un système de contrôle d'accès basé sur des cartes FNP, utilisant des technologies comme le RFID, améliorera la sécurité et simplifiera la gestion des accès. Ces évolutions visent à maximiser l'efficacité de l'application et l'expérience utilisateur.

Conclusion

Ce chapitre a présenté les principales interfaces de notre application web ainsi que ses fonctionnalités essentielles. Malgré quelques limites identifiées, des perspectives d'amélioration ont été envisagées pour enrichir l'expérience utilisateur et répondre aux besoins futurs. Ces évolutions permettront d'assurer une solution plus complète, plus efficace et mieux adaptée aux attentes du centre.

Conclusion Générale

Grâce aux technologies numériques, nous avons résolu le problème de gestion des abonnements pour la salle de gym. L'objectif de notre projet était de concevoir une application web dédiée à la gestion et au suivi des abonnements des clients du Russian Gym Fitness Club. Cette application permet aux adhérents de s'abonner en choisissant un type d'abonnement, de renouveler leurs abonnements après expiration, et de recevoir un planning adapté à leurs objectifs. Les coachs peuvent gérer les plannings, et un chat de discussion a été mis en place pour faciliter la communication entre les adhérents et les coachs. L'objectif a été atteint grâce à l'utilisation de technologies et d'outils spécifiques. Toutefois, plusieurs améliorations sont envisagées pour enrichir l'application. Parmi celles-ci, l'intégration de l'intelligence artificielle pour optimiser les recommandations d'entraînements, l'ajout d'une fonctionnalité hors ligne, la personnalisation des thèmes, le développement d'une version mobile et l'intégration d'un système de contrôle d'accès basé sur des cartes RFID.

Bibliographie

[1] D. Gabay and J. Gabay. *UML 2 - Analyse et conception mise en œuvre guidée avec études de cas (2ème édition)*. Infopro. Dunod, April 2008.

Webographie

[1] Wikipedia, *Application web*, https://fr.wikipedia.org/wiki/Application_web, consulté le 31 janvier 2025.

[2] Spa, *Gestion des abonnements*, https://www.sap.com/france/products/financial-management/billing-revenue-innovation-management/what-is-subscription-management.html, consulté le 31 janvier 2025.

[3] Spa, *Suivi des abonnements*, https://www.sap.com/france/products/financial-management/billing-revenue-innovation-management/what-is-subscription-management.html, consulté le 31 janvier 2025.

[4] Wiktionary, *Salle de gym*, https://fr.wiktionary.org/wiki/salle_de_fitness, consulté le 31 janvier 2025.

[5] EditionsENI. *Merise-guide pratique(2ième édition)*, https://www.editions-eni.fr/, consulté le 31 janvier 2025.

[6] nutcache. *Démarche de projet.*, https://www.editions-eni.fr/, 2020.

Table des matières

Dédicace ii

Remerciements iii

Résumé iv

. iv

Abstract v

. v

List of Figures vi

List of Tables vii

Liste des acronymes viii

Introduction 1

1 Revue de la littérature 3

Introduction 3

1.1 Clarification Conceptuelle 3

1.1.1 Une application web 3

1.1.2 Gestion des abonnements 3

1.1.3 Suivi des abonnements 3

1.1.4 Salle de gym 4

1.2 Étude des solutions existantes 4

1.2.1 Resamania 4

1.2.2 FLiiP 5

1.2.3 Virtuagym 5

1.2.4 Tableau récapitulatif 6

Conclusion 7

2 Analyse, Conception et Choix technique 9

Introduction 9

2.1 Analyse des besoins 9

2.1.1 Les besoins fonctionnels 10

2.1.1.1 Suivi des abonnements des clients 10

2.1.1.2 Fonctionnalités offertes aux clients 10

2.1.1.3 Fonctionnalités réservées à la secrétaire 10

2.1.1.4 Fonctionnalités réservées au coach . . . 10
2.1.1.5 Fonctionnalités réservées à l'administrateur . . . 10
2.1.2 Les besoins non fonctionnels . . . 11
2.2 Conception . . . 11
2.2.1 Modélisation avec UML . . . 11
2.2.2 Diagramme de cas d'utilisation . . . 11
2.2.3 Diagramme de classes . . . 13
2.2.4 Diagramme de séquence . . . 14
2.2.5 Diagramme de séquence pour l'authentification . . . 14
2.2.6 Diagramme de séquence pour la souscription à un abonnement . . . 14
2.3 Choix techniques . . . 15
2.3.1 Choix du langage de modélisation . . . 15
2.3.2 Architecture du Système : Modèle MVC . . . 16
2.3.3 Front-End . . . 17
2.3.3.1 React.js . . . 17
. . . 17
2.3.3.2 Tailwind CSS . . . 17
2.3.3.3 Axios . . . 17
2.3.4 Back-End . . . 17
2.3.4.1 Spring Boot . . . 17
. . . 18
2.3.5 Base de données . . . 18
2.3.5.1 MySQL . . . 18
. . . 18
2.3.5.2 Comparaison avec d'autres SGBD . . . 19
2.4 Outils de développement utilisés . . . 19
2.4.1 Git & Github . . . 19
2.4.2 IntelliJ IDEA Community Edition . . . 19
2.4.3 VS Code . . . 20
2.5 Authentification et Sécurité . . . 20
Conclusion . . . 20

3 Résultats de l'application **21**
Introduction . . . 21
3.1 Présentation des interfaces de l'application Web . . . 21
3.1.1 Page d'accueil . . . 21
3.1.2 Page d'authentification . . . 22
3.1.3 Tableau de bord de l'adhérent . . . 23
3.1.4 Souscription à un abonnement . . . 24
3.1.5 Message de succès après abonnement . . . 24
3.1.6 Sélection d'objectif par un adhérent . . . 25
3.1.7 Page de visualisation des plannings par l'adhérent . . . 25
3.1.8 Chat de discussion entre coachs et adhérents . . . 26
3.1.9 Tableau de bord de la secrétaire . . . 26
3.1.10 Tableau de bord du coach . . . 27

3.1.11 Gestion des plannings . . . 27
3.1.11.1 Création d'une séance par le coach . . . 27
3.1.11.2 Visualisation des séances créées par un coach . . . 28
3.1.11.3 Création d'une activité par un coach . . . 28
3.1.11.4 Interface de visualisation des activités créées par un coach . . . 29
3.1.12 Tableau de bord de l'administrateur . . . 29
3.2 Tests et Validation de l'Application . . . 30
3.2.1 Tests Unitaires . . . 30
3.2.1.1 Objectif des Tests Unitaires . . . 30
3.2.1.2 Méthodologie . . . 30
3.2.1.3 Résultats . . . 30
3.3 Tests Manuels . . . 31
3.3.1 Souscription à un abonnement . . . 31
3.3.2 Gestion des plannings . . . 31
3.3.3 Suivi des abonnements . . . 32
3.4 Conclusion des Tests . . . 32
3.5 Discussion . . . 32
3.5.1 Difficultés rencontrées . . . 32
3.5.2 Perspectives . . . 33
Conclusion . . . 33

Conclusion **34**

Bibliographie **35**

Webographie **36**

Table des matières **37**