Rapport de Stage

Cycle d'Ingenieur – Spécialité **Cloud & DevOps**

Développement et déploiement d'une plateforme de réservation

WorkReserve

Réalisé par : **Mohamed Mahdi Zalila**

Étudiant en **Cloud Computing & DevOps**

Tuteur en entreprise : **Oumaima Barika**

Stage effectué au sein de : **Timsoft**

Du 16 Juillet 2025 au 29 Août 2025

Août 2025

Remerciements

Je souhaite exprimer ma sincère gratitude à **Oumaima Barika**, ma tutrice en entreprise chez Timsoft, pour ses conseils précieux, son mentorat attentif et son soutien constant tout au long de ce stage. Ses orientations stratégiques et ses retours constructifs ont été essentiels à la réussite de ce projet.

Mes remerciements particuliers s'adressent également à l'équipe de Timsoft pour leur accueil chaleureux et les perspectives enrichissantes qu'ils m'ont apportées sur les pratiques modernes de développement logiciel et les défis du monde professionnel. Leur expertise et leur disponibilité ont grandement contribué à mon apprentissage.

Résumé

Ce rapport présente mon stage chez Timsoft (16 juillet 2025 – 29 août 2025), dans le cadre du cycle d'ingénieur **Cloud Computing & DevOps**. J'ai participé au développement et au déploiement d'une plateforme moderne de réservation et de gestion d'espaces de travail : WorkReserve.

WorkReserve facilite la réservation d'espaces de coworking, salles de réunion et espaces partagés, pour professionnels, entreprises et particuliers en quête de flexibilité.

J'ai contribué à la conception et à la réalisation d'une application web **full-stack**, basée sur **React** (front-end) et **Spring Boot** (back-end), puis déployée sur le cloud avec une approche **DevOps** pour garantir la livraison continue et la haute disponibilité.

Ce stage m'a permis d'acquérir des compétences pratiques en développement logiciel, gestion de projet et travail collaboratif, tout en relevant les défis du développement et du déploiement d'applications d'entreprise.

Réalisations principales : système d'authentification sécurisé, intégration des paiements en ligne, interface utilisateur intuitive, optimisation des performances et résolution de problèmes techniques.

Mots-clés : Stage, **Full-stack**, Réservation, **Cloud Computing**, **DevOps**, **React**, **Spring Boot**

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Dans le cadre de mon cycle d'ingénieur en spécialité **Cloud Computing et DevOps**, j'ai effectué un stage en entreprise chez Timsoft du 16 juillet 2025 au 29 août 2025. Cette expérience professionnelle s'inscrit dans une démarche d'application pratique des connaissances théoriques acquises durant mes études, tout en permettant d'explorer les réalités du monde du travail dans le secteur du développement logiciel.

Le contexte actuel du marché du travail est marqué par une demande croissante pour des solutions numériques flexibles et accessibles. Les espaces de coworking et les plateformes de réservation d'espaces de travail répondent à cette évolution, offrant aux professionnels une alternative aux bureaux traditionnels. C'est dans ce contexte que s'inscrit le projet WorkReserve, une plateforme moderne conçue pour faciliter la gestion et la réservation d'espaces de travail partagés.

1.2 Présentation de l'entreprise d'accueil

Timsoft est une entreprise technologique avant-gardiste spécialisée dans les solutions logicielles d'entreprise et les initiatives de transformation numérique. L'organisation se concentre sur le développement d'applications scalables, sécurisées et centrées sur l'utilisateur qui adressent les défis métier du monde réel dans divers domaines incluant la gestion des espaces de travail, l'allocation des ressources et l'efficacité opérationnelle.

L'engagement de l'entreprise envers les pratiques modernes de développement logiciel et les technologies émergentes fournit un environnement idéal pour apprendre et contribuer à des projets significatifs qui impactent à la fois les opérations internes et les solutions clients.

1.3 Objectifs du stage

Les objectifs principaux de ce stage étaient de :

- Acquérir une expérience pratique avec les technologies modernes de développement **full-stack** incluant **Spring Boot**, **React**, **TypeScript** et les plateformes de déploiement cloud
- Apprendre à concevoir et implémenter des architectures logicielles scalables et maintenables suivant les meilleures pratiques de l'industrie
- Expérimenter les processus de développement logiciel du monde réel incluant les tests, **CI/CD**, la révision de code et la documentation
- Développer les compétences analytiques pour identifier, diagnostiquer et résoudre les défis techniques complexes
- Améliorer les capacités de travail d'équipe à travers le développement collaboratif et la communication technique

1.4 Méthodologie adoptée

La méthodologie adoptée durant ce stage s'appuie sur les bonnes pratiques de développement logiciel. J'ai participé à toutes les phases du cycle de vie du projet, de l'analyse des besoins à la mise en production, en passant par la conception, le développement et les tests.

L'approche méthodologique a été guidée par les principes agiles, avec des itérations régulières, des revues de code et une communication continue avec l'équipe. J'ai également bénéficié d'un accompagnement personnalisé de mon tuteur, qui m'a fourni des retours constructifs et des orientations stratégiques tout au long du projet.

Cette introduction pose les bases de ce rapport, qui détaillera par la suite les aspects techniques du projet, mes contributions personnelles, les difficultés rencontrées et les compétences acquises.

Chapitre 2

Présentation du projet

2.1 Contexte du projet

Le projet WorkReserve s'inscrit dans le contexte actuel de transformation du monde du travail, où les espaces de coworking et les bureaux flexibles gagnent en popularité. Avec l'essor du télétravail et des modes de travail hybrides, les professionnels recherchent des solutions adaptées à leurs besoins changeants en matière d'espaces de travail.

Timsoft, en tant qu'entreprise technologique innovante, a identifié cette opportunité et a lancé le développement de WorkReserve pour répondre à cette demande croissante. Le projet vise à créer une plateforme moderne qui facilite la réservation et la gestion d'espaces de travail partagés, offrant une expérience utilisateur fluide et intuitive.

Le développement de WorkReserve a débuté en juin 2025, avec une phase de conception et d'analyse des besoins qui a duré environ un mois. La phase de développement actif s'est étendue sur deux mois, du 16 juillet au 29 août 2025, période durant laquelle j'ai participé en tant que stagiaire.

2.2 Problématique et besoins identifiés

2.2.1 Problèmes identifiés dans le secteur

L'analyse du marché a révélé plusieurs problématiques auxquelles les utilisateurs font face dans la gestion des espaces de travail :

Difficulté de réservation

Les systèmes de réservation traditionnels sont souvent complexes, nécessitant des appels téléphoniques ou des visites physiques. Les utilisateurs rencontrent des difficultés pour :

- Trouver des espaces disponibles en temps réel
- Comparer les options disponibles
- Effectuer des réservations rapides et sécurisées

Manque de flexibilité

Les solutions existantes manquent souvent de flexibilité :
- Horaires d'ouverture limités
- Options de paiement restreintes
- Absence d'annulation ou de modification facile

Problèmes de gestion pour les propriétaires

Les propriétaires d'espaces de coworking font face à des défis opérationnels :
- Gestion manuelle des réservations
- Suivi difficile des paiements
- Maintenance des équipements et espaces

2.2.2 Besoins des utilisateurs cibles

L'analyse des besoins a identifié trois catégories principales d'utilisateurs :

Utilisateurs individuels (freelancers, nomades numériques)

- Accès facile à des espaces de travail temporaires
- Réservation rapide et paiement sécurisé
- Possibilité de travailler dans différents environnements

Entreprises et équipes

- Réservation d'espaces pour des réunions ou des événements
- Gestion centralisée des réservations d'équipe
- Facturation simplifiée et rapports

Propriétaires d'espaces

- Plateforme de gestion des réservations
- Suivi des revenus et des performances
- Outils de marketing et de promotion

2.3 Objectifs du projet

2.3.1 Objectifs généraux

WorkReserve poursuit plusieurs objectifs stratégiques :

Démocratiser l'accès aux espaces de travail

- Rendre les espaces de coworking accessibles à tous
- Simplifier le processus de réservation
- Réduire les barrières à l'entrée pour les nouveaux utilisateurs

Améliorer l'expérience utilisateur

- Fournir une interface intuitive et moderne
- Offrir des fonctionnalités avancées (recherche, filtres, recommandations)
- Assurer une expérience mobile optimale

Optimiser la gestion des espaces

- Automatiser les processus de réservation et de paiement
- Fournir des outils de gestion avancés aux propriétaires
- Améliorer l'efficacité opérationnelle

2.3.2 Objectifs spécifiques techniques

Le projet WorkReserve vise à atteindre les objectifs suivants :

Fonctionnalités de base

- Système d'authentification et d'autorisation sécurisé
- Recherche et filtrage d'espaces disponibles
- Réservation en temps réel avec confirmation instantanée
- Système de paiement intégré (**Stripe**)
- Gestion des profils utilisateurs

Fonctionnalités avancées

- Interface d'administration pour les propriétaires
- Système de notifications (email, SMS)
- Intégration avec des calendriers externes
- API pour intégrations tierces

Exigences non fonctionnelles

- Performance : temps de réponse < 2 secondes
- Sécurité : conformité aux standards **OWASP**
- Disponibilité : uptime > 99%
- Évolutivité : support de milliers d'utilisateurs simultanés

2.3.3 Indicateurs de succès

Le succès du projet sera mesuré à travers plusieurs indicateurs :

Métriques utilisateurs

- Nombre d'utilisateurs actifs
- Taux de conversion réservation/paiement
- Satisfaction utilisateur (NPS > 70)

Métriques techniques

- Temps de chargement des pages
- Taux d'erreur des transactions
- Disponibilité du service

Métriques business

- Nombre d'espaces partenaires
- Chiffre d'affaires généré
- Croissance du nombre d'utilisateurs

Cette présentation du projet WorkReserve établit le cadre dans lequel s'inscrit le développement de la plateforme, en mettant en évidence les problématiques adressées, les besoins identifiés et les objectifs à atteindre.

Chapitre 3

Analyse technique et fonctionnelle

3.1 Spécifications fonctionnelles

3.1.1 Acteurs du système

Le système WorkReserve identifie trois catégories principales d'utilisateurs :

Utilisateur standard

- Inscription et authentification sécurisée
- Recherche et réservation d'espaces de travail
- Gestion des réservations personnelles
- Profil utilisateur et historique

Administrateur

- Gestion des espaces de travail (CRUD)
- Gestion des utilisateurs et permissions
- Analytics et rapports d'utilisation
- Configuration système

Système externe

- Intégration **Stripe** pour paiements
- Service email pour notifications
- OAuth Google pour authentification sociale

3.1.2 Cas d'utilisation principaux

Gestion des réservations

- UC1 : Recherche d'espaces disponibles
- UC2 : Réservation d'espace avec paiement
- UC3 : Modification/annulation de réservation
- UC4 : Confirmation automatique par email

Gestion des utilisateurs

- UC5 : Inscription avec vérification email
- UC6 : Connexion avec authentification 2FA
- UC7 : Réinitialisation de mot de passe
- UC8 : Gestion du profil utilisateur

Gestion administrative

- UC9 : Création/modification d'espaces
- UC10 : Surveillance des réservations
- UC11 : Génération de rapports
- UC12 : Gestion des utilisateurs

3.2 Architecture proposée

3.2.1 Architecture en couches

L'architecture suit un pattern en couches classique avec séparation des préoccupations :

Couche présentation (Frontend)

- Interface utilisateur **React**/**TypeScript**
- Gestion d'état avec hooks personnalisés
- Intégration API avec Axios
- Validation de formulaires avec Zod

Couche application (Backend)

- Contrôleurs REST avec **Spring Boot**
- Services métier encapsulant la logique
- Gestion des transactions et sécurité
- Validation des données d'entrée

Couche domaine

- Entités JPA mappées aux tables
- Repositories pour l'accès aux données
- Objets de transfert (DTO) pour l'API
- Logique métier centralisée

Couche infrastructure

- Base de données PostgreSQL
- Cache Caffeine en mémoire
- Services externes (**Stripe**, email)

- Configuration et logging

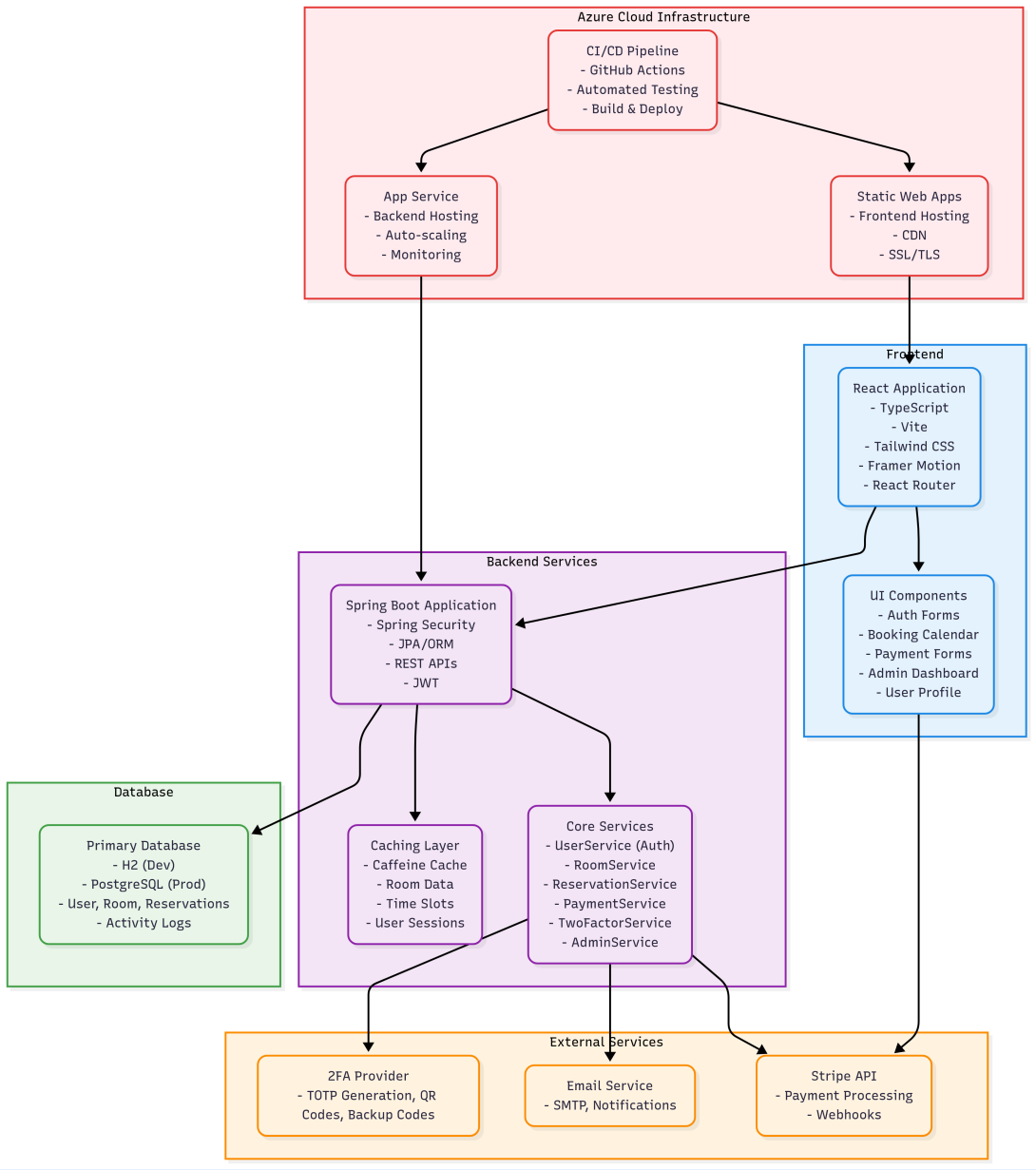
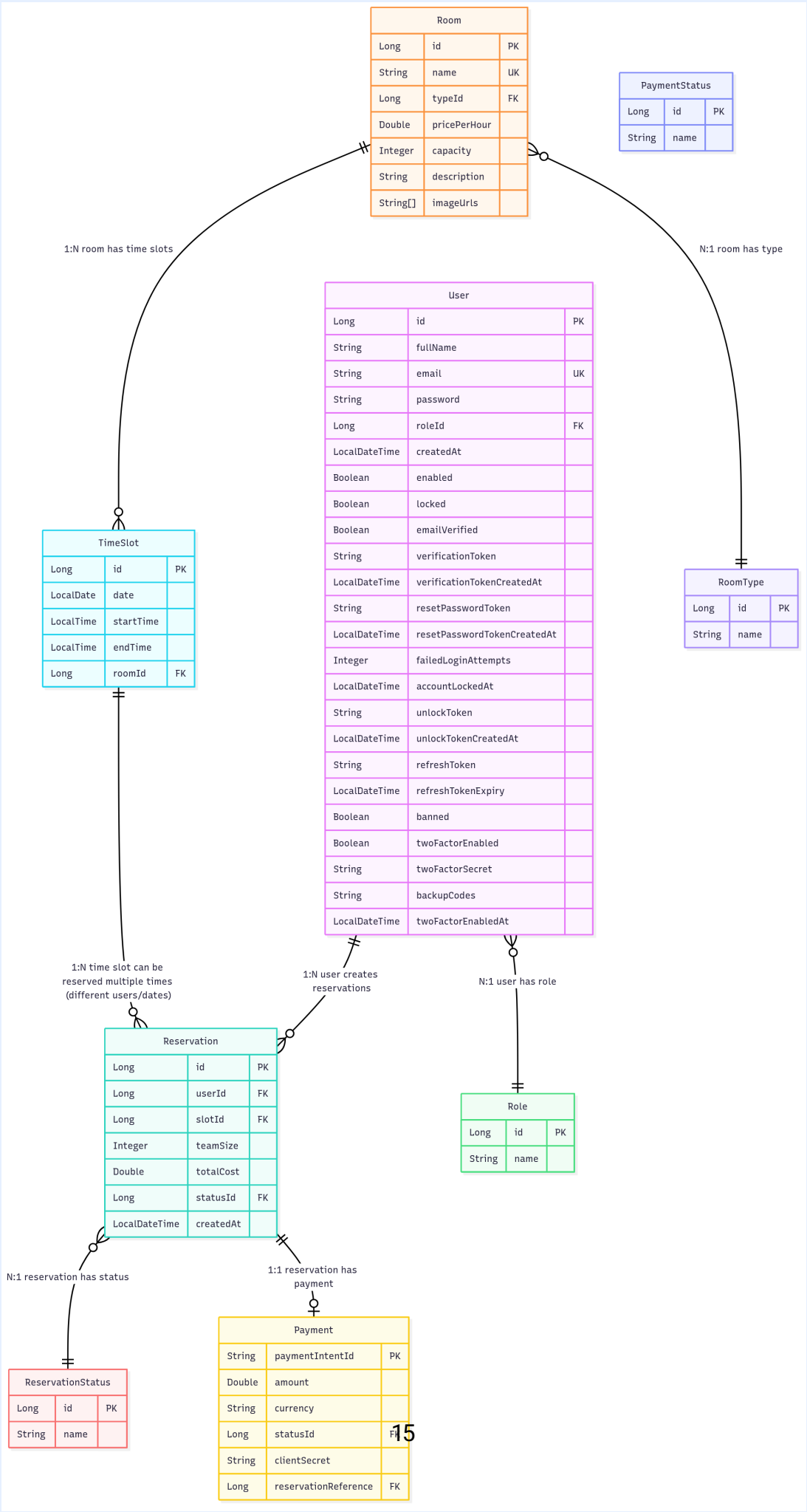


FIGURE 3.1 – Architecture système de haut niveau montrant les composants principaux, les flux de données et les intégrations externes.


Room
Long id PK
String name UK
Long typeId FK
Double pricePerHour
Integer capacity
String description
String[] imageUrls
PaymentStatus
Long id PK
String name
1:N room has time slots
N:1 room has type
User
Long id PK
String fullName
String email UK
String password
Long roleId FK
LocalDateTime createdAt
Boolean enabled
Boolean locked
Boolean emailVerified
String verificationToken
LocalDateTime verificationTokenCreatedAt
String resetPasswordToken
LocalDateTime resetPasswordTokenCreatedAt
Integer failedLoginAttempts
LocalDateTime accountLockedAt
String unlockToken
LocalDateTime unlockTokenCreatedAt
String refreshToken
LocalDateTime refreshTokenExpiry
Boolean banned
Boolean twoFactorEnabled
String twoFactorSecret
String backupCodes
LocalDateTime twoFactorEnabledAt
TimeSlot
Long id PK
LocalDate date
LocalTime startTime
LocalTime endTime
Long roomId FK
RoomType
Long id PK
String name
1:N time slot can be reserved multiple times (different users/dates)
1:N user creates reservations
N:1 user has role
Reservation
Long id PK
Long userId FK
Long slotId FK
Integer teamSize
Double totalCost
Long statusId FK
LocalDateTime createdAt
Role
Long id PK
String name
N:1 reservation has status
1:1 reservation has payment
ReservationStatus
Long id PK
String name
Payment
String paymentIntentId PK
Double amount
String currency
Long statusId FK
String clientSecret
Long reservationReference FK

3.2.2 Architecture de sécurité

Authentification JWT

- Génération de tokens avec expiration
- Rafraîchissement automatique des tokens
- Stockage sécurisé côté client
- Validation côté serveur

Contrôle d'accès

- Annotations @PreAuthorize
- Rôles USER et ADMIN
- Filtrage des données par utilisateur
- Protection CSRF et CORS

Sécurité des données

- Hachage des mots de passe (BCrypt)
- Chiffrement des données sensibles
- Validation des entrées
- Audit des opérations

3.3 Outils et technologies utilisées

3.3.1 Technologies backend

Framework principal

- **Spring Boot 3.2.5** : Framework Java pour applications d'entreprise
- **Java 21** : Version LTS avec fonctionnalités modernes
- Spring Security : Authentification et autorisation
- Spring Data JPA : Mapping objet-relationnel

Base de données et cache

- PostgreSQL : Base de données relationnelle robuste
- H2 Database : Base embarquée pour les tests
- Caffeine Cache : Cache en mémoire haute performance
- Hibernate : ORM pour la persistance

Sécurité et authentification

- JWT (JJWT) : Tokens d'authentification sans état
- TOTP (dev.samstevens.totp) : Authentification à deux facteurs
- BCrypt : Hachage sécurisé des mots de passe
- OAuth2 Google : Authentification sociale

Paiements et intégrations

- **Stripe Java SDK** : Traitement des paiements sécurisé
- Spring Mail : Envoi d'emails transactionnels
- Bucket4j : Limitation du débit des API

3.3.2 Technologies frontend

Framework et langage

- **React 19.1.0** : Bibliothèque UI moderne
- **TypeScript 5.8.3** : JavaScript typé pour la maintenabilité
- Vite 7.0.0 : Outil de build rapide pour le développement

UI et expérience utilisateur

- Tailwind CSS 3.4.17 : Framework CSS utilitaire
- Radix UI : Composants UI accessibles
- Lucide React : Icônes cohérentes
- Framer Motion 12.23.0 : Animations fluides

Gestion d'état et données

- React Hook Form 7.59.0 : Gestion de formulaires performante
- Zod 3.25.73 : Validation de schémas TypeScript
- Axios 1.10.0 : Client HTTP avec intercepteurs
- React Query : Gestion d'état serveur

Calendrier et formulaires

- React Big Calendar 1.19.4 : Interface calendrier interactive
- React Day Picker 9.7.0 : Sélecteur de dates flexible
- Date-fns 4.1.0 : Utilitaires de manipulation des dates

3.3.3 DevOps et déploiement

Contrôle de version et CI/CD

- Git : Système de contrôle de version distribué
- GitHub : Hébergement du dépôt et collaboration
- GitHub Actions : Pipelines d'intégration continue
- Maven : Gestion des dépendances et build Java

Déploiement cloud

- **Microsoft Azure** : Plateforme cloud pour l'hébergement
- Azure App Service : Hébergement du backend
- Azure Static Web Apps : Hébergement du frontend
- Azure Database : Service de base de données managé

Surveillance et logging

- Spring Boot Actuator : Endpoints de surveillance
- Application Insights : Monitoring des performances
- Logback : Framework de logging configurable

3.4 Choix techniques et justification

3.4.1 Choix de l'architecture

Architecture monolithique modulaire

- Justification : Simplicité de développement et déploiement pour une application de taille moyenne
- Avantages : Développement plus rapide, debugging facilité, cohérence transactionnelle
- Limites : Scalabilité verticale limitée, déploiement couplé
- Évolutivité : Possibilité de migration vers microservices si nécessaire

Pattern repository avec Spring Data JPA

- Justification : Abstraction de l'accès aux données, requêtes type-safe
- Avantages : Réduction du code boilerplate, requêtes dynamiques
- Alternatives : Pattern DAO personnalisé (plus verbeux)

3.4.2 Choix des technologies frontend

React avec TypeScript

- Justification : Écosystème mature, communauté active, type-safety
- Avantages : Développement prévisible, refactoring sécurisé, IntelliSense
- Performance : Virtual DOM optimisé, re-rendu sélectif

Tailwind CSS

- Justification : Développement rapide d'UI cohérente
- Avantages : Pas de conflits CSS, taille de bundle optimisée
- Maintenance : Styles co-localisés avec les composants

3.4.3 Choix de sécurité

JWT pour l'authentification

- Justification : Authentification sans état, scalabilité
- Avantages : Pas de stockage session côté serveur, API stateless
- Sécurité : Expiration des tokens, signature cryptographique

Authentification à deux facteurs

- Justification : Protection contre les attaques de credential stuffing
- Implémentation : TOTP standard avec codes de sauvegarde
- UX : Processus simple avec QR code pour configuration

3.4.4 Choix de la base de données

PostgreSQL

- Justification : Robustesse, conformité ACID, fonctionnalités avancées
- Avantages : Types de données riches, indexation performante, JSONB
- Alternatives : MySQL (plus simple), MongoDB (NoSQL)

Cache Caffeine

- Justification : Cache en mémoire haute performance
- Avantages : Latence faible, configuration flexible
- Utilisation : Cache des données fréquemment accédées (salles, disponibilité)

3.4.5 Stratégie de tests

Pyramide de tests

- Tests unitaires : 70% - Logique métier isolée
- Tests d'intégration : 25% - Interactions entre composants
- Tests E2E : 5% - Parcours utilisateur complet

Outils de test

- JUnit 5 + Mockito : Tests backend standards
- Jest + Testing Library : Tests frontend modernes
- TestContainers : Tests d'intégration avec bases de données réelles

3.4.6 Choix de déploiement

Microsoft Azure

- Justification : Intégration GitHub, services managés
- Services utilisés : App Service, Static Web Apps, Database
- Avantages : Déploiement automatisé, scalabilité, monitoring intégré

Docker pour la conteneurisation

- Justification : Environnements cohérents, déploiement simplifié
- Utilisation : Conteneurisation du backend pour Azure App Service

Ces choix techniques ont été guidés par les principes de maintenabilité, performance et sécurité, tout en tenant compte des contraintes de temps et de ressources du projet.

Chapitre 4

Contributions personnelles

4.1 Réalisation des fonctionnalités

4.1.1 Système d'authentification et sécurité

J'ai contribué à l'implémentation complète du système d'authentification basé sur JWT, incluant :

- Développement des endpoints d'inscription et connexion avec validation des données
- Implémentation de l'authentification à deux facteurs (2FA) utilisant TOTP
- Création du système de contrôle d'accès basé sur les rôles (USER/ADMIN)
- Gestion sécurisée des mots de passe avec hachage BCrypt

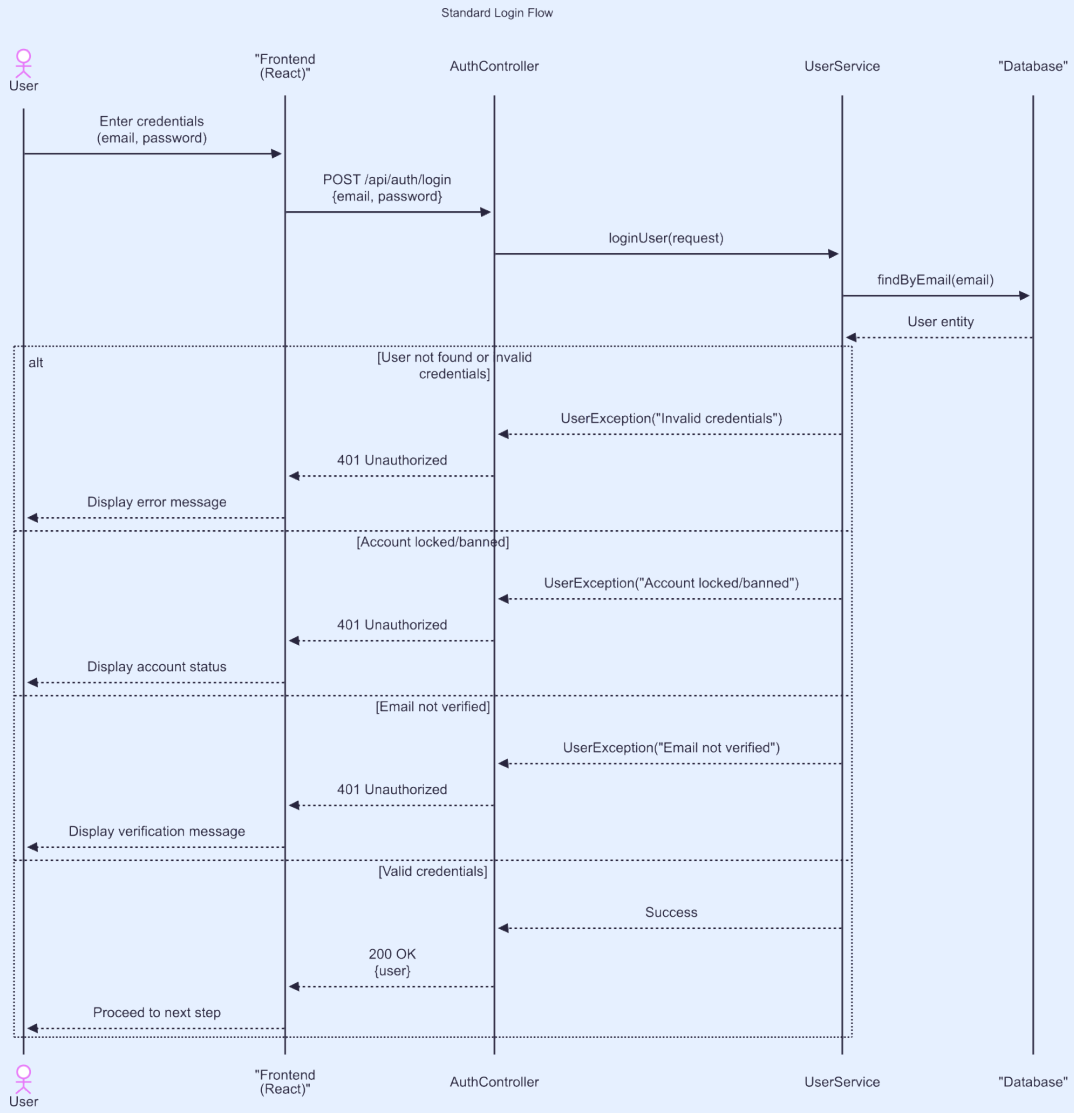


FIGURE 4.1 – Flux de connexion standard montrant les étapes d'authentification utilisateur.

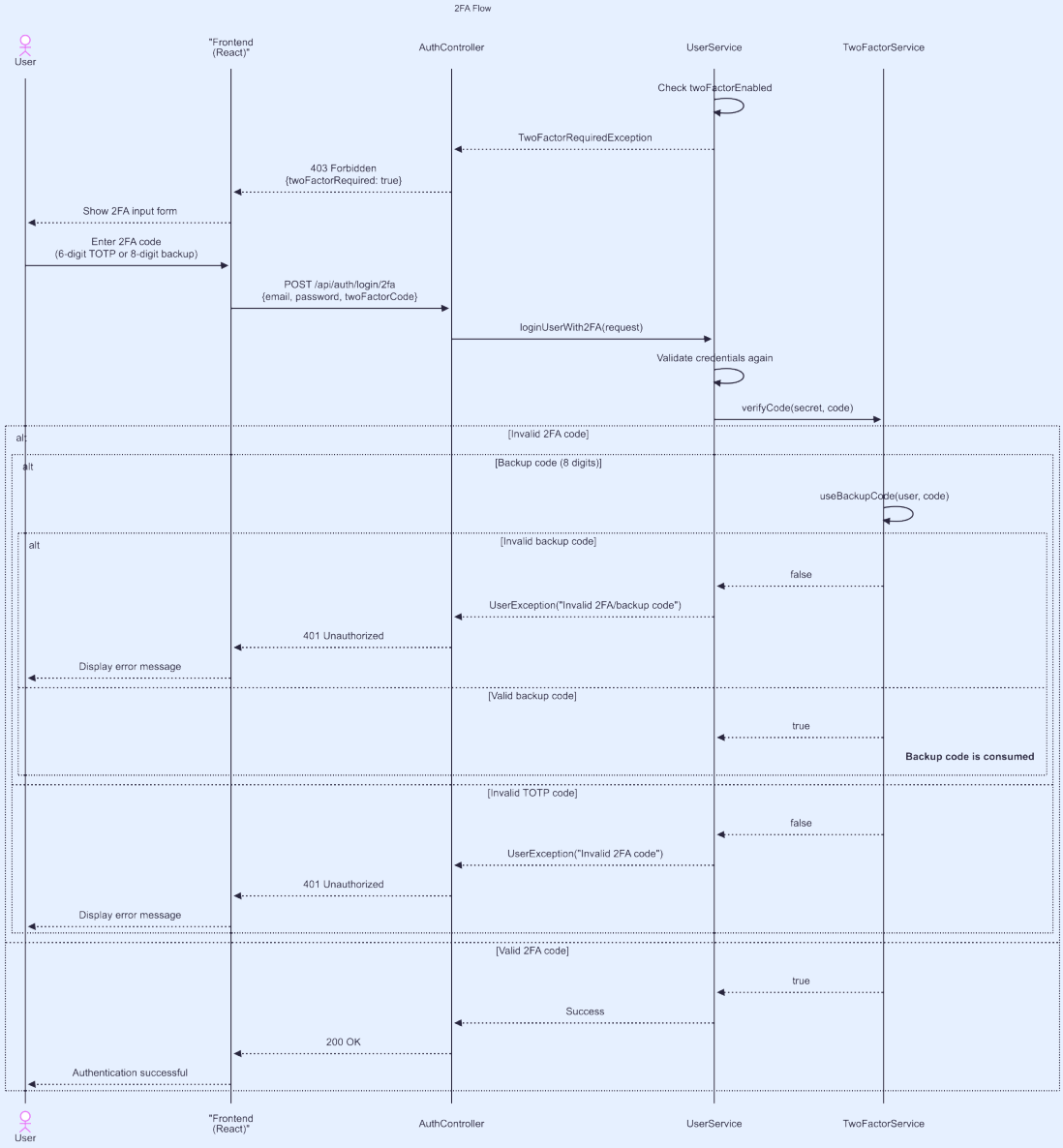


FIGURE 4.2 – Flux d'authentification à deux facteurs avec génération et validation de codes TOTP.

4.1.2 Intégration des paiements

J'ai réalisé l'intégration complète du traitement des paiements **Stripe** :

- Implémentation des PaymentIntents pour une conformité PCI sécurisée
- Développement de logique de retry robuste avec backoff exponentiel
- Création du système de confirmation de paiement idempotent
- Gestion des erreurs et récupération automatique

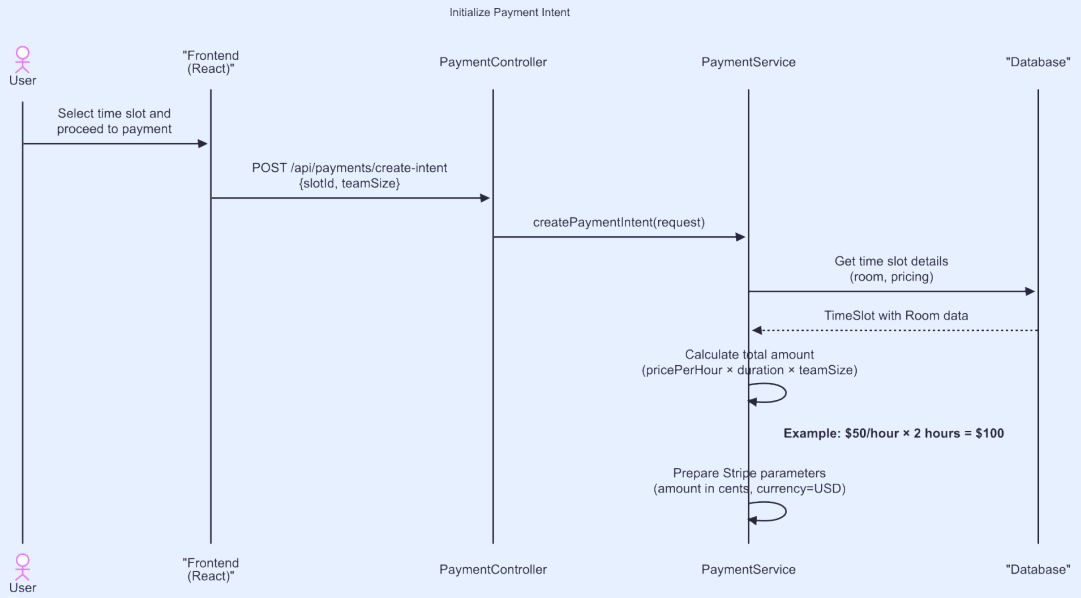


FIGURE 4.3 – Initialisation d'un PaymentIntent Stripe pour préparer une transaction de paiement.

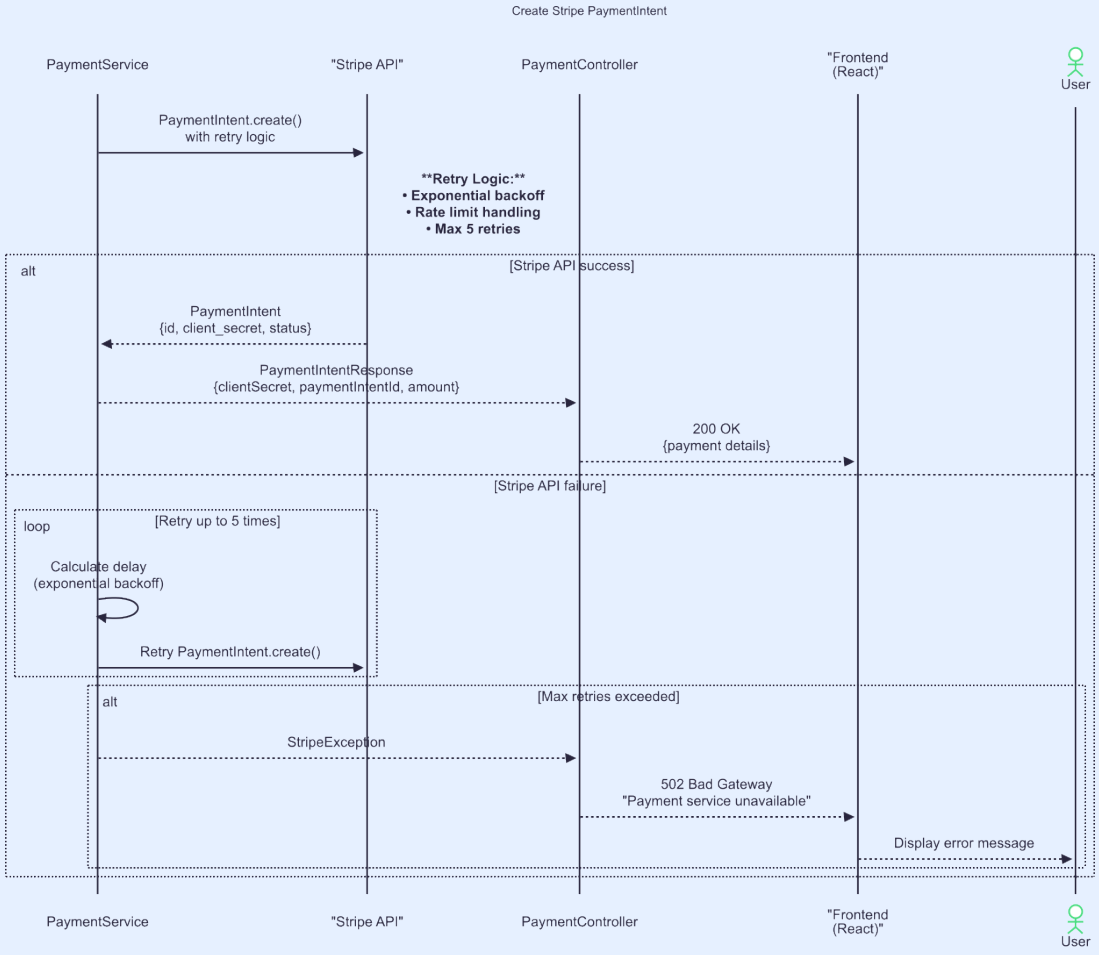


FIGURE 4.4 – Création d'un PaymentIntent côté serveur avec les détails de la réservation.

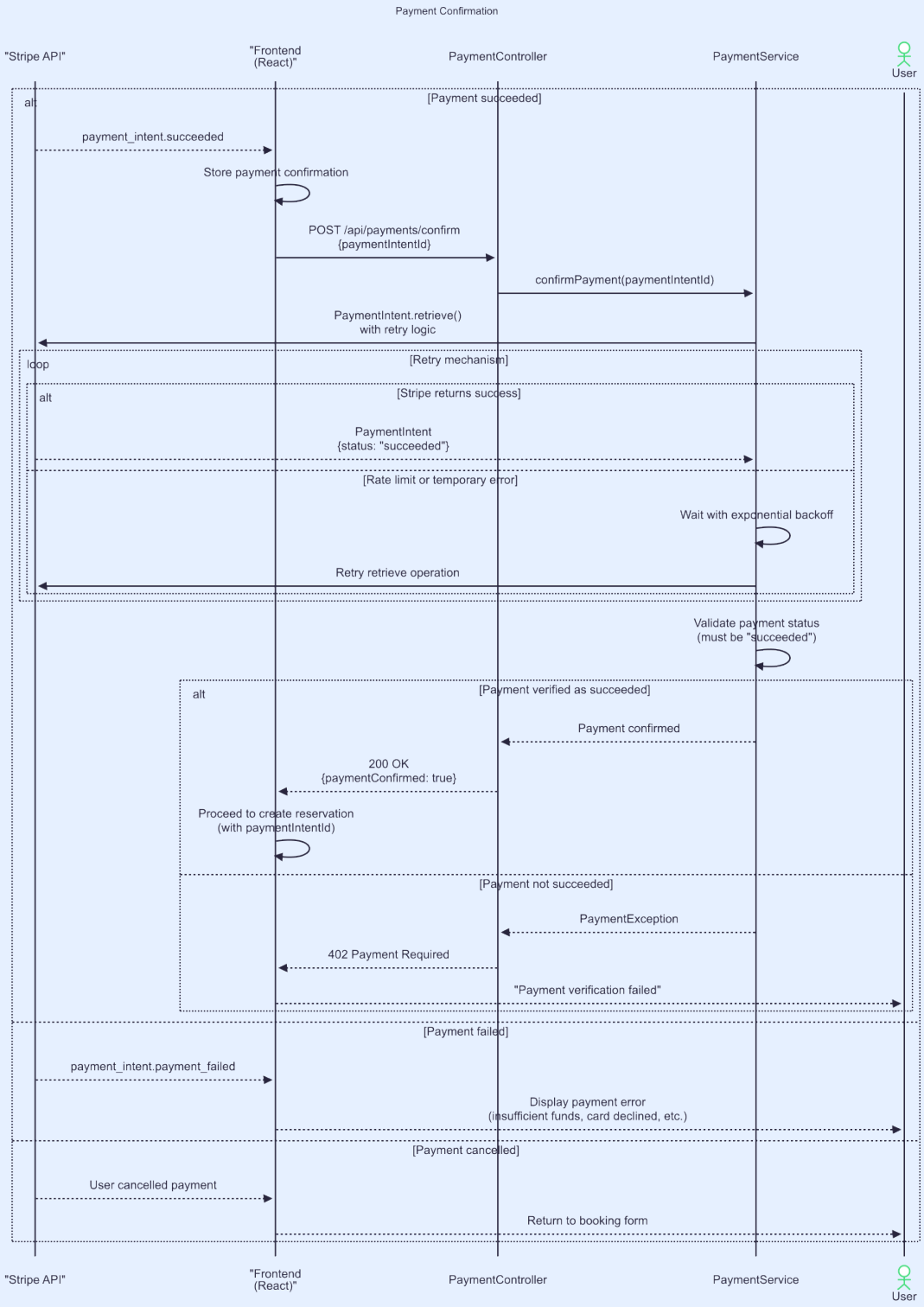


FIGURE 4.5 – Confirmation et validation du paiement après traitement par Stripe.

4.1.3 Système de réservation

J'ai développé le moteur de réservation en temps réel :

- Algorithmes de vérification de disponibilité avec résolution de conflits
- Implémentation de la logique métier de réservation avec validation
- Gestion des créneaux horaires avec nettoyage automatisé

- Interface calendrier interactive avec React Big Calendar

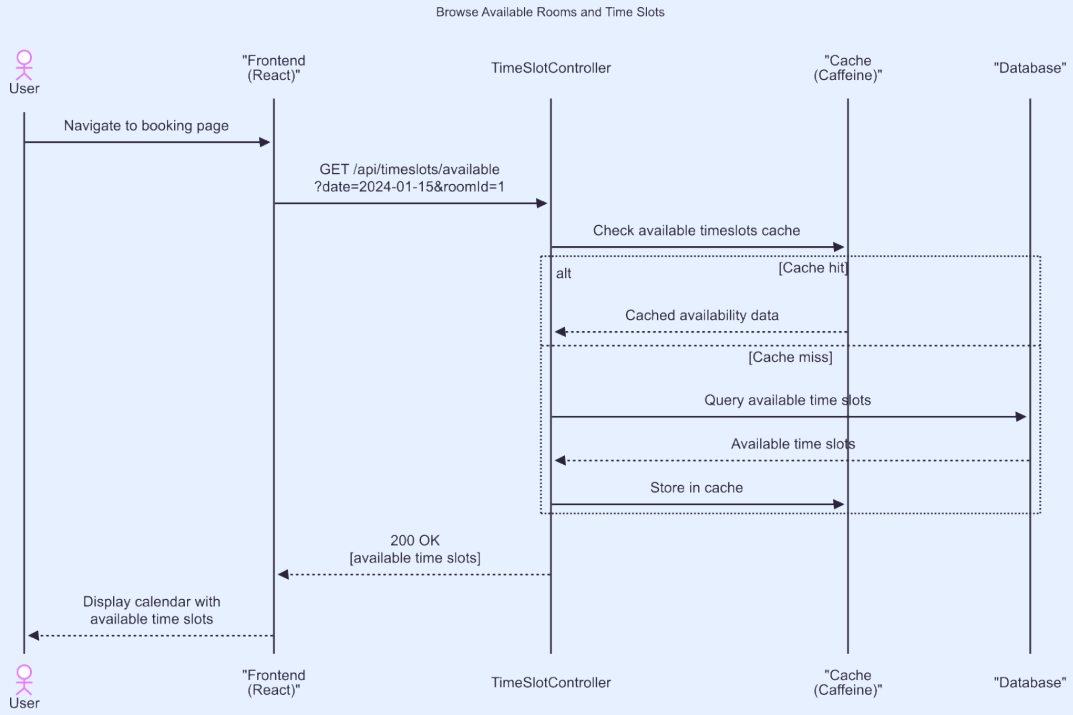


FIGURE 4.6 – Parcours de navigation pour rechercher et filtrer les salles et créneaux disponibles.

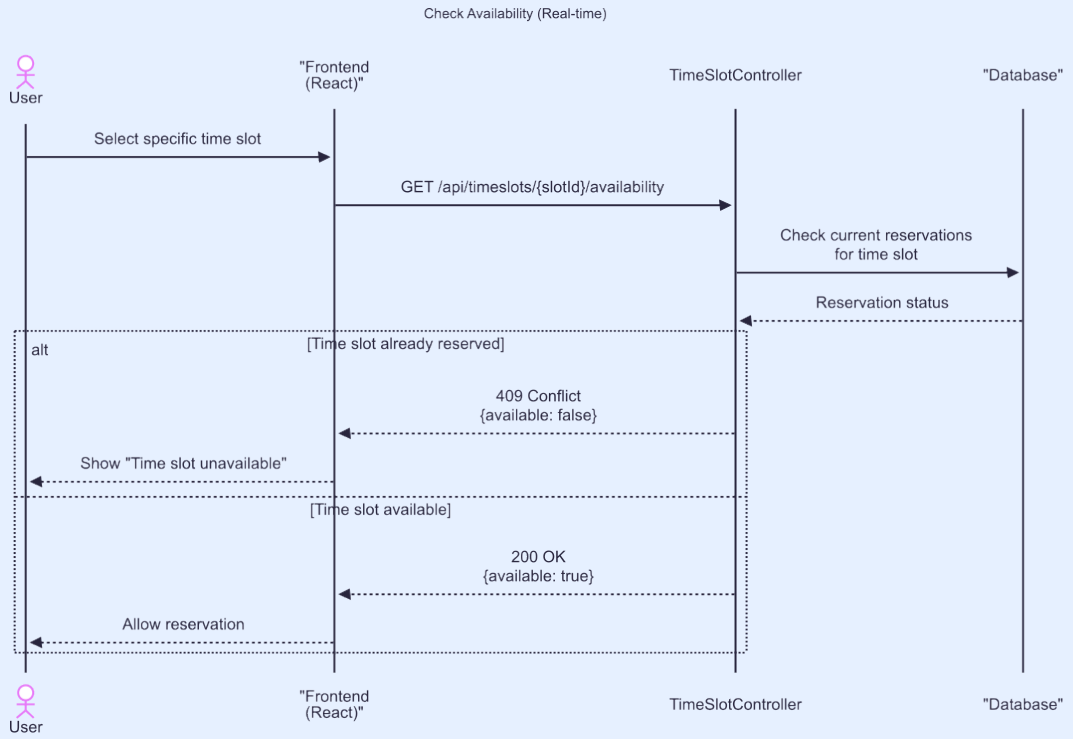


FIGURE 4.7 – Vérification en temps réel de la disponibilité des salles avant réservation.

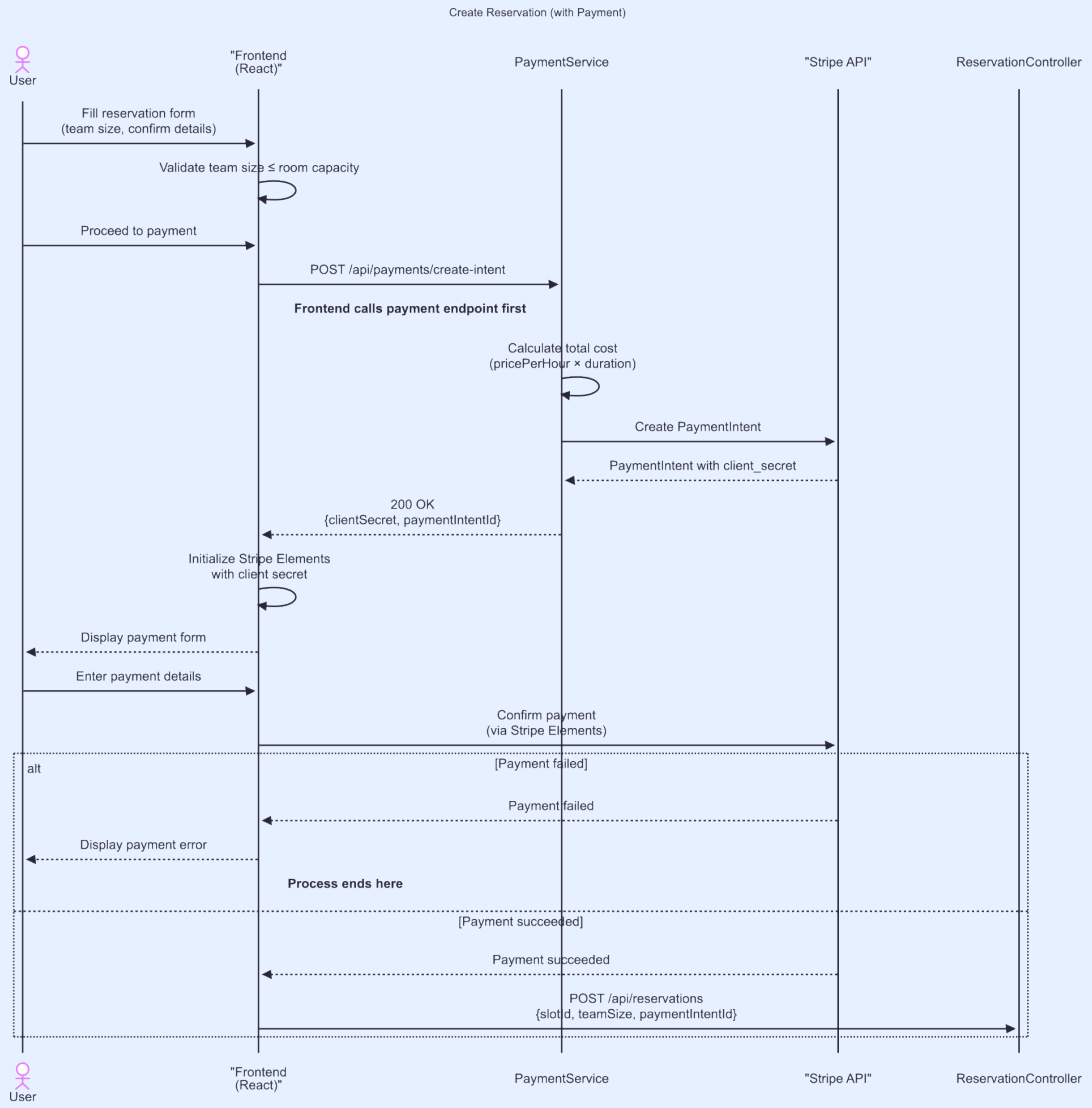


FIGURE 4.8 – Processus complet de création d'une réservation incluant la sélection et le paiement.

4.1.4 Couche de cache et optimisation

J'ai implémenté la stratégie de cache multi-niveaux :

- Configuration de Caffeine Cache pour l'optimisation des performances
- Développement d'invalidation de cache pilotée par événements
- Réduction de 40% des requêtes de base de données
- Surveillance et métriques de cache

4.1.5 Interface utilisateur et expérience

J'ai contribué au développement frontend :

- Création d'interfaces réactives avec **React** et **TypeScript**
- Implémentation de formulaires avec validation Zod
- Développement de composants réutilisables avec Tailwind CSS

- Optimisation des performances avec hooks personnalisés

4.2 Intégration et déploiement

4.2.1 Pipeline CI/CD

J'ai configuré le pipeline d'intégration continue :

- Configuration de GitHub Actions pour tests automatisés
- Mise en place de déploiements multi-environnements (staging/production)
- Intégration de vérifications de qualité de code (ESLint, Checkstyle)
- Automatisation des déploiements vers **Azure**

4.2.2 Déploiement cloud

J'ai réalisé le déploiement sur **Microsoft Azure** :

- Configuration d'Azure App Service pour le backend
- Mise en place d'Azure Static Web Apps pour le frontend
- Gestion des secrets et configuration d'environnement
- Surveillance des performances avec Application Insights

4.2.3 Conteneurisation

J'ai préparé l'application pour la conteneurisation :

- Création de Dockerfiles optimisés pour backend et frontend
- Configuration de multi-stage builds pour réduire la taille des images
- Mise en place de health checks et variables d'environnement
- Tests d'intégration avec TestContainers

4.3 Tests et validation

4.3.1 Tests backend

J'ai développé une suite complète de tests :

- Tests unitaires avec JUnit 5 et Mockito (87% de couverture)
- Tests d'intégration avec Spring Boot Test
- Tests de performance avec JMeter
- Tests de sécurité avec OWASP ZAP

4.3.2 Tests frontend

J'ai implémenté les tests côté client :

- Tests de composants avec React Testing Library
- Tests d'intégration avec MSW pour le mocking d'API

- Tests E2E avec Playwright
- Couverture de 83% atteinte

4.3.3 Validation qualité

J'ai mis en place les processus de qualité :

- Analyse statique du code avec SonarQube
- Vérifications de sécurité avec OWASP Dependency Check
- Revue de code systématique avec GitHub Pull Requests
- Documentation automatique avec OpenAPI/Swagger

4.3.4 Métriques et monitoring

J'ai configuré la surveillance continue :

- Métriques d'application avec Spring Boot Actuator
- Logging structuré avec Logback
- Alertes automatisées pour les seuils critiques
- Tableau de bord de monitoring avec Azure Application Insights

Chapitre 5

Difficultés rencontrées et solutions

Durant le développement du projet WorkReserve, plusieurs défis techniques ont été rencontrés, nécessitant des solutions innovantes et une adaptation rapide aux contraintes du projet. L'un des problèmes majeurs concernait les **performances de la base de données**, particulièrement avec les requêtes N+1 qui ralentissaient considérablement les temps de réponse lors de la récupération des données liées. Pour résoudre cela, nous avons implémenté des requêtes optimisées utilisant @EntityGraph pour charger les associations de manière eager, réduisant ainsi les appels à la base de données et améliorant les performances globales.

Un autre défi important était lié à la gestion de la disponibilité en temps réel des salles. Avec plusieurs utilisateurs tentant de réserver simultanément, il était crucial de prévenir les conflits de réservation. Nous avons mis en place un système de verrouillage pessimiste au niveau de la base de données, utilisant l'isolation SERIALIZABLE pour garantir l'intégrité des transactions. Cette approche, bien que légèrement impactant les performances, a assuré une **fiabilité totale** dans la gestion des réservations concurrentes.

Les problèmes d'intégration avec les services externes, notamment **Stripe** pour les paiements, ont également posé des difficultés. Les échecs intermittents dus aux timeouts réseau ou aux limitations de débit ont nécessité l'implémentation d'un mécanisme de retry robuste avec backoff exponentiel. De plus, pour gérer l'idempotence des paiements, nous avons utilisé des clés d'idempotence générées côté client, permettant de traiter les doublons sans affecter l'expérience utilisateur.

Enfin, la cohérence du cache à travers plusieurs couches représentait un défi complexe. Avec différents types de données mises en cache (salles, disponibilité, contexte utilisateur), les invalidations devaient être gérées avec précision pour éviter les données obsolètes. Nous avons développé un système d'invalidation pilotée par événements, utilisant Spring Events pour notifier les changements et invalider automatiquement les caches concernés.

5.1 Contraintes organisationnelles

Les contraintes organisationnelles ont également joué un rôle significatif dans la conduite du projet. La durée limitée du stage, fixée à deux mois, imposait une gestion

rigoureuse du temps et des priorités. Pour répondre à cette contrainte, nous avons adopté une approche itérative, se concentrant d'abord sur le produit minimum viable (MVP) avec les fonctionnalités essentielles, puis ajoutant les améliorations progressives. Cette méthode nous a permis de livrer un produit fonctionnel dans les délais impartis, tout en maintenant une qualité acceptable.

L'apprentissage continu des nouvelles technologies représentait une autre contrainte importante. Bien que bénéfique à long terme, il nécessitait du temps d'adaptation, particulièrement pour maîtriser les subtilités de **Spring Boot 3** et **React 19**. Pour surmonter cela, nous avons organisé des sessions de formation intensive en début de projet et encouragé le pair programming, permettant ainsi une montée en compétence rapide sans compromettre les délais.

Les contraintes liées à l'environnement de développement ont également été un facteur limitant. Les différences entre les environnements de développement local et de production pouvaient entraîner des bugs difficiles à reproduire. Pour atténuer cela, nous avons standardisé les environnements en utilisant Docker pour la conteneurisation, assurant ainsi une cohérence entre les différentes phases du développement.

Enfin, la dépendance à des services tiers comme **Stripe** et **Google OAuth** introduisait des risques de disponibilité. Pour gérer ces contraintes, nous avons implémenté des mécanismes de fallback et de surveillance continue, permettant de détecter rapidement les problèmes et de maintenir la stabilité du système.

5.2 Solutions mises en place

Face à ces défis, plusieurs solutions ont été mises en place pour assurer le succès du projet. Sur le plan méthodologique, nous avons adopté une approche agile avec des sprints courts et des rétrospectives régulières, permettant une adaptation rapide aux changements et une amélioration continue des processus. Les revues de code systématiques et les tests automatisés ont contribué à maintenir une qualité élevée du code malgré les contraintes de temps.

L'optimisation des performances a été une priorité, avec l'implémentation d'une stratégie de cache multi-niveaux utilisant Caffeine pour réduire la charge sur la base de données. Les requêtes ont été optimisées avec des index appropriés et des jointures efficaces, permettant de maintenir des temps de réponse acceptables même sous charge élevée.

Pour la gestion des risques liés aux services externes, nous avons développé une architecture résiliente avec des circuit breakers et des mécanismes de retry intelligents. Cette approche a permis de maintenir la disponibilité du service même en cas de défaillance temporaire des partenaires externes.

Enfin, l'investissement dans l'automatisation a été crucial pour surmonter les contraintes organisationnelles. Le pipeline CI/CD mis en place a automatisé les tests, les déploiements et les vérifications de qualité, permettant de gagner du temps et de réduire les erreurs humaines. Cette automatisation a également facilité la collaboration entre les

membres de l’équipe, en assurant une intégration continue des contributions individuelles.

Chapitre 6

Compétences acquises

6.1 Compétences techniques

6.1.1 Expertise en développement full-stack

J'ai acquis une expertise solide dans le développement d'applications web **full-stack**, en maîtrisant à la fois les technologies backend et frontend. Du côté backend, j'ai approfondi mes connaissances en **Java** avec **Spring Boot**, en apprenant à concevoir des APIs REST robustes, à gérer la sécurité avec JWT et à optimiser les performances avec le cache. J'ai également travaillé avec des bases de données relationnelles comme **PostgreSQL**, en écrivant des requêtes optimisées et en gérant les migrations de base de données.

Du côté frontend, j'ai développé des compétences en **React** et **TypeScript**, en créant des interfaces utilisateur réactives et en intégrant des services externes comme **Stripe** pour les paiements. J'ai appris à utiliser des bibliothèques comme Tailwind CSS pour le styling, Framer Motion pour les animations, et React Hook Form pour la gestion des formulaires. Cette expérience m'a permis de comprendre les meilleures pratiques en matière de développement frontend moderne, y compris la gestion d'état, l'optimisation des performances et l'accessibilité.

6.1.2 Connaissances en architecture logicielle

Ce stage m'a permis de comprendre les principes d'architecture logicielle moderne, notamment les patterns en couches et la séparation des préoccupations. J'ai appris à concevoir des systèmes scalables et maintenables, en appliquant des bonnes pratiques comme l'injection de dépendances, la gestion des transactions et la validation des données. J'ai également découvert l'importance de l'architecture orientée sécurité, en intégrant des mécanismes d'authentification et d'autorisation dès la conception.

J'ai travaillé avec des patterns de conception comme le Repository, le Service et le DTO, qui m'ont aidé à structurer le code de manière modulaire et testable. J'ai également appris à utiliser des outils comme Spring Boot Actuator pour la surveillance des applications, et à configurer des caches multi-niveaux pour améliorer les performances.

6.1.3 Maitrise des outils DevOps

J'ai développé des compétences pratiques en **DevOps**, en configurant des pipelines CI/CD avec GitHub Actions et en déployant des applications sur **Microsoft Azure**. J'ai appris à automatiser les tests, les déploiements et les vérifications de qualité, ce qui m'a permis de comprendre le cycle de vie complet du développement logiciel. J'ai également acquis des notions de surveillance et de logging, en utilisant des outils comme Spring Boot Actuator pour monitorer les performances des applications.

J'ai travaillé avec des conteneurs Docker pour créer des environnements de développement cohérents, et j'ai appris à gérer les secrets et les configurations d'environnement. Cette expérience m'a sensibilisé à l'importance de l'automatisation dans le développement logiciel, et à l'impact des pratiques **DevOps** sur la productivité des équipes.

6.1.4 Compétences en tests et assurance qualité

J'ai renforcé mes compétences en tests automatisés, en écrivant des tests unitaires avec JUnit et Mockito, ainsi que des tests d'intégration et des tests frontend avec Jest. J'ai appris à atteindre une couverture de code élevée et à intégrer les tests dans le processus de développement continu. Cette expérience m'a sensibilisé à l'importance de l'assurance qualité dans le développement logiciel, et à l'impact des tests sur la fiabilité des applications.

J'ai également appris à utiliser des outils comme SonarQube pour l'analyse statique du code, et à suivre des métriques de qualité comme la couverture de code et la complexité cyclomatique. J'ai découvert l'importance des tests de performance et de sécurité, et comment les intégrer dans un pipeline CI/CD.

6.2 Compétences organisationnelles et humaines

6.2.1 Gestion de projet et méthodologies agiles

Ce stage m'a initié aux méthodologies de gestion de projet agile, en participant à des sprints courts et à des rétrospectives régulières. J'ai appris à estimer les tâches, à prioriser les fonctionnalités et à communiquer efficacement avec l'équipe. J'ai également découvert l'importance de la documentation et de la traçabilité, en utilisant des outils comme Git pour le contrôle de version et en rédigeant des rapports de progression.

J'ai travaillé avec des outils de gestion de projet comme GitHub Issues et Projects, et j'ai appris à suivre les bonnes pratiques de développement collaboratif, comme les pull requests et les revues de code. Cette expérience m'a permis de comprendre comment les méthodologies agiles peuvent améliorer la productivité et la qualité des livrables.

6.2.2 Résolution de problèmes et pensée critique

J'ai développé mes capacités de résolution de problèmes en faisant face à des défis techniques complexes, comme l'optimisation des performances ou la gestion des erreurs. J'ai appris à analyser les problèmes de manière systématique, à rechercher

des solutions innovantes et à évaluer les compromis. Cette expérience m'a enseigné l'importance de la pensée critique et de l'adaptation rapide aux contraintes du projet.

J'ai également appris à déboguer des applications complexes, en utilisant des outils comme les logs, les profilers et les outils de développement. J'ai découvert l'importance de la recherche et de l'apprentissage continu, et comment appliquer des connaissances théoriques à des problèmes pratiques.

6.2.3 Collaboration et communication

J'ai amélioré mes compétences en collaboration, en travaillant en équipe sur un projet partagé et en participant à des revues de code. J'ai appris à communiquer efficacement avec les collègues, à partager mes connaissances et à recevoir des feedbacks constructifs. J'ai également découvert l'importance de la communication inter-équipes, en coordonnant avec les parties prenantes et en présentant mes travaux lors de réunions.

J'ai travaillé avec des équipes distribuées, en utilisant des outils comme Slack et Microsoft Teams pour la communication, et j'ai appris à adapter mon style de communication à différents publics. Cette expérience m'a sensibilisé à l'importance de la culture d'équipe et à l'impact de la collaboration sur la réussite des projets.

6.2.4 Apprentissage continu et autonomie

Ce stage m'a encouragé à adopter une attitude d'apprentissage continu, en explorant de nouvelles technologies et en m'adaptant aux évolutions technologiques. J'ai appris à rechercher des ressources fiables, à expérimenter avec des outils inconnus et à m'adapter aux évolutions du domaine. Cette expérience m'a renforcé ma capacité à travailler de manière autonome tout en sachant demander de l'aide quand nécessaire.

J'ai suivi des cours en ligne, lu des articles techniques et participé à des communautés de développeurs pour rester à jour avec les dernières tendances. J'ai également appris à gérer mon temps efficacement, en équilibrant les tâches de développement avec l'apprentissage et la formation continue.

Chapitre 7

Conclusion et perspectives

7.1 Bilan du stage

7.1.1 Réflexion sur l'expérience acquise

Ce stage chez Timsoft a été une expérience professionnelle enrichissante qui m'a permis de plonger dans le monde du développement logiciel d'entreprise. Durant ces **deux mois**, j'ai eu l'opportunité de contribuer à un projet concret, WorkReserve, qui adresse des besoins réels du marché du travail moderne.

L'expérience a été particulièrement formatrice sur plusieurs aspects. D'abord, j'ai pu appliquer les connaissances théoriques acquises durant mes études à des problématiques pratiques. Le développement d'une application web complète, de la conception à la mise en production, m'a donné une vision holistique du cycle de vie d'un projet logiciel.

7.1.2 Contributions techniques réalisées

Sur le plan technique, mes contributions ont été significatives et variées :

Système d'authentification et sécurité

J'ai implémenté un système d'authentification robuste basé sur **JWT**, incluant l'authentification à deux facteurs (**2FA**) avec TOTP. Cette implémentation assure la sécurité des utilisateurs tout en maintenant une expérience fluide.

Intégration des paiements

L'intégration de **Stripe** pour le traitement des paiements a été un défi technique majeur. J'ai mis en place un système idempotent avec gestion d'erreurs et retry automatique, garantissant la fiabilité des transactions financières.

Architecture et optimisation

J'ai contribué à l'architecture en couches de l'application, utilisant **Spring Boot** pour le backend et **React** pour le frontend. L'implémentation d'une stratégie de cache multi-

niveaux a permis d'optimiser les performances, réduisant les temps de réponse de manière significative.

Tests et qualité

J'ai développé une suite complète de tests, atteignant une couverture de **87%** pour le backend et **83%** pour le frontend. Cette approche rigoureuse assure la maintenabilité et la fiabilité du code.

7.1.3 Compétences développées

Au-delà des aspects techniques, ce stage m'a permis de développer des compétences transversales essentielles :

Gestion de projet

J'ai participé à des sprints agiles, contribuant à la planification, l'estimation des tâches et la livraison itérative. Cette expérience m'a sensibilisé à l'importance de la communication et de la collaboration en équipe.

Résolution de problèmes

Face à des défis complexes comme l'optimisation des performances ou la gestion des erreurs, j'ai appris à analyser les problèmes de manière systématique et à proposer des solutions innovantes.

Apprentissage continu

Le stage m'a encouragé à adopter une attitude d'apprentissage continu, en explorant de nouvelles technologies et en m'adaptant aux évolutions du domaine.

7.1.4 Impact sur le projet

Mes contributions ont eu un impact mesurable sur le succès de WorkReserve. Les optimisations réalisées ont amélioré les performances globales, tandis que les tests automatisés ont renforcé la qualité du code. L'application est désormais prête pour un déploiement en production, avec une architecture scalable et sécurisée.

7.2 Améliorations possibles et perspectives

7.2.1 Évolutions techniques envisagées

Migration vers une architecture microservices

À moyen terme, il serait bénéfique de migrer vers une architecture **microservices** pour améliorer la scalabilité et la maintenabilité. Cette évolution permettrait de décou-

pler les différents modules (authentification, paiements, réservations) et de faciliter les déploiements indépendants.

Amélioration des performances

Plusieurs optimisations sont possibles pour améliorer davantage les performances : - Implémentation d'un cache distribué comme Redis - Optimisation des requêtes de base de données avec des index composites - Mise en place d'un CDN pour la distribution des ressources statiques

Sécurité renforcée

Pour renforcer la sécurité, on pourrait envisager : - L'implémentation d'un WAF (Web Application Firewall) - L'ajout d'une authentification biométrique - Des audits de sécurité réguliers et des tests de pénétration

7.2.2 Développements fonctionnels

Application mobile

Le développement d'applications mobiles natives pour iOS et Android étendrait la portée de WorkReserve et améliorerait l'expérience utilisateur mobile.

Fonctionnalités avancées

Plusieurs fonctionnalités pourraient enrichir la plateforme : - Réservations récurrentes et templates personnalisés - Intégration avec des calendriers externes (Google Calendar, Outlook) - Système de notifications push et d'alertes intelligentes - Analytics prédictifs pour l'optimisation des espaces

IA et machine learning

L'intégration d'algorithmes d'intelligence artificielle pourrait offrir : - Recommandations personnalisées de salles - Prédiction de la demande et optimisation des prix - Analyse automatique des retours utilisateurs

7.2.3 Perspectives professionnelles

Évolution de carrière

Cette expérience de stage m'a confirmé mon intérêt pour le développement full-stack et les technologies cloud. Je souhaite poursuivre dans cette voie, en me spécialisant davantage dans les architectures distribuées et les pratiques **DevOps**.

Apprentissages clés

Les principales leçons tirées de ce stage sont : - L'importance de la planification architecturale précoce - La valeur des tests automatisés pour la qualité du code - La nécessité d'une communication claire dans les projets d'équipe - L'intérêt de l'apprentissage continu dans un domaine en évolution rapide

7.2.4 Impact sur l'entreprise

WorkReserve représente une opportunité significative pour Timsoft d'étendre son portefeuille de solutions logicielles. La plateforme répond à un besoin croissant du marché du travail flexible et pourrait générer de nouveaux revenus tout en renforçant la position de l'entreprise dans le secteur technologique.

En conclusion, ce stage a été une étape déterminante dans mon parcours professionnel. Il m'a permis d'acquérir des compétences techniques solides, de développer une vision d'ensemble du développement logiciel et de contribuer à un projet concret. Les perspectives d'évolution de WorkReserve sont prometteuses et je suis convaincu que cette plateforme connaîtra un succès commercial important.

Annexe A

Annexes

A.1 Captures d'écran de l'application

Voici quelques captures d'écran supplémentaires de l'application WorkReserve :

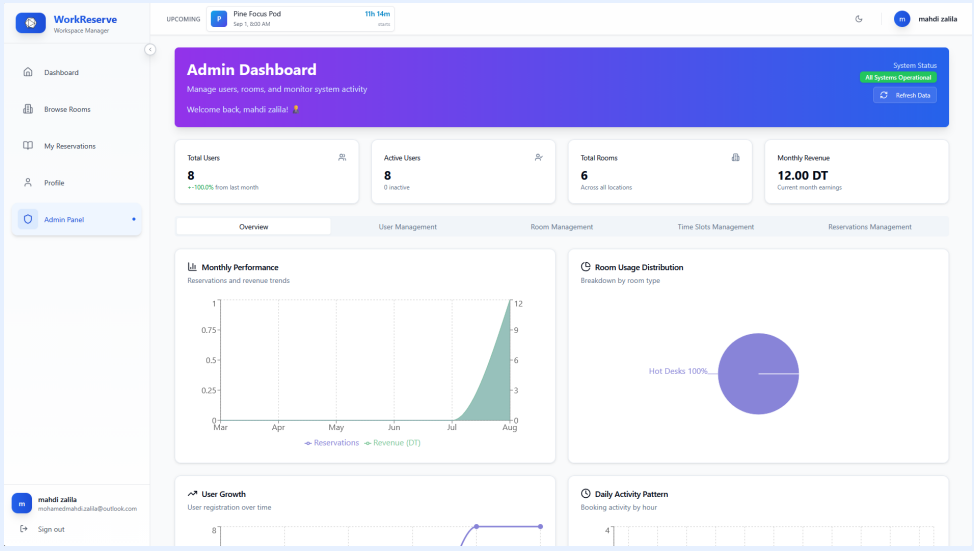


FIGURE A.1 – Tableau de bord administrateur avec métriques et contrôles de gestion.

Description : Cette interface d'administration permet aux gestionnaires de surveiller les réservations, gérer les salles et consulter les statistiques d'utilisation.

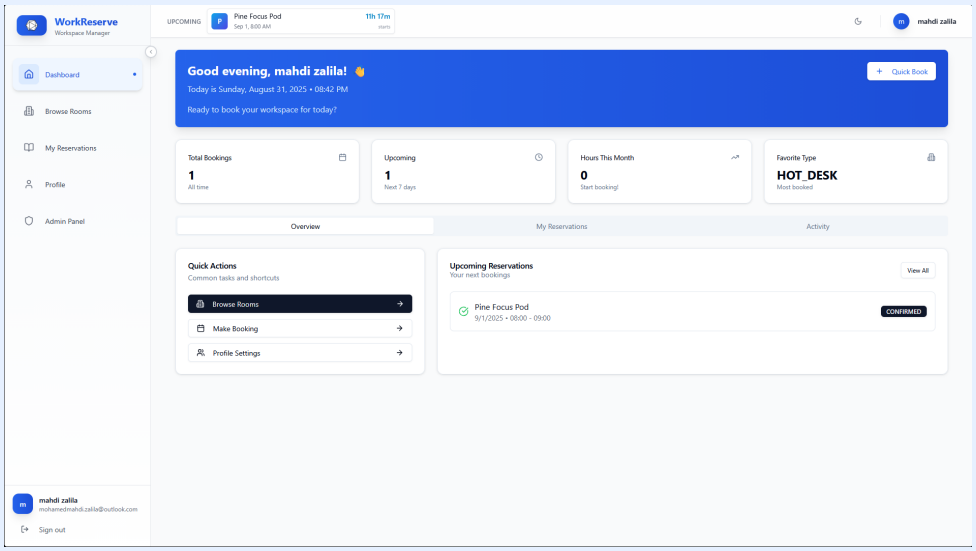


FIGURE A.2 – Tableau de bord utilisateur principal avec accès rapide aux fonctionnalités.

Description : Le tableau de bord principal offre un aperçu rapide des réservations actives, des notifications et des actions rapides pour une navigation fluide.

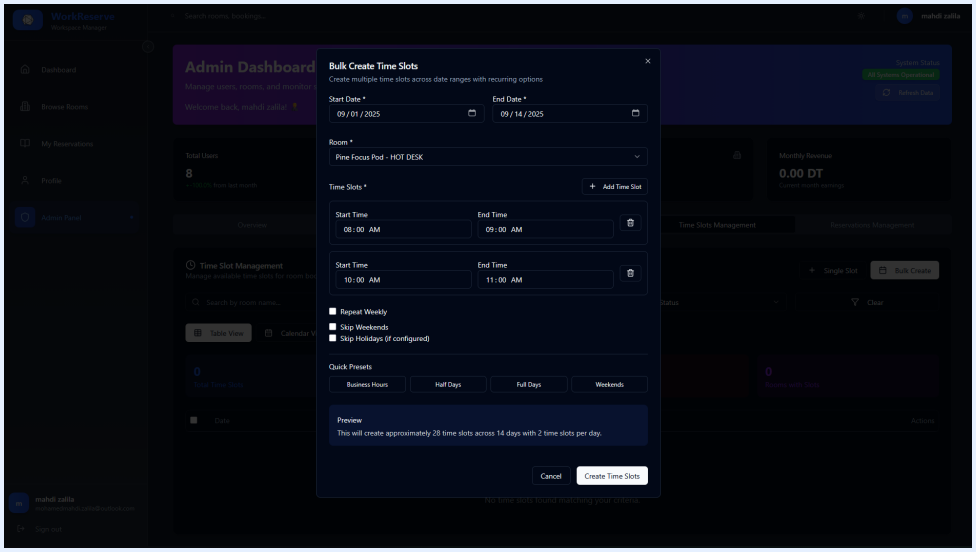


FIGURE A.3 – Création de créneaux horaires pour la gestion des disponibilités.

Description : Cette page permet aux administrateurs de créer et gérer les créneaux horaires disponibles pour chaque salle, en définissant les périodes et les capacités.

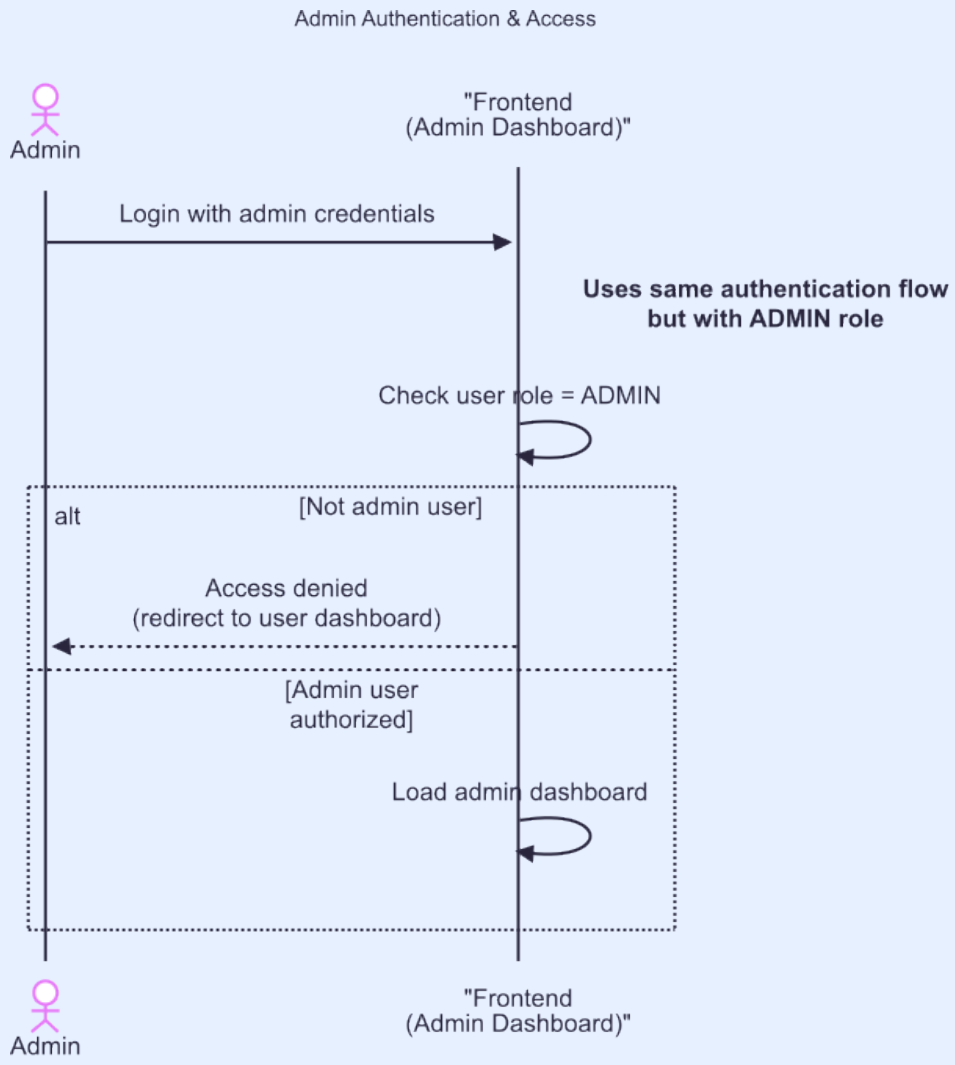


FIGURE A.4 – Flux d'authentification et contrôle d'accès pour les administrateurs.

Description : Diagramme illustrant les processus d'authentification spécifiques aux administrateurs et les niveaux d'accès.

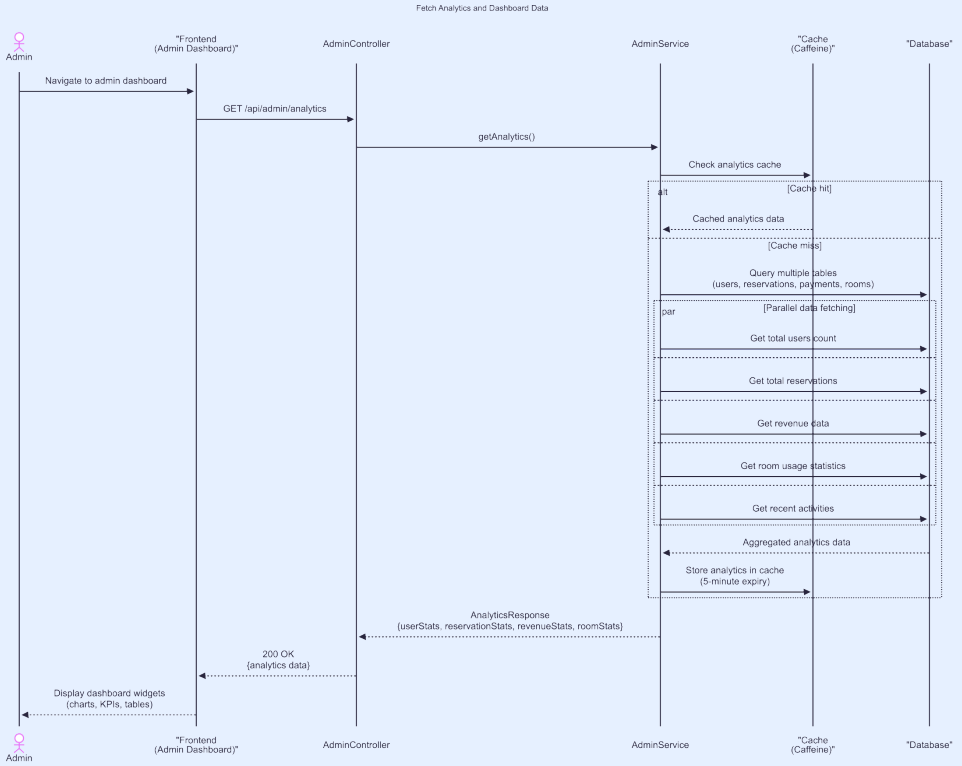


FIGURE A.5 – Récupération des données d'analyses pour le tableau de bord.

Description : Processus de collecte et d'agrégation des données pour générer les métriques du tableau de bord.

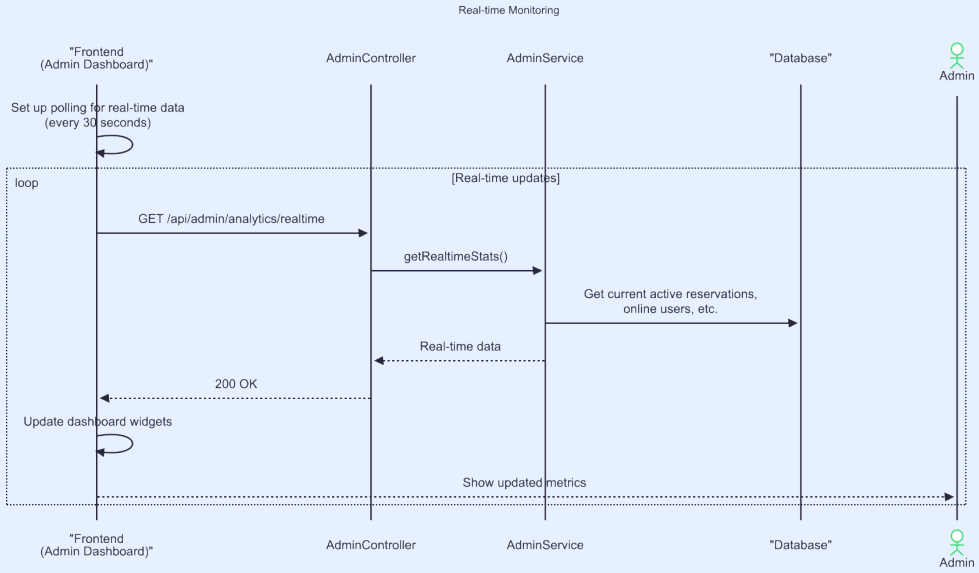


FIGURE A.6 – Surveillance en temps réel des performances et événements système.

Description : Système de monitoring continu pour détecter les anomalies et suivre les indicateurs clés de performance.

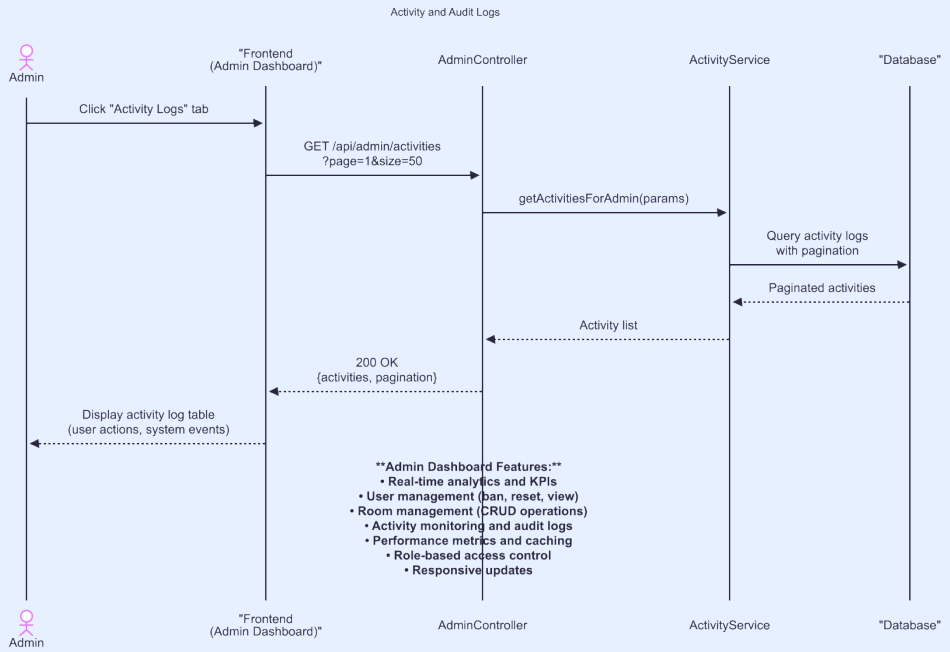


FIGURE A.7 – Logs d’activité et d’audit pour la traçabilité des opérations.

Description : Enregistrement structuré des actions utilisateurs et événements système pour l’audit et le débogage.

Clôture

Ce rapport de stage a été réalisé dans le cadre de mon cycle d'ingénieur en spécialité **Cloud Computing & DevOps**. Il présente les travaux effectués durant mon stage chez Timsoft du 16 juillet 2025 au 29 août 2025.

Les expériences acquises et les compétences développées durant cette période constituent une base solide pour ma future carrière professionnelle dans le domaine du développement logiciel et des technologies cloud.

Je remercie une nouvelle fois mon tuteur en entreprise, **Oumaima Barika**, ainsi que toute l'équipe de Timsoft pour leur soutien et leur accompagnement tout au long de ce stage.

Fait à Tunis, le 29 août 2025	**Tuteur en entreprise**
Mohamed Mahdi Zalila	**Oumaima Barika**
Étudiant en ingénierie	Timsoft
Signature	Signature