REPUBLIQUE TUNISIENNE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Carthage
Institut National des Sciences Appliquées et de Technologie

Projet personnel et professionnel

Diplôme National d'Ingénieur
en Sciences Appliquées et Technologie

Spécialité : Génie Logiciel

Parki
Solution intelligente de gestion de parking

Présenté par

Adam Ladhari
Wided Ouesleti
Skander BenAchour

Madame TAKTAK Hajer INSAT

Année Universitaire : 2023 / 2024

Remerciements

Nous exprimons notre gratitude envers toutes les personnes qui ont joué un rôle crucial dans notre parcours tout au long de ce projet professionnel.

Tout d'abord, nous sommes reconnaissants envers la providence qui nous a octroyé la force et la résilience nécessaires pour traverser cette période complexe.

Un merci spécial à notre mentor, Madame Hajer Taktak, pour son soutien indéfectible et ses orientations précieuses qui ont été déterminants tout au long de notre stage. Sa patience, son expertise et son accompagnement ont été fondamentaux pour la réussite de notre projet.

Nous tenons également à remercier chaleureusement nos professeurs à l'INSAT pour leur enseignement rigoureux et passionné. Leur engagement envers l'excellence et leur volonté de partager leur savoir ont enrichi notre expérience éducative et préparé le terrain pour notre future carrière.

Table des matières

Table des figures

Introduction Générale

Dans un monde où l'efficacité urbaine et la technologie convergent rapidement pour améliorer la qualité de vie, la gestion des parkings se présente comme un défi majeur pour les villes modernes. Face à cela, le projet Parki a été développé pour offrir une solution innovante et intégrée à cette problématique. Parki est une application de gestion de parking qui utilise des technologies avancées pour optimiser l'utilisation des espaces de stationnement urbains, réduire la congestion, et améliorer l'expérience utilisateur pour les conducteurs.

Le système est conçu pour fonctionner avec une interface mobile intuitive et un backend robuste, intégrant des fonctionnalités clés telles que la visualisation des places disponibles en temps réel, la réservation de places, et le paiement automatisé. En outre, Parki offre un tableau de bord administratif pour les gestionnaires de parking, permettant une gestion efficace des ressources, des tarifications dynamiques et une analyse détaillée de l'utilisation des parkings.

Ce projet tire parti des dernières avancées en matière de développement logiciel, notamment l'utilisation de Flutter pour le développement cross-platform de l'application mobile, de Django pour la gestion des données backend, et de Unity pour les simulations de parking en temps réel. Les tests sont réalisés avec des outils tels que Postman pour assurer la fiabilité et la robustesse des interfaces API.

L'objectif de Parki est double : améliorer l'expérience des utilisateurs en simplifiant l'accès au stationnement et aider les gestionnaires de parkings à optimiser leurs opérations. En répondant à ces besoins, Parki se positionne non seulement comme une réponse aux défis actuels de mobilité urbaine mais aussi comme une plateforme adaptable pour les innovations futures dans la gestion des espaces urbains.

Chapitre 1

Cadre du projet

Introduction

1.1 Problématique

Trouver une place de parking devient de plus en plus difficile dans les zones urbaines. Les utilisateurs de parkings rencontrent fréquemment des difficultés à localiser des places disponibles, ce qui leur fait perdre du temps et augmente la pollution. L'utilisation des tickets cause également des pertes de temps et génère encore plus de pollution.

D'autre part, les propriétaires de parkings sont confrontés au problème des fraudes de tickets, et aux plaintes des clients concernant les attentes. Il est également difficile de suivre les revenus et les données de leur parking, rendant difficile l'implémentation de stratégies de tarification dynamique.

Enfin, l'adoption croissante de politiques environnementales encourage les entreprises à réduire leur dépendance au papier. Les systèmes de gestion de parking actuels, souvent dépendants des tickets et des reçus papier, ne correspondent pas à cette orientation vers des solutions plus durables.

1.2 Solution proposée

Face aux défis grandissants dans la gestion des parkings urbains, notre solution, Parki, propose un système intégré utilisant les dernières technologies pour améliorer l'expérience des utilisateurs et optimiser la gestion pour les propriétaires. Voici les éléments clés de notre solution :

1. **Application Mobile pour les Utilisateurs**
   - **Localisation en Temps Réel :** L'application permet aux utilisateurs de localiser rapidement les parkings disponibles à proximité de leur emplacement actuel ou de toute adresse saisie, grâce à une intégration de système de géolocalisation.
   - **Paiement Mobile :** Pour éliminer les files d'attente et réduire l'utilisation du papier, notre application offre une fonction de paiement mobile sécurisé qui permet aux utilisateurs

de payer leur stationnement directement depuis leur téléphone.

— **Gestion des Comptes Personnels :** Les utilisateurs peuvent créer et gérer leur compte, enregistrer plusieurs véhicules, visualiser l'historique de leurs stationnements et les transactions effectuées.

2. **Système de Reconnaissance Automatique des Plaques**

— **Accès Automatisé :** À l'entrée et à la sortie des parkings, notre système reconnaît automatiquement les plaques d'immatriculation, permettant un accès et une sortie rapides sans intervention manuelle.

— **Intégration avec le Système de Paiement :** Le système identifie les véhicules associés à des comptes actifs et traite les paiements selon la durée du stationnement, facilitant ainsi une tarification dynamique et précise.

3. **Dashboard Administratif pour les Propriétaires**

— **Gestion des Revenus et des Promotions :** Les outils intégrés permettent aux propriétaires de configurer des tarifs dynamiques basés sur la demande et de créer des promotions pour attirer plus d'utilisateurs pendant les périodes de faible activité.

— **Rapports Détaillés :** Des rapports automatisés fournissent des analyses détaillées des performances et des tendances, aidant les propriétaires à prendre des décisions éclairées pour optimiser leurs opérations.

4. **Durabilité et Conformité Environnementale**

— **Réduction de l'Utilisation du Papier :** En remplaçant les tickets papier par des transactions numériques, Parki contribue à réduire significativement la consommation de papier, alignant les opérations des parkings avec les politiques environnementales modernes.

Chapitre 2

Specification des besoins

Introduction

Ce chapitre détaille les besoins essentiels pour le développement de Parki, notre système de gestion de parking. Nous définirons les exigences fonctionnelles et non fonctionnelles pour les utilisateurs et les gestionnaires de parking, en nous concentrant sur l'amélioration de l'expérience utilisateur et l'optimisation de la gestion des parkings. Une compréhension claire de ces besoins est cruciale pour concevoir une solution qui répond non seulement aux attentes des utilisateurs mais assure également performance, sécurité, et fiabilité.

2.1 Identification des Acteurs

Les acteurs identifiés dans le système Parki :

Utilisateur Final (Conducteur)

Les utilisateurs finaux sont les conducteurs de véhicules qui utilisent l'application Parki pour trouver, réserver et payer les places de parking. Ils interagissent directement avec l'application mobile pour faciliter leur expérience de stationnement.

Gestionnaire de Parking

Les gestionnaires de parking, souvent propriétaires ou opérateurs des espaces de stationnement, utilisent Parki pour gérer les aspects administratifs et opérationnels des parkings. Ils accèdent au système via une interface web pour mettre à jour les informations du parking, consulter les revenus et analyser l'utilisation des places.

2.2 Analyse des besoins

Cette phase et responsable de l'ensemble des fonctionnalités offertes par le système. Les besoins ont été répartis en deux catégories, les besoins fonctionnels et les besoins non fonctionnels.

2.2.1 Besoins Fonctionnels

Pour les Utilisateurs Finaux (Conducteurs)

— **Recherche de Parkings :** Les utilisateurs doivent pouvoir rechercher des places de parking disponibles à proximité de leur localisation actuelle ou d'une adresse spécifique.

— **Réservation de Places :** Les utilisateurs doivent avoir la possibilité de réserver une place de parking à l'avance, en spécifiant la date et l'heure de début et de fin de leur stationnement.

— **Paiement Mobile :** Les utilisateurs doivent pouvoir payer leur stationnement de manière sécurisée via l'application mobile.

— **Gestion de Profil :** Permettre aux utilisateurs de créer et gérer leur profil, y compris l'ajout et la suppression de véhicules, ainsi que la consultation de l'historique de leurs réservations et paiements.

Pour les Gestionnaires de Parking

— **Dashboard de Gestion :** Offrir un tableau de bord complet pour la gestion des places de parking, incluant des visualisations en temps réel de l'occupation des places.

— **Gestion des Tarifs :** Permettre la modification dynamique des tarifs en fonction de la demande, des événements spéciaux, ou d'autres critères.

— **Rapports et Analyses :** Fournir des rapports détaillés sur les revenus, l'utilisation des places, les périodes de pic, et d'autres statistiques pertinentes pour la gestion efficace du parking.

— **Gestion des Utilisateurs :** Permettre aux gestionnaires de créer, modifier, et supprimer des comptes d'utilisateurs, et de gérer les droits d'accès à l'interface d'administration.

Système Intégré

— **Synchronisation avec la Reconnaissance de Plaques :** Assurer une intégration fluide avec le système de reconnaissance de plaques pour automatiser l'enregistrement des entrées et sorties des véhicules.

— **Sécurité du Système :** Implémenter des mesures de sécurité robustes pour protéger les données personnelles et financières des utilisateurs, y compris le chiffrement des données et des communications.

2.3 Besoins Non Fonctionnels

Les besoins non fonctionnels sont essentiels pour assurer que le système Parki est robuste, sécurisé et évolutif. Ils garantissent que l'application fonctionne efficacement et répond aux attentes des utilisateurs et des gestionnaires de parking.

Performance

— **Temps de réponse :** Le système doit offrir des temps de réponse rapides pour assurer une expérience utilisateur fluide.

— **Capacité de traitement :** Capable de gérer simultanément un grand nombre d'utilisateurs et de transactions sans perte de performance.

Sécurité

— **Protection des données :** Mettre en place des mesures robustes pour sécuriser les données des utilisateurs contre les accès non autorisés, les altérations ou pertes, incluant le chiffrement des données sensibles.

— **Authentification et autorisation :** Implémenter des mécanismes d'authentification forte pour les utilisateurs et les administrateurs.

Disponibilité

— **Haute disponibilité :** Le système doit être opérationnel et accessible 24/7, avec des stratégies de redondance et de reprise après sinistre pour minimiser le temps d'arrêt.

— **Maintenance :** Planifier des maintenances régulières pour assurer le bon fonctionnement continu du système.

Scalabilité

— **Extensibilité :** Le système doit être conçu pour faciliter l'ajout de capacités supplémentaires sans modifications majeures de l'architecture existante.

— **Adaptabilité :** Capacité à s'adapter à l'augmentation du volume de données et du nombre d'utilisateurs sans dégradation de la performance.

Interopérabilité

— **Intégration :** Le système doit être capable de s'intégrer facilement avec d'autres systèmes ou technologies, facilitant ainsi les extensions ou les mises à jour futures.

2.4 Diagramme de cas d'utilisation général

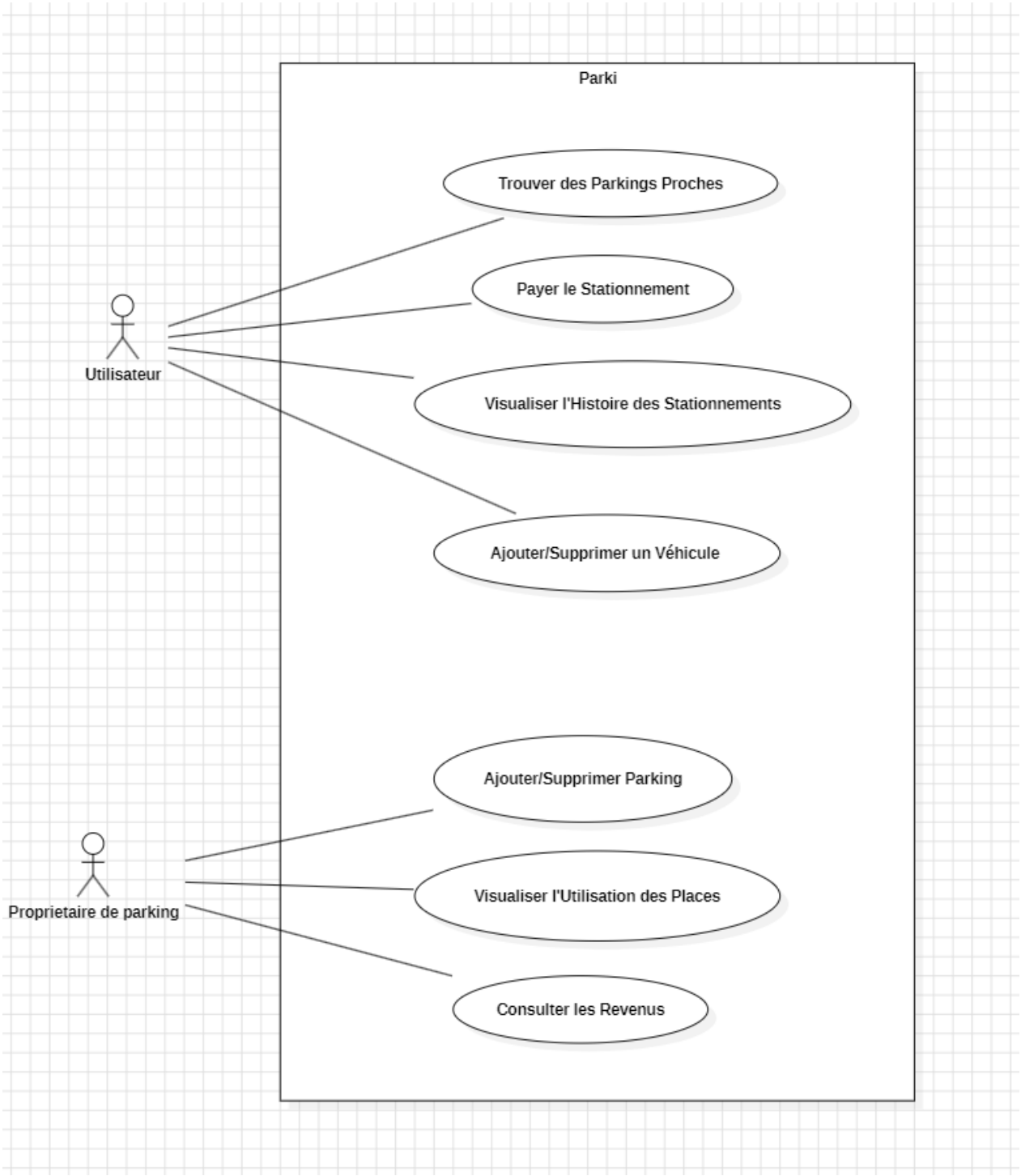

Figure 2.1 : Diagramme de cas d'utilisation général

2.5 Choix technologique

Après avoir planifié les sprints et releases de notre projet, nous présentons ici l'architecture logique et physique de notre application, ainsi que les environnements matériel et logiciel utilisés, suivi d'une conclusion qui résume l'adéquation des technologies choisies avec les objectifs du projet.

2.5.1 Architecture Logique globale

L'architecture logique de notre application Parki est conçue pour assurer une modularité élevée et une maintenance aisée. Elle comprend plusieurs couches principales :

— **Couche de présentation :** Interface utilisateur développée avec Flutter pour fournir une expérience fluide sur mobile. Et une simple interface robuste pour la version web.

— **Couche métier :** Logique métier implémentée en Python avec le framework Django, gérant les processus tels que les réservations, les paiements et la gestion des utilisateurs.

— **Couche de données :** Base de données utilisée pour stocker toutes les données de l'application de manière sécurisée et performante.

2.5.2 Environnement matériel

Bien que le déploiement réel ne soit pas encore effectué, notre conception prévoit l'utilisation de matériels spécifiques pour soutenir la fonctionnalité du système :

— **Caméras :** Des caméras hautement performantes placées à des points stratégiques pour capturer les plaques d'immatriculation des véhicules entrant et sortant des parkings.

— **Capteurs :** Utilisation de capteurs pour détecter la présence de véhicules dans les places de parking et pour assurer une gestion efficace de l'espace.

Conclusion

La sélection de ces technologies a été guidée par les besoins de performance, sécurité, et scalabilité du projet Parki. Cette infrastructure technologique nous permettra de répondre aux attentes de nos utilisateurs tout en assurant une gestion efficace et sécurisée des données.

Chapitre 3

Conception

Introduction

Le terme release peut être défini comme une période de temps qui permet de produire une version distribuée d'une application. Un release est constitué d'une suite d'itérations (sprint) qui se terminent quand les incréments de ces derniers construisent un produit. Dans ce chapitre, nous allons présenter le travail réalisé lors des deux premiers sprints comportant le diagramme de classe du premier release suitée par une phase de planification et les différentes tâches tout en spécifiant à chaque sprint les différentes parties : Backlog sprint et réalisation.

3.1 Architecture du Système

3.1.1 Vue d'ensemble de l'architecture

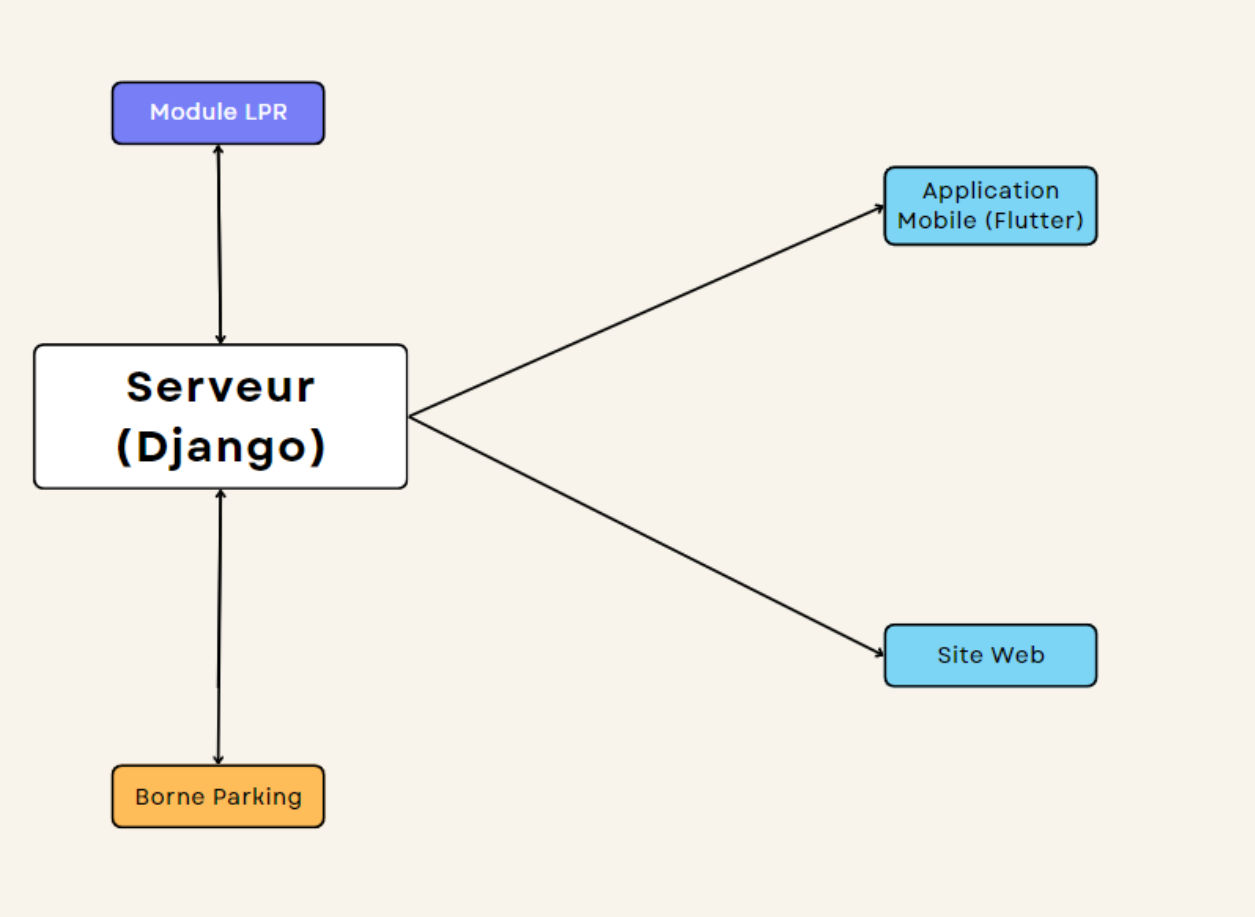

Figure 3.1 : schema de l'architecture

3.2 Conception des Interactions

Entrée au parking

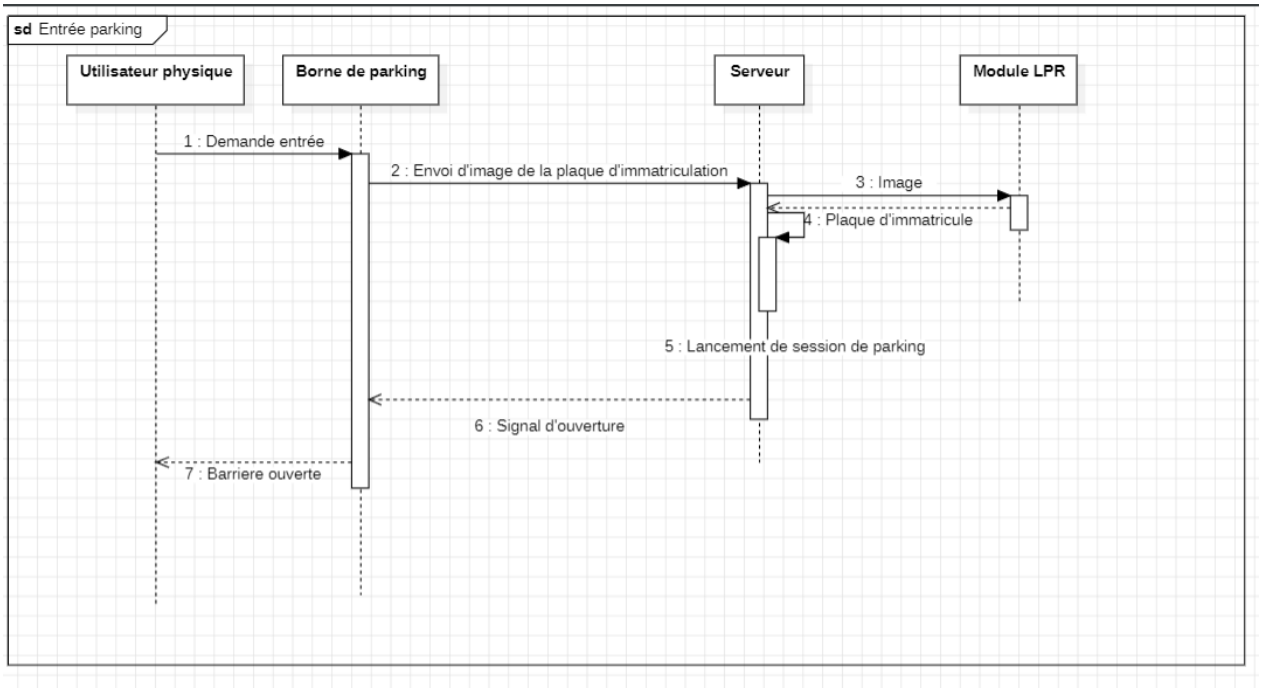

Figure 3.2 : Diagramme de sequence d'entrée au parking

Sortie du parking

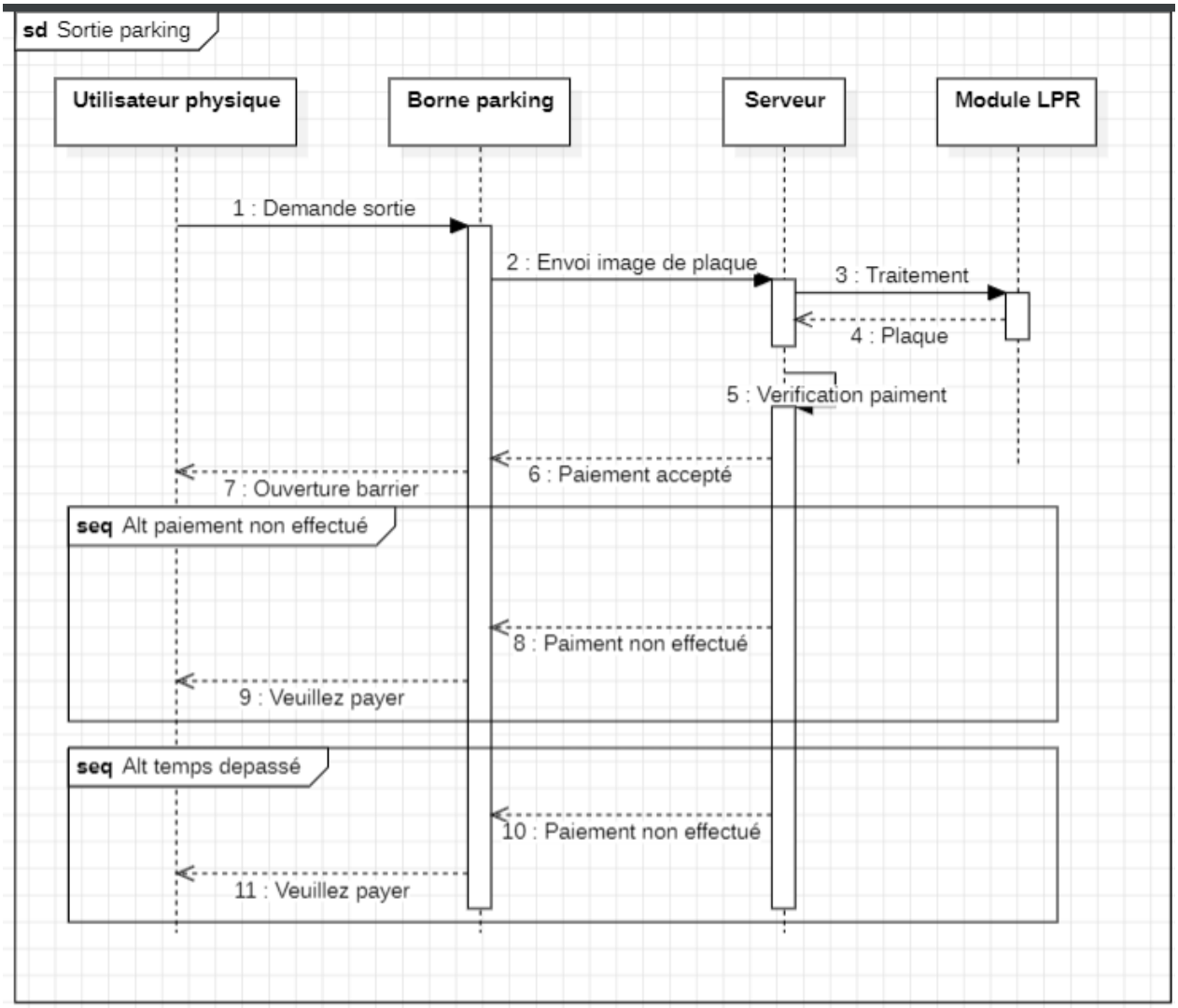

Figure 3.3 : Diagramme de sequence de sortie du parking

Paiement

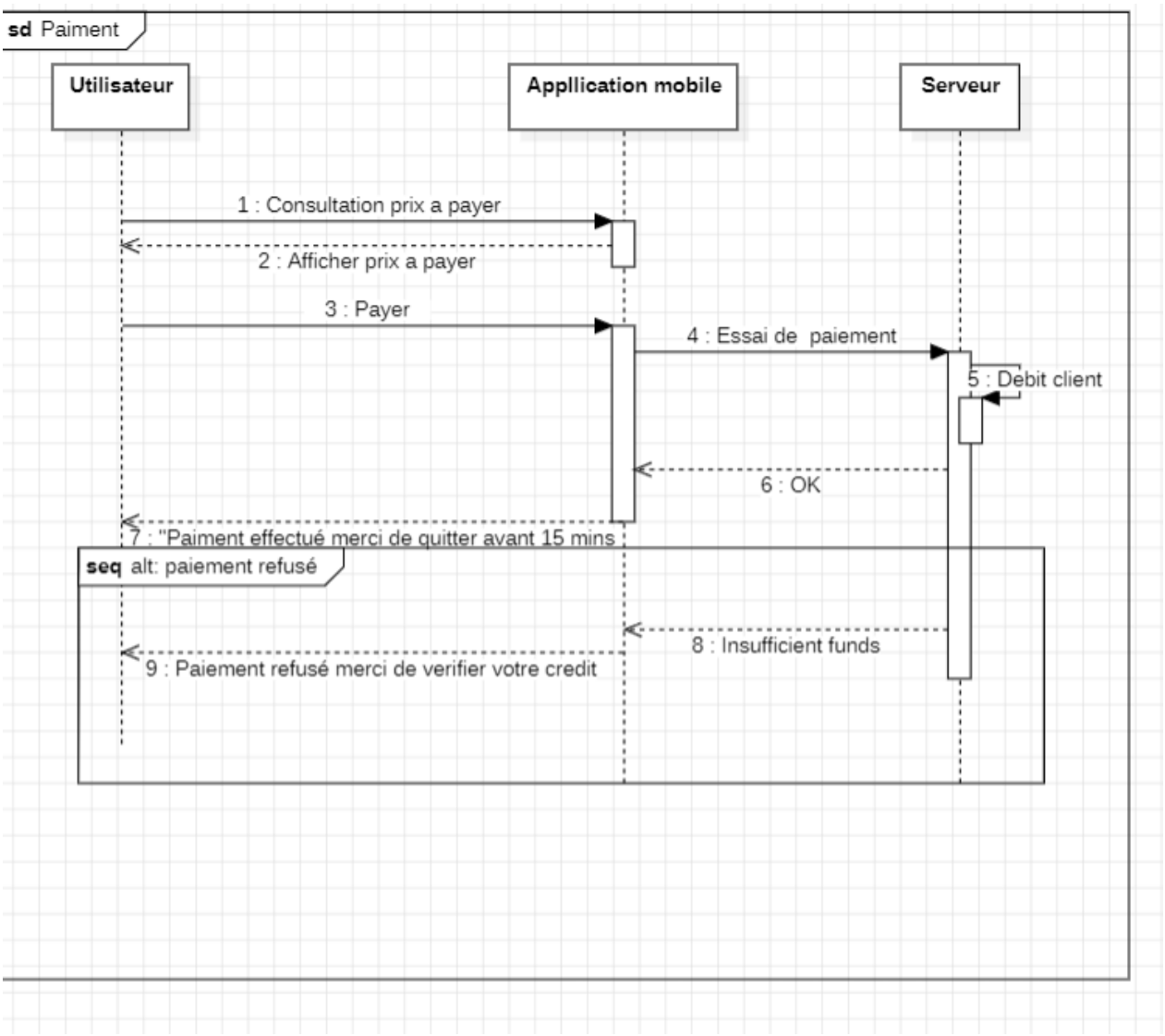

Figure 3.4 : Diagramme de sequence de paiement

Reservation de place

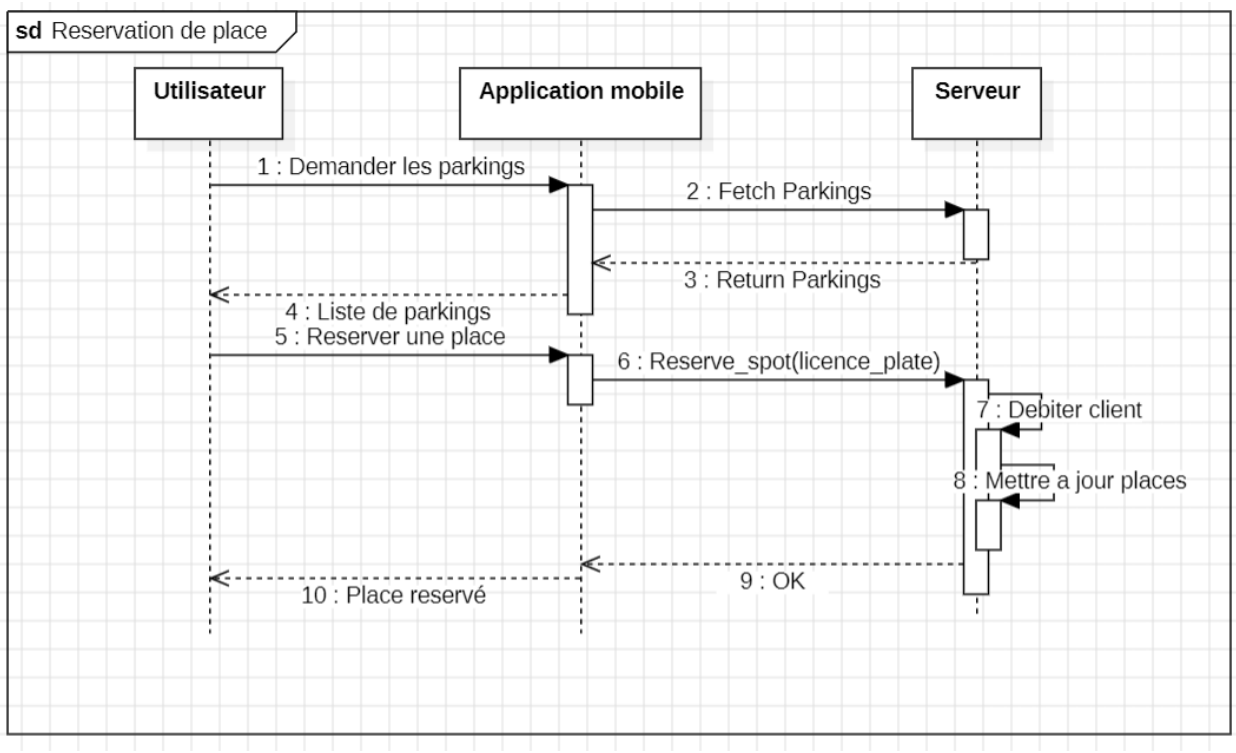

Figure 3.5 : Diagramme de sequence de reservation de place

Conclusion

Dans ce chapitre, nous sommes arrivés à l'objectif désiré qui est le fait de pouvoir gérer la recherche des capteurs convenables pour répondre aux besoins fixés pour cette release.

Chapitre 4

Realisation

Introduction

Ce chapitre de réalisation du projet Parki décrit la transformation des concepts et des designs détaillés du système en une application fonctionnelle et prête à l'emploi. Cette phase est cruciale car elle transforme les spécifications théoriques en solutions pratiques, en validant l'efficacité des idées à travers leur implémentation, leur test, et leur ajustement basé sur des feedbacks réels.

4.1 Développement du Système

Le développement de Parki a été réalisé en utilisant des outils et technologies avancés, adaptés aux exigences spécifiques du projet. Cette section décrit en détail notre environnement de développement.

4.1.1 Environnement de Développement

Nous avons utilisé une variété d'outils pour coder, tester, et déployer les différentes composantes de Parki :

— **Visual Studio Code (VS Code)** : Utilisé pour le développement général, notamment pour la programmation en Python avec le framework Django et pour l'écriture de scripts de test. VS Code est apprécié pour ses nombreuses extensions, son intégration avec Git et sa polyvalence.

— **Android Studio** : Choisi pour le développement de l'application mobile avec Flutter. Android Studio fournit des outils puissants pour la conception d'interfaces utilisateur, le débogage, et la simulation sur appareils Android.

— **Flutter** : Employé pour créer une interface utilisateur réactive et esthétiquement agréable pour les applications mobiles iOS et Android, permettant une compilation en code natif pour

les deux plateformes.

— **Django** : Sélectionné pour le développement backend en raison de sa robustesse, sa sécurité, et ses capacités de développement rapide. Django facilite également la gestion de la base de données et l'authentification.

— **Unity** : Utilisé pour les simulations en temps réel, essentiel pour modéliser et simuler les comportements de trafic et de stationnement dans des environnements virtuels.

— **Git** : Employé pour la gestion de versions, permettant une collaboration fluide entre développeurs grâce à GitHub, qui a servi de répertoire central pour le code source.

— **Postman** : Utilisé intensivement pour tester les API développées dans le cadre du projet. Postman a permis de valider les requêtes, les réponses et le comportement des API avant leur déploiement en production.

Cette configuration diversifiée a facilité une collaboration efficace et a permis une gestion agile et précise du développement du projet Parki, assurant que toutes les fonctionnalités répondent aux normes de qualité et aux attentes des utilisateurs.

4.2 Implémentation des Fonctionnalités

La phase d'implémentation des fonctionnalités de Parki a été essentielle pour transformer les conceptions en solutions opérationnelles efficaces. Voici un aperçu détaillé du processus, des défis rencontrés, et des solutions apportées.

4.2.1 Développement de l'Application Mobile avec Flutter

Processus : L'application mobile a été développée avec Flutter, offrant une interface utilisateur réactive pour les appareils iOS et Android.
Défis : Assurer la performance et la fluidité sur divers appareils.
Solutions : Optimisation du code Flutter et tests exhaustifs sur différents appareils pour garantir la compatibilité et la fluidité.

4.2.2 Backend avec Django

Processus : Utilisation de Django pour gérer les interactions entre l'application mobile et la base de données, incluant la gestion des utilisateurs et les réservations de parking.
Défis : Gérer les hautes charges de transactions tout en assurant la sécurité des données.

Solutions : Optimisation des requêtes de base de données et implémentation de mesures de sécurité robustes.

4.2.3 Simulations en Temps Réel avec Unity

Processus : Création de simulations en temps réel des flux de trafic et des configurations de parking avec Unity.

Défis : Intégrer ces simulations avec les données réelles du backend Django.

Solutions : Développement d'interfaces API pour l'échange de données en temps réel.

4.2.4 Gestion des Utilisateurs

Processus : Implémentation de la gestion des utilisateurs dans l'application mobile, utilisant Flutter pour créer des formulaires sécurisés et réactifs.

Défis : Maintenir la confidentialité et la sécurité des informations personnelles.

Solutions : Utilisation de protocoles d'authentification modernes et mesures de sécurité telles que le hashing de mots de passe.

4.2.5 Dashboard Administratif

Processus : Développement d'un dashboard avec Django et extensions JavaScript pour visualiser et gérer les places, revenus, et statistiques d'utilisation.

Défis : Gérer de grands volumes de données en temps réel.

Solutions : Optimisation des requêtes de base de données et mise en place d'un système de mise en cache.

4.3 Tests avec Postman et Simulations avec Unity

4.3.1 Introduction aux Tests

Les tests jouent un rôle essentiel pour assurer que le système Parki fonctionne correctement et répond aux exigences techniques sans erreurs significatives. Notre stratégie a combiné des tests automatisés des API avec des simulations pour évaluer le système sous conditions contrôlées.

4.3.2 Stratégie de Tests

Nous avons adopté une approche multi-niveaux pour tester le système Parki, incluant les éléments suivants :

— Tests Unitaires pour valider chaque composant isolément.

— Tests d'Intégration pour vérifier l'interaction correcte entre les composants.

— Tests de Charge pour simuler un environnement de forte utilisation et évaluer la capacité du système à gérer un grand volume de transactions.

4.3.3 Utilisation de Postman pour les Tests d'API

Postman a joué un rôle clé dans notre processus de test, notamment pour :

— **Configuration :** Création de collections de requêtes pour tester l'inscription des utilisateurs, les réservations et les opérations administratives.

— **Automatisation des Tests :** Les tests ont été automatisés pour effectuer des validations continues, permettant la détection rapide des régressions.

— **Validation des Réponses :** Nous avons utilisé Postman pour valider les réponses des API par rapport aux attentes.

— **Rapports de Test :** Génération de rapports détaillés post-test pour faciliter le débogage et les ajustements nécessaires.

4.3.4 Simulations avec Unity

Nous avons utilisé Unity pour simuler l'interaction des véhicules dans le parking sous conditions contrôlées, incluant :

— Simulation de la logique de gestion des places et des réservations sans trafic réel.

— Reproduction de conditions réelles simulées pour tester la performance du système sous divers scénarios.

Conclusion

Les tests approfondis avec Postman et les simulations avec Unity ont été cruciaux pour valider la fonctionnalité et la robustesse de Parki. Cette démarche nous a permis de garantir que le système était prêt à être déployé, performant et fiable sous des conditions variées d'utilisation.

Conclusion générale et perspectives

Conclusion Générale

Nous espérons que le projet Parki franchira des étapes significatives vers la modernisation de la gestion des parkings en intégrant des technologies avancées pour améliorer l'expérience utilisateur et optimiser les opérations des gestionnaires de parkings. À travers l'utilisation de solutions de développement mobile cross-platform avec Flutter et le robuste framework backend Django, nous anticipons que Parki démontrera son efficacité à simplifier et sécuriser le stationnement urbain.

Les tests réalisés, notamment avec Postman, ont préparé la voie pour assurer la fiabilité des interfaces API. Les utilisateurs bénéficieront d'une interface intuitive pour la réservation et le paiement de parking, tandis que les gestionnaires disposeront d'outils puissants pour la surveillance et la gestion des ressources.

Perspectives

Malgré les progrès déjà envisagés, le projet Parki dispose de plusieurs axes d'amélioration pour l'avenir :

1. **Amélioration du Traitement des Images pour la LPR :** Nous prévoyons d'améliorer le système de reconnaissance de plaques d'immatriculation en intégrant des techniques de traitement d'image plus avancées et en utilisant l'intelligence artificielle pour augmenter la précision dans des conditions diverses.
2. **Scalabilité du Système :** À mesure que le nombre d'utilisateurs et de parkings augmente, il sera crucial de renforcer la scalabilité du système. Nous envisageons l'adoption d'une architecture de systèmes répartis pour une meilleure gestion des charges et une distribution efficace des ressources.
3. **Expansion Géographique :** Nous envisageons également une expansion géographique de l'application à de nouveaux marchés, ce qui nécessitera des adaptations locales spécifiques pour répondre aux réglementations et aux comportements des utilisateurs dans différentes régions.

Ces améliorations et expansions envisagées positionneront Parki non seulement comme une solution viable à court terme mais aussi comme un leader potentiel dans la technologie de gestion

de parking intelligent à l'échelle globale.

Bibliographie

[1] Flutter : https://flutter.dev/docs.

[2] Django : https://docs.djangoproject.com/en/stable/.

[3] Postman : https://learning.postman.com/.

[4] Unity : https://docs.unity3d.com/Manual/index.html.

[5] API Testing : https://www.apitesting.com/best-practices.

[6] Mobile Development : https://developer.android.com/guide.

[7] Load Testing : https://www.loadtestingtool.com/basics.shtml.

[8] Software Testing Techniques : https://www.softwaretestinghelp.com/software-testing-techniques/.

[9] Android Studio : https://developer.android.com/studio.