

Ma Borne

Application mobile pour la gestion de bornes
de recharge pour véhicules électrique

Projet de fin d'étude de 5ème année
Rapport de synthèse

Équipe : Lucas Leroy, Auriane Poirier et Margaux Thirion

Demandeurs : Crédit Agricole Technologies et Services de Montpellier,
représenté par Damien Teissier et Thomas Deffains

Tutrice École : Lysiane Buisson-Lopez



SOMMAIRE

I. Glossaire	3
II. Table des figures	4
III. Remerciements	5
IV. Introduction	6
A. Présentation du projet	6
B. Présentation de l'entreprise	6
C. Contexte et enjeux	6
1. Enjeux du projet	7
2. Contraintes du projet	7
V. Travail réalisé	8
A. Démarche de conception	8
1. Conception Backend	8
2. Conception Frontend	9
a) Architecture du projet	9
b) Maquettes & prototype	11
B. Résultats	12
1. Résultats Intermédiaires	12
2. Résultat final	13
a) Backend	13
b) Frontend - Utilisateurs	13
c) Frontend - Administrateur	15
d) Frontend - Général	16
VI. Gestion du projet	19
A. Méthodologies utilisées	19
B. Bonnes pratiques	19
1. Organisationnelles	19
2. Techniques	20
VII. Analyse critique	21
A. Analyse des choix techniques et organisationnels	21
B. Difficultés rencontrées	22
1. Front-end	22
2. Back-end	23
C. Éco-conception	24
1. Critères détaillés	24
2. Analyse critique	25
D. Améliorations futures	25
VIII. Expérience tirée	27
A. Valorisation du travail	27
B. Compétences techniques et soft skills développées	27
IX. Annexes	29
X. Résumés	41
A. En français	41
B. En anglais	41
XI. Keywords	41

I. Glossaire

API (Application Programming Interface) : Une interface qui permet à différents logiciels de communiquer entre eux en définissant des règles et des formats d'échange de données.

Architecture MVVM (Model-View-ViewModel) : Un modèle d'architecture logicielle qui sépare les données (Modèle), l'interface utilisateur (Vue) et la logique de présentation (ViewModel).

Architecture MVC: modèle de conception qui sépare une application en trois composants : le Modèle (données et logique métier), la Vue (affichage des données) et le Contrôleur (gestion des interactions entre le modèle et la vue). Elle favorise la modularité et la maintenabilité du code.

Diagramme de Gantt : Un outil visuel utilisé pour planifier et suivre l'avancement d'un projet en représentant les tâches, leurs durées et leurs dépendances.

Github : Une plateforme en ligne pour héberger et gérer du code source, utilisée par les développeurs pour collaborer sur des projets.

JetBrains : Une société de logiciels qui développe des outils pour les développeurs, notamment l'IDE IntelliJ IDEA et le langage Kotlin.

JetPack Compose :outil pour créer des interfaces utilisateur sur Android. Il utilise Kotlin pour simplifier et accélérer le développement d'applications avec une approche déclarative.

Kotlin Multiplateforme : Un framework qui permet d'écrire du code une seule fois et de l'utiliser sur différentes plateformes (Android, iOS, web, etc.).

Loader : Un composant qui permet de charger des données en arrière-plan et de les afficher une fois qu'elles sont disponibles.

MCD : Diagramme qui représente la structure d'une base de données. Il décrit les différents éléments (entités) qui composent la base de données et les relations entre ces éléments.

MongoDB : Une base de données populaire, conçue pour stocker et gérer de grandes quantités de données sans structure rigide.

Pop-up : Une petite fenêtre qui apparaît soudainement à l'écran pour afficher une information ou une option.

Render : Un service cloud qui permet de déployer et d'exécuter des applications web et des API.

Requête HTTP : Message envoyé par un client (généralement un navigateur web) à un serveur pour effectuer une action sur une ressource. Cette ressource peut être une page web, une image, une vidéo, des données, etc.

SQL (Structured Query Language) : Un langage de programmation utilisé pour interagir avec les bases de données relationnelles et effectuer des opérations comme la création, la lecture, la modification et la suppression de données.

II. Table des figures

Figure 1 : Schéma de l'architecture du projet.....	10
Figure 2 : Page d'accueil.....	14
Figure 3 : Pop-up Signaler une borne.....	14
Figure 4 : Page de réservation.....	15
Figure 5 : Page pour créer une nouvelle réservation.....	15
Figure 6: Choix des bornes.....	15
Figure 7 : Page d'accueil admin.....	16
Figure 8 : Changement d'état d'une borne.....	16
Figure 9 : Création d'une borne.....	16
Figure 10 : Carte non zoomée	17
Figure 11 : Carte zoomée.....	17
Figure 12 : Différentes Alertes de notre application.....	18
Figure 13 : Implémentation du mode sombre.....	19
Figure 14 : Page de connexion Android	26
Figure 15 : Page de connexion iOS.....	26

III. Remerciements

Nous tenons tout d'abord à exprimer notre profonde gratitude à **Thomas Deffains** et **Damien Teissier**, Lead-Tech au CATS Montpellier, pour leur implication précieuse et leur accompagnement tout au long du développement de cette application. Leur expertise technique, leurs retours constructifs et leur disponibilité ont été des atouts majeurs dans l'avancement du projet. Grâce à eux, nous avons pu travailler sur une problématique concrète et porteuse d'avenir, en répondant à un besoin réel des employés du CATS. Cette opportunité nous a permis de mettre en pratique nos compétences tout en développant une solution utile et innovante.

Un immense merci également à **Lysiane Buisson-Lopez**, notre tutrice école, pour son soutien et sa disponibilité tout au long de ce projet. Son accompagnement attentif nous a permis d'avancer et de surmonter les différentes difficultés rencontrées. Toujours à l'écoute, elle a su nous apporter des conseils avisés et des pistes de réflexion pertinentes pour améliorer notre travail, et nous aider à approfondir nos réflexions.

Un grand merci également au **jury de soutenance** qui prend le temps d'examiner notre travail et de s'intéresser à notre projet. Nous sommes reconnaissants de pouvoir présenter le fruit de plusieurs mois d'efforts devant des professionnels et enseignants dont les retours seront, à coup sûr, enrichissants et constructifs.

Nous souhaitons aussi remercier **Polytech Montpellier** pour l'opportunité qui nous est offerte à travers ce projet. Grâce à cette expérience, nous avons pu nous confronter à un cas concret, similaire à ce que nous pourrions rencontrer en entreprise. Ce cadre pédagogique nous a permis de développer nos compétences techniques et organisationnelles tout en nous familiarisant avec les réalités d'un projet informatique en milieu professionnel.

Enfin, une pensée pour nos camarades de **IG5**, avec qui nous avons partagé de nombreux moments d'entraide et de bonne humeur. Ce projet a été une belle expérience collective, et nous sommes reconnaissants d'avoir pu le mener dans un tel cadre.

IV. Introduction

A. Présentation du projet

Ce projet s'inscrit dans le cadre de notre dernière année au département Informatique et Gestion de Polytech Montpellier. Réalisé en collaboration avec le Crédit Agricole Technologies et Services de Montpellier, et notamment nos tuteurs entreprise: Damien Teissier et Thomas Deffains.

D'une durée de deux mois et demi, ce travail a été mené par Lucas Leroy, Auriane Poirier, et Margaux Thirion, répondant à une demande spécifique de l'entreprise. Le projet a bénéficié de l'accompagnement de notre tutrice école, Lysiane Lopez, qui a guidé nos efforts et contribué à orienter nos démarches tout au long du projet.

B. Présentation de l'entreprise

Le projet a été réalisé à la demande de Damien Teissier et Thomas Deffains, tous deux membres du Chapitre Digital Front chez Crédit Agricole Technologies et Services (CA-TS) à Montpellier. Leur équipe prend en charge l'intégralité des développements Frontend, incluant les projets d'applications mobiles, et joue un rôle transverse d'expertise technique pour garantir la cohérence des développements au sein de l'entreprise. Ils ont également une autorité de conception dans le choix des technologies utilisées.

CA-TS occupe une place essentielle au sein du groupe Crédit Agricole, première banque de proximité mutualiste en France, en assurant la conception, le développement et la maintenance des systèmes d'information bancaires pour les 39 Caisses régionales. Grâce à son expertise technique et son engagement envers l'innovation, CA-TS facilite les opérations bancaires quotidiennes de millions de clients et soutient le développement de solutions numériques sécurisées et performantes. La collaboration avec Damien Teissier et Thomas Deffains, experts dans le domaine du développement Android, a permis de cibler précisément les besoins technologiques et d'adapter notre projet pour répondre efficacement à la demande du projet.

C. Contexte et enjeux

Le projet consiste à **développer une application mobile** destinée aux employés du Crédit Agricole Technologies et Services (CA-TS) de Montpellier, leur permettant de **réserver des bornes de recharge pour leurs véhicules électriques** directement depuis leur smartphone. Cette solution vise à optimiser la gestion et l'utilisation des bornes disponibles sur le parking de l'entreprise, tout en offrant une interface intuitive et accessible.

Au-delà de la simple réservation, l'application doit également intégrer un système de suivi de l'état des bornes. Les utilisateurs doivent pouvoir signaler si une borne est dysfonctionnelle, hors service ou en cours de maintenance, afin d'améliorer la fiabilité du service et d'optimiser l'expérience utilisateur. En centralisant ces informations et en facilitant leur mise à jour en temps réel, l'application contribue à une gestion plus efficace des infrastructures de recharge.

1. Enjeux du projet

Optimisation de la gestion des ressources de transport électrique :

Il s'agit de centraliser et simplifier la gestion des bornes de recharges de l'entreprise. L'application doit permettre une utilisation efficace des bornes tout en maximisant leur disponibilité.

Transition écologique et responsabilité sociétale :

Ce projet s'inscrit dans les objectifs du Crédit Agricole de promouvoir une mobilité durable et de contribuer à la réduction de notre empreinte carbone. En facilitant l'utilisation de moyens de transport électriques et en sensibilisant les utilisateurs à des pratiques écoresponsables

Amélioration de l'expérience utilisateur :

L'application doit offrir une interface intuitive et des fonctionnalités adaptées pour répondre aux attentes des employés. Une adoption rapide et un usage régulier par les utilisateurs finaux sont essentiels pour garantir le succès du projet.

Fiabilité et disponibilité des services :

Il est crucial de fournir une solution performante pour assurer la continuité des services, notamment en termes de réservation, d'évaluation de la disponibilité des bornes, et de visualisation de l'état des bornes.

2. Contraintes du projet

Contraintes techniques : L'application a pour but la gestion des bornes de recharge pour véhicule électrique, cela inclut un côté utilisateur, ce qui permettait de réserver des bornes sur le site ainsi que de signaler des bornes défectueuses. Quant à lui, le côté administrateur permet la gestion des bornes.

Contraintes temporelles : Nous devions respecter des délais de livraison stricts afin d'aligner la finalisation de l'application avec le planning défini pour le projet.

Contraintes technologiques : Le développement de l'application en Kotlin Multiplatform (KMP) (iOS et Android) a été imposé par les demandeurs, car il s'agit de la technologie utilisée au sein du CA-TS pour les projets mobiles. Ce choix se justifie par son adoption croissante, notamment dans l'écosystème Android, où il est recommandé officiellement par Google, ainsi que par sa compatibilité avec iOS, offrant ainsi une approche cross-platform efficace.

KMP permet de mutualiser le code métier, facilitant ainsi la maintenance et l'évolution de l'application, tout en conservant la possibilité d'intégrer des fonctionnalités spécifiques à chaque plateforme via des bibliothèques dédiées. Son architecture, proche des standards du développement mobile, simplifie la prise en main pour les développeurs déjà familiers avec Android ou iOS et assure une cohérence dans la gestion des règles métiers au sein du projet.

V. Travail réalisé

A. Démarche de conception

Pour ce projet, il était essentiel de mettre en place une architecture robuste comprenant un backend pour gérer le stockage des informations et la logique des données, ainsi qu'un frontend destiné à communiquer les informations aux utilisateurs.

1. Conception Backend

Choix de la base de données :

Après une évaluation approfondie, nous avons opté pour MongoDB. Cette technologie est largement utilisée dans l'industrie pour sa flexibilité, sa facilité de prise en main et sa capacité à être déployée automatiquement de manière gratuite.

MongoDB offre non seulement une gestion simplifiée et dynamique des schémas de données, mais aussi une intégration aisée avec les environnements de déploiement gratuits, facilitant ainsi la mise en œuvre et l'accès à distance par toute l'équipe.

Choix de la technologie du back :

Dans la conception de notre application mobile, le choix de la technologie backend revêtait une importance cruciale pour assurer la performance et la maintenabilité du système. Après mûre réflexion, nous avons opté pour Java en tant que langage de programmation, en raison de plusieurs facteurs clés qui alignaient cette technologie avec les besoins de notre projet.

Java, avec sa robustesse et son écosystème étendu, nous offrait la stabilité et la flexibilité nécessaires pour développer un backend performant. L'un des avantages significatifs de l'utilisation de Java pour notre backend était la compatibilité avec Gradle, un système de gestion et d'automatisation de build puissant. Gradle supporte spécifiquement des dépendances liées à MongoDB, ce qui simplifiait considérablement l'intégration et la gestion de notre base de données nouvellement adoptée, MongoDB.

De plus, cette décision nous permettait de mettre en pratique dans un contexte professionnel les compétences et les connaissances acquises lors de notre formation en Informatique et Gestion à Polytech Montpellier. Utiliser Java et Gradle nous a non seulement aidés à appliquer les théories apprises en cours, mais a également renforcé notre compréhension pratique de ces outils dans des situations réelles de développement de logiciels.

Choix de la conception du back :

Pour la structure de notre backend, nous avons adopté une variante de l'architecture Modèle-Vue-Contrôleur (MVC), spécifiquement adaptée pour répondre aux exigences d'une application backend. Cette approche est communément désignée sous le terme de modèle MVC ou parfois référencée comme une architecture en trois couches, comprenant :

- **Modèle** : Représente la couche de données de l'application. Dans notre cas, il correspond aux objets et mécanismes de gestion de la base de données MongoDB, manipulés à travers des classes de modèles qui reflètent les structures de données et les logiques métiers.
- **Repository** : Agit comme une couche d'abstraction entre le modèle et la source de données. Les repositories facilitent les interactions avec la base de données pour récupérer, stocker et mettre à jour les données nécessaires.
- **Service** : Constitue la logique métier de l'application. Les services définissent les règles métiers et les opérations complexes en utilisant les données fournies par les repositories. Cette couche s'assure que les demandes du frontend sont traitées efficacement et correctement.
- **Contrôleur** : Sert de lien entre le frontend et le backend, gérant les interactions utilisateur. Il reçoit les requêtes du frontend, les dirige vers les services appropriés pour traitement, et renvoie les réponses adéquates.

Cette structuration permet non seulement une séparation claire et une gestion efficace des responsabilités dans notre backend, mais elle assure aussi une maintenabilité et une évolutivité optimales du système.

Déploiement :

L'une des étapes finales de notre projet consistait à déployer notre application pour permettre une utilisation en conditions réelles. Pour ce faire, nous avons choisi d'utiliser Render, une plateforme de déploiement d'applications offrant un service gratuit, ce qui représentait un avantage significatif pour notre projet.

Render se distingue par sa simplicité d'utilisation, permettant une mise en œuvre rapide et efficace sans nécessiter de connaissances approfondies en configuration de serveurs. Cette accessibilité nous a permis de concentrer nos efforts sur le développement de l'application plutôt que sur les détails techniques du déploiement.

2. Conception Frontend

a) Architecture du projet

Organisation du Projet avec Kotlin Multiplatform :

Notre projet est structuré en trois parties principales conformément à l'architecture Kotlin Multiplatform, qui optimise la réutilisation du code tout en permettant des adaptations spécifiques à chaque plateforme :

- **iOS App et AndroidApp** : Ces modules servent essentiellement de points d'entrée pour les applications respectives sur chaque plateforme. Ils sont principalement vides et contiennent seulement le code nécessaire pour lancer l'émulateur ou l'appareil réel correspondant. Cette approche permet de garder ces modules aussi légers que possible.
- **Shared** : Le module Shared est le cœur de notre architecture et est subdivisé en trois sous-modules pour organiser la logique de l'application :
 - **Common Main** : Ce sous-module contient la majorité de notre code, y compris les vues, modèles, repositories, et viewmodels. Toute la logique métier et la gestion des données sont centralisées ici, ce qui facilite la maintenance et la cohérence entre les différentes plateformes.
 - **android main et ios Main** : Ces sous-modules contiennent du code spécifique à chaque plateforme. Nous utilisons le mécanisme `expect/actual` de Kotlin Multiplatform pour définir des interfaces communes dans `commonMain` tandis que les implémentations spécifiques à chaque plateforme se trouvent dans `androidMain` et `iosMain`. Ce système est particulièrement utile pour gérer des fonctionnalités comme les clients HTTP qui nécessitent des adaptations selon l'environnement d'exécution.

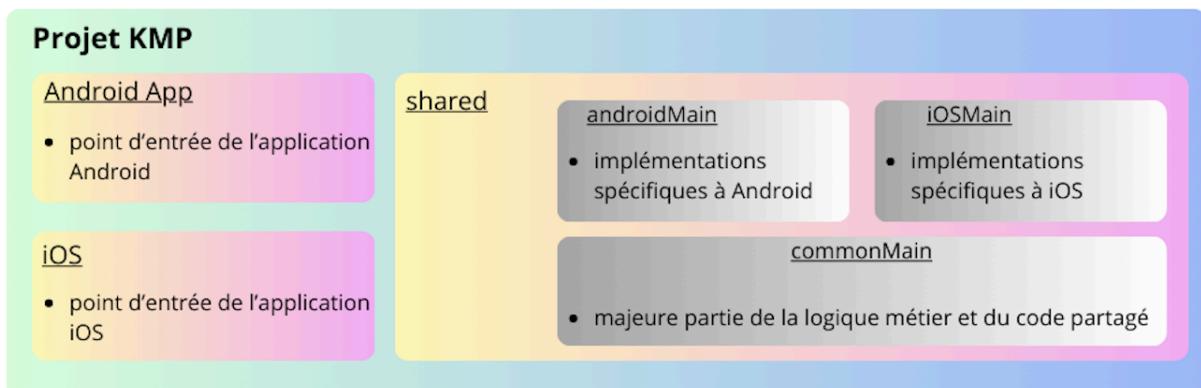


Figure 1 : Schéma de l'architecture du projet

Cette organisation rigoureuse permet une mutualisation maximale du code via `commonMain` tout en offrant la flexibilité nécessaire pour incorporer des fonctionnalités spécifiques à chaque plateforme grâce aux sous-modules `androidMain` et `iosMain`. Grâce à cette structure, nous avons pu minimiser les efforts de développement redondants et maintenir une haute cohérence fonctionnelle à travers les différentes versions de l'application.

Choix de l'architecture :

Pour notre application mobile, nous avons opté pour l'architecture MVVM (Model-View-View Model), qui est particulièrement adaptée à Kotlin et aux environnements de développement modernes comme Jetpack Compose. Ce choix offre plusieurs avantages significatifs :

1. Séparation des préoccupations

L'un des principaux avantages de l'architecture MVVM est la claire séparation des préoccupations qu'elle propose :

- **Modèle (Model)** : Gère les données et la logique métier.
- **Vue (View)** : Présente les données (l'interface utilisateur) et déclenche des événements au modèle de vue.
- **Modèle de Vue (ViewModel)** : Agit comme un intermédiaire entre le modèle et la vue, gérant la logique de présentation et les états de l'interface utilisateur.

Cette séparation facilite la maintenance et le test des différentes composantes de l'application, puisque chaque partie peut être développée et testée indépendamment des autres.

2. Facilitation des tests

Avec MVVM, il est plus facile de tester les composantes de l'application car le ViewModel ne dépend pas directement de la vue. Cela permet de tester la logique de présentation sans nécessiter une interface utilisateur, ce qui peut considérablement accélérer le développement et garantir une plus grande qualité du code.

3. Réutilisation du code

Le ViewModel dans MVVM peut souvent être réutilisé entre les plateformes lorsqu'il est combiné avec des solutions comme Kotlin Multiplatform Mobile. Cela réduit le besoin de dupliquer la logique de présentation pour Android et iOS, minimisant ainsi les efforts de développement et les risques d'erreurs.

b) Maquettes & prototype

Réalisation des maquettes :

La conception visuelle et l'interaction utilisateur de notre application ont été soigneusement élaborées à travers des maquettes détaillées, qui ont joué un rôle crucial dans la validation du design et de l'expérience utilisateur avant le développement proprement dit.

1. Maquettes pour les Utilisateurs et les Administrateurs

Nous avons développé des maquettes distinctes pour les interfaces utilisateur et administrateur afin de répondre aux besoins spécifiques de chaque type d'utilisateur :

- **Côté Utilisateur (Annexe 1)** : Les maquettes ont été conçues pour offrir une expérience intuitive et engageante, facilitant la navigation entre les différentes fonctionnalités de l'application.

- **Côté Administrateur (Annexe 2)** : Les maquettes pour l'interface administrateur ont été élaborées pour permettre une gestion efficace et simplifiée des opérations backend, telles que la surveillance des données utilisateur et la gestion des paramètres de l'application.

2. Interactivité des Maquettes

Les maquettes étaient animées et interactives, permettant de simuler l'expérience utilisateur finale. En cliquant sur les différents éléments, comme les boutons, les utilisateurs pouvaient naviguer de manière fluide d'une page à l'autre. Cette interactivité a été essentielle pour tester la logique de navigation et l'ergonomie de l'interface, assurant que les transitions et les interactions étaient à la fois logiques et esthétiquement plaisantes.

B. Résultats

1. Résultats Intermédiaires

Tout au long de ce projet, nous avons mis en place un système de suivi rigoureux pour assurer la transparence et la continuité dans nos avancements. Ce système comprenait des rapports préliminaires, hebdomadaires, et un rapport final.

1. Rendus Préliminaires

Dès le début du projet, nous avons établi les fondations de notre planification :

- **Gantt prévisionnel (voir Annexe 3)** : Ce diagramme de Gantt initial a servi à définir les étapes clés du projet, les délais associés, et la répartition des tâches. Il a été un outil essentiel pour visualiser la progression du projet et pour anticiper les besoins en ressources et en temps.
- **Lettre de mission (Voir Annexe 7)** : Ce document a formalisé l'objectif du projet, les attentes des tuteurs, et les responsabilités de chaque membre de l'équipe. Elle a servi de référence tout au long du projet pour aligner nos efforts avec les objectifs fixés.
- **Maquettes (Voir Annexe 1 et 2)** : Les maquettes ont été développées pour visualiser les interfaces utilisateur et administrateur de l'application. Elles ont permis de simuler l'expérience utilisateur et de valider le design avant le passage au développement. Les maquettes animées ont également facilité la compréhension des flux de navigation et des interactions prévues.
- **MCD (Voir Annexe 6)** : Le Modèle Conceptuel de Données a été crucial pour structurer la base de données nécessaire au fonctionnement de l'application. Ce modèle a permis de définir clairement les entités, leurs attributs, et les relations entre elles, garantissant ainsi une gestion efficace et cohérente des données à travers l'application.

2. Rapports Hebdomadaires

Pour maintenir une dynamique de projet efficace et réactive, nous avons instauré des réunions hebdomadaires chaque vendredi de 14h à 15h. Ces réunions étaient l'occasion de:

- **Présenter les minutes** : Documenter les discussions, les décisions prises, et les actions à entreprendre, assurant ainsi la continuité entre les différentes sessions de travail.
- **Présenter les travaux accomplis durant la semaine** : Chaque membre de l'équipe partageait ses progrès, permettant ainsi une évaluation continue de l'avancement du projet et l'ajustement rapide des plans en fonction des résultats obtenus et des défis rencontrés.

2. Résultat final

Le point culminant de notre projet a été la présentation du produit final :

Rendu final de l'application : Cette étape a consisté à démontrer le fonctionnement de l'application, à valider les fonctionnalités développées par rapport aux exigences initiales, et à recevoir les retours des tuteurs et autres parties prenantes. Le produit final reflétait l'intégration des différentes composantes développées et testées tout au long du projet.

a) Backend

Notre back-end repose sur **huit tables** principales, structurant les données essentielles au bon fonctionnement de l'application. Il est **déployé et opérationnel**, accessible via le lien suivant : [Back_CATS](#).

Afin de faciliter son utilisation et son intégration avec le front-end, nous avons produit un **document de référence (Annexe 8)** détaillant toutes les routes disponibles dans l'API. Ce document recense pour chaque route :

- L'**URL** de l'endpoint,
- Le **type** de requête (POST, GET, PUT, DELETE, PATCH),
- Le **body** attendu (paramètres à fournir dans la requête),
- La **réponse** renvoyée par l'API.

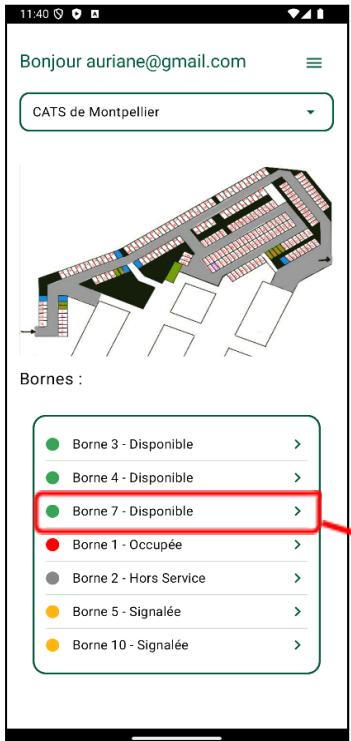
Ce document permet aux développeurs de consulter rapidement les différentes fonctionnalités du back-end, facilitant ainsi l'intégration et la communication entre les services.

b) Frontend - Utilisateurs

Le code de la partie frontend est trouvable sur github à cette adresse : [Front_CATS](#).

- Page d'accueil

La page d'accueil de notre application offre une vue d'ensemble intuitive et interactive des fonctionnalités clés, permettant aux utilisateurs de naviguer efficacement dans le système.



À l'ouverture de la page, les utilisateurs sont immédiatement informés du site sur lequel ils se trouvent. La carte du site en question est affichée pour l'utilisateur.

États des bornes : Chaque borne est clairement marquée selon son état actuel :

- **Signalée** : Indique une borne ayant un problème et ayant été signalé récemment.
- **Occupée** : Montre que la borne est actuellement utilisée.
- **Hors Service (HS)** : Indique que la borne n'est pas opérationnelle.
- **Disponible** : Signifie que la borne est libre et peut être réservée.

Figure 2 : Page d'accueil

Interaction avec les bornes : En cliquant sur une borne spécifique, une **fenêtre pop-up** s'ouvre, offrant à l'utilisateur la possibilité de signaler un problème avec la borne en question.

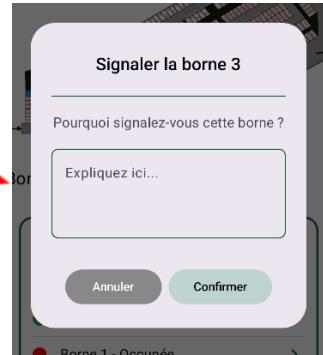


Figure 3 : Pop-up Signaler une borne

La page d'accueil de notre application intègre également un menu de navigation accessible. On peut donc accéder aux pages "**Accueil**", "**Réservation**" et "**Déconnexion**".

- Page de réservation

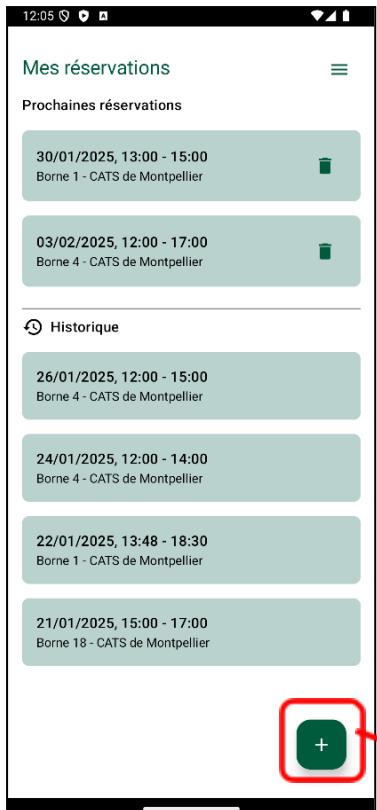


Figure 4 : Page de réservation

La section intitulée "**Prochaines réservations**", liste les réservations futures par date et heure, spécifiant la borne réservée et son emplacement. Chaque réservation peut être supprimée grâce au logo poubelle.

La section intitulée "**Historique**", liste les réservations passées par date et heure, spécifiant la borne réservée et son emplacement.

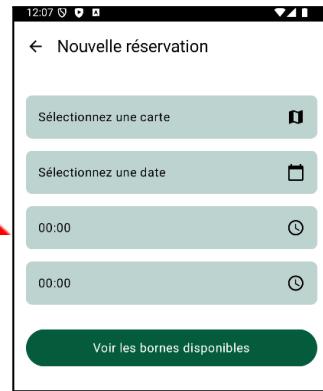


Figure 5 : Page pour créer une nouvelle réservation



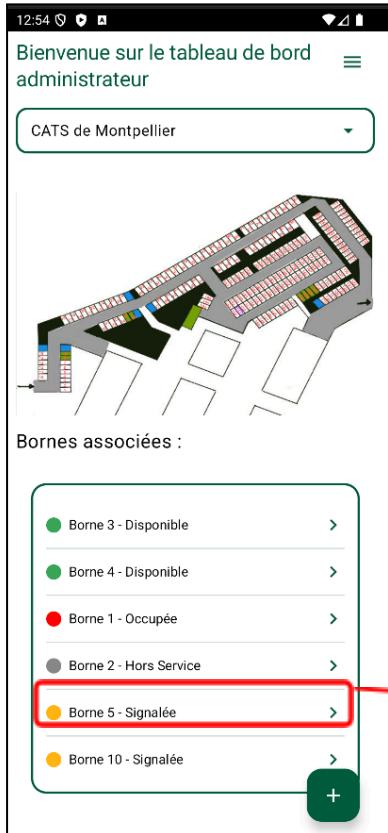
Figure 6: Choix des bornes

Nous avons implémenté un bouton qui permet de créer une nouvelle réservation.

Ce bouton envoie vers une nouvelle page qui permet de remplir les informations nécessaires pour réserver une borne sur un créneau horaire.

c) Frontend - Administrateur

- Page d'accueil



À l'ouverture de la page, les administrateurs sont immédiatement informés du site sur lequel ils se trouvent. La carte du site en question est affichée pour l'admin.

États des bornes : Chaque borne est clairement marquée selon son état actuel :

- **Signalée** : Indique une borne ayant un problème et ayant été signalé récemment.
- **Occupée** : Montre que la borne est actuellement utilisée.
- **Hors Service (HS)** : Indique que la borne n'est pas opérationnelle.
- **Disponible** : Signifie que la borne est libre et peut être réservée.

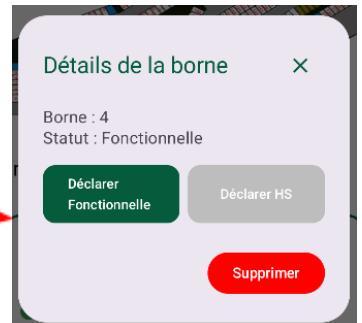


Figure 7 : Page d'accueil admin

Figure 8 : Changement d'état d'une borne

Figure 9 : Crédit d'une borne



Lorsqu'un administrateur clique sur une borne il peut alors changer l'état de cette borne, la passer en HS ou fonctionnelle ou la supprimer.

Sur cette page, l'administrateur peut également cliquer sur le bouton “+” afin de créer une nouvelle borne.

L'administrateur commence par choisir le **site** pour lequel une nouvelle borne doit être ajoutée. Une fois le site sélectionné, l'administrateur peut consulter la carte interactive du site. Cette carte aide à identifier le **numéro** de l'emplacement où la nouvelle borne sera installée. Il est ensuite requis de spécifier le **type de borne** qu'aura la nouvelle borne.

La création de la nouvelle borne est conditionnée par le remplissage complet de tous les champs nécessaires dans le formulaire. Seulement après avoir correctement renseigné ces informations, le bouton de création de la borne devient actif, permettant ainsi à l'administrateur de finaliser l'ajout.

d) Frontend - Général

- **carte zoomable**

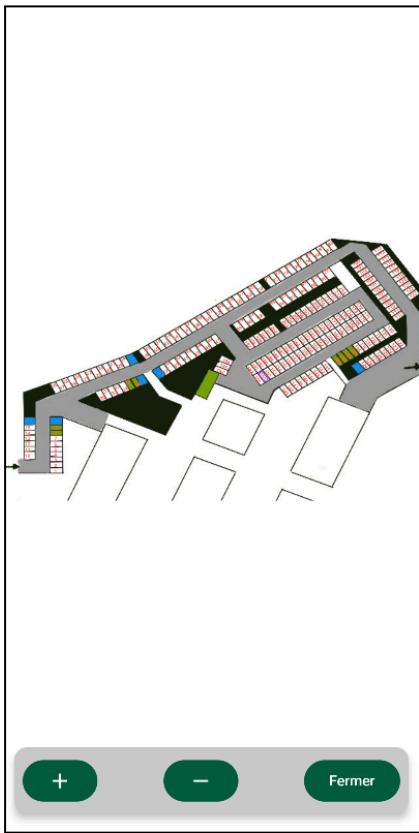


Figure 10 : Carte non zoomée



Figure 11 : Carte zoomée

L'une des fonctionnalités de notre application est la carte zoomable, elle est présente dans plusieurs pages, autant du côté administrateur qu'utilisateur. Elle est conçue pour améliorer la navigation et la compréhension des localisations des bornes sur le parking.

Vue initiale : Au départ, la carte est affichée en petit format pour donner une vue d'ensemble du site concerné. Cette perspective permet aux utilisateurs de voir rapidement la disposition générale.

Fonctionnalité de zoom : En cliquant sur la carte, les utilisateurs peuvent activer la fonction de zoom. Cette interaction permet d'agrandir une zone spécifique de la carte pour une meilleure visualisation des détails.

- **Implémentation de loader**

Dans notre application, l'attente de chargement des données depuis le backend est gérée à l'aide de loaders visuels. Ces éléments d'interface utilisateur sont essentiels pour maintenir une expérience fluide et informer l'utilisateur que des données sont en cours de récupération.

Les loaders sont activés automatiquement dès que l'application doit récupérer des informations du backend. Cela peut se produire lors de l'accès à des détails spécifiques sur les bornes, la mise à jour des états des bornes, ou lors de la consultation des historiques de réservation.

- **Alertes**

Afin d'améliorer la communication interactive et de renforcer la sûreté des opérations au sein de notre application, un système d'alertes a été intégré. Ce système est essentiel pour guider les utilisateurs lors de leurs interactions avec l'application et pour prévenir les erreurs potentielles. Les alertes sont classées en trois catégories principales :

1. **Alertes d'Erreur**

Ces alertes sont déclenchées lorsqu'une action ne peut être complétée correctement.

Exemple pratique : Si un utilisateur saisit incorrectement son adresse email lors de l'enregistrement ou de la connexion.

2. **Alertes de Succès**

Ces notifications confirment le succès d'une opération, renforçant ainsi la satisfaction de l'utilisateur et la clarté des processus.

Exemple pratique : Lorsqu'une nouvelle borne est ajoutée avec succès à un site, une alerte de succès apparaît avec le message "Borne ajoutée avec succès au site." Cela assure à l'utilisateur que son action a eu l'effet désiré.

3. **Alertes de Confirmation**

Les alertes de confirmation sont cruciales pour éviter les actions irréversibles ou importantes sans une vérification explicite de l'utilisateur.

Exemple pratique : Avant de supprimer une borne ou une réservation, une alerte de confirmation demande à l'utilisateur "Êtes-vous sûr de vouloir supprimer cette borne ?" Ceci est accompagné d'options pour annuler ou confirmer l'action, minimisant ainsi le risque d'erreurs accidentelles.

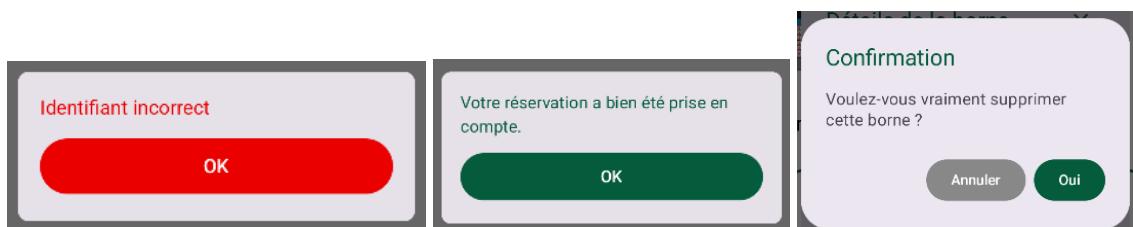


Figure 12 : Différentes Alertes de notre application

- **Intégration du Mode Sombre**

Afin de répondre aux préférences et au confort des utilisateurs, notre application inclut une option de mode sombre, ou Dark Mode. Cette fonctionnalité est devenue essentielle dans les interfaces modernes pour plusieurs raisons, notamment la réduction de la fatigue oculaire et l'économie d'énergie.

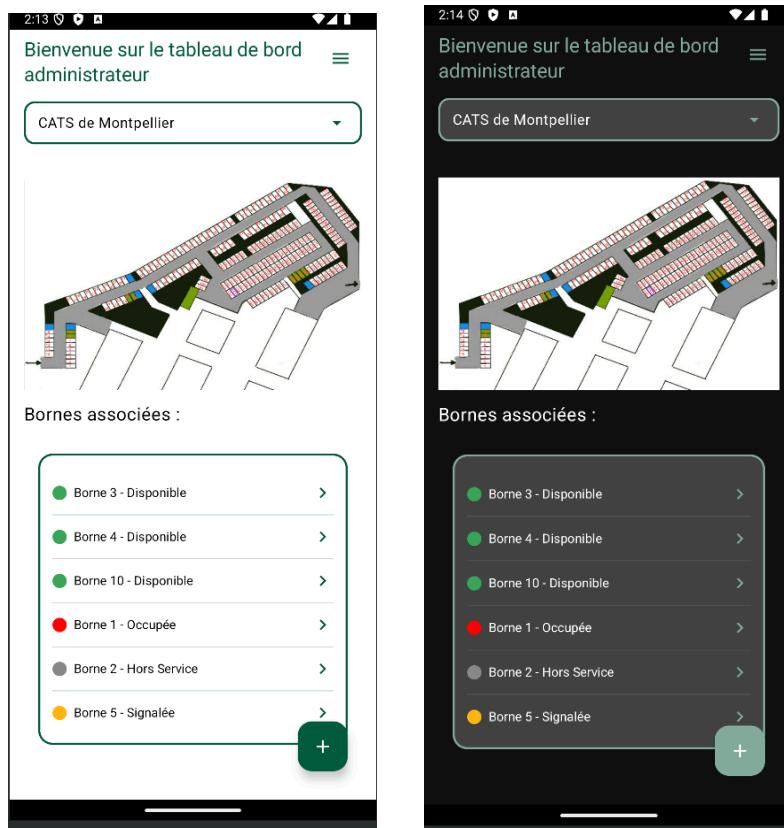


Figure 13 : Implémentation du mode sombre

VI. Gestion du projet

A. Méthodologies utilisées

Pour assurer un suivi efficace du projet, nous avons mis en place plusieurs outils et méthodes de gestion. Tout d'abord, nous avons élaboré un **Gantt prévisionnel** (**Voir Annexe 3**) basé sur une **liste des tâches classées par ordre de priorité** (élévée, moyenne, faible) (**Voir Annexe 5**). Pour chaque tâche, nous avons estimé le temps nécessaire à sa réalisation, prenant en compte la complexité et les défis potentiels associés. Ces estimations ont été intégrées dans notre Gantt pour créer une timeline prévisionnelle, fournissant ainsi un cadre de travail clair et des objectifs temporels précis pour chaque phase du projet.

Ce planning a servi de guide tout au long du développement, mais a dû être mis à jour régulièrement en fonction des imprévus et des ajustements nécessaires.

Nous avons également instauré des **réunions hebdomadaires avec les demandeurs**, qui se tenaient chaque vendredi après-midi, afin de faire le point sur l'avancement du projet, discuter des difficultés rencontrées et définir les objectifs pour la semaine suivante. Pour chaque réunion, nous rédigions des minutes de réunion, détaillant les tâches effectuées jour par jour, les questions à poser aux demandeurs, ainsi que leurs retours sur notre travail. Ces minutes étaient ensuite transmises à notre tutrice école afin d'assurer un suivi rigoureux du projet et garantir une bonne communication entre toutes les parties prenantes. Ces échanges réguliers nous ont permis de maintenir une vision claire du projet et d'adapter notre organisation en conséquence.

Enfin, un **dépôt Git dédié** a été mis en place pour partager les documents liés au projet avec les demandeurs. Ce dépôt centralise des fichiers essentiels tels que les diagrammes de Gantt, les minutes hebdomadaires, la lettre de mission et d'autres livrables, permettant ainsi aux demandeurs d'y accéder facilement à tout moment. Cette organisation nous a permis d'assurer une transparence totale sur l'avancement du projet, tout en simplifiant la transmission et la validation des documents par nos interlocuteurs.

B. Bonnes pratiques

1. Organisationnelles

L'un des éléments clés qui a contribué à la bonne gestion du projet a été **une répartition efficace du travail** entre les membres de l'équipe. Dès le départ, nous avons décidé d'attribuer des responsabilités spécifiques afin de maximiser notre productivité et d'assurer une progression fluide. Lucas s'est concentré sur le front-end côté administrateur, Auriane sur le front-end côté utilisateur, tandis que Margaux s'est initialement chargée du back-end. Cette répartition nous a permis d'avancer en parallèle sur les différentes parties du projet et de gagner en efficacité. De plus, le fait d'avoir des tâches bien distinctes nous a évité d'empiéter mutuellement sur le travail des autres, limitant ainsi les conflits de développement et garantissant une meilleure fluidité dans l'intégration du code. Une fois le back-end mis en place et opérationnel, Margaux a pu apporter son aide sur le développement du front-end, notamment pour accélérer l'intégration des fonctionnalités et assurer une meilleure cohérence entre les différentes parties de l'application. Ce mode de fonctionnement nous a permis d'optimiser notre temps et de nous adapter aux besoins du projet au fur et à mesure de son avancement.

Par ailleurs, **l'utilisation rigoureuse de Git** a joué un rôle central dans notre organisation et notre collaboration. Nous avons mis en place un dépôt distinct pour le front-end et un autre pour le back-end, afin de bien séparer les responsabilités et de faciliter le suivi des développements. Chaque membre de l'équipe travaillait sur sa propre branche, ce qui nous permettait de tester et d'intégrer nos fonctionnalités sans risquer d'impacter le travail des autres. Cette méthodologie a non seulement facilité la collaboration au sein de l'équipe, mais elle a aussi permis aux demandeurs du projet de récupérer facilement notre travail et de suivre l'avancement du développement en temps réel. Grâce à cette gestion méthodique du code source, nous avons pu éviter de nombreux conflits et simplifier l'intégration des différentes contributions, garantissant ainsi une meilleure stabilité du projet.

Un autre élément qui a grandement facilité notre organisation a été la mise en place d'un **fichier récapitulatif des routes API**, mis en place par Margaux, responsable du back-end. Ce document (**voir annexe 8**) listait toutes les routes mises en place, avec les requêtes attendues (body) et les réponses associées, permettant ainsi à Auriane et Lucas, en charge du front-end, d'accéder rapidement aux informations nécessaires sans avoir à explorer directement le code du back-end. Grâce à cette documentation claire et mise à jour régulièrement, nous avons pu éviter des allers-retours inutiles et optimiser le temps de développement, garantissant ainsi une intégration fluide entre les différentes parties de l'application.

Ces bonnes pratiques organisationnelles nous ont permis de travailler de manière structurée et efficace, tout en assurant une certaine flexibilité pour s'adapter aux imprévus et aux ajustements nécessaires.

2. Techniques

Avant de nous lancer dans le développement du front-end, nous avons pris le temps de **concevoir des maquettes sur Figma (Voir Annexe 1 et 2)**. Cette phase de conception a été essentielle pour visualiser l'interface utilisateur, définir les différents écrans de l'application et assurer une cohérence graphique et ergonomique dès le début du projet. Cela nous a également permis d'anticiper d'éventuelles contraintes et d'ajuster certains éléments avant même de commencer le codage, évitant ainsi des modifications coûteuses en temps par la suite.

Concernant la base de données, nous avons réalisé un **Modèle Conceptuel de Données (MCD) (Voir Annexe 6)** afin de structurer au mieux les informations à stocker et d'identifier les tables et relations nécessaires. Cette réflexion préalable nous a permis d'avoir une base de données bien conçue, garantissant ainsi une gestion efficace des données et facilitant leur exploitation dans l'application.

Une autre bonne pratique technique que nous avons mise en place est **le test systématique après chaque implémentation de fonctionnalités**. Plutôt que d'attendre la fin du développement pour tester l'application dans son ensemble, nous avons adopté une approche incrémentale en vérifiant le bon fonctionnement de chaque nouvelle fonctionnalité sur émulateur android dès son ajout. Cette méthode nous a permis d'identifier rapidement d'éventuels bugs et de les corriger avant qu'ils ne s'accumulent, garantissant ainsi une plus grande stabilité du projet au fil du développement.

Dans un souci d'**optimisation de l'expérience utilisateur (UX)**, nous avons intégré plusieurs bonnes pratiques. Tout d'abord, nous avons mis en place des alertes avant et après chaque action importante, comme la suppression d'une réservation ou la confirmation du signalement d'une borne. Cela permet d'éviter les erreurs involontaires et d'offrir à l'utilisateur un retour clair sur ses actions. De plus, nous avons adopté le principe du "1 action par écran", qui consiste à limiter chaque écran de l'application à une seule action principale. Cette approche améliore la lisibilité et l'intuitivité de l'interface, en guidant l'utilisateur étape par étape sans surcharge visuelle ni complexité excessive.

En combinant ces différentes bonnes pratiques, nous avons cherché à garantir un développement structuré, une architecture optimisée et une expérience utilisateur fluide, tout en facilitant la maintenance et l'évolutivité du projet.

VII. Analyse critique

A. Analyse des choix techniques et organisationnels

Au début du projet, nous avons établi un fichier Excel listant les fonctionnalités en leur attribuant une priorité (élevée, moyenne, faible). Sur cette base, nous avons conçu un **diagramme de Gantt prévisionnel** (*Voir Annexe 3*) pour estimer le temps nécessaire au développement, aux tests et aux livrables. Cependant, la prise en main de Kotlin Multiplatform (KMP) s'est révélée plus complexe que prévu, ralentissant notre progression. Face à ces imprévus, nous avons dû réévaluer nos priorités en cours de projet et abandonner certaines fonctionnalités de priorité faible.

De plus, notre Gantt a dû être ajusté plusieurs fois pour s'adapter aux contraintes techniques et au temps réel d'implémentation. Pour un suivi précis de notre progression, nous avons maintenu un **Gantt réel** (*voir Annexe 4*) qui documentait les dates réelles de début et de fin de chaque tâche. Ce suivi détaillé nous a permis de visualiser clairement les avancements et les retards, facilitant les décisions rapides concernant les réallocations de ressources et les changements de priorités. En plus du Gantt, nous avons utilisé **une liste de tâches** (*Voir Annexe 5*). Chaque tâche était assignée avec une date prévisionnelle de début et de fin, mais, comme souvent dans les projets dynamiques, certaines tâches ont été complétées plus rapidement tandis que d'autres ont pris plus de temps que prévu. Pour chaque tâche, nous avons également enregistré les dates réelles de début et de fin. Cette méthode de suivi nous a aidé à identifier les goulets d'étranglement et à ajuster nos efforts en conséquence, assurant ainsi que les tâches critiques pour le lancement du produit n'étaient pas retardées.

Cette expérience nous a appris l'importance d'une planification flexible, intégrant des marges pour gérer les imprévus. À l'avenir, nous anticiperons mieux les défis techniques afin d'ajuster plus précisément nos estimations et nos priorités.

Un autre point qui aurait pu être mieux anticipé concerne les tests sur iOS. Dans notre approche, nous avons d'abord développé et testé nos fonctionnalités sur Android, avant de vérifier leur compatibilité avec iOS en fin de parcours. Or, cette stratégie nous a posé un problème majeur : certaines des dépendances que nous avions intégrées étaient compatibles uniquement avec Android, ce qui a conduit à des incompatibilités bloquantes au moment du passage sur iOS. Si nous avions testé régulièrement sur les deux plateformes dès le début du développement, nous aurions pu identifier et corriger ces problèmes beaucoup plus tôt, évitant ainsi des modifications lourdes en fin de projet.

Cependant, tester sur iOS n'était pas une tâche aisée. Chaque modification nécessitait un temps de build d'environ 10 minutes, rendant les itérations longues et peu pratiques. Ce facteur a largement contribué à notre choix initial de nous concentrer d'abord sur Android, bien que, rétrospectivement, une approche plus équilibrée entre les deux plateformes aurait permis d'éviter ces ajustements tardifs.

Ces différentes difficultés nous ont apporté des enseignements importants sur la gestion du temps et des priorités dans un projet technique. Elles nous ont appris l'importance d'une planification réaliste, d'un testing régulier sur toutes les plateformes ciblées, et d'une sélection rigoureuse des dépendances dès le début du développement. Ces leçons seront précieuses pour de futurs projets, où nous veillerons à mieux anticiper ces défis afin d'optimiser notre efficacité et d'éviter les ajustements coûteux en fin de parcours.

B. Difficultés rencontrées

1. Front-end

L'un des premiers défis auxquels nous avons été confrontés lors du développement du front-end a été **l'apprentissage de Kotlin Multiplatform (KMP) et la mise en place initiale du projet**. N'étant pas familiers avec cette technologie, nous avons dû nous adapter à son fonctionnement particulier, notamment en ce qui concerne la gestion du partage de code entre les différentes plateformes. Cette phase d'apprentissage a nécessité une recherche approfondie sur la documentation officielle et les ressources disponibles en ligne. Heureusement, nous avons pu compter sur le soutien des demandeurs du projet, qui nous ont apporté des explications et des conseils pour nous aider à structurer correctement notre code et exploiter pleinement les capacités offertes par KMP.

Le choix des dépendances utilisées a également été un point de réflexion important. Étant donné que notre application repose sur Kotlin Multiplatform, il était essentiel d'opter pour des bibliothèques compatibles avec cet environnement afin de garantir une compatibilité optimale entre Android et iOS. Nous avons initialement fait quelques choix de dépendances qui se sont révélés inadaptés, ce qui nous a conduit à devoir refondre une partie du projet en cours de route. Cette refonte a été une perte de temps et d'énergie, mais elle nous a permis de mieux comprendre l'importance d'anticiper ces aspects techniques dès le début. Cette expérience nous a appris qu'il est crucial d'évaluer minutieusement les dépendances et d'adopter une approche plus méthodique pour éviter de refaire inutilement certaines parties du développement.

Une autre difficulté majeure a été **l'affichage de la carte récupérée depuis le back-end**. Dès le départ, l'objectif était non seulement d'afficher cette carte, mais aussi de la rendre interactive en permettant aux utilisateurs de zoomer, de la déplacer et d'interagir avec les bornes de recharge. Cependant, l'intégration d'une carte dans une application multiplateforme s'est révélée bien plus complexe que prévu. Nous avons dû tester différentes approches, effectuer de nombreux essais et ajustements avant d'obtenir simplement un affichage fonctionnel. Ce processus a demandé beaucoup de temps et d'efforts, ce qui a impacté notre avancement sur d'autres fonctionnalités, et nous a forcé à faire preuve d'adaptabilité.

Après discussion avec les demandeurs du projet, nous avons fait le choix de mettre de côté l'interactivité de la carte afin de nous concentrer sur l'essentiel : afficher correctement les bornes de recharge et garantir une expérience utilisateur fluide. Pour pallier cette limitation, nous avons adapté notre approche en proposant un affichage des bornes sous forme de liste plutôt qu'au sein de la carte. Cette décision nous a permis de respecter les délais du projet tout en assurant une utilisation efficace de l'application.

En somme, ces différentes difficultés nous ont permis de gagner en expérience et en rigueur dans le développement front-end. Elles nous ont aussi appris l'importance d'une bonne anticipation technique, d'une recherche approfondie avant l'implémentation et d'un échange constant avec les demandeurs du projet pour garantir un produit final conforme aux attentes.

2. Back-end

Choix initial de la base de données :

Initialement, nous avions opté pour une solution SQL en utilisant PostgreSQL, reconnue pour sa fiabilité et ses performances dans la gestion des données complexes. Cependant, nous avons rencontré des difficultés lors du déploiement de notre solution. La première étape consistait à initialiser la base de données puis de déployer notre solution. Ce processus devrait permettre à notre équipe de trois personnes de travailler simultanément sans la contrainte de devoir cloner en local le backend.

Face aux défis de déploiement de PostgreSQL et aux limitations des options de déploiement gratuit disponibles sur le marché, nous avons dû reconstruire notre choix. Après une évaluation approfondie, nous avons opté pour MongoDB.

Hébergement :

Cependant, nous avons rencontré certains défis avec Render, notamment en ce qui concerne les performances. Nous avons observé une latence prolongée dans les réponses aux requêtes HTTP, ce qui pourrait impacter l'expérience utilisateur. Bien que cet inconvénient ne compromette pas la fonctionnalité globale de l'application, il est crucial de considérer cet aspect pour les futures itérations ou pour un déploiement à plus grande échelle.

C. Éco-conception

Face à l'essor de la mobilité électrique et aux enjeux environnementaux, cette application vise à optimiser l'utilisation des infrastructures de recharge existantes, tout en simplifiant la gestion pour les équipes du Crédit Agricole.

Dans cette perspective, nous avons accordé une attention particulière à l'éco-conception de l'application, en mettant en œuvre des pratiques durables tout au long du processus de développement. Pour ce faire, nous avons utilisé une fiche d'autoévaluation de l'éco-conception fournie par le gouvernement français ("[NumEcoDiag](#)").

Pour notre application, nous avons identifié 50 critères évaluables. Sur les 50 critères évaluables nous sommes conformes pour 23 d'entre eux. Nous allons détailler quelques critères que l'on respecte dans cette partie.

1. Critères détaillés

- **Critère 1.1 : Le service numérique a-t-il été évalué favorablement en termes d'utilité en tenant compte de ses impacts environnementaux ?**

Valeur ajoutée pour les utilisateurs :

- L'application répond à un **besoin concret** en centralisant et en simplifiant la gestion des bornes de recharge pour les employés du Crédit Agricole Montpellier.
- Elle permet aux utilisateurs de réserver facilement des bornes, d'évaluer leur disponibilité en temps réel et de consulter l'état des bornes, optimisant ainsi leur expérience et leur efficacité.
- Elle contribue à une gestion plus rationnelle des ressources en maximisant la disponibilité des bornes et en évitant les conflits ou les surcharges inutiles.

Réponse à un besoin essentiel :

- Ce projet s'inscrit dans le contexte de la transition énergétique et de la mobilité durable, des enjeux cruciaux à l'heure actuelle.
 - L'application améliore non seulement les opérations internes de l'entreprise, mais contribue également aux objectifs de **responsabilité sociétale** du Crédit Agricole en facilitant l'adoption des moyens de transport électriques.
- **Critère 4.12 : Le service numérique informe-t-il l'utilisateur du format de saisie attendu avant sa validation ?**

Les champs de saisie, tels que les formulaires de réservation ou d'identification, affichent des **indications explicites** (placeholders ou libellés) pour guider l'utilisateur. Par exemple : **Champs de date** : Picker brider pour n'accepter que des indications valides.

Champs de texte : Exemples de 'placeholders' clairs (par exemple : 'Entrez votre identifiant')."

Lors de la saisie, un système d'alerte dynamique vérifie la validité des données (par exemple, format d'une adresse email). Cela permet de prévenir l'utilisateur immédiatement en cas d'erreur et d'assurer une validation finale sans encombre.

De fait, le service numérique respecte ce critère en guidant les utilisateurs sur le format de saisie attendu dès le départ, et en fournissant des feedbacks clairs avant la validation. Cela garantit une expérience utilisateur fluide et réduit les erreurs liées à des saisies incorrectes.

2. Analyse critique

NumEcoDiag est un outil utile pour soutenir la transition écologique de notre application, mais il n'est pas sans défauts. Certaines limitations conceptuelles nous ont empêchés d'implémenter pleinement toutes les fonctionnalités éco-conçues que nous souhaitions. Par exemple, le critère "*Le service numérique utilise-t-il des mécanismes de mise en cache pour la totalité des contenus transférés dont il a le contrôle ?*" pose problème. En effet, nous ne pouvons pas mettre en cache tous les contenus, car, par exemple, les informations concernant les bornes changent trop fréquemment pour être stockées de cette manière. Toutefois, nous avons réussi à mettre en place un système de cache pour d'autres types de données, comme les cartes. Pour celles-ci, l'application vérifie d'abord si une mise à jour est nécessaire avant de demander une nouvelle carte au serveur ou d'afficher celle déjà enregistrée localement.

D. Améliorations futures

Bien que l'application soit pleinement fonctionnelle, certaines améliorations pourraient être apportées afin d'optimiser l'expérience et l'interface utilisateur.

L'un des principaux points à améliorer concerne **l'affichage des images locales sur iOS**, notamment sur la page de connexion. En raison de contraintes techniques et du temps limité, nous n'avons pas réussi à intégrer correctement les images sur la version iOS de l'application. Cette amélioration permettrait d'uniformiser l'interface entre les deux plateformes et d'offrir une meilleure expérience visuelle aux utilisateurs.



Figure 14 : Page de connexion Android

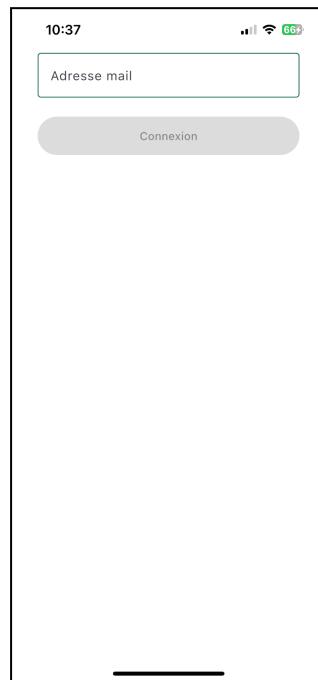


Figure 15 : Page de connexion iOS

Par ailleurs, suite aux retours des demandeurs, plusieurs ajustements UX ont été identifiés pour améliorer la navigation de l'application :

- **Refonte de la page d'accueil** : Actuellement, la liste des bornes récupérée sur la page d'accueil permet de signaler un dysfonctionnement. Une meilleure approche consisterait à proposer une réservation rapide en fonction des bornes disponibles, avec les champs de date et d'heure pré-remplis à la date actuelle, simplifiant ainsi le processus pour l'utilisateur.
- **Ajout d'un bouton de réservation sur la page d'accueil** : Pour permettre aux utilisateurs de créer une réservation directement sans avoir à naviguer dans d'autres menus, un bouton de nouvelle réservation serait ajouté à l'écran principal.
- **Déplacement du signalement des bornes dans un onglet dédié** : Actuellement intégré sur la page d'accueil, cette fonctionnalité pourrait être déplacée vers un onglet spécifique. Il faudrait aussi permettre aux utilisateurs de signaler toutes les bornes, y compris celles déjà occupées.

Malheureusement, nous n'avons pas eu le temps d'intégrer ces améliorations, ayant dû prioriser d'autres aspects essentiels du projet. Celles-ci n'étaient pas indispensables pour avoir une application fonctionnelle et les demandeurs ont été clairs sur ce point là, nous encourageant à d'abord nous préoccuper de l'essentiel : rendre une application fonctionnelle sur Android et iOS.

VIII. Expérience tirée

A. Valorisation du travail

L'un des aspects les plus satisfaisants de ce projet est que **toutes les fonctionnalités principales ont été implémentées dans les délais impartis**. Grâce à une bonne organisation et une gestion efficace du développement, nous avons su respecter le planning tout en assurant la qualité des livrables. Malgré les défis techniques rencontrés, nous avons su nous adapter et prioriser les éléments essentiels, garantissant ainsi un produit final fonctionnel et conforme aux attentes initiales.

À l'origine, ce projet avait avant tout une dimension conceptuelle, servant de point de **collaboration entre Polytech Montpellier et CA-TS Montpellier**. Bien que l'idée d'une application mobile pour la gestion des emplacements de recharge sur le parking de l'entreprise ait déjà été envisagée par CA-TS, il ne s'agissait initialement que d'une réflexion à court ou moyen terme, sans intention immédiate de mise en production. Pour les demandeurs, l'objectif principal de ce projet était de réaliser une étude d'opportunité sur le développement d'une application mobile Android et iOS en Kotlin Multiplatform (KMP) à travers un cas concret, celui de la gestion des places de parking. Cette étude devait permettre de valider si KMP était une solution pertinente pour développer rapidement ce type d'application.

Le bilan de cette expérimentation s'est avéré très positif pour CA-TS. En effet, nous avons pu monter en compétences sur Kotlin Multiplatform, tout en livrant une application fonctionnelle en fin de projet. Cette réussite a confirmé le potentiel de KMP pour ce type de

développement et a ouvert la réflexion sur son intégration dans les futurs projets mobiles de l'entreprise.

Suite à la présentation de notre travail, les responsables du projet ont commencé à considérer la possibilité de s'en inspirer pour leur propre développement. Bien que leur intention initiale ait été de concevoir une solution indépendante, notre application leur a permis d'explorer une alternative et d'évaluer dans quelle mesure notre prototype pourrait être réutilisé et adapté pour un usage interne. Cette adaptation nécessiterait des ajustements en matière de robustesse, de sécurité et d'intégration à leur charte graphique. Cette réflexion reste ouverte et dépendra des besoins et contraintes spécifiques de l'entreprise.

B. Compétences techniques et soft skills développées

L'un des aspects clés de ce projet a été la **montée en compétence sur Kotlin Multiplatform (KMP)**, une technologie de plus en plus adoptée par les grandes entreprises pour le développement d'applications multiplateformes. Bien que très intéressante à maîtriser, cette technologie s'est avérée complexe à prendre en main, notamment pour notre équipe qui n'avait jamais travaillé sur du développement mobile Android auparavant. La principale difficulté ne résidait pas tant dans le langage lui-même, mais plutôt dans l'organisation du projet entre ses différentes parties (shared, Android, iOS) et la gestion des dépendances compatibles avec chaque plateforme. À titre de comparaison, un projet similaire en React Native, technologie que nous avions déjà utilisée par le passé et qui permet aussi une approche multiplateforme, nous semblait beaucoup plus accessible, et semblait tout à fait adapté pour notre projet à échelle universitaire. La gestion du code et l'intégration des différentes fonctionnalités y étaient plus fluides car nous connaissions déjà la technologie, notamment par des projets de développement web utilisant React, alors que KMP exige une structuration plus rigoureuse et impose des contraintes supplémentaires pour assurer une compatibilité optimale entre les plateformes.

Ce projet nous a également permis de développer **notre prise d'initiative**, dont nous avons dû faire preuve tout au long du projet. Par exemple, face aux difficultés techniques rencontrées avec l'intégration d'une carte interactive, nous avons proposé une solution alternative en affichant les bornes sous forme de liste, garantissant ainsi une expérience utilisateur fluide malgré la contrainte technique. Nous avons aussi suggéré l'implémentation d'une page de statistiques pour l'administrateur de l'application, permettant d'analyser l'utilisation des bornes de recharge. Bien que cette fonctionnalité n'ait pas pu être intégrée par manque de temps, cette démarche a démontré notre capacité à proposer des améliorations pertinentes pour enrichir l'application.

Enfin, nous avons renforcé nos compétences en **communication et en collaboration**, essentielles dans un projet impliquant plusieurs parties prenantes. Les réunions hebdomadaires avec les demandeurs nous ont permis d'améliorer notre capacité à synthétiser et à transmettre des informations, nous avons appris à exposer clairement nos

avancées, nos difficultés et nos propositions d'amélioration. De plus, la rédaction des minutes de réunion et le dépôt Git dédié aux documents du projet ont facilité le suivi par toutes les parties prenantes, garantissant une transparence et une traçabilité essentielles au bon déroulement du projet.

IX. Annexes

Annexe 1 : Maquettes Utilisateur

The wireframes illustrate the following user interface components and features:

- Row 1:**
 - connexion_sans_mdp**: Login screen without password.
 - connexion_avec_mdp**: Login screen with password.
 - Accueil_user_v2**: Home screen showing a map of parking bays and their status (Available, Occupied, Reserved).
 - Profil_user_menu_close**: Profile screen with a menu icon.
- Row 2:**
 - réservoir_born...**: Screen for reserving a charging station.
 - réservoir_borne**: Screen for reserving a charging station.
 - signaler_borne**: Screen for reporting a charging station.
 - ajouter_un_v...**: Screen for adding a vehicle.
- Row 3:**
 - réservoir_born...**: Screen for reserving a charging station.
 - réservoir_borne**: Screen for reserving a charging station.
 - créer_mdp**: Screen for creating a password.
 - réservations_user_me...**: Screen for viewing reservations.

Annexe 2 : Maquettes Administrateur

page_accueil_admin

Zommer_la_carte

Menu_Accueil

Acceuil_Status_borne_oc...

Visualsion_differents_pole

Stat_visualisation_semaine

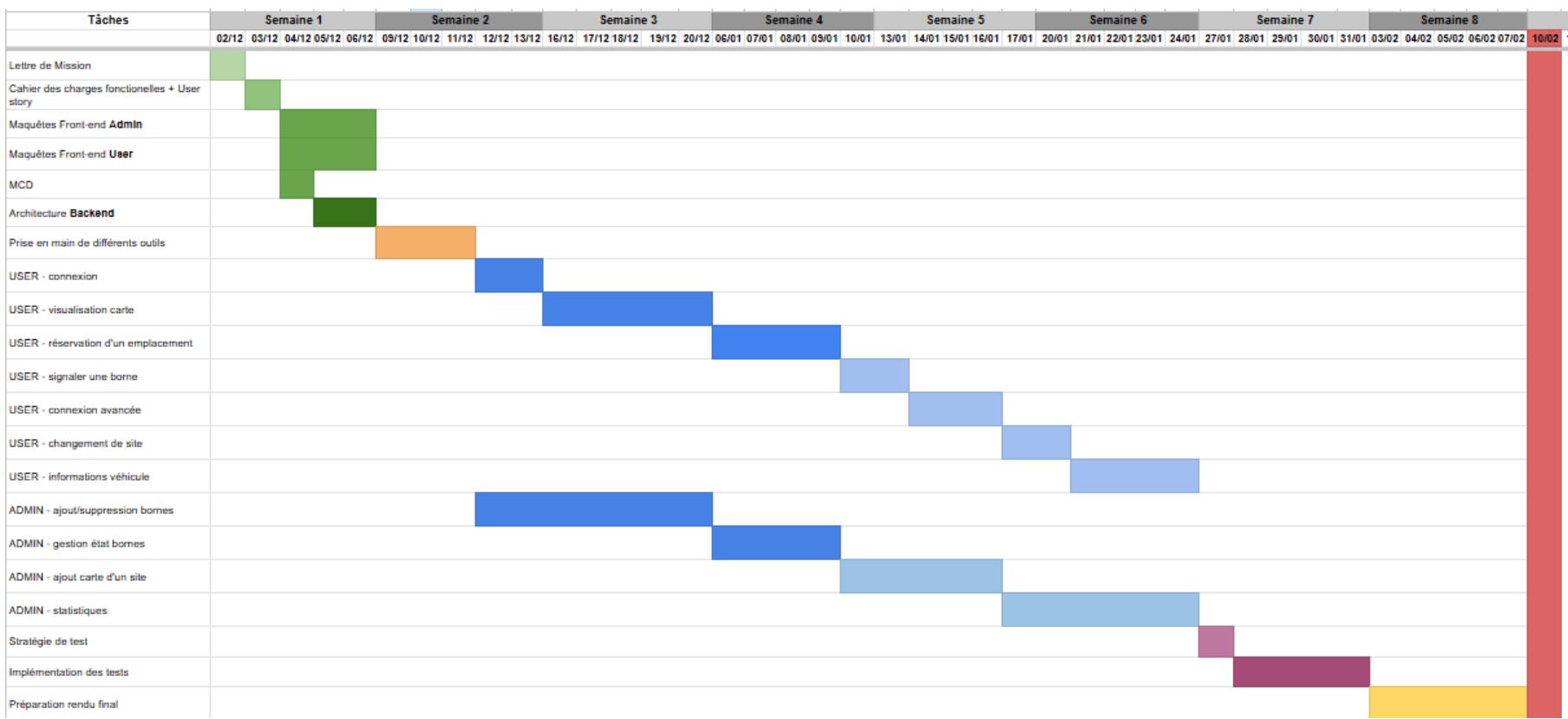
Stat_Semaine

Page_Signealement

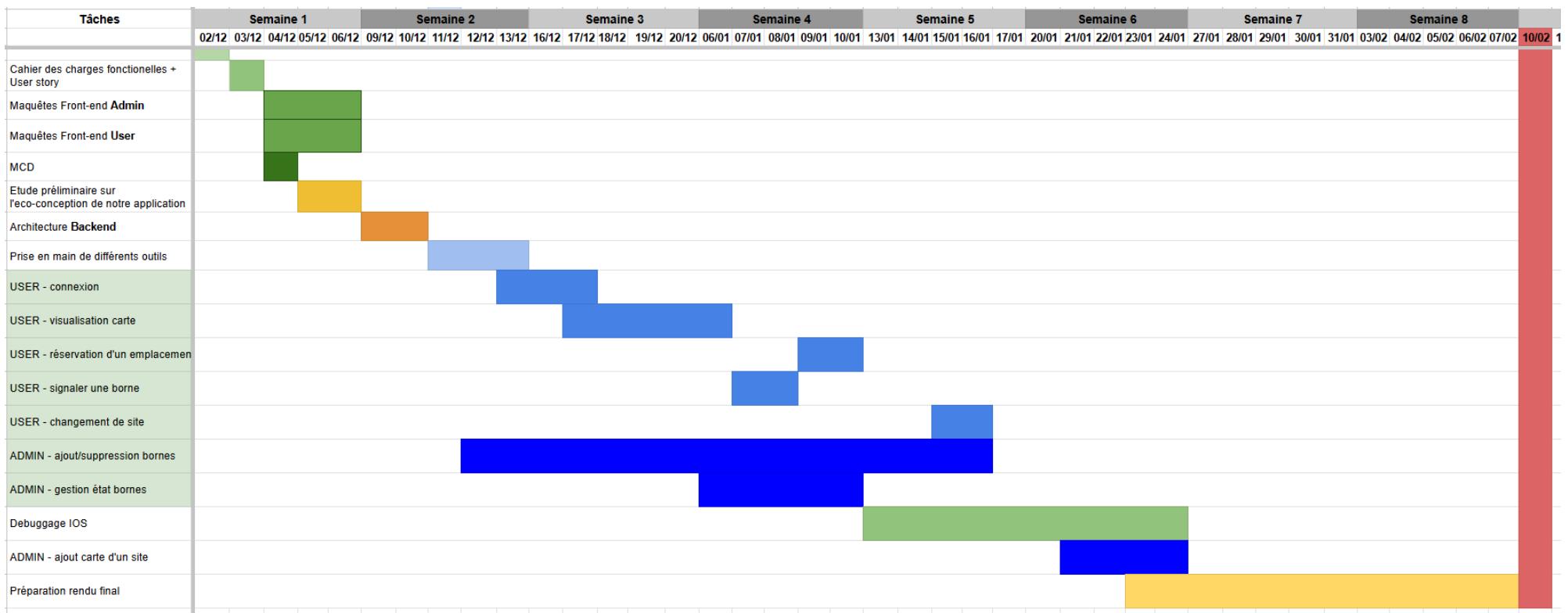
Signalement_status_Dispo

Ajout_carte

Annexe 3 : Diagramme de Gantt prévisionnel



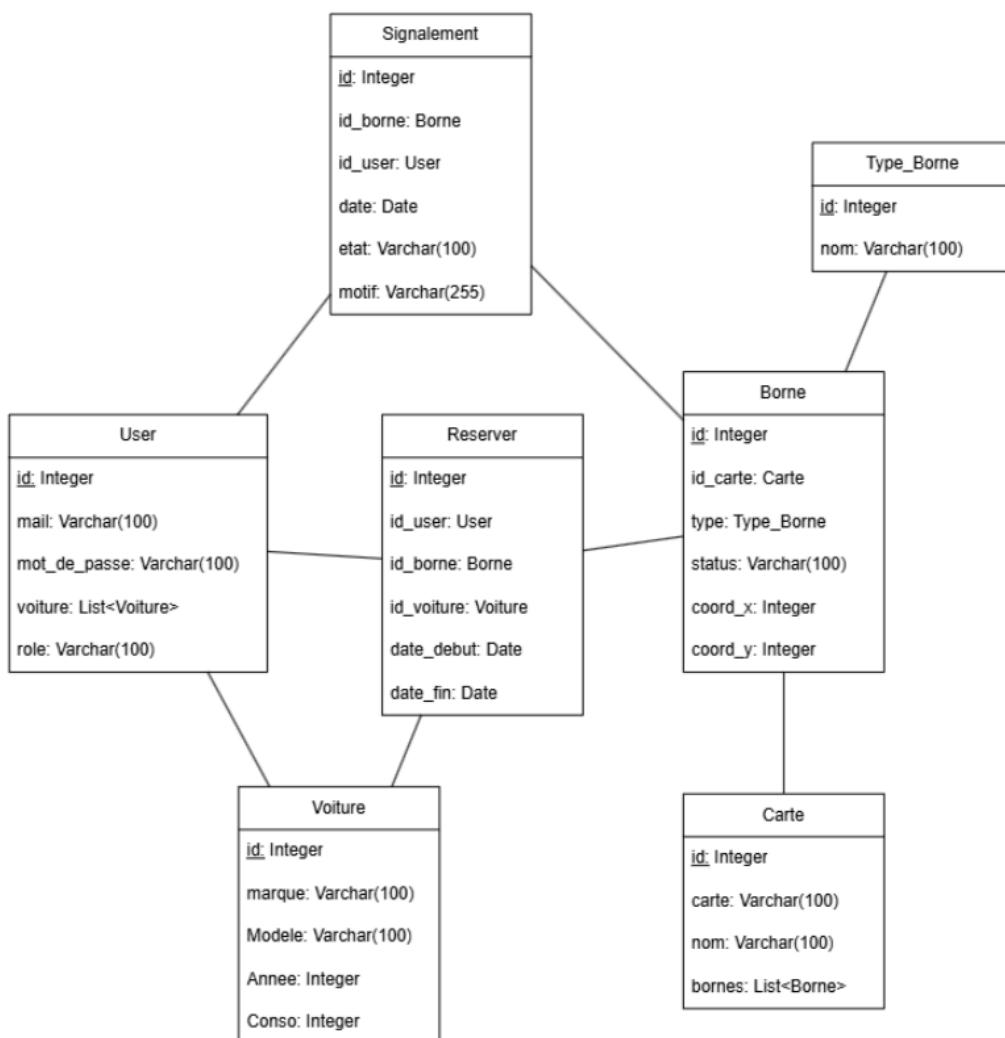
Annexe 4 : Diagramme de Gantt Réel



Annexe 5 : liste des tâches classées par ordre de priorité

Fonctionnalités à implémenter	Tâche	Priorité	Propriétaire	État	Date début prévisionnelle	Date fin prévisionnelle	Date début réelle	Date fin réelle
Connexion	Connexion	Élevée	Auriane	Terminé	12/12/2024	13/12/2024	13/12/2024	17/12/2024
Visualisation carte	Visualisation carte	Élevée	Auriane	Terminé	13/12/2024	20/12/2024	17/12/2024	06/01/2025
Réservation d'un emplacement	Réservation d'un emplacement	Élevée	Auriane	Terminé	06/01/2025	08/01/2025	07/01/2025	17/01/2025
Signalement borne	Signalement borne	Élevée	Auriane	Terminé	08/01/2025	09/01/2025	07/01/2025	08/01/2025
Connexion avancée	Connexion avancée	Moyenne	Auriane	Pas commencé	20/01/2025	21/01/2025	dd/mm/yyyy	dd/mm/yyyy
Informations véhicules	Informations véhicules	Faible	Auriane	Pas commencé	22/01/2025	24/01/2025	dd/mm/yyyy	dd/mm/yyyy
Changement de site	Changement de site	Moyenne	Margaux	Terminé	13/01/2025	17/01/2025	14/01/2025	16/01/2025
Politique de confidentialité & condition	Politique de confidentialité & condition	Faible	Lucas	Pas commencé	dd/mm/yyyy	dd/mm/yyyy	dd/mm/yyyy	dd/mm/yyyy
Notifications	Notifications	Faible	Lucas	Terminé	dd/mm/yyyy	dd/mm/yyyy	17/01/2025	21/01/2025
Ajout/Suppression de bornes	Ajout/Suppression de bornes	Élevée	Lucas	Terminé	12/12/2024	20/12/2024	15/12/2024	17/01/2025
Mise à jour de l'état des bornes	Mise à jour de l'état des bornes	Élevée	Lucas	Terminé	06/01/2025	08/01/2025	10/01/2025	15/01/2025
Ajout d'une nouvelle carte pour un site	Ajout d'une nouvelle carte pour un site	Moyenne	Margaux	En cours	09/01/2025	15/01/2025	20/01/2025	dd/mm/yyyy
Visualisation de statistiques	Visualisation de statistiques	Faible	Lucas	Pas commencé	16/01/2025	21/01/2025	dd/mm/yyyy	dd/mm/yyyy

Annexe 6 : Modèle Conceptuel de Donnée



Annexe 7: Lettre de Mission



Lettre de mission

CA-TS

Sommaire

Contexte du projet.....	2
1.1 Domaine métier, existant et acteurs du projet.....	2
1.2 Enjeux et contraintes.....	2
Objectifs.....	3
2.1 Problématique.....	3
2.2 Description détaillée de la mission.....	3
2.3 Livrables attendus.....	4
Critère de succès.....	4
Gestion de projet.....	5
4.1 Planification : étapes, échéances.....	5
4.2 Modalités et fréquences des réunions.....	5
4.3 Moyens humains et matériels mis à disposition.....	6
Signatures.....	7

Contexte du projet

Dans cette partie, nous allons présenter l'existant de notre projet de développement d'application mobile de suivi et de gestion des équipements de recharges (bornes) pour les moyens de transports électriques (voiture principalement, scooter, vélo ...) Nous traiterons également des enjeux et contraintes auxquels nous allons devoir nous adapter tout au long du projet.

1.1 Domaine métier, existant et acteurs du projet

L'équipe du Crédit Agricole, représentée par Damien Teissier et Thomas Deffains ont proposé un projet de fin d'étude que l'équipe composée de Auriane Poirier, Lucas Leroy et Margaux Thirion réalisera.

Le projet s'inscrit dans le domaine de la gestion des bornes de recharge pour les moyens de transport électrique pour le Crédit Agricole Montpellier, un secteur en plein essor face aux enjeux de mobilité durable et de transition énergétique. Il vise à répondre aux besoins du crédit agricole souhaitant optimiser l'utilisation et la gestion de leurs bornes de recharges électriques tout en facilitant leur gestion par et pour les employés.

1.2 Enjeux et contraintes

Enjeux du projet :

- **Optimisation de la gestion des ressources de transport électrique :**
Il s'agit de centraliser et simplifier la gestion des bornes de recharges de l'entreprise. L'application doit permettre une utilisation efficace des bornes tout en maximisant leur disponibilité.
- **Transition écologique et responsabilité sociétale :**
Ce projet s'inscrit dans les objectifs du Crédit Agricole de promouvoir une mobilité durable et de contribuer à la réduction de notre empreinte carbone. En facilitant l'utilisation de moyens de transport électriques et en sensibilisant les utilisateurs à des pratiques écoresponsables
- **Amélioration de l'expérience utilisateur :**
L'application doit offrir une interface intuitive et des fonctionnalités adaptées pour répondre aux attentes des employés. Une adoption rapide et un usage régulier par les utilisateurs finaux sont essentiels pour garantir le succès du projet.
- **Fiabilité et disponibilité des services :**
Il est crucial de fournir une solution performante pour assurer la continuité des services, notamment en termes de réservation, d'évaluation de la disponibilité des bornes, et de visualisation de l'état des bornes.

Contraintes du projet :

- **Contraintes techniques** : Gestion des bornes, incluant les localisations des bornes sur le parking, les historiques d'utilisation, et les états et utilisations des bornes. Développement en Kotlin Multiplatform (iOS et Android) pour garantir l'accessibilité à tous les employés.
- **Contraintes temporelles** : Respect des délais de livraison pour aligner le rendu de l'application avec les objectifs stratégiques de l'entreprise.

Objectifs

2.1 Problématique

Comment concevoir et développer une application mobile qui permettra au Crédit Agricole de Montpellier d'optimiser la gestion et l'utilisation des bornes de recharge pour véhicules électriques (voiture, scooter, vélo...), tout en répondant aux enjeux de transition énergétique, de mobilité durable, et d'efficacité opérationnelle.

2.2 Description détaillée de la mission

Afin de réaliser ce projet nous avons défini certaines fonctionnalités, autant du côté utilisateur que pour l'administrateur de l'application, nous allons les détailler séparément.

Utilisateur:

Fonctionnalités prioritaires :

- Connexion à l'application
- Visualisation de la carte
- Réservation d'un emplacement

Fonctionnalités supplémentaires:

- Signalement d'une borne dysfonctionnelle
- Personnalisation du mot de passe
- Ajout des informations de véhicules
- Changement de site

Administrateur:

Fonctionnalités prioritaires:

- Ajout/Suppression de bornes
- Mise à jour de l'état des bornes

Fonctionnalités supplémentaires:

- Ajout d'une nouvelle carte pour un site et ses bornes
- Visualisation de statistiques
- Ajout et gestion du type de borne

2.3 Livrables attendus

Les demandeurs souhaitent suivre l'avancement du projet de manière continue. Pour répondre à ce besoin, nous fournirons une version fonctionnelle de l'application à chaque implémentation d'une nouvelle fonctionnalité, permettant ainsi une évaluation régulière et interactive.

Les livrables suivants sont également attendus au cours du projet :

- **Diagramme de Gantt** : pour une visualisation claire et précise du planning du projet, incluant les principales étapes et jalons.
- **User stories** : pour détailler les besoins fonctionnels des utilisateurs et les traduire en spécifications techniques.
- **Maquettes du frontend** : pour proposer une représentation visuelle de l'interface utilisateur, validée avant le développement.

Enfin, le **code source** complet de l'application, incluant les parties frontend et backend, sera régulièrement mis à disposition sur GitHub tout au long du projet. Cela permettra un suivi transparent des développements et une gestion collaborative efficace.

Critère de succès

Le critère de réussite du projet est la satisfaction du demandeur. Avec le temps fixé à 9 semaines ouvrés qui nous a été donné pour ce projet, nous souhaitons qu'à la fin de cette période ce dernier soit satisfait du travail que nous aurons fourni. La satisfaction pourra être mesurée avec la réalisation effective des fonctionnalités prioritaires suivantes :

Utilisateur:

Fonctionnalités prioritaires :

- Connexion à l'application
- Visualisation de la carte
- Réservation d'un emplacement

Administrateur:

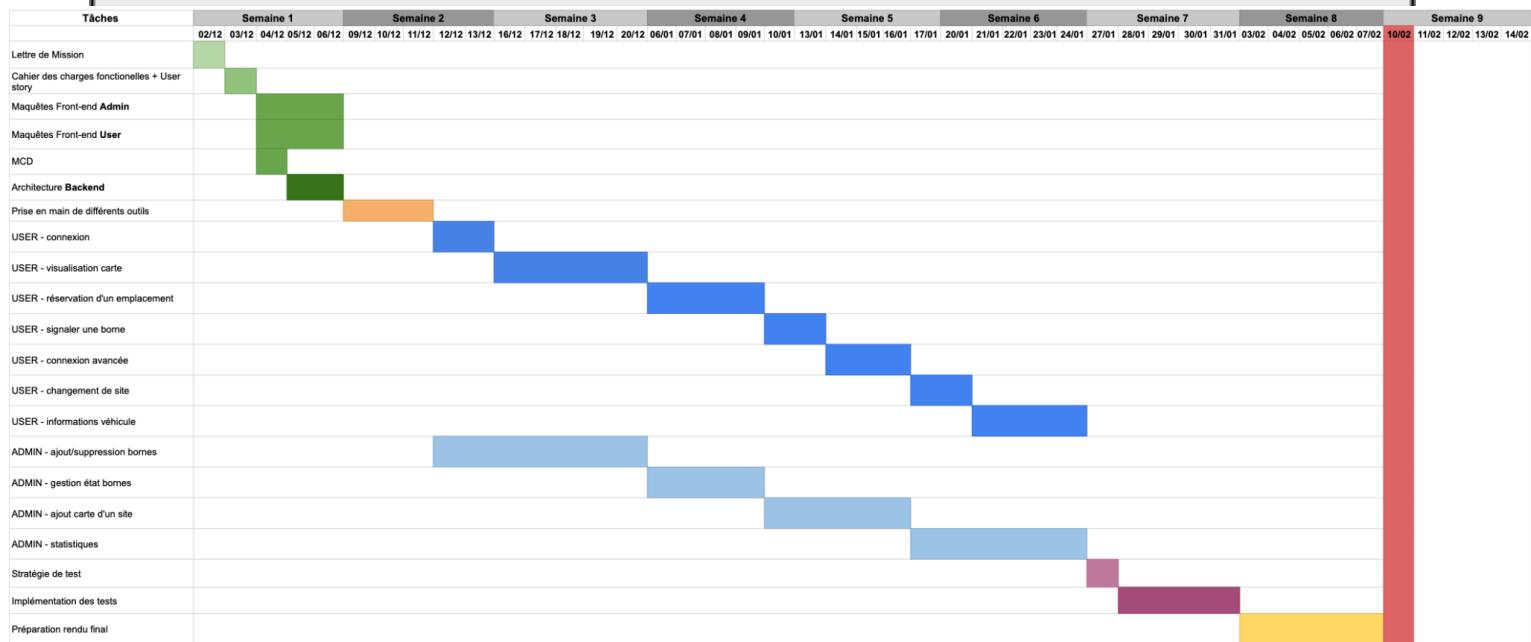
Fonctionnalités prioritaires:

- Ajout/Suppression de bornes
- Mise à jour de l'état des bornes

Des fonctionnalités secondaires pourraient être implémentées si le temps nous le permet.

Gestion de projet

4.1 Planification : étapes, échéances



Nous avons élaboré un diagramme de Gantt prévisionnel servant de guide pour la gestion et l'exécution du projet. Bien que des retards puissent survenir, nous nous engageons à respecter les délais définis dans la mesure du possible. En cas de besoin, nous solliciterons l'expertise de l'équipe du Crédit Agricole pour surmonter les éventuels obstacles.

Les tuteurs entreprises s'engagent à apporter leur soutien aux étudiants, notamment en contribuant à fournir des conseils ou des solutions pratiques pour permettre une progression optimale.

Cette collaboration étroite vise à garantir le bon déroulement du projet et à renforcer la cohésion entre les étudiants et les partenaires du Crédit Agricole.

4.2 Modalités et fréquences des réunions

Nous nous engageons à suivre des horaires de travail type bureau, du lundi au vendredi, à l'exception des deux semaines de vacances scolaires prévues, qui auront lieu du **21 décembre 2024 au 6 janvier 2025**.

Conformément aux discussions avec les parties prenantes, les étudiants travailleront à Polytech Montpellier dans la salle projet dédiée. Ils pourront exceptionnellement travailler en distanciel depuis chez eux après en avoir informé leurs demandeurs et leur tutrice école.

Des réunions hebdomadaires en distanciel seront organisées chaque vendredi, de **14 à 15 heures**, entre les trois étudiants et les tuteurs d'entreprise, Damien Teissier et Thomas Deffains. Une réunion exceptionnelle en présentiel est prévue le **17 décembre 2024** avec la participation des étudiants et des tuteurs d'entreprise dans les locaux de Polytech Montpellier.

Objectifs des réunions :

- Partager les avancées des travaux à l'aide de Minutes.
- Identifier et discuter les difficultés rencontrées.
- Définir les actions nécessaires pour progresser efficacement dans le projet.

Ces modalités visent à assurer un suivi régulier du projet pour les étudiants vis à vis l'équipe du Crédit Agricole et une collaboration harmonieuse entre Auriane Poirier, Margaux Thirion, Lucas Leroy et l'équipe du Crédit Agricole.

4.3 Moyens humains et matériels mis à disposition

Les ressources humaines mobilisées pour ce projet sont constituées de notre équipe : Auriane Poirier, Margaux Thirion et Lucas Leroy, qui s'engagent à collaborer de manière efficace et proactive avec l'équipe du Crédit Agricole.

Aucun matériel spécifique n'est mis à disposition pour ce projet, à l'exception d'une salle au sein de Polytech Montpellier, qui sera utilisée pour la réalisation des travaux. Le projet se déroulera sur la période allant du **2 décembre 2024 au 12 février 2025**.

Cette organisation vise à garantir une communication fluide entre les différentes parties et à assurer le bon déroulement des travaux dans les délais impartis.

Annexe 8: extrait du fichier récapitulatif des routes API

3. Borne

• Créer une borne

Route: **POST** <https://back-cats.onrender.com/borne>

Body:

```
{  
    "numero":14,  
    "typeBorne": {  
        "_id": "675c2adb6fc93512a0050c43"  
    },  
    "carte": {  
        "_id": "6763ed3c4545c40e2a6c7e80"  
    }  
}
```

Réponse:

```
{  
    "_id": "676004aac23444334a63caa3",  
    "status": "Fonctionnelle",  
    "numero": 14,  
    "coord_x": 100,  
    "coord_y": 200,  
    "typeBorne": {  
        "nom": "voiture",  
        "_id": "675c2adb6fc93512a0050c43"  
    },  
    "carte": {  
        "_id": "67613acdf18b073e149a2afe"  
    }  
}
```

Status possible: Fonctionnelle, HS et Signalée

X. Résumés

A. En français

Pour le Crédit Agricole Montpellier, nous avons développé une application mobile multiplateforme en Kotlin afin d'optimiser la gestion de leurs bornes de recharge électrique. Ce projet s'inscrit dans une démarche de mobilité durable et de transition énergétique. L'application, compatible iOS et Android, offre une interface intuitive et des fonctionnalités robustes pour simplifier l'utilisation des bornes et maximiser leur disponibilité. Elle permet la gestion centralisée des bornes, incluant la réservation et le signalement de dysfonctionnements pour les utilisateurs, ainsi que des outils de gestion pour les administrateurs. Ce développement contribue à améliorer l'expérience utilisateur et soutient les objectifs de développement durable de l'entreprise, en lien avec le CATS de Montpellier.

B. En anglais

For Crédit Agricole Montpellier, we developed a cross-platform mobile application in Kotlin to optimize the management of their electric charging stations. This project is part of a sustainable mobility and energy transition approach. The application, compatible with iOS and Android, offers an intuitive interface and robust functionalities to simplify the use of charging stations and maximize their availability. It enables centralized management of the kiosks, including booking and reporting of malfunctions for users, as well as management tools for administrators. This development contributes to improving the user experience and supports the company's sustainable development objectives, in conjunction with CATS Montpellier.

XI. Keywords

Electric mobility, charging station management, Kotlin Multiplatform, IOS, Android, CA-TS de Montpellier, mobile application