ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

PROJET DE STAGE

Année 2024-2025

Conception et réalisation d'une application web et mobile de géolocalisation des bus.

Ingénierie Informatique et Réseaux

Réalisé par :

KOUBRI Hatim

Encadré par :

Pr. Nourane Laaffat

Remerciements

Je tiens à exprimer ma sincère reconnaissance à **Madame Nourane Laaffat**, mon encadrante académique à l'EMSI, pour son accompagnement, sa disponibilité et ses précieux conseils lors de la mise à jour et de l'adaptation de ce rapport. Son soutien et ses orientations m'ont permis de respecter les normes académiques de mon nouvel établissement et d'améliorer la présentation de ce travail.

Je souhaite également remercier **Pr. LACHGAR Mohamed**, mon ancien encadrant à l'École Normale Supérieure, pour son suivi, son expertise et ses conseils tout au long de la première réalisation de ce projet. Son encadrement a grandement contribué à la réussite du travail initial.

Je tiens aussi à remercier mon binôme, **Rayan Layhi**, avec qui j'ai eu le plaisir de collaborer lors de la première version du projet. Son sérieux, son esprit d'équipe et sa motivation ont été essentiels à la qualité du travail accompli.

Enfin, j'adresse mes remerciements les plus sincères à ma famille et à mes proches pour leur soutien indéfectible, leur patience et leurs encouragements constants, qui m'ont accompagné tout au long de ce parcours.

Résumé

Ce projet de fin d'études a pour objectif la mise en place d'un système intelligent, interactif et géolocalisé en temps réel des bus dans le cadre d'une plateforme globale de gestion des lignes de bus, des arrêts et des transports sur une ligne donnée, suscitant un besoin parfaitement identifié d'optimisation du réseau de transport urbain et d'information des usagers au cœur des systèmes de transport intelligents (ITS) tout en étant une solution numérique modulaire permettant une gestion efficace et intelligente des données d'un réseau de bus et de la géolocalisation dynamique des véhicules.

L'architecture technologique se décline ainsi : un front-end en React.js avec Tailwind CSS, un back-end en PHP 5.6 connecté à une base de données MySQL et exécuté sous XAMPP, une application mobile Android en JAVA, les API REST, testées directement dans Postman et intégrées par Axios. Cette architecture s'inscrit dans une méthodologie agile de type Scrum supportée par la modélisation UML riches des diagrammes de cas d'utilisation, de séquence, de classes et d'activités qui structurent le besoin, les interactions prévues et la planification du système avant sa mise en œuvre.

Les fonctions principales mises en œuvre sont variées. Le système permet de gérer les lignes, les arrêts, les parcours et les véhicules. Il offre aussi la planification des trajets. L'affichage des bus se fait en temps réel sur une carte interactive. Les usagers peuvent ajouter des favoris pour accéder rapidement à leurs trajets habituels. Un outil de gestion performant est disponible pour les administrateurs. La synchronisation entre les modules web, mobile et serveur est fluide. Cela garantit une expérience rapide, efficace et confortable pour tous les utilisateurs.

Ce projet a pour ojectif d'apporter une contribution concrète au domaine des systèmes de transport intelligents à travers la proposition d'une solution numérique intégrée et interconnectée. L'originalité du projet réside dans sa démonstration que des accessibilités différentes de disponibilité peuvent permettre une réponse efficace à un besoin de gestion, en particulier pour un service de mobilité urbaine, grâce à une parfaite optimisation et information en temps réel. Les résultats obtenus ouvrent la voie à une amélioration continue, aussi bien sur le plan technique avec l'optimisation des performances et l'enrichissement fonctionnel que sur le plan ergonomique, à travers une interface utilisateur plus intuitive. et personnalisable,sur un socle UML à la structure claire et solide pour accompagner les évolutions nécessaires.

Mots-clés : géolocalisation en temps réel, transport intelligent, API REST, React.js, PHP, Android, MySQL, modélisation UML, Tailwind CSS, Axios, Postman.

Compte de mots : 12 546
Nombre de pages : 68
GitLab : https://gitlab.com/geolocalisation-des-bus

Abstract

The aim of this final year project is to implement an intelligent, interactive and real-time geolocation system for buses as part of a comprehensive platform for managing routes, stops and transport on a bus line, addressing a clearly identified need to optimise the urban transport network and provide user information at the heart of intelligent transport systems (ITS) systems, while also being a modular digital solution enabling efficient and intelligent management of bus network data and dynamic vehicle geolocation.

The technological architecture is as follows : a front-end in React.js with Tailwind CSS, a back-end in PHP 5.6 connected to a MySQL database and running under XAMPP, an Android mobile application in JAVA, REST APIs, tested directly in Postman and integrated by Axios. This architecture is part of an agile Scrum-type methodology supported by UML modelling rich in use case, sequence, class and activity diagrams that structure the requirements, planned interactions and system planning prior to implementation.

The main functions implemented include the management of lines, stops, routes and vehicles, route planning, real-time display of vehicles on the map, the possibility for users to add favourites, a powerful management tool for the administrator called admin, and simple synchronization between the web, mobile and server modules, in order to benefit from a responsive user experience (average response time of 2 to 3 seconds), guaranteed efficiency and ease of use.

This project aims to make a concrete contribution to the field of intelligent transport systems by proposing an integrated and interconnected digital solution. The originality of the project lies in its demonstration that affordability can provide an effective response to a management need, particularly for an urban mobility service, thanks to perfect optimization and real-time information. The results obtained will pave the way for continuous improvement, both in the technical field (performance optimization, functional enhancement) and in the ergonomic field (more intuitive and customizable user interface) and customisable, on a UML base with a clear and solid structure to support the necessary developments.

Keywords : real-time geolocation, intelligent transportation, REST API, React.js, PHP, Android, MySQL, UML modeling, Tailwind CSS, Axios, Postman.

Word count : 12,546
Page count : 68
GitLab : https://gitlab.com/geolocalisation-des-bus

Table des matières

Remerciements ii

Résumé iii

Abstract iv

Table des matières v

Liste des abréviations vii

Liste des tableaux viii

Table des figures ix

Introduction Générale 1

1 Contexte général du projet 2
- 1.1 Introduction . . . 2
- 1.2 Présentation du projet . . . 2
  - 1.2.1 Sujet du projet . . . 2
  - 1.2.2 Origine du projet . . . 2
  - 1.2.3 Public cible . . . 2
  - 1.2.4 Intérêt du projet . . . 3
- 1.3 Problématique et état de l'existant . . . 3
  - 1.3.1 Solutions existantes . . . 3
  - 1.3.2 Limites observées . . . 3
- 1.4 Objectifs du projet . . . 4
  - 1.4.1 Objectifs fonctionnels . . . 4
  - 1.4.2 Objectifs techniques . . . 4
  - 1.4.3 Choix technologiques . . . 5
  - 1.4.4 Contraintes . . . 5
- 1.5 Démarche de gestion de projet . . . 5
  - 1.5.1 Méthodologie adoptée : Scrum . . . 5
  - 1.5.2 Planification prévisionnelle . . . 6
- 1.6 Conclusion . . . 7

2 Étude préliminaire et fonctionnelle 8
- 2.1 Introduction . . . 8
- 2.2 Étude des besoins . . . 8
  - 2.2.1 Besoins fonctionnels . . . 8
  - 2.2.2 Besoins non fonctionnels . . . 9
- 2.3 Fonctionnalités principales du système . . . 9
  - 2.3.1 Fonctionnalités côté front-end . . . 9
  - 2.3.2 Fonctionnalités côté back-end . . . 10
  - 2.3.3 Fonctionnalités côté application mobile . . . 10
- 2.4 Identification des acteurs . . . 11
- 2.5 Diagramme BPMN . . . 11

2.5.1 Processus de gestion pour l'Admin . . . 11
2.5.2 Processus de localisation pour un dispositif de localisation . . . 12
2.6 Conclusion . . . 13

3 Analyse et conception **14**
3.1 Introduction . . . 14
3.2 Formalisme de modélisation (UML, etc.) . . . 14
3.2.1 Motivation et choix . . . 14
3.2.2 Diagrammes retenus . . . 14
3.3 Architecture logicielle . . . 15
3.3.1 Approche architecturale . . . 15
3.3.2 Architecture logicielle du système . . . 15
3.3.3 Architecture technique . . . 16
3.3.4 Discussion . . . 17
3.4 Diagrammes de conception . . . 17
3.4.1 Les cas d'utilisation . . . 17
3.4.2 Diagrammes de séquence . . . 20
3.4.3 Diagrammes d'activités . . . 22
3.4.4 Diagramme de classes . . . 24
3.5 Conclusion . . . 25

4 Technologies et réalisation **26**
4.1 Introduction . . . 26
4.2 Choix techniques et outils . . . 26
4.2.1 Langages, Frameworks, Base de données et Outils . . . 26
4.2.2 Outils de développement (IDE, gestion de versions...) . . . 27
4.3 Implémentation et interfaces . . . 27
4.3.1 Structure du code . . . 28
4.3.2 Interfaces utilisateur : . . . 29
4.3.3 Implémentation de l'application mobile . . . 47
4.4 Stratégie de tests . . . 48
4.5 Indicateurs de performance . . . 49
4.6 Conclusion . . . 49

Conclusion générale **50**

Annexes **52**
Annexe A — Tableau de validation des fonctionnalités . . . 52

Liste des abréviations

API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Deployment
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DAO	Data Access Object
DOM	Document Object Model
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
ITS	Intelligent Transportation Systems
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-View-Controller
MySQL	Structured Query Language
PHP	Hypertext Preprocessor
RGPD	Règlement Général sur la Protection des Données
SQL	Structured Query Language
SQLi	SQL Injection
UI	User Interface
UML	Unified Modeling Language
VS Code	Visual Studio Code

Liste des tableaux

1.1 Limites des solutions existantes et conséquences associées. 4
1.2 Découpage du projet en sprints. 7

2.1 Description des profils et des fonctionnalités associées. 11

A.1 Tableau de validation des principales fonctionnalités du système 53

Table des figures

1.1 Cycle Scrum et ses étapes clés. . . . 6
1.2 Visualisation des sprints sous forme de diagramme de Gantt. . . . 7

2.1 Processus de gestion des éléments administratifs. . . . 12
2.2 Processus de localisation d'un bus pour un dispositif de localisation. . . . 13

3.1 Schéma de l'architecture hybride : monolithe modulaire avec communication via API REST. 16
3.2 Diagramme des cas d'utilisation pour l'Usager. . . . 18
3.3 Diagramme des cas d'utilisation pour l'Administrateur. . . . 19
3.4 Diagramme des cas d'utilisation pour le Système de géolocalisation. . . . 20
3.5 Diagramme de séquence : Vérification des identifiants et création de session. . . . 21
3.6 Diagramme d'activité : Processus d'affichage de la localisation des bus. . . . 23
3.7 Diagramme de classes du système de géolocalisation des bus. . . . 24

4.1 Page d'inscription (interface usager). . . . 30
4.2 Page de connexion (interface usager). . . . 31
4.3 Page de mot de passe oublié (interface usager). . . . 31
4.4 Page de saisie du code de vérification (interface usager). . . . 32
4.5 Page de création du nouveau mot de passe (interface usager). . . . 33
4.6 Page d'accueil – utilisateur non connecté. . . . 34
4.7 Page d'accueil – utilisateur connecté. . . . 34
4.8 Page des arrêts avec filtrage et ajout aux favoris. . . . 35
4.9 Page des lignes avec filtrage, favoris et accès aux trajets. . . . 36
4.10 Page du trajet – affichage de la ligne et des arrêts. . . . 37
4.11 Page du trajet – localisation du bus en temps réel. . . . 37
4.12 Carte globale interactive affichant toutes les lignes de bus. . . . 38
4.13 Carte filtrée – affichage d'une seule ligne avec informations. . . . 39
4.14 Affichage des détails d'un arrêt sélectionné sur la carte. . . . 39
4.15 Page des favoris avec filtrage par type et options de gestion. . . . 40
4.16 Formulaire de planification de trajet. . . . 41
4.17 Détail de l'itinéraire proposé avec position du bus. . . . 41
4.18 Dashboard de l'administrateur avec indicateurs et statistiques. . . . 42
4.19 Page de gestion des trajets. . . . 43
4.20 Message de confirmation lors de l'ajout d'une ligne. . . . 43
4.21 Message de confirmation avant la suppression d'une ligne. . . . 44
4.22 Page de gestion des utilisateurs statistiques. . . . 45
4.23 Page de gestion des utilisateurs liste. . . . 45
4.24 Page Carte de l'administrateur : filtrage des lignes et localisation des bus. . . . 46
4.25 Page de rapports avec statistiques détaillées. . . . 47
4.26 Page de rapports avec liste détaillée des lignes. . . . 47

Introduction Générale

L'invention des bus au XIX^{e} siècle a marqué un tournant majeur dans l'histoire des transports urbains. De simples véhicules hippomobiles, ils ont évolué vers des modèles motorisés, électriques, voire autonomes, intégrant au fil du temps des innovations majeures en matière de performance, de confort et d'impact environnemental **??**. Ces progrès ont permis d'accompagner la croissance urbaine, de fluidifier la circulation dans les métropoles et de répondre aux besoins croissants en mobilité.

Cependant, malgré ces avancées n'ont toutefois pas permis d'éliminer certains défis majeurs, comme les retards dans les transports publics, en particulier au Maroc et dans d'autres pays en développement. Les causes sont multiples : embouteillages, pannes mécaniques, imprévus climatiques ou encore gestion inefficace des itinéraires. Ces retards impactent non seulement l'expérience des usagers, qui exigent davantage de fiabilité, mais compliquent aussi la gestion opérationnelle pour les régies de transport **??**. À Marrakech, où le réseau compte 45 lignes régulières **?** et une flotte de 500 bus (dont 10 électriques sur la ligne BHNS T01) **?**, ces défis sont particulièrement marqués. Le BHNS illustre parfaitement les déséquilibres entre l'offre et la demande en matière de mobilité. Sur les 52,7 km prévus **?**, seulement 8 km sont actuellement opérationnels. De plus, la ligne accueille environ 45 000 passagers par jour, alors que les prévisions tablaient sur 60 000 usagers **?**.

Dans un contexte où la mobilité urbaine est un enjeu central des villes, il devient nécessaire de proposer des solutions numériques innovantes, capables de garantir transparence de l'information, efficacité et réactivité. C'est dans ce cadre qu'intervient le présent projet, intitulé Conception et réalisation d'une application web et mobile de géolocalisation des bus. Ce projet vise à développer un système numérique permettant de visualiser en temps réel la position des bus, leurs itinéraires et leurs horaires. L'objectif est double : offrir aux usagers une meilleure visibilité sur le service, avec des informations précises et actualisées, et de l'autre, fournir aux opérateurs un outil de gestion centralisé des lignes et des arrêts, afin de mieux anticiper les incidents et optimiser le réseau.

La solution proposée repose sur une approche multidisciplinaire combinant technologies de géolocalisation, interfaces utilisateurs intuitives et architecture web scalable. Une méthodologie rigoureuse a été suivie, allant de l'analyse des besoins fonctionnels jusqu'à la mise en production du système, en passant par le développement d'applications réactives et l'évaluation de leur efficacité.

Au-delà de l'aspect technique, ce projet s'inscrit dans une réflexion plus large sur l'apport des technologies intelligentes à la transformation des services publics. Il démontre comment une solution conçue de manière rigoureuse et adaptée aux besoins réels peut améliorer la qualité de vie urbaine, fluidifier la mobilité quotidienne et renforcer la relation entre les citoyens et les systèmes de transport. Ce type d'initiative contribue également à restaurer la confiance des usagers dans les transports publics, en particulier les services de bus.

Le présent rapport est structuré en quatre chapitres. Après cette introduction, le chapitre 1 expose le contexte général du projet et en définit les objectifs. Le chapitre 2 est consacré à l'étude préliminaire et fonctionnelle, incluant l'analyse des besoins et la spécification des fonctionnalités attendues. Le chapitre 3 présente l'analyse détaillée et la conception du système, notamment l'architecture et les diagrammes UML. Enfin, le chapitre 4 décrit les technologies utilisées, les étapes de réalisation, la stratégie de tests mise en place ainsi que les résultats obtenus.

Chapitre 1

Contexte général du projet

1.1 Introduction

Cette section fournit un aperçu général du projet, en se concentrant sur les systèmes actuels de gestion et de localisation dans le domaine des transports publics. Il offre une étude des solutions disponibles, de leurs caractéristiques et des technologies qu'elles emploient. Cette recherche met en évidence diverses contraintes fréquentes, comme l'absence de données fiables en temps réel, la complexité des interfaces utilisateurs ou l'inadéquation avec les besoins locaux.

En se basant sur cette observation, le chapitre détaille les principes de base du projet, ses buts principaux, son audience visée et les améliorations tangibles qu'il aspire à instaurer, aussi bien pour les utilisateurs que pour les opérateurs de transport.

1.2 Présentation du projet

1.2.1 Sujet du projet

L'objectif du projet consiste à concevoir et à mettre en œuvre une application web et mobile dédiée à la localisation des bus de transport public en temps réel.cette technologie est fondamentale pour améliorer l'expérience des utilisateurs in situ, car elle leur offre un accès permanent à des informations instantanées, précises et constantes.le design ergonomique, contemporain et interactif favorisé par l'application permet aux usagers de suivre les bus en temps réel, de consulter les horaires de passage et d'effectuer ses trajets, le tout en apportant des précisions sur les lignes, les arrêts desservis et les parcours effectués.

1.2.2 Origine du projet

L'idée de ce projet découle d'un constat de terrain : dans de nombreuses villes, les retards fréquents des bus et le manque de prévisions claires perturbent considérablement l'organisation des déplacements quotidiens des citoyens. Ces problèmes engendrent une insatisfaction croissante chez les usagers et compliquent la gestion des réseaux de transport pour les opérateurs. Face à ces défis, le projet ambitionne d'exploiter les données GPS en temps réel pour offrir une solution innovante qui simplifie la planification des trajets et optimise la gestion des lignes de transport.

1.2.3 Public cible

Le projet s'adresse à plusieurs acteurs clés du réseau de transport, chacun ayant des besoins spécifiques en matière d'information et de gestion.

- **Les usagers du réseau de bus :**Ces derniers recherchent une meilleure visibilité sur les horaires et les positions des bus afin de planifier leurs trajets de manière plus efficace et sans stress ;
- **Les administrateurs du service de transport :** Les opérateurs et responsables du réseau ont besoin d'une interface centralisée leur permettant de visualiser les lignes, les arrêts, les trajets associés et la position des bus en temps réel, afin de faciliter la supervision du réseau et la gestion quotidienne du service.

1.2.4 Intérêt du projet

Le projet de géolocalisation des bus présente un intérêt majeur tant pour les usagers que pour les opérateurs de transport. Il vise à répondre aux problématiques actuelles en proposant des améliorations concrètes qui transforment l'expérience utilisateur et optimisent la gestion des réseaux de transport.

- **Améliorations proposées**
  - **Réactivité :** L'application garantit une mise à jour en temps réel de la position des bus, permettant aux usagers de suivre leurs trajets avec précision et sans attente ;
  - **Ergonomie :**Une interface fluide, intuitive et interactive a été développée afin d'offrir une expérience utilisateur simplifiée et engageante, accessible à tous, y compris aux personnes peu familières avec les outils numériques ;
  - **Personnalisation :** Les utilisateurs bénéficient de fonctionnalités personnalisées, telles que la possibilité d'ajouter des arrêts ou des lignes à leurs favoris, ainsi que de planifier leurs trajets en fonction de leur départ et de leur destination, afin d'accéder rapidement à leurs itinéraires habituels et de faciliter leurs déplacements quotidiens.
- **Impact :**
  - **Pour les usagers :** Le projet permet un gain de temps significatif en réduisant les périodes d'attente incertaine aux arrêts. Il contribue également à diminuer le stress lié à l'imprévu et aide à mieux organiser les déplacements quotidiens ;
  - **Pour les opérateurs :** L'application constitue un outil de supervision leur permettant de visualiser en temps réel les bus, les lignes, les arrêts et les trajets. Elle facilite ainsi la gestion opérationnelle du réseau et contribue à une meilleure organisation du service.

1.3 Problématique et état de l'existant

1.3.1 Solutions existantes

Au Maroc, comme dans de nombreuses régions du monde, plusieurs plateformes numériques proposent des solutions pour améliorer l'expérience des usagers des transports en commun. Parmi ces solutions, des applications telles que Moovit [1] ou Alsa [2] sont déjà utilisées dans certaines grandes villes comme Casablanca ou Rabat. Ces outils permettent généralement de consulter les itinéraires, les lignes desservies, ainsi que les horaires estimés de passage. Toutefois, ils ne fournissent pas une localisation en temps réel des bus basée sur les données GPS directement issues du terrain.

1.3.2 Limites observées

Plusieurs limitations sont observées, particulièrement dans le contexte marocain :

- **Dépendance aux données de tiers :** Ces applications reposent souvent sur des données fournies par des opérateurs extérieurs ou des API tierces, ce qui peut entraîner des informations imprécises, incomplètes ou obsolètes, notamment dans les villes moyennes ou les zones où l'infrastructure numérique est limitée ;
- **Complexité de déploiement local :** L'intégration de ces services dans des villes de taille moyenne ou rurales est souvent freinée par des coûts élevés, l'absence de données locales structurées ou un manque de collaboration avec les acteurs institutionnels.

Ces limitations mettent en évidence la nécessité de concevoir une solution locale, légère, et adaptée aux réalités du terrain marocain, capable de fournir une véritable géolocalisation des bus et une visualisation claire des trajets, arrêts et lignes, à travers une plateforme web accessible à tous.

Le tableau 1.1 résume les principales limites identifiées dans les solutions existantes de transport en commun au Maroc, ainsi que leurs impacts sur l'expérience des usagers et sur la gestion opérationnelle des réseaux.

1. https://moovitapp.com
2. https://www.alsa.ma/en/

Table 1.1 : Limites des solutions existantes et conséquences associées.

Limite identifiée	Conséquences
Données non fiables ou obsolètes	Affichage d'horaires erronés, mauvaise synchronisation des trajets, perte de confiance des usagers.
Interfaces complexes et peu intuitives	Navigation difficile, accessibilité limitée, mauvaise expérience utilisateur.
Coûts élevés de déploiement	Inaccessibilité pour les villes moyennes ou zones rurales disposant de budgets limités.
Manque d'infrastructures locales	Absence de données GPS fiables, couverture incomplète du réseau, impossibilité de proposer des services en temps réel.

Ces constats mettent en évidence l'écart entre les solutions disponibles sur le marché et les besoins spécifiques du contexte marocain. Dans un environnement où les infrastructures sont inégalement réparties et les moyens limités, il devient impératif de développer une application locale, flexible, légère et peu coûteuse. Celle-ci doit permettre une meilleure visualisation des trajets, des lignes et des arrêts, tout en assurant une accessibilité maximale aux différents types d'utilisateurs.

La conception d'un tel système de géolocalisation, adapté aux contraintes locales, repose sur trois piliers essentiels : la fiabilité des données fournies, la simplicité d'utilisation de l'interface, et la facilité de déploiement sans dépendance à des infrastructures lourdes. C'est précisément dans cette optique que s'inscrit le projet présenté dans ce rapport.

1.4 Objectifs du projet

1.4.1 Objectifs fonctionnels

Le projet vise à répondre aux besoins spécifiques des usagers et des opérateurs de transport en proposant une solution web complète, interactive et accessible. Les principaux objectifs fonctionnels sont les suivants :

- **Suivi des bus sur carte interactive** : Permettre aux utilisateurs de visualiser en temps réel la position des bus sur une carte dynamique intégrée à l'application ;
- **Consultation des horaires, lignes et arrêts** : Offrir une interface claire permettant aux usagers de consulter les horaires prévus, les arrêts desservis ainsi que les itinéraires associés à chaque ligne ;
- **Gestion des favoris** : Autoriser les usagers à ajouter des arrêts ou des lignes à une liste personnalisée pour un accès rapide à leurs trajets fréquents ;
- **Planification des trajets** : Permettre aux usagers de planifier leurs déplacements en sélectionnant un point de départ et une destination, afin d'obtenir l'itinéraire optimal et les correspondances disponibles ;
- **Espace administrateur** : Fournir un tableau de bord dédié aux administrateurs leur permettant de gérer les lignes, les arrêts, les bus et les trajets, de suivre les localisations en temps réel, ainsi que de consulter des statistiques détaillées sur l'activité du réseau avec la possibilité d'exporter le rapport.

1.4.2 Objectifs techniques

Sur le plan technique, le projet a pour ambition de mettre en œuvre une architecture logicielle robuste, modulaire et performante, capable de répondre aux exigences fonctionnelles identifiées. Les objectifs techniques principaux sont les suivants :

- Concevoir un système de géolocalisation des bus en temps réel avec une actualisation rapide et fiable des positions sur la carte ;
- Implémenter une communication efficace et sécurisée entre le front-end, le back-end et la base de données, en utilisant des API REST ;
- Assurer la synchronisation des données entre les différents modules (web, mobile, serveur) pour garantir la cohérence des informations affichées ;

- Développer une architecture évolutive permettant l'ajout futur de nouvelles fonctionnalités (notifications, etc.) ;
- Optimiser les performances de l'application afin de garantir une navigation fluide et réactive, même en cas de forte charge ;
- Mettre en œuvre des interfaces utilisateur ergonomiques et responsives, compatibles avec différents types de terminaux (ordinateurs, smartphones) ;
- Garantir la fiabilité et la stabilité du système à travers des tests manuels et des tests d'intégration réguliers.

1.4.3 Choix technologiques

Pour garantir le bon fonctionnement de l'application et assurer sa maintenabilité, plusieurs choix technologiques ont été retenus en fonction des besoins fonctionnels et des contraintes techniques :

- **Back-end :** Le langage PHP a été utilisé pour le développement du serveur, assurant une gestion fiable des requêtes et des échanges de données entre le client et la base de données ;
- **Front-end :** La bibliothèque JavaScript React.js a été adoptée pour développer une interface utilisateur moderne, dynamique et modulaire, facilitant la réutilisation des composants ;
- **Base de données :** Une base de données relationnelle MySQL a été choisie pour stocker les informations relatives aux lignes, arrêts, trajets, horaires et utilisateurs ;
- **API de cartographie :** L'API Google Maps a été intégrée afin de visualiser en temps réel la position des bus et les itinéraires sur une carte interactive, améliorant ainsi la lisibilité du système ;
- **Application mobile :** Une application Android a été développée avec le langage Java. Elle permet de capturer et de transmettre en temps réel la position GPS des bus vers le serveur.

1.4.4 Contraintes

Le projet a été développé en tenant compte de plusieurs contraintes qui ont influencé les choix techniques et la planification :

- **Temps :** Le développement devait être achevé en cinq mois, imposant une gestion rigoureuse du calendrier et une hiérarchisation des fonctionnalités à implémenter en priorité ;
- **Sécurité :** Le respect des principes de protection des données personnelles, en cohérence avec le RGPD, a été pris en considération afin de garantir la confidentialité des informations utilisateur collectées ;
- **Budget :** Le budget limité du projet a conduit à privilégier des solutions gratuites ou accessibles. L'API Google Maps a ainsi été utilisée en version gratuite d'essai (90 jours), permettant l'affichage des cartes et la géolocalisation sans surcoût initial.

Ces objectifs et contraintes définissent clairement le cadre du projet et servent de guide pour sa conception et son développement. Ils assurent que la solution finale soit à la fois fonctionnelle, techniquement solide et adaptée aux ressources disponibles.

1.5 Démarche de gestion de projet

1.5.1 Méthodologie adoptée : Scrum

Pour garantir une organisation efficace, une visibilité continue de l'avancement, et une meilleure adaptation aux besoins évolutifs du projet, la méthodologie agile Scrum a été adoptée. Scrum repose sur un cycle de développement itératif et incrémental, structuré en sprints, ce qui permet une planification par étapes, un suivi régulier de la progression, et une amélioration continue basée sur les retours.

Les éléments clés de cette méthode sont :

- **Sprints courts (2 à 6 semaines)** : Le travail est divisé en périodes fixes appelées sprints, au cours desquelles une série de tâches ou fonctionnalités précises est définie, développée, testée et intégrée. Cette organisation permet de mesurer l'avancement du projet étape par étape et de livrer des versions intermédiaires exploitables ;

- **Réunions de planification et démonstrations** : Chaque sprint débute par une réunion de planification (Sprint Planning) visant à fixer les objectifs du cycle. À la fin, une démonstration des fonctionnalités implémentées est organisée afin de présenter les évolutions et d'obtenir des retours constructifs de la part des parties prenantes (usagers, encadrants, etc.) ;
- **Adaptation continue et rétrospective** : Les retours collectés permettent d'ajuster les priorités et de réorganiser le backlog produit pour les cycles suivants. Une rétrospective (Sprint Retrospective) permet aussi de discuter de l'organisation, d'identifier les points à améliorer et d'optimiser les pratiques d'équipe pour les prochaines itérations.

La figure 1.1 illustre le cycle Scrum dans sa globalité. Elle met en évidence les différentes phases du processus : planification, exécution du sprint, réunions quotidiennes (daily meetings), revue des livrables, et rétrospective. Ce modèle itératif a permis d'assurer une gestion flexible et progressive du projet, en livrant des versions fonctionnelles à chaque étape tout en restant à l'écoute des besoins réels.

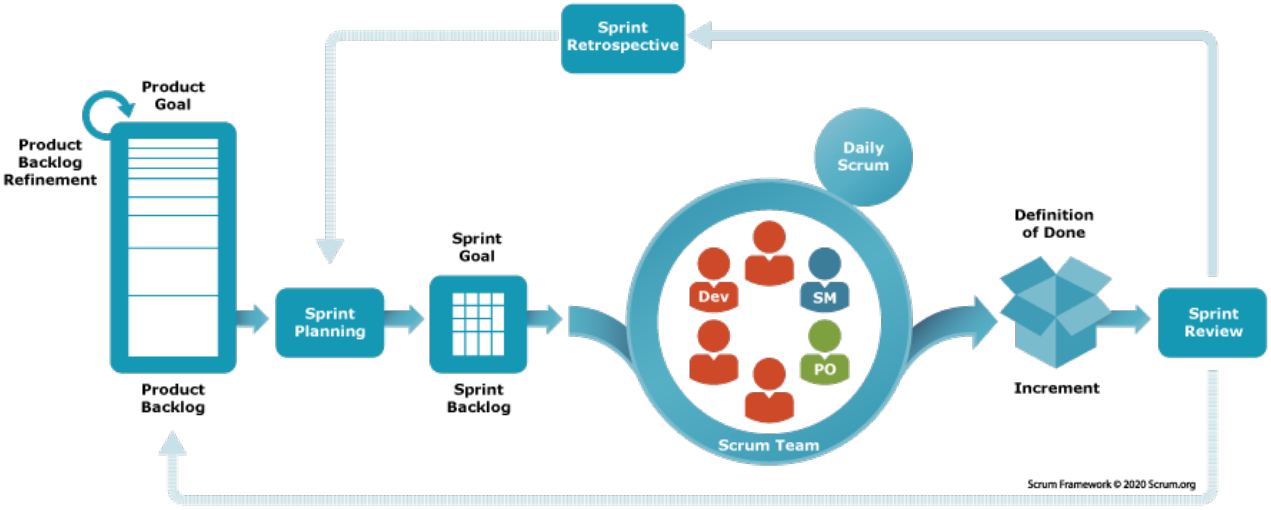


Figure 1.1 : Cycle Scrum et ses étapes clés.

1.5.2 Planification prévisionnelle

La planification du projet a été élaborée selon une approche agile, en s'appuyant sur un découpage en sprints successifs. Afin de structurer efficacement les différentes phases de développement, un diagramme de Gantt a été utilisé. Celui-ci permet de visualiser clairement l'enchaînement des tâches dans le temps et de suivre l'évolution du projet étape par étape.

Le tableau 1.2 présente les six sprints planifiés, en précisant pour chacun leur intitulé, leur date de démarrage et leur date de clôture.

Table 1.2 : Découpage du projet en sprints.

Sprint	Nom de la phase	Date de début	Date de fin
Sprint 1	Gestion des lignes et des arrêts	25/02/2025	12/03/2025
Sprint 2	Gestion des bus et enregistrement des positions GPS	12/03/2025	27/03/2025
Sprint 3	Mise en place des trajets et horaires associés	27/03/2025	11/04/2025
Sprint 4	Authentification et gestion des utilisateurs	11/04/2025	28/04/2025
Sprint 5	planification des trajets et favoris	30/04/2025	15/05/2025
Sprint 6	Statistiques et finalisation	13/05/2025	28/05/2025

La figure 1.2 illustre cette planification de manière graphique sous forme de diagramme de Gantt, permettant une visualisation claire des périodes de développement prévues pour chaque module.

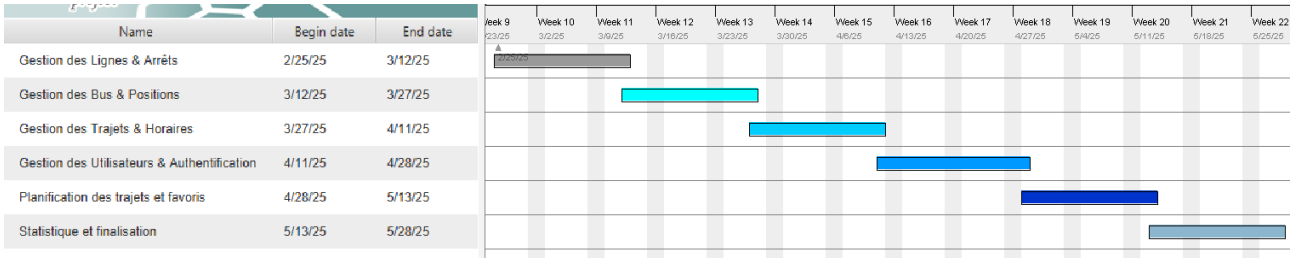


Figure 1.2 : Visualisation des sprints sous forme de diagramme de Gantt.

1.6 Conclusion

Au terme de ce premier chapitre, les éléments de base du projet ont été introduits, à savoir son périmètre général, les grands enjeux de la mobilité urbaine, ainsi que les problématiques actuelles telles que le respect des horaires, la fiabilité de l'information, et les limites des systèmes de transport en commun existants dans de nombreuses villes de taille moyenne, où les solutions numériques disponibles restent inadaptées.

Par ailleurs, l'analyse des outils existants a mis en évidence le faible nombre de plateformes offrant une visualisation en temps réel des bus. Leur déploiement est souvent freiné par des contraintes techniques, économiques, ou d'accessibilité.

C'est dans ce contexte qu'une solution locale, simple et efficace a été proposée, dédiée à l'affichage dynamique des lignes, des arrêts, des parcours et des positions, sur une carte interactive.

Le chapitre suivant présentera l'étude fonctionnelle, l'analyse des besoins utilisateurs, les fonctionnalités prévues, ainsi que les contraintes et spécifications techniques à mettre en œuvre.

Chapitre 2

Étude préliminaire et fonctionnelle

2.1 Introduction

Grâce aux éléments présentés dans le chapitre précédent, le cadre général du projet a été délimité, et plusieurs faiblesses des solutions existantes concernant la localisation des bus au Maroc ont été mises en évidence, de même que les axes d'amélioration à privilégier pour la nouvelle application : fiabilité des données, simplicité d'usage et facilité d'accès.

Ce chapitre sera consacré à un approfondissement de la réflexion, incluant l'identification des besoins concrets découlant de ces constats. Les attentes des utilisateurs finaux et des transporteurs seront précisées, ainsi que les contraintes techniques à prendre en compte et les principales exigences fonctionnelles qui orienteront la conception et le développement de l'application.

2.2 Étude des besoins

Dans cette section, deux grandes catégories de besoins sont distinguées : les besoins fonctionnels, qui décrivent ce que le système doit accomplir, et les besoins non fonctionnels, qui portent sur les aspects techniques tels que la performance, la sécurité et la fiabilité. Cette analyse permet de définir précisément les attentes du projet et d'orienter les choix techniques ultérieurs.

2.2.1 Besoins fonctionnels

Les besoins fonctionnels définissent les principales fonctionnalités que le système doit offrir pour répondre aux attentes des usagers et des administrateurs.

- **Création et connexion :** Permettre aux utilisateurs de créer un compte et de se connecter à l'application ;
- **Suivi en temps réel des bus :** Affichage dynamique de la position des bus sur une carte interactive, à partir des coordonnées GPS transmises par les appareils mobiles ;
- **Consultation des lignes, arrêts et trajets :** Accès aux différentes lignes disponibles, aux arrêts associés, ainsi qu'aux trajets planifiés (horaires de départ et d'arrivée) ;
- **Ajout aux favoris :** Permettre aux utilisateurs d'enregistrer leurs arrêts ou lignes préférés pour un accès plus rapide ;
- **Tableau de bord administrateur :** Interface permettant la gestion des lignes, des trajets, des bus et des arrêts, avec la possibilité d'ajouter ou de supprimer des éléments. Le tableau de bord permet également de superviser en temps réel la position des bus et des trajets, et d'accéder à des rapports détaillés sur l'efficacité des trajets, les retards et les statistiques de fréquentation des lignes et des arrêts ;
- **Système de recommandation :** Proposition automatique des lignes et trajets les plus pertinents en fonction de l'arrêt de départ et de la destination sélectionnée.

Pour illustrer ces besoins, voici quelques scénarios d'usage concrets :

- **Création de compte et personnalisation :** Un nouvel utilisateur crée un compte, explore les lignes disponibles, et ajoute ses arrêts favoris pour y accéder rapidement lors de ses prochaines connexions ;

- **Suivi d'un bus en temps réel :** Un usager consulte la carte interactive afin de localiser son bus et planifie son départ en fonction de la progression affichée ;
- **Gestion du réseau par un administrateur :** Un opérateur de transport ajoute ou supprime un trajet à une ligne existante et consulte le rapport associé ;
- **Système de recommandation :** Un utilisateur sélectionne un arrêt de départ, choisit un arrêt d'arrivée souhaité, et consulte les lignes et trajets recommandés par le système pour optimiser son itinéraire. Les trajets proposés s'affichent avec les horaires et la distance estimée jusqu'au dernier arrêt.

Ces cas d'utilisation permettent de mieux comprendre les interactions prévues entre les différents types d'utilisateurs et le système, et servent de base à la conception des interfaces.

2.2.2 Besoins non fonctionnels

Les besoins non fonctionnels regroupent l'ensemble des contraintes techniques et des exigences qualitatives auxquelles le système doit répondre. Ils influencent directement les choix d'architecture, les technologies employées et les modalités de développement. Ces besoins visent à garantir la performance, la sécurité, la fiabilité et la compatibilité de l'application.

- **Performance** :
  - Le temps de réponse pour le chargement des données critiques (comme la position des bus ou les horaires) ne doit pas excéder 30 secondes, même en cas de forte affluence ;
  - Le système doit être capable de supporter plusieurs centaines d'utilisateurs simultanés sans dégradation significative des performances.
- **Sécurité** :
  - Les données sensibles (telles que les mots de passe ou les informations personnelles des utilisateurs) doivent être stockées et transmises de manière sécurisée, en conformité avec les recommandations du RGPD.
- **Convivialité** :
  - L'application doit offrir une expérience utilisateur intuitive avec une interface soignée et des instructions claires ;
  - La navigation doit être fluide et facile, permettant à l'utilisateur de s'orienter rapidement dans l'application.
- **Compatibilité** :
  - L'interface utilisateur doit être accessible depuis l'ensemble des navigateurs modernes (Chrome, Firefox, Edge, Safari) sans nécessiter d'installation supplémentaire.

Ainsi, ces besoins non fonctionnels garantissent que l'application ne se limite pas à répondre aux attentes fonctionnelles, mais qu'elle offre également un haut niveau de qualité, de fiabilité et d'évolutivité pour une utilisation durable.

2.3 Fonctionnalités principales du système

Après l'identification des besoins fonctionnels, cette section détaille la manière dont ces besoins ont été concrètement implémentés dans le système, en présentant les principales fonctionnalités développées côté front-end, back-end et application mobile. L'ensemble de ces fonctionnalités vise à offrir une solution complète et cohérente aux usagers comme aux administrateurs.

2.3.1 Fonctionnalités côté front-end

- **Navigation fluide** : barre de navigation dynamique permettant un accès rapide aux différentes pages de l'application.
- **Formulaires interactifs** :
  - Formulaire de connexion et d'inscription utilisateur ;

  - Formulaires de gestion des entités pour l'administrateur (ajout des lignes, arrêts, bus, trajets) ;
- **Affichage des données** :
  - Listes dynamiques des lignes, arrêts, trajets, bus ;
  - Carte interactive pour la visualisation en temps réel des bus et des lignes ;
  - Page des favoris personnalisée ;
  - Dashboard avec indicateurs clés pour l'administrateur ;
  - Page de rapports avec graphiques et tableaux exportables.
- **Système de recommandation** :
  - Interface de sélection d'un arrêt de départ et d'un arrêt d'arrivée ;
  - Affichage des lignes et trajets recommandés en fonction des critères sélectionnés, avec les horaires correspondants et la distance estimée.
- **Expérience utilisateur améliorée** :
  - Utilisation d'Axios pour les appels API ;
  - Gestion des états de chargement et des erreurs ;
  - Interface réactive et adaptée aux différents écrans.

2.3.2 Fonctionnalités côté back-end

- **Gestion des requêtes API** :
  - Web Services REST pour le chargement des lignes, arrêts, bus, trajets, positions, favoris ;
  - Requêtes GET et POST sécurisées ;
  - Réponses au format JSON pour une intégration fluide avec le front-end.
- **Traitement des données** :
  - Insertion et suppression des entités (bus, lignes, arrêts, trajets, utilisateurs) ;
  - Enregistrement des positions GPS envoyées depuis l'application mobile ;
  - Génération de statistiques pour les rapports ;
  - Gestion des favoris des utilisateurs ;
  - Traitement des requêtes du système de recommandation et génération des suggestions de trajets optimisés.
- **Sécurité** :
  - Gestion sécurisée de l'authentification des utilisateurs (hachage des mots de passe) ;
  - Protection contre les injections SQL via l'utilisation de requêtes préparées (PDO) ;
  - Contrôle des droits d'accès (différenciation usagers / administrateurs) ;
  - Validation des données côté serveur.

2.3.3 Fonctionnalités côté application mobile

- **Acquisition de la position GPS** :
  - Utilisation des API Google Location Services pour obtenir la position géographique (latitude, longitude) de l'appareil ;
  - Vérification et gestion des autorisations d'accès à la localisation.
- **Envoi des positions GPS** :
  - Envoi périodique des positions vers le serveur via des requêtes HTTP POST à l'API REST `/LoadPositions.php` ;
  - Chaque envoi contient l'identifiant du bus, les coordonnées GPS et un horodatage.
- **Interaction utilisateur** :
  - Sélection d'une position sur la carte et affectation à un bus via un menu déroulant ;
  - Consultation des positions enregistrées pour un bus sélectionné (affichage sur la carte : anciennes positions en rouge, dernière position en vert).

- **Gestion des erreurs et de la connectivité** :
  - Contrôle de la validité des coordonnées GPS ;
  - Vérification de la connectivité réseau avant l'envoi ;
  - Gestion automatique des tentatives de ré-envoi en cas d'échec.

Grâce à ces fonctionnalités clés, le système offre une solution complète de visualisation et de gestion du réseau de transport en temps réel, adaptée aux besoins des usagers comme des administrateurs.

2.4 Identification des acteurs

L'identification des acteurs constitue une étape clé pour modéliser les interactions entre les utilisateurs et le système. Elle permet de définir les rôles, les droits d'accès, ainsi que les fonctionnalités propres à chaque profil.

Le tableau 2.1 présente les principaux types d'acteurs identifiés dans ce projet et les actions qui leur sont associées.

Table 2.1 : Description des profils et des fonctionnalités associées.

Profil	**Fonctionnalités associées**
Usager (Utilisateur classique)	Création d'un compte personnel, authentification, consultation des lignes, arrêts, trajets et horaires, visualisation en temps réel de la position des bus sur une carte interactive, ajout et suppression de lignes/arrêts aux favoris, consulter une recommandation de trajet
Administrateur	Accès à une interface d'administration permettant la gestion des lignes, arrêts, trajets et bus (ajout, suppression), visualisation des positions GPS en temps réel, gestion des utilisateurs, et consultation des données du réseau à des fins de suivi et d'analyse.
Système de géolocalisation (dispositif mobile)	Envoi régulier des coordonnées GPS des bus via un module mobile Android, synchronisation avec la base de données pour mise à jour en temps réel, identification du bus associé à chaque appareil.

2.5 Diagramme BPMN

Le diagramme BPMN (Business Process Model and Notation) est une méthode visuelle permettant de modéliser et de décrire les processus d'affaires de manière claire et structurée. Il permet aux différents acteurs (administrateurs, utilisateurs, etc.) de comprendre le flux des processus, d'identifier les points de décision, et de visualiser l'interaction entre les différents acteurs et systèmes. Les diagrammes BPMN sont particulièrement utiles pour illustrer des workflows complexes, ce qui aide à la prise de décision et à l'optimisation des processus.

2.5.1 Processus de gestion pour l'Admin

Le premier diagramme BPMN illustre le processus de gestion des éléments pour un administrateur dans un système de gestion de transport. Ce processus commence par un choix d'élément, où l'administrateur doit décider quel type d'élément gérer (arrêts, trajets, bus, lignes). Ensuite, après avoir sélectionné l'élément, l'administrateur peut choisir de l'ajouter ou de le supprimer.

- **Choix d'action** : L'administrateur peut choisir d'ajouter ou de supprimer un élément. Si l'option est "Ajouter", il doit remplir un formulaire pour ajouter l'élément. Si l'option est "Supprimer", l'administrateur sélectionne l'élément à supprimer.

- **Suivi des opérations** : Chaque action est suivie par une vérification de son succès ou de son échec, avec une réponse appropriée : succès ou erreur. Ce processus garantit que les opérations administratives se déroulent sans faille et avec une gestion appropriée des erreurs.

La figure 2.1 illustre le flux de gestion des éléments administratifs pour un administrateur.

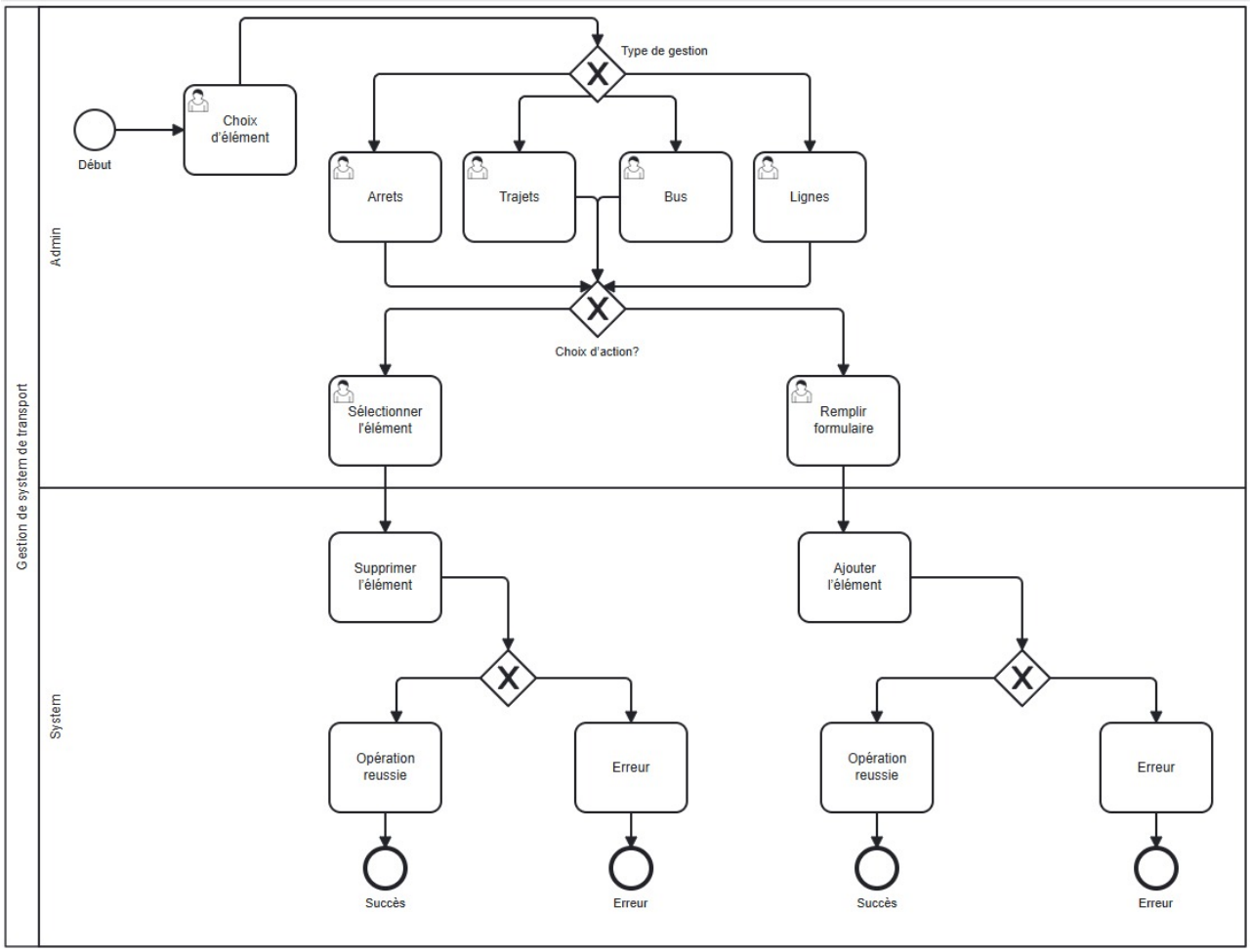


Figure 2.1 : Processus de gestion des éléments administratifs.

2.5.2 Processus de localisation pour un dispositif de localisation

Le deuxième diagramme BPMN décrit le processus de localisation d'un bus pour un tracker. Ce processus commence par l'ouverture de l'application par l'administrateur. L'administrateur doit sélectionner l'ID du bus qu'il souhaite suivre. Ensuite, le système obtient la position GPS du bus toutes les 30 secondes et la sauvegarde dans la base de données.

- **Sélection du bus** : L'administrateur choisit l'ID du bus à suivre. Après chaque sélection, le système actualise la carte en affichant la dernière position du bus.
- **Suivi en temps réel** : Le système met à jour la position du bus toutes les 30 secondes. La carte se met à jour toutes les 5 secondes, affichant la dernière position du bus en vert pour une visualisation optimale en temps réel.

La figure 2.2 illustre le flux de suivi de la position d'un bus pour un tracker.

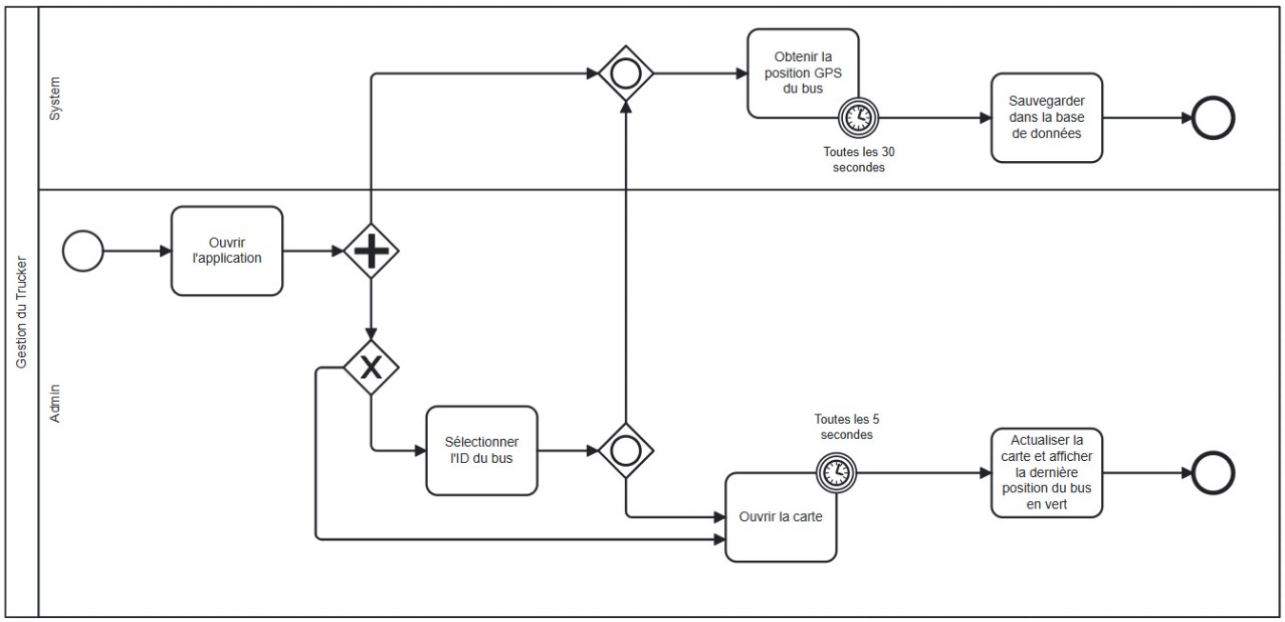


Figure 2.2 : Processus de localisation d'un bus pour un dispositif de localisation.

2.6 Conclusion

Ce chapitre a permis d'établir les bases fonctionnelles du projet en identifiant les besoins des utilisateurs, les fonctionnalités essentielles à développer et les contraintes techniques à respecter. L'approche par cas d'utilisation a structuré l'analyse, facilitant la planification du développement.

Les fonctionnalités majeures suivi en temps réel, consultation d'horaires, personnalisation via favoris, gestion administrative répondent aux attentes des deux profils d'acteurs : usagers et administrateurs.les contraintes (performance, sécurité, disponibilité) ont également été prises en compte pour garantir la viabilité du système.

Le prochain chapitre approfondira la phase d'analyse et de conception en modélisant les entités, les interactions et les structures logiques du système.

Chapitre 3

Analyse et conception

3.1 Introduction

Le chapitre précédent a permis une identification précise des besoins fonctionnels et non fonctionnels du projet, en précisant les attentes des usagers et des administrateurs ainsi que les contraintes techniques que l'on s'impose. Ce sont donc les bases sur lesquelles on va s'appuyer pour faire la conception du système.

Ce chapitre traite de la formalisation de la solution technique choisie. Il traduit les besoins exprimés sous forme de modèles conceptuels et d'architecture logicielle, il justifie les choix de conception pour garantir la cohérence, la maintenabilité et l'efficacité de l'application.

3.2 Formalisme de modélisation (UML, etc.)

3.2.1 Motivation et choix

Le langage UML a été choisi comme formalisme de modélisation en raison de ses nombreux avantages :

- Il permet une représentation visuelle du système à travers différents types de diagrammes (classes, séquences, activités).
- Une représentation claire et intuitive est offerte, compréhensible aussi bien par les développeurs que par les non-techniciens.
- Les relations entre les entités, les interactions et les processus du système sont clarifiées dès les premières étapes du développement.

3.2.2 Diagrammes retenus

Afin de modéliser les différents aspects fonctionnels et structurels du système, plusieurs types de diagrammes UML ont été utilisés. Chacun offre une vision complémentaire du fonctionnement de l'application :

- **Diagramme de cas d'utilisation :** Il modélise les interactions entre les différents acteurs du système (Usager, Administrateur, Système de géolocalisation) et les fonctionnalités principales qu'ils peuvent utiliser. Ce diagramme permet de représenter de manière globale les services proposés par le système ;
- **Diagramme de classes :** Il modélise les entités principales du système ainsi que leurs relations, attributs et opérations. Il permet de structurer les données et de définir l'architecture du système. Les principales classes représentées incluent notamment Bus, Ligne, Arrêt, Trajet, Utilisateur, Favori, et les entités liées à la géolocalisation ;
- **Diagramme de séquence :** Il illustre les échanges dynamiques entre les acteurs du système (Usager, Administrateur, Système), en représentant le déroulement temporel des interactions pour un scénario spécifique, tel que le processus d'authentification ;
- **Diagramme d'activités :** Il est utilisé pour représenter les processus dynamiques du système, en modélisant les différentes étapes, décisions et flux d'informations dans un enchaînement logique. Il permet de visualiser de manière détaillée des scénarios tels que le processus de localisation d'un bus en temps réel.

3.3 Architecture logicielle

Le système repose sur une architecture hybride combinant les principes de l'architecture monolithique et de l'architecture modulaire orientée API REST. Cette combinaison permet de tirer parti des avantages des deux approches : simplicité de développement et de déploiement d'un côté, flexibilité et évolutivité de l'autre. Les différentes composantes du système (frontend web, application mobile, backend PHP, base de données) interagissent via des échanges standardisés en format JSON grâce aux API REST, assurant ainsi un découplage logique tout en conservant une base de code unifiée.

3.3.1 Approche architecturale

Le choix de l'architecture logicielle a constitué un facteur déterminant dans la réussite du projet. Deux options principales ont été analysées lors de la conception :

- **Architecture monolithique :**
  - **Avantages :** Développement initial simplifié, déploiement centralisé, configuration rapide et maintenance cohérente au sein d'un seul projet ;
  - **Inconvénients :** Couplage fort entre les composants, faible modularité, complexité croissante lors de l'évolution du système.
- **Architecture modulaire orientée API REST :**
  - **Avantages :** Séparation claire des responsabilités, modularité du code, communication standardisée via API REST (format JSON), compatibilité avec les clients web et mobile, meilleure testabilité et évolutivité ;
  - **Inconvénients :** Coordination plus complexe entre les modules, nécessité d'une bonne gestion des échanges de données et des erreurs réseau.

Choix retenu : Pour ce projet, une approche hybride a été adoptée. Le backend a été développé sous forme d'un monolithe modulaire, structuré en couches (entités, services, DAO, contrôleurs, API), tandis que la communication avec les clients (web et mobile) se fait via des API REST, assurant un découplage fonctionnel. Cette solution permet de :

- bénéficier de la simplicité de déploiement d'un monolithe ;
- structurer proprement le code en modules indépendants ;
- faciliter les échanges multiplateformes grâce aux API REST ;
- garantir une bonne maintenabilité et une évolutivité progressive du système.

3.3.2 Architecture logicielle du système

Le système est organisé autour des modules suivants :

- **Frontend Web :** Développé avec React.js et Tailwind CSS, il gère l'interface utilisateur pour le web. La communication avec le backend s'effectue via des appels API REST réalisés avec la bibliothèque Axios. Les données reçues (au format JSON) sont affichées sur des composants interactifs, notamment sur la carte Google Maps ;
- **Backend PHP :** Implémenté en PHP, ce module centralise la logique applicative. Il expose des Web Services REST qui reçoivent les requêtes HTTP (GET/POST) des clients (web et mobile), interagit avec la base de données MySQL, et renvoie les réponses au format JSON. L'architecture suit un modèle inspiré MVC (Model / Controller), sans couche Vue (les réponses sont purement API) ;
- **Base de données :** Gérée par MySQL, elle assure la persistance des informations (bus, lignes, trajets, arrêts, utilisateurs, positions GPS, favoris). Les échanges avec le backend sont réalisés via des requêtes SQL sécurisées ;
- **Application mobile (Android) :** Développée en Java sous Android Studio, l'application mobile permet la collecte et l'envoi en temps réel des positions GPS des bus, via des requêtes HTTP POST utilisant la bibliothèque Volley. Elle utilise également les API Google Location Services pour l'acquisition des coordonnées de géolocalisation ;
- **Outil de test (Postman) :** utilisé pour tester manuellement les Web Services REST exposés par le backend, vérifier la conformité des échanges (format JSON) et valider le bon fonctionnement des API.

3.3.3 Architecture technique

La figure 3.1 présente l'organisation générale du système, en mettant en évidence les échanges entre les différentes composantes et les flux de données.

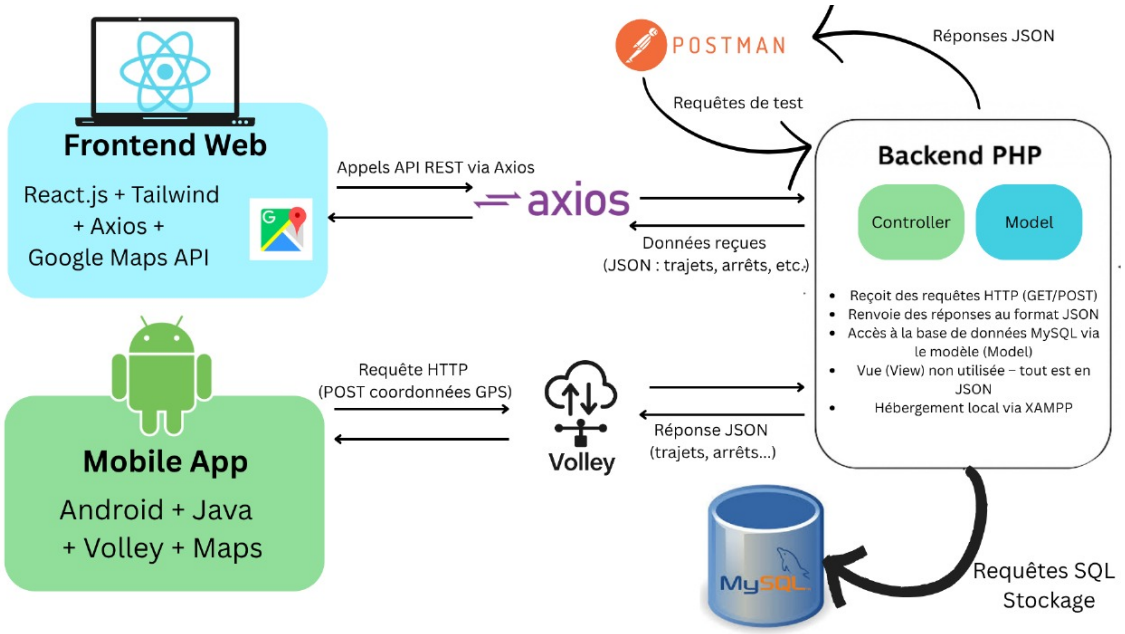


Figure 3.1 : Schéma de l'architecture hybride : monolithe modulaire avec communication via API REST.

Description textuelle

Le schéma présenté illustre de manière claire le fonctionnement interne du système de géolocalisation en temps réel des bus, en mettant en évidence les échanges de données entre les différents modules. Il montre d'abord comment l'application mobile, installée sur les smartphones placés à bord des bus, récupère les coordonnées GPS grâce aux services de localisation d'Android, puis les envoie au backend PHP via des requêtes HTTP POST, en utilisant la bibliothèque Volley. Ces requêtes sont reçues par les contrôleurs du backend, qui traitent les données et les stockent dans la base de données MySQL via des requêtes SQL sécurisées.

En parallèle, les usagers accèdent à une interface web développée avec React.js, stylisée avec Tailwind CSS. Cette interface interagit avec le backend en envoyant des appels API REST via Axios, notamment pour récupérer les données relatives aux lignes, arrêts, trajets ou positions actualisées des bus. Ces requêtes (souvent de type GET) sont traitées par les mêmes contrôleurs du backend, qui consultent la base de données, construisent des réponses au format JSON, puis les renvoient au frontend. Ces données sont ensuite exploitées par les composants React, notamment pour l'affichage dynamique sur la carte Google Maps.

Le schéma met également en évidence l'utilisation de Postman comme outil de test des Web Services. Cet outil permet d'émettre manuellement des requêtes HTTP vers le backend, sans passer par les interfaces client, afin de vérifier la validité des réponses JSON, tester les points d'entrée de l'API et détecter d'éventuelles erreurs.

La communication dans ce système est entièrement basée sur le protocole HTTP et le format JSON, garantissant une interopérabilité maximale. Elle est aussi bidirectionnelle et asynchrone : le frontend et l'application mobile envoient des requêtes indépendantes, sans connexion persistante, et attendent une réponse immédiate du serveur. Axios et Volley jouent un rôle crucial en encapsulant les objets côté client, en les envoyant sous forme de requêtes HTTP, puis en traitant les réponses JSON pour les intégrer dans l'interface utilisateur ou les journaux du système.

Enfin, le schéma souligne l'organisation interne du backend : bien que monolithique, il est structuré de manière modulaire avec une séparation entre contrôleurs (chargés de recevoir les requêtes et de formuler les réponses) et modèles (chargés de l'accès aux données MySQL). Aucun rendu HTML n'est généré : l'ensemble des interactions est purement orienté API, ce qui renforce la modularité, la maintenabilité et la

sécurité du système. Ainsi, le schéma offre une lecture fonctionnelle complète du cycle de vie des données, de leur collecte sur le terrain jusqu'à leur affichage en temps réel dans l'interface web.

3.3.4 Discussion

L'architecture retenue combine les principes d'un **monolithe structuré en modules** avec une communication orientée **API REST**. Ce choix répond pleinement aux exigences fonctionnelles et non fonctionnelles identifiées lors de l'analyse des besoins. Il présente plusieurs avantages majeurs pour ce type de système de géolocalisation en temps réel et de gestion de réseau de transport.

- **Modularité :** Le backend, bien que monolithique, est structuré en modules logiques indépendants (entités, services, DAO, contrôleurs). Chaque composant du système (frontend web, application mobile, backend, base de données) peut interagir via des API REST standardisées, permettant une maintenance souple et une bonne évolutivité ;
- **Adaptabilité aux volumes croissants :** L'architecture orientée API permet de supporter un nombre croissant de clients (usagers, administrateurs, dispositifs mobiles) en répartissant la charge de manière efficace. Le frontend reste léger, tandis que le backend peut être optimisé indépendamment pour améliorer les performances ;
- **Performance :** Les échanges de données via API REST sont conçus pour être rapides et efficaces. L'usage d'appels asynchrones côté client et le traitement optimisé côté serveur garantissent une bonne réactivité, y compris lors de l'actualisation en temps réel des positions de bus ;
- **Interopérabilité :** L'utilisation de standards ouverts (JSON pour les échanges, HTTP/REST pour les communications) garantit une compatibilité maximale avec des clients et services tiers, facilitant l'intégration future d'autres systèmes (portails municipaux, systèmes de billettique...) ;
- **Sécurité :** La séparation entre frontend, backend et base de données permet un meilleur contrôle des flux et l'application de règles de sécurité. Le backend centralise l'authentification, la gestion des droits d'accès, et la validation des données. Les requêtes préparées assurent une protection efficace contre les injections SQL.

En conclusion, cette architecture hybride assure une base technique robuste, modulaire et sécurisée, parfaitement adaptée aux exigences d'un système de géolocalisation en temps réel. Elle permet également d'accompagner sereinement les évolutions futures du projet, comme une potentielle migration vers des microservices.

3.4 Diagrammes de conception

Cette section présente les principaux diagrammes utilisés pour modéliser le comportement, les processus et la structure du système. Ces représentations visuelles permettent de vérifier la conformité de l'implémentation aux exigences fonctionnelles et facilitent la communication entre les parties prenantes.

3.4.1 Les cas d'utilisation

Afin d'assurer une meilleure lisibilité et de faciliter la compréhension des interactions entre les différents types d'acteurs et le système, les cas d'utilisation ont été représentés sous la forme de trois diagrammes distincts, chacun centré sur un acteur principal :

- **Un diagramme des cas d'utilisation pour l'Usager**, qui formalise les actions accessibles aux utilisateurs finaux (voyageurs) ;
- **Un diagramme des cas d'utilisation pour l'administrateur**, qui regroupe les fonctionnalités de gestion, de supervision et de maintenance du système ;
- **Un diagramme des cas d'utilisation du système**, qui met en évidence les processus automatiques et les interactions avec les services externes (géolocalisation des bus en temps réel).

Chaque diagramme est complété par une description textuelle détaillée de chaque cas d'utilisation, structurée selon les éléments suivants :

- **Préconditions** ;

- **Flux principal** ;
- **Flux alternatifs** ;
- **Postconditions**.

Cas d'utilisation pour l'Usager

Le diagramme des cas d'utilisation pour l'Usager présente l'ensemble des fonctionnalités et interactions possibles entre l'utilisateur final (voyageur) et le système. Il permet de visualiser de manière synthétique les actions principales accessibles à l'Usager via l'application web ou mobile.

La figure 3.2 illustre le diagramme des cas d'utilisation associé à l'acteur Usage.

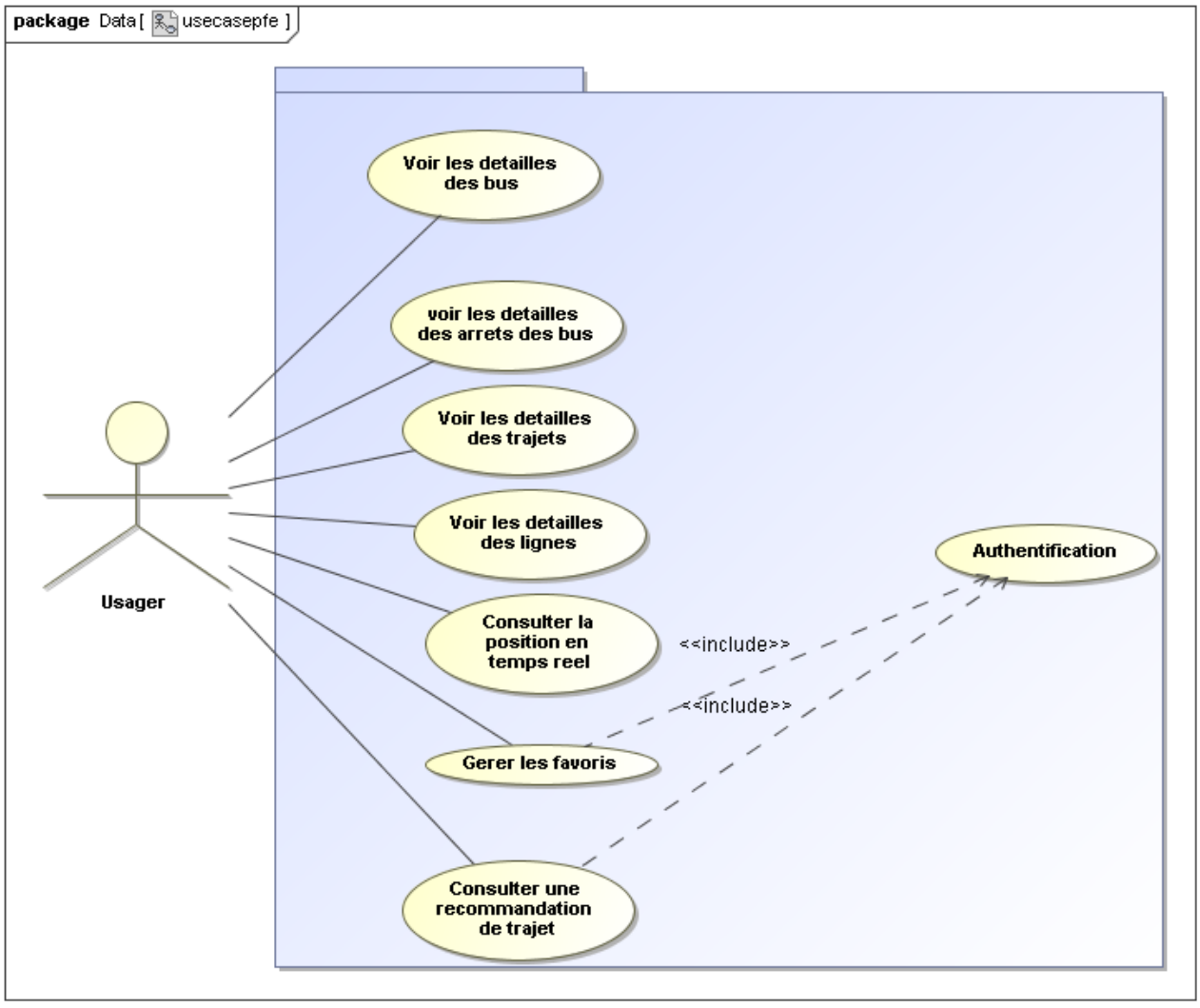


Figure 3.2 : Diagramme des cas d'utilisation pour l'Usager.

Description textuelle des fonctionnalités accessibles à l'Usager.

- **Cas d'utilisation associés :** Le diagramme des cas d'utilisation pour l'Usager présente l'ensemble des fonctionnalités accessibles via l'application. L'Usager peut consulter les détails des bus, des arrêts, des trajets et des lignes, visualiser la position des bus en temps réel, gérer ses favoris et obtenir des recommandations de trajet. L'authentification est requise pour certaines fonctionnalités avancées ;
- **Préconditions :**L'application web doit être pleinement opérationnelle, avec le serveur et les interfaces de programmation (API) accessibles via Internet. L'Usager doit disposer d'un navigateur web compatible (sur ordinateur, tablette ou smartphone). Pour accéder aux fonctionnalités avancées, telles que la gestion des favoris et les recommandations personnalisées, une authentification préalable est requise ;

- **Flux principal :** L'Usager lance l'application, accède à l'interface et consulte les différentes informations relatives aux transports. Il peut suivre en temps réel la position des bus et, après authentification, personnaliser son espace en ajoutant des favoris ou en consultant des recommandations adaptées ;
- **Flux alternatifs :** Si l'authentification échoue, un message invite l'Usager à réessayer. En cas d'indisponibilité temporaire du service temps réel ou de l'absence de bus en circulation, des messages d'information appropriés sont affichés ;
- **Postconditions :** À l'issue de l'interaction, l'Usager aura consulté les informations du réseau de transport, suivi les bus en temps réel et, le cas échéant, mis à jour ses préférences personnelles (favoris, recommandations).

Cas d'utilisation pour l'Administrateur

Le diagramme des cas d'utilisation pour l'Administrateur présente l'ensemble des fonctionnalités de gestion, de supervision et de maintenance que cet acteur peut exercer sur le système. Il formalise les interactions possibles entre l'Administrateur et l'application web, notamment la gestion des différentes entités du réseau de transport et la consultation des données opérationnelles.

La figure 3.3 illustre le diagramme des cas d'utilisation associé à l'acteur Administrateur.

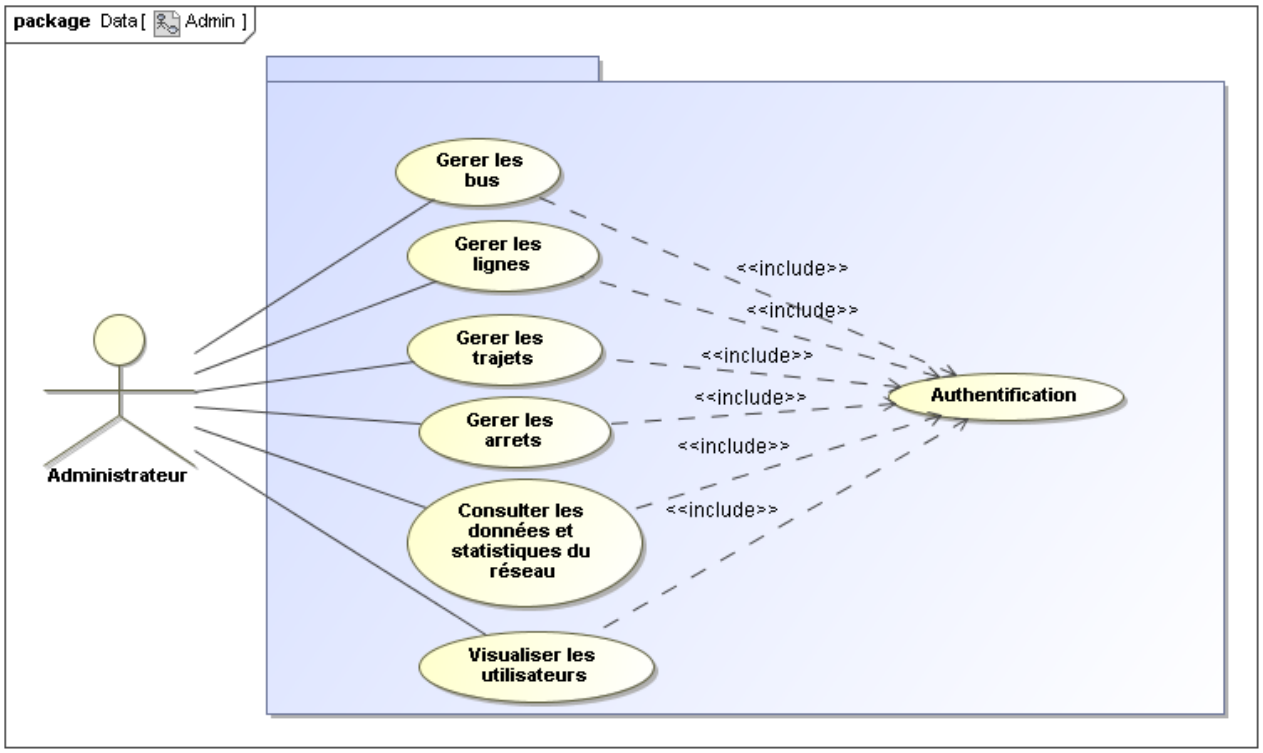


Figure 3.3 : Diagramme des cas d'utilisation pour l'Administrateur.

Description textuelle des fonctionnalités de gestion disponibles pour l'Administrateur.

- **Cas d'utilisation associés :** L'Administrateur peut gérer les différentes entités du réseau de transport, notamment les bus, les lignes, les trajets et les arrêts. Il peut également consulter les données et les statistiques du réseau et visualiser les informations sur les utilisateurs. L'accès à ces fonctionnalités nécessite une authentification préalable ;
- **Préconditions :** L'application web doit être opérationnelle et le serveur, ainsi que les API, doivent être accessibles. L'Administrateur doit se connecter à l'interface d'administration via un navigateur web compatible. L'authentification est obligatoire pour accéder aux fonctions de gestion ;
- **Flux principal :** L'Administrateur s'authentifie sur l'interface web. Une fois connecté, il accède aux différentes sections de gestion lui permettant de consulter, ajouter, modifier ou supprimer les bus, lignes, trajets et arrêts. Il peut également consulter les statistiques du réseau et visualiser les informations des utilisateurs ;

- **Flux alternatifs :** En cas d'échec d'authentification, un message d'erreur est affiché et l'Administrateur est invité à réessayer. Si certaines données sont momentanément indisponibles (par exemple, problème de connexion au serveur), des messages d'information sont fournis ;
- **Postconditions :** Après l'interaction, les modifications apportées par l'Administrateur sur les données du réseau sont enregistrées et prises en compte par le système.

Cas d'utilisation pour le Système de géolocalisation

Le diagramme des cas d'utilisation pour le Système de géolocalisation met en évidence les processus automatiques gérés par ce composant. Il représente les interactions entre le service de géolocalisation et le reste du système, en particulier pour la collecte et le suivi en temps réel des positions des bus sur le réseau.

La figure 3.4 illustre le diagramme des cas d'utilisation associé au Système de géolocalisation.

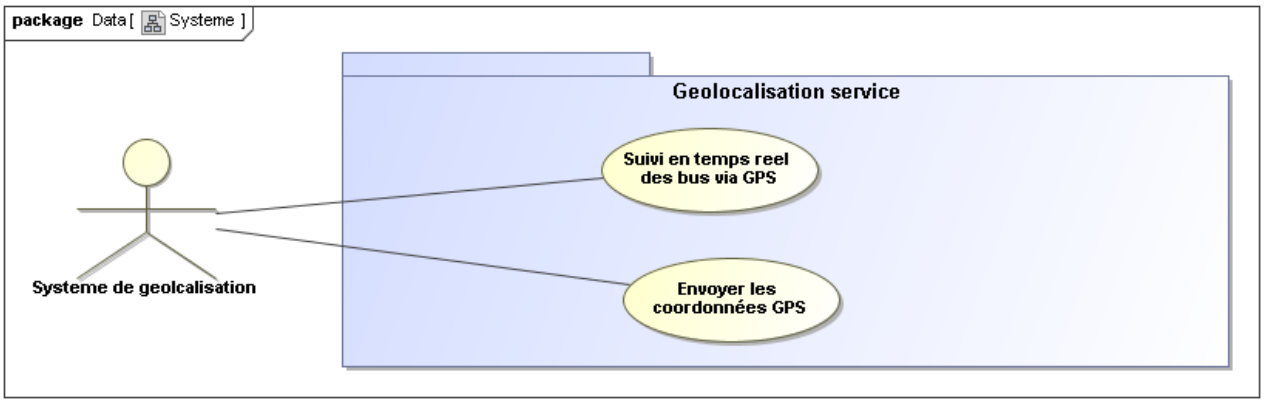


Figure 3.4 : Diagramme des cas d'utilisation pour le Système de géolocalisation.

Description textuelle des fonctionnalités assurées par le Système de géolocalisation.

- **Cas d'utilisation associés :** Le Système de géolocalisation assure le suivi en temps réel des bus via GPS. Il collecte et envoie régulièrement les coordonnées GPS des véhicules au serveur central afin de permettre la visualisation dynamique de leur position dans l'application ;
- **Préconditions :** Les équipements GPS installés à bord des bus doivent être opérationnels. Le système doit disposer d'une connectivité réseau pour transmettre les données GPS vers le serveur ;
- **Flux principal :** Le système embarqué collecte en continu les coordonnées GPS des bus. Ces données sont ensuite envoyées automatiquement et en temps réel au serveur de l'application, où elles sont traitées et mises à disposition des utilisateurs via l'interface web ;
- **Flux alternatifs :** En cas de perte temporaire de signal GPS ou de connectivité réseau, les coordonnées peuvent ne pas être transmises. Lorsque la connexion est rétablie, l'envoi des nouvelles positions reprend automatiquement ;
- **Postconditions :** Les positions actualisées des bus sont enregistrées et affichées dans l'application pour permettre un suivi en temps réel par les usagers et les administrateurs.

3.4.2 Diagrammes de séquence

Scénarios clés

Le scénario présenté dans ce diagramme illustre le processus d'authentification d'un utilisateur, depuis la saisie des identifiants jusqu'à la création d'une session ou l'affichage d'un message d'erreur en cas d'échec. Ce processus constitue un scénario clé du système, car il conditionne l'accès de l'utilisateur aux différentes fonctionnalités offertes par l'application.

La figure 3.5 montre le diagramme de séquence correspondant à ce scénario.

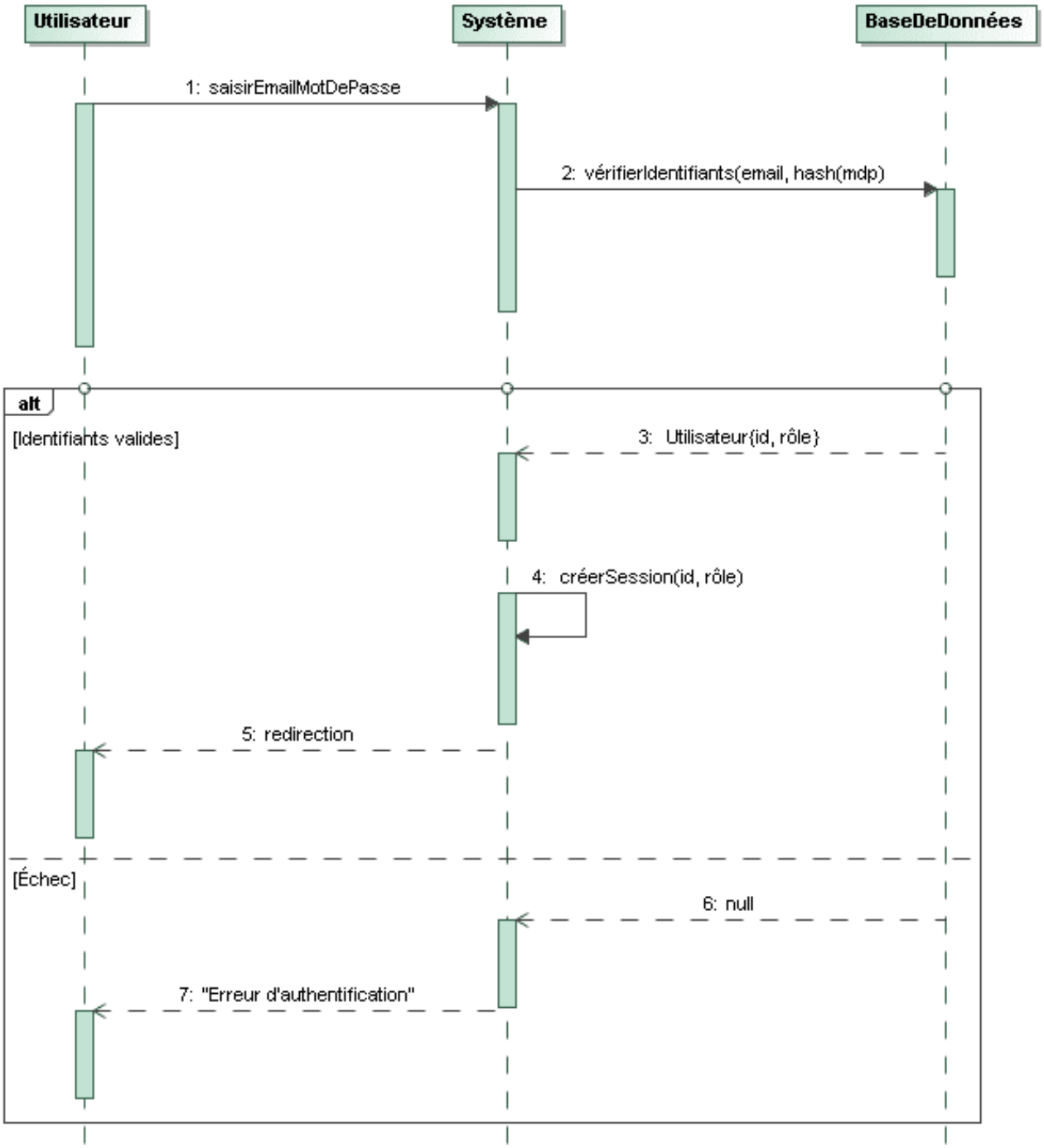


Figure 3.5 : Diagramme de séquence : Vérification des identifiants et création de session.

Description textuelle du scénario

Le scénario décrit dans ce diagramme de séquence correspond au processus complet d'authentification d'un utilisateur. Ce processus implique trois composants essentiels :

- **L'utilisateur** : acteur principal qui initie l'authentification ;
- **Le système applicatif** : composant métier qui traite les identifiants ;
- **La base de données** : composant de persistance contenant les données utilisateur.

Étape 1 : Saisie des identifiants Le processus débute lorsque l'utilisateur accède à l'interface de connexion de l'application. Il saisit son adresse e-mail et son mot de passe dans les champs du formulaire. Cette action est représentée dans le diagramme par le message 1 :saisirEmailMotDePasse(email, mdp).

Étape 2 : Transmission et sécurisation des données Les informations saisies sont transmises au système applicatif, qui applique une fonction de hachage sécurisée au mot de passe avant toute vérification. Cette étape garantit que le mot de passe n'est jamais stocké ni transmis en clair, ce qui est essentiel pour la sécurité.

Étape 3 : Vérification dans la base de données Le système effectue ensuite un appel à la base de données en invoquant la méthode vérifierIdentifiants(email, hash(mdp)) (message 2). Cette méthode vérifie s'il existe un enregistrement utilisateur correspondant à l'adresse e-mail fournie et au mot de passe haché.

Deux cas alternatifs peuvent se produire :

- **Cas 1 : Authentification réussie :**
  Si les identifiants sont corrects, la base de données retourne un objet Utilisateur contenant des informations essentielles telles que l'id de l'utilisateur et son rôle (par exemple, admin ou usager) (message 3 : Utilisateur(id, rôle)). Le système utilise ensuite ces informations pour initialiser une session sécurisée via la méthode créerSession(id, rôle) (message 4). Cela permet de conserver l'état de connexion de l'utilisateur tout au long de sa navigation. Une fois la session créée, le système redirige l'utilisateur vers l'interface adaptée à son profil : page d'accueil générale ou espace personnalisé (par exemple, un tableau de bord administrateur ou un espace utilisateur) (message 5) ;
- **Cas 2 : Authentification échouée :**
  Si les identifiants sont incorrects ou absents de la base de données, cette dernière retourne une valeur nulle(message 6 : null), indiquant l'échec de la vérification. En réponse, le système applicatif renvoie un message d'erreur d'authentification à l'utilisateur, tel que Identifiants incorrects ou Erreur lors de la connexion (message 7). Ce message s'affiche directement dans l'interface de connexion, invitant l'utilisateur à vérifier ses informations et à réessayer.

3.4.3 Diagrammes d'activités

Processus : Affichage de la localisation des bus

Le diagramme d'activité de la figure 3.6 modélise le processus de traitement permettant d'afficher en temps réel la localisation d'un bus sur la carte. Il décrit l'enchaînement des différentes activités, depuis l'action initiale de l'usager jusqu'à la restitution visuelle de la position du bus ou, le cas échéant, l'affichage d'un message d'erreur.

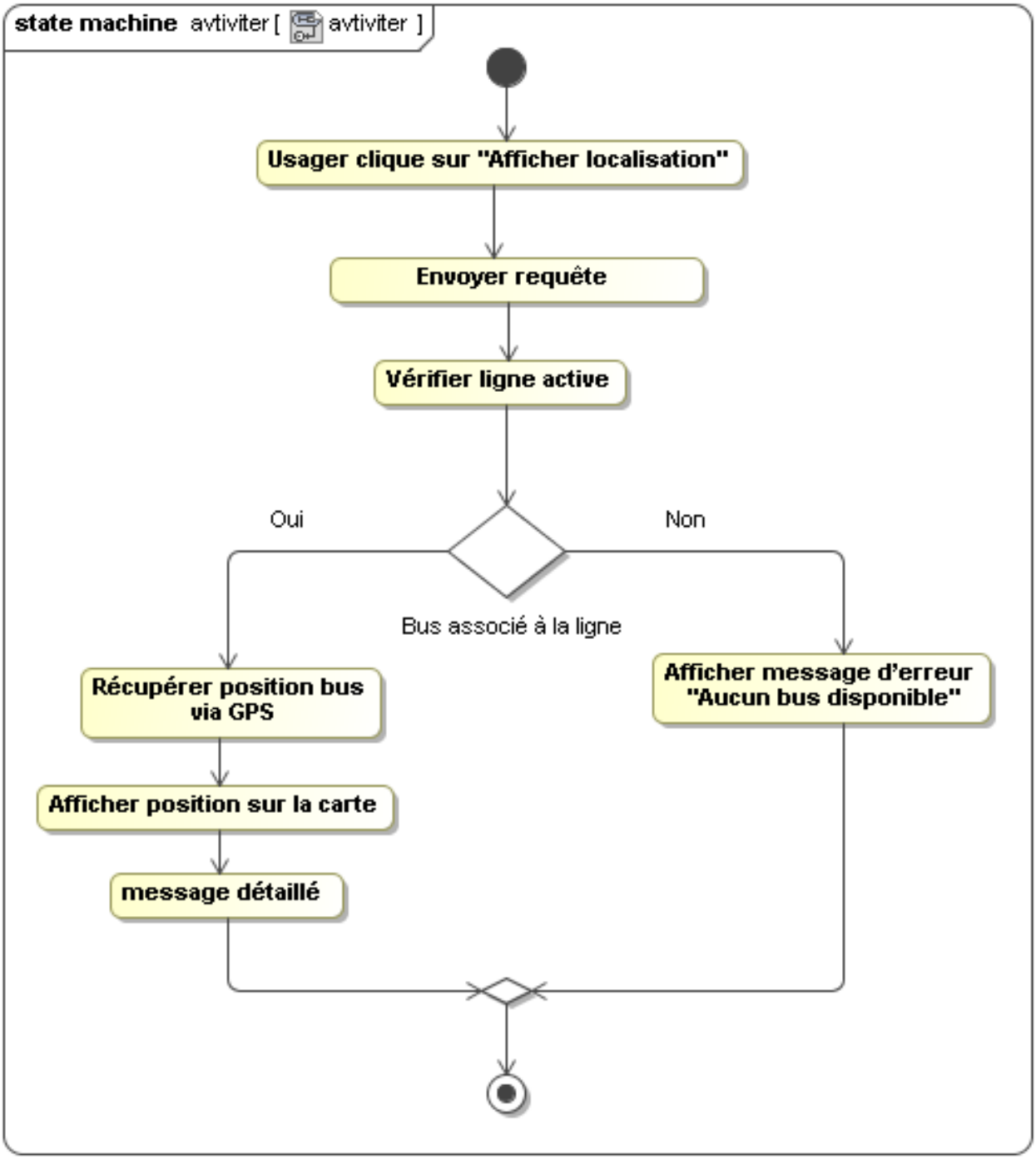


Figure 3.6 : Diagramme d'activité : Processus d'affichage de la localisation des bus.

Description textuelle du processus

Le processus est initié lorsque l'usager clique sur le bouton afficher localisation dans l'interface de l'application. Cette action déclenche l'envoi d'une requête au système applicatif afin de vérifier si la ligne de bus sélectionnée est active et si un bus y est actuellement affecté.

Étape 1 : Vérification de l'état de la ligne Le système commence par contrôler l'état de la ligne demandée. Il vérifie en particulier si un bus est associé à cette ligne à l'instant de la requête.

Étape 2 : Récupération de la position GPS Si un bus est bien affecté à la ligne, le système interroge le service de géolocalisation (via GPS) pour récupérer la position en temps réel de bus. Cette information est obtenue à partir des coordonnées transmises périodiquement par le dispositif embarqué.

Étape 3 : Affichage sur la carte Une fois la position du bus obtenue, elle est immédiatement affichée sur la carte interactive de l'application accessible à l'usager. Ce rendu cartographique permet de visualiser l'emplacement exact du bus en déplacement.

Étape 4 : Informations complémentaires En complément de l'affichage cartographique, un message informatif détaillé est généré.

Dans le cas où aucun bus ne serait associé à la ligne sélectionnée, le système affiche un message d'erreur : Aucun bus disponible, indiquant à l'usager que la géolocalisation n'est pas possible pour le moment sur cette ligne.

3.4.4 Diagramme de classes

Modèle de données : Diagramme de classes du système

Le diagramme de classes présenté dans la figure 3.7 modélise la structure statique du système de géolocalisation des bus. Il définit les principales entités manipulées par l'application, ainsi que les relations qui existent entre elles. Ce modèle de données sert de base à la conception de la base de données et à l'implémentation des composants logiciels.

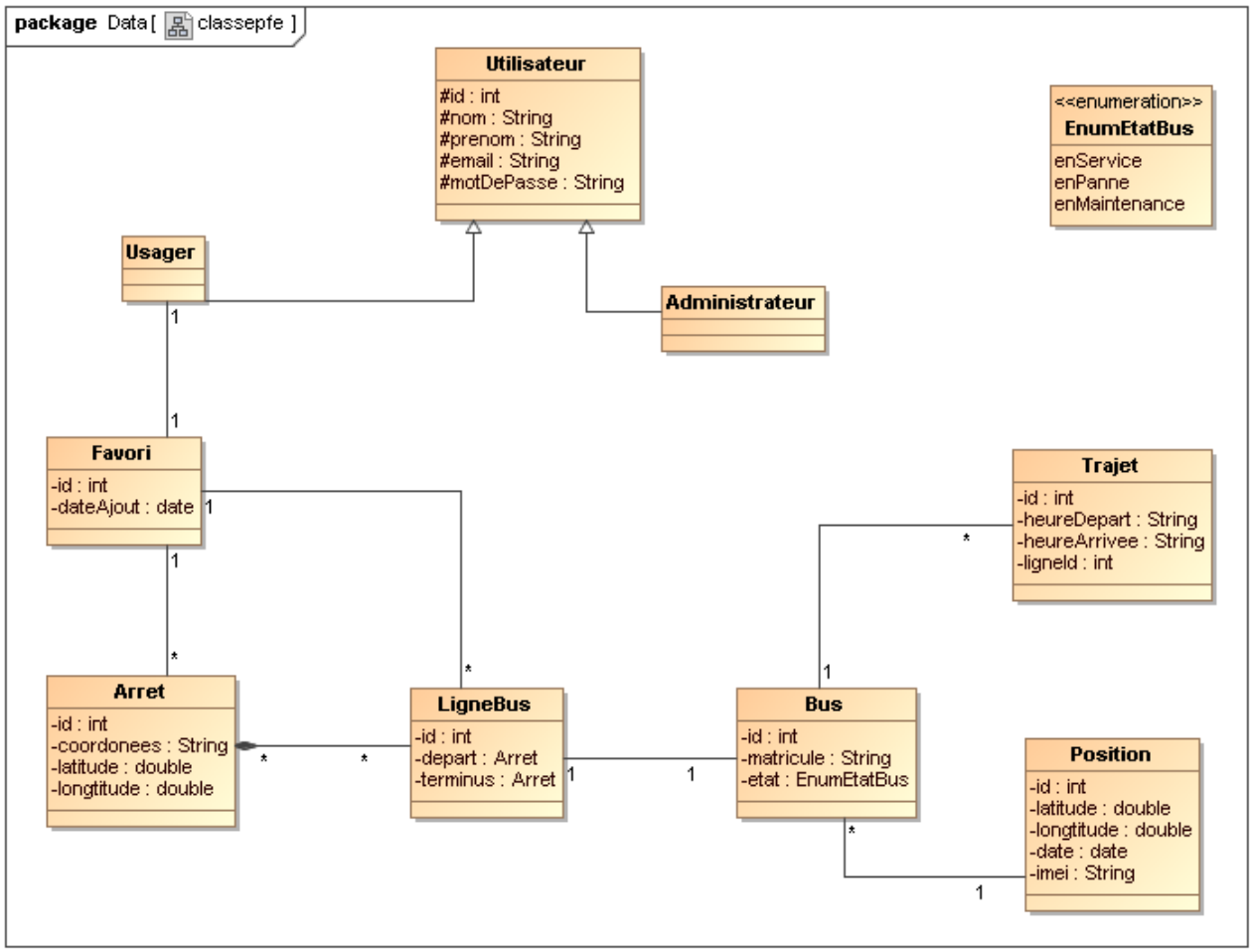


Figure 3.7 : Diagramme de classes du système de géolocalisation des bus.

Description textuelle du modèle

Le système repose sur une hiérarchie d'utilisateurs, modélisée par la classe Utilisateur, qui regroupe les attributs communs à l'ensemble des utilisateurs : identifiant, nom, prénom, adresse e-mail et mot de passe. Deux spécialisations de cette classe sont définies : Usager, qui représente un utilisateur final du système, et Administrateur, qui dispose de droits de gestion avancés.

Chaque Usager peut enregistrer des Favoris, qui peuvent concerner soit un arrêt de bus, soit une ligne de bus, en fonction des préférences de l'utilisateur. La classe Favori est associée à un Usager et peut référencer un Arrêt et/ou une LigneBus. La date d'ajout est également enregistrée.

La classe Arrêt représente un arrêt de bus, identifié par des coordonnées géographiques (latitude et longitude) et par un champ texte qui correspond au nom de l'arrêt. La classe LigneBus est définie par un point de départ et un terminus (deux arrêts), et regroupe l'ensemble des arrêts associés à cette ligne.

Les bus sont modélisés par la classe Bus, qui contient un identifiant, le matricule du véhicule et son état, représenté par l'énumération EnumEtatBus (valeurs possibles : enService, enPanne, enMaintenance). Chaque bus peut être associé à plusieurs Positions au cours du temps. La classe Position enregistre les coordonnées GPS du bus (latitude, longitude), la date de la mesure et l'identifiant du dispositif GPS (IMEI).

Enfin, la classe Trajet permet de modéliser les parcours planifiés des bus. Chaque trajet comprend une heure de départ, une heure d'arrivée et un lien vers la LigneBus concernée.

Ce modèle de classes assure une représentation cohérente des entités du domaine et facilite l'implémentation des principales fonctionnalités du système de géolocalisation.

3.5 Conclusion

Ce chapitre a permis de présenter les choix d'architecture et de modélisation adoptés pour le projet. Une architecture microservices a été privilégiée pour répondre aux exigences de modularité, de capacité de montée en charge et de performance. Les diagrammes UML (classes, séquence, Activités.) ont été utilisés pour formaliser la conception du système et garantir sa cohérence. Ces éléments serviront de base pour l'implémentation effective du système, qui sera détaillée dans le chapitre suivant.

Chapitre 4

Technologies et réalisation

4.1 Introduction

Le chapitre précédent a permis de formaliser la conception du système, en modélisant son architecture logicielle et ses principaux composants sous la forme de diagrammes UML d'une part et d'un schéma d'architecture en couches d'autre part, et ces choix conceptuels constituent désormais le socle utile au passage à l'implémentation concrète du projet.

Ce chapitre décrit la mise en œuvre de l'application : technologies et outils, choix techniques, environnement de déploiement, structure du code, conception de l'interface utilisateur, tests effectués, avant de conclure par une mise en perspective des résultats et des difficultés rencontrées.

4.2 Choix techniques et outils

Le choix des technologies s'est fait selon plusieurs critères. Il s'agit de la facilité d'implémentation, la compatibilité entre les outils, la documentation accessible, la taille d'activité de la communauté et sa volumétrie, ainsi que selon les besoins fonctionnels, techniques et pédagogiques. L'apprentissage de la technologie, la possibilité que l'abstraction du code soit maintenable pour une période prolongée, la simplicité d'évolution contribuaient au choix. Ainsi, un tel choix garantit un développement efficace, soutenable et conforme au contexte éducatif.

4.2.1 Langages, Frameworks, Base de données et Outils

- **Langages de programmation :** Le langage PHP a été choisi pour le développement côté serveur, en raison de sa simplicité d'utilisation, de son caractère open-source et de sa compatibilité native avec MySQL. Il permet une mise en œuvre efficace pour un projet de taille moyenne, tout en offrant un contrôle précis sur la logique métier.

  Pour le front-end, le langage JavaScript a été utilisé via la bibliothèque React.js. Cette solution permet de concevoir des interfaces dynamiques et interactives, en s'appuyant sur une architecture de composants réutilisables et sur l'optimisation du Virtual DOM, garantissant ainsi de bonnes performances et une expérience utilisateur fluide et réactive.

  Enfin, l'application mobile a été développée en Java, un choix motivé par la compatibilité native de Java avec Android, la maturité de l'écosystème, et la disponibilité des API de géolocalisation de Google Maps, essentielles pour le suivi des bus en temps réel ;
- **Frameworks et bibliothèques :** Côté serveur, aucun framework PHP n'a été utilisé pour une compréhension fine du fonctionnement du projet (routes, logique métier, MVC). Le front-end utilise React.js pour les interfaces interactives et Tailwind CSS pour un design rapide et moderne. L'application mobile repose sur le SDK Android et les API de Google Maps pour la géolocalisation. La communication réseau est gérée par Axios côté front-end pour effectuer des requêtes HTTP et Volley côté mobile pour la gestion des données et des requêtes réseau ;
- **Base de données :** Le système repose sur la base de données relationnelle open source MySQL, reconnue pour sa robustesse, sa fiabilité et sa facilité d'intégration avec PHP. Ce choix s'est imposé pour sa compatibilité avec les outils du projet, ainsi que pour sa capacité à gérer des données bien

structurées et des relations complexes, essentielles dans le cadre de la gestion des réseaux de transport (lignes, arrêts, trajets, utilisateurs, positions GPS) ;

- **Critères de sélection :** Les technologies retenues ont été choisies selon plusieurs critères complémentaires : facilité d'intégration entre les différents composants, compatibilité avec les objectifs pédagogiques du projet, disponibilité de ressources documentaires et d'une communauté active, performances adaptées aux besoins ciblés, et enfin, valeur éducative permettant de renforcer la compréhension des mécanismes internes du développement web et mobile.

  Par ailleurs, l'expérience préalable avec ces technologies (PHP, MySQL, React.js, Java) a permis de réduire la courbe d'apprentissage et de garantir une progression rapide et efficace du développement.

Dans ce travail, plusieurs sources ont été mobilisées pour éclairer et justifier les choix techniques. Les avantages de PHP pour le développement serveur sont décrits dans la documentation officielle de PHP **?**, qui met en avant sa simplicité d'utilisation et sa compatibilité avec MySQL, ainsi que dans des articles professionnels tels que **?** et **?**, qui soulignent la flexibilité et la portabilité de PHP dans les projets web.

Du côté du front-end, les bénéfices de React.js sont détaillés dans sa documentation officielle **?**, notamment l'architecture par composants et les performances du Virtual DOM. Des analyses techniques telles que celles de **?** et **?** viennent compléter cette perspective. Enfin, l'étude académique comparative de **?** démontre l'efficacité de React.js pour la gestion d'interfaces complexes.

Concernant la partie mobile, l'utilisation de Java et des API Google Maps s'appuie sur la documentation officielle Android Developers **?** et sur les guides de Google Maps Platform **?**, qui précisent les meilleures pratiques pour l'intégration des services de géolocalisation.

4.2.2 Outils de développement (IDE, gestion de versions...)

Plusieurs outils professionnels ont été utilisés dans le développement du projet, chacun étant adapté à une composante technique spécifique de l'application.

- **Environnement de développement intégré (IDE) :**
  - **Visual Studio Code :** a été employé pour le développement du front-end en React.js, en raison de sa légèreté, de ses extensions riches (notamment pour JavaScript, Tailwind CSS, et la gestion des composants React) et de son intégration efficace avec Git ;
  - **NetBeans :**a été utilisé pour la partie back-end en PHP, notamment pour sa prise en charge native de PHP 5.6, parfaitement adaptée à l'environnement XAMPP utilisé, et pour son interface complète facilitant la gestion des projets PHP ;
  - **Android Studio :**a été mobilisé pour le développement de l'application mobile de géolocalisation des bus, car il constitue l'environnement officiel et le plus complet pour la création d'applications Android, offrant un émulateur performant, des outils de debugging avancés, ainsi qu'une intégration fluide avec les bibliothèques de Google Maps.
- **Système de gestion de versions :** Le système Git a été utilisé pour assurer le suivi des versions du projet.l'ensemble du code source a été hébergé sur GitHub, permettant de sauvegarder l'historique de développement, de gérer les différentes branches du projet, et de faciliter la collaboration et les retours sur les modifications.
- **Autres outils de collaboration et de tests :**
  - **XAMPP :**a servi d'environnement local pour exécuter le back-end, avec Apache, MySQL et phpMyAdmin pour gérer la base de données relationnelle ;
  - **Postman :**Un constructeur de requêtes a ensuite été utilisé pour faire des demandes indépendantes de PHP via des api, avec des requêtes simulées à l'aide de différents types déterminant le type pour l'obtention ou l'envoi de données et une inspection détaillée de la reponse pour faciliter le débogage de la structure ;
  - **Google Maps API :** Utilisée pour intégrer des fonctionnalités de cartographie et de géolocalisation, permettant d'afficher des données spatiales ou d'interagir avec des services de localisation.

4.3 Implémentation et interfaces

Cette section est consacrée à la description de l'implémentation du système ainsi qu'à la présentation des interfaces utilisateur.

4.3.1 Structure du code

Structure de back-end

La structure du code back-end est organisée de manière modulaire, selon une logique de séparation des responsabilités. Chaque dossier joue un rôle bien défini dans l'architecture de l'application :

- **/classes** : contient les classes métier représentant les entités du système (Bus.php, LigneBus.php, Trajet.php, etc.). Ces classes correspondent aux tables de la base de données et encapsulent les attributs et comportements de chaque entité ;
- **/service** : regroupe les services métier qui implémentent la logique applicative associée aux entités (recherche, filtrage, insertion, etc.), comme BusService.php ou FavorisService.php ;
- **/controller** : contient les contrôleurs permettant de gérer les opérations CRUD (ajout, modification, suppression, etc.) via des scripts PHP comme AddBus.php ou DeleteFavoris.php ;
- **/ws** : réunit les Web Services (API REST) servant à exposer les données vers le front-end (React.js) au format JSON. Par exemple : LoadBus.php, LoadLigneBus.php, Recommandation.php ;
- **/dao** : contient les classes d'accès aux données, notamment IDao.php qui définit une interface générique pour les opérations de base sur les entités (CRUD) ;
- **/connexion** : inclut les fichiers de connexion à la base de données, notamment Connexion.php, qui configure l'accès à MySQL via PDO ;
- **/localisation/** : dédié à la gestion de la géolocalisation, ce dossier contient des fichiers comme `shoPosition.php` pour afficher ou manipuler les données de position GPS.

Structure du front-end

La structure de code front-end est organisée de manière modulaire afin de séparer les responsabilités fonctionnelles (navigation, gestion, affichage, services API, etc.). Chaque dossier joue un rôle spécifique dans l'architecture de l'application :

- **/src/components** : contient les composants réutilisables de l'interface utilisateur, tels que Navbar.jsx, NavItem.jsx, Footer.jsx, LigneCard.jsx ou UserGreeting.jsx ;
- **/src/contexts** : regroupe les contextes applicatifs gérant les états globaux comme l'authentification (AuthContext.jsx).
- **/src/pages/gestion** : regroupe les pages dédiées à la gestion des entités du réseau, accessibles par l'administrateur : GestionArrets.jsx, GestionBus.jsx, GestionLignes.jsx, GestionTrajets.jsx ;
- **/src/pages/rapports** : pages affichant des rapports statistiques sur les bus, les lignes, les arrêts, etc. : RapportsBus.jsx, RapportsLignes.jsx, etc ;
- **/src/pages/utilisateurs** : pages permettant à l'administrateur de consulter la liste des utilisateurs (administrateurs, usagers) ainsi que des statistiques associées (nombre total, répartition par rôle, etc.), sans possibilité de modification ou de suppression : TousUtilisateurs.jsx ;
- **/src/pages/** (racine) : regroupe les pages principales comme AdminDashboard.jsx, HomePage.jsx, Favoris.jsx, Connexion.jsx, Inscription.jsx, etc.
- **/src/pages/** (cartographie) : pages dédiées à l'affichage des cartes interactives et de la géolocalisation des bus : MapGlobalePage.jsx, TrajetsPage.jsx, LignesPage.jsx, etc ;
- **/src/services** : contient les services responsables des appels API vers le back-end, centralisés dans api.js ;
- **/src/assets** : regroupe les ressources graphiques de l'interface (images, icônes, logos), telles que bus-pattern.png, hero-bus.png, etc ;
- **App.js** et **index.js** : fichiers d'entrée de l'application, chargés de l'initialisation et du routage global entre les pages ;

Structure de l'application mobile (Android – Java)

L'application mobile Android est dédiée à la géolocalisation en temps réel des bus. Elle est installée sur un smartphone embarqué dans chaque véhicule et se charge de collecter les coordonnées GPS pour les envoyer automatiquement au serveur via API. La structure du code est organisée comme suit :

- **/manifests** : contient le fichier AndroidManifest.xml, qui déclare les activités principales de l'application et les permissions nécessaires, notamment pour l'accès à la localisation GPS, à Internet et au réseau.
- **/java/com.example.localisationapp** : contient la logique principale de l'application, incluant :
  - MainActivity.java : écran principal, point d'entrée de l'application ;
  - MapsActivity.java : affichage de la carte Google Maps avec la position GPS en temps réel du bus.
- **/res/layout** : regroupe les fichiers XML définissant l'interface utilisateur, comme :
  - activity_main.xml : mise en page de l'accueil ;
  - activity_maps.xml : interface intégrant la carte et les éléments de navigation.
- **/res/values** : contient les ressources globales de l'application :
  - google_maps_api.xml : contient la clé API Google Maps ;
  - strings.xml, colors.xml : textes statiques, couleurs, et autres paramètres visuels.
- **Gradle Scripts** : contient les fichiers de configuration du projet :
  - build.gradle.kts, settings.gradle.kts, gradle.properties : gestion des dépendances, configuration du compilateur et paramètres du projet.

4.3.2 Interfaces utilisateur :

Cette section présente les différentes interfaces proposées par l'application, en fonction des profils d'utilisateurs et des points d'entrée disponibles. Chaque interface est adaptée à un usage spécifique, assurant une navigation claire et un accès sécurisé aux fonctionnalités.

L'application propose trois types d'interfaces principales :

- **Interface d'authentification** : accessible à tous les utilisateurs (usagers et administrateurs), elle permet la connexion, l'inscription, la réinitialisation de mot de passe, et la vérification de code. Elle constitue le point d'entrée sécurisé de l'application ;
- **Interface usager** : destinée au grand public, cette interface permet la consultation des arrêts, lignes, trajets, la visualisation en temps réel des bus sur la carte, la gestion des favoris et la réception de recommandations personnalisées ;
- **Interface administrateur** : réservée aux gestionnaires du système, elle permet de visualiser les données du réseau (bus, lignes, trajets, arrêts), d'accéder aux tableaux de bord statistiques, et de consulter la liste des utilisateurs.

Interfaces d'accès à l'application

Les interfaces d'accès à l'application ont été conçues pour permettre une authentification simple et sécurisée des utilisateurs.elles offrent une navigation claire et intuitive, garantissant une prise en main rapide. Les principales pages disponibles sont :

- **Page d'inscription** :

  L'internaute atterrit sur la page d'inscription afin de créer un nouveau compte. Il découvre un formulaire à remplir indiquant son nom, son prénom, son adresse électronique, son mot de passe. Il renseigne les champs et clique sur S'inscrire.

  Si une erreur est détectée, comme une adresse e-mail déjà utilisée ou un mot de passe non conforme, le système en informe immédiatement l'utilisateur en affichant un message d'erreur sous le champ concerné. Celui-ci peut alors corriger les données erronées avant de valider à nouveau le formulaire.

Une fois les informations corrigées et validées, l'inscription est acceptée. Un message de confirmation s'affiche brièvement à l'écran pour signaler la création du compte. L'utilisateur est alors redirigé automatiquement vers la page de connexion.

La figure 4.1 représente la page d'inscription de l'utilisateur.

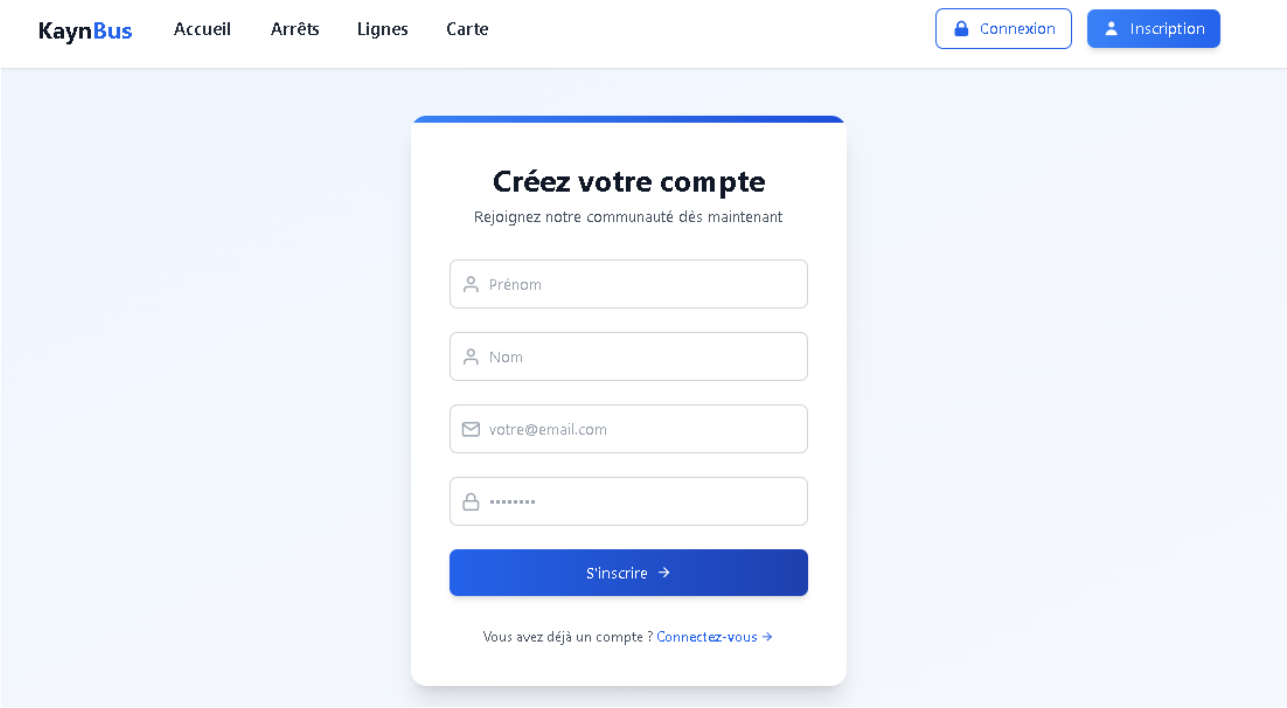


Figure 4.1 : Page d'inscription (interface usager).

- **Page de connexion** : Depuis l'interface d'accueil, l'utilisateur accède à la page de connexion afin d'accéder à son espace personnel. Il saisit son adresse e-mail et son mot de passe dans les champs prévus à cet effet, puis clique sur le bouton Se connecter.

  Si les informations sont incomplètes ou mal formatées (ex. : champ vide, adresse e-mail incorrecte), le système affiche un message d'erreur pour inviter l'utilisateur à corriger les données saisies. Si les identifiants ne correspondent à aucun compte existant, un message d'erreur s'affiche également, signalant que l'e-mail ou le mot de passe est incorrect.

  Une fois les données valides et authentifiées, l'utilisateur est redirigé automatiquement vers l'interface correspondant à son rôle (usager ou administrateur), où il peut accéder à ses fonctionnalités.

  La figure 4.2 représente la page de connexion de l'utilisateur.

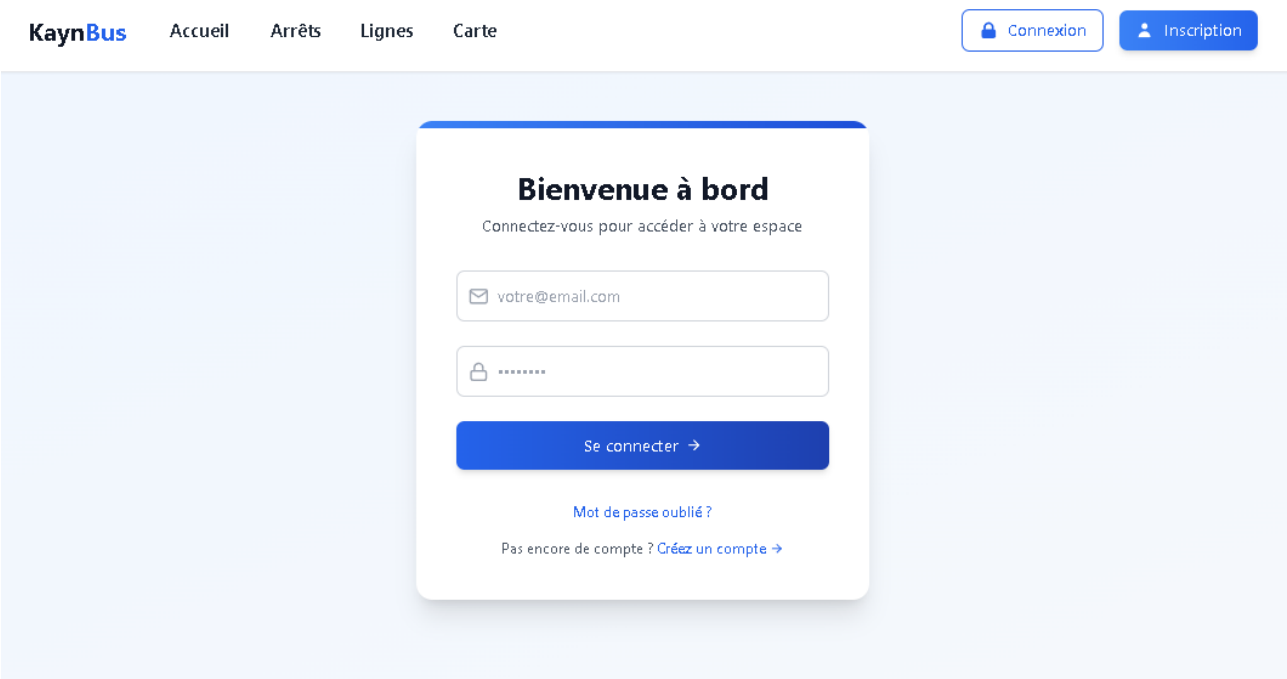


Figure 4.2 : Page de connexion (interface usager).

- **Page de mot de passe oublié** :

  Cette page s'adresse aux utilisateurs ayant déjà un compte, mais qui ont oublié leur mot de passe. Depuis la page de connexion, ils peuvent cliquer sur le lien Mot de passe oublié pour accéder à ce formulaire.

  L'utilisateur est invité à renseigner l'adresse e-mail associée à son compte. Une fois l'adresse saisie, il clique sur le bouton Envoyer le code. Le système vérifie si un compte existe pour cet e-mail. Si ce n'est pas le cas, un message d'erreur s'affiche pour signaler que l'adresse est inconnue.

  Si l'adresse est valide, un e-mail contenant un code de vérification ou un lien de réinitialisation est automatiquement envoyé à l'utilisateur. Un message de confirmation s'affiche à l'écran pour l'informer de l'envoi de cet e-mail, et l'utilisateur est ensuite redirigé vers la page de saisie du code de vérification.

  La figure 4.3 représente la page de mot de passe oublié.

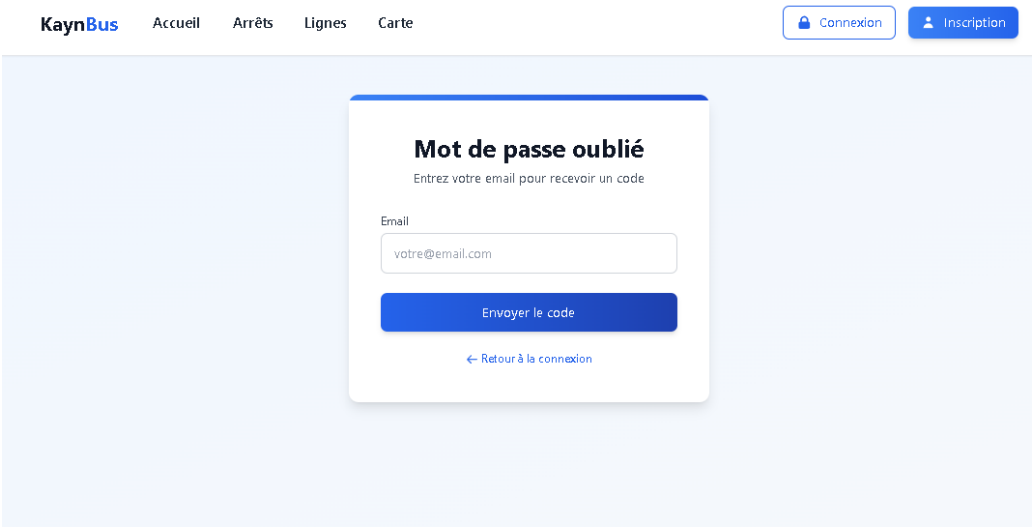


Figure 4.3 : Page de mot de passe oublié (interface usager).

- **Page de saisie du code de vérification** :

Après avoir demandé la réinitialisation de son mot de passe, l'utilisateur reçoit un code de vérification par e-mail. Il est ensuite redirigé vers cette page, où il est invité à saisir ce code à six chiffres dans le champ prévu à cet effet. Le code reçu est valable pendant une durée limitée de 10 minutes.

L'utilisateur entre le code reçu, puis clique sur le bouton Vérifier. Le système compare le code saisi avec celui qui a été envoyé et vérifie également qu'il n'a pas expiré. Si le code est incorrect ou a dépassé la durée de validité, un message d'erreur s'affiche pour inviter l'utilisateur à vérifier le code ou à en redemander un nouveau.

Si le code est valide et encore actif, la vérification est confirmée et l'utilisateur est automatiquement redirigé vers la page de création d'un nouveau mot de passe.

La figure 4.4 présente la page de saisie du code de vérification.

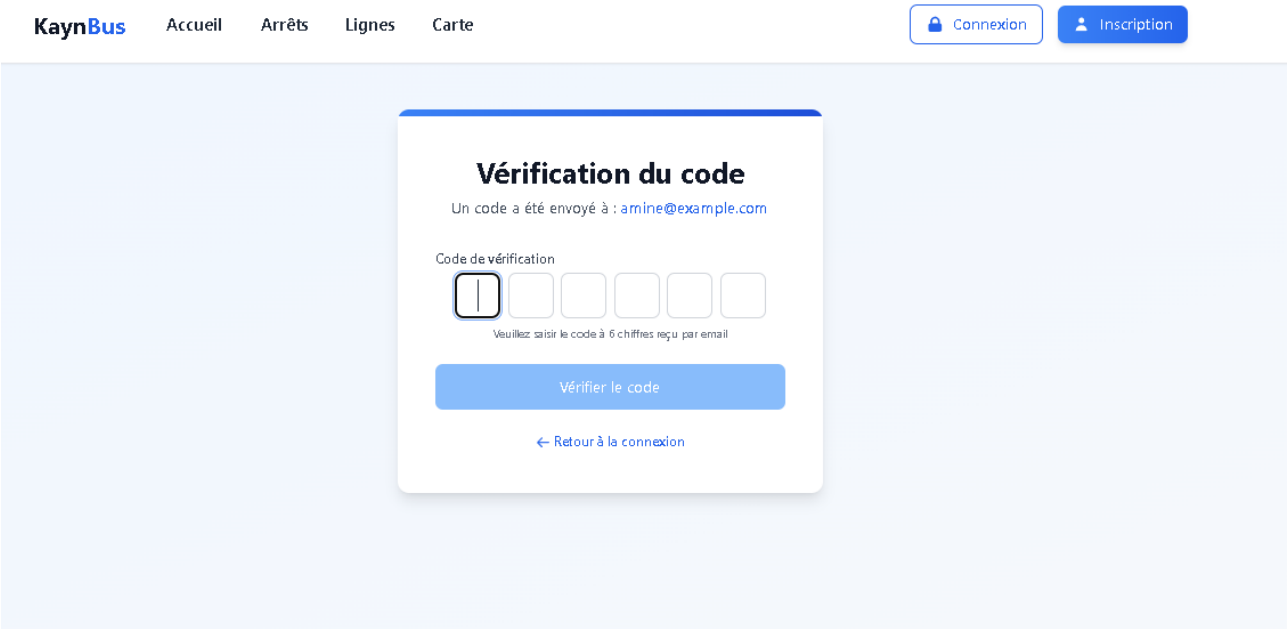


Figure 4.4 : Page de saisie du code de vérification (interface usager).

- **Page de création d'un nouveau mot de passe** :

  Après avoir validé avec succès le code de vérification, l'utilisateur est redirigé vers cette page afin de définir un nouveau mot de passe pour son compte. Le formulaire contient deux champs : un pour le nouveau mot de passe et un autre pour sa confirmation.

  L'utilisateur saisit son nouveau mot de passe, puis le confirme dans le second champ. Il clique ensuite sur le bouton Valider. Le système vérifie que les deux champs sont remplis, que les mots de passe correspondent et qu'ils respectent les critères de sécurité (par exemple : longueur minimale).

  Si une erreur est détectée (champs vides, mots de passe non identiques, ou non conformes), un message d'erreur s'affiche pour permettre à l'utilisateur de corriger sa saisie. Si tout est correct, le mot de passe est mis à jour avec succès, et un message de confirmation s'affiche brièvement.

  L'utilisateur est alors redirigé vers la page de connexion, où il peut désormais se connecter en utilisant son nouveau mot de passe.

  La figure 4.5 illustre cette page.

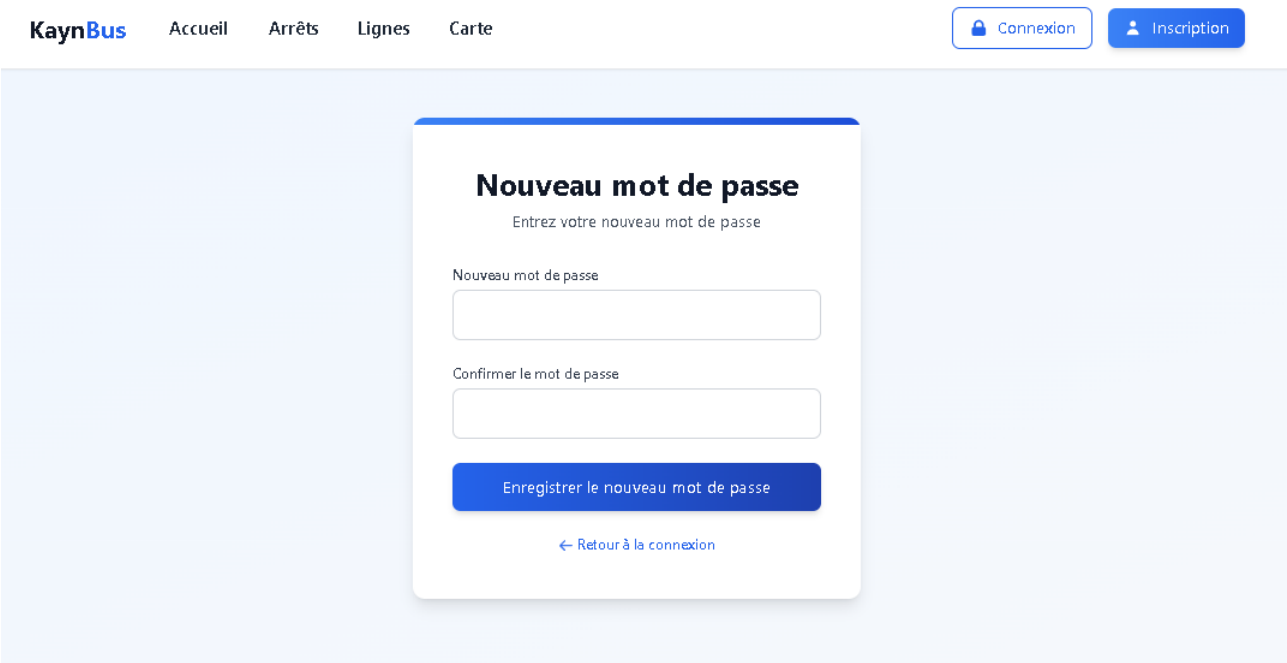


Figure 4.5 : Page de création du nouveau mot de passe (interface usager).

Interface utilisateur (usager)

L'interface usager a été conçue pour offrir une expérience fluide et intuitive. Les principales pages disponibles sont :

- **Page d'accueil** :
  Cette page présente un aperçu global de l'application et permet d'accéder rapidement aux principales fonctionnalités. Elle constitue le point d'entrée commun pour tous les utilisateurs.
  Deux cas d'affichage sont possibles :
  - **Utilisateur non connecté** : Dans ce cas, seules les fonctionnalités de base sont disponibles, comme la consultation des lignes, des arrêts, l'affichage de la carte et la localisation des bus en temps réel. Les fonctionnalités avancées telles que la gestion des favoris ou la planification de trajets sont invisibles dans la barre de navigation ;
  - **Utilisateur connecté** : une fois connecté, l'usager accède à l'ensemble des fonctionnalités avancées de l'application. Il peut gérer ses favoris et planifier ses trajets. Ces fonctionnalités apparaissent dans la barre de navigation uniquement après connexion.

  La figure 4.6 illustre la page d'accueil pour un utilisateur non connecté.



Figure 4.6 : Page d'accueil – utilisateur non connecté.

La figure 4.7 illustre la page d'accueil pour un utilisateur connecté.



Figure 4.7 : Page d'accueil – utilisateur connecté.

- **Page des arrêts**

  Lorsque l'utilisateur clique sur l'onglet Arrêts dans la barre de navigation, il est redirigé vers une page dédiée à la consultation de l'ensemble des arrêts de bus disponibles dans le réseau. Ces arrêts sont affichés sous forme de cartes individuelles, claires et informatives.

  Chaque carte d'arrêt présente plusieurs informations essentielles : le nom de l'arrêt (affiché sous le champ intitulé Coordonnées), la ligne de bus associée (indiquée sous forme de badge), la position GPS précise (latitude et longitude), ainsi qu'un segment d'itinéraire indiquant le point de départ et le terminus de la ligne correspondante.

  Deux fonctionnalités principales enrichissent l'expérience utilisateur :

  – **Ajout aux favoris** : En haut à droite de chaque carte, une icône en forme d'étoile permet à l'utilisateur d'ajouter ou retirer l'arrêt de ses favoris d'un simple clic. Cela lui permet de retrouver

rapidement les arrêts qu'il utilise fréquemment ;

– **Filtrage des arrêts** :En haut de la page, un champ de recherche et un menu déroulant permettent à l'utilisateur de filtrer les arrêts soit par nom, soit par ligne. Cela réduit instantanément l'affichage aux seuls arrêts correspondant à ses critères, facilitant la navigation.

La figure 4.8 illustre la page des arrêts avec les options d'ajout aux favoris et de filtrage.



Figure 4.8 : Page des arrêts avec filtrage et ajout aux favoris.

Page des lignes

Lorsque l'utilisateur clique sur l'onglet « Lignes » dans la barre de navigation, il est redirigé vers une page dédiée à la consultation de toutes les lignes de bus disponibles dans le réseau. Chaque ligne est présentée sous forme de carte visuelle, facilitant la lecture et la compréhension.

Chaque carte de ligne présente les informations suivantes : le numéro de la ligne, le point de départ, le terminus, ainsi qu'un aperçu du trajet affiché sur une carte intégrée. Ces éléments permettent à l'usager de repérer rapidement le parcours et les zones desservies.

Deux fonctionnalités principales améliorent l'expérience utilisateur :

– **Ajout aux favoris** : En haut à droite de chaque carte, une icône en forme d'étoile permet à l'utilisateur d'ajouter une ligne de ses favoris d'un simple clic. Cela facilite l'accès aux lignes les plus utilisées ;

– **Filtrage des lignes** : En haut de la page, deux champs de recherche permettent de filtrer les lignes selon le point de départ ou le terminus. L'affichage s'adapte instantanément aux critères saisis, ce qui simplifie la recherche ciblée.

Enfin, un bouton Voir les trajets situé au bas de chaque carte permet d'accéder à une carte détaillée affichant tous les trajets associés à la ligne sélectionnée.

La figure 4.9 illustre l'interface de la page des lignes avec les options de filtrage et d'interaction.

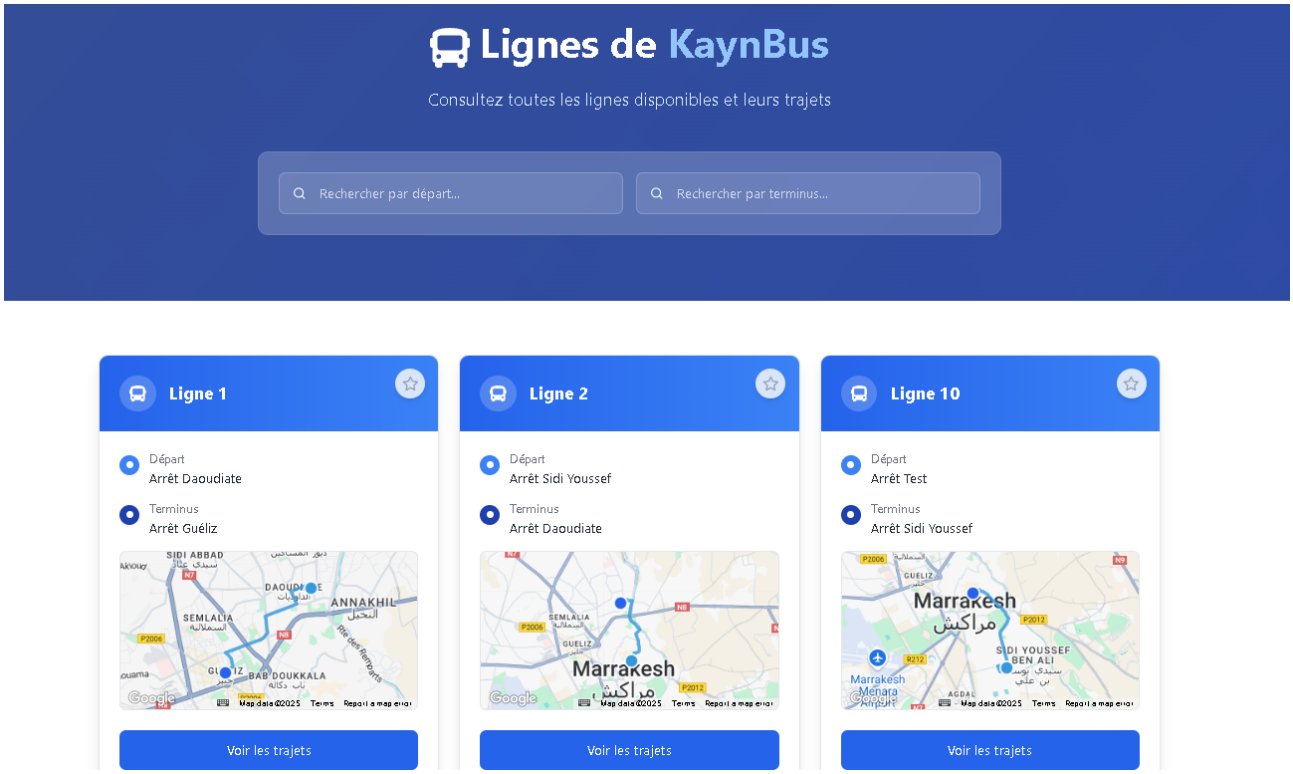


Figure 4.9 : Page des lignes avec filtrage, favoris et accès aux trajets.

- **Page de la carte du trajet** :
  Lorsqu'un utilisateur clique sur le bouton Voir trajet depuis la page des lignes, il est redirigé vers une page dédiée à la visualisation de la ligne sélectionnée sur une carte.

  Avant de cliquer sur "Localiser les bus" :

  La page affiche le nom et le numéro de la ligne en haut à gauche, accompagnés d'une section indiquant le bus affecté à cette ligne ainsi que son état (par exemple, en service). En dessous, une liste ordonnée présente tous les arrêts desservis par cette ligne, chacun accompagné de ses coordonnées GPS. Enfin, une carte Google Maps affiche visuellement le tracé complet du trajet entre le point de départ et le terminus.
  La figure 4.10 illustre l'interface de la carte du trajet avant localisation en temps réel.

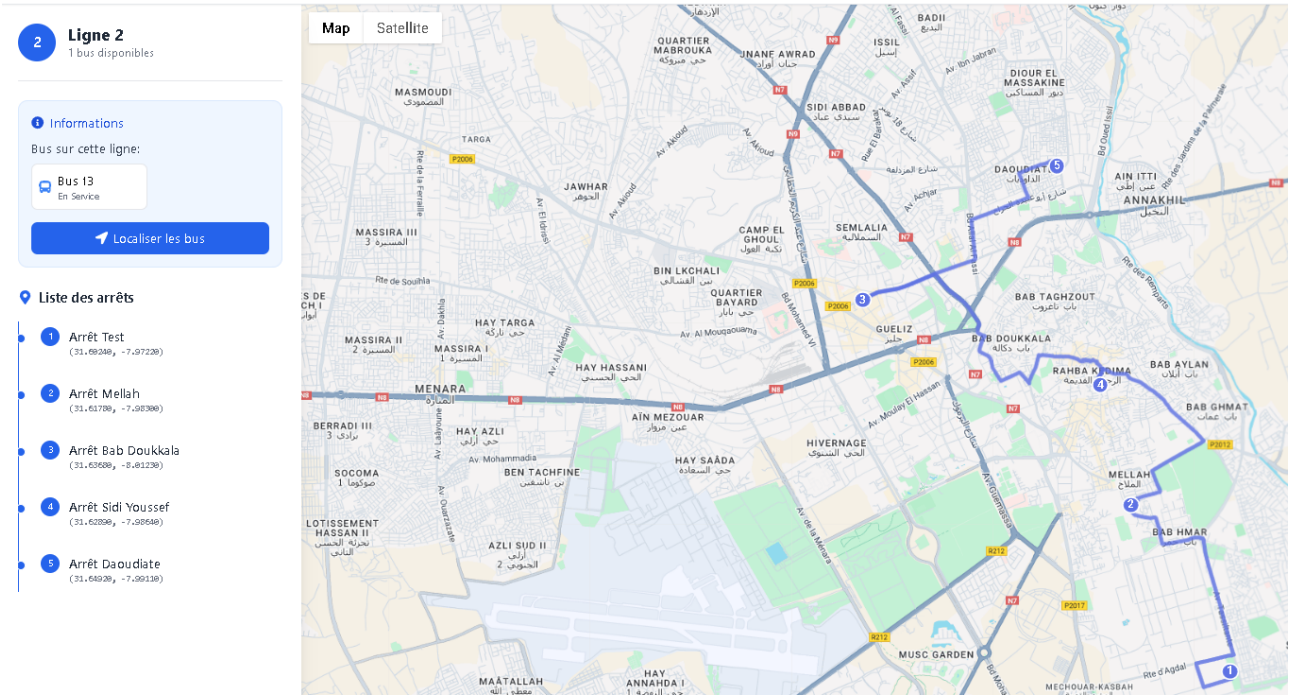


Figure 4.10 : Page du trajet – affichage de la ligne et des arrêts.

Après avoir cliqué sur "Localiser les bus" :

Lorsque l'utilisateur clique sur le bouton Localiser les bus, le système récupère en temps réel la dernière position GPS du bus affecté à la ligne. Cette position est alors affichée sur la carte à l'aide d'un marqueur distinct, accompagné d'un encadré informatif indiquant le matricule du bus, ses coordonnées géographiques (latitude et longitude), ainsi que la date et l'heure de la dernière mise à jour.

Un bouton Actualiser positions permet également de recharger les données à tout moment pour obtenir la position la plus récente.

La figure 4.11 montre la carte mise à jour avec la position du bus en temps réel.

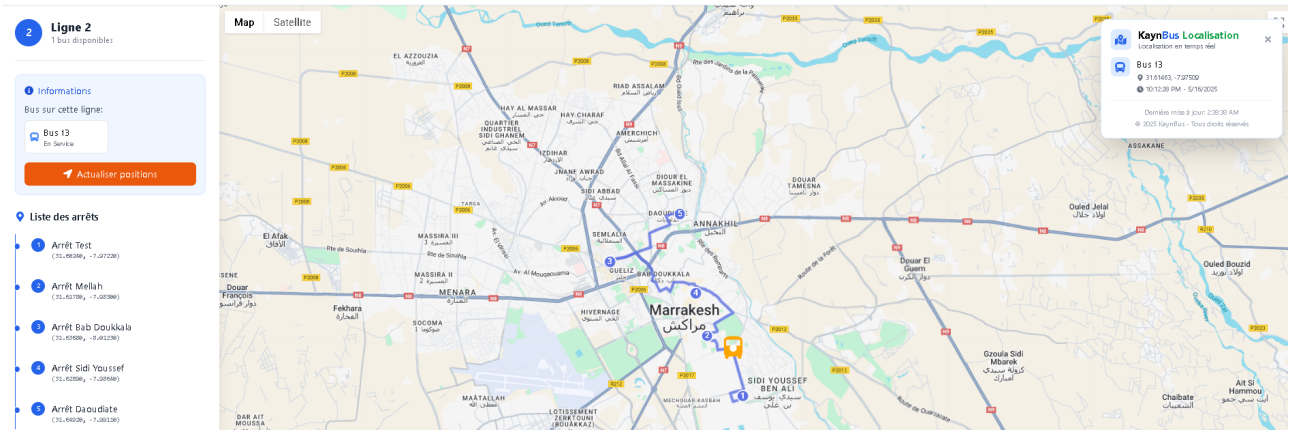


Figure 4.11 : Page du trajet – localisation du bus en temps réel.

- **Carte globale** :

  Accessible depuis la barre de navigation via l'onglet « Carte », cette page permet à l'utilisateur de visualiser l'ensemble du réseau de bus en un seul coup d'œil. Toutes les lignes disponibles sont affichées sur une carte interactive, avec des tracés de couleurs différentes pour chaque ligne, facilitant la lecture du réseau urbain.

  Plusieurs fonctionnalités interactives enrichissent cette carte :

– **Filtrage des lignes** : Un système de filtres intégré permet à l'utilisateur de sélectionner une ligne spécifique à afficher. Lorsqu'une ligne est sélectionnée dans le menu déroulant, seul son itinéraire est affiché sur la carte, avec un encadré résumant le nom de la ligne, l'arrêt de départ et le terminus ;
– **Affichage des arrêts** : L'ensemble des arrêts du réseau est également visible sur la carte. Lorsque l'utilisateur clique sur un marqueur d'arrêt, une infobulle s'affiche, indiquant le nom de l'arrêt, ses coordonnées GPS (latitude, longitude), ainsi que la ou les lignes de bus qui y passent. Cette fonctionnalité est particulièrement utile pour identifier les arrêts desservis par plusieurs lignes et faciliter les correspondances.

Ces options combinées permettent une exploration ciblée et claire du réseau, utile aussi bien pour la planification d'un trajet que pour la compréhension globale du fonctionnement du transport urbain. La figure 4.12 illustre la carte avec toutes les lignes visibles.

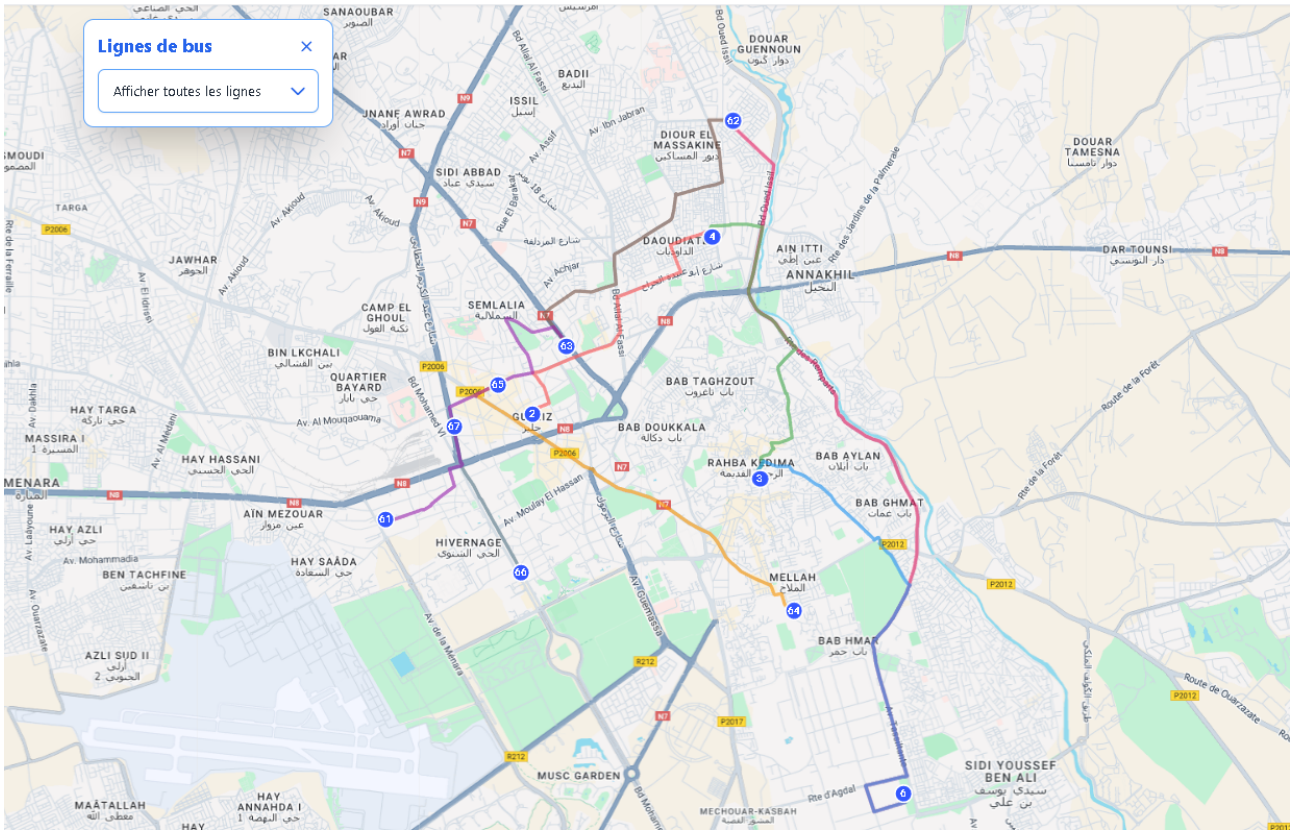


Figure 4.12 : Carte globale interactive affichant toutes les lignes de bus.

La figure 4.13 illustre la carte avec une ligne filtrée.

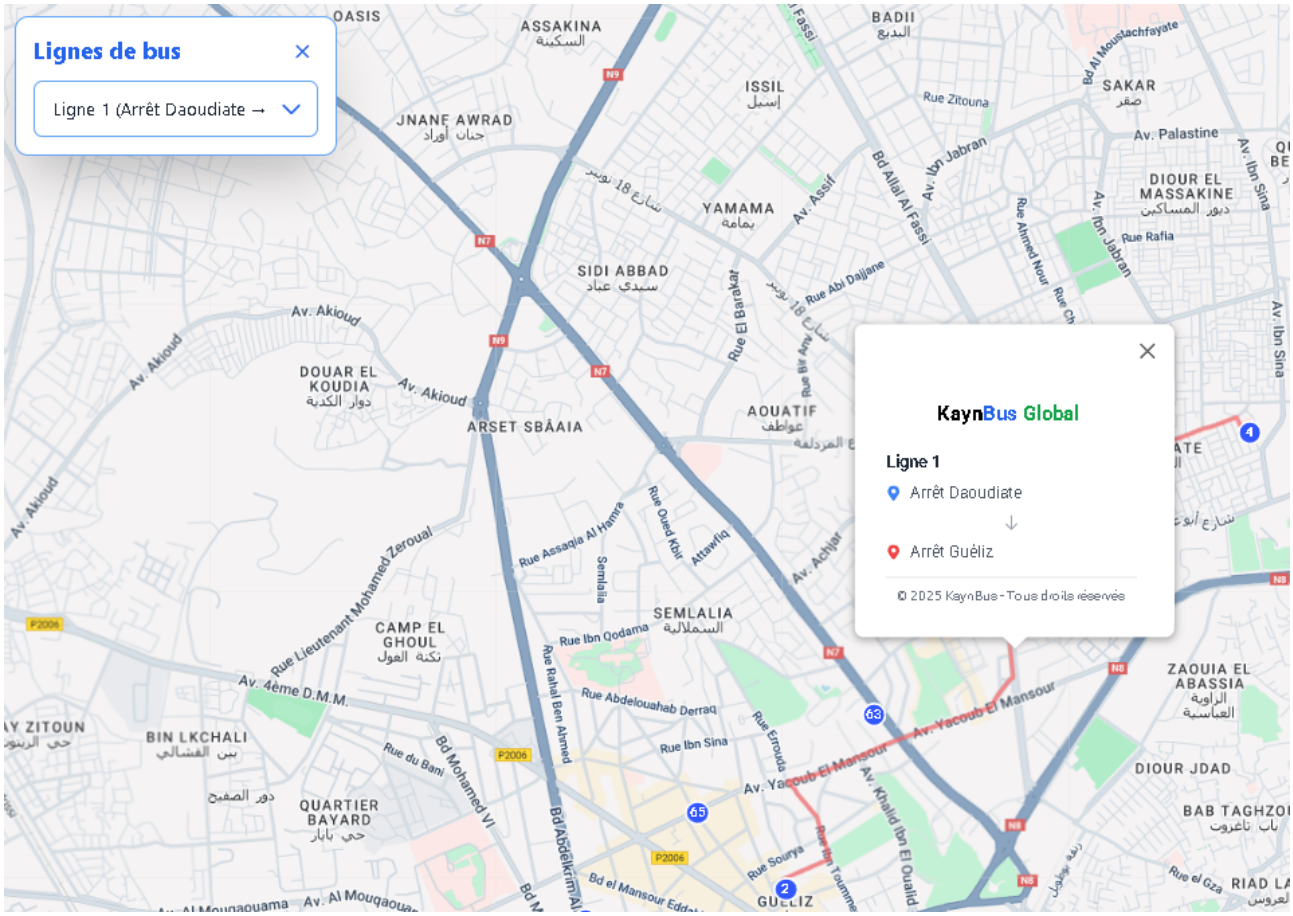


Figure 4.13 : Carte filtrée – affichage d'une seule ligne avec informations.

La figure 4.14 présente le détail affiché lorsqu'un arrêt est sélectionné.

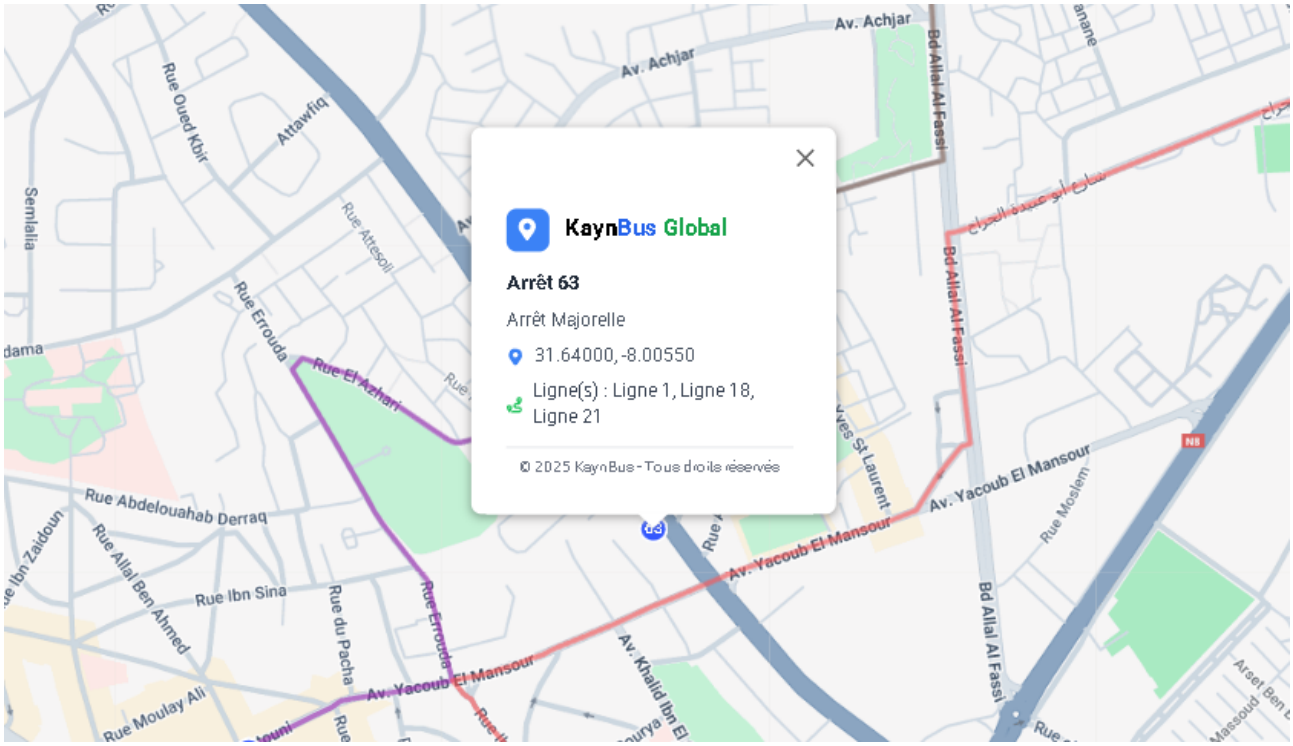


Figure 4.14 : Affichage des détails d'un arrêt sélectionné sur la carte.

- **Page des favoris** :
  Cette page est accessible uniquement si l'utilisateur est connecté. Elle devient visible dans la barre

de navigation après authentification, et permet à chaque usager de consulter et gérer ses éléments favoris, qu'il s'agisse de lignes de bus ou d'arrêts spécifiques.

Les favoris sont affichés sous forme de cartes visuelles, regroupant les informations clés :

- Pour une ligne : nom de la ligne, date d'ajout, arrêt de départ, terminus, bouton Voir les détails de la ligne ;
- Pour un arrêt : nom, position GPS, lignes qui y passent.

L'utilisateur dispose d'un système de filtrage dynamique lui permettant d'afficher uniquement les arrêts ou uniquement les lignes, selon ses besoins. Ce filtre améliore la lisibilité et facilite l'accès rapide aux favoris pertinents.

Enfin, chaque favori peut être supprimé à tout moment grâce à un bouton de suppression accessible directement sur la carte correspondante.

La figure 4.15 illustre l'interface de la page des favoris avec les filtres et les options de gestion.

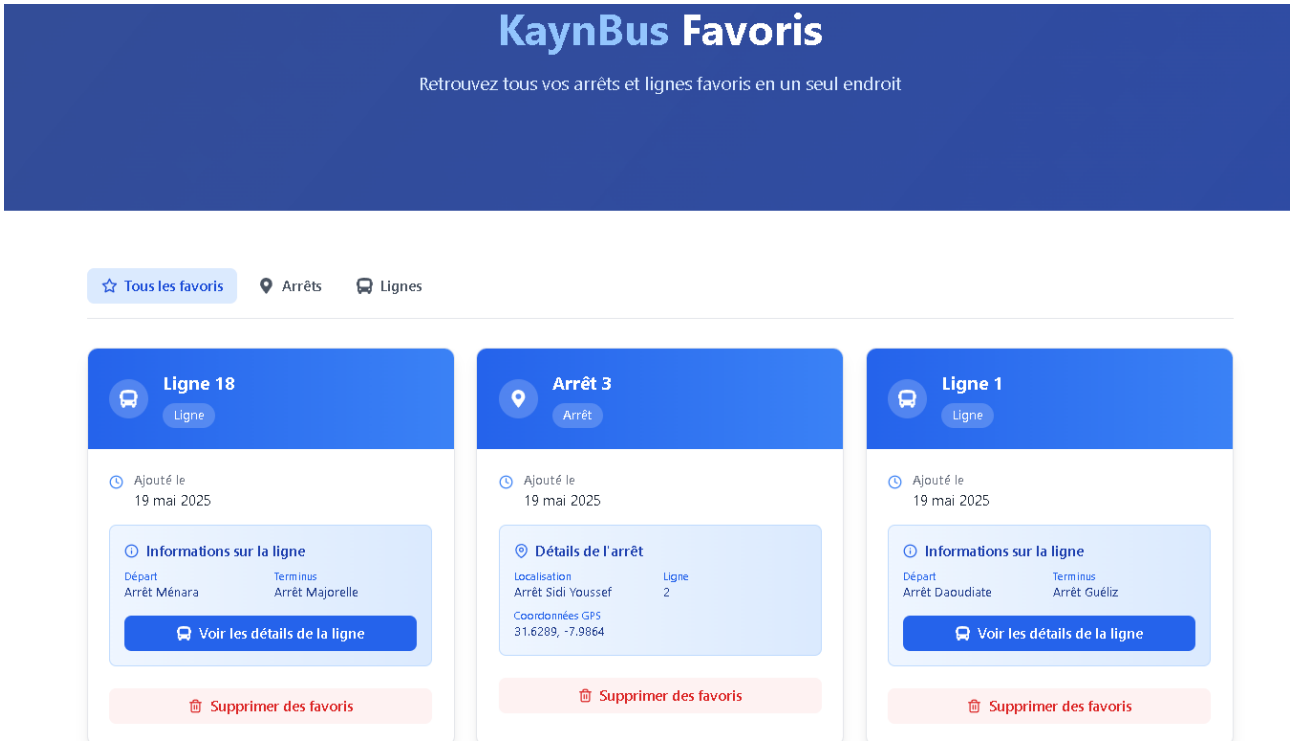


Figure 4.15 : Page des favoris avec filtrage par type et options de gestion.

- **Page de planification de trajet** :

  Accessible depuis l'onglet Planification dans la barre de navigation, cette page permet à l'utilisateur de rechercher le meilleur itinéraire en bus entre deux arrêts. L'interface propose un seul mode d'entrée :

  - **Saisie manuelle des arrêts** : l'utilisateur sélectionne un arrêt de départ et un arrêt d'arrivée parmi les arrêts disponibles dans la liste.

  Après avoir cliqué sur le bouton Trouver mon trajet, le système affiche un itinéraire optimal sous forme de carte enrichie, accompagné de plusieurs éléments :

  - Le point de départ et le point d'arrivée ;
  - Le nom de la ligne à emprunter et le bus affecté ;
  - Une carte affichant la position en temps réel du bus correspondant ;
  - La durée estimée du trajet, la distance totale et le prochain passage.

  L'interface est pensée pour être claire, interactive et adaptée à tous les usagers, facilitant la planification des déplacements en bus, en tenant compte à la fois de l'horaire et des données en temps réel.

  La figure 4.16 illustre le formulaire de recherche.

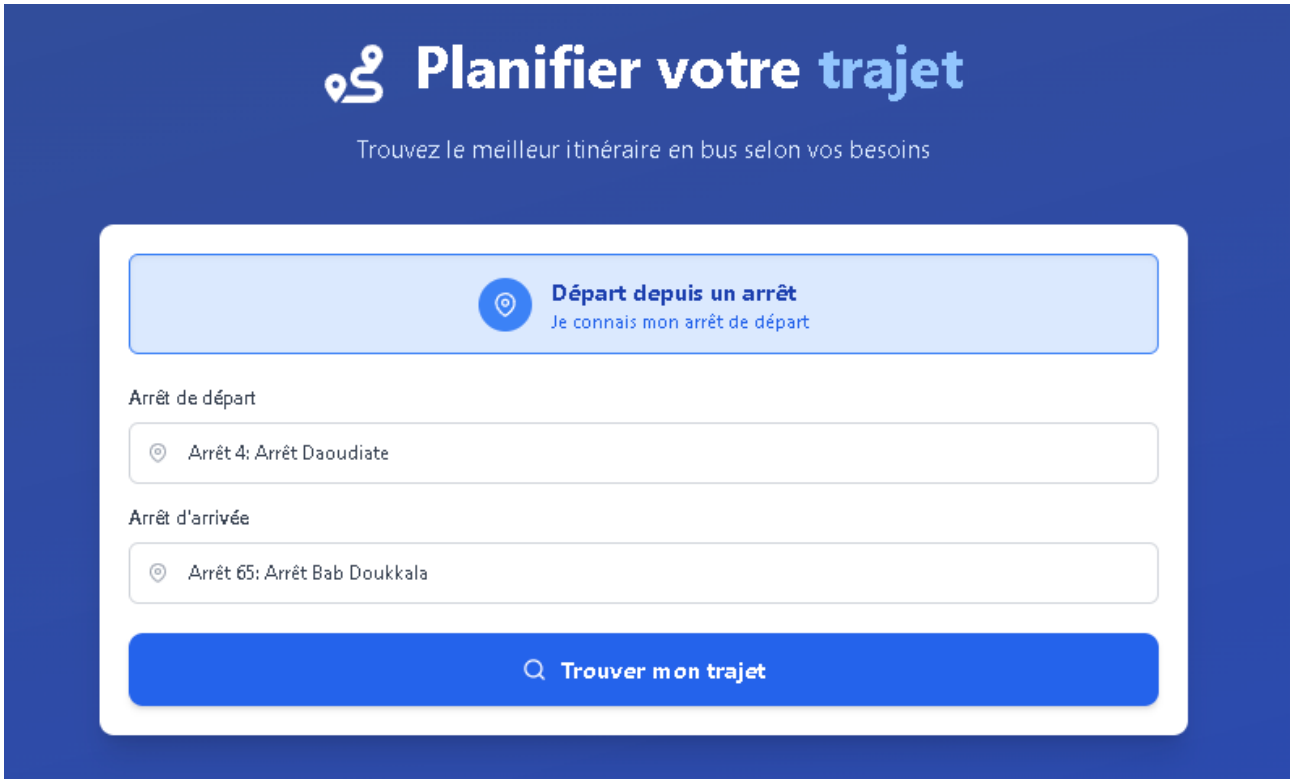


Figure 4.16 : Formulaire de planification de trajet.

La figure 4.17 illustre le détail de l'itinéraire et la carte globale de la position de bus.

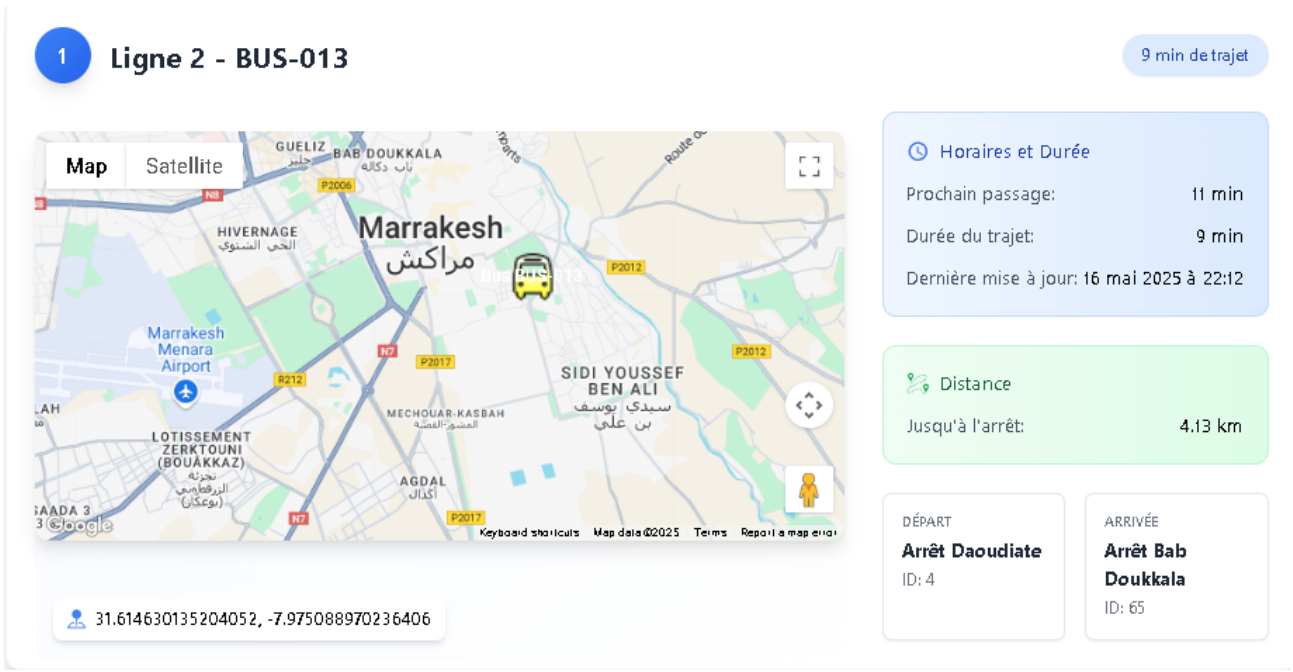


Figure 4.17 : Détail de l'itinéraire proposé avec position du bus.

Interface administrateur

L'interface administrateur permet aux gestionnaires du réseau de superviser et de modifier les données du système.

- **Dashboard** :

Une fois connecté avec un compte administrateur, l'utilisateur est automatiquement redirigé vers son tableau de bord. Ce dashboard centralise une vue d'ensemble synthétique de l'état du réseau de transport. Il affiche les indicateurs clés suivants : le nombre total de lignes enregistrées, le nombre d'arrêts de bus disponibles, le nombre de trajets planifiés, ainsi que le nombre de bus actuellement gérés dans le système. Ces statistiques permettent à l'administrateur d'évaluer rapidement la couverture et le bon fonctionnement du réseau.

La figure 4.18 montre l'interface du dashboard avec les indicateurs et graphiques.



Figure 4.18 : Dashboard de l'administrateur avec indicateurs et statistiques.

- **Page de gestion du réseau** :
  Accessible depuis le tableau de bord administrateur, cette page permet la gestion complète de l'ensemble des entités du réseau : les lignes de bus, les arrêts, les bus et les trajets.

  Chaque section (lignes, arrêts, bus, trajets) dispose d'une interface dédiée, avec des formulaires clairs pour ajouter de nouveaux éléments, ainsi que des tableaux récapitulatifs listant les entités existantes.

  L'administrateur peut effectuer les opérations suivantes :

  – **Ajouter** une ligne en choisissant un arrêt de départ et un arrêt terminus ;
  – **Ajouter** un arrêt en renseignant ses coordonnées et la ligne associée ;
  – **Ajouter** un bus avec son matricule, son état (en service ou non) et la ligne assignée ;
  – **Planifier un trajet** en définissant les heures, les arrêts, la ligne et le bus.

  Chaque action d'ajout ou de suppression est accompagnée d'un message de confirmation, garantissant que l'administrateur valide consciemment les modifications importantes, notamment pour les suppressions critiques impactant les trajets liés. Par exemple, avant de supprimer une ligne, un encadré d'alerte s'affiche pour avertir que tous les trajets associés seront également supprimés.

  La figure 4.19 illustre un exemple d'interface de gestion (gestion des trajets).

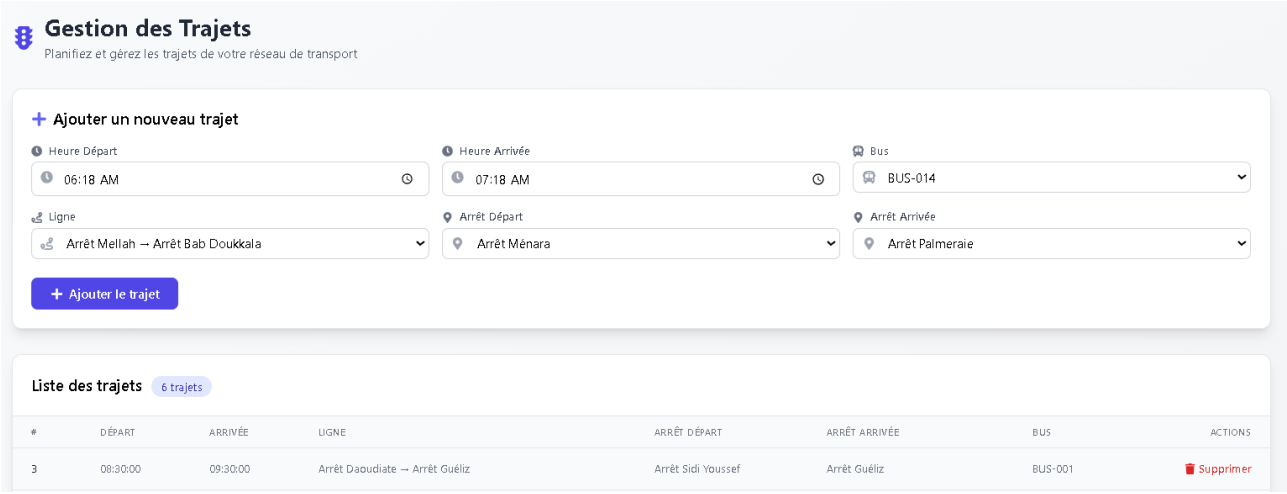


Figure 4.19 : Page de gestion des trajets.

La figure 4.20 illustre le message de confirmation lors de l'ajout.

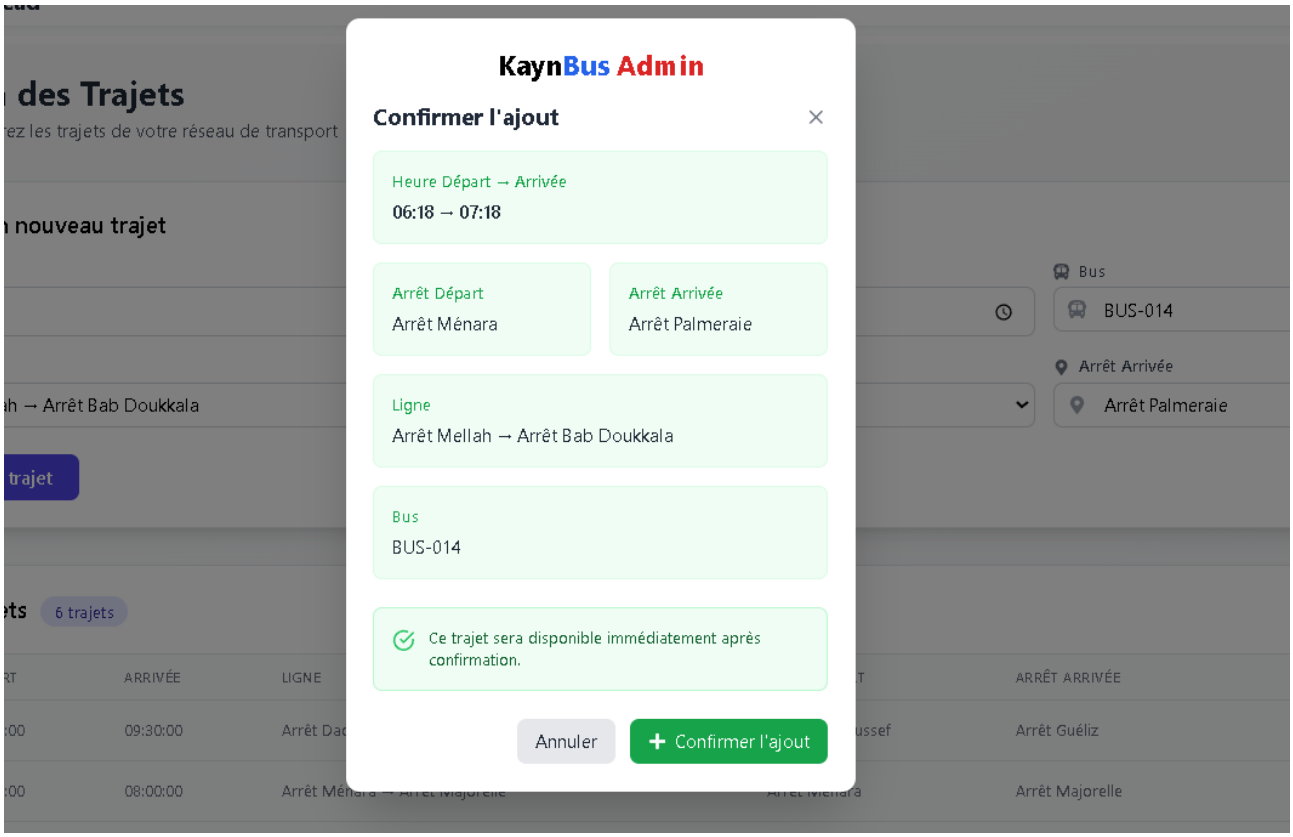


Figure 4.20 : Message de confirmation lors de l'ajout d'une ligne.

La figure 4.21 illustre le message de confirmation avant la suppression.

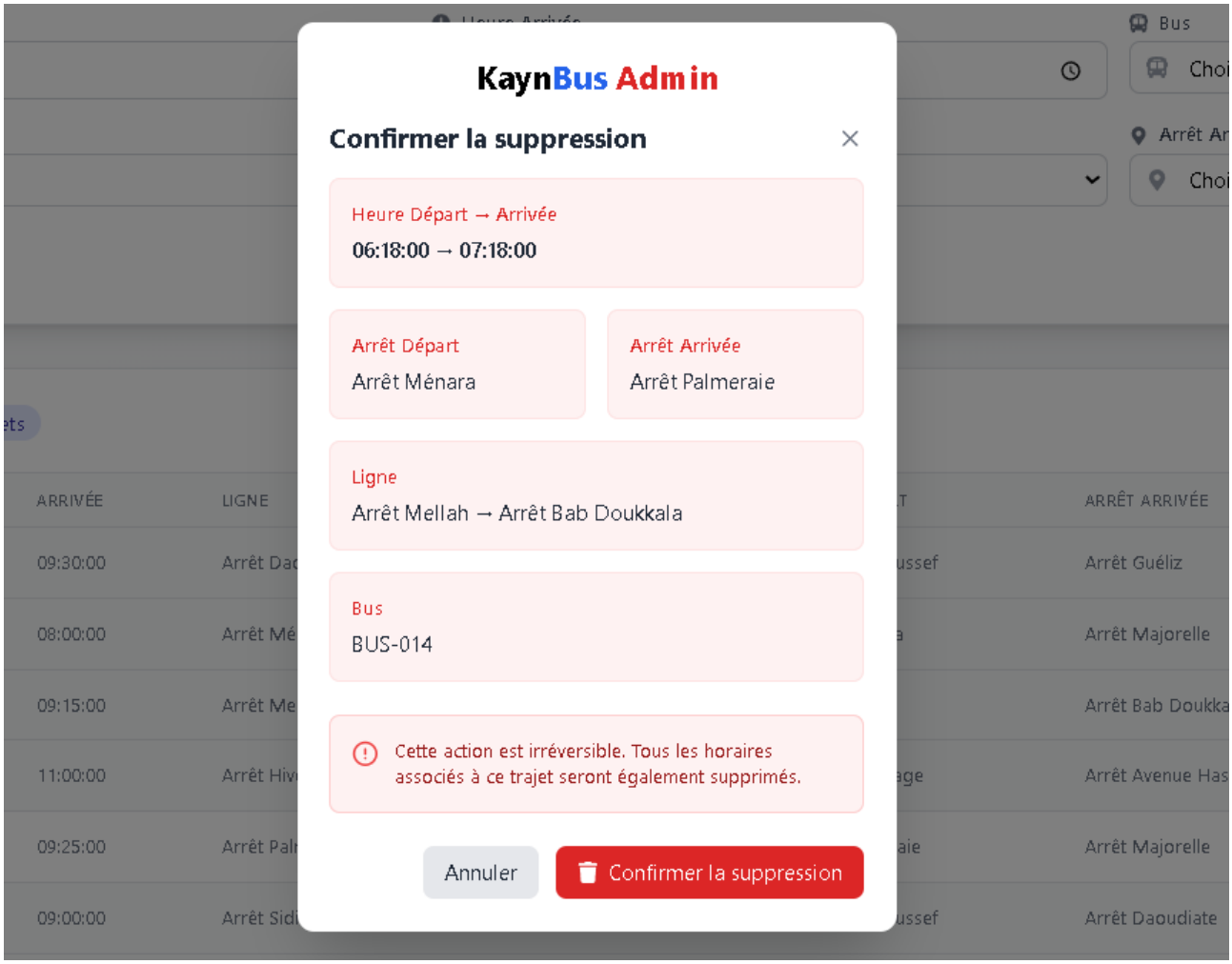


Figure 4.21 : Message de confirmation avant la suppression d'une ligne.

- **Gestion des utilisateurs** :
  Cette section, accessible via l'onglet Utilisateurs dans la barre de navigation, permet à l'administrateur de visualiser et d'analyser l'ensemble des utilisateurs enregistrés sur la plateforme, qu'il s'agisse des usagers ou des administrateurs. Elle offre une vue d'ensemble à travers des indicateurs synthétiques :
  - Le nombre total d'utilisateurs inscrits sur le système ;
  - La répartition entre usagers et administrateurs, exprimée en valeurs absolues et en pourcentage ;
  - Deux types de visualisation : un diagramme circulaire et un graphique à barres illustrant la répartition des rôles.

  Un tableau dynamique permet également de lister les utilisateurs avec leurs informations principales : identifiant, nom, adresse e-mail et rôle.
  La figure 4.22 présente l'interface de gestion des utilisateurs avec les statistiques générales.

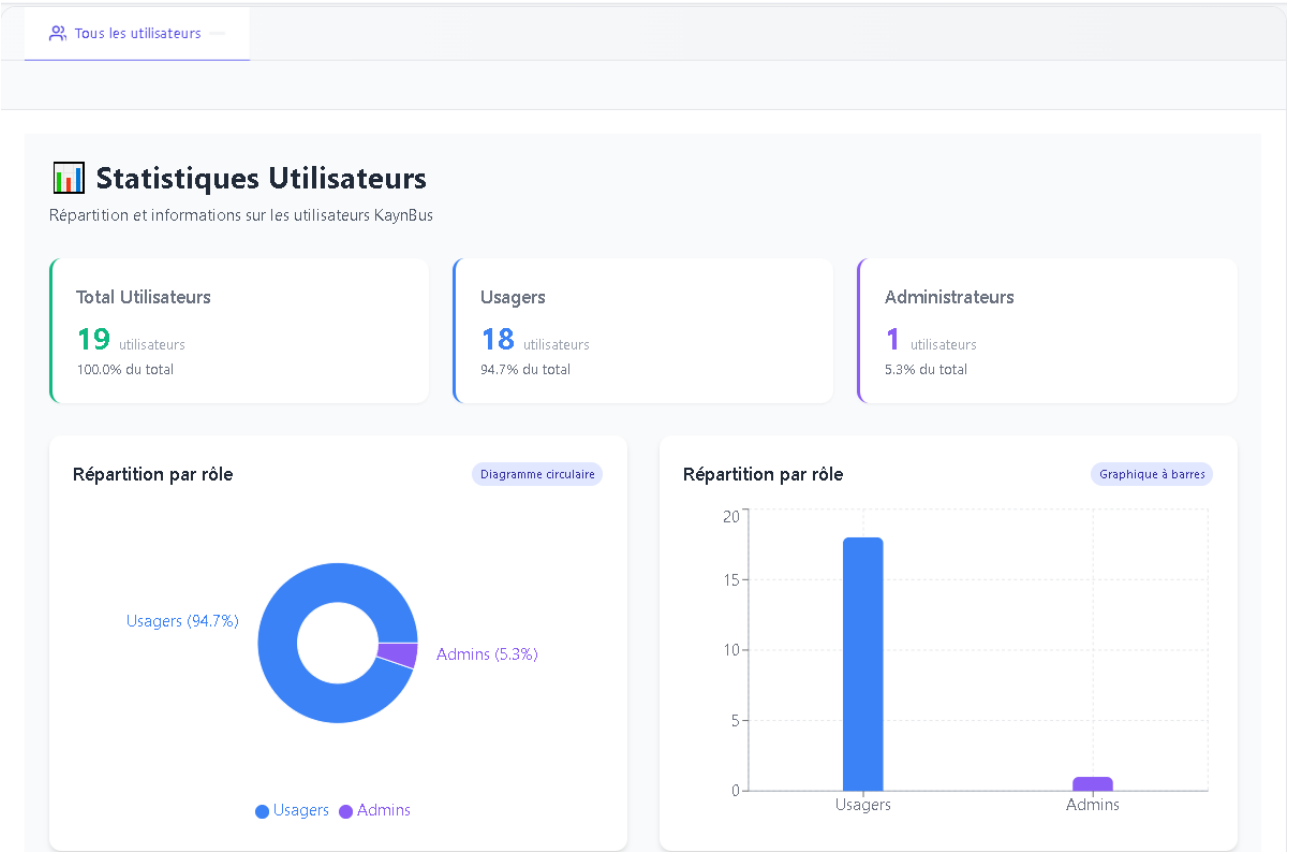
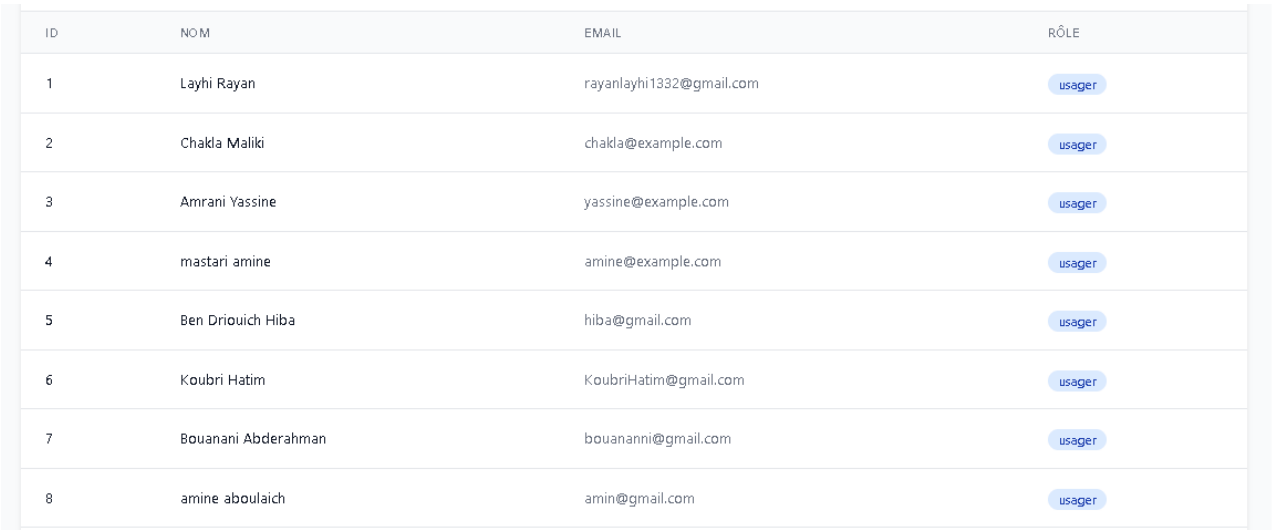


Figure 4.22 : Page de gestion des utilisateurs statistiques.

La figure 4.23 présente l'interface de gestion des utilisateurs avec la liste correspondante.



ID	NOM	EMAIL	RÔLE
1	Layhi Rayan	rayanlayhi1332@gmail.com	usager
2	Chakla Maliki	chakla@example.com	usager
3	Amrani Yassine	yassine@example.com	usager
4	mastari amine	amine@example.com	usager
5	Ben Driouich Hiba	hiba@gmail.com	usager
6	Koubri Hatim	KoubriHatim@gmail.com	usager
7	Bouanani Abderahman	bouananni@gmail.com	usager
8	amine aboulaich	amin@gmail.com	usager

Figure 4.23 : Page de gestion des utilisateurs liste.

- **Page Carte (interface administrateur)** :
  Accessible depuis la barre de navigation, cette page offre à l'administrateur une vue globale et dynamique du réseau de transport. Elle intègre une carte interactive sur laquelle toutes les lignes de bus sont affichées avec des tracés de couleurs distinctes, facilitant ainsi leur identification.
  L'administrateur dispose de plusieurs fonctionnalités essentielles :
  - **Filtrage par ligne** : Un menu déroulant permet d'afficher une seule ligne à la fois pour se concentrer sur un itinéraire précis ;
  - **Consultation des arrêts** : En cliquant sur un arrêt, l'administrateur peut visualiser ses coordonnées GPS, son nom, ainsi que les lignes auxquelles il appartient ;

– **Suivi en temps réel des bus** : Il est possible de sélectionner un bus spécifique dans le menu de gauche, puis de cliquer sur Localiser pour afficher sa position actuelle sur la carte, accompagnée d'un encadré informatif contenant son matricule, ses coordonnées, la date et l'heure de la dernière mise à jour.

La figure 4.24 illustre la page Carte de l'administrateur avec les fonctionnalités de filtrage et de localisation.

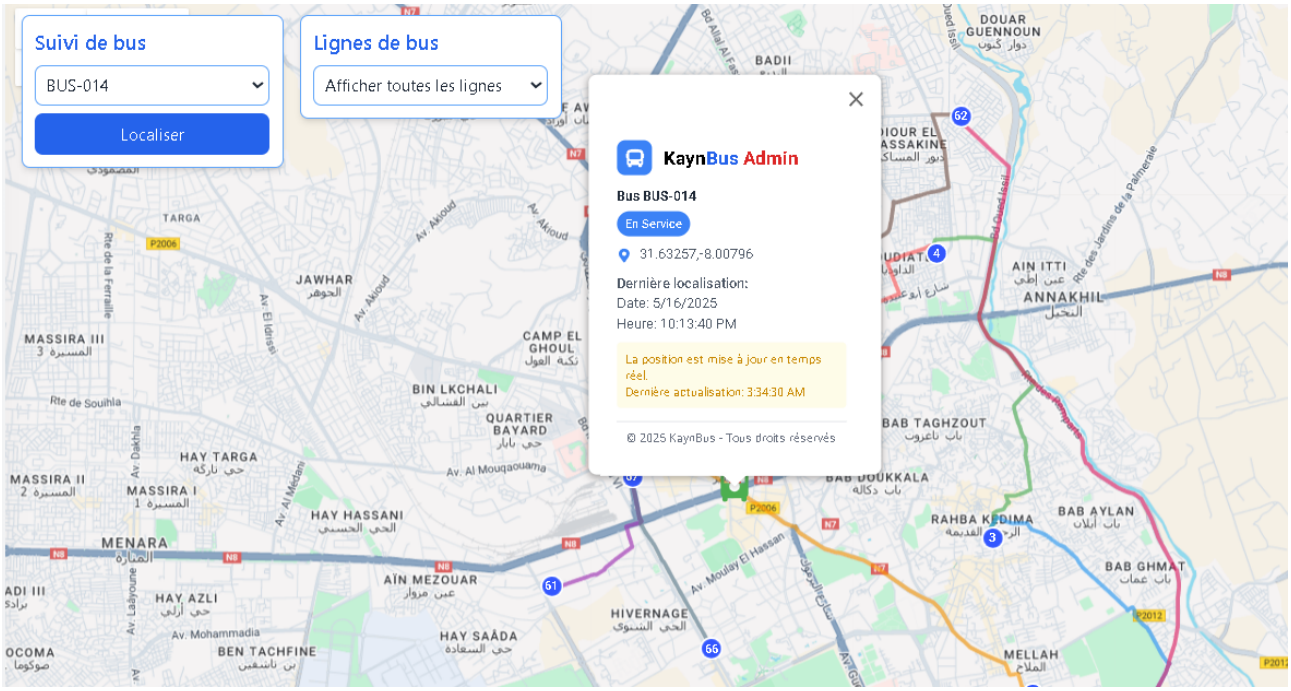


Figure 4.24 : Page Carte de l'administrateur : filtrage des lignes et localisation des bus.

- **Page de rapports** :
  Cette page permet à l'administrateur de consulter des statistiques détaillées sur l'ensemble du réseau. Elle regroupe les données clés relatives aux entités suivantes :

  – Les arrêts de bus : nombre total, répartition par ligne, arrêts les plus fréquentés, etc. ;
  – Les trajets : durée moyenne, trajets actifs, distribution par ligne ;
  – Les lignes de bus : lignes actives ;
  – Les bus : nombre total, état (en service, hors service), répartition par ligne.

  Les informations sont présentées de manière visuelle sous forme de graphiques (camemberts, barres, histogrammes) et de tableaux dynamiques, offrant ainsi une lecture claire et synthétique des indicateurs du système.
  Une fonctionnalité d'exportation est également intégrée, permettant de télécharger les rapports sous format PDF ou Excel, facilitant ainsi le suivi et le partage des données.
  La figure 4.25 illustre l'interface des rapports (rapport des lignes) avec les représentations graphiques.

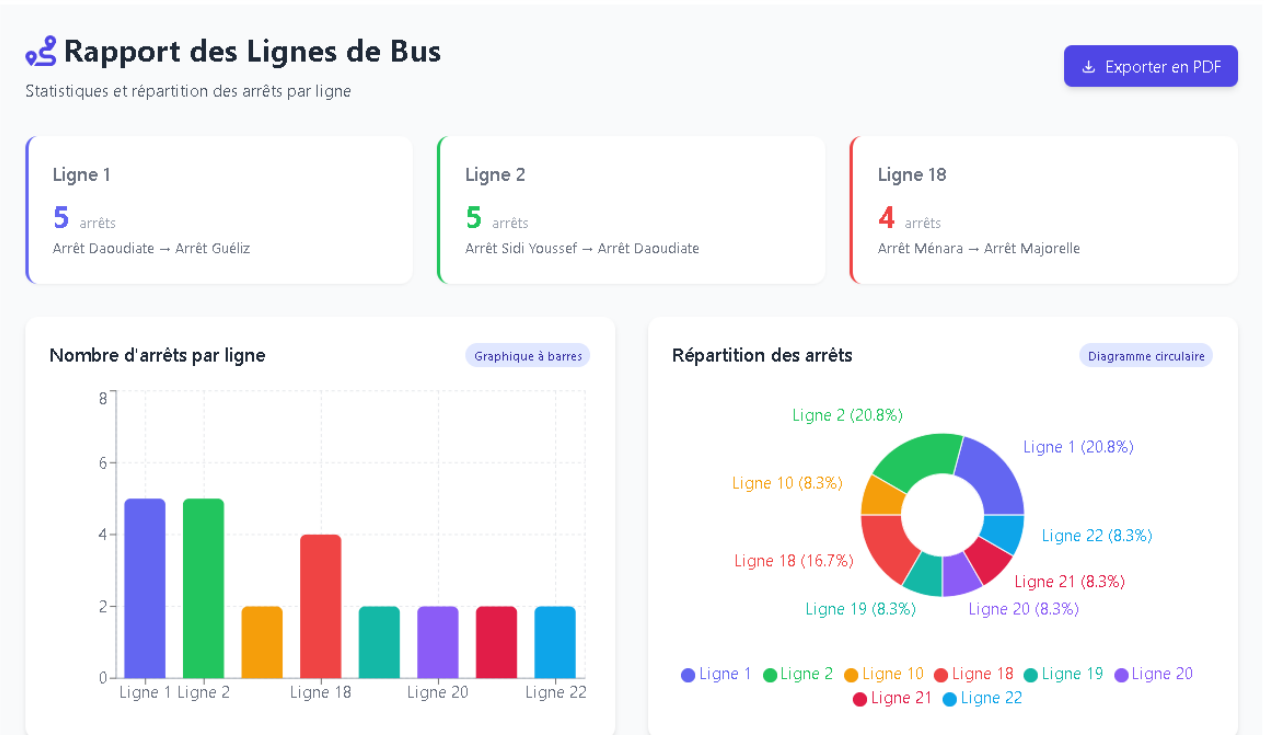


Figure 4.25 : Page de rapports avec statistiques détaillées.

La figure 4.26 illustre l'interface des rapports (rapport des lignes) avec la liste des lignes.

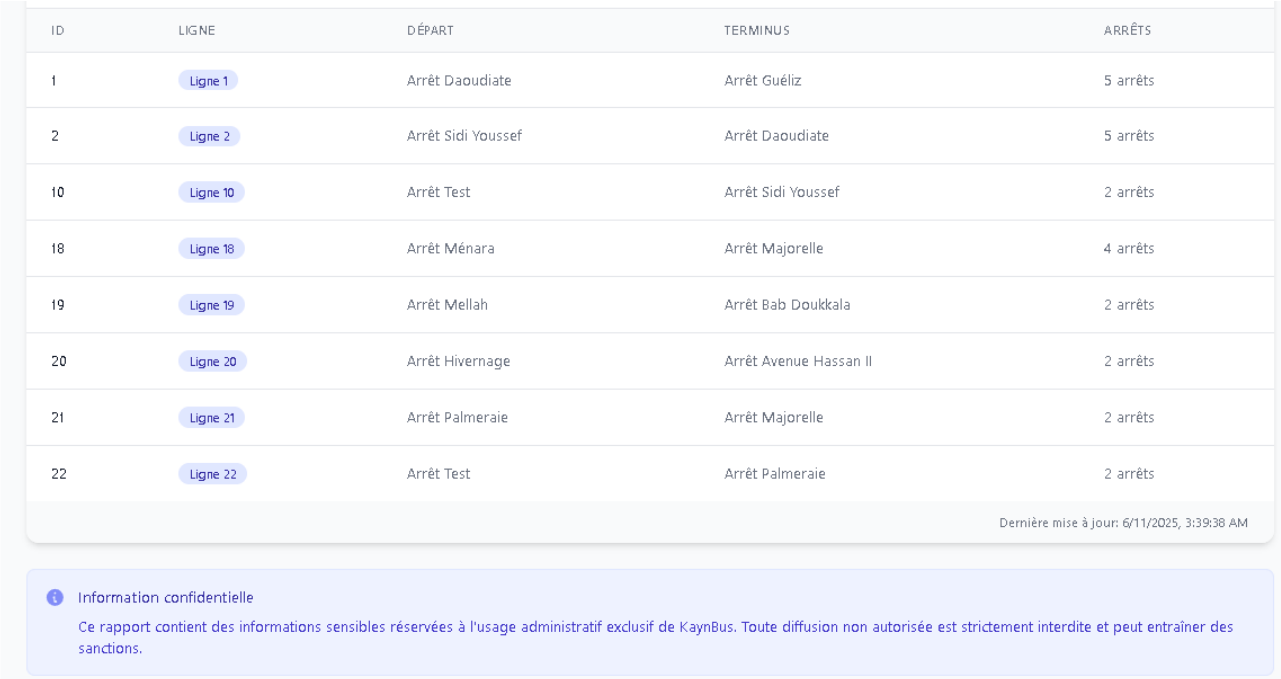


Figure 4.26 : Page de rapports avec liste détaillée des lignes.

4.3.3 Implémentation de l'application mobile

Algorithme de localisation des bus

Le fonctionnement de l'application de localisation des bus repose sur un algorithme simple mais efficace, conçu pour assurer la remontée des positions GPS en temps réel. Ce processus est implémenté dans l'application mobile Android, développée en Java.

Le principe général est le suivant :

- À l'ouverture de l'application, l'utilisateur (chauffeur, administrateur ou dispositif embarqué) autorise l'accès à la localisation de l'appareil ;
- Le service de localisation de l'appareil Android utilise les API de Google Location Services pour obtenir régulièrement la position GPS (latitude, longitude) ;
- Un timer (ou un service en arrière-plan) est déclenché pour envoyer périodiquement la position au serveur via une requête HTTP POST vers l'API REST (/LoadPositions.php) ;
- Chaque envoi contient l'identifiant du bus, les coordonnées GPS, et un horodatage.
- Côté serveur, la position est enregistrée dans la base de données MySQL (table position) ;
- Côté application web (React), les positions sont régulièrement interrogées via des requêtes GET, permettant de mettre à jour l'affichage en temps réel sur la carte (via l'API Google Maps).

Fréquence de mise à jour : Dans cette version du système, la fréquence d'envoi est réglée à environ une position toutes les 5 secondes, ce qui permet un suivi fluide et suffisamment précis des déplacements des bus.

Traitement des erreurs : L'algorithme inclut des vérifications pour garantir que :

- Les coordonnées GPS sont valides (non nulles et à jour) ;
- La connectivité réseau est disponible avant l'envoi de la position ;
- En cas d'échec de l'envoi, le système réessaie automatiquement après un délai défini.

Ce mécanisme garantit la robustesse de la transmission des données de géolocalisation, élément central du bon fonctionnement du système de suivi des bus.

4.4 Stratégie de tests

Afin d'assurer la qualité, la fiabilité et la solidité du système, une approche de tests organisée a été instaurée, englobant divers niveaux de vérification.

- **Tests unitaires :** Les tests unitaires ont été exécutés manuellement et ont porté sur les fonctions les plus importantes du back-end écrites en PHP. Chaque élément du système (fonctions de traitement de données, connexion à la base de données, interaction avec l'API, etc.) a été testé indépendamment pour vérifier qu'il se comporte comme attendu. L'absence de framework de test standard tel que PHPUnit a été compensée par des scripts locaux exécutant des scénarios de tests distincts ;
- **Tests d'intégration :** Les contrôles d'intégration ont validé l'échange correct d'informations entre les différentes couches du système : React pour l'interface utilisateur, Axios pour le transfert de données, et PHP/MySQL côté serveur. Divers scénarios ont été testés : soumission de formulaires, récupération de données depuis la base, gestion des erreurs serveur. Cette approche a permis de vérifier que les modules interagissent correctement et de détecter rapidement d'éventuelles anomalies ;
- **Tests système :** Des essais de système ont été réalisés pour valider les performances globales de l'application Web en environnement local (XAMPP). Les tests ont porté sur le chargement des interfaces utilisateur, la navigation entre les pages, l'interaction avec les API, ainsi que la persistance des données dans MySQL. La compatibilité a également été vérifiée sur plusieurs navigateurs ;
- **Tests de l'application mobile (Android) :** Pour l'application mobile développée en Java sous Android Studio, des tests manuels ont été réalisés sur différents appareils Android (émulateur et smartphone réel). Ces tests ont vérifié :
  - le bon fonctionnement de la géolocalisation et l'accès aux services de Google Maps ;
  - l'envoi correct des positions GPS vers le serveur via les API REST ;
  - l'affichage en temps réel des informations sur la carte,
- **Outils de test et plan de validation :**
  - **Postman** a été utilisé pour tester séparément les API PHP, en simulant des requêtes HTTP (GET, POST, etc.) et en vérifiant la conformité des réponses ;
  - Axios a permis l'intégration des appels API côté client dans l'application Web, les tests ayant validé la gestion des erreurs, des statuts et des réponses dynamiques ;

- Sur Android Studio, le logcat et le débogueur intégré ont été employés pour suivre les requêtes réseau, le traitement des positions GPS, et le comportement de l'application sur différents cycles de vie ;
- Un plan de validation fonctionnel a été établi à partir des cas d'usage définis en phase de conception. Chaque fonctionnalité (Web et mobile) a été testée selon des scénarios réalistes, avec un suivi des résultats. L'ensemble des tests a confirmé la stabilité du système dans les conditions prévues.

4.5 Indicateurs de performance

Bien que ce chapitre ait détaillé les interfaces et les fonctionnalités clés de l'application, il est également important d'évaluer les performances techniques du système. Quelques indicateurs de base ont été mesurés lors des tests système pour apprécier la réactivité et la fluidité de l'application.

En environnement local (XAMPP, machine de développement standard), les temps de réponse moyens observés pour les principales requêtes API étaient les suivants :

- **Chargement initial de la carte interactive** avec les positions des bus : entre 1, 2 et 1, 5 secondes ;
- **Requête pour la récupération des lignes de bus** : environ 500 ms ;
- **Requête pour la récupération des arrêts associés à une ligne** : environ 400 ms ;
- **Actualisation des positions GPS des bus en temps réel** : rafraîchissement toutes les 5 secondes, avec un temps de traitement par cycle de l'ordre de 300 à 500 ms.

Le système a été testé avec un volume simulé de 10 à 20 utilisateurs simultanés sans dégradation notable des performances. Toutefois, ces résultats doivent être relativisés : le déploiement uniquement en local (sans hébergement distant ni infrastructure optimisée) ne permet pas de garantir les mêmes performances en production réelle.

4.6 Conclusion

Pour clore ce chapitre, il est important de faire un bilan sur l'ensemble des décisions techniques et de réalisation qui ont été prises :

- **Synthèse des choix techniques :** Le projet a été réalisé à l'aide de technologies éprouvées, notamment PHP pour le back-end, React.js pour le front-end, et MySQL pour la base de données. Le développement de l'application mobile a été effectué en Java sous Android Studio, en s'appuyant sur les API de géolocalisation de Google Maps. Les technologies complémentaires, telles que Axios, Postman, XAMPP, Visual Studio Code, NetBeans et Android Studio, ont également facilité la mise en œuvre du projet. Ces choix ont été motivés par leur facilité d'accès, leur maturité, et leur adéquation aux exigences fonctionnelles et techniques du système ;
- **Retour sur l'architecture de déploiement et l'implémentation :** L'environnement de déploiement local, configuré à l'aide de XAMPP, a permis de simuler un environnement serveur stable pour l'application.la structure modulaire du code, associée à une gestion claire des couches (front-end, back-end, API), contribue à assurer la stabilité et la pérennité du système ;
- **Perspectives et défis :** Plusieurs limites ont été relevées dans le système existant, telles que l'absence d'un pipeline CI/CD et la gestion manuelle des opérations de déploiement.par ailleurs, l'intégration de certains modules dans l'environnement mobile demeure particulièrement complexe. afin d'améliorer la situation, il conviendrait de mettre en place une automatisation des tests et du déploiement, tout en optimisant le code pour en accroître la "capacité de montée en charge ;
- **Transition vers le chapitre suivant :** Les choix techniques présentés ici constituent une base solide pour la réalisation concrète de l'application.le prochain chapitre détaillera l'implémentation pratique du système, à travers les interfaces utilisateur, les fonctionnalités implémentées et les tests finaux.

Ce chapitre constitue ainsi un socle technique solide, garantissant que les choix de développement sont cohérents avec les objectifs du projet et prêts à être mis en œuvre lors de la phase de réalisation.

Conclusion générale

Ce projet a permis de mobiliser un ensemble de compétences techniques, méthodologiques et organisationnelles dans le but de concevoir et mettre en œuvre une solution numérique fonctionnelle pour la géolocalisation en temps réel des bus. La conclusion générale permet de faire le point sur les enseignements tirés, les résultats obtenus, les limites rencontrées, ainsi que les perspectives d'évolution.

L'objectif principal du projet est de concevoir un système de géolocalisation des bus avec gestion des lignes, des arrêts, des trajets, et un affichage dynamique sur carte interactive. Ces objectifs visent à améliorer la gestion du réseau urbain et à offrir aux usagers une visibilité en temps réel. Malgré certaines simplifications liées au cadre académique, comme le déploiement local, les principaux objectifs techniques et fonctionnels ont été atteints. Le projet constitue ainsi une simulation fonctionnelle proche d'une application réelle.

Le projet repose sur une architecture hybride monolithique combinant un backend en PHP avec une base de données MySQL, un frontend en React.js pour l'interface web, et une application mobile développée sous Android pour la géolocalisation des bus. La communication entre le front-end et le back-end a été assurée par des API REST, avec Axios côté web et Volley côté mobile. Toutes les fonctionnalités de gestion des lignes, des arrêts, des trajets, de géolocalisation en temps réel et d'affichage cartographique ont été implémentées avec succès, en suivant une organisation modulaire du code inspirée du modèle MVC, malgré l'absence de framework côté serveur.

Toutefois, plusieurs limites ont été identifiées durant le projet. L'utilisation de PHP 5.6 a restreint l'accès à certaines fonctionnalités modernes, tandis que l'absence d'un pipeline CI/CD a nécessité un déploiement manuel, ce qui n'est pas optimal pour un projet évolutif. De plus, le déploiement local via XAMPP rend l'application difficilement accessible à grande échelle et son installation nécessite des configurations spécifiques, comme la version de PHP et celle de NetBeans. Enfin, le temps disponible n'a pas permis d'implémenter un système de gestion d'erreurs avancé ni de tests automatisés, ce qui réduit la robustesse du système dans un environnement de production.

Avec ces contraintes ; nous nous proposons de contourner le problème par plusieurs pistes d'amélioration qui ont été identifiées. Sur le plan technique, l'adoption d'un framework moderne permettrait de renforcer la sécurité, la maintenabilité et la structuration du code. Un déploiement sur une infrastructure cloud (comme AWS ou Azure) garantirait une accessibilité continue en ligne et une gestion optimisée des montées en charge. L'introduction d'un pipeline CI/CD via GitHub Actions ou GitLab CI permettrait d'automatiser les tests et les mises à jour, tandis que l'ajout de tests unitaires renforcerait la stabilité du système. En termes de fonctionnalités, des améliorations telles que l'intégration de notifications en temps réel en cas de retard ou de changement de trajet, ainsi que des statistiques d'analyse de la fréquentation du réseau, viendraient enrichir l'expérience utilisateur.

Concernant la pérennité du projet, l'architecture locale actuelle et l'absence d'automatisation des déploiements limitent son utilisation à grande échelle. Pour assurer sa viabilité à long terme, il est essentiel de mettre en place une infrastructure de déploiement automatisée, intégrée à un processus de gestion de versions. Cela garantirait la portabilité, la stabilité et l'évolutivité du système tout en facilitant les mises à jour et en répondant aux besoins futurs.

Apport du projet

La conception et la réalisation d'un système de géolocalisation des bus ont représenté pour nous, en binôme, une expérience particulièrement formatrice, tant sur le plan technique qu'organisationnel et collaboratif. Travailler ensemble nous a permis de croiser nos compétences, de confronter nos idées et de répartir efficacement les tâches à chaque étape du projet. Cette collaboration a joué un rôle essentiel dans la réussite du développement de la plateforme de suivi et de gestion du réseau de transport urbain.

Dès le départ, nous avons défini des objectifs clairs : proposer une solution simple, interactive et efficace permettant aux usagers de consulter en temps réel la position des bus, tout en offrant aux administrateurs des outils de gestion centralisée des lignes, arrêts, trajets et véhicules. Le projet a nécessité une réflexion poussée sur l'architecture logicielle, la structuration des données, l'expérience utilisateur et la synchronisation entre les différentes interfaces web, mobile et serveur.

Sur le plan technique, nous avons consolidé nos compétences en développement web et mobile, en travaillant avec React.js pour le front-end, PHP et MySQL pour le back-end, et Java pour l'application Android. Nous avons utilisé Axios pour les requêtes HTTP côté web, et Volley pour la transmission des données GPS depuis l'application mobile. La modélisation UML rigoureuse nous a permis d'assurer la cohérence de l'ensemble du système. Par ailleurs, la rédaction du rapport en LaTeX a renforcé la qualité de notre documentation, et l'utilisation de la méthode Scrum nous a aidés à avancer de manière itérative, avec une organisation efficace.

Ce projet nous a également confrontés à des défis concrets : assurer la fiabilité des données GPS, gérer les interactions entre modules, optimiser les performances en temps réel et garantir la compatibilité entre plateformes. Il nous a permis de mieux comprendre les enjeux liés à la mobilité urbaine intelligente, d'améliorer notre communication en binôme, et de documenter chaque choix technique dans une logique d'amélioration continue.

Avec le recul, une mise en ligne effective du système, des tests plus approfondis en conditions réelles et une gestion automatisée des erreurs auraient permis d'approfondir davantage notre démarche. Cette expérience représente une étape clé dans notre formation et confirme notre intérêt pour la conception de solutions numériques utiles à la société.

ANNEXES

ANNEXE A — TABLEAU DE VALIDATION DES FONCTIONNALITÉS

Le tableau suivant A.1 présente une synthèse des tests réalisés sur les principales fonctionnalités de l'application.

Table A.1 : Tableau de validation des principales fonctionnalités du système

Fonction testée	**Entrée**	**Sortie attendue**	**Résultat obtenu**
Connexion utilisateur	Email : utilisateur@test.com, Mot de passe : correct	Redirection vers /dashboard	Redirection effectuée avec session utilisateur active
Connexion utilisateur	Email : faux@mail.com, Mot de passe : quelconque	Message d'erreur : utilisateur introuvable	Message affiché : "Email ou mot de passe incorrect"
Ajout d'un arrêt	Nom : Gare Centrale, Coordonnées : (31.63, -8.01)	Arrêt ajouté et visible dans la liste	Arrêt affiché instantanément dans le tableau React
Consultation des lignes	Requête GET envoyée par Axios à l'API PHP	JSON contenant les données des lignes	Liste complète des lignes reçue et affichée
Ajout d'un trajet	POST avec : ligneId, heure départ, heure arrivée	Trajet inséré dans la base de données	Trajet enregistré et confirmé par retour JSON
Affichage de la carte	Chargement de la page Carte avec données chargées	Affichage des bus et arrêts sur Google Maps	Carte interactive fonctionnelle avec tous les éléments visibles
Localisation du bus	Coordonnées GPS envoyées via l'application mobile	Position affichée sur la carte en temps réel	Position exacte visible sur la carte avec mise à jour dynamique
Ajout d'une ligne de bus	Nom de la ligne + liste d'arrêts	Nouvelle ligne enregistrée et affichée	Ligne ajoutée et visible dans la liste côté admin
Ajout d'un arrêt au système	Nom + coordonnées géographiques	Nouvel arrêt sauvegardé et lié à une ligne	Arrêt visible dans les sélections de trajets
Ajout d'un favori (ligne)	ID utilisateur + ligne sélectionnée	Ligne enregistrée dans les favoris de l'utilisateur	Favori ajouté, affiché dans la section personnelle
Ajout d'un favori (arrêt)	ID utilisateur + arrêt sélectionné	Arrêt enregistré dans les favoris de l'utilisateur	Favori enregistré et affiché côté interface utilisateur
Ajout d'un bus	Matricule du bus + état + capacité	Bus enregistré dans la base de données	Bus visible dans l'interface admin
Ajout d'un trajet	Ligne sélectionnée + heure départ/arrivée + bus affecté	Trajet sauvegardé et lié à une ligne	Trajet affiché dans la liste des trajets
Suppression d'un bus	ID du bus sélectionné	Bus supprimé de la base de données	Bus retiré de la liste sans erreur
Suppression d'un trajet	ID du trajet sélectionné	Trajet supprimé de la base	Trajet effacé et interface mise à jour
Suppression d'une ligne	ID ligne sélectionnée	Ligne supprimée avec ses trajets liés	Ligne retirée sans erreur, dépendances traitées
Suppression d'un arrêt	ID arrêt sélectionné	Arrêt supprimé de la base	Arrêt disparu de la carte et de la liste



KOUBRI Hatim

Résumé du Projet

Ce projet de fin d'études a pour objectif la mise en place d'un système intelligent, interactif et géolocalisé en temps réel des bus dans le cadre d'une plateforme globale de gestion des lignes, des arrêts et des transports sur une ligne donnée, suscitant un besoin parfaitement identifié d'optimisation du réseau de transport urbain et d'information des usagers au cœur des systèmes de transport intelligents (ITS) tout en étant une solution numérique modulaire permettant une gestion efficace et intelligente des données d'un réseau de bus et de la géolocalisation dynamique des véhicules. [...]

Mots clés : géolocalisation en temps réel, transport intelligent, API REST, React.js, PHP, Android, MySQL, modélisation UML, Tailwind CSS, Axios, Postman.