



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

PROJET DE FIN D'ANNÉE

RouteChain
Optimisation des Itinéraires de Livraison
avec Traçabilité Blockchain

Réalisé par :
RAFI Mohamed Anas
KOSSARA Youness

Encadré par :
Dr. EL KSIMI Ali

Résumé

L'essor fulgurant du commerce électronique et l'augmentation constante des volumes de livraison représentent des défis majeurs pour le secteur logistique, exigeant une optimisation accrue des opérations tout en garantissant la traçabilité et la transparence des échanges. Ce projet de fin d'année s'inscrit dans le cadre du développement d'une solution innovante combinant optimisation des tournées de livraison et technologie blockchain.

Notre approche intègre des techniques avancées d'optimisation combinatoire pour résoudre le problème de tournées de véhicules (VRP). Le système implémenté utilise Google OR-Tools avec des stratégies d'optimisation avancées pour calculer les itinéraires optimaux minimisant la distance totale parcourue. L'intégration de l'API OpenRouteService permet d'obtenir des matrices de distances réelles basées sur le réseau routier.

La composante blockchain du projet assure l'immuabilité et la vérifiabilité des données de livraison. Un smart contract *RouteRegistry*, développé en Solidity et déployé sur une blockchain Ethereum locale via Ganache, enregistre le hash cryptographique SHA-256 de chaque tournée. Ce mécanisme permet de garantir l'intégrité des données et de constituer une preuve irréfutable en cas de litige.

L'implémentation pratique comprend une application web full-stack moderne développée avec FastAPI pour le backend et React avec Vite pour le frontend. L'interface offre un dashboard intuitif pour la gestion des tournées, la visualisation cartographique via Leaflet, le suivi des livraisons en temps réel et la vérification de l'intégrité blockchain. Le système intègre également une authentification JWT sécurisée et une gestion des rôles (administrateur/chauffeur).

Ce projet contribue significativement à moderniser les opérations de livraison du dernier kilomètre, offrant une solution adaptée aux défis de transparence et d'efficacité du secteur logistique et ouvrant la voie à de futures innovations dans la traçabilité des chaînes d'approvisionnement.

Abstract

The rapid growth of e-commerce and the ever-increasing volume of deliveries pose significant challenges for the logistics sector, requiring enhanced operational optimization while ensuring traceability and transparency of exchanges. This final year project focuses on developing an innovative solution that combines delivery route optimization with blockchain technology.

Our approach integrates advanced combinatorial optimization techniques to solve the Vehicle Routing Problem (VRP). The implemented system uses Google OR-Tools with advanced optimization strategies to calculate optimal routes that minimize total travel distance. Integration with the OpenRouteService API provides real distance matrices based on the actual road network.

The blockchain component of the project ensures immutability and verifiability of delivery data. A *RouteRegistry* smart contract, developed in Solidity and deployed on a local Ethereum blockchain via Ganache, records the SHA-256 cryptographic hash of each route. This mechanism guarantees data integrity and provides irrefutable proof in case of disputes.

The practical implementation includes a modern full-stack web application developed with FastAPI for the backend and React with Vite for the frontend. The interface offers an intuitive dashboard for route management, map visualization via Leaflet, real-time delivery tracking, and blockchain integrity verification. The system also integrates secure JWT authentication and role management (administrator/driver).

This project significantly contributes to modernizing last-mile delivery operations, providing a solution tailored to the transparency and efficiency challenges of the logistics sector and paving the way for future innovations in supply chain traceability.

Table des matières

Résumé	1
Abstract	2
Introduction Générale	1
1 Cadrage du Projet	2
1.1 Introduction	2
1.2 Contexte et Motivations du Projet	2
1.2.1 Le Défi de la Logistique du Dernier Kilomètre	2
1.2.2 Le Besoin de Transparence et de Confiance dans les Chaînes de Livraison	2
1.3 Étude des Solutions Existantes	3
1.3.1 Outils Traditionnels de Planification d'Itinéraires	3
1.3.2 La Blockchain dans la Gestion de la Chaîne d'Approvisionnement	3
1.4 Critique des Solutions Existantes	3
1.4.1 Absence d'Intégration entre Optimisation et Traçabilité	3
1.4.2 Enregistrements Centralisés et Mutables	3
1.5 Solution Proposée : RouteChain	4
1.5.1 Combinaison de l'Optimisation VRP et de l'Immuabilité Blockchain	4
1.5.2 Objectifs et Périmètre du Projet	4
1.6 Conclusion	4
2 Spécification des Besoins	5
2.1 Introduction	5
2.2 Identification des Acteurs	5
2.2.1 Administrateur	5
2.2.2 Chauffeur	5
2.3 Besoins Fonctionnels	6
2.3.1 Gestion des Utilisateurs	6
2.3.1.1 Authentification	6
2.3.1.2 Gestion des Rôles	6
2.3.2 Gestion des Clients et Dépôts	7
2.3.3 Création et Optimisation des Tournées	7
2.3.4 Exécution des Livraisons	7
2.3.5 Interaction avec la Blockchain	8
2.3.6 Analytique et Reporting	8
2.4 Besoins Non Fonctionnels	8
2.4.1 Performance et Scalabilité	8
2.4.2 Sécurité	9
2.4.3 Utilisabilité et Accessibilité	9
2.4.4 Intégrité des Données	9
2.5 Conclusion	9
3 Conception du Système	10
3.1 Introduction	10
3.2 Architecture Globale du Système	10
3.2.1 Architecture Trois-Tiers	10
3.2.2 Diagramme d'Interaction des Composants	11
3.3 Modélisation UML : Diagrammes de Cas d'Utilisation	11
3.3.1 Diagramme de Cas d'Utilisation Global	11

3.3.2	Diagramme de Cas d'Utilisation : Sous-système Chauffeur	12
3.3.3	Diagramme de Cas d'Utilisation : Sous-système Administrateur	12
3.4	Modélisation UML : Diagramme de Classes	13
3.4.1	Modèle du Domaine	13
3.5	Modélisation UML : Diagrammes de Séquence	13
3.5.1	Diagramme de Séquence : Authentification	13
3.5.2	Diagramme de Séquence : Processus d'Optimisation VRP	14
3.5.3	Diagramme de Séquence : Vérification de l'Intégrité Blockchain	14
3.6	Conception du Schéma de Base de Données	15
3.6.1	Collections MongoDB	15
3.7	Architecture du Smart Contract	15
3.7.1	Structure du Contrat RouteRegistry	15
3.8	Conclusion	16
4	Réalisation	17
4.1	Introduction	17
4.2	Environnement de Développement	17
4.2.1	Outils de Développement	17
4.2.2	Structure du Projet	17
4.3	Technologies Backend	17
4.3.1	Python	17
4.3.2	FastAPI	18
4.3.3	MongoDB	18
4.3.4	Google OR-Tools	18
4.4	Technologies Frontend	19
4.4.1	React	19
4.4.2	Vite	19
4.4.3	Tailwind CSS	20
4.4.4	Leaflet	20
4.4.5	Axios	20
4.5	Technologies Blockchain	21
4.5.1	Ganache	21
4.6	Présentation des Interfaces Utilisateur	21
4.6.1	Écrans d'Authentification	21
4.6.2	Dashboard et Gestion des Tournées	22
4.6.3	Création de Tournée	23
4.6.4	Détail de Tournée et Visualisation Cartographique	24
4.6.5	Profil Utilisateur	25
4.6.6	Gestion des Clients	25
4.6.7	Gestion des Dépôts	26
4.6.8	Panel Administrateur	26
4.6.9	Dashboard Analytique	27
4.7	Conclusion	27
5	Discussion et Évaluation	28
5.1	Introduction	28
5.2	Défis Techniques Rencontrés	28
5.2.1	Latence de l'Optimisation VRP	28
5.2.2	Coûts de Gas et Délais Blockchain	28
5.2.3	Synchronisation MongoDB - Blockchain	28
5.3	Gestion des Données et Intégrité	29
5.3.1	Mécanisme de Vérification par Hash	29
5.3.2	Distinction entre Données Mutables et Immuables	29
5.4	Conclusion	29
Conclusion Générale		30

Table des figures

1.1	Architecture globale de RouteChain	4
3.1	Architecture trois-tiers de RouteChain	10
3.2	Diagramme d'interaction des composants	11
3.3	Diagramme de cas d'utilisation global du système RouteChain	11
3.4	Diagramme de cas d'utilisation - Sous-système Chauffeur	12
3.5	Diagramme de cas d'utilisation - Sous-système Administrateur	12
3.6	Diagramme de classes - Modèle du domaine	13
3.7	Diagramme de séquence - Processus d'authentification JWT	13
3.8	Diagramme de séquence - Processus complet d'optimisation VRP avec enregistrement blockchain	14
3.9	Diagramme de séquence - Vérification de l'intégrité via blockchain	14
3.10	Structure du smart contract RouteRegistry	16
4.1	Logo Python	17
4.2	Logo FastAPI	18
4.3	Logo MongoDB	18
4.4	Logo Google OR-Tools	18
4.5	Logo React	19
4.6	Logo Vite	19
4.7	Logo Tailwind CSS	20
4.8	Logo Leaflet	20
4.9	Logo Axios	20
4.10	Logo Ganache	21
4.11	Page de connexion de l'application RouteChain	21
4.12	Page d'inscription avec formulaire de création de compte chauffeur	22
4.13	Dashboard principal affichant la liste des tournées	22
4.14	Formulaire de création de tournée - Section dépôt et informations	23
4.15	Formulaire de création de tournée - Points de livraison et carte	23
4.16	Page de détail d'une tournée avec informations et carte	24
4.17	Visualisation cartographique et informations blockchain	24
4.18	Page de profil du chauffeur avec informations personnelles	25
4.19	Interface de gestion des clients avec recherche et géocodage	25
4.20	Interface de gestion des dépôts avec géocodage automatique	26
4.21	Panel d'administration avec gestion des chauffeurs	26
4.22	Tableau de bord analytique avec indicateurs de performance	27

Liste des tableaux

1.1	Comparaison des solutions de planification d'itinéraires existantes	3
2.1	Récapitulatif des acteurs et leurs droits	6
2.2	Exigences fonctionnelles - Authentification	6
2.3	Exigences fonctionnelles - Gestion des rôles	6
2.4	Exigences fonctionnelles - Clients et Dépôts	7
2.5	Exigences fonctionnelles - Tournées	7
2.6	Exigences fonctionnelles - Exécution	7
2.7	Exigences fonctionnelles - Blockchain	8
2.8	Exigences fonctionnelles - Analytique	8
2.9	Exigences non fonctionnelles - Performance	8
2.10	Exigences non fonctionnelles - Sécurité	9
2.11	Exigences non fonctionnelles - Utilisabilité	9
2.12	Exigences non fonctionnelles - Intégrité	9
3.1	Structure de la collection drivers	15
3.2	Structure de la collection routes	15
4.1	Outils de développement utilisés	17
5.1	Données incluses dans le calcul du hash	29
5.2	Classification des données de tournée	29

Introduction Générale

Dans un contexte économique mondial marqué par l'essor fulgurant du commerce électronique et l'augmentation constante des attentes des consommateurs en matière de rapidité et de fiabilité des livraisons, l'optimisation logistique est devenue un enjeu stratégique majeur pour les entreprises. La gestion efficace des tournées de véhicules, communément désignée sous l'acronyme VRP (*Vehicle Routing Problem*), représente l'un des défis les plus complexes et les plus étudiés dans le domaine de la recherche opérationnelle.

Parallèlement à ces préoccupations d'efficacité opérationnelle, une nouvelle exigence s'impose progressivement dans le secteur de la logistique : la traçabilité et la transparence des opérations. Les clients, qu'ils soient particuliers ou professionnels, souhaitent désormais disposer d'une visibilité complète sur le parcours de leurs colis, depuis l'entrepôt jusqu'à leur porte. Cette demande de transparence s'accompagne d'un besoin croissant de garanties quant à l'intégrité des données de livraison, notamment dans un contexte où les litiges liés aux preuves de livraison sont fréquents.

C'est dans ce double contexte d'optimisation et de transparence que s'inscrit le projet **RouteChain**. Cette application web full-stack propose une solution innovante combinant deux technologies de pointe : les algorithmes d'optimisation de tournées de véhicules, implémentés via la bibliothèque Google OR-Tools, et la technologie Blockchain, permettant d'assurer l'immuabilité et la vérifiabilité des enregistrements de livraison.

L'objectif principal de RouteChain est de fournir aux entreprises de livraison un outil complet leur permettant non seulement d'optimiser leurs itinéraires de manière à minimiser les distances parcourues et les temps de trajet, mais également de constituer une preuve irréfutable de chaque livraison effectuée grâce à l'enregistrement des données sur une blockchain Ethereum.

Le présent rapport est structuré en cinq chapitres principaux. Le **premier chapitre** présente le cadrage du projet, incluant le contexte, l'étude des solutions existantes et la proposition de notre solution. Le **deuxième chapitre** détaille la spécification des besoins fonctionnels et non fonctionnels. Le **troisième chapitre** expose la conception du système à travers différents diagrammes UML. Le **quatrième chapitre** présente la réalisation technique et les technologies utilisées. Enfin, le **cinquième chapitre** propose une discussion sur les défis rencontrés et l'évaluation du système.

Chapitre 1

Cadrage du Projet

1.1 Introduction

Ce premier chapitre a pour objectif de poser les fondements du projet RouteChain en présentant le contexte général dans lequel il s'inscrit. Nous commencerons par analyser les problématiques actuelles liées à la logistique du dernier kilomètre et au besoin de transparence dans les chaînes de livraison. Ensuite, nous examinerons les solutions existantes sur le marché, avant d'en proposer une critique constructive. Enfin, nous présenterons notre proposition de solution et les objectifs que nous nous sommes fixés.

1.2 Contexte et Motivations du Projet

1.2.1 Le Défi de la Logistique du Dernier Kilomètre

La logistique du dernier kilomètre (*Last-Mile Delivery*) représente l'étape finale de la chaîne d'approvisionnement, consistant à acheminer un produit depuis un centre de distribution jusqu'au consommateur final. Cette phase, bien que représentant généralement moins de 20% de la distance totale parcourue par un colis, génère paradoxalement entre 40% et 50% des coûts logistiques totaux.

Plusieurs facteurs expliquent cette disproportion :

- **La fragmentation des livraisons** : Chaque colis a une destination unique, contrairement aux phases précédentes où les marchandises sont consolidées.
- **Les contraintes temporelles** : Les créneaux de livraison imposés par les clients réduisent la flexibilité d'organisation des tournées.
- **L'environnement urbain** : La congestion routière, les restrictions de circulation et la difficulté à stationner complexifient les opérations.
- **Les échecs de livraison** : L'absence du destinataire engendre des coûts supplémentaires liés aux tentatives répétées.

Face à ces défis, l'optimisation des tournées de véhicules devient un levier essentiel pour réduire les coûts opérationnels tout en maintenant un niveau de service satisfaisant.

1.2.2 Le Besoin de Transparence et de Confiance dans les Chaînes de Livraison

Au-delà de l'efficacité opérationnelle, un second enjeu majeur émerge dans le secteur de la livraison : la transparence et la traçabilité des opérations. Les consommateurs modernes, habitués aux services numériques, attendent une visibilité en temps réel sur l'état de leurs commandes.

Plus fondamentalement, la question de la **preuve de livraison** constitue un point de friction récurrent entre les expéditeurs, les transporteurs et les destinataires. Les litiges portant sur des colis prétendument non livrés ou livrés endommagés représentent un coût significatif pour l'ensemble des acteurs de la chaîne.

Les systèmes traditionnels de preuve de livraison, qu'il s'agisse de signatures manuscrites ou de photographies, présentent plusieurs limites :

- **Falsifiabilité** : Les données stockées dans des bases de données centralisées peuvent être modifiées.
- **Manque de confiance** : En cas de litige, aucune partie ne dispose d'une preuve incontestable.
- **Absence d'horodatage fiable** : Les timestamps peuvent être manipulés.

C'est précisément pour répondre à ces problématiques que la technologie Blockchain apparaît comme une solution particulièrement adaptée, offrant des garanties d'immutabilité et de transparence que les systèmes traditionnels ne peuvent égaler.

1.3 Étude des Solutions Existantes

1.3.1 Outils Traditionnels de Planification d’Itinéraires

Le marché propose aujourd’hui de nombreuses solutions de planification d’itinéraires, allant des applications grand public aux plateformes professionnelles sophistiquées.

TABLE 1.1 – Comparaison des solutions de planification d’itinéraires existantes

Solution	Points Forts	Limitations	Blockchain
Google Maps	Interface intuitive, données cartographiques précises	Limité à la navigation simple, pas d’optimisation VRP	Non
Route4Me	Optimisation multi-stops, API disponible	Coût élevé, pas de traçabilité blockchain	Non
OptimoRoute	Gestion de flotte, contraintes temporelles	Propriétaire, données centralisées	Non
Circuit	Simple d’utilisation, gratuit pour petits volumes	Fonctionnalités limitées	Non

1.3.2 La Blockchain dans la Gestion de la Chaîne d’Approvisionnement

L’adoption de la technologie Blockchain dans le domaine de la supply chain connaît une croissance significative. Plusieurs initiatives majeures ont émergé ces dernières années :

- **IBM Food Trust** : Plateforme de traçabilité alimentaire utilisée par des géants comme Walmart et Carrefour.
- **TradeLens** : Solution de suivi des conteneurs maritimes développée par IBM et Maersk.
- **VeChain** : Blockchain publique dédiée à la traçabilité des produits de luxe et pharmaceutiques.

Ces initiatives démontrent la pertinence de la Blockchain pour assurer la traçabilité et l’intégrité des données logistiques. Cependant, elles se concentrent principalement sur le suivi des marchandises à grande échelle et n’adressent pas spécifiquement la problématique de l’optimisation des tournées de livraison.

1.4 Critique des Solutions Existantes

1.4.1 Absence d’Intégration entre Optimisation et Traçabilité

L’analyse des solutions existantes révèle une dichotomie marquée entre deux catégories d’outils :

1. **Les outils d’optimisation** qui se concentrent sur la planification des itinéraires sans offrir de mécanismes de traçabilité avancés.
2. **Les solutions blockchain** qui assurent la traçabilité mais n’intègrent pas de fonctionnalités d’optimisation logistique.

Cette fragmentation oblige les entreprises à utiliser plusieurs systèmes en parallèle, engendrant des problèmes d’interopérabilité et une complexité accrue dans la gestion des opérations.

1.4.2 Enregistrements Centralisés et Mutables

La majorité des solutions de gestion de livraison reposent sur des architectures centralisées où les données sont stockées dans des bases de données traditionnelles. Cette approche présente plusieurs inconvénients :

- Les données peuvent être modifiées a posteriori par l’administrateur du système.
- En cas de litige, l’entreprise détentrice des données est à la fois juge et partie.
- La confiance repose entièrement sur la bonne foi de l’opérateur du système.

1.5 Solution Proposée : RouteChain

1.5.1 Combinaison de l'Optimisation VRP et de l'Immuabilité Blockchain

Face aux limitations identifiées, nous proposons **RouteChain**, une application web full-stack qui combine de manière innovante :

- **Un moteur d'optimisation VRP** basé sur Google OR-Tools, permettant de calculer les itinéraires optimaux en minimisant la distance totale parcourue.
- **Un smart contract Ethereum** déployé sur une blockchain locale (Ganache), assurant l'enregistrement immuable des données de chaque tournée.
- **Une interface utilisateur moderne** permettant aux chauffeurs et aux administrateurs de gérer efficacement les opérations de livraison.

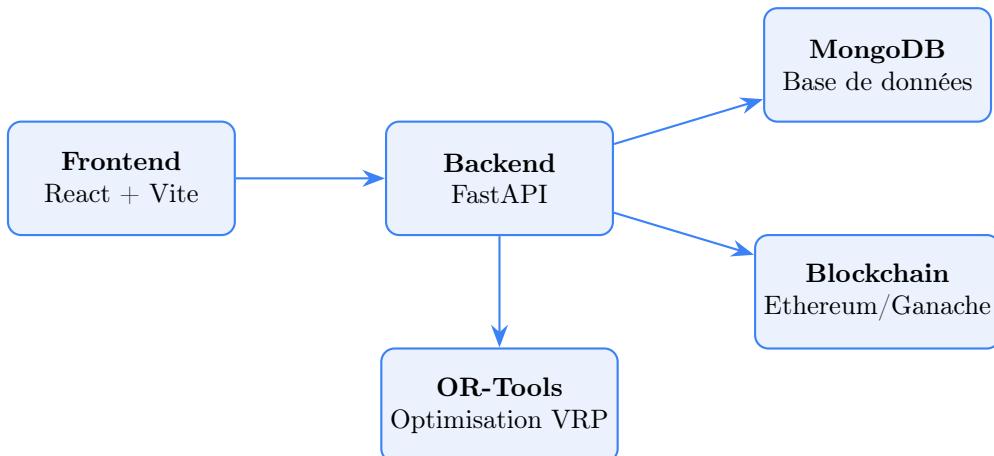


FIGURE 1.1 – Architecture globale de RouteChain

1.5.2 Objectifs et Périmètre du Projet

Les objectifs principaux de RouteChain sont les suivants :

1. **Optimiser les tournées de livraison** en utilisant des algorithmes de résolution du VRP pour minimiser les distances et les temps de parcours.
2. **Garantir l'intégrité des données** en enregistrant un hash cryptographique de chaque tournée sur la blockchain.
3. **Offrir une interface intuitive** permettant aux utilisateurs de créer, visualiser et gérer leurs itinéraires de livraison.
4. **Fournir des outils d'administration** pour la gestion des chauffeurs, des clients et des dépôts.
5. **Proposer des analyses statistiques** sur les performances des livraisons.

Le périmètre du projet se limite à une application web responsive, déployée localement avec une blockchain privée (Ganache). L'évolution vers une blockchain publique (testnet ou mainnet) est envisagée comme perspective future.

1.6 Conclusion

Ce premier chapitre a permis de situer le projet RouteChain dans son contexte économique et technologique. Nous avons identifié les deux problématiques majeures auxquelles notre solution répond : l'optimisation des tournées de livraison et la garantie d'intégrité des données via la blockchain.

L'analyse des solutions existantes a révélé l'absence d'outils combinant efficacement ces deux dimensions, justifiant ainsi la pertinence de notre proposition. Le chapitre suivant sera consacré à la spécification détaillée des besoins fonctionnels et non fonctionnels de l'application RouteChain.

Chapitre 2

Spécification des Besoins

2.1 Introduction

Ce chapitre est consacré à la spécification détaillée des besoins de l'application RouteChain. Nous commencerons par identifier les différents acteurs du système, puis nous détaillerons les exigences fonctionnelles regroupées par domaine métier. Enfin, nous présenterons les exigences non fonctionnelles qui garantissent la qualité globale de la solution.

2.2 Identification des Acteurs

L'application RouteChain distingue deux profils d'utilisateurs principaux, chacun disposant de droits et de fonctionnalités spécifiques.

2.2.1 Administrateur

L'administrateur est responsable de la gestion globale de la plateforme. Son rôle comprend :

- La gestion des comptes chauffeurs (création, modification, suppression)
- La supervision de l'ensemble des tournées créées dans le système
- L'accès aux tableaux de bord analytiques globaux
- La gestion des clients et des dépôts
- La configuration des paramètres système

2.2.2 Chauffeur

Le chauffeur est l'utilisateur principal de l'application sur le terrain. Ses responsabilités incluent :

- La création et l'optimisation de ses propres tournées de livraison
- L'exécution des livraisons avec confirmation de chaque point
- La consultation de l'historique de ses tournées
- L'accès aux statistiques personnelles de performance
- La vérification de l'intégrité blockchain de ses données

TABLE 2.1 – Récapitulatif des acteurs et leurs droits

Fonctionnalité	Administrateur	Chauffeur
Créer une tournée	✓	✓
Gérer ses propres tournées	✓	✓
Voir toutes les tournées	✓	–
Gérer les chauffeurs	✓	–
Gérer les clients	✓	✓
Gérer les dépôts	✓	✓
Accès analytics global	✓	–
Vérification blockchain	✓	✓

2.3 Besoins Fonctionnels

2.3.1 Gestion des Utilisateurs

2.3.1.1 Authentification

TABLE 2.2 – Exigences fonctionnelles - Authentification

ID	Description
RF-AUTH-01	Le système doit permettre l'inscription d'un nouveau chauffeur avec email, mot de passe, nom et prénom.
RF-AUTH-02	Le système doit permettre la connexion via email et mot de passe.
RF-AUTH-03	Le système doit générer un token JWT valide pour 7 jours après authentification réussie.
RF-AUTH-04	Le système doit permettre la déconnexion en invalidant le token côté client.
RF-AUTH-05	Le système doit afficher le profil de l'utilisateur connecté.

2.3.1.2 Gestion des Rôles

TABLE 2.3 – Exigences fonctionnelles - Gestion des rôles

ID	Description
RF-ROLE-01	Le système doit distinguer deux rôles : Administrateur et Chauffeur.
RF-ROLE-02	L'administrateur doit pouvoir promouvoir un chauffeur au rôle d'administrateur.
RF-ROLE-03	Le système doit restreindre l'accès aux fonctionnalités selon le rôle de l'utilisateur.

2.3.2 Gestion des Clients et Dépôts

TABLE 2.4 – Exigences fonctionnelles - Clients et Dépôts

ID	Description
RF-CUST-01	Le système doit permettre la création d'un client avec nom, adresse et coordonnées GPS.
RF-CUST-02	Le système doit permettre la modification et la suppression d'un client.
RF-CUST-03	Le système doit proposer un géocodage automatique pour convertir une adresse en coordonnées.
RF-DEPOT-01	Le système doit permettre la création d'un dépôt avec nom, adresse et coordonnées GPS.
RF-DEPOT-02	Le système doit permettre de sélectionner un dépôt comme point de départ d'une tournée.

2.3.3 Crédation et Optimisation des Tournées

TABLE 2.5 – Exigences fonctionnelles - Tournées

ID	Description
RF-ROUTE-01	Le système doit permettre la création d'une tournée avec un nom, un dépôt et une liste de points de livraison.
RF-ROUTE-02	Le système doit calculer la matrice des distances entre tous les points via OpenRouteService.
RF-ROUTE-03	Le système doit exécuter l'algorithme VRP (Google OR-Tools) pour déterminer l'ordre optimal de visite.
RF-ROUTE-04	Le système doit afficher l'itinéraire optimisé sur une carte interactive.
RF-ROUTE-05	Le système doit calculer la distance totale et la durée estimée de la tournée.
RF-ROUTE-06	Le système doit permettre de modifier une tournée non démarrée.

2.3.4 Exécution des Livraisons

TABLE 2.6 – Exigences fonctionnelles - Exécution

ID	Description
RF-EXEC-01	Le système doit permettre de démarrer une tournée (passage au statut "En cours").
RF-EXEC-02	Le système doit permettre de confirmer la livraison de chaque point individuellement.
RF-EXEC-03	Le système doit mettre à jour le compteur de livraisons effectuées.
RF-EXEC-04	Le système doit marquer automatiquement la tournée comme "Terminée" lorsque tous les points sont livrés.
RF-EXEC-05	Le système doit permettre l'ouverture de l'itinéraire dans une application de navigation externe (Google Maps, Waze).

2.3.5 Interaction avec la Blockchain

TABLE 2.7 – Exigences fonctionnelles - Blockchain

ID	Description
RF-BC-01	Le système doit enregistrer automatiquement le hash des données d'une tournée lors de sa création sur la blockchain.
RF-BC-02	Le système doit mettre à jour l'enregistrement blockchain lors du changement de statut d'une tournée.
RF-BC-03	Le système doit permettre de vérifier l'intégrité des données d'une tournée en comparant le hash actuel avec celui stocké sur la blockchain.
RF-BC-04	Le système doit afficher les informations de transaction blockchain (hash, numéro de bloc).
RF-BC-05	Le système doit fonctionner en mode dégradé (sans blockchain) si Ganache n'est pas disponible.

2.3.6 Analytique et Reporting

TABLE 2.8 – Exigences fonctionnelles - Analytique

ID	Description
RF-STAT-01	Le système doit afficher des statistiques globales : nombre de tournées, distance totale, livraisons effectuées.
RF-STAT-02	Le système doit présenter des graphiques d'évolution des performances.
RF-STAT-03	Le système doit permettre l'export d'une tournée au format PDF.
RF-STAT-04	Le système doit permettre l'export d'une tournée au format CSV.

2.4 Besoins Non Fonctionnels

2.4.1 Performance et Scalabilité

TABLE 2.9 – Exigences non fonctionnelles - Performance

ID	Description
RNF-PERF-01	Le temps de réponse de l'API doit être inférieur à 500ms pour les opérations courantes.
RNF-PERF-02	L'optimisation VRP doit s'exécuter en moins de 10 secondes pour une tournée de 20 points.
RNF-PERF-03	L'interface utilisateur doit rester fluide (60 fps) lors des interactions cartographiques.

2.4.2 Sécurité

TABLE 2.10 – Exigences non fonctionnelles - Sécurité

ID	Description
RNF-SEC-01	Les mots de passe doivent être hashés avec bcrypt avant stockage.
RNF-SEC-02	L'authentification doit utiliser des tokens JWT signés avec une clé secrète.
RNF-SEC-03	Les données sensibles doivent être hashées (SHA-256) avant enregistrement blockchain.
RNF-SEC-04	L'API doit valider toutes les entrées utilisateur pour prévenir les injections.

2.4.3 Utilisabilité et Accessibilité

TABLE 2.11 – Exigences non fonctionnelles - Utilisabilité

ID	Description
RNF-UX-01	L'interface doit être responsive et utilisable sur mobile, tablette et desktop.
RNF-UX-02	L'application doit être installable en tant que PWA (Progressive Web App).
RNF-UX-03	Les messages d'erreur doivent être explicites et guider l'utilisateur.

2.4.4 Intégrité des Données

TABLE 2.12 – Exigences non fonctionnelles - Intégrité

ID	Description
RNF-INT-01	Les enregistrements blockchain doivent être immuables une fois créés.
RNF-INT-02	Le système doit permettre de prouver qu'une donnée n'a pas été modifiée depuis son enregistrement.
RNF-INT-03	Les horodatages doivent être synchronisés avec le timestamp du bloc blockchain.

2.5 Conclusion

Ce chapitre a permis de définir de manière exhaustive les exigences fonctionnelles et non fonctionnelles de l'application RouteChain. Nous avons identifié deux acteurs principaux (Administrateur et Chauffeur) et détaillé leurs besoins respectifs.

Les exigences fonctionnelles couvrent l'ensemble du cycle de vie d'une tournée de livraison : création, optimisation, exécution, et vérification blockchain. Les exigences non fonctionnelles garantissent la performance, la sécurité, l'utilisabilité et l'intégrité du système.

Le chapitre suivant sera consacré à la conception du système, avec la modélisation UML des différents aspects de l'application.

Chapitre 3

Conception du Système

3.1 Introduction

Ce chapitre présente la conception détaillée du système RouteChain à travers différents niveaux d'abstraction. Nous commencerons par une vue d'ensemble de l'architecture globale, puis nous utiliserons le langage de modélisation UML pour représenter les différents aspects du système : cas d'utilisation, structure statique et comportement dynamique. Nous terminerons par la conception du schéma de base de données et l'architecture du smart contract.

3.2 Architecture Globale du Système

3.2.1 Architecture Trois-Tiers

RouteChain adopte une architecture trois-tiers classique, séparant clairement les responsabilités entre la présentation, la logique métier et la persistance des données.

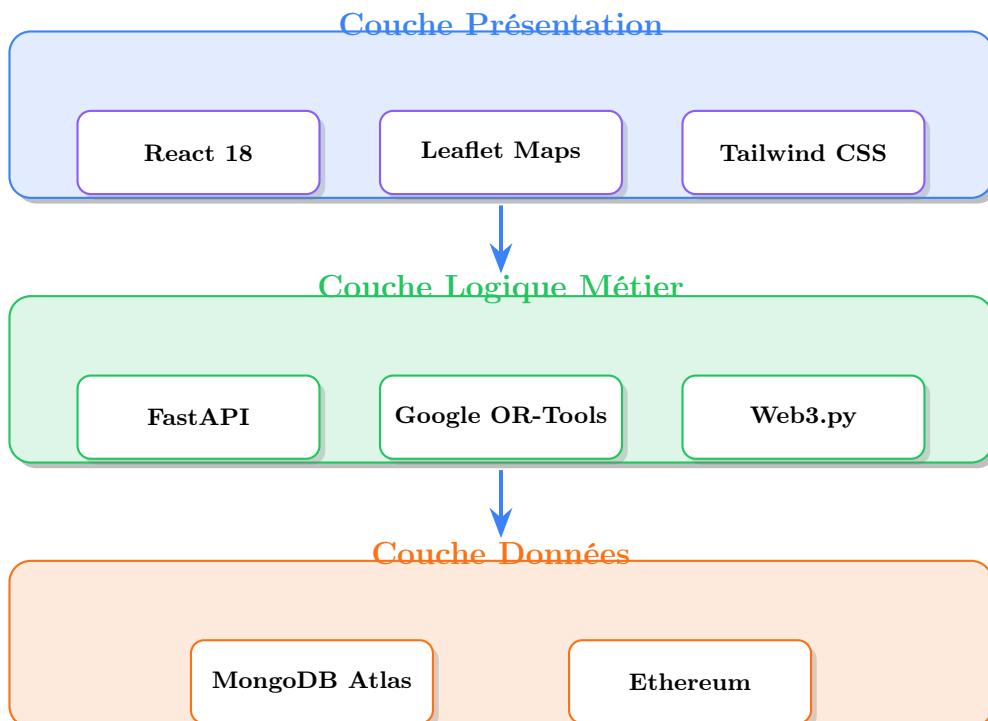


FIGURE 3.1 – Architecture trois-tiers de RouteChain

3.2.2 Diagramme d'Interaction des Composants

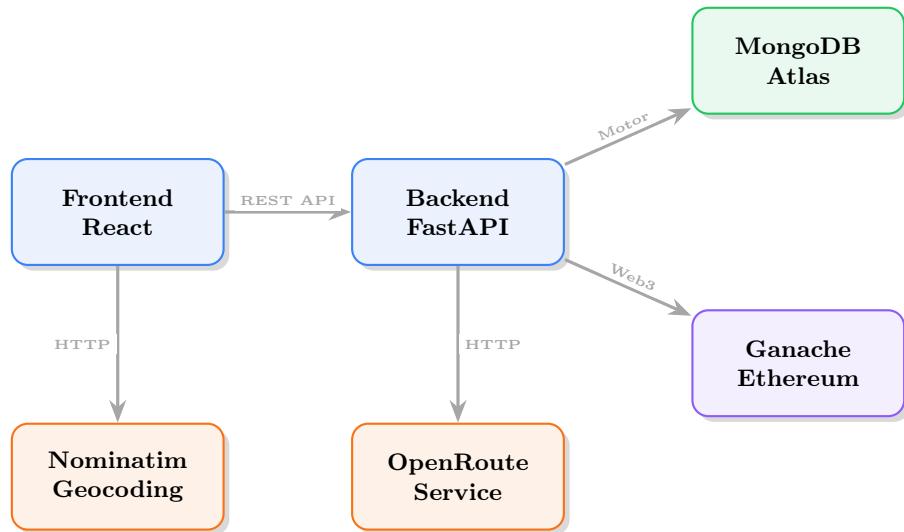


FIGURE 3.2 – Diagramme d'interaction des composants

3.3 Modélisation UML : Diagrammes de Cas d'Utilisation

3.3.1 Diagramme de Cas d'Utilisation Global

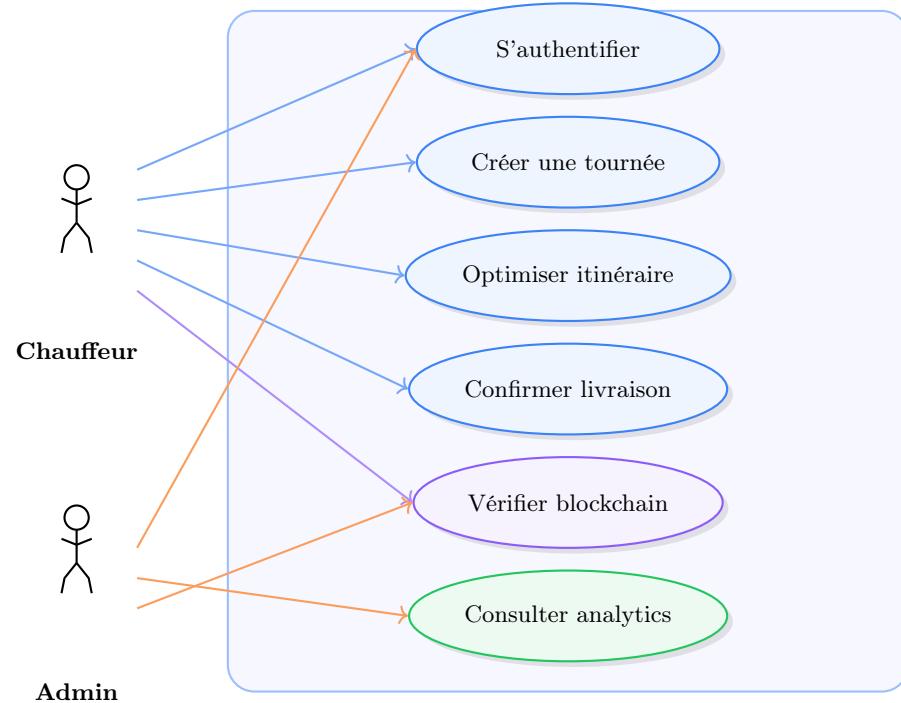


FIGURE 3.3 – Diagramme de cas d'utilisation global du système RouteChain

3.3.2 Diagramme de Cas d'Utilisation : Sous-système Chauffeur

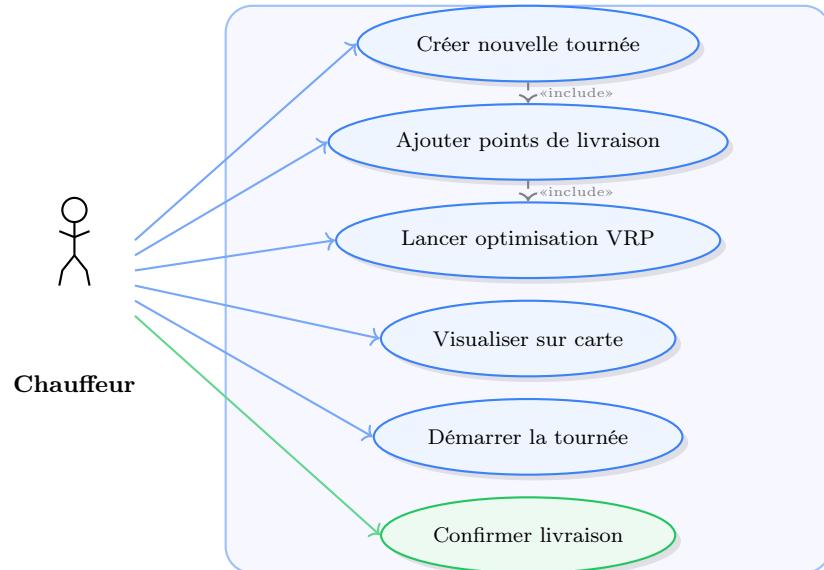


FIGURE 3.4 – Diagramme de cas d'utilisation - Sous-système Chauffeur

3.3.3 Diagramme de Cas d'Utilisation : Sous-système Administrateur

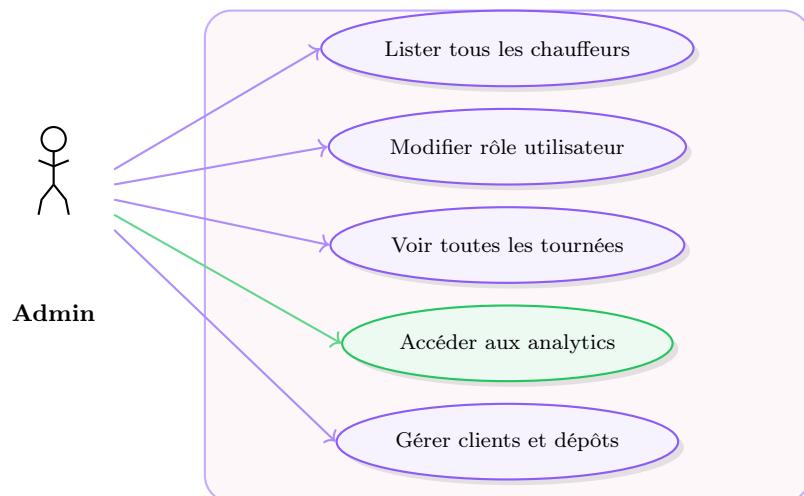


FIGURE 3.5 – Diagramme de cas d'utilisation - Sous-système Administrateur

3.4 Modélisation UML : Diagramme de Classes

3.4.1 Modèle du Domaine

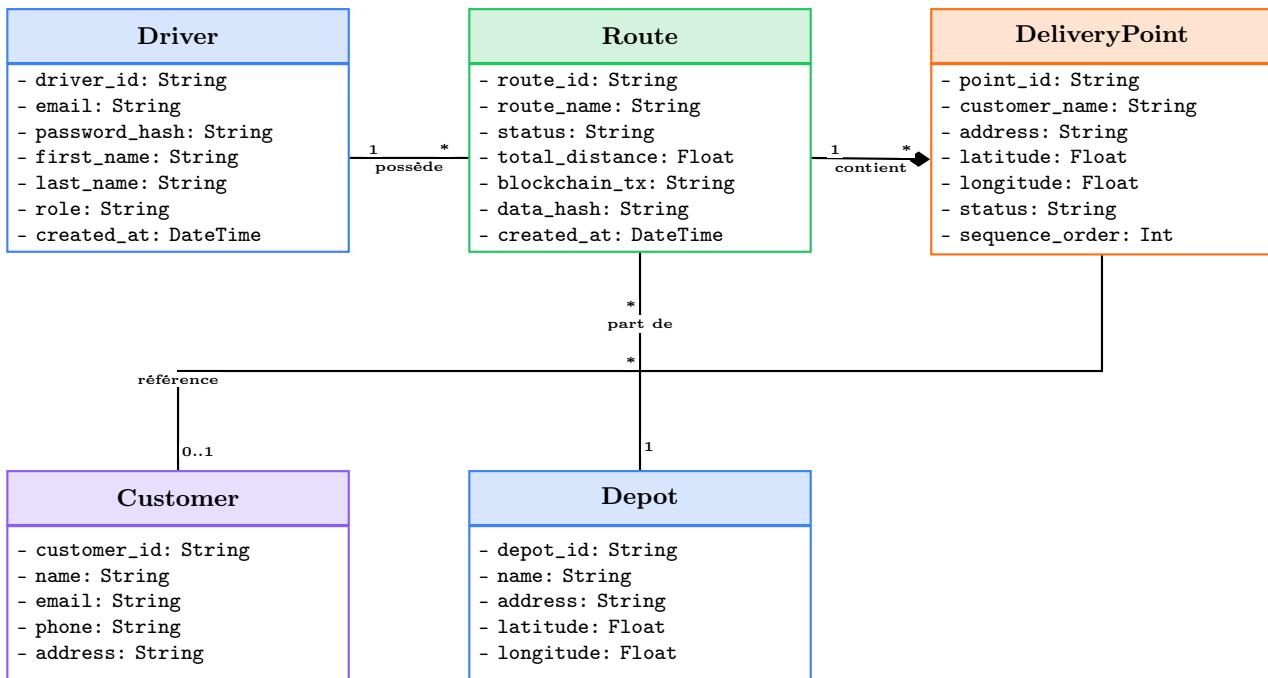


FIGURE 3.6 – Diagramme de classes - Modèle du domaine

3.5 Modélisation UML : Diagrammes de Séquence

3.5.1 Diagramme de Séquence : Authentification

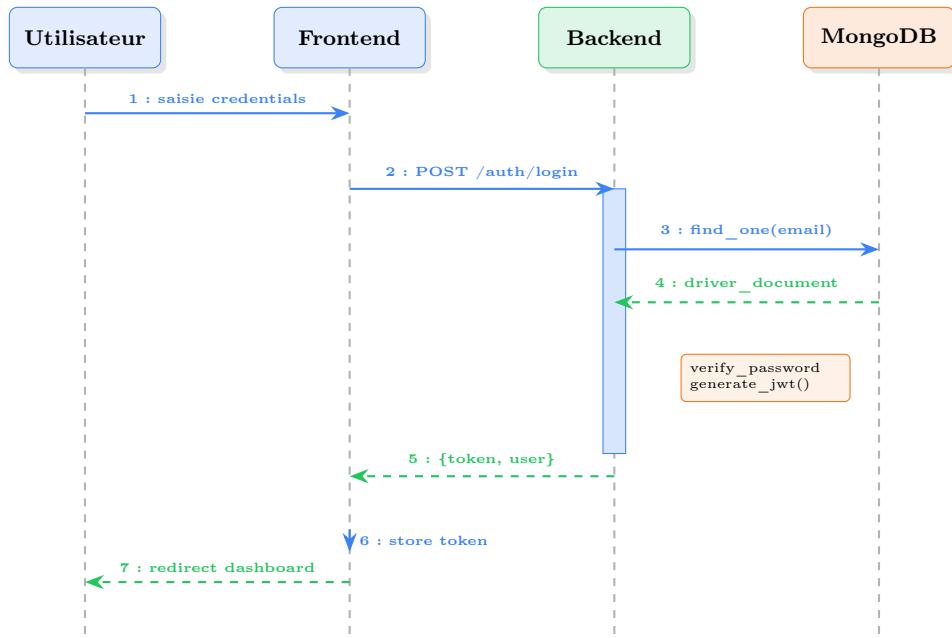


FIGURE 3.7 – Diagramme de séquence - Processus d'authentification JWT

3.5.2 Diagramme de Séquence : Processus d'Optimisation VRP

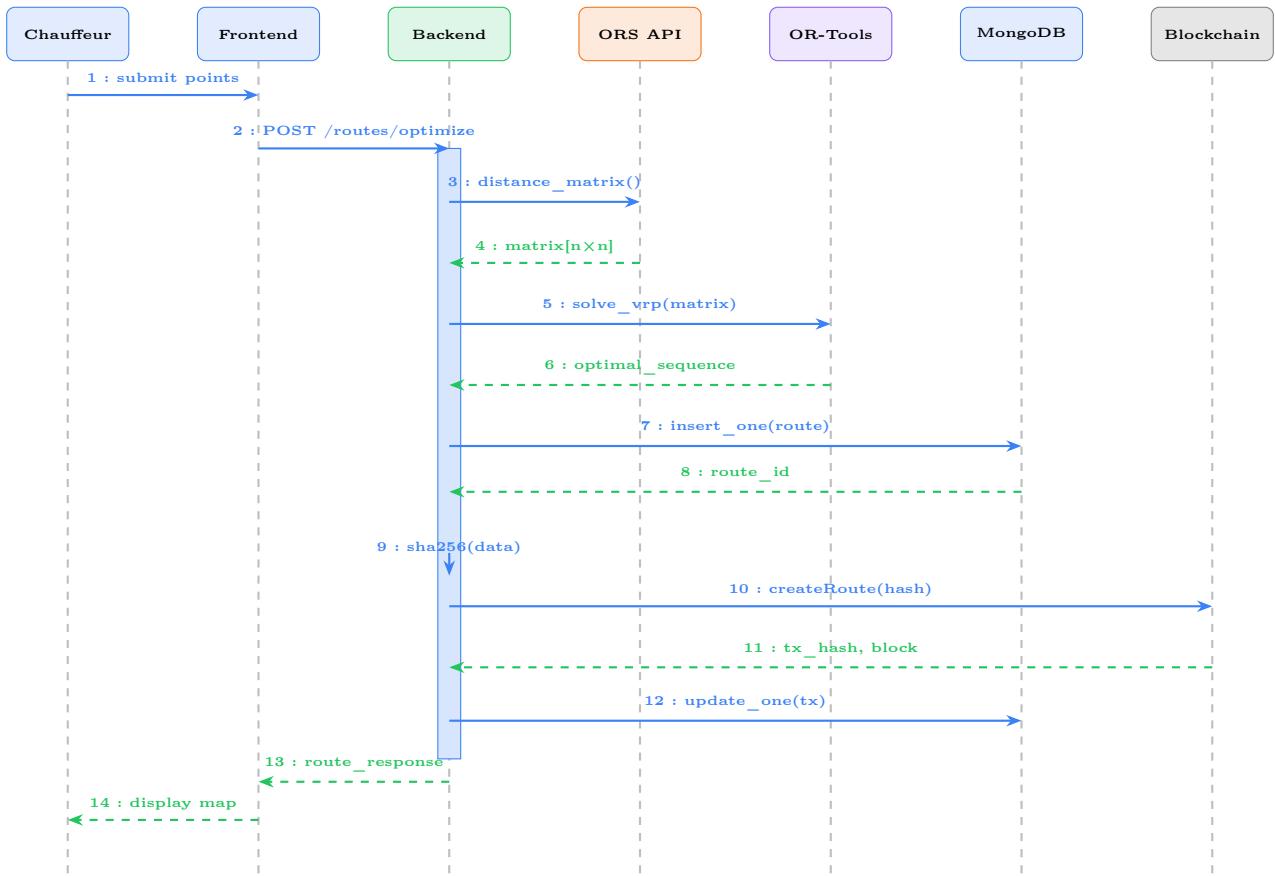


FIGURE 3.8 – Diagramme de séquence - Processus complet d'optimisation VRP avec enregistrement blockchain

3.5.3 Diagramme de Séquence : Vérification de l'Intégrité Blockchain

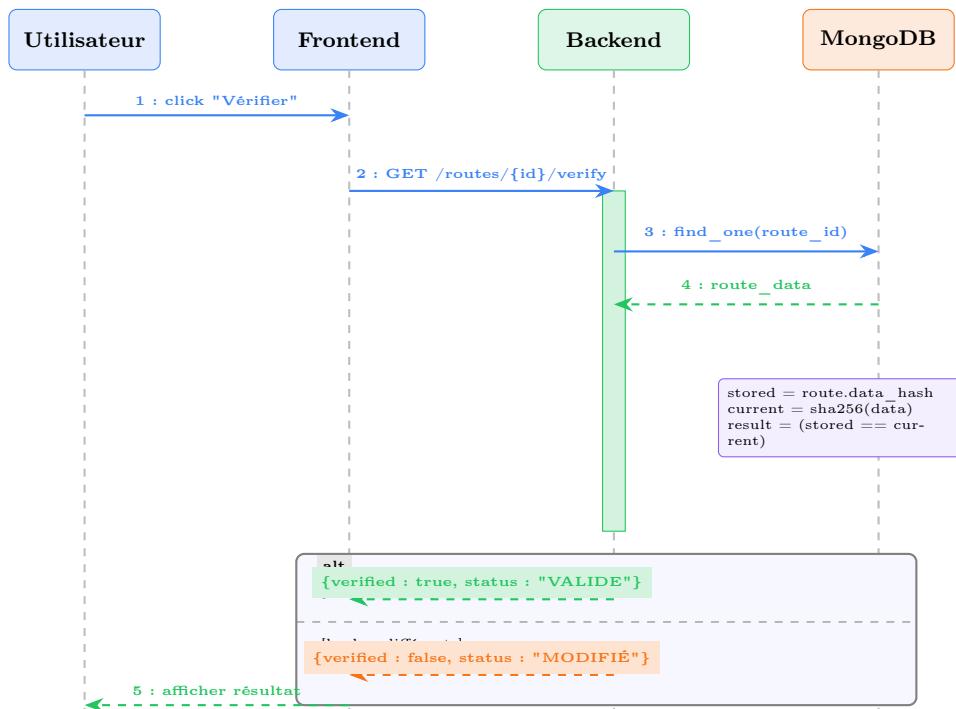


FIGURE 3.9 – Diagramme de séquence - Vérification de l'intégrité via blockchain

3.6 Conception du Schéma de Base de Données

3.6.1 Collections MongoDB

RouteChain utilise MongoDB, une base de données NoSQL orientée documents. Les principales collections sont décrites ci-dessous.

TABLE 3.1 – Structure de la collection `drivers`

Champ	Type	Description
<code>_id</code>	ObjectId	Identifiant MongoDB
<code>driver_id</code>	String	Identifiant unique (DRV_YYYYMMDD_HHMMSS)
<code>email</code>	String	Adresse email (unique)
<code>password_hash</code>	String	Mot de passe hashé (bcrypt)
<code>first_name</code>	String	Prénom
<code>last_name</code>	String	Nom de famille
<code>role</code>	String	Rôle (admin ou driver)
<code>created_at</code>	DateTime	Date de création

TABLE 3.2 – Structure de la collection `routes`

Champ	Type	Description
<code>_id</code>	ObjectId	Identifiant MongoDB
<code>route_id</code>	String	Identifiant unique (ROUTE_YYYYMMDD_HHMMSS)
<code>route_name</code>	String	Nom de la tournée
<code>driver_id</code>	String	Référence au chauffeur
<code>status</code>	String	Statut (optimized, in_progress, completed)
<code>depot_location</code>	Object	Coordonnées du dépôt
<code>delivery_points</code>	Array	Liste des points de livraison
<code>optimization_result</code>	Object	Résultats de l'optimisation
<code>blockchain_tx_hash</code>	String	Hash de la transaction blockchain
<code>blockchain_block</code>	Integer	Numéro du bloc
<code>data_hash</code>	String	Hash SHA-256 des données
<code>created_at</code>	DateTime	Date de création
<code>completed_at</code>	DateTime	Date de complétion

3.7 Architecture du Smart Contract

3.7.1 Structure du Contrat RouteRegistry

Le smart contract `RouteRegistry` est déployé sur la blockchain Ethereum (Ganache en développement) et assure l'immuabilité des enregistrements de tournées.

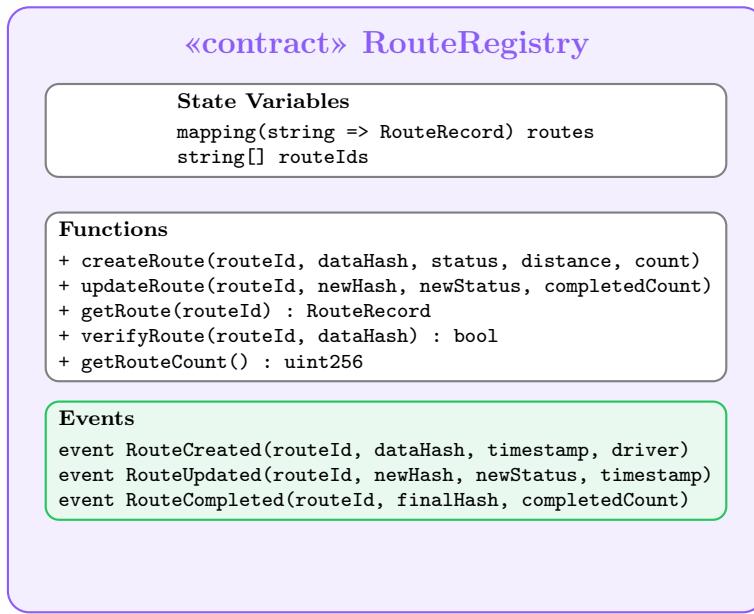


FIGURE 3.10 – Structure du smart contract RouteRegistry

3.8 Conclusion

Ce chapitre a présenté la conception complète du système RouteChain à travers différents niveaux d'abstraction. L'architecture trois-tiers garantit une séparation claire des responsabilités, tandis que les diagrammes UML offrent une vision globale et détaillée du fonctionnement du système.

Les diagrammes de séquence mettent particulièrement en évidence les deux aspects innovants de l'application : le processus d'optimisation VRP avec ses multiples composants et l'interaction avec la blockchain pour la traçabilité des données.

Le chapitre suivant sera consacré à la réalisation technique, présentant les technologies utilisées et les interfaces développées.

Chapitre 4

Réalisation

4.1 Introduction

Ce chapitre présente la réalisation technique de l'application RouteChain. Nous décrirons l'environnement de développement utilisé, puis nous détaillerons les technologies employées pour chaque couche de l'application avec leurs justifications de choix. Enfin, nous présenterons les principales interfaces utilisateur développées.

4.2 Environnement de Développement

4.2.1 Outils de Développement

TABLE 4.1 – Outils de développement utilisés

Catégorie	Outil	Version
IDE	Visual Studio Code	1.85+
Contrôle de version	Git	2.40+
Gestionnaire de paquets Python	pip	24.0+
Gestionnaire de paquets Node	npm	10.0+
Navigateur de test	Google Chrome	120+
Client API	Postman / Swagger UI	-

4.2.2 Structure du Projet

Le projet est organisé en trois répertoires principaux : `backend/` pour l'API Python, `frontend/` pour l'interface React, et `blockchain/` pour les smart contracts Solidity.

4.3 Technologies Backend

4.3.1 Python

Définition : Python est un langage de programmation interprété, multi-paradigme et dynamiquement typé. Reconnu pour sa syntaxe claire et sa lisibilité, il est largement utilisé dans le développement web, la science des données et l'automatisation.

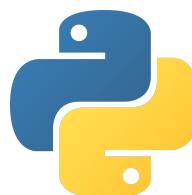


FIGURE 4.1 – Logo Python

Justification du choix :

- Écosystème riche pour le développement web (FastAPI, Flask, Django)
- Excellentes bibliothèques pour l'optimisation (OR-Tools, SciPy)
- Intégration native avec Web3.py pour la blockchain
- Syntaxe expressive accélérant le développement

4.3.2 FastAPI

Définition : FastAPI est un framework web Python moderne et performant pour la création d'APIs RESTful. Basé sur les standards OpenAPI et JSON Schema, il offre une validation automatique des données et une documentation interactive générée automatiquement.



FIGURE 4.2 – Logo FastAPI

Justification du choix :

- Performances exceptionnelles comparables à Node.js et Go
- Validation automatique via les type hints Python
- Documentation Swagger/OpenAPI générée automatiquement
- Support natif de l'asynchrone (async/await)
- Intégration facile avec Pydantic pour la sérialisation

4.3.3 MongoDB

Définition : MongoDB est une base de données NoSQL orientée documents. Elle stocke les données sous forme de documents JSON flexibles (BSON), permettant une modélisation naturelle des données et une évolutivité horizontale.



FIGURE 4.3 – Logo MongoDB

Justification du choix :

- Schéma flexible adapté aux données de tournées variables
- Stockage natif des objets imbriqués (points de livraison)
- MongoDB Atlas offre un hébergement cloud gratuit
- Driver Motor asynchrone pour de meilleures performances
- Requêtes géospatiales natives pour les coordonnées GPS

4.3.4 Google OR-Tools

Définition : Google OR-Tools est une suite open-source d'outils pour l'optimisation combinatoire. Elle inclut des solveurs pour les problèmes de programmation linéaire, de routage de véhicules (VRP), et de satisfaction de contraintes.



FIGURE 4.4 – Logo Google OR-Tools

Justification du choix :

- Solveur VRP de référence développé par Google
- Multiples stratégies d'optimisation configurables
- Support des contraintes complexes (capacité, fenêtres temporelles)
- Performances optimisées pour les problèmes à grande échelle
- Documentation complète et exemples Python

4.4 Technologies Frontend

4.4.1 React

Définition : React est une bibliothèque JavaScript développée par Meta pour la construction d'interfaces utilisateur. Basée sur le concept de composants réutilisables et le DOM virtuel, elle permet un développement efficace d'applications web interactives.

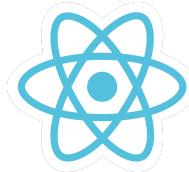


FIGURE 4.5 – Logo React

Justification du choix :

- Architecture composants favorisant la réutilisabilité
- Virtual DOM assurant des performances optimales
- Écosystème mature avec nombreuses bibliothèques
- Large communauté et documentation abondante
- Hooks simplifiant la gestion de l'état

4.4.2 Vite

Définition : Vite est un outil de build nouvelle génération pour le développement web. Il exploite les modules ES natifs du navigateur pour offrir un démarrage instantané et un Hot Module Replacement (HMR) ultra-rapide.



FIGURE 4.6 – Logo Vite

Justification du choix :

- Démarrage du serveur de développement instantané
- Hot Module Replacement en moins de 50ms
- Build de production optimisé avec Rollup
- Configuration minimale requise
- Support natif de TypeScript et JSX

4.4.3 Tailwind CSS

Définition : Tailwind CSS est un framework CSS utilitaire qui fournit des classes de bas niveau pour construire des designs personnalisés directement dans le HTML. Il privilégie la composition plutôt que l'abstraction.



FIGURE 4.7 – Logo Tailwind CSS

Justification du choix :

- Développement rapide sans quitter le HTML/JSX
- Design system cohérent avec configuration centralisée
- Purge automatique du CSS non utilisé en production
- Responsive design intégré avec préfixes (sm :, md :, lg :)
- Personnalisation complète via tailwind.config.js

4.4.4 Leaflet

Définition : Leaflet est une bibliothèque JavaScript open-source pour les cartes interactives. Légère et modulaire, elle offre toutes les fonctionnalités de cartographie nécessaires tout en restant simple à utiliser.



FIGURE 4.8 – Logo Leaflet

Justification du choix :

- Bibliothèque légère (42KB gzippée)
- Compatible avec OpenStreetMap (gratuit)
- API simple et bien documentée
- Support des marqueurs, polylinéaires et popups personnalisés
- Plugin react-leaflet pour intégration React

4.4.5 Axios

Définition : Axios est un client HTTP basé sur les promesses pour le navigateur et Node.js. Il simplifie les requêtes AJAX avec une API intuitive et des fonctionnalités avancées comme les intercepteurs.



FIGURE 4.9 – Logo Axios

Justification du choix :

- Syntaxe simple basée sur les promesses/async-await
- Intercepteurs pour gérer l'autentification JWT
- Transformation automatique des données JSON
- Gestion des erreurs centralisée
- Annulation des requêtes avec AbortController

4.5 Technologies Blockchain

4.5.1 Ganache

Définition : Ganache est un simulateur de blockchain Ethereum personnel pour le développement. Il permet de déployer des contrats, développer des DApps et exécuter des tests dans un environnement local contrôlé.



FIGURE 4.10 – Logo Ganache

Justification du choix :

- Blockchain locale instantanée sans synchronisation
- Interface graphique pour visualiser les transactions
- Comptes pré-financés pour les tests
- Mining instantané des blocs
- Persistance des données via workspaces

4.6 Présentation des Interfaces Utilisateur

4.6.1 Écrans d'Authentification

L'application RouteChain propose un système d'authentification sécurisé comprenant une page de connexion et une page d'inscription. La page de connexion présente une interface épurée et moderne, permettant aux chauffeurs de s'authentifier avec leur email et mot de passe. Le design utilise une palette de couleurs professionnelle dominée par le bleu, avec des champs de saisie clairement identifiés et un bouton de connexion bien visible. Un lien vers la page d'inscription est proposé pour les nouveaux utilisateurs.

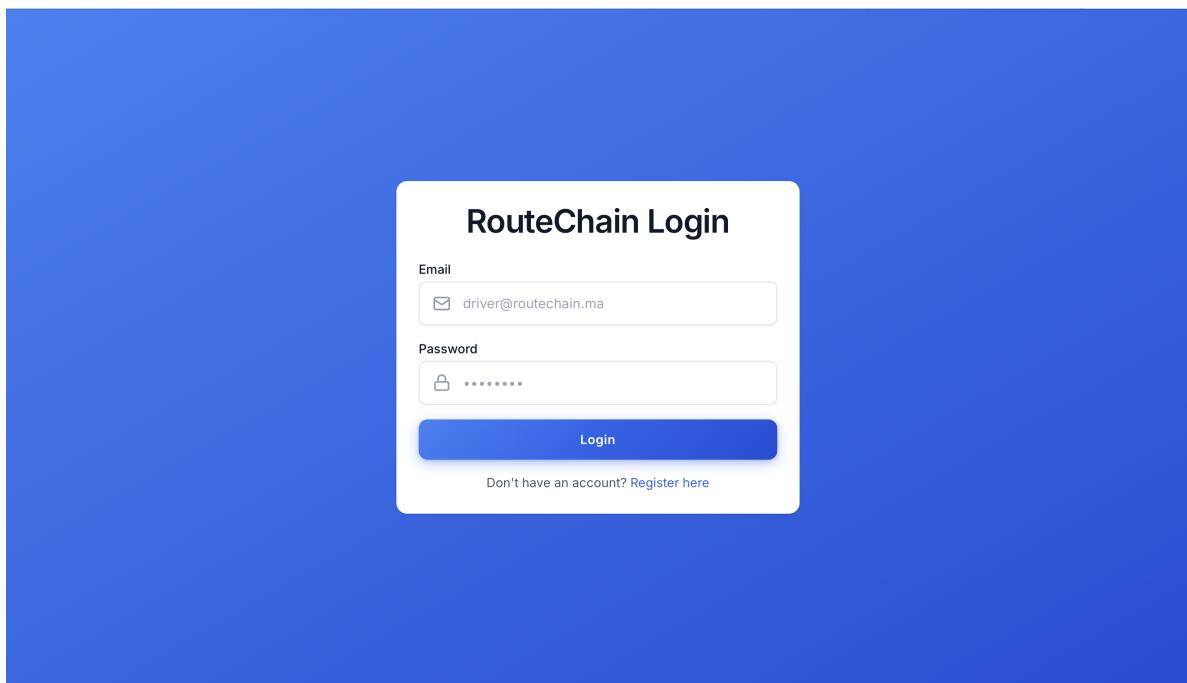


FIGURE 4.11 – Page de connexion de l'application RouteChain

Le formulaire d'inscription permet la création d'un nouveau compte chauffeur avec validation en temps réel des champs. L'interface guide l'utilisateur à travers les informations requises : nom complet, adresse email, mot de passe avec indicateur de force, numéro de téléphone au format marocain, type de véhicule (moto, voiture, van, camion) et capacité maximale de colis. Cette conception assure que les données des chauffeurs sont complètes dès leur inscription.

FIGURE 4.12 – Page d'inscription avec formulaire de création de compte chauffeur

4.6.2 Dashboard et Gestion des Tournées

Le tableau de bord principal constitue le point d'entrée après connexion. Il affiche une vue d'ensemble des tournées du chauffeur avec des indicateurs clés : nombre de tournées actives, tournées complétées aujourd'hui, et distance totale parcourue. Chaque tournée est représentée sous forme de carte avec son nom, statut (planifiée, en cours, complétée), distance totale et nombre de points de livraison. Des boutons d'accès rapide permettent de créer une nouvelle tournée, gérer les clients et consulter les dépôts.

FIGURE 4.13 – Dashboard principal affichant la liste des tournées

4.6.3 Crédation de Tournée

L'interface de création de tournée est le cœur fonctionnel de l'application. Elle se divise en trois sections principales : les informations de la tournée (nom), la sélection du dépôt de départ, et l'ajout des points de livraison. L'utilisateur peut soit sélectionner un dépôt préexistant dans la base de données, soit saisir manuellement une adresse. Le géocodage automatique via l'API Nominatim (OpenStreetMap) convertit instantanément les adresses textuelles en coordonnées GPS précises.

The screenshot shows the 'Create New Route' page. At the top, there are navigation links: Dashboard, + New Route, Analytics, More, and a user profile for 'Anas truck • No plate'. The main area has two sections: 'Route Information' (with a 'Route Name' field containing 'COD shipping - Casa') and 'Depot Location' (with a dropdown for 'Select Saved Depot' set to 'Dépôt Central Ain Sebaâ (Default) - Zone Industrielle Ain Sebaâ', an address input for 'Zone Industrielle Ain Sebaâ', and coordinates inputs for 'Latitude' (33.6036) and 'Longitude' (-7.5275)). To the right is a 'Map Preview' window showing a map of Casablanca with several red and blue location markers corresponding to the route points entered.

FIGURE 4.14 – Formulaire de création de tournée - Section dépôt et informations

La section des points de livraison permet d'ajouter entre 1 et 20 destinations. Pour chaque point, l'utilisateur peut soit sélectionner un client existant (qui remplit automatiquement les coordonnées), soit saisir un nouveau client avec son nom, téléphone, adresse et instructions de livraison. Une prévisualisation cartographique interactive affiche en temps réel tous les points sur une carte Leaflet, permettant de valider visuellement les positions avant l'optimisation.

This screenshot shows the 'Delivery Points (1/20)' section. It includes fields for 'Customer' (with 'Select Existing' and 'Enter New' options, currently showing 'Selected: Fatima Zahra Bennani'), 'Street Address' (15 Rue Allal Ben Abdellah), and coordinates ('Latitude' 33.588, 'Longitude' -7.6114). A 'Packages' field shows '2'. A note at the bottom says '⚠ Minimum 5 delivery points required for optimization'. Below the form are 'Hide Map' and 'Optimize Route' buttons. To the right is a 'Map Preview' showing the same map as Figure 4.14, with red and blue markers indicating the current state of the route.

FIGURE 4.15 – Formulaire de création de tournée - Points de livraison et carte

4.6.4 Détail de Tournée et Visualisation Cartographique

La page de détail présente toutes les informations d'une tournée optimisée. Elle affiche la liste ordonnée des points de livraison selon l'ordre optimal calculé par Google OR-Tools, avec pour chaque point : le nom du client, l'adresse, le numéro de séquence et le statut de livraison. Les statistiques globales incluent la distance totale, le temps estimé et le nombre de colis. Des boutons d'action permettent de démarrer la tournée, confirmer les livraisons et exporter les données.

The screenshot shows the RouteChain application interface. At the top, there's a navigation bar with 'RouteChain', 'Dashboard', 'New Route', 'Analytics', 'More', and a user profile for 'Anas truck + No plate'. Below the header, a breadcrumb '← Back to Dashboard' leads to the current page, 'COD Shipping - Casa', with a route ID: ROUTE_20251228_221908. A yellow button labeled 'OPTIMIZED' is visible. The main area is divided into two sections: 'Route Map' on the left, which displays a map of Casablanca with several delivery points marked by red pins, and 'Actions' on the right, which includes buttons for 'Start Route' (blue), 'Cancel Route' (orange), 'Export CSV' (blue), 'Export PDF' (blue), and 'Delete Route' (red). Below these actions is a section for 'Blockchain Verification' with a blue icon.

FIGURE 4.16 – Page de détail d'une tournée avec informations et carte

La carte interactive affiche l'itinéraire optimisé avec des marqueurs numérotés pour chaque point de livraison. Le tracé polyline suit les routes réelles grâce à l'API OpenRouteService, offrant une visualisation précise du parcours. L'interface affiche également les informations de traçabilité blockchain : hash de transaction, numéro de bloc et horodatage d'enregistrement. Le bouton de vérification permet de contrôler l'intégrité des données en comparant le hash stocké avec celui recalculé.

This screenshot shows the RouteChain application interface with a different layout. It features a 'Route Statistics' section with summary data: 12.7 km total distance, 24 min estimated duration, 5 stops, and 0 completed stops. Below this is a 'Start Navigation (Free)' section with links to Google Maps, Waze, and Apple Maps. To the right is a 'Blockchain Verification' section showing a transaction hash (0x63b4314f...), block number (6), and data hash (0x9784cded...). A green box indicates 'Data Verified' and 'Route data is valid'. At the bottom right is a 'Route Info' section with creation date (28/12/2025) and driver information (Driver: DRV_20251209231715).

FIGURE 4.17 – Visualisation cartographique et informations blockchain

4.6.5 Profil Utilisateur

La page de profil permet au chauffeur de consulter et modifier ses informations personnelles. Elle affiche le nom complet, l'adresse email, le numéro de téléphone, le type de véhicule utilisé et la plaque d'immatriculation. Des statistiques de performance sont également présentées : nombre total de tournées effectuées, distance cumulée parcourue et taux de compléction. L'utilisateur peut mettre à jour ses informations via un formulaire d'édition accessible.

The screenshot shows the RouteChain driver profile interface. At the top, there's a header with the logo, navigation links (Dashboard, New Route, Analytics, More), and user information (Anas, truck + No plate). The main area is divided into two sections: 'Profile Information' and 'Account Information'. 'Profile Information' includes fields for Full Name (Anas), Email Address (anas@gmail.com, Read-only), Phone Number (Not set), Vehicle Type (Truck), License Plate (Not set), and Max Capacity (packages) (18 packages). 'Account Information' shows Member since: December 9, 2025 and Last login: December 26, 2025. On the left, a sidebar has tabs for Profile (selected), Security, Statistics, Settings, and Logout. A summary box on the left shows 2 Routes and 22% Success.

FIGURE 4.18 – Page de profil du chauffeur avec informations personnelles

4.6.6 Gestion des Clients

L'interface de gestion des clients offre une vue complète de la base de données clients. Elle présente la liste des clients enregistrés avec leurs coordonnées : nom, email, téléphone, entreprise et adresse. Une barre de recherche permet de filtrer rapidement les clients par nom ou email. Le formulaire d'ajout intègre le géocodage automatique pour convertir les adresses en coordonnées GPS, facilitant l'ajout de nouveaux clients lors de la création de tournées.

The screenshot shows the RouteChain customer management interface. At the top, there's a header with the logo, navigation links (Dashboard, New Route, Analytics, More), and user information (Anas, truck + No plate). The main area is titled 'Customers' with a subtitle 'Manage your customer database'. It features a search bar labeled 'Search customers...'. Below it, two customer entries are listed: 'Anas' (email: anas@gmail.com, location: Casablanca) and 'Youness' (email: youness@gmail.com, location: Casablanca). Each entry includes a small profile icon, the name, email, location, delivery count (0), and edit/delete icons. A blue button '+ Add Customer' is located at the top right of the customer list area.

FIGURE 4.19 – Interface de gestion des clients avec recherche et géocodage

4.6.7 Gestion des Dépôts

La gestion des dépôts permet de définir les points de départ des tournées. L'interface affiche les dépôts enregistrés sous forme de cartes avec leur nom, adresse, coordonnées GPS et horaires d'ouverture. Un dépôt peut être marqué comme défaut pour la sélection automatique lors de la création de tournées. Le formulaire d'ajout inclut le géocodage automatique via Nominatim, permettant de convertir une adresse textuelle en latitude/longitude précises.

FIGURE 4.20 – Interface de gestion des dépôts avec géocodage automatique

4.6.8 Panel Administrateur

Le panel administrateur est réservé aux utilisateurs disposant du rôle admin. Il offre une vue d'ensemble de tous les chauffeurs enregistrés dans le système avec leurs informations : nom, email, type de véhicule et rôle actuel. L'administrateur peut modifier les rôles des utilisateurs (promotion en admin ou rétrogradation en driver), consulter l'historique complet des tournées de chaque chauffeur et gérer les accès au système.

Driver	Role	Status	Vehicle	Actions
A Admin admin@gmail.com	Admin	Available	car	
A Anas anas@gmail.com	Driver	Available	truck	
Y Youness youness@gmail.com	Driver	Available	truck	

FIGURE 4.21 – Panel d'administration avec gestion des chauffeurs

4.6.9 Dashboard Analytique

Le tableau de bord analytique présente des indicateurs clés de performance (KPI) sous forme de visualisations graphiques. Il affiche le nombre total de tournées effectuées, la distance totale parcourue, le nombre de livraisons complétées et le taux de compléction moyen. Des graphiques d'évolution temporelle permettent d'analyser les tendances sur différentes périodes. Cette interface aide les gestionnaires à évaluer l'efficacité des opérations de livraison.

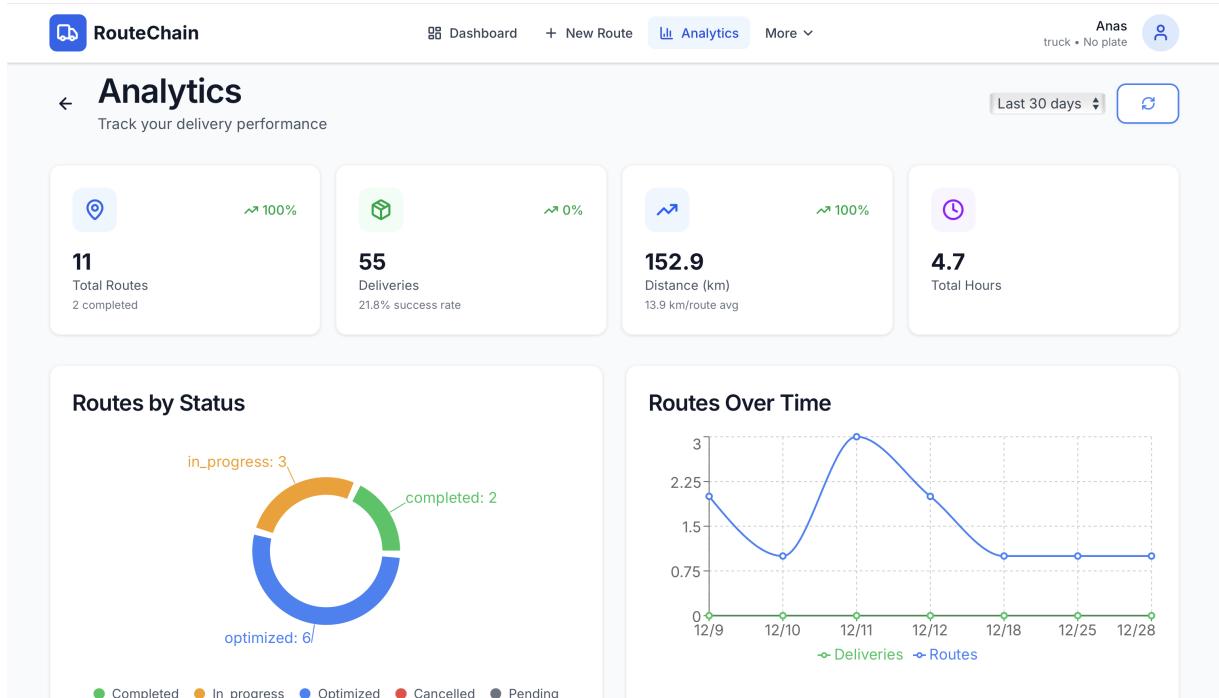


FIGURE 4.22 – Tableau de bord analytique avec indicateurs de performance

4.7 Conclusion

Ce chapitre a présenté les technologies utilisées pour le développement de RouteChain. Chaque choix technologique a été justifié en fonction des besoins spécifiques du projet : performance pour FastAPI, flexibilité pour MongoDB, puissance d'optimisation pour OR-Tools, et traçabilité pour Ganache/Ethereum.

L'architecture modulaire et les technologies modernes sélectionnées garantissent la maintenabilité et l'évolutivité du système. Le chapitre suivant proposera une discussion sur les défis rencontrés et l'évaluation du système.

Chapitre 5

Discussion et Évaluation

5.1 Introduction

Ce chapitre propose une analyse critique du projet RouteChain. Nous examinerons les défis techniques rencontrés durant le développement, les stratégies adoptées pour la gestion des données, ainsi qu'une évaluation globale du système. Nous conclurons par une discussion sur les limitations actuelles et les perspectives d'évolution.

5.2 Défis Techniques Rencontrés

5.2.1 Latence de l'Optimisation VRP

L'un des premiers défis rencontrés concerne le temps de calcul de l'algorithme d'optimisation VRP. Pour des tournées comportant un grand nombre de points de livraison (au-delà de 20 points), le temps de réponse pouvait dépasser les attentes utilisateur.

Solutions mises en œuvre :

- Limitation du temps d'exécution du solveur OR-Tools à 30 secondes maximum
- Utilisation de la stratégie PATH_CHEAPEST_ARC pour obtenir rapidement une solution initiale
- Affichage d'un indicateur de progression pendant le calcul

5.2.2 Coûts de Gas et Délais Blockchain

L'interaction avec la blockchain Ethereum, même en environnement local (Ganache), introduit des contraintes spécifiques liées aux transactions.

Problématiques identifiées :

- Chaque transaction blockchain consomme du "gas" (unité de coût computationnel)
- Les transactions nécessitent un temps de confirmation
- En cas d'erreur, les transactions échouées consomment quand même du gas

Optimisations réalisées :

- Stockage uniquement du hash des données (32 bytes) plutôt que des données complètes
- Mode dégradé permettant le fonctionnement sans blockchain disponible
- Gestion des erreurs avec retry automatique

5.2.3 Synchronisation MongoDB - Blockchain

Un défi architectural majeur réside dans la synchronisation entre les données stockées dans MongoDB et les enregistrements blockchain.

Stratégie adoptée :

- MongoDB reste la source principale des données détaillées
- La blockchain stocke uniquement les hash et métadonnées critiques
- Les informations de transaction (hash, bloc) sont enregistrées dans MongoDB pour référence

5.3 Gestion des Données et Intégrité

5.3.1 Mécanisme de Vérification par Hash

Le système de vérification d'intégrité repose sur le calcul d'un hash SHA-256 des données immuables d'une tournée.

TABLE 5.1 – Données incluses dans le calcul du hash

Champ	Justification
route_id	Identifiant unique de la tournée
route_name	Nom donné par l'utilisateur
depot_location	Coordonnées du point de départ
delivery_points (adresses et coordonnées)	Définition des livraisons

Important : Les champs mutables (statut, horodatages, livraisons confirmées) sont exclus du calcul du hash pour permettre les mises à jour légitimes sans invalider la vérification.

5.3.2 Distinction entre Données Mutables et Immuables

TABLE 5.2 – Classification des données de tournée

Catégorie	Exemples	Traitement
Immuables	route_id, points de livraison, dépôt	Inclus dans le hash blockchain
Mutables	status, completed_at, confirmations	Exclus du hash, mis à jour librement

Cette distinction permet de garantir que les données fondamentales de la tournée (qui l'a créée, où sont les livraisons) restent vérifiables, tout en autorisant l'évolution normale du statut d'exécution.

5.4 Conclusion

Ce chapitre a permis d'analyser en profondeur les défis techniques rencontrés lors du développement de RouteChain et les solutions adoptées. L'évaluation du système montre une couverture complète des exigences fonctionnelles avec des performances satisfaisantes.

Les limitations identifiées, principalement liées à l'environnement de développement (blockchain locale) et à la scalabilité, ouvrent des perspectives d'évolution intéressantes pour transformer RouteChain en une solution de production complète.

Le système actuel constitue une preuve de concept fonctionnelle démontrant la faisabilité et l'intérêt de combiner optimisation VRP et traçabilité blockchain dans le domaine de la logistique du dernier kilomètre.

Conclusion Générale

Le projet RouteChain, présenté dans ce rapport, constitue une contribution significative à l'intersection de deux domaines technologiques majeurs : l'optimisation combinatoire appliquée à la logistique et la technologie Blockchain pour la garantie d'intégrité des données. Face aux défis croissants de la logistique du dernier kilomètre, exacerbés par l'explosion du commerce électronique, nous avons développé une solution complète permettant aux entreprises de livraison d'optimiser leurs tournées tout en garantissant une traçabilité irréfutable de leurs opérations.

L'application développée répond pleinement aux objectifs initialement fixés. L'intégration de Google OR-Tools permet de calculer des séquences de livraison optimales, réduisant significativement les distances parcourues et les temps de trajet. Le smart contract RouteRegistry, déployé sur Ethereum, assure l'enregistrement immuable des données de chaque tournée, offrant une preuve vérifiable de l'intégrité des informations. L'interface utilisateur moderne, développée avec React et Tailwind CSS, offre une expérience fluide tant sur desktop que sur mobile. Enfin, l'architecture évolutive basée sur FastAPI et MongoDB garantit la maintenabilité et l'extensibilité du système.

D'un point de vue technique, ce projet nous a permis d'approfondir nos compétences dans plusieurs domaines : le développement full-stack avec des frameworks modernes, la résolution de problèmes d'optimisation combinatoire, et le développement de smart contracts pour la blockchain Ethereum.

Cependant, la solution actuelle présente certaines limitations qu'il convient de mentionner. L'utilisation de Ganache comme blockchain locale limite le déploiement à un environnement de développement, car un passage vers un testnet ou mainnet Ethereum nécessiterait une gestion des coûts de gas réels. L'algorithme VRP implémenté, bien que performant pour des tournées de taille moyenne, devrait être optimisé ou distribué pour gérer des flottes multi-véhicules ou des centaines de points de livraison. Par ailleurs, le système ne propose pas de suivi GPS en temps réel des véhicules, et son fonctionnement optimal dépend de la disponibilité des APIs externes OpenRouteService et Nominatim.

Ces limitations ouvrent néanmoins des perspectives d'évolution prometteuses. Le déploiement sur une blockchain publique, en commençant par un testnet Ethereum comme Sepolia avant d'envisager le mainnet, constitue une première étape vers une solution de production. Le développement d'une application mobile native avec Flutter ou React Native, intégrant le GPS du smartphone, améliorerait considérablement l'expérience des chauffeurs sur le terrain. L'intégration de modèles de machine learning pour la prédiction des temps de livraison et l'optimisation dynamique représente une piste d'amélioration significative. L'extension du solveur VRP pour gérer des flottes multi-véhicules avec contraintes de capacité élargirait le champ d'application de la solution. Enfin, le développement de connecteurs pour les systèmes de gestion d'entreprise (SAP, Odoo) faciliterait l'intégration dans les écosystèmes existants.

En définitive, RouteChain démontre la faisabilité et la pertinence de combiner optimisation algorithmique et technologie blockchain pour répondre aux enjeux contemporains de la logistique. Cette approche hybride, alliant efficacité opérationnelle et transparence des données, représente une voie d'avenir pour le secteur de la livraison du dernier kilomètre.

Bibliographie

- [1] Volodymyr Agafonkin. Leaflet : Open-source javascript library for interactive maps. <https://leafletjs.com>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [2] George B. Dantzig and John H. Ramser. The truck dispatching problem. *Management Science*, 6 :80–91, 1959. Consulté entre le 21 novembre et le 25 décembre 2025.
- [3] Ethereum Foundation. Ethereum : A decentralized platform for smart contracts. <https://ethereum.org>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [4] Ethereum Foundation. Solidity programming language. <https://soliditylang.org>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [5] Ethereum Foundation. Web3.py : Python library for ethereum. <https://web3py.readthedocs.io>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [6] OpenStreetMap Foundation. Nominatim : Openstreetmap geocoding. <https://nominatim.org>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [7] Google. Google or-tools. <https://developers.google.com/optimization>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [8] HeiGIT. Openrouteservice : Routing api. <https://openrouteservice.org>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [9] MongoDB Inc. Mongodb : Document database. <https://www.mongodb.com>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [10] Tailwind Labs. Tailwind css : Utility-first css framework. <https://tailwindcss.com>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [11] Meta. React : A javascript library for building user interfaces. <https://react.dev>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [12] Truffle Suite. Ganache : Personal ethereum blockchain. <https://trufflesuite.com/ganache>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [13] Sebastián Ramírez Tiangolo. Fastapi : Modern web framework for python apis. <https://fastapi.tiangolo.com>, 2025. Consulté entre le 21 novembre et le 25 décembre 2025.
- [14] Paolo Toth and Daniele Vigo. *Vehicle Routing : Problems, Methods, and Applications*. SIAM, 2014. Consulté entre le 21 novembre et le 25 décembre 2025.