INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

INSA Lyon

Département Informatique

PLD AGILE : Optimod'Lyon
Premier livrable

Auteur : H4302
Hazim Asri
Nihal Boutadghart
Jassir Habba
Ana Martin
Junior Noukam
Simon Perret

Professeur :
Mme. Laforest

8 décembre 2024

Table des matières

Introduction

Ce document constitue le livrable de la première itération de notre projet, qui correspond à la phase de démarrache selon la méthodologie USDP (Unified Software Development Process). L'objectif principal de cette phase est de poser les bases de l'application en identifiant les cas d'utilisation majeurs, en concevant une première architecture, et en mettant en œuvre un ensemble limité de cas d'utilisation pour aboutir au MVP, une premiere version limitée mais fonctionnel.

Au cours de cette itération, nous avons concentré nos efforts sur le glossaire, les différents diagrammes tels que le diagramme des cas d'utilisation, le diagramme de classe et package, la mise en place du serveur git, le MVP et encore la documentation.

Le résultat de cette itération est un MVP (Minimum Viable Product) accompagné d'un livrable pour illustrer les avancées réalisées. Ces livrables incluent des diagrammes UML détaillant la conception, une documentation claire des choix architecturaux effectués, ainsi qu'un README structuré pour orienter les utilisateurs et les contributeurs.

1 Glossaire

— **Coord** : Représente une paire de coordonnées géographiques (latitude et longitude). Elles fournissent des informations géographiques pour localiser les intersections.
— **Adjacent** : Modélise une connexion entre deux intersections et permet de modéliser le réseau routier.
— **Intersection** : Représente un point du réseau routier, identifié par ses coordonnées géographiques. Sert de noeud dans le réseau routier.
— **Map** : Représente la carte complète du réseau routier et fournit une vue d'ensemble du réseau routier.
— **DeliveryRequest** : Modélise une demande de livraison avec des points de collecte et de livraison. Structure les informations nécessaires à une livraison.
— **Warehouse** : Représente un entrepôt dans le système. Point central pour les départs des tournées.
— **TourRequest** : Regroupe une liste de demandes de livraison associée à un entrepôt. Prépare et organise les demandes de livraison pour une tournée.
— **Courier** : Modélise un livreur.
— **Tour** : Définit une tournée complète incluant plusieurs points d'intersections.

2 Diagrammes UML

2.1 Diagramme des cas d'utilisation

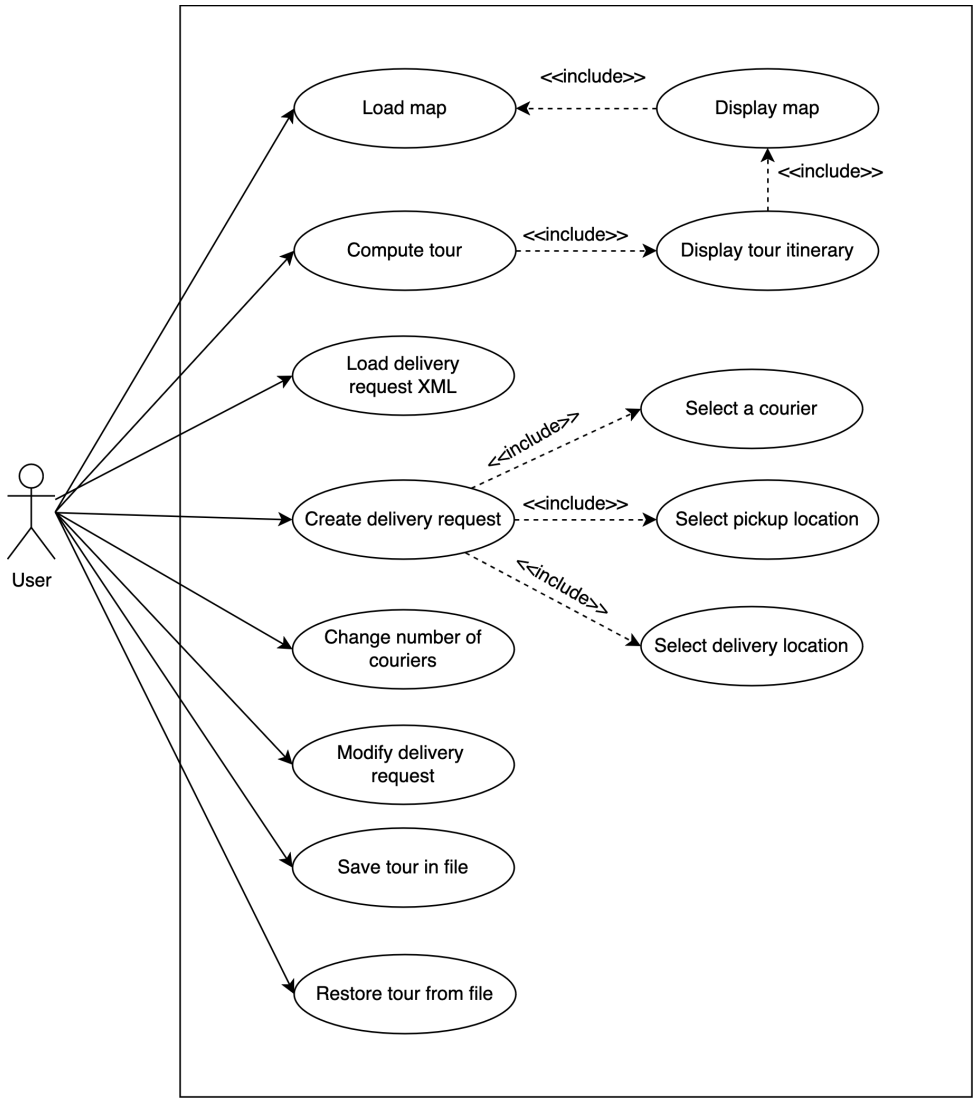


FIGURE 1 – Diagramme des cas d'utilisation

2.2 Diagramme de classe

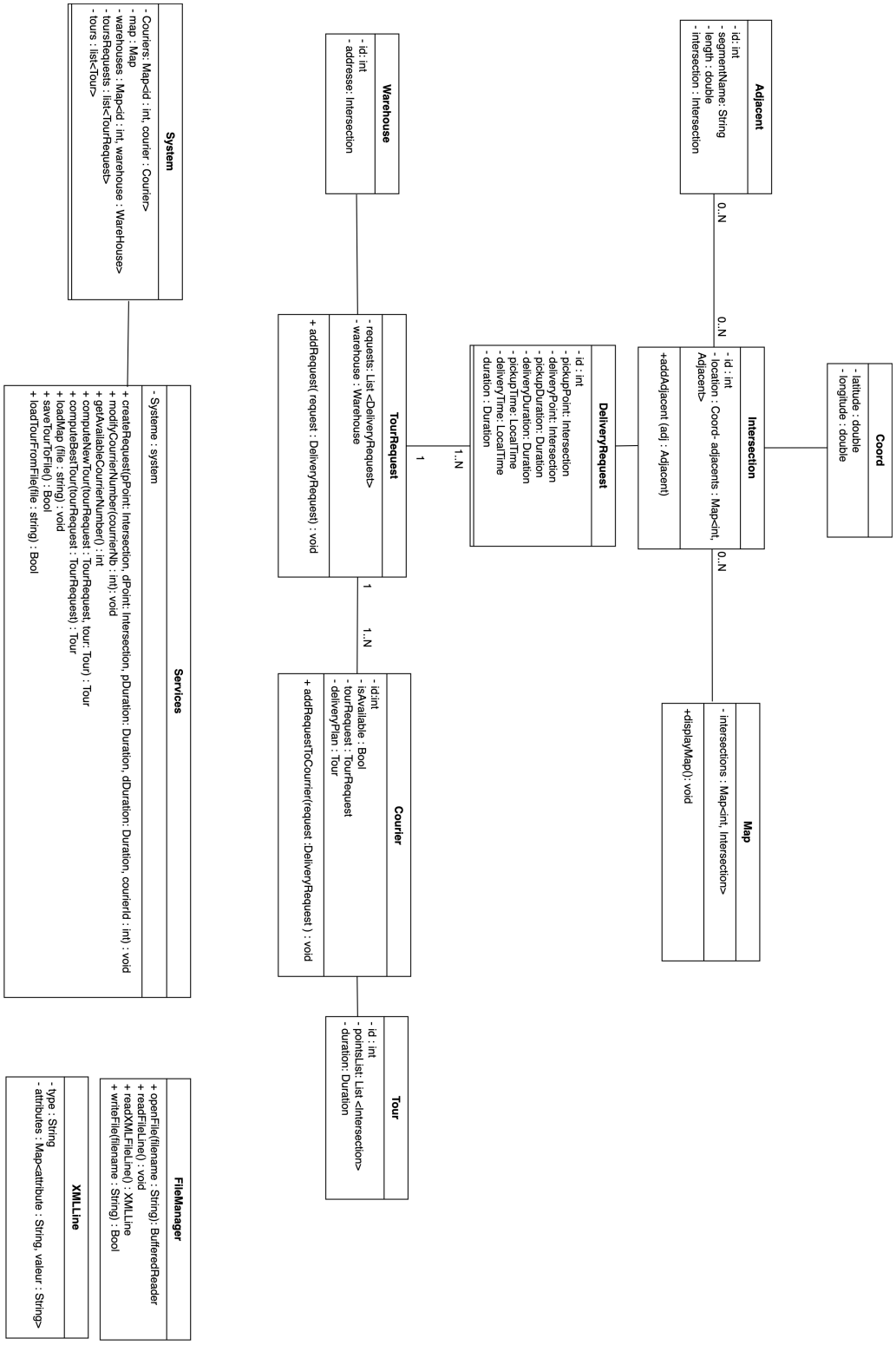


FIGURE 2 – Diagramme de classe

2.3 Package de classe

2.4 Diagramme de séquence

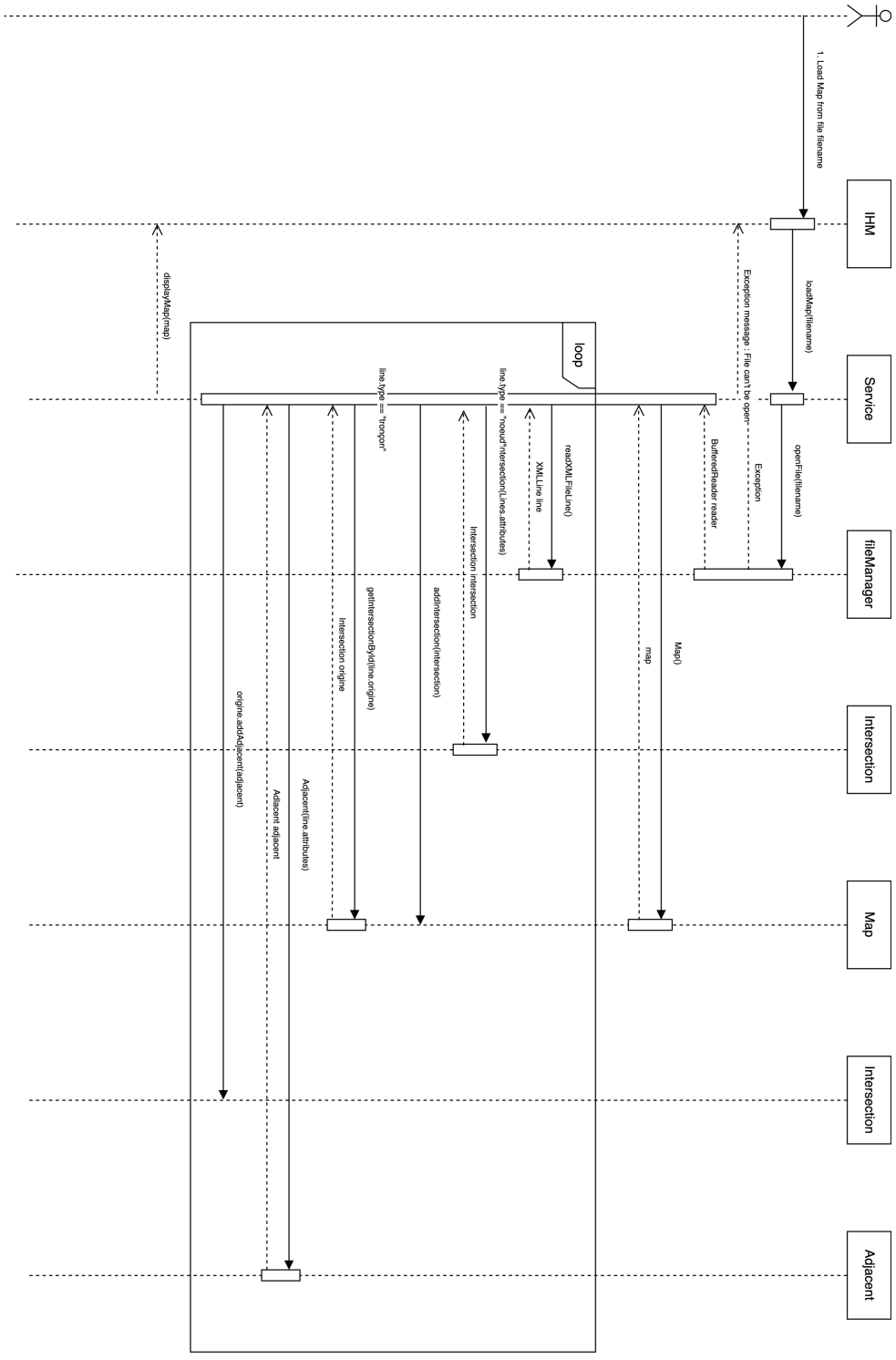
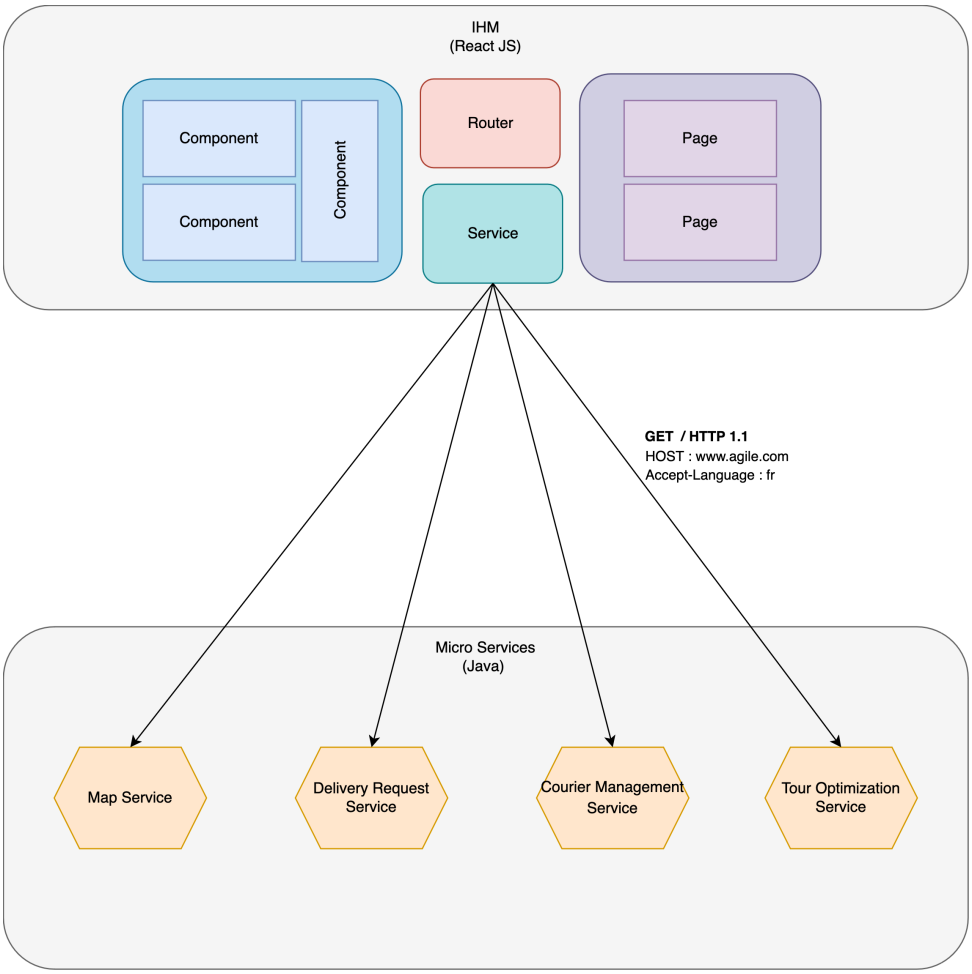


FIGURE 3 – Diagramme de séquence pour chargement de carte

3 Architecture



Notre architecture repose sur une structure en microservices, où le backend, développé en Java, est constitué de plusieurs services spécialisés : le Map Service, le Delivery Request Service, le Courier Management Service, et le Tour Optimization Service. Chaque service est indépendant et se concentre sur une fonctionnalité spécifique, ce qui garantit modularité, évolutivité et facilité de maintenance.

Le frontend, développé en ReactJS, suit une approche modulaire avec des composants, pages, et un routeur pour gérer la navigation. Le lien entre le frontend et le backend est assuré par un système d'API REST, qui permet une communication standardisée à travers des requêtes HTTP. Les services exposent des endpoints pour échanger des données avec le frontend, par exemple via des requêtes GET, en utilisant des protocoles courants comme HTTP 1.1.

Cette séparation claire entre le frontend et le backend, et l'utilisation d'API pour leur interaction, garantit la flexibilité de l'application et facilite l'intégration de nouvelles fonctionnalités ou services dans le futur.

4 Planning

Tâche	Assignement	Durée	Date de début	Date de fin
Use case	All	01:00:00	25/11/2024	25/11/2024
Architecture	Jassir, Hazim	00:30:00	25/11/2024	25/11/2024
Diagramme de classe	Nihal, Junior, Ana	02:00:00	25/11/2024	02/12/2024
Implémenter map, intersections	Simon	00:30:00	27/11/2024	27/11/2024
Map using Google API	Hazim	00:45:00	27/11/2024	27/11/2024
Documenter code pour MVP	Simon	00:15:00	02/12/2024	02/12/2024
Design Document	Simon	00:15:00	02/12/2024	02/12/2024
Glossaire	Simon	00:15:00	02/12/2024	02/12/2024
Package	Nihal, Junior, Ana	00:15:00	02/12/2024	02/12/2024
Bouton pour charger XML	Ana	01:00:00	02/12/2024	02/12/2024
FileManager (backend)	Jassir, Hazim	04:00:00	27/11/2024	02/12/2024
Diagramme de séquence	Nihal, Junior, Ana	02:00:00	02/12/2024	02/12/2024

FIGURE 4 – Planning des tâches

Conclusion

Au terme de ce premier sprint, nous avons concentré nos efforts sur la phase de conception, afin d'assurer une architecture solide et cohérente pour le développement futur de notre application. Bien que cette approche ait renforcé les fondations techniques du projet, elle a également réduit le temps disponible pour le développement, ce qui nous a empêchés de finaliser toutes les fonctionnalités initialement prévues.

Malgré ces contraintes, nous avons réussi à produire un MVP fonctionnel permettant d'afficher une carte grâce à l'intégration de l'API Google Maps. Cette base visuelle et interactive constitue un point de départ robuste pour les prochaines itérations. Cependant, la fonctionnalité de chargement de cartes depuis un fichier XML, initialement prévue dans ce sprint, n'a pas pu être implémentée dans les délais impartis.

Cette itération a mis en lumière l'importance d'équilibrer les phases de conception et de développement pour les prochains sprints. Elle nous a également permis d'identifier des axes d'amélioration organisationnelle. Par exemple, nous avons tous pris conscience de l'importance d'utiliser efficacement le système de gestion de version Git, ce qui a permis une meilleure collaboration en équipe. En revanche, nous avons perdu du temps en raison de tâches bloquantes, qui ont freiné les autres membres de l'équipe dans leur travail.

Pour remédier à cela, nous prévoyons de définir des tâches plus courtes et plus simples dans les prochains sprints, afin de faciliter des commits réguliers. Cette approche devrait améliorer notre cadence et limiter les temps d'attente.