PLD AGILE : Dossier de Cadrage, Suivi et Bilan de Projet

Youssef LAATAR, Mohamed FAKRONI, Yousef CHAOUKI, Djalil CHIKHI, Mehdi GRIGUER

Hexanôme 44

Contents

1 Phase de préparation du projet

1.1 Description du projet

Le projet consiste à développer une application innovante qui vise à optimiser les tournées de livraison en milieu urbain. En intégrant des algorithmes avancés de graphes et une interface utilisateur intuitive, cette solution permettra de planifier efficacement les trajets de livraison

tout en respectant les contraintes spécifiques liées au temps, à la distance et aux priorités des clients. Il s'inscrit dans la cadre du projet longue durée AGILE.

1.2 Contexte et Justification

1.2.1 Origine du projet

Le projet est né d'un besoin croissant d'optimisation des processus logistiques dans un contexte urbain complexe. Avec l'essor du commerce en ligne et des services de livraison à la demande, les entreprises font face à des défis importants liés à la gestion des tournées, aux contraintes de temps, et à l'efficacité des itinéraires. Ces problématiques, combinées à la nécessité de réduire l'impact environnemental des transports, ont mis en évidence l'importance de solutions numériques innovantes pour répondre à ces exigences.

1.2.2 Problématique à résoudre

Les entreprises de livraison doivent régulièrement relever des défis tels que :

- **Complexité des itinéraires :** Trouver le chemin optimal entre plusieurs points de collecte et de dépôt en tenant compte des contraintes de temps et de distance.
- **Limitation des ressources :** Utiliser efficacement les véhicules et les ressources humaines disponibles tout en respectant les contraintes budgétaires.
- **Environnement urbain imprévisible :** Faire face à des perturbations telles que le trafic, les restrictions de circulation, ou les changements de dernière minute.
- **Satisfaction client :** Maintenir un haut niveau de satisfaction grâce à des délais respectés et une communication claire.

Ces défis nécessitent une approche technique et organisationnelle robuste pour garantir la fluidité des opérations.

1.2.3 Opportunités identifiées

Ce projet représente une opportunité de répondre à ces enjeux en proposant une application qui :

1. **Optimise les tournées de livraison** grâce à des algorithmes avancés de graphes capables de générer des itinéraires efficaces en temps réel.
2. **Améliore la gestion des ressources** en permettant aux entreprises de maximiser l'utilisation des véhicules et des coursiers, tout en réduisant les coûts opérationnels.
3. **Renforce l'engagement environnemental** en minimisant les trajets inutiles et en promouvant des itinéraires économes en carburant.
4. **Offre une interface intuitive** permettant aux utilisateurs de visualiser et de modifier facilement les itinéraires selon leurs besoins.

Ce projet s'inscrit dans une dynamique de transformation numérique, visant à rendre les services de livraison plus intelligents, durables, et accessibles à un large éventail d'entreprises.

2 Méthode et phasage du projet

2.1 Méthodologie choisie

Le projet sera mené en utilisant une approche **Agile**, plus précisément la méthodologie **Scrum**. Ce choix permet de répondre efficacement aux besoins évolutifs du projet et d'assurer une collaboration étroite entre les parties prenantes. Les principaux avantages de cette méthodologie sont :

- Une livraison itérative et incrémentale des fonctionnalités.
- Une meilleure réactivité face aux changements de priorités ou aux retours du client.
- Une communication continue entre les membres de l'équipe grâce à des cérémonies régulières (meetings, rétrospectives, etc.).

2.2 Phases principales du projet

Le projet est structuré en quatre sprints, chacun ayant des objectifs spécifiques permettant une progression incrémentale vers la finalisation de l'application.

1. **Sprint 1 : Découverte du sujet et conception de l'application**
   - Analyse approfondie du sujet et identification des besoins métiers.
   - Conception de l'architecture logicielle (back-end, front-end, base de données).
   - Élaboration des maquettes et prototypes pour l'interface utilisateur.
2. **Sprint 2 : Implémentation du squelette de l'application (MVP)**
   - Développement du squelette fonctionnel de l'application.
   - Mise en place des bases du back-end et du front-end.
   - Intégration des premières API.
3. **Sprint 3 : Implémentation des fonctionnalités clés de l'application**
   - Développement des fonctionnalités principales (optimisation des tournées, visualisation des trajets, export des données).
   - Tests fonctionnels complets et validation des scénarios d'utilisation.
4. **Sprint 4 : Correction des bugs et implémentation des fonctionnalités mineures**
   - Identification et correction des bugs identifiés lors des premiers tests.
   - Développement de fonctionnalités mineures et amélioration de la qualité de vie de l'application (exemple : import de fichier pas réduit à un dossier, amélioration de la rapidité de chargement des cartes, etc...)

2.3 Livrables attendus

Le livrable final du projet inclue :

- **Documentation initiale :** Analyse fonctionnelle, architecture logicielle, User stories.

- **Prototypes et maquettes :** Maquettes figma initiale de l'interface utilisateur.
- **Application fonctionnelle :** Code source de la version finale.
- **Documentation utilisateur :** Guide de démarrage et manuel d'utilisation.
- **Diagramme de suivi de projet :** Burndown Chart, Tableau d'organisation du projet (Trello).

3 Organisation de l'équipe

L'organisation de l'équipe est cruciale pour assurer le bon déroulement du projet. L'équipe est structurée de manière à favoriser la collaboration, la communication et l'efficacité tout au long des différentes phases du projet.

3.1 Rôles et responsabilités

- **Chef de Projet :** Responsable de la planification, de la coordination et de la supervision de l'ensemble du projet. Il veille à ce que les objectifs soient atteints dans les délais et respecte le budget.
- **Développeurs Front-End :** Chargés de la conception et du développement de l'interface utilisateur (React, cartes interactives, etc.), en veillant à une expérience utilisateur fluide.
- **Développeurs Back-End :** Gèrent la logique métier, les API, et l'intégration des algorithmes d'optimisation dans l'application.
- **Responsable Qualité :** Supervise les phases de test et s'assure que les livrables respectent les critères définis (tests unitaires, fonctionnels, et de performance).

3.2 Structure hiérarchique et parties prenantes

- Le chef de projet coordonne les différentes équipes fonctionnelles (Front-End, Back-End tout en facilitant la communication entre elles.
- Des réunions régulières sont organisées pour faire le point sur l'avancement du projet et ajuster les priorit´es si n´ecessaire.

3.3 Collaboration et communication

Pour garantir une communication fluide et efficace, l'équipe utilise des outils modernes et des processus collaboratifs :

- **Réunions régulières :**
  - **Meetings réguliers :** Suivi ponctuel des progrès.
  - **Rétrospectives de début de sprint :** Avant chaque nouveau sprint, une rétrospective est réalisée pour :
    - Réévaluer les priorités du projet en fonction des retours et des évolutions.
    - Identifier les points d'amélioration à appliquer dans le sprint suivant.

    * Planifier les tâches à réaliser, en définissant clairement les objectifs du sprint.

- **Outils collaboratifs :**
  - **Slack :** Communication instantanée au sein de l'équipe.
  - **Trello**[3]**:** Gestion des tâches, backlog, et suivi des sprints.
  - **GitHub**[1]**:** Collaboration sur le code source et gestion des versions.

4 Technologies et architecture de l'application

4.1 Technologies utilisées

L'application repose sur un ensemble de technologies modernes permettant de garantir sa robustesse, sa scalabilité et sa maintenabilité. Voici un aperçu des technologies sélectionnées :

- **Front-end :** Développé avec React pour bénéficier d'une interface utilisateur dynamique et réactive, associée à des bibliothèques comme Leaflet pour la visualisation des cartes interactives.
- **Back-end :** Développé avec Spring Boot, un framework Java robuste et performant, permettant de structurer le back-end de manière modulaire et maintenable. Spring Boot offre une intégration fluide pour la création d'API REST, la gestion de la logique métier, et l'interaction avec la base de données MySQL.
- **Base de données :** MySQL a été choisi pour stocker les données de l'application, grâce à sa simplicité, sa robustesse, et son efficacité pour gérer des structures relationnelles complexes.
- **Outils de développement :**
  - **GitHub :** Pour la gestion des versions et la collaboration en équipe.
  - **Postman**[2]**:** Pour tester et documenter les API.
  - **Visual Studio Code :** Environnement de développement utilisé par les développeurs Front-end.
  - **IntelliJ :** Environnement de développement utilisé par les développeurs Back-end.

4.2 Architecture technique

L'architecture de l'application repose sur un modèle **Modèle-Vue-Contrôleur (MVC)**, permettant une séparation claire des responsabilités entre les différentes couches de l'application. Cette approche garantit une meilleure maintenabilité, évolutivité, et organisation du code.

- **Modèle (Model) :** Représente la couche de gestion des données et la logique métier. Elle inclut :
  - Les entités principales de l'application (utilisateurs, itinéraires, livraisons, etc.).
  - La communication avec la base de données MySQL à travers des services gérés par Spring Boot.

- **Vue (View) :** Responsable de l'interface utilisateur, développée avec React. Cette couche permet :
  - L'affichage dynamique des données (par exemple, itinéraires optimisés, cartes interactives).
  - Les interactions utilisateurs, via des boutons ou carte intégrée à l'aide de la bibliothèque Leaflet.
- **Contrôleur (Controller) :** Sert d'intermédiaire entre la vue et le modèle, gérant les requêtes entrantes et les réponses sortantes. Il est implémenté avec Spring Boot et expose :
  - Des API REST permettant au front-end de communiquer avec le back-end.
  - La logique de traitement des données, notamment la gestion des itinéraires et des livraisons.

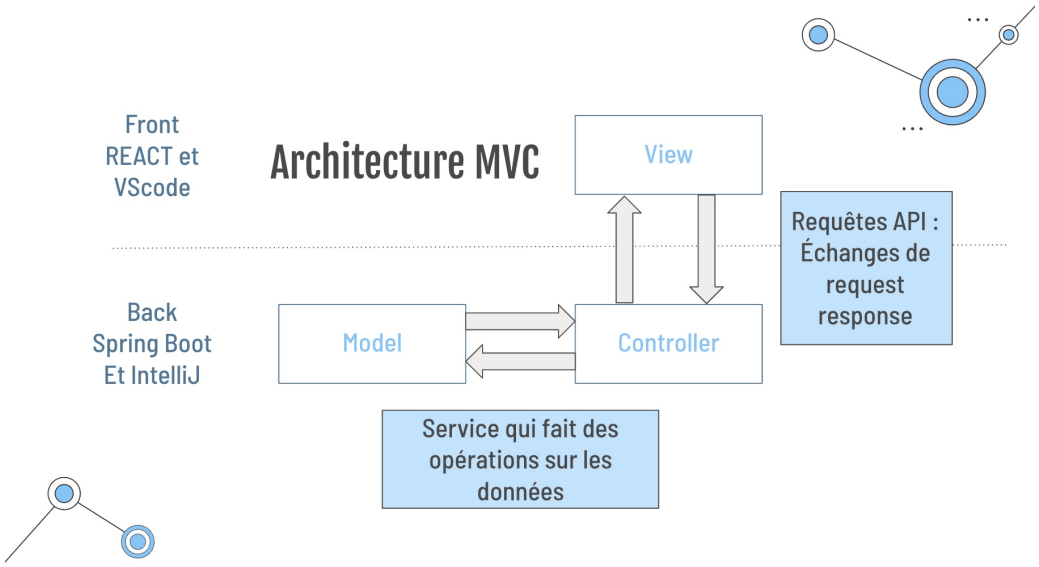

Figure 1: Architecture technique de l'application

Cette architecture MVC assure une séparation des préoccupations, facilitant ainsi le développement collaboratif et les tests indépendants de chaque couche.

5 Fonctionnalités de l'application

5.1 Liste des fonctionnalités principales

L'application offre un ensemble de fonctionnalités clés destinées à répondre aux besoins identifiés.

5.1.1 Fonctionnalités essentielles

- **Optimisation des tournées de livraison :** Utilisation d'algorithmes avancés pour générer des itinéraires optimisés en fonction des contraintes de temps et de distance.

- **Visualisation des itinéraires :** Affichage des trajets optimisés sur une carte interactive, permettant une meilleure compréhension et un suivi des tournées.
- **Gestion des livraisons :** Ajout et suppression de points de collecte et de livraison via une interface intuitive.
- **Import et export des données :** Possibilité d'exporter les trajets sous forme de fichiers standardisés (XML) pour intégration avec d'autres outils.

5.2 Expérience utilisateur (UX/UI)

L'interface utilisateur a été conçue pour être intuitive et efficace, avec une attention particulière portée à l'ergonomie et à la fluidité d'utilisation.

- **Interface visuelle interactive :** La carte interactive permet de visualiser les trajets, d'ajouter des points de collecte et de livraison, et de modifier les itinéraires.
- **Simplicité d'utilisation :** Des formulaires clairs et des boutons bien définis permettent aux utilisateurs de naviguer facilement dans l'application, même sans formation technique préalable.
- **Réactivité :** L'application s'adapte rapidement aux actions des utilisateurs grâce à des mises à jour instantanées des données affichées.

6 Difficultés rencontrées et solutions apportées

6.1 Difficultés techniques

Le projet a présenté plusieurs défis techniques, notamment :

- **Apprentissage de nouvelles technologies :** L'utilisation de Spring Boot et React ainsi que la gestion de la communication entre le front-end et le back-end ont nécessité un temps d'adaptation.
- **Débogage complexe :** Certaines erreurs liées à l'intégration des API REST ont été difficiles à identifier et à corriger, ce qui a retardé certaines fonctionnalités.
- **Algorithmes complexes:** Le développement d'un algorithme complet et de ce fait complexe était nécessaire pour optimiser au mieux les résultats et les itinéraires de livraison. Son développement laborieux a entraîné des légers retards.

6.2 Difficultés organisationnelles

Les aspects organisationnels ont également posé des défis :

- **Manque de clarté initiale :** La répartition des tâches n'était pas clairement définie au début du projet, ce qui a conduit à des chevauchements.
- **Synchronisation de l'équipe :** Les phases critiques ont parfois souffert d'un manque de communication entre les membres.

- **Estimation temporelle des tâches :** Certaines fonctionnalités ont nécessité davantage de temps que prévu en raison de leur complexité technique ou un manque de comfort avec les technologies utilisées.
- **Manque de clarté des priorités :** Un manque de consultation du Product Owner sur les premiers sprints a entraîné des confusions sur les priorités, retardant certaines tâches clés. Des réunions plus régulières ont ensuite permis d'y remédier.

6.3 Solutions apportées

Pour surmonter ces obstacles, plusieurs mesures ont été prises :

- **Formation rapide :** Une documentation ciblée et des tutoriels ont été utilisés pour accélérer la prise en main des technologies.
- **Réunions de synchronisation :** Des réunions quotidiennes ont permis de clarifier les priorités et de suivre les progrès.
- **Optimisation du workflow :** L'utilisation d'outils comme Trello et GitHub a facilité la gestion des tâches et la collaboration.
- **Exploitation des forces des membres :** L'équipe a réparti les tâches en s'appuyant sur les compétences et affinités de chacun. Les membres spécialisés en back-end se sont concentrés sur les API, tandis que ceux plus à l'aise avec le design ont pris en charge le front-end. Cette approche a permis d'avancer plus efficacement et de respecter les délais.

References

[1] GitHub. Référentiel du code source du projet, 2024. URL du back : https://github.com/mehdigriguer/ubereats-clone et du front : https://github.com/mflwu/agile-front. URL: https://github.com/yolaatar/PLD-AGILE.

[2] Postman. Collection des api du projet, 2024. URL: https://app.getpostman.com/join-team?invite_code=f8400535894f0678612b691e3c55b1977a37490b52c535025db0cb0e354d293target_code=275f783ba686e6d0d821ae3eec5fb4c9.

[3] Trello. Tableau de gestion des tâches du projet, 2024. URL: https://trello.com/invite/b/67457d467d52f7b608431600/ATTI63a71301dec19b5ff85a4f6ca65538f0E5DBCC1C/pld-agile.