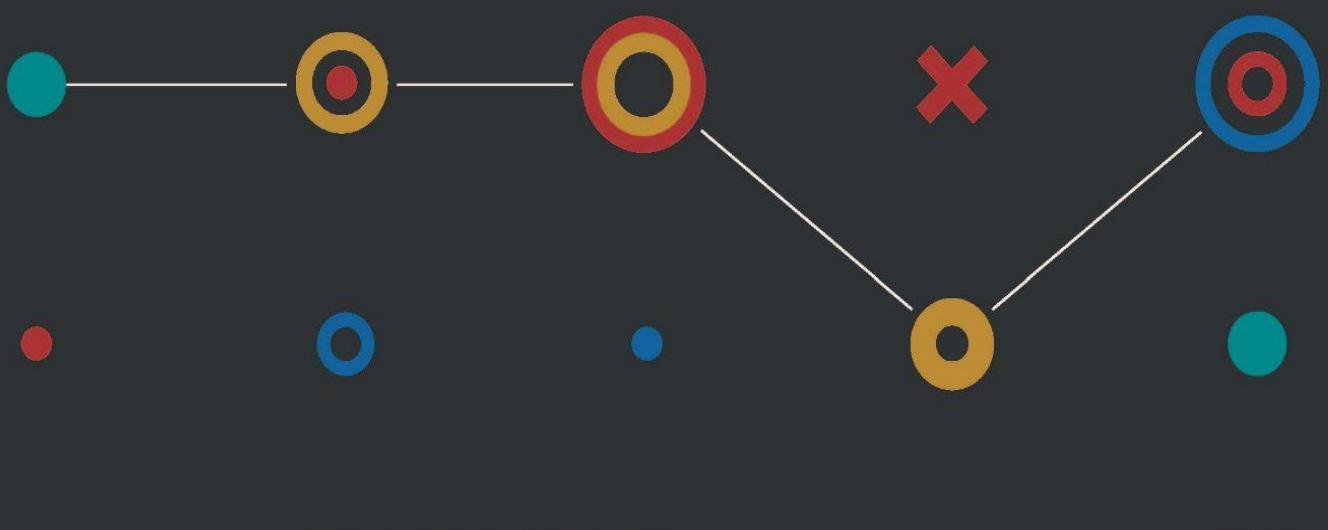


# PLD Agile

## Compte-rendu du projet

Equipe numéro **H4124**

**Arrivé** Maël  
**Flandre** Corentin  
**Hmidi** Mohamed  
**Poulet** Simon  
**Reis** Alexis  
**Thomas** Colin  
**Ugnon-Coussioz** Eva



# Sommaire

|  |           |
|--|-----------|
| <b>1. Introduction - Mise en contexte</b>      | <b>3</b>  |
| <b>2. Design Pattern</b>                       | <b>5</b>  |
| 2.1. State Pattern                             | 5         |
| 2.2. Command Pattern                           | 5         |
| 2.3. Observable Pattern                        | 5         |
| <b>3. Diagrammes</b>                           | <b>6</b>  |
| 3.1. Diagramme package                         | 6         |
| 3.2. Diagramme global                          | 6         |
| 3.3. Diagrammes modèle                         | 6         |
| 3.4. Diagrammes contrôleur                     | 7         |
| 3.5. Diagramme vue                             | 7         |
| <b>4. Maquettes</b>                            | <b>7</b>  |
| 4.1. Maquette scénario                         | 7         |
| 4.2. Captures d'écrans de Deliverapp           | 9         |
| <b>5. Tests Unitaires et Algo</b>              | <b>10</b> |
| 5.1. Algorithmie                               | 10        |
| 5.2. Test Unitaire Algorithmes                 | 10        |
| 5.3. TU Chargement Plan                        | 12        |
| 5.4. Implémentation des tests                  | 12        |
| <b>6. Partie gestion de projet</b>             | <b>13</b> |
| 6.1. Projet Agile                              | 13        |
| 6.2. Outils utilisés                           | 15        |
| 6.3. Planning du projet                        | 16        |
| 6.4. Retour de la part de chaque collaborateur | 17        |
| 6.5. Axes d'améliorations                      | 20        |
| <b>7. Annexes</b>                              | <b>20</b> |
| 7.1. Acronyme                                  | 20        |
| 7.2. Glossaire                                 | 20        |
| 7.3. Diagramme d'utilisation                   | 22        |

# 1. Introduction - Mise en contexte

L'objectif de ce projet est de développer une application, Deliverapp, permettant à un manager de livreurs de générer des feuilles de route pour ses livreurs. L'application permet de charger une carte, d'ajouter, pour les points de cette carte, des livraisons à certains horaires et d'ensuite calculer des trajets optimaux pour organiser les tournées des livreurs.

Dans ce projet conséquent, en plus des connaissances techniques que nous avons apprises, l'organisation du groupe et le travail d'équipe ont été une part importante du développement. Afin de mieux nous organiser dans ce groupe de 7 personnes, nous nous sommes attribués des rôles afin d'avoir des référents en fonction des domaines :

|   |                 |
|---|-----------------|
| <b>Chef de projet</b>                           | Eva             |
| <b>Développeur front-end</b>                    | Simon, Maël     |
| <b>Expert configuration (IDE, Git)</b>          | Alexis          |
| <b>Expert lecture de la carte et sauvegarde</b> | Mohamed         |
| <b>Expert algorithme de calcul de tournée</b>   | Corentin, Colin |

Pour mieux se repérer dans l'application, voici la liste des fonctionnalités développées :

| Fonctionnalités essentielles                             | Description   |
|--|---|
| Charger une carte  | Charger une carte dans l'application (fichier xml), repérage d'erreurs dans le fichier xml.   |
| Ajouter/Supprimer des livraisons                         | Créer des livraisons, choisir leur point de livraison, leur horaire, et leur livreur. Pouvoir les supprimer.  |
| Calculer une tournée                                     | Calculer l'ordre optimisé de livraisons effectuées, ainsi que les trajets à effectuer entre chaque.   |
| Pouvoir annuler/refaire une action                       | Annuler ou refaire une action (ajouter/supprimer une livraison avant/après le calcul de tournée)  |
| Ajouter/Supprimer des livraisons après calcul de tournée | Créer une livraison après le calcul de tournée, en choisissant l'emplacement, l'horaire, le livreur, ainsi que la livraison qui précédera la nouvelle, en recalculant les trajets et les horaires des livraisons qui ont changé. Supprimer une livraison en |

|                             |   |
|-----------------------------|---|
|                             | recalculant les trajets et les horaires de livraisons qui ont changé.   |
| Générer la feuille de route | Générer une feuille de route .txt après le calcul de la tournée, indiquant le détail du trajet à effectuer, les différentes livraisons et leur horaire, ainsi que les moments d'attente du livreur. |

| Fonctionnalités bonus   | Description   |
|---|---|
| Sauvegarde de livraisons  | Sauvegarder une liste de livraisons créées par un utilisateur                           |
| Chargement de livraisons  | Charger une liste de livraisons au lancement de l'application à partir d'une sauvegarde |
| Affichage de plusieurs tournées                                 | Afficher les tournées de plusieurs livreurs. (une seule tournée par livreur)            |
| Modification du livreur d'une livraison avant calcul de tournée | Modifier le livreur associé à une livraison avant calcul de la livraison.               |

## 2. Design Pattern

### 2.1. State Pattern

L'application Deliverapp est une application assez conséquente en elle-même. En plus du fait d'être nombreux pour réaliser le projet, il a fallu utiliser des règles et des outils pour pouvoir travailler de manière organisée. Le design pattern "State" est approprié à l'utilisation puisqu'il permet de structurer le projet afin de répartir correctement les fonctionnalités à développer dans les bonnes classes et d'éviter les comportements inattendus de l'application. De plus, le design pattern State est propice à la montée de version (ajouts de fonctionnalités), utile dans le cas d'un projet où le client peut changer d'avis sur les fonctionnalités qu'il souhaite.

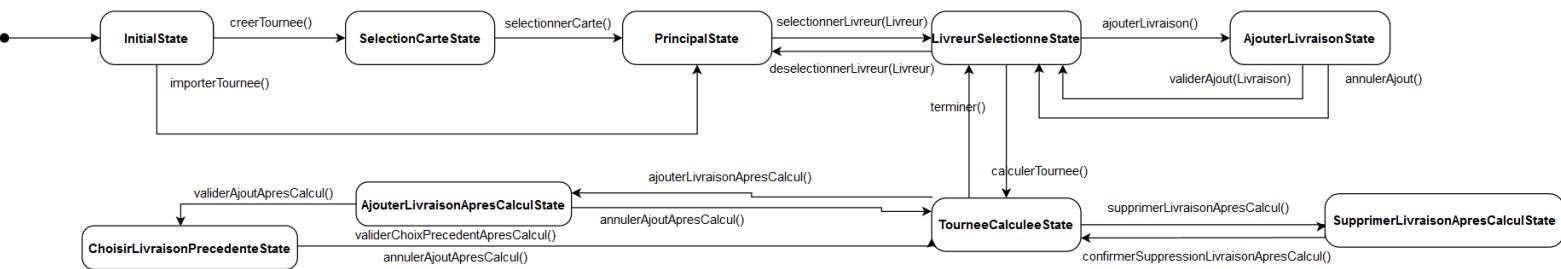


Figure 1: Diagramme d'état/transition

En ce qui concerne l'évolution depuis l'itération 1, nous avons modifié le StateChart Diagram car nous nous sommes rendus compte que certains états que nous avions déterminés n'étaient pas pertinents.

### 2.2. Command Pattern

L'utilisateur pouvant ajouter et retirer des livraisons via un clic sur l'interface, l'implémentation des boutons annuler et rétablir était un choix pertinent. Si l'utilisateur supprime une livraison par erreur, et s'il souhaite par la suite annuler le retour en arrière, ce design pattern permet de le faire rapidement et de manière élégante.

### 2.3. Observable Pattern

Deliverapp permet de visualiser les livreurs et les livraisons dans deux vues différentes : une vue textuelle et une vue graphique. Ces deux vues doivent lors d'un changement du modèle, être notifiés de ce changement pour mettre à jour leur affichage.

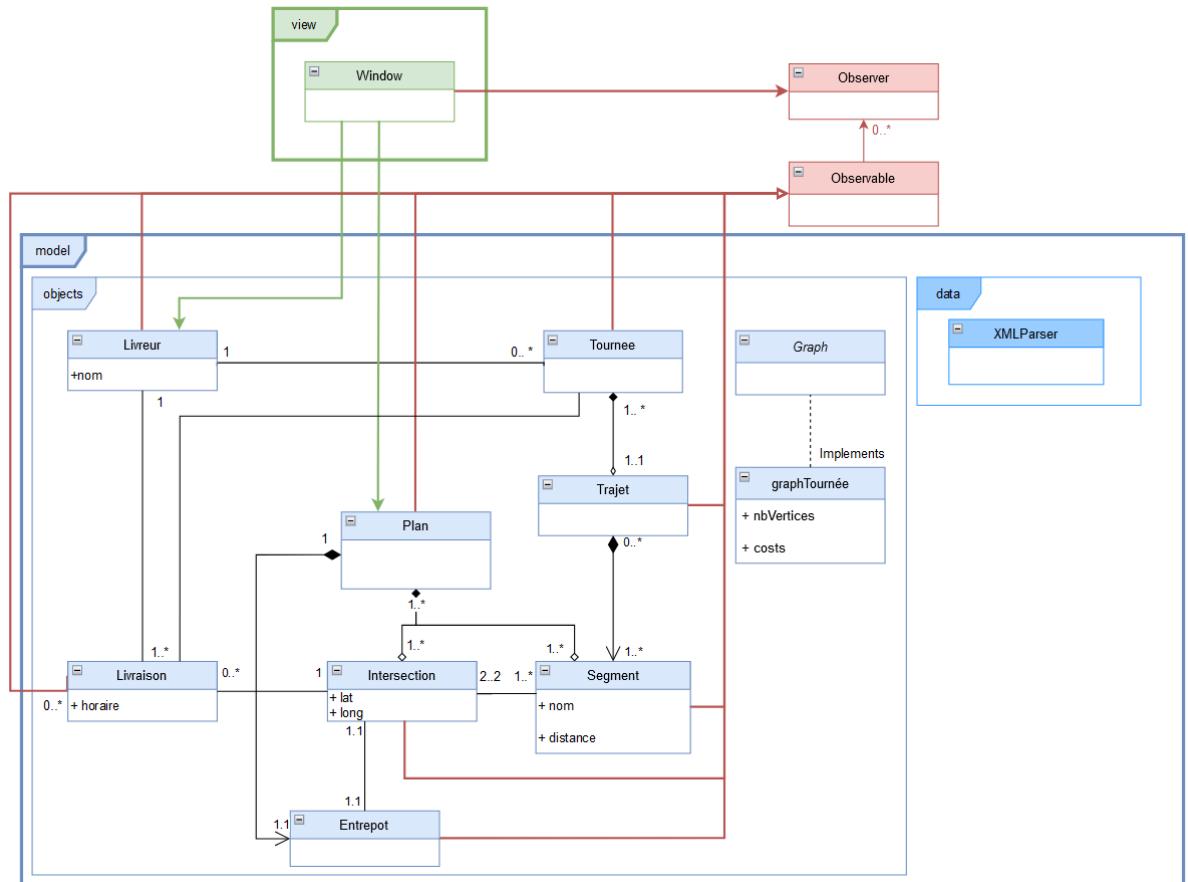


Figure 2 : implémentation du design pattern observable/observer dans l'application

### 3. Diagrammes

Pour notre application Deliverapp, nous avons opté pour une architecture MVC (Modèle-Vue-Contrôleur) que nous avons déjà utilisée en 3IF. De ce fait, outre le diagramme de package et le diagramme global de l'application, 3 diagrammes détaillés peuvent être établis: un pour la partie modèle, un pour la partie contrôleur et un dernier pour la partie vue.

#### 3.1. Diagramme package

VOIR ANNEXES

#### 3.2. Diagramme global

VOIR ANNEXES

#### 3.3. Diagrammes modèle

VOIR ANNEXES

### 3.4. Diagrammes contrôleur

VOIR ANNEXES

### 3.5. Diagramme vue

VOIR ANNEXES

## 4. Maquettes

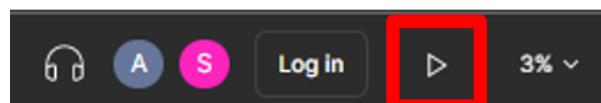
Dans la phase de conception de Deliverapp, nous avons réalisé des maquettes IHM respectant une charte graphique que nous avons établie. De toute évidence, au cours du développement et suite aux échanges avec le client (en l'occurrence Elod Egyed-Zsigmond joue ce rôle lors du projet), nous n'avons pas totalement respecté ces maquettes. Cependant, ces maquettes nous ont aidé lors de la phase de développement.

### 4.1. Maquette scénario

La maquette conçue de base est disponible en consultation interactive avec clics à l'adresse suivante et en annexe en PDF.

<https://www.figma.com/file/rsh3wxkUjazN5b10OGy7oE/Version-Simon?node-id=0%3A1&t=kEwl3B/oJ8dtMYIx-1>

Pour voir les animations, cliquer sur le bouton « Play » en haut à droite :



Puis choisir « Flow 1 » dans le menu déroulant gauche



Vous pourrez ensuite naviguer entre les pages de l'interface. Tous les boutons ne sont évidemment pas fonctionnels mais vous pourrez avoir un bon aperçu de l'idée d'enchaînement des fenêtres que nous avions à la conception.



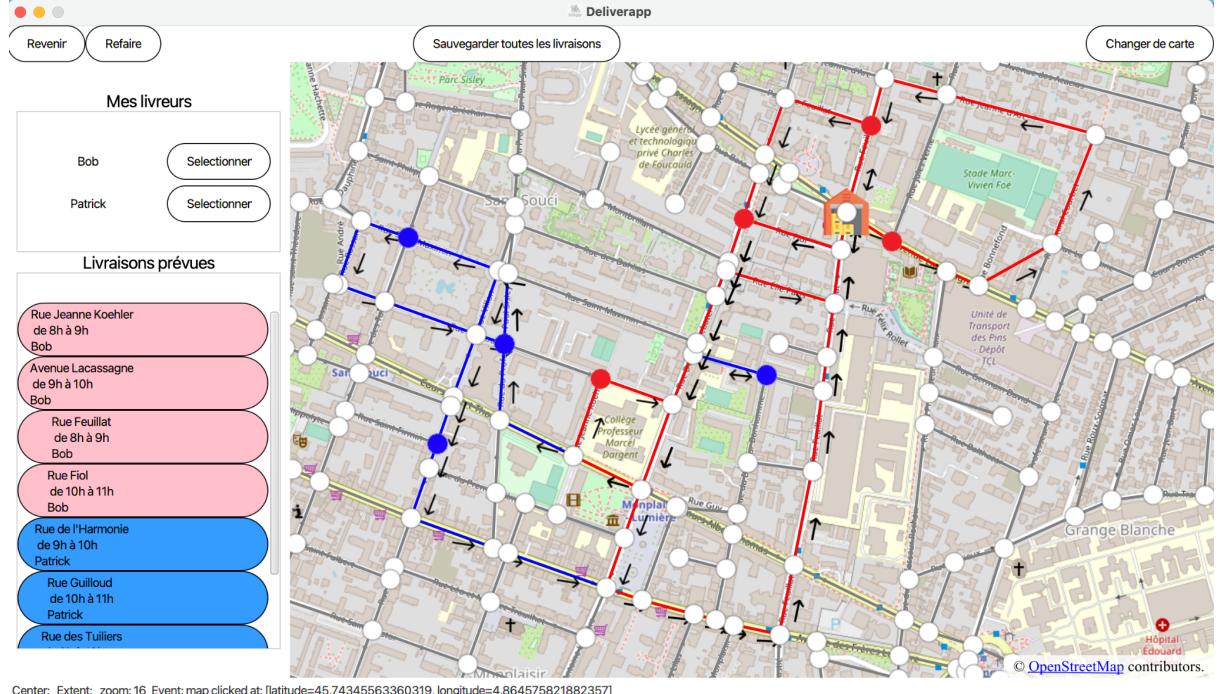
Si le site ne fonctionne pas pour diverses raisons, nous vous invitons à vous reporter à l'IHM en annexe.

## 4.2. Captures d'écrans de Deliverapp



Center: Extent: zoom: 16 Event: marker clicked: marker-100 [latitude=45.748405, longitude=4.875445]

Figure 3 : Interface de l'application avec les informations du livreur "Bob", affichage de l'état "AjouterLivraison"



Center: Extent: zoom: 16 Event: map clicked at: [latitude=45.74345563360319, longitude=4.864575821882357]

Figure 4 : Page principale de l'application avec les informations sur l'ensemble des livreurs, les tournées ayant déjà été calculées

## 5. Tests Unitaires et Algo

### 5.1. Algorithmie

Au niveau de l'algorithme, pour calculer la tournée, nous procédons en étapes et sous étapes:

1. Calcul de la tournée
  - a. Construction d'un GraphTournee à destination du TSP
    - i. Création des arcs entre livraisons et entrepôt en fonction des fenêtre de temps
    - ii. Calculs des distances pour les plus courts chemins pour chaque livraison d'origine et de destination pour chaque arc (Dijkstra avec heuristique)
  - b. Calcul de l'ordre des livraisons avec le TSP à partir du GraphTournee
2. Reconstruction de la tournée
  - a. Calcul des horaires associées à chaque livraison (avec les prises en compte des pauses et des 5 minutes pour livrer)
  - b. Calcul du détail des trajets entre chaque points de livraison/entrepôt
    - i. Calcul des distances pour les plus courts chemins avec prédecesseur pour fabriquer le trajet (Dijkstra avec mémorisation des prédecesseurs avec heuristique)

### 5.2. Test Unitaire Algorithmes

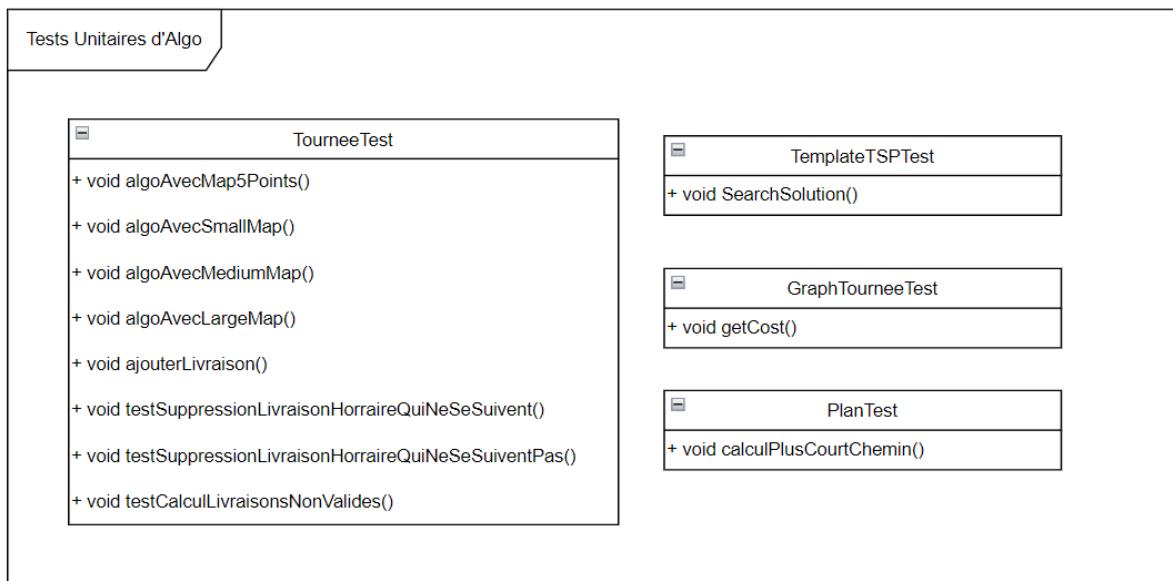


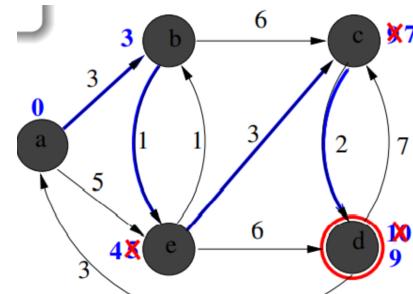
Figure 5 : Tests unitaires de l'algorithme

PlanTest: Calcul du plus court chemin entre deux points ou recherche des prédecesseurs avec heuristique:

Dans ce test, nous vérifions le bon fonctionnement de notre algorithme Dijkstra.

Nous avons créé cette liste d'intersections et de segments suivante, et vérifié que :

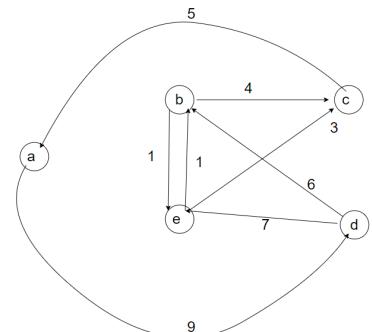
- Le plus court chemin de a à b est de 3
- Le plus court chemin de a à c est de 7
- Le plus court chemin de a à d est de 9
- Le plus court chemin de a à e est de 4



GraphTourneeTest: construction d'un GraphTournee avant de le passer dans l'algorithme TSP:

Pour les livraisons qui suivent, nous devons obtenir le GrapheTournee représenté située à droite:

- Entrepôt point a (ab=3, ae=5)
- Livraison point b à 10h (bc=6, be=1)
- Livraison point c à 11h (cd=2)
- Livraison point d à 8h (da=3, dc=7)
- Livraison point e à 10h (eb=1, ec=3, ed=6)



Ainsi lors du test effectué dans la classe GraphTourneeTest, nous vérifions que tout les arcs construits ont les bonnes valeurs et qu'aucun autre arcs supplémentaire n'a été créé (en l'occurrence une valeur -1 dans les coûts)

TemplateTSPTest: Calcul de la meilleure tournée possible pour un GraphTournee donnée :

Dans ce test, on vérifie que le TSP trie les livraisons du GraphTournee dans l'ordre le plus optimisé. En l'occurrence “a → d → b → e → c” pour le GraphTournee du test précédent.

TourneeTest: Calcul de la meilleure tournée pour des maps importantes et tests de performance :

Dans cette classe de tests, on vérifie :

- Le bon fonctionnement du calcul de trajets sur une carte de 5 points (a,b,c,d,e) affichée auparavant.
- Le bon fonctionnement du calcul de trajets sur la smallmap, mediummap et largemap, avant tout dans le but de vérifier le temps que prend l'algorithme à calculer les trajets sur les différentes tailles de Map.

- L'ajout et la suppression de livraisons après calcul de tournée. Nous vérifions que la livraison s'ajoute/se supprime bien, ainsi que l'horaire d'arrivée des livraisons suivantes se mettent bien à jour.

### 5.3. TU Chargement Plan

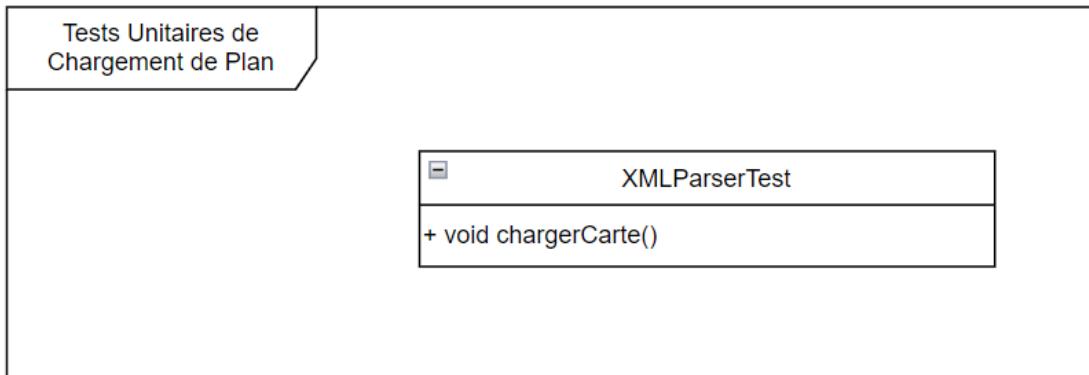


Figure 6 : Package Diagram pour les tests unitaires

#### XMLParserTest: test de chargement de carte XML

Dans ce test, on essaye de charger une carte, et on y vérifie :

- L'existence de l'entrepôt
- La latitude et longitude d'une intersection
- L'intersection de départ, d'arrivée, le nom et la distance d'un segment.

### 5.4. Implémentation des tests

Afin de réaliser nos tests, nous avons utilisé le framework **Junit 5.0** pour créer nos classes de tests et nos méthodes de tests. IntelliJ, l'IDE utilisé par tous les collaborateurs, propose un outil pour intégrer Junit 5.0 simplement. Ainsi à tout moment et surtout avant de faire une “Pull Request” il est possible de relancer les méthodes de tests pour vérifier que les modifications n'ont pas amené de bugs.

|   |               |
|---|---------------|
| ✓ ✓ deliverapp (fr)                     | 16 sec 643 ms |
| ✓ ✓ TourneeTest                         | 16 sec 564 ms |
| ✓ algoAvecLargeMap()                    | 3 sec 444 ms  |
| ✓ algoAvecMap5Points()                  | 7 ms          |
| ✓ algoAvecSmallMap()                    | 16 ms         |
| ✓ testSuppressionLivraisonHorraireQuiSe | 2 sec 855 ms  |
| ✓ testCalculLivraisonsNonValides()      | 4 sec 225 ms  |
| ✓ algoAvecMediumMap()                   | 156 ms        |
| ✓ testSuppressionLivraisonHorraireQuiNe | 1 sec 892 ms  |
| ✓ ajouterLivraison()                    | 3 sec 969 ms  |
| > ✓ PlanTest                            |               |
| > ✓ GraphTourneeTest                    |               |
| ✓ ✓ XMLParserTest                       | 65 ms         |
| ✓ chargerCarte()                        | 65 ms         |
| ✓ ✓ TemplateTSPTest                     | 14 ms         |
| ✓ searchSolution()                      | 14 ms         |

## 6. Partie gestion de projet

### 6.1. Projet Agile

Le projet a été mené en respectant une gestion agile avec une équipe projet de 7 personnes. L'organisation interne de notre groupe a été axée autour de quatre rôles : un chef de projet en charge de la supervision de l'équipe et de l'établissement d'un planning prévisionnel et réel en fonction du découpage en sous-tâches; des ingénieurs front en charge de l'établissement de la maquette IHM, de la réalisation d'une charte graphique et de la réalisation du style en CSS par la suite; un ingénieur parsing XML qui était en charge de la lecture de fichiers XML et de la sauvegarde des données; des ingénieurs spécialisés sur l'algorithme qui ont établi l'algorithme de plus court chemin (appliqué à notre cas d'usage) auquel ils ont ajouté les contraintes temporelles des fenêtres de livraison; et enfin un ingénieur expert en configuration qui s'est chargé de la mise en place de l'IDE et de la formation de l'équipe sur les outils utilisés.

La gestion du projet a été mise en place grâce à un découpage du projet en plusieurs itérations. Le projet comportait 32h de travail en séance, réparties sur quatre semaines. Les 16 premières heures ont été regroupées en une première itération comportant les phases de découverte du sujet, la définition des cas d'usage, l'installation de l'environnement de travail sur chacune des machines, l'établissement du glossaire des termes utilisés dans l'application ainsi que le diagramme d'état / transition. A la fin de cette première période, nous avons rassemblé la documentation générée dans un livrable et nous avons effectué une démonstration de l'application afin de montrer au client l'avancée du projet et de corriger les éléments qui ne correspondaient pas à ses attentes. En ce qui concerne les 16 heures suivantes, nous avions initialement prévu de diviser les séances en 2 sous-itérations (une sous-itération de développement et une sous-itération de test), mais nous avons finalement effectué trois sous-itérations (implémentation des design pattern, développement et test).

En plus de ce découpage en itérations, la gestion de ce projet s'est axée sur une forte communication d'équipe. En effet, une des plus grandes difficultés que nous avons rencontré est la répartition du travail et le fait de travailler à 7 sur le même projet sur un laps de temps réduit (sur une séance de 4 heures par exemple). Pour se répartir le travail, nous faisions un point d'équipe (après le retour client/professeur en classe entière) pendant lequel chaque membre de l'équipe présente ce qu'il a fait depuis le dernier point, et où l'équipe se répartit le travail.

En ce qui concerne le travail personnel, chaque membre de l'équipe a travaillé hors séance, que ce soit pendant les vacances ou hors séance (voir le total des heures ci-dessous). La grande majorité du travail personnel s'est effectué pendant la seconde itération et a concerné principalement l'implémentation des design patterns, l'enchaînement des différentes fenêtres (désactivation des boutons en accord avec les scénarios d'utilisation) et la gestion de la vue textuelle et de la vue graphique. Quand un des membres du groupe entamait une séance de travail personnel, il envoyait un message sur la conversation du groupe en précisant la fonctionnalité implémentée/modifiée afin que les autres ne travaillent pas sur les mêmes fichiers et ainsi éviter les conflits de version.

En plus des problèmes de répartition de travail, nous avons rencontré des incohérences de versionnage au lancement du projet. Ne sachant pas forcément utiliser GitHub efficacement, nous avons eu besoin des deux premières séances pour nous former sur l'utilisation de cet outil afin de maîtriser les concepts de branches, de push/pull et ne plus avoir d'erreurs de versions.

Pour ce qui est de l'organisation de l'équipe générale, les séances en présentiel le matin ont été très utiles. Grâce aux salles projets du département, nous avons pu nous réunir fréquemment ce qui a permis de faire un point régulier et de coordonner l'équipe.

La gestion agile du projet nous a permis de nous adapter facilement et rapidement aux attentes du client. Des attentes qui ont évolué durant le projet. Par exemple, lors de la 6ème séance, suite aux points client effectués en début de séance de cours, nous avons ajouté le design pattern command car le client nous a indiqué vouloir faire des retour arrière/avant pour annuler les dernières modifications effectuées. A l'inverse, lors de la 7ème séance le client nous a annoncé ne pas vouloir modifier le livreur associé à une livraison, contrairement à ce qui était annoncé dans le cahier des charges initial. Suite aux changements annoncés, nous avons effectué une réunion de groupe projet durant laquelle nous avons défini les modifications à effectuer et les fichiers concernés. Si nous avions des points à clarifier, nous posions nos questions aux clients.

| Membre de l'équipe | Sujet Principaux hors séance   | Nombre d'heures totale (séance + hors séance) |
|--------------------|--|---|
| Maël               | Cohérence vue graphique/vue textuelle, front,  | 32+40   |
| Corentin           | Algorithme, design pattern, test unitaire calcul de tournée  | 32+40   |
| Mohamed            | Import et sauvegarde livraison, génération feuille de route  | 32 +40  |
| Simon              | Front, Enchaînement de fenêtre, State  | 32 +40  |
| Alexis             | Implémentation design pattern, enchaînement de fenêtre, gestion Git/IDE et rédaction de la documentation | 32+40   |
| Colin              | Ajout/suppression livraison post calcul de tournée, algo   | 32+40   |
| Eva                | Implémentation design pattern, gestion erreur ajout livraison, modification livraison                    | 32+40   |

## 6.2. Outils utilisés

Afin de respecter la méthode Agile, nous avons utilisé l'outil de gestion de versions **Git** et la plateforme de collaboration **GitHub** pour utiliser des répertoires distants. Nous avons choisi de créer une branche par collaborateur pour le développement. Ainsi, chaque collaborateur travaille sur sa branche en implémentant la fonctionnalité qu'il souhaite en s'assurant être seul à la développer.

Afin de réaliser les maquettes IHM, nous avons utilisé l'outil **Figma**. Cela permet un travail collaboratif simplifié et la création d'une visualisation d'un

scénario interactif pour utiliser l'application et détecter des fonctionnalités intéressantes à implémenter.

L'IDE utilisé par tous les collaborateurs est **IntelliJ**. Cela a permis de facilement mettre en place les tests avec **JUnit 5** et le framework **JavaFX**. Cet IDE permettait aussi d'utiliser Git directement dans son interface ainsi que de debugger précisément notre code. IntelliJ nous a aussi permis de générer la documentation du code à partir des commentaires laissés. Afin d'afficher la carte OpenStreetMap, nous avons utilisé la librairie **mapjfx** de sothawo. Cette librairie open source est disponible ici : [mapjfx | sothawo](#)

Pour la réalisation des différents diagrammes, l'outil **Draw.io** a été utilisé.

**Notion** a aussi été utilisé pour la prise de notes de toutes remarques au cours du projet. Cet outil collaboratif est très pratique et rassemble toutes les informations nécessaires aux collaborateurs que ce soit pour effectuer un suivi des fonctionnalités ou mettre au courant tout le monde de l'avancée du projet et des fonctionnalités qui reste à implémenter. Afin de rédiger ce rapport, nous avons utilisé **Google Docs**.

## 6.3. Planning du projet

*Annexe : Planning Prévisionnel, Planning réel et Analyse GitHub*

En regardant les planning prévisionnel et réel, on peut analyser rapidement les écarts entre les délais prévisionnels et réels.

Dans un premier point, nous constatons une que la phase de pré-développement de l'itération 1 a duré deux semaines de plus que prévu. Cela s'explique par le fait que nous avons développé plus de fonctionnalités que prévu dans cette première phase, allant de l'ajout/suppression de livraison jusqu'au chargement de la carte et l'affichage d'une tournée, ce qui nous a permis de nous avancer sur la phase de développement.

De plus, nous avons modifié le découpage de l'itération 2 à laquelle nous avons ajouté la sous-itération "Implémentation des design patterns". En effet, nous avons sous-estimé l'importance des design pattern dans la structure du code et nous avons passé un long moment à modifier la disposition de nos méthodes.

Un troisième élément d'analyse pourrait être la phase de développement en elle-même. En effet, le planning prévisionnel nous semblait très court car il prévoyait d'implémenter une grande partie des fonctionnalités de

l'application en seulement 1 semaine et demi. Nous avons pu respecter ces délais en ajoutant une importante dose de travail personnel, mais il aurait été impossible d'implémenter l'ensemble de l'application uniquement avec le temps en séance.

## 6.4. Retour de la part de chaque collaborateur

### **Arrivé Maël:**

(Rôle : expert Frontend)

Ce PLD a été pour moi une première opportunité de développer une application complexe en équipe, de la partie conception à la partie production. De plus, j'ai trouvé très intéressant l'approche AGILE du projet, permettant via l'architecture et les designs patterns une grande adaptabilité.

Du côté Frontend, le maquettage réalisé sur Figma était intéressant. En effet, les interfaces devaient être agréables et efficaces, sans être trop difficile à développer, ce qui n'était pas évident (surtout sans trop savoir quelles fonctionnalités javafx nous offrait). Je pense que nous avons bien réussi cela, car notre figma ressemble fortement à l'interface finale.

Par ailleurs, je pense que ce qui m'a le plus plu dans ce projet est le côté "relation client" faussement créé par vous afin de nous mettre dans une situation réelle. Du glossaire à la démonstration, passer du langage technique à fonctionnel est un exercice primordial qui est au cœur du métier d'ingénieur.

Pour finir, je pense que le projet s'est passé d'une manière idéale au sein de notre groupe. Sans se mettre de pression, tout le monde s'est mis au travail, ce qui a permis à chacun d'évoluer dans une atmosphère sereine et agréable.

### **Flandre Corentin:**

(Rôle : expert algorithme & architecture Design Pattern)

Le projet Agile m'a permis de mieux appréhender les projets de groupes. J'ai compris l'intérêt majeur de la gestion de projet (surtout dans une équipe assez nombreuse). J'ai enfin appris à utiliser dans un cas concret la gestion de version. De toute évidence, le plus dur a été de répondre aux attentes du client qui sont variables sans cesse, mais c'est le principe de l'agilité. J'ai beaucoup travaillé sur la partie algorithmique et j'ai appris beaucoup sur la réflexion à optimiser un algorithme. En effet toutes les optimisations et heuristiques nous permettent finalement de calculer une tournée en temps raisonnable même pour une map avec de nombreuses intersections pour un grand nombre de livraisons. De plus, j'ai appris à mettre en place des "design pattern" que je trouve super utile pour la montée de version et l'organisation.

Finalement, pour résumer, je dirais que nous avons totalement maîtrisé le travail d'équipe.

**Hmidi Mohamed:**

(Rôle : Expert parsing/lecture/écriture XML-Text)

Je trouve que le projet Agile est une bonne opportunité d'une part pour appliquer et améliorer nos connaissances techniques et d'autre part pour nous habituer à travailler dans une équipe assez nombreuse.

Il nous a permis de comprendre et surtout d'appliquer le mécanisme d'Agilité dans un projet concret. De plus, il nous a permis d'avoir une idée de ce qui se passe dans les grandes entreprises(les méthodes de travail, les méthodes de gestion de projet, les méthodes d'organisation des équipes...) même si on est dans une échelle plus petite.

Les techniques apprises et améliorées :

- Git (gestion des versions et des conflits)
- IntelliJ IDEA (gestion des configurations et relation avec Git grâce à Alexis Reis)
- FXML et SceneBuilder (outils pour interagir avec l'IHM)
- JAVA et INPUT/OUTPUT (gestion de l'entrées/sorties texte et XML en java)

**Poulet Simon:**

(Rôle : expert Frontend)

Le PLD Agile a été une vraie expérience de travail en équipe. C'est véritablement la première fois que j'ai pu travailler avec une équipe de cette taille sur un projet commun avec l'aide d'un dépôt git, de travail collaboratif, etc. Ça a été réellement formateur sur ce point, car même en stage je n'ai pas pu vraiment travailler sur un code où tout le monde travaillait en même temps dessus, résoudre les soucis de merge, etc. Les outils utilisés et l'intégration que nous en avons faite me sera également très utile plus tard. J'ai également beaucoup pu apprendre sur les technologies utilisées et notamment le JFX, les FXML que je n'avais jamais utilisés précédemment.

J'ai notamment travaillé sur le frontend de l'application ce qui m'a beaucoup plus et était très intéressant. D'autres technologies que le Java auraient pu être aussi intéressantes à aborder (backend en Java et frontend dans un autre langage ?).

En somme le projet a été très intéressant à réaliser car il abordait des thématiques différentes, que nous avons pu travailler en groupe dessus de manière collaborative et tout s'est très bien déroulé. Je suis au sein d'une très bonne équipe qui s'est très bien impliquée dans le projet ce qui a été un moteur pour avancer. Je trouve cependant le projet légèrement trop important en termes de charge de travail en dehors des cours.

**Reis Alexis:**

(Rôle : expert configuration IDE / Git)

Projet très chronophage et fastidieux mais aussi très formateur pour l'organisation d'un projet. Nous avons dû nous adapter chaque semaine à

de nouvelles contraintes et de nouveaux problèmes et c'est ensemble finalement que nous avons avancé le plus rapidement et le plus efficacement. Le plus grand challenge de ce projet a été la mise en place des design pattern que nous n'avions pas encore touchés. La partie graphique sur Java était elle-aussi très compliquée à implémenter. Les outils que nous avons utilisés (IntelliJ et Github) ont été parfaitement intégrés par l'équipe après un petit brief et aucun problème irrésolvable n'est survenu ensuite.

### **Thomas Colin:**

(Rôle : co-expert algorithme et tests unitaires)

Le projet nous a demandé beaucoup de temps mais nous a beaucoup appris. Nous avons eu l'impression de réaliser un vrai projet d'équipe, et nous avons appris l'organisation en sprint et Git sur le terrain, avec quelques petits accrocs. On a dû beaucoup communiquer pendant même qu'on codait pour résoudre les différents problèmes, et je trouve que l'on avait une bonne rythmique et une bonne façon d'avancer dans le projet, en particulier vers les dernières heures.

### **Ugnon-Coussioz Eva:**

(Rôle : chef de projet)

Le projet AGILE ressemblait en quelque sorte à un "sprint" en entreprise où une équipe travaille sur un projet avec un taux horaire important sur une plage de temps réduite. Il nous a permis de nous former sur de nouveaux outils (Git, IntelliJ) et nous a donné l'occasion de mettre en pratique les principes du fonctionnement projet AGILE. Le fait d'avoir des fonctionnalités peu précises au départ simule bien la réalité d'une entreprise qui répond à l'appel d'offre d'un client dont les besoins sont peu détaillés. Ce projet est très intéressant d'un point de vue humain car il nous apprend à gérer des périodes de haute intensité en terme de rendu/développement tout en maintenant une organisation cohérente au sein de l'équipe et en ayant une atmosphère de travail agréable. Le fait d'alterner avec un autre projet en parallèle (PLD Mars) nous force à communiquer et nous partager le travail de manière efficace.

Pour ce qui est des problèmes rencontrés, ils ont concerné majoritairement les problèmes de version avec le Git et l'implémentation des design pattern. Pour pallier cela, nous avons eu recours à des points réguliers en séance, mais également hors séance où nous nous retrouvions pour travailler ensemble.

D'un point de vue personnel, j'ai grandement apprécié tenir le rôle de chef de projet car il m'a permis d'avoir une vision globale du travail de chacun, de répartir le travail en fonction des compétences/préférences de chacun, et de tenir à jour un planning d'avancé. Nous avons réussi à maintenir les délais grâce à une grande adaptabilité de travail qui nous a permis de travailler hors séance très régulièrement.

## 6.5. Axes d'améliorations

Telle que nous avons développé l'application, lorsque l'utilisateur clique sur la carte de la vue graphique, le point sur lequel il a cliqué n'est pas mis en surbrillance directement, ce qui détériore l'expérience utilisateur car il n'a pas de confirmation visuelle que le clic a été pris en compte. En effet, les intersections cliquées ne sont mises en surbrillance qu'après avoir été affectées à un livreur. Nous pourrions ajouter un icône de localisation, comme sur google map par exemple.

Pendant la création de livraison, nous avions décidé d'afficher les champs "plage horaire" et "sélection de point de livraison" en rouge lorsque l'on valide l'ajout de livraison alors que les champs n'étaient pas remplis. Nous avons remarqué que d'un point de vue utilisateur, il serait préférable et plus clair de désactiver le bouton valider de la création livraison tant que tous les champs ne sont pas remplis. De plus, nous pourrions afficher les champs en vert lorsque ceux-ci sont corrects.

Actuellement, nous ne pouvons pas naviguer entre les fenêtres. En effet, si on se trompe de bouton à l'entrée dans l'application, l'utilisateur doit fermer et rouvrir celle-ci afin de changer de mode de fonctionnement (mode importation/mode création). Cela ne peut pas exister dans la version finale d'une application, dans laquelle l'utilisateur ne doit jamais être amené à fermer l'application par nécessité.

## 7. Annexes

### 7.1. Acronyme

| Acronyme      | Définition                      |
|---------------|---------------------------------|
| <b>MVC</b>    | Modèle-Vue-Contrôleur           |
| <b>IHM</b>    | Interface Homme Machine         |
| <b>DSD(s)</b> | Diagrammes Séquences Système(S) |
| <b>PLD</b>    | Projet Longue Durée             |

### 7.2. Glossaire

| Mot                         | Définition   |
|-----------------------------|--|
| <b>Adresse de livraison</b> | Adresse physique associée à une livraison. Une livraison est livrée à une intersection appartenant à la carte.   |
| <b>Carte</b>                | Représentation d'une zone géographique. Elle est constituée d'un entrepôt, de plusieurs intersections et de segments. Dans notre cas, les cartes représentent une partie de la ville de Lyon.            |
| <b>Client</b>               | La personne physique qui va recevoir son colis.<br>OU (selon cas d'usage du mot)<br>Dans le cadre du projet, le client a demandé l'application.<br>Elod Egyed-Zsigmond joue ce rôle durant le PLD Agile. |
| <b>Entrepôt</b>             | Intersection de départ et d'arrivée du livreur à la fin de sa tournée.   |
| <b>Fenêtre de temps</b>     | Période d'une heure, commençant à 8h, 9h, 10h ou 11h du matin. Une livraison à une fenêtre de temps pendant laquelle le colis doit être livré au client.   |
| <b>Feuille de route</b>     | Fichier texte téléchargeable après le calcul d'une tournée qui détaille les horaires et les lieux des livraisons et le chemin à faire  |
| <b>Intersection</b>         | Composée de deux coordonnées (latitude et longitude).  |
| <b>Livraison</b>            | Dépôt d'un colis par un livreur à une intersection et à une fenêtre de temps précise.  |
| <b>Livraison non valide</b> | Une livraison non valide est une livraison appartenant à une tournée dont l'horaire de livraison ne correspond pas à la fenêtre de temps associée à celle-ci.  |
| <b>Livreur</b>              | Personne qui livre des colis à vélo  |
| <b>Segment</b>              | Représentation d'une portion de route qui rejoint deux intersections. Il possède un nom commun et une longueur renseignée en mètres.   |
| <b>Tournée</b>              | Listes de livraisons associés chacune à l'horaire de livraison. Une tournée se calcule afin d'optimiser les livraisons à faire pour un livreur.  |
| <b>Trajet</b>               | Liste de segments qui représente donc un chemin physique.  |
| <b>Utilisateur</b>          | Personne qui utilise l'application pour planifier les livraisons des livreurs et calculer des tournées.  |

### 7.3. Diagramme d'utilisation

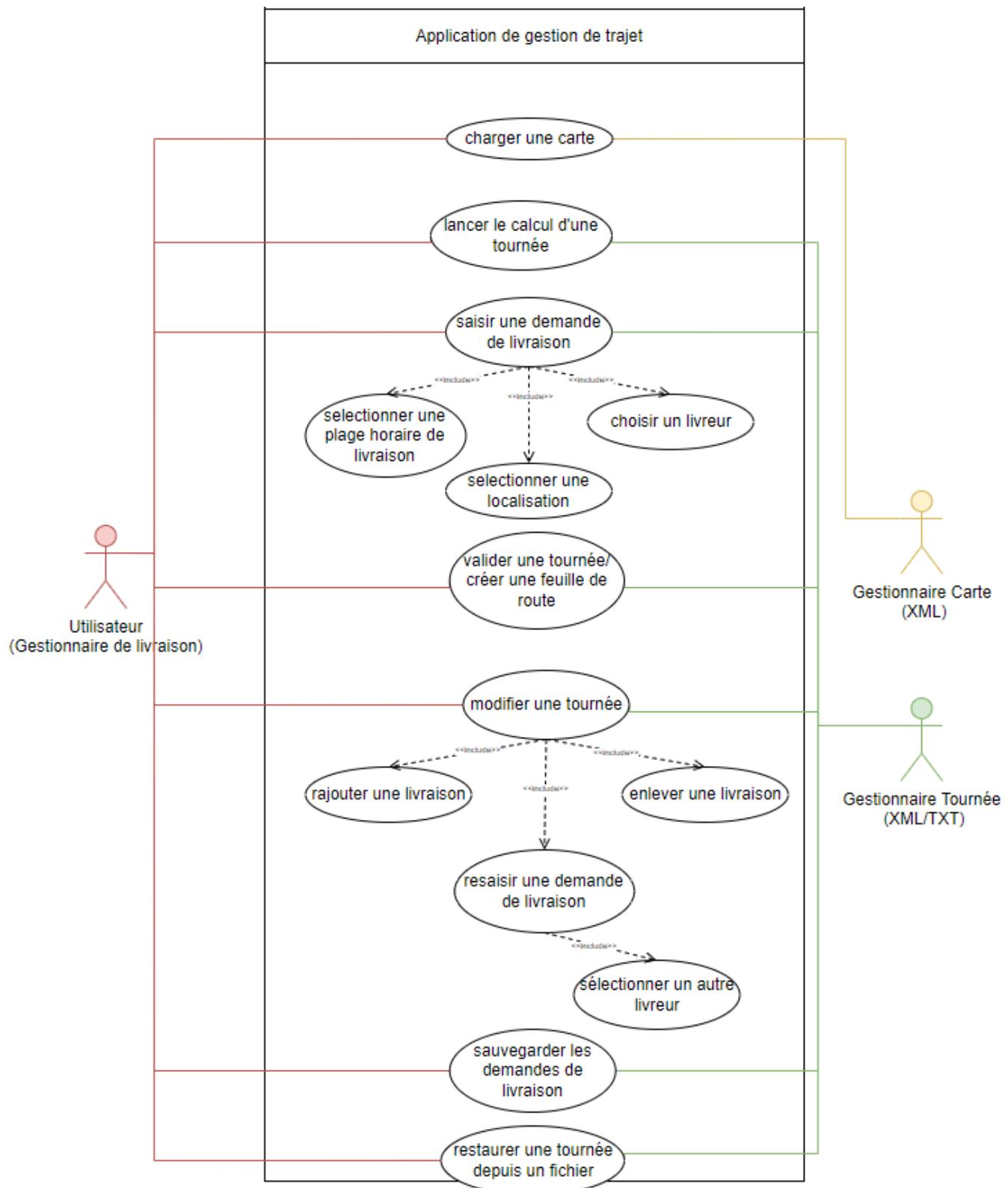
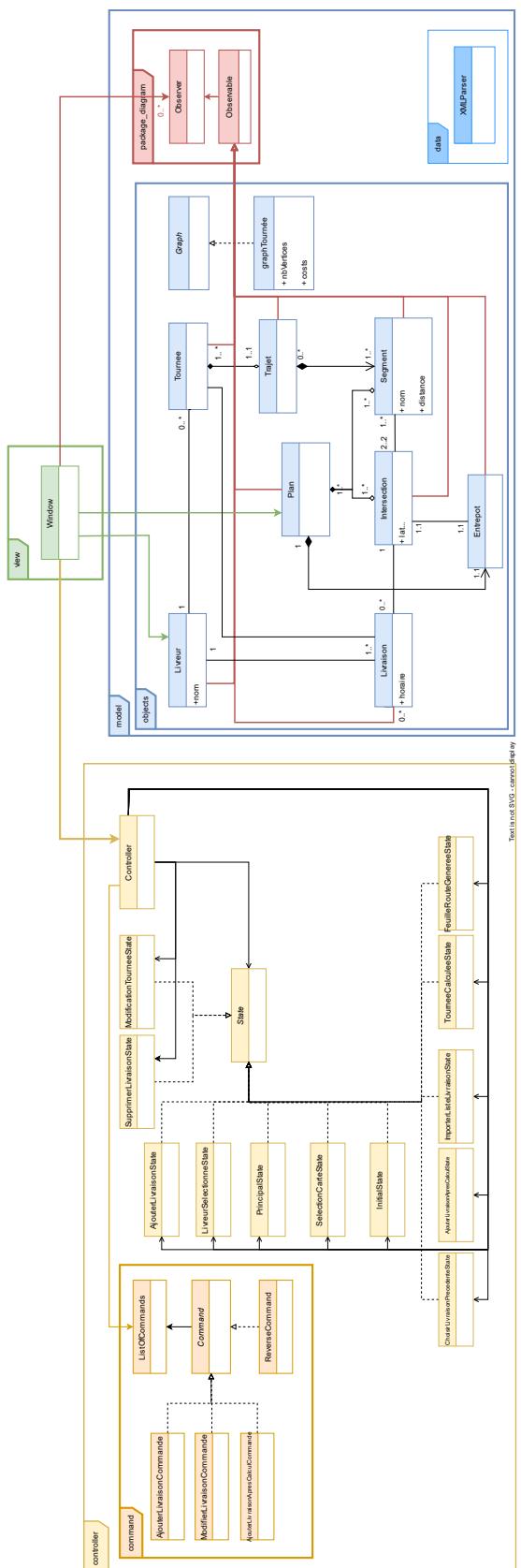


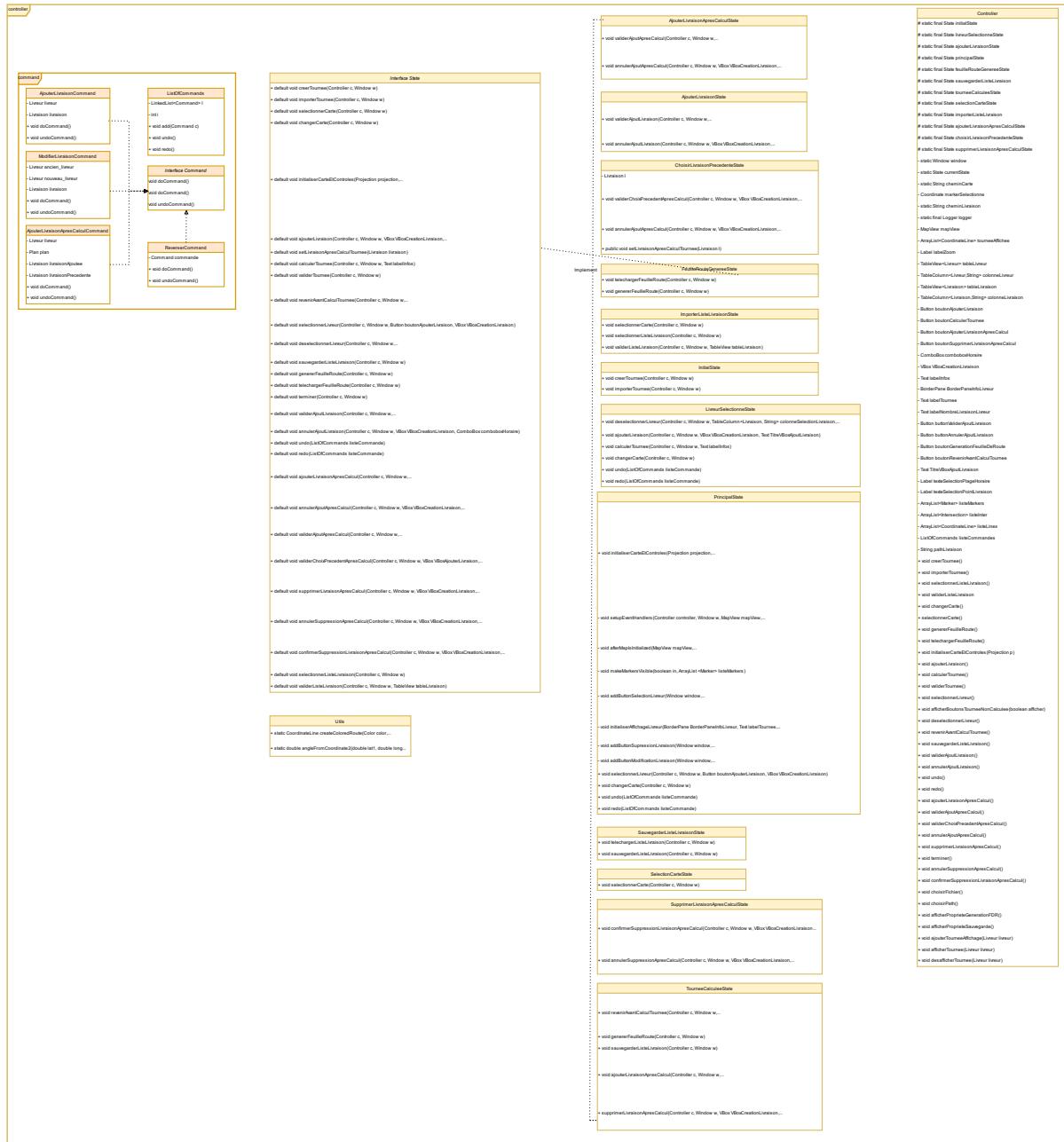
Figure 7 : Diagramme de classe d'utilisation

## 7.5. Diagramme de package



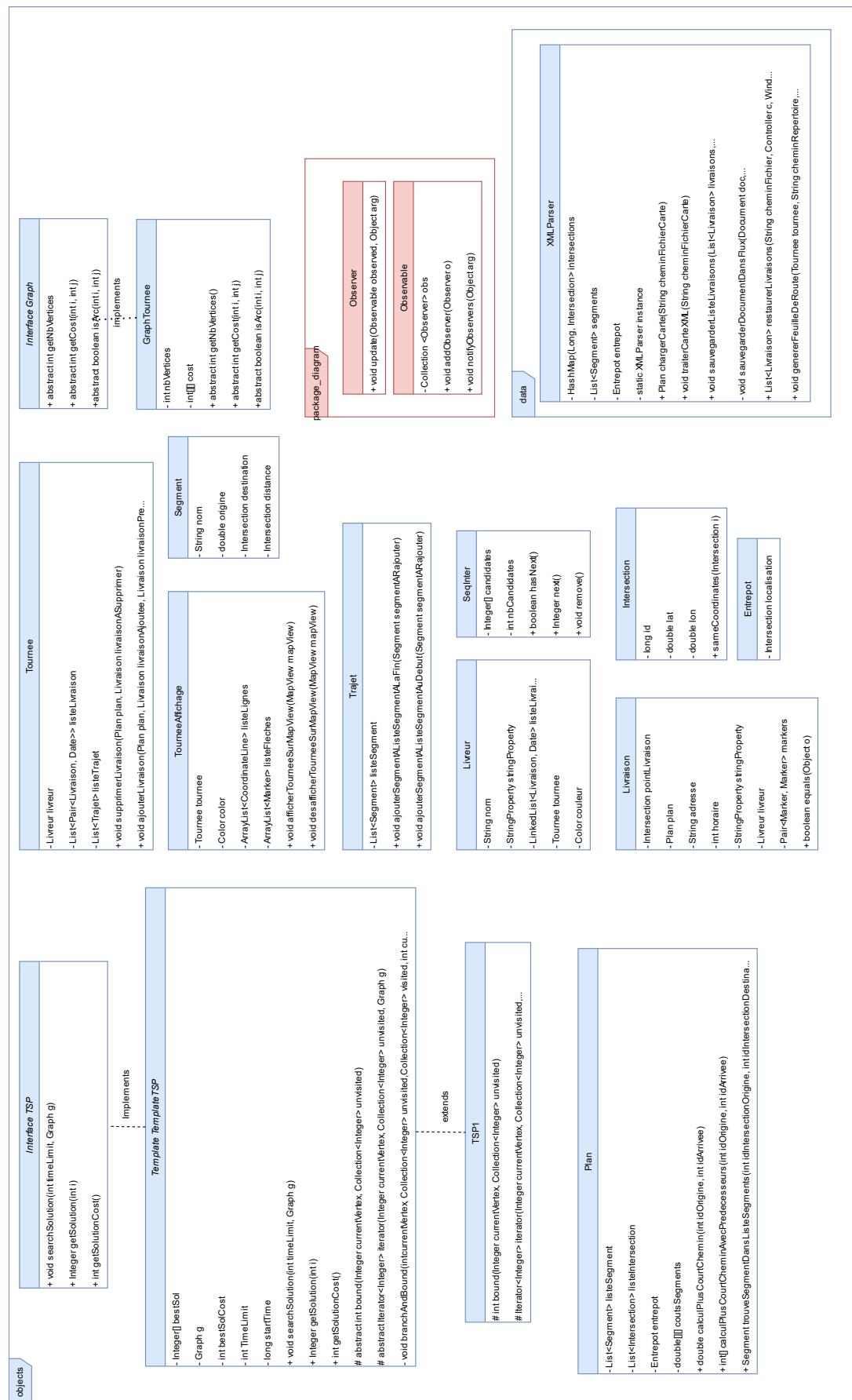
## 7.6. Diagramme de classes

### 7.6.1. Diagramme de classe du package controller





## 7.6.2. Diagramme de classe du package Model

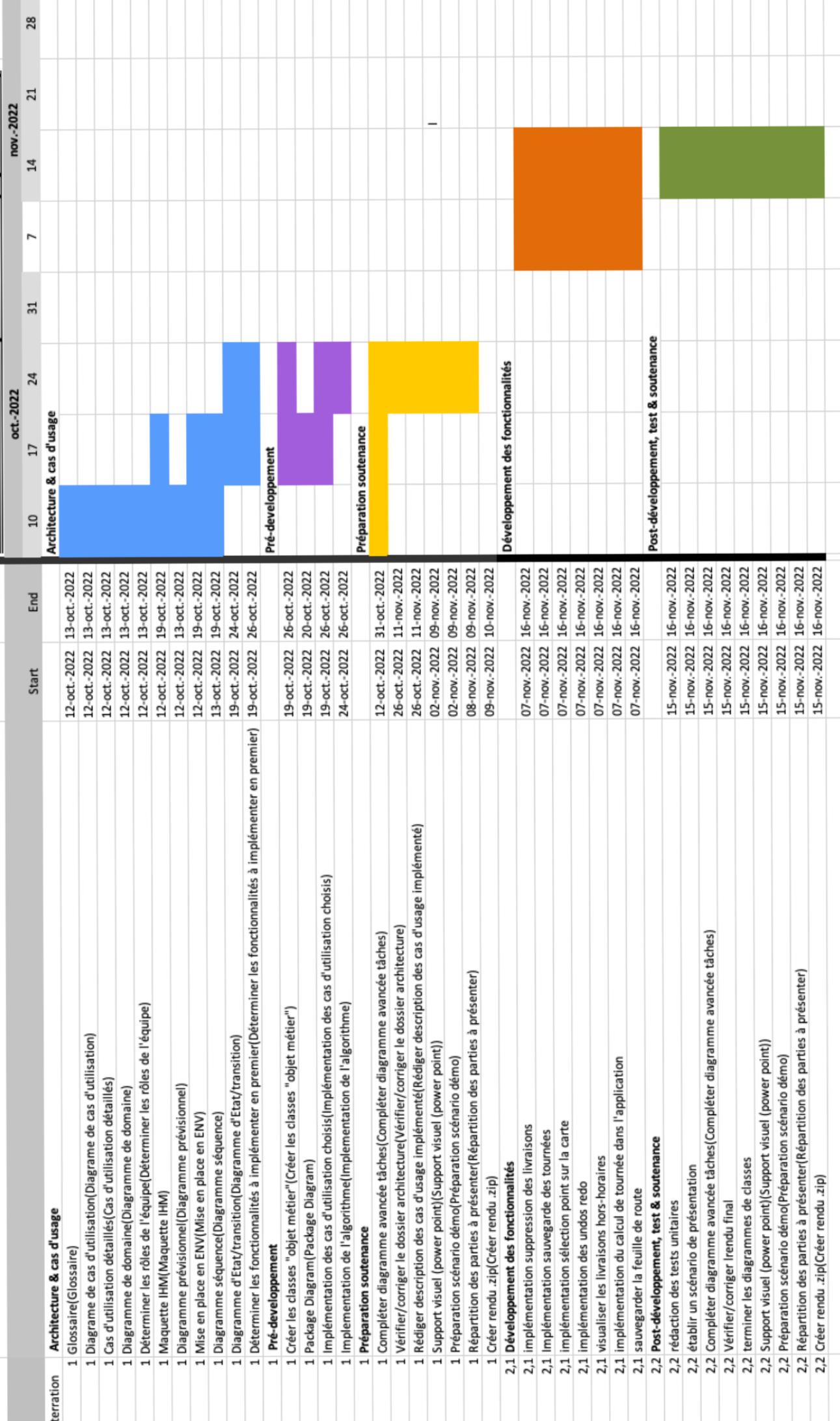


Text is not SVG - cannot display

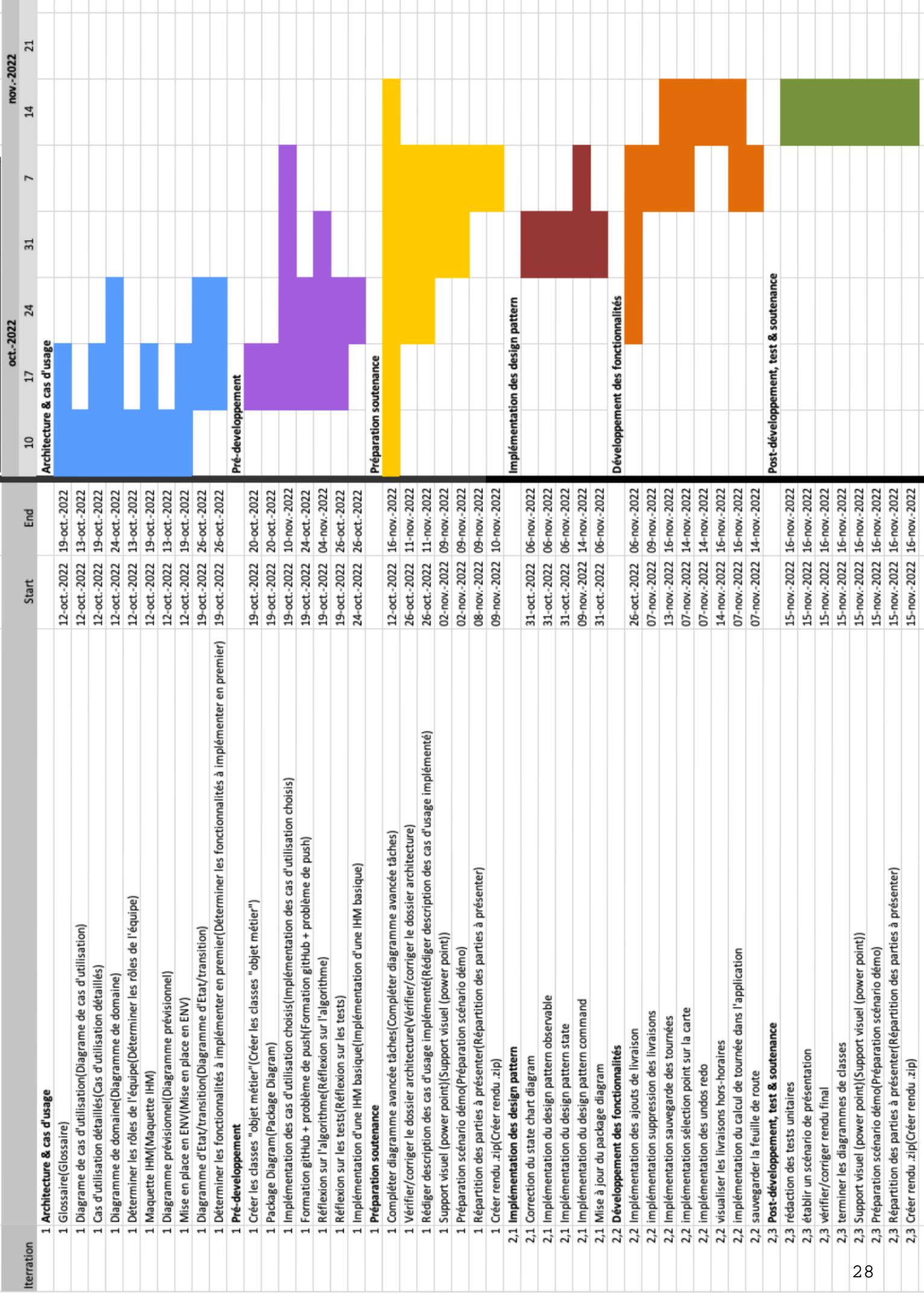
### 7.6.3. Diagramme de classe du package view

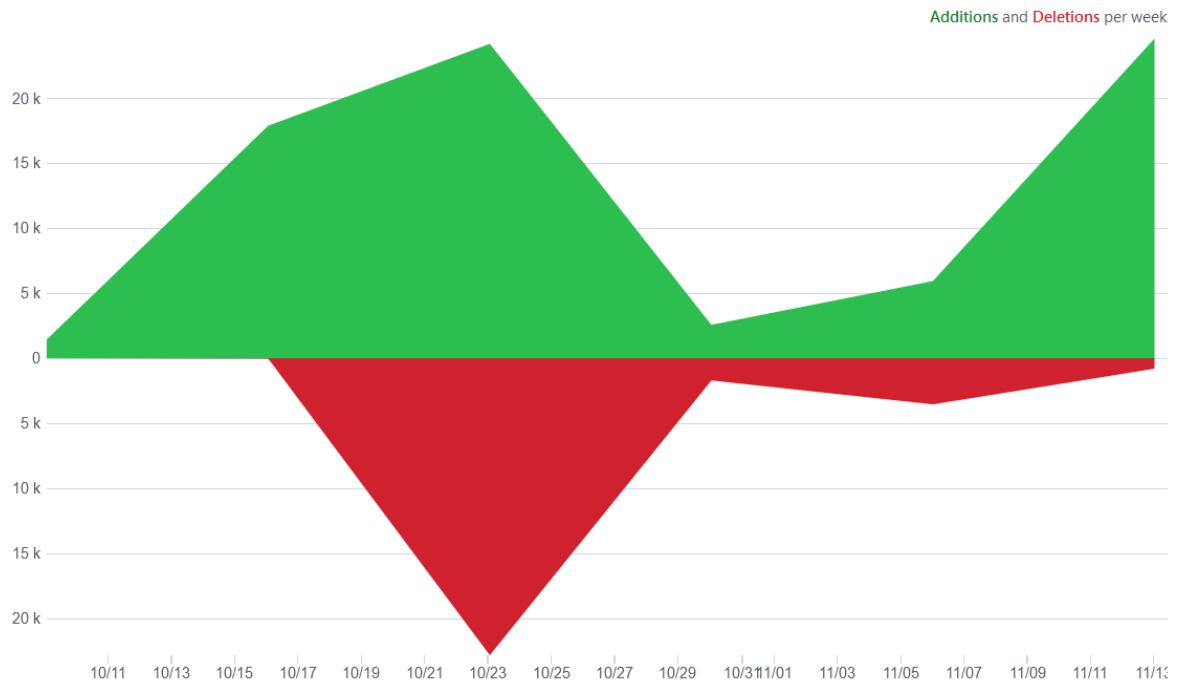
| Window  |
|---|
| <ul style="list-style-type: none"> <li>- static final Logger logger</li> <li>- static ObservableList&lt;Livreur&gt; livreur</li> <li>- Stage primaryStage</li> <li>- Controller controller</li> <li>- Plan plan</li> <li>- ObservableList&lt;Livraison&gt; livraison</li> <li>- Livreur livreurSelectionne</li> <li>- Pair&lt;Marker, Marker&gt; markerSelectionne</li> <li>- List&lt;Livraison&gt; livraisonsSelectionnees</li> <li>- List&lt;Pair&lt;Livraison, Marker&gt;&gt; listeLivraisonMarker</li> <li>+ static void main(String[] args)</li> <li>+ ObservableList&lt;Livreur&gt; getLivreurs()</li> <li>+ ObservableList&lt;Livraison&gt; getLivraisons()</li> <li>+ void start(Stage primaryStage)</li> <li>+ void setChoixCarteWindow()</li> <li>+ void setChoixImporterWindow()</li> <li>+ void setPagePrincipale()</li> <li>+ Stage getPrimaryStage()</li> <li>+ Livreur getLivreurSelectionne()</li> <li>+ void setLivreurSelectionne(Livreur livreurSelectionne)</li> <li>+ Plan getPlan()</li> <li>+ void setPlan(Plan p)</li> <li>+ List&lt;Livraison&gt; getLivraisonsSelectionnees()</li> <li>+ void ajouteLivraisonsSelectionnees(Livraison livraisonSelectionne)</li> <li>+ void supprimerLivraisonsSelectionnees(Livraison livraisonSelectionne)</li> <li>+ Pair&lt;Marker, Marker&gt; getMarkerSelectionne()</li> <li>+ void setMarkerSelectionne(Marker markerSelectionne)</li> <li>+ void setMarkerVisible(Marker markerClicked)</li> <li>+ void update(Observable o, Object arg)           <br/>Text is not SVG - cannot display</li> </ul> |

## Board - PLD-AGILE-prévisionnel (by months)



## Board - PLD-AGILE (by months)





# IHM/GUI prévisionnel du PLD

## AGILE H4124

