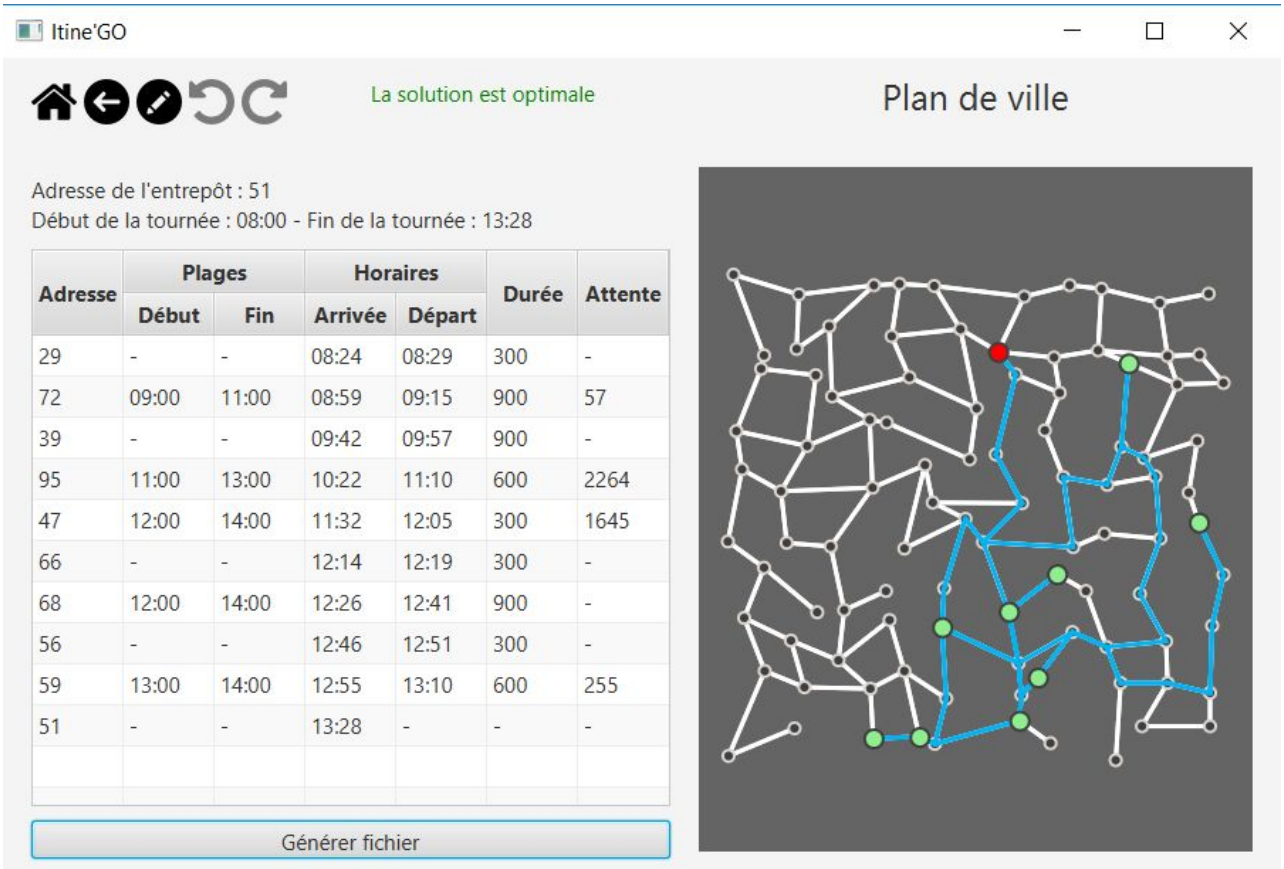
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Rapport PLD-AGILE

H4103

Pierre Bayle - Nicolas Chazeau - Ludovic Cros - Jacques Folleas - Nicolas Gripont - Pierre Jaglin - Quentin Vecchio

Itine’GO



Adresse	Plages		Horaires		Durée	Attente
	Début	Fin	Arrivée	Départ		
29	-	-	08:24	08:29	300	-
72	09:00	11:00	08:59	09:15	900	57
39	-	-	09:42	09:57	900	-
95	11:00	13:00	10:22	11:10	600	2264
47	12:00	14:00	11:32	12:05	300	1645
66	-	-	12:14	12:19	300	-
68	12:00	14:00	12:26	12:41	900	-
56	-	-	12:46	12:51	300	-
59	13:00	14:00	12:55	13:10	600	255
51	-	-	13:28	-	-	-

SOMMAIRE

I - Introduction **3**

II - Cas d'utilisation **4**

A. Charger fichier plan de la ville 4
B. Charger fichier livraison 5
C. Calculer une tournée 6
D. Modifier la tournée 7
E. Ajouter une nouvelle livraison 9
F. Supprimer une livraison 10
G. Changer une plage horaire 11
H. Échanger l'ordre d'une livraison 13
I. Générer une feuille de route 14

III - Explications des choix architecturaux et design patterns utilisés **15**

A. MVC 15
B. Pattern State 15
C. Pattern Undo/Redo 15

IV - Diagramme de Package et de Classe **17**

A. Diagramme de Package 17
B. Diagramme de Classe 17

V - Couverture de Test **18**

VI - Itérations **27**

A. 1ere Itération 27
B. 2éme Itération 29
C. 3éme Itération 32

VII - Utilisation du code (Instructions) **34**

VIII - Glossaire **35**

IX - Conclusion **36**

I - Introduction

Dans le cadre de l'unité d'enseignement PLD-Agile, nous avons conçu et implémenté un logiciel orienté objet, permettant à des sociétés de livraison de préparer leurs tournées, et de guider les chauffeurs de camion pendant leurs livraison. Les tournées calculées devaient être optimales en temps et respecter des plages horaires. Ce logiciel avait également de nombreuses fonctionnalités supplémentaires, mentionnées dans le cahier des charges, que nous expliciterons par la suite.

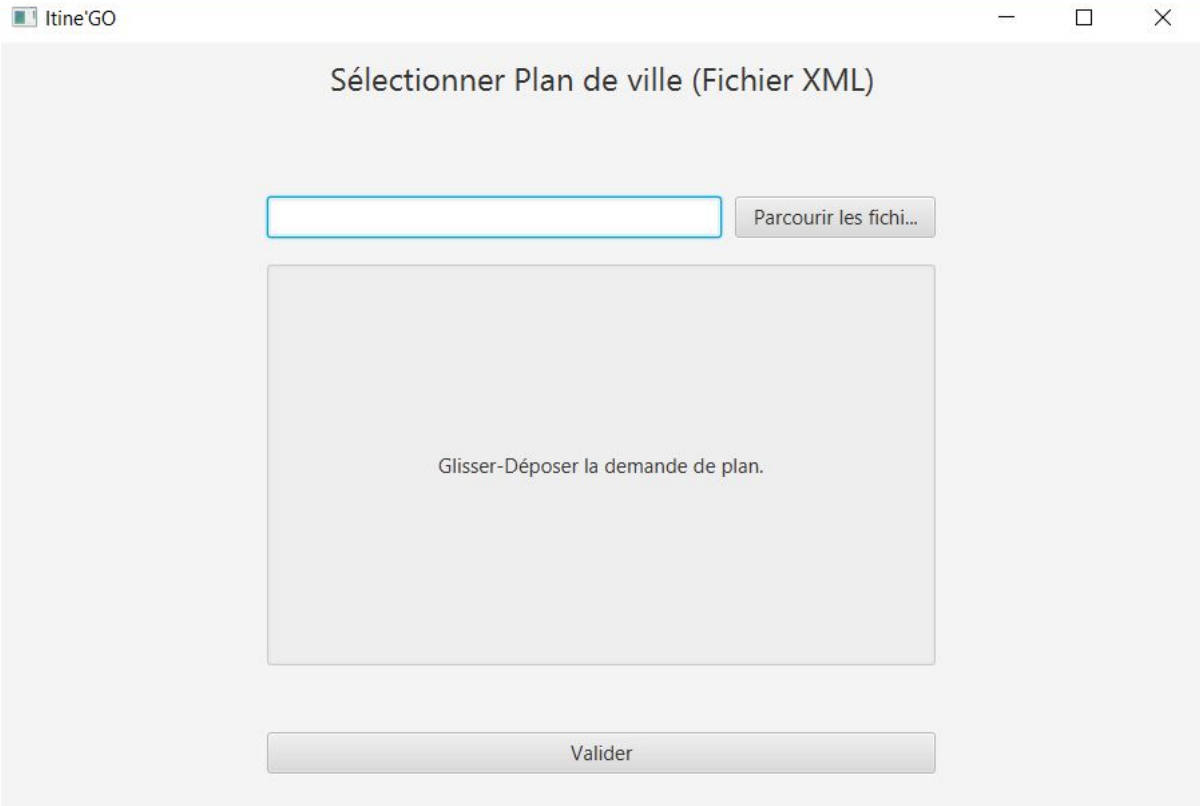
Nous étions invités à utiliser une méthode de développement Agile ainsi qu'un environnement de développement JAVA. Il nous était également demandé d'utiliser les pattern Template rendant plus facile l'aspect itératif de la gestion du projet. Nous avons donc suivi ces indications pour mener à bien le projet Itine'GO.

Equipe :

- **Nicolas Chazeau** : Scrum Manager - Developer
- **Jacques Folleas** : Product Owner - Developer
- **Quentin Vecchio** : Responsable qualité - Developer
- **Ludovic Cros** : Developer
- **Pierre Bayle** : Developer
- **Pierre Jaglin** : Developer
- **Nicolas Gripont** : Developer

II - Cas d’utilisation

A. Charger fichier plan de la ville



Précondition : L’utilisateur a lancé l’application

Scénario Principal :

1. Le système affiche une fenêtre contenant l’explorateur de fichier.

1Bis. L’utilisateur dépose un fichier XML via un Glisser-Déposer.

2. L’utilisateur choisit un fichier XML depuis l’explorateur de fichier.
3. Le système affiche le plan de la ville chargé dans la carte.

Scénario Alternatif :

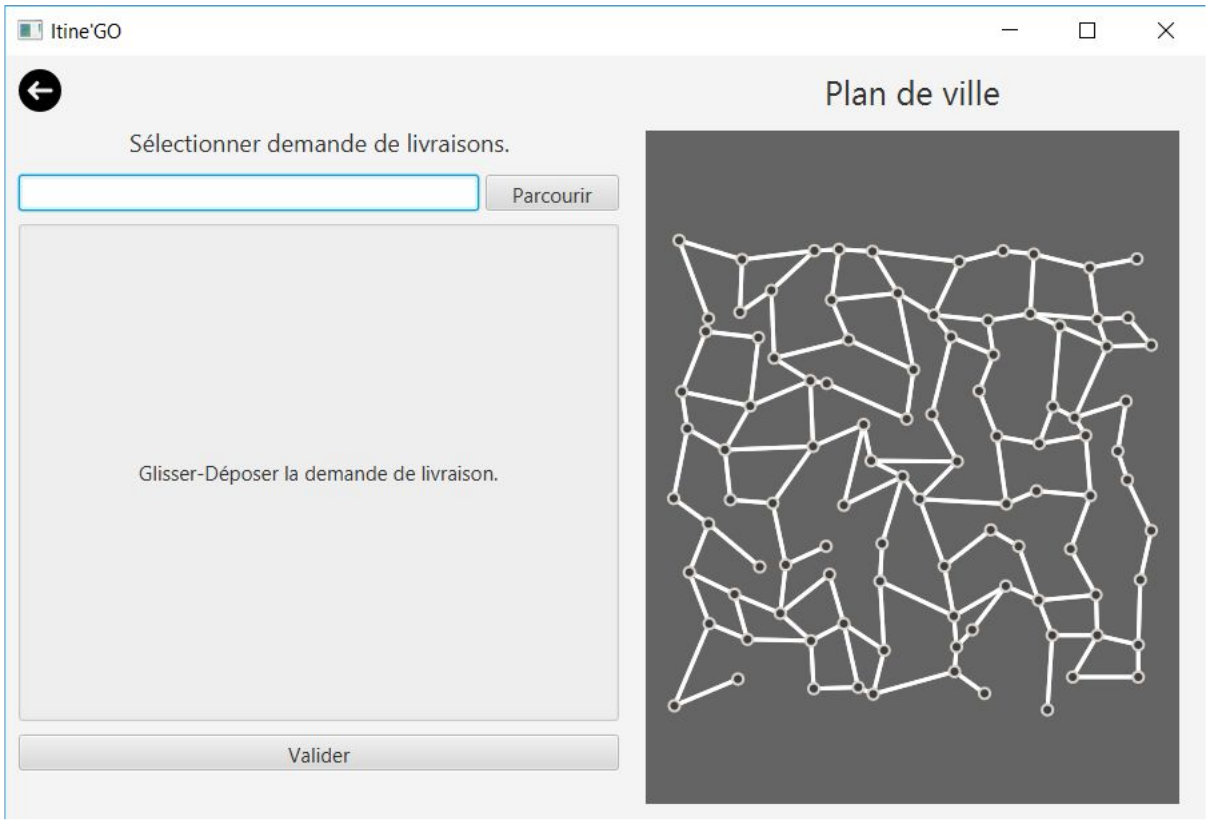
2a. Le fichier choisi n’est pas un fichier XML

Le système indique à l’utilisateur que le fichier n’est pas au format XML et retourne à la page d’accueil/démarrage.

2b et 1Bis. Le fichier choisi contient des données erroné et/ou “mal écrite”.

Le système indique que le fichier ne peut être ajouté en raison d’un problème de lecture et retourne à la page d’accueil/démarrage.

B. Charger fichier livraison



Précondition : L'utilisateur a lancé l'application

Scénario Principal :

1. Le système affiche une fenêtre contenant l'explorateur de fichier.
1Bis. L'utilisateur dépose un fichier XML via un Glisser-Déposer.
2. L'utilisateur choisit un fichier XML depuis l'explorateur de fichier.
3. Le système affiche le plan de la ville chargé dans la carte.

Scénario Alternatif :

2a. Le fichier choisi n'est pas un fichier XML

Le système indique à l'utilisateur que le fichier n'est pas au format XML et retourne à la page d'accueil/démarrage.

2b et 1Bis. Le fichier choisi contient des données erroné et/ou "mal écrite".

Le système indique que le fichier ne peut être ajouté en raison d'un problème de lecture et retourne à la page d'accueil/démarrage.

C. Calculer une tournée

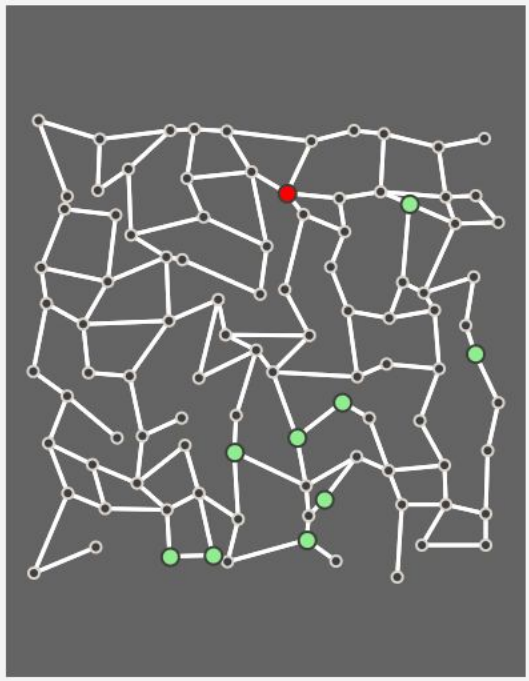
Itine'GO

Adresse de l'entrepôt : 51
Début Livraison à 08:00

Adresse	Plages		Durée
	Début	Fin	
66	-	-	300
68	12:00	14:00	900
39	-	-	900
72	09:00	11:00	900
56	-	-	300
59	13:00	14:00	600
29	-	-	300
95	11:00	13:00	600
47	12:00	14:00	300

Calculer tournée

Plan de ville



Itine'GO

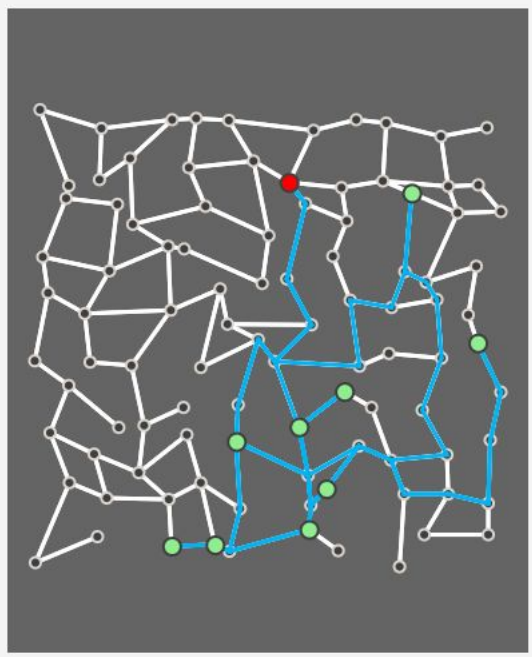
La solution est optimale

Adresse de l'entrepôt : 51
Début de la tournée : 08:00 - Fin de la tournée : 13:28

Adresse	Plages		Horaires		Durée	Attente
	Début	Fin	Arrivée	Départ		
29	-	-	08:24	08:29	300	-
72	09:00	11:00	08:59	09:15	900	57
39	-	-	09:42	09:57	900	-
95	11:00	13:00	10:22	11:10	600	2264
47	12:00	14:00	11:32	12:05	300	1645
66	-	-	12:14	12:19	300	-
68	12:00	14:00	12:26	12:41	900	-
56	-	-	12:46	12:51	300	-
59	13:00	14:00	12:55	13:10	600	255
51	-	-	13:28	-	-	-

Générer fichier

Plan de ville



Précondition : L'utilisateur a chargé un plan de la ville et un plan de livraison

Scénario Principal :

1. L'utilisateur appuie sur le bouton dédié à la procédure de calcul d'une tournée de livraison.

2. Le système calcul la tournée de livraison via une procédure de calcul puis affiche l'itinéraire de livraison dans le plan et la liste des livraisons dans l'ordre des livraisons (adresse, plage horaire début, plage horaire fin, arrivée, départ, durée) dans le tableau se trouvant à gauche du plan.

Scénario Alternatif :

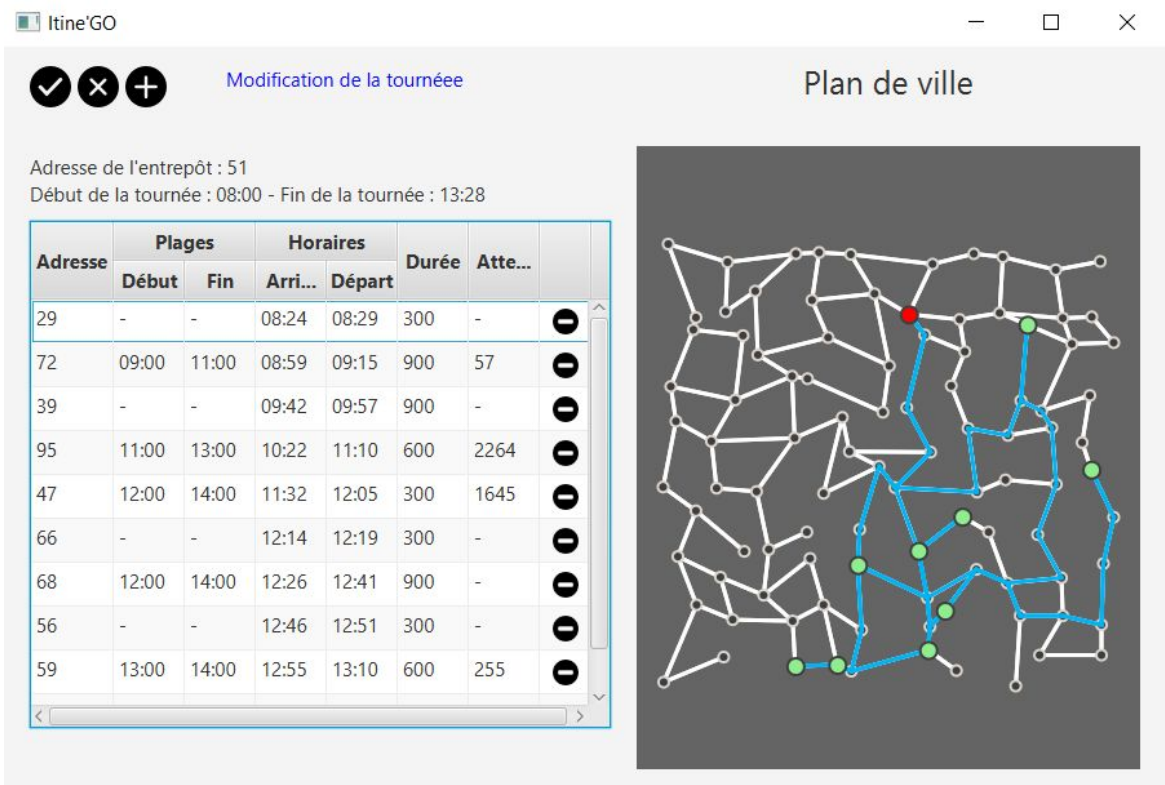
2a Il n'existe pas de tournée respectant les contraintes liées aux plages horaires.

Le système ne prend pas en compte le calcul de la tournée et le signale à l'utilisateur via un message dans l'interface. Le système retourne à la page principale avec le plan et la liste des livraisons.

1-2a L'utilisateur indique au système qu'il souhaite annuler le calcul de la tournée.

Le système annule le calcul de tournée en cours

D. Modifier la tournée



Précondition : Le système a calculé une tournée de livraison valide.

Scénario Principal :

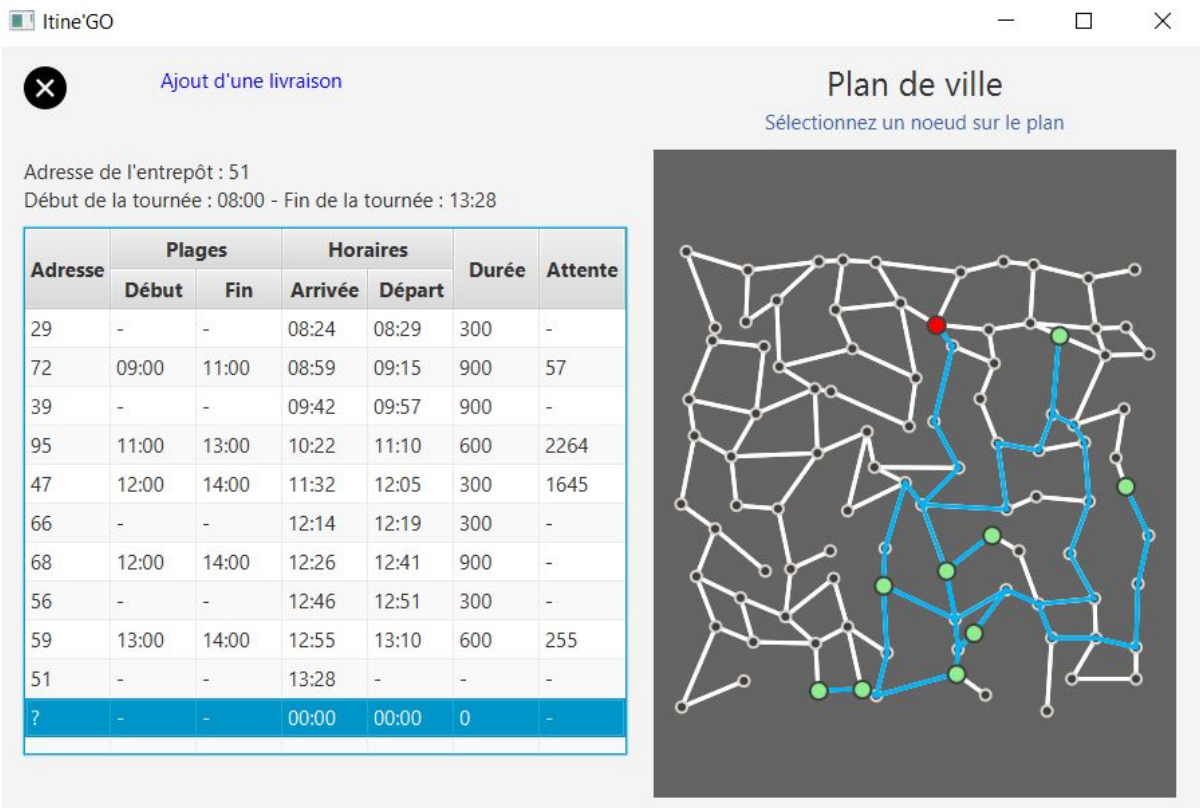
1. L'utilisateur sélectionne le bouton permettant d'effectuer des modifications sur le plan de livraison
2. Le système demande à l'utilisateur quel type de modification il souhaite effectuer sur le plan
3. L'utilisateur sélectionne le type de modification, via un bouton ou une interaction avec le tableau ou le plan de la vie, entre: Ajouter une livraison, Supprimer une livraison, Changer la plage horaire d'une livraison, Échanger l'ordre de deux livraisons.
4. Le système, en fonction du choix de l'utilisateur, va dans le cas d'utilisation de la modification souhaitée

Scénario Alternatif :

1-4a: L'utilisateur indique au système qu'il souhaite annuler la modification de la tournée

Le système annule la modification de la tournée

E. Ajouter une nouvelle livraison



Adresse	Plages		Horaires		Durée	Attente
	Début	Fin	Arrivée	Départ		
29	-	-	08:24	08:29	300	-
72	09:00	11:00	08:59	09:15	900	57
39	-	-	09:42	09:57	900	-
95	11:00	13:00	10:22	11:10	600	2264
47	12:00	14:00	11:32	12:05	300	1645
66	-	-	12:14	12:19	300	-
68	12:00	14:00	12:26	12:41	900	-
56	-	-	12:46	12:51	300	-
59	13:00	14:00	12:55	13:10	600	255
51	-	-	13:28	-	-	-
?	-	-	00:00	00:00	0	-

Précondition : Le système a calculé une tournée de livraison valide et l'utilisateur a choisi de faire une modification sur le plan de livraison.

Scénario Principal :

1. Le système demande à l'utilisateur de lui indiquer la nouvelle livraison directement sur la carte en cliquant sur le noeud voulu
2. L'utilisateur choisit le noeud sur la carte correspondant à la nouvelle livraison
3. Le système met l'adresse du noeud de la nouvelle livraison dans le tableau des livraisons et demande à l'utilisateur de sélectionner la livraison qui suit la nouvelle livraison
4. L'utilisateur via la carte sélectionne la livraison suivante
5. Le système demande à l'utilisateur de modifier la durée de la livraison dans le tableau
6. L'utilisateur renseigne la durée de livraison dans le tableau
7. Le système calcule le plus court chemin entre le nouveau point de livraison et la livraison avant et après le nouveau point de livraison puis met à jour les horaires d'arrivées
8. L'utilisateur confirme le choix de la modification effectuée

Scénario Alternatif :

2a.L'utilisateur sélectionne un point sur la carte qui n'est pas un noeud

Le système ne prend pas en compte la sélection et retourne à l'étape 1

2b. L'utilisateur sélectionne un point sur la carte qui est déjà un point de livraison

Le système renvoie un message d'erreur et met en évidence la livraison sélectionné et retourne à l'étape 1

2c: L'utilisateur sélectionne l'entrepôt comme livraison suivante

Le système ne prend pas en compte la sélection et retourne à l'étape 1

4a: L'utilisateur sélectionne l'entrepôt comme livraison suivante

Le système ne prend pas en compte la sélection et retourne à l'étape 3

4b.L'utilisateur sélectionne un point sur la carte qui n'est pas un noeud

Le système ne prend pas en compte la sélection et retourne à l'étape 3

4c. L'utilisateur sélectionne un point sur la carte qui n'est pas un point de livraison se trouvant dans la liste des livraisons

Le système renvoie une erreur et ne prend pas en compte la sélection et retourne à l'étape 3

6a: L'utilisateur définit une durée qui n'est pas dans le système "horaire" ou négative

Le système renvoie à l'utilisateur un message d'erreur et ne prend pas en compte la durée mise et retourne à l'étape 5

7a: La modification des horaires d'arrivées provoque le non respect de la contrainte des plages horaires

Le système met en surbrillance les plages horaires non valide

8a: L'utilisateur décide de ne pas confirmer son choix

Le système annule les modifications effectuées et retourne à l'étape 1

1-8a: L'utilisateur indique au système qu'il souhaite annuler l'ajout d'une nouvelle livraison

Le système annule l'ajout d'une nouvelle livraison

F. Supprimer une livraison

Précondition : Le système a calculé une tournée de livraison valide et l'utilisateur a choisi de faire une modification sur le plan de livraison

Scénario Principal :

1. Le système propose à l'utilisateur de choisir le point de livraison à supprimer via le tableau des livraisons
2. L'utilisateur choisit un point de livraison à supprimer dans le tableau de livraison
3. Le système supprime le point de livraison sélectionné du plan de livraison et de la carte
4. Le système calcul le plus court chemin entre la livraison d'avant et la livraison suivante puis met à jour toutes les heures d'arrivées des livraisons suivantes
5. L'utilisateur confirme le choix de la modification effectuée

Scénario Alternatif :

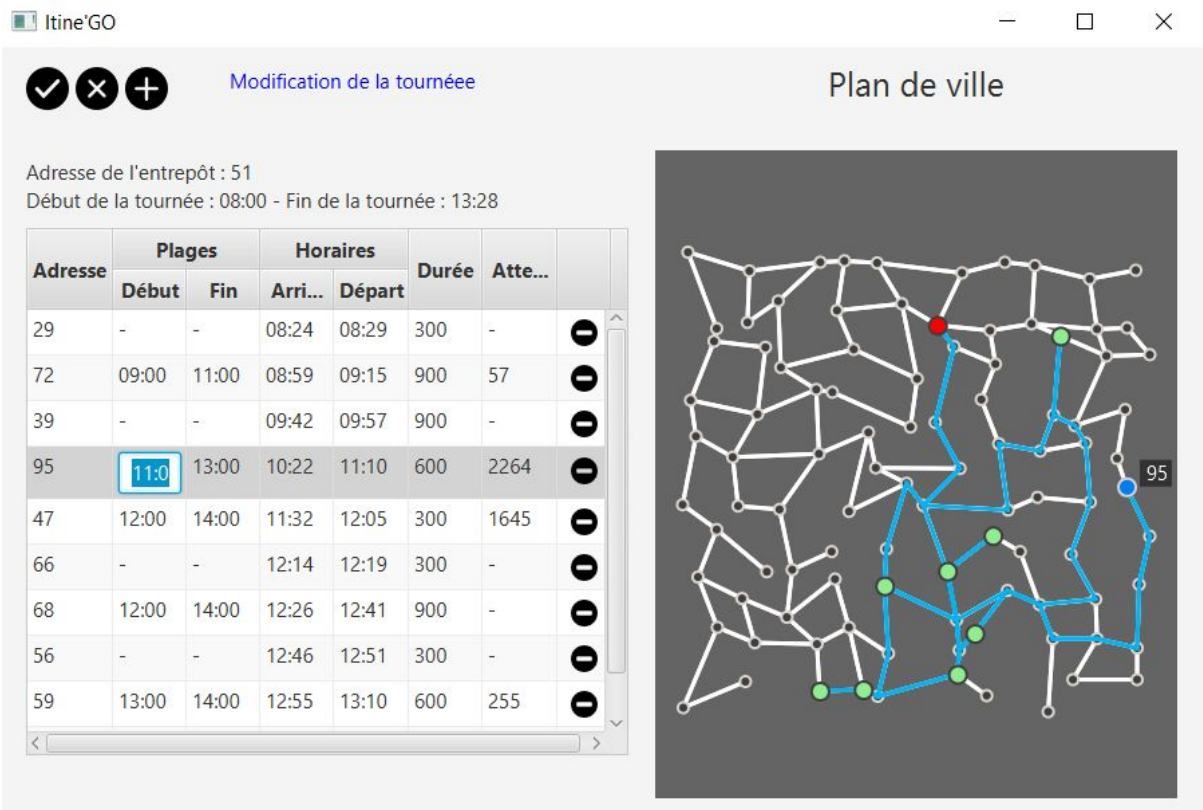
2b. L'utilisateur sélectionne un des points de livraison sur la carte

Le système met en évidence le point de livraison dans la liste de livraison et retourne à l'étape 1

1-5a: L'utilisateur indique au système qu'il souhaite annuler la suppression d'une livraison

Le système annule la suppression d'une livraison

G. Changer une plage horaire



Précondition : Le système a calculé une tournée de livraison valide et l'utilisateur a choisi de faire une modification sur le plan de livraison.

Scénario Principal :

1. Le système demande à l'utilisateur de sélectionner le point de livraison via le tableau de livraison dont il souhaite modifier la plage horaire
2. L'utilisateur sélectionne le point et défini la nouvelle plage horaire du point de livraison sélectionné
3. Le système modifie la plage horaire du point de livraison et met à jour toutes les heures d'arrivées des points de livraisons suivants
4. L'utilisateur confirme le choix de la modification effectuée

Scénario Alternatif :

2a. L'utilisateur définit une plage horaire qui n'est pas dans le "système horaire"

Le système indique à l'utilisateur que la plage horaire n'est pas bonne et retourne à l'étape 1

2b. L'utilisateur définit une plage horaire d'heure de début qui se trouve avant la plage horaire d'heure de fin (ou inversement)

Le système indique à l'utilisateur que la plage horaire n'est pas bonne et retourne à l'étape 1

2c. L'utilisateur sélectionne un point de livraison sur la carte

Le système met la livraison en évidence dans la liste de livraison et retourne à l'étape 1

3a. La modification ne respecte pas la contrainte des plages horaires

Le système met en surbrillance les plages horaires non valide

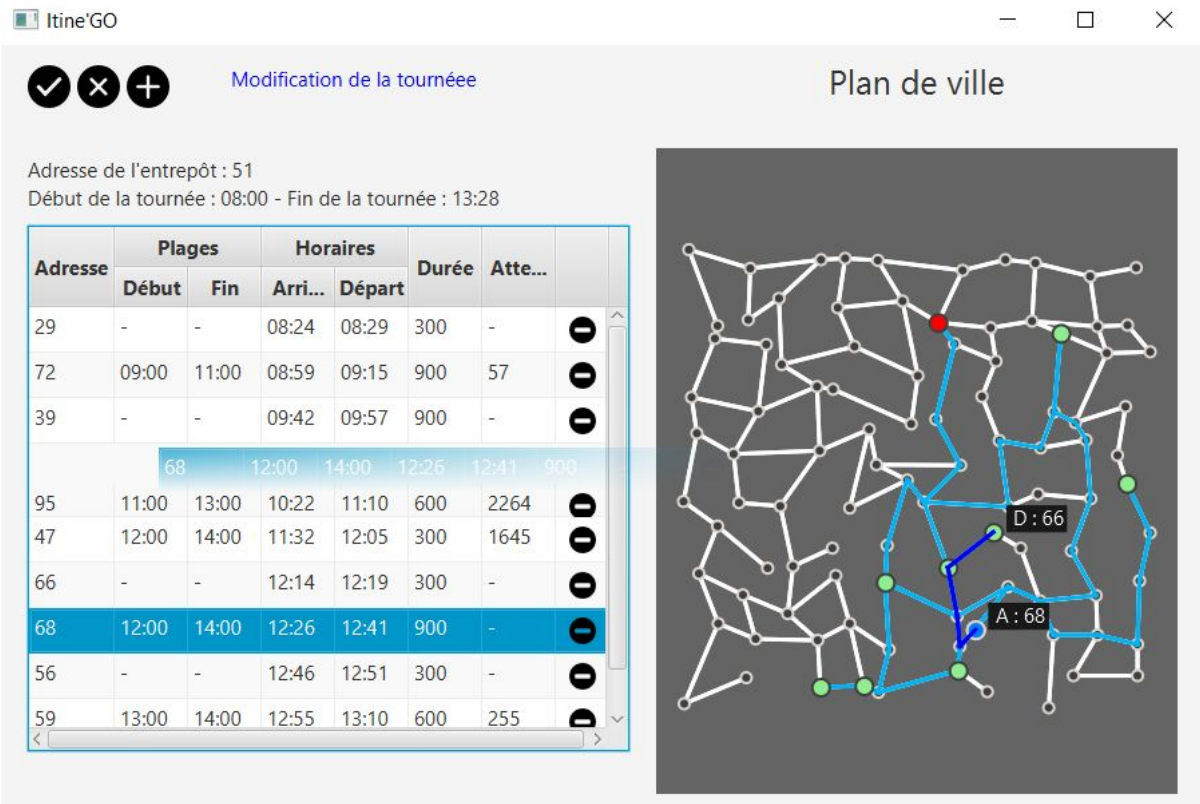
4a: L'utilisateur décide de ne pas confirmer son choix

Le système annule les modifications effectuées et retourne à l'étape 1

1-4a: L'utilisateur indique au système qu'il souhaite annuler la modification de la plage horaire d'un point de livraison

Le système annule la modification de la plage horaire d'un point de livraison

H. Échanger l'ordre d'une livraison



Précondition : Le système a calculé une tournée de livraison valide et l'utilisateur a choisi de faire une modification sur le plan de livraison.

Scénario Principal :

1. L'utilisateur sélectionne une livraison et la fait glisser à l'endroit où il veut l'insérer
2. Le système insère la livraison
3. L'utilisateur confirme le choix de la modification effectuée

Scénario Alternatif :

2a. L'utilisateur décide d'intervertir l'entrepôt avec un des noeud de livraisons.

Le système renvoie un message d'erreur et retourne à l'étape 1

3a. Le changement de l'ordre des livraisons ne respectes pas la contrainte des plages horaires.

Le système indique à l'utilisateur que la contrainte de plage horaire ne peut être respectée.

4a. L'utilisateur indique au système qu'il souhaite annuler l'échange de l'ordre de deux livraisons

Le système annule l'échange de l'ordre de deux points de livraison

I. Générer une feuille de route

```
FEUILLE DE ROUTE

Depart 08:00 du noeud 51
    Prendre : v5 pendant 2504m  (10min)
    Prendre : h5 pendant 331m  (>1min)
    Prendre : v4 pendant 2597m  (10min)
    Prendre : h9 pendant 713m  (2min)
Arrivée 08:24 au noeud 29
    Durée de livraison estimée : 5min

Depart 08:29 du noeud 29
    Prendre : h9 pendant 713m  (2min)
    Prendre : v4 pendant 2597m  (10min)
    Prendre : h5 pendant 1348m  (5min)
    Prendre : v6 pendant 799m  (3min)
    Prendre : h4 pendant 503m  (>1min)
    Prendre : v7 pendant 1407m  (5min)
    Attente de 1min avant livraison
Arrivée 09:00 au noeud 72
    Durée de livraison estimée : 15min

Depart 09:15 du noeud 72
    Prendre : v7 pendant 947m  (4min)
    Prendre : h3 pendant 286m  (>1min)
    Prendre : v8 pendant 2228m  (8min)
    Prendre : h7 pendant 1094m  (4min)
    Prendre : v6 pendant 644m  (2min)
    Prendre : h8 pendant 271m  (>1min)
    Prendre : v5 pendant 288m  (>1min)
    Prendre : h9 pendant 1181m  (4min)
Arrivée 09:42 au noeud 39
    Durée de livraison estimée : 15min
```

Précondition : Le système a calculé une tournée de livraison valide et l'utilisateur ne souhaite plus effectuer de modification sur le plan de livraison

Scénario Principal :

1. L'utilisateur appuie sur le bouton dédié à la procédure de génération d'une feuille de route puis décide du nom de la feuille de route et de son emplacement dans l'explorateur de fichier
2. Le système génère une feuille de route au format texte contenant la liste des livraisons devant être effectué.

Scénario Alternatif :

1-2a L'utilisateur indique au système qu'il souhaite annuler la génération d'une feuille de route.

Le système annule la génération de la feuille de route

III - Explications des choix architecturaux et design patterns utilisés

A. MVC

Le premier design pattern que nous avons choisi d'implémenter est le MVC (Modèle Vue Contrôleur). Il permet de séparer de façon claire et précise le modèle, des vues. Un contrôleur est également déployé pour permettre la communication entre le modèle et la vue.

Un des autres avantages de ce design pattern est qu'il permet d'optimiser le travail en équipe en séparant correctement le travail entre chaque personne. Cependant un tel design pattern a également un inconvénient, il augmente la complexité de l'architecture logicielle.

B. Pattern State

Nous avons choisi d'utiliser ce design pattern car il permet de définir des actions par défaut dans un "état défaut" et de créer d'autres états qui vont redéfinir les actions qui leur sont spécifiques en fonction des besoins des vues de l'application. De plus, les actions de ces états pourront entraîner une transition vers un autre état avec ces actions spécifiques. Cela nous permet d'améliorer la lisibilité du code car il n'y a que les méthodes spécifiques à l'état qui sont définie dans celui-ci et lors de la lecture du diagramme d'état transition les fonctions et transition peuvent être facilement comprise .

Ce pattern va aussi nous permettre de faciliter l'ajout de nouvelles fonctionnalités en ajoutant des états ou des actions dans la classe état par défaut et les états concernés.

C. Pattern Undo/Redo

Pour cette partie nous avons décidé de mettre en place une gestion des commandes lorsque l'on fait une action qui modifie la tournée. Chaque commande correspond à une action de modification (ajouter une livraison, modifier une plage horaire, etc…). Ces commandes stockent les informations nécessaire pour exécuter leurs propres actions de modification ainsi que l'action inverse qui annulera la

modification. Cette implémentation va permettre d'éviter de garder toutes les informations des tournées et de ne garder que celles qui sont utiles à la commande.

Notre modèle d'undo/redo permet d'annuler toutes les commandes des actions qui ont été sauvegardées cependant nous voulions undo et redo des listes de commandes. Pour cela, nous avons rajouté une classe listeModifications qui stocke les liste de commande et undo/redo celles-ci. Avec ce système, on peut par exemple faire plusieurs commandes puis sauvegarder la modification, puis refaire des commandes et sauvegarder, et undo/redo la totalité des commandes d'une modification.

IV - Diagramme de Package et de Classe

A. Diagramme de Package

Voir annexes

B. Diagramme de Classe

Voir annexes

V - Couverture de Test

Package Modele :

Tests sur la classe Plan:

Nom Test	Description	Résultat
testCalculerTournee	Teste si le calcul d'une tourné simple fonctionne	La tournée doit partir de l'entrepôt, aller à la livraison <livraison1>, puis la livraison <livraison 2> et enfin revenir à l'entrepôt en prenant à chaque fois le chemin direct
testAjoutLivraison	Teste si l'ajout d'une livraison à une tourné préalablement calculé fonctionne	La tournée dans partire de l'entrepôt, aller à la livraison <livraison1>, puis la livraison <livraison2> et enfin revenir à l'entrepôt en prenant à chaque fois le chemin direct
testSuppressionLivraison	Teste si la suppression d'une livraison à une tourné préalablement calculé fonctionne	La tournée dans partire de l'entrepôt, aller à la livraison <livraison2>, sans faire la livraison <livraison1> ni passé par son noeud et enfin revenir à l'entrepôt en prenant à chaque fois le chemin direct
testModifOrdreLivraisonMontant	Teste si le déplacement de l'ordre de passage d'une livraison à une tournée préalablement calculé fonctionne. On	La tournée devra partir de l'entrepôt, et passer dans l'ordre par les livraisons : 2-3-1-4 et revenir à l'entrepôt. Elle devra

	s'intéresse particulièrement à une livraison effectué plus tard.	toujours emprunté les tronçons: 1->2, 2->3, 3->4, 4->5, 5->1
testModifOrdreLivraisonDescendant	Teste si le déplacement de l'ordre de passage d'une livraison à une tournée préalablement calculé fonctionne. On s'intéresse particulièrement à une livraison effectué plus tôt.	La tournée devra partir de l'entrepôt, et passer dans l'ordre par les livraisons : 3-1-2-4 et revenir à l'entrepôt. Elle devra toujours emprunté les tronçons: 1->2, 2->3, 3->4, 4->5, 5->1

Tests sur la classe Entrepot:

Nom Test	Description	Résultat
testEntrepotNoeudIntIntInt	Test du constructeur de la classe Entrepot avec un noeud, une heure, une minute et une seconde	L'affiche de l'entrepôt doit correspondre au résultat souhaité
testEntrepotHoraireImpossible	Test du constructeur avec une horaire impossible (Ex: 25h70min100s)	L'heure de départ de l'entrepôt doit être à 0h0m0s
testEntrepotNoeudHoraire	Test du constructeur avec une horaire de type Horaire	L'affiche de l'entrepôt doit correspondre au résultat souhaité
testEntrepotNoeudString	Test de l'affichage texte de la classe Entrepot	L'affiche de l'entrepôt doit correspondre au résultat souhaité

testEntrepotNoeudStringMauvaisFormat	Test de l'affichage texte de la classe Entrepot avec un mauvais format pour l'heure	L'heure de départ de l'entrepôt doit être à 0h0m0s
testSetNoeudNull	Test lorsqu'on met le noeud de l'entrepôt à null	L'identifiant du noeud de l'entrepôt doit être à -1 avec [0,0] comme coordonnée
testSetHoraireDepart	Test pour la modification de l'heure de départ de l'entrepôt	L'heure de départ doit être modifié à 0h0m0s

Tests sur la classe Horaire:

Nom Test	Description	Résultat
testHoraireIntIntInt	Test constructeur de la classe Horaire avec 3 entier (Heure,Minute,Seconde)	
testHoraireIntIntIntMauvaiseHeure/Minute/Seconde	Test constructeur de la classe Horaire avec une heure/minute/seconde impossible	Heure/Minute/Seconde est mise à 0
testHoraireString	est affichage texte de Horaire	L'affiche doit correspondre au résultat attendu
testHoraireMalForme	Test affiche texte mal formé d'Horaire	L'affiche doit correspondre au résultat attendu
testAjouterHeure/Minute/Seconde	Test permettant de modifier l'heure/minute/seconde d'Horaire	L'horaire doit correspondre au résultat attendu

testSetHeure/Minute/SecondeMauvais	Test permettant de modifier l'heure/minute/seconde de l'Horaire avec une heure/minute/seconde mauvaise	L'horaire doit correspondre au résultat attendu
testSetHeure/Minute/Seconde	Test permettant de modifier la heure/minute/seconde de l'Horaire	L'affiche de l'attribut doit correspondre au résultat souhaité
testGetHoraireEnMinute	Test permettant d'avoir l'horaire en minute	L'horaire doit correspondre au résultat attendu

Tests sur la classe Livraison:

Nom Test	Description	Résultat
testLivraisonNoeudIntIntIntIntIntIntInt	Test du constructeur de la classe Livraison avec un noeud et que des entiers	L'affiche de la livraison doit correspondre au résultat souhaité
testLivraisonHorairesImpossible	Test du constructeur de la classe Livraison avec un noeud et que des entiers et une horaire impossible	L'affiche de la livraison doit correspondre au résultat souhaité
testLivraisonNoeudIntHoraireHoraire	Test du constructeur de la classe Livraison avec un noeud et un entier et deux types horaires	L'affiche de la livraison doit correspondre au résultat souhaité
testLivraisonNoeudIntHoraireHoraireAvecNull	Test du constructeur de la classe Livraison avec un noeud, un entier, une horaire non nulle et une horaire nulle	L'affiche de la livraison doit correspondre au résultat souhaité

testLivraisonNoeudIntStringString	Test du constructeur de la classe Livraison avec un noeud, un entier et des horaires en string	L'affiche de la livraison doit correspondre au résultat souhaité
testLivraisonNoeudIntStringStringMalForme	Test du constructeur de la classe Livraison avec un noeud, un entier, une horaire et une horaire mal formée	L'affiche de la livraison doit correspondre au résultat souhaité
testGet….	Test sur les getter des attributs	Doit retourner le …(attribut) de la livraison
testSet...	Test sur les setter des attributs	L'affiche de l'attribut doit correspondre au résultat souhaité

Tests sur la classe Noeud:

Nom Test	Description	Résultat
testNoeud	Test du constructeur de la classe Noeud avec trois entiers	L'affiche du noeud doit correspondre au résultat souhaité
testGet….	Test sur les getter des attributs	Doit retourner le …(attribut) du noeud
testSet...	Test sur les setter des attributs	L'affiche de l'attribut doit correspondre au resultat souhaité

Tests sur la classe ParseurLivraison:

Nom Test	Description	Résultat

testParseurLivraison	Test du Parseur sur un fichier de Livraison correct	Le fichier est parsé
testParseurLivraisonAvecPlage	Test du Parseur sur un fichier de Livraison correct ayant des plages horaire	Le fichier est parsé
testMauvaisFormat	Test du Parseur sur un fichier de Livraison au mauvais format	Le fichier n'est pas parsé et renvoie une exception
testEntrepotInconnue	Test du Parseur sur un fichier de Livraison ayant un noeud pour l'entrepot inconnu	Le fichier n'est pas parsé et renvoie une exception
testPasDeLivraison	Test du Parseur sur un fichier de Livraison n'ayant pas de livraison	Le fichier n'est pas parsé et renvoie une exception
testEntrepotMoinsUn	Test du Parseur sur un fichier de Livraison ayant un identifiant d'entrepôt égale à -1	Le fichier n'est pas parsé et renvoie une exception
testLivraisonMoinsUn	Test du Parseur sur un fichier de Livraison ayant un identifiant de livraison égale à -1	Le fichier n'est pas parsé et renvoie une exception
testIdNoeudInexistant	Test du Parseur sur un fichier de Livraison ayant un noeud de livraison qui n'existe pas	Le fichier n'est pas parsé et renvoie une exception
testParseurLivraisonAvecPlageDébutSupPlageFin	Test du Parseur sur un fichier de Livraison correct ayant une plage horaire de début plus	Le fichier est parsé sans la plage horaire fausse

	tard que la plage horaire de fin	
testParseurLivraisonAvecPlageFinInfPlageDebut	Test du Parseur sur un fichier de Livraison correct ayant une plage horaire de fin plus tôt que la plage horaire de début	Le fichier est parsé sans la plage horaire fausse
testParseurLivraisonAvecPlageIdentique	Test du Parseur sur un fichier de Livraison correct ayant une plage horaire de début égale à la plage horaire de fin	Le fichier est parsé sans la plage horaire fausse

Tests sur la classe ParseurPlan:

Nom Test	Description	Résultat
testParseurPlan	Test du Parseur sur un fichier Plan correct	Le fichier est parsé
testMauvaisFormat	Test du Parseur sur un fichier Plan au mauvais format	Le fichier n'est pas parsé et renvoie une exception
test2IdentifiantIdentique	Test du Parseur sur un fichier Plan ayant deux identifiants identiques	Le fichier n'est pas parsé et renvoie une exception
testOrigineInconnue	Test du Parseur sur un fichier Plan ayant un tronçon dont l'origine est un noeud non connue	Le fichier n'est pas parsé et renvoie une exception
testDestinationInconnue	Test du Parseur sur un fichier Plan ayant un tronçon dont la destination est un noeud non connue	Le fichier n'est pas parsé et renvoie une exception

testIdentifiantNegatif	Test du Parseur sur un fichier Plan ayant un identifiant de noeud ou de tronçon à -1	Le fichier n'est pas parsé et renvoie une exception
testUnNoeud	Test du Parseur sur un fichier Plan ayant un seul noeud	Le fichier n'est pas parsé et renvoie une exception
testVitesseNegative	Test du Parseur sur un fichier Plan ayant des tronçons avec une vitesse négative	Le fichier n'est pas parsé et renvoie une exception
testlongueurNegative	Test du Parseur sur un fichier Plan ayant des tronçons avec une longueur négative	Le fichier n'est pas parsé et renvoie une exception
testCoordonneeNegatif	Test du Parseur sur un fichier Plan ayant des tronçons avec des coordonnées négatif	Le fichier n'est pas parsé et renvoie une exception

Tests sur la classe Trajet:

Nom Test	Description	Résultat
testTrajet	Test du constructeur de la classe Trajet plusieurs noeud formant plusieurs tronçons formant un trajet	La liste des trajet doit correspondre au résultat souhaité
TestGet….	Test sur les getters des attributs de trajets	Doit retourner le …(attribut) du trajet

Tests sur la classe Tronçon:

Nom Test	Description	Résultat

testTroncon	Test du constructeur de la classe Tronçon avec un nom, une longueur, une vitesse et un noeud de départ et d'arrivé	L'affiche du tronçon doit correspondre au résultat souhaité
testGet….	Test sur les getter des attributs	Doit retourner le …(attribut) de tronçon
testSet...	Test sur les setter des attributs	L'affiche de l'attribut doit correspondre au résultat souhaité

VI - Itérations

A. 1ere Itération

Planning Prévisionnel :

Tâches	**H/P**
Diagramme modèle du domaine	4
Glossaire	2
Diagramme cas d'utilisation (CU)	4
Description textuelle CU	2
Diagramme États-transitions	4
Diagramme de classe contrôleur	4
Diagramme de classe vue	4
Diagramme de packages	4
Recherche Algorithme	8
Diagramme de séquence calcul d'une tournée	4
Mockups	4
Implémentation des pages (et prise en main JavaFX)	5
Implémentation algorithme de calcul	8
Implémentation modèle domaine	6
Implémentaton contrôleur	6
Tests JUnit (unitaires et fonctionnels)	4
Planning + Taiga	1
TOTAL h/P	**82**

Planning Effectif :

Membres	**Jacques F.**	**Ludovic C.**	**Nicolas C.**	**Nicolas G.**	**Pierre B.**	**Pierre J.**	**Quentin V.**	**TOTAL h/P**
Tâches								
Diagramme modèle du domaine		1	4		4			**9**
Glossaire			1		1			**2**
Diagramme cas d'utilisation (CU)						4	2	**6**
Description textuelle CU						4		**4**
Diagramme États-transitions		2		2,5				**4,5**
Diagramme de classe modèle		2	1		4		1	**8**
Diagramme de classe controleur		4			4			**8**
Diagramme de classe vue				1			1	**2**
Diagramme de packages		1					1	**2**
Recherche Algorithme	8							**8**
Diagramme de séquence calcul d'une tournée	2							**2**
Mockups		3		3				**6**
Implémentation des pages (et prise en main JavaFX)				8			6	**14**
Implémentation algorithme de calcul	8		4	2				**14**
Implémentation parseurs			3			5		**8**
Implémentation modèle domaine			2	0,5			3	**5,5**
Implémentation contrôleur		4		1	4		1	**10**
Tests JUnit (unitaires et fonctionnels)						4	3	**7**
Planning + Taiga			2					**2**
TOTAL h/P	**18**	**17**	**17**	**18**	**17**	**17**	**18**	**122**

B. 2éme Itération

Planning Prévisionnel :

Tâches	TOTAL h/P
Description textuelle structurée des cas d'utilisation	2
Description choix architecturaux et design patterns utilisés	2
Description JavaDoc	4
Amélioration de l'algorithme de calcul	6
Amélioration design de l'application	6
Mise à jour du diagramme de classe	1
Mise à jour des mockups existant	2
Implémentation de la génération d'une feuille de route	4
Implémentation des erreurs parseurs	2
Implémentation de la visualisation avancée du plan	8
Implémentation de la visualisation avancée du tableau	3
Amélioration graphique (Affichage Solution optimale, Mise à jour pendant la recherche d'une solution, ...)	8
Implémentation de l'ajout d'une livraison	4
Implémentation de la suppression d'une livraison	4
Implémentation de la modification d'une plage horaire	4
Implémentation de la modification ordre de tournée	4
Implémentation de la plage horaire dans algorithme de calcul	4
Implémentation de TSP2	4
Implémentation de TSP3	4
Amélioration contrôleur et état	6
Rédaction rapport	4
Implémentation test JUnit	10
Correction de bug	4
Reunion de projet	4
Planning + Taiga	3
Total H/P	107

Planning Effectif :

Membres	**Jacques F.**	**Ludovic C.**	**Nicolas C.**	**Nicolas G.**	**Pierre B.**	**Pierre J.**	**Quentin V.**	**TOTAL h/P**
Tâches								
Description textuelle structurée des cas d'utilisation						4		4
Description choix architecturaux et design patterns utilisés			4					4
Description JavaDoc			4		2	**4**		10
Amélioration de l'algorithme de calcul	4	4		**2**				10
Amélioration design de l'application						**4**		
Mise à jour du diagramme de classe				3			6	9
Mise à jour des mockups existant		2						2
Implémentation de la génération d'une feuille de route		2		2				4
Implémentation des erreurs parseurs			4			3		7
Implémentation de la visualisation avancée du plan			**2**	2	4		2	10
Implémentation de la visualisation avancée du tableau				2			6	8
Amélioration graphique (Affichage Solution optimale, Mise à jour pendant la recherche d'une solution, ...)				2			4	6
Implémentation de l'ajout d'une livraison		2		1	1		4	8
Implémentation de la suppression d'une livraison	4						**2**	6

Implémentation de la modification d'une plage horaire			2	**2**				4
Implémentation de la modification ordre de tournée				2	4			6
Implémentation de la plage horaire dans algorithme de calcul	2	2		**2**			**2**	8
Implémentation de TSP2	4	2			4			10
Implémentation de TSP3	4	2						6
Amélioration contrôleur et état	**4**		2					6
Rédaction rapport		3	2		3			8
Implémentation test JUnit						5		5
Correction de bug	4	4	1	2	1		4	16
Reunion de projet	**1**	**1**	1	**1**	**1**	1	**1**	7
Planning + Taiga				**1**	**2**			**3**
TOTAL h/P	**27**	**24**	**22**	**24**	**22**	**21**	**31**	**167**

C. 3éme Itération

Planning Prévisionnel :

Tâches	**H/P**
Description JavaDoc	4
Amélioration de l'algorithme de calcul	4
Amélioration design de l'application	3
Mise à jour du diagramme de classe	2
Mise à jour des mockups existant	3
Finir l'ajout d'une livraison	4
Finir la suppression d'une livraison	6
Finir la modification d'une plage horaire	6
Finir la modification ordre de tournée	6
Finir l'implémentation de la plage horaire dans algorithme de calcul	4
Correction de bug graphique	6
Finir l'implémentation de TSP2	4
Finir l'implémentation de TSP3	4
Finir l'implémentation de TSP4	4
Implémentation test JUnit	8
Finir l'implémentation de l'UNDO/REDO	4
Correction de bug	8
Ecriture du rapport	2
Reunion de projet	4
Planning + Taiga	3
Total H/P	**91**

Planning Effectif :

Membres	**Jacques F.**	**Ludovic C.**	**Nicolas C.**	**Nicolas G.**	**Pierre B.**	**Pierre J.**	**Quentin V.**	**TOTAL h/P**
Tâches								
Description JavaDoc			2			4		4
Amélioration de l'algorithme de calcul	2							2
Amélioration design de l'application				2			3	5
Mise à jour du diagramme de classe		2			2			4
Mise à jour des mockups existant		1		1				
Finir l'ajout d'une livraison	1	1	1				2	5
Finir la suppression d'une livraison	2	2		2			2	8
Finir la modification d'une plage horaire	2	2	2	2			2	10
Finir la modification ordre de tournée	4	4			2		4	14
Finir l'implémentation de la plage horaire dans algorithme de calcul	1		**2**					3
Correction de bug graphique				4	2		3	8
Finir l'implémentation de TSP2	1							1
Finir l'implémentation de TSP3	1	2			1			4
Finir l'implémentation de TSP4	1							1
Implémentation test JUnit	2	2	2	**2**	2	3	2	15
Finir l'implémentation de l'UNDO/REDO	2	2	2	2	2	1	2	13
Correction de bug	2	2	2	**2**			**2**	8
Ecriture du rapport			6	1		6		12
Reunion de projet	1	1	1	1	1	1	1	7
Planning + Taiga				1	2			3
TOTAL h/P	**22**	**21**	**20**	**20**	**14**	**15**	**23**	**126**

VII - Utilisation du code (Instructions)

Importer le projet :

- Dans la barre d'outils, cliquez sur "File" > "Import…"
- Choisissez "Import projects from Folder or Archive"
- Cliquez sur "Directory…"
- Sélectionnez le dossier contenant les sources du projet
- Cochez la case devant ItineGo
- Cliquez sur "Finish"

JRE :

- Vérifiez que vous utilisez Java 8.

Installation du plugins JavaFX :

- Dans la barre d'outils, cliquez sur "Help" > "Eclipse Marketplace..."
- Recherchez "javafx"
- Installez "e(fx)clipse"

Vous pouvez maintenant compiler/exécuter le code.

Ouvrir les IHM dans le Scene Builder (fichiers '.fxml'):

- Téléchargez et installez JavaFX Scene Builder 2.0
- Dans Eclipse, clique droit sur un fichier '.fxml' > "Open with Scene Builder"

VIII - Glossaire

Plan : Carte de l'ensemble des noeuds et tronçons.

Noeud : Cela correspond à un point sur le plan (coordonnées X, Y), il a un identifiant unique.

Adresse : Cela correspond à un noeud.

Tronçon : Une arête, segment de route. Qui a une origine (noeud de départ), une destination (noeud d'arrivée), cela détermine donc un sens. Contient également une longueur, vitesse et nom de rue.

Trajet : Suite de tronçons qui permet d'aller d'une livraison de départ à une livraison d'arrivée.

Entrepôt/dépôt : Noeud de départ de la tournée. Contient une heure de départ.

Tournée : Liste triée des livraisons à effectuer. Une suite de trajets est ensuite calculée permettant d'obtenir un ordre de livraison.

Livraison : Noeud à livrer, contient une durée de livraison, et éventuellement une plage horaire.

Coût (d'une tournée) : temps total que va prendre le parcours total d'une tournée ainsi que les livraisons à effectuer.

Noeud Départ : C'est un noeud correspondant au point de départ d'un tronçon.

Noeud Arrivée : C'est un noeud correspondant au point d'arrivée d'un tronçon.

Feuille de route : Correspond à un fichier donnant la liste des livraisons dans l'ordre à effectuer.

Origine : C'est un noeud de départ d'un tronçon.

Destination : C'est un noeud d'arrivée d'un tronçon.

IX - Conclusion

À la suite des 6 semaines de développement, nous avons réussi à avoir un livrable complet, comprenant l'ensemble des fonctionnalités demandées par le client, et réfléchi dans le but que son utilisation soit la plus agréable possible.

La réalisation de ce projet nous a permis de monter en compétence sur de nombreux points.

Tout d'abord d'un point de vue technique, le logiciel à implémenter étant relativement complet il nous a permis de consolider nos connaissances en orienté objet avec le langage Java, et en réalisation d'IHM via la technologie JavaFX, mais également en qualité logiciel via les bandes de test Junit, et enfin en mise en place d'architecture logiciel complexe avec l'utilisation de design pattern comme MVC, Pattern State, ou encore le Pattern Template.

Il a également été formateur du point de vue de la gestion de projet avec l'utilisation des méthodes de développement agile, ainsi que le travail en coopération avec une équipe importante sur une longue période.