Informaticien/-ne CFC
Travail pratique individuel 2024 (TPI)

ICT Berufsbildung
Formation professionnelle
Freiburg-Fribourg

Nom du candidat : Elvin Kuci

« Road Traffic Simulator »

Sommaire

Résumé du rapport du TPI

Situation de départ

Le projet « Road Traffic Simulator » a été entrepris sans aucune base préexistante, ce qui signifie qu'il n'y avait pas de situation initiale à analyser. L'objectif principal était de développer une application informatique capable de simuler un carrefour routier. Cette application devait permettre la modification et la sauvegarde de divers paramètres tels que les feux de signalisation, la vitesse de la route et autres variables essentielles pour tester et visualiser différentes configurations de trafic.

Mise en œuvre

Pour mener à bien ce projet, une méthodologie en cascade a été adoptée. Cette approche linéaire et séquentielle a permis de structurer le développement en plusieurs phases distinctes, chacune devant être complétée avant de passer à la suivante. Le processus a débuté par la mise en place d'un projet JavaFX, suivi de la création et de l'intégration de divers modules nécessaires pour simuler le trafic routier. Des tests approfondis ont été effectués pour s'assurer que chaque composant fonctionnait correctement avant l'intégration finale.

Résultats

Le résultat final est une application interactive permettant aux utilisateurs de simuler et de tester différentes configurations de carrefour. L'application gère les routes à circulation bidirectionnelle et les feux de signalisation, avec une extension prévue pour les ronds-points. L'utilisateur peut ajuster les paramètres de la simulation, tels que la vitesse, la densité du trafic, et les temps de réaction, puis sauvegarder ces paramètres pour des tests ultérieurs. Toutefois, certains aspects du projet n'ont pas pu être réalisés dans le délai imparti, ce qui inclut des fonctionnalités avancées et des optimisations supplémentaires. Malgré ces limitations, le projet a atteint ses principaux objectifs en fournissant un outil flexible et paramétrable pour la gestion et l'analyse des flux de trafic.

1 Les grandes lignes du projet

1.1 Analyse de la situation initiale

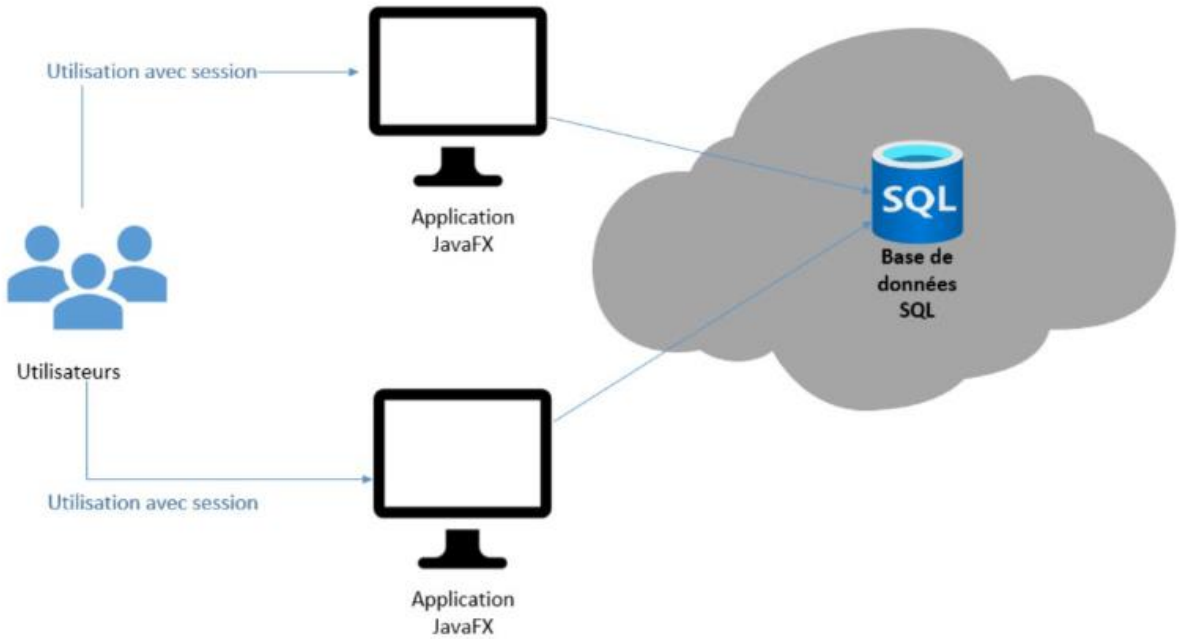
Cette application n'a pas de situation initiale. Ce mandat est le premier mandat sur ce projet et tout est à faire depuis zéro.

1.2 Analyse de l'état désiré

L'objectif final est d'avoir une application pour ordinateur permettant de lancer une simulation d'un carrefour. Il sera possible de modifier et sauvegarder de nombreux paramètres telle que les feux, la vitesse de la route, etc… Cette application permettra de tester et visualiser un ensemble de paramètres appliqué sur un carrefour.

1.3 Cahier des charges / exigences du système

L'application devra gérer les 4 tronçons de routes, à circulation bidirectionnelle, liée par un carrefour équipé de feux de signalisation. Une extension possible devra être la simulation de ces mêmes tronçons de route avec un rond-point. Ce développement ne devra pas être réalisé, mais pris en compte dans l'analyse et la conception, afin que l'application puisse être modifiée facilement.



L'objectif est aussi de fournir à l'utilisateur une application paramétrable pour agir sur le résultat visuel de la simulation :

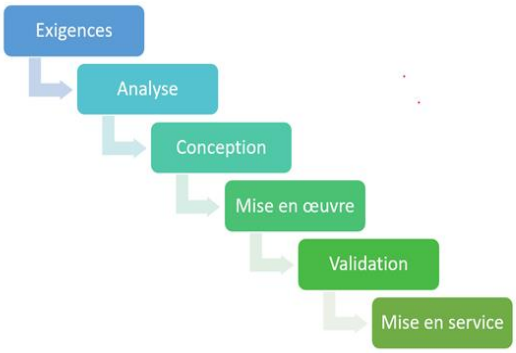
- Échelle pour la représentation graphique en pixel par mètres
- Limitation de la vitesse, distance des et densité du trafic

- Temps de réaction, accélération, décélération et temps de sécurité
- Temps des feux et ordre de ceux-ci
- Sauvegarde du jeu de paramètre sous un nom

1.4 Organisation du projet

1.4.1 Méthodologie

Pour ce projet, une méthodologie en cascade, également connue sous le nom de modèle séquentiel. Cette méthodologie est une approche traditionnelle de gestion de projet qui se caractérise par un processus linéaire et séquentiel. Chaque phase du projet doit être complétée avant de passer à la suivante.

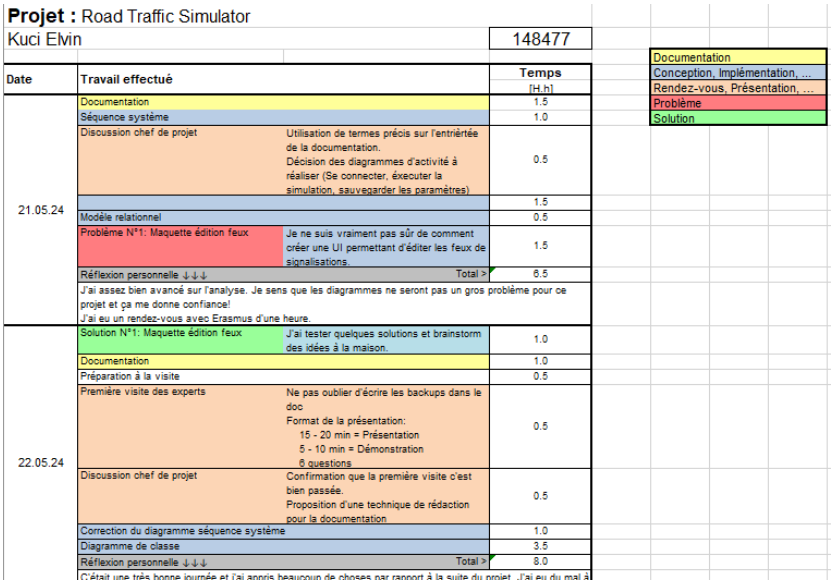


Ses avantages sont sa simplicité, chaque étape produit une documentation détaillée et elle est facile à gérer. Cette méthodologie a aussi des inconvénients telle que sa rigidité, son adaptabilité aux problèmes.

1.4.2 Supports

L'utilisation d'un planning et d'un journal de travail dans le cadre de la méthodologie en cascade est essentielle pour garantir la rigueur et la clarté du projet. La nature séquentielle de cette méthodologie nécessite une documentation précise à chaque étape pour s'assurer que les exigences sont bien comprises et que chaque phase est complétée avant de passer à la suivante. Un planning détaillé permet de définir des jalons clairs et des délais pour chaque phase, facilitant ainsi la gestion et le suivi de l'avancement du projet.

Le journal de travail joue un rôle crucial en fournissant une trace écrite des décisions, des modifications et des progrès réalisés. Cela est particulièrement important dans la méthodologie en cascade, où revenir en arrière pour apporter des modifications peut être difficile et coûteux. En tenant un journal de travail, il est possible de documenter les raisons des choix effectués et de conserver un historique des activités, facilitant ainsi la gestion de la phase de maintenance et la résolution des problèmes après la livraison.



Projet : Road Traffic Simulator

Kuci Elvin 148477

Documentation
Conception, Implémentation, ...
Rendez-vous, Présentation, ...
Problème
Solution

Date	Travail effectué		Temps [H.h]
21.05.24	Documentation		1.5
	Séquence système		1.0
	Discussion chef de projet	Utilisation de termes précis sur l'entriértée de la documentation. Décision des diagrammes d'activité à réaliser (Se connecter, éxecuter la simulation, sauvegarder les paramètres)	0.5
			1.5
	Modèle relationnel		0.5
	Problème N°1: Maquette édition feux	Je ne suis vraiment pas sûr de comment créer une UI permettant d'éditer les feux de signalisations.	1.5
	Réflexion personnelle ↓↓↓	Total >	6.5
	J'ai assez bien avancé sur l'analyse. Je sens que les diagrammes ne seront pas un gros problème pour ce projet et ça me donne confiance! J'ai eu un rendez-vous avec Erasmus d'une heure.		
22.05.24	Solution N°1: Maquette édition feux	J'ai tester quelques solutions et brainstorm des idées à la maison.	1.0
	Documentation		1.0
	Préparation à la visite		0.5
	Première visite des experts	Ne pas oublier d'écrire les backups dans le doc Format de la présentation: 15 - 20 min = Présentation 5 - 10 min = Démonstration 8 questions	0.5
	Discussion chef de projet	Confirmation que la première visite c'est bien passée. Proposition d'une technique de rédaction pour la documentation	0.5
	Correction du diagramme séquence système		1.0
	Diagramme de classe		3.5
	Réflexion personnelle ↓↓↓	Total >	8.0
	C'était une très bonne journée et j'ai appris beaucoup de choses par rapport à la suite du projet. J'ai eu du mal à		

1.4.3 Participants

Pour garantir le succès du projet, voici la liste des participants impliqués dans ce projet de TPI :

Participant	Rôle
Kuci Elvin	Candidat, développeur de l'application
Bütschi Daniel	Supérieur professionnel
Hertling Frédéric	Formateur en entreprise
Perroud Didier	Expert principal
Thomet Samuel	Expert secondaire

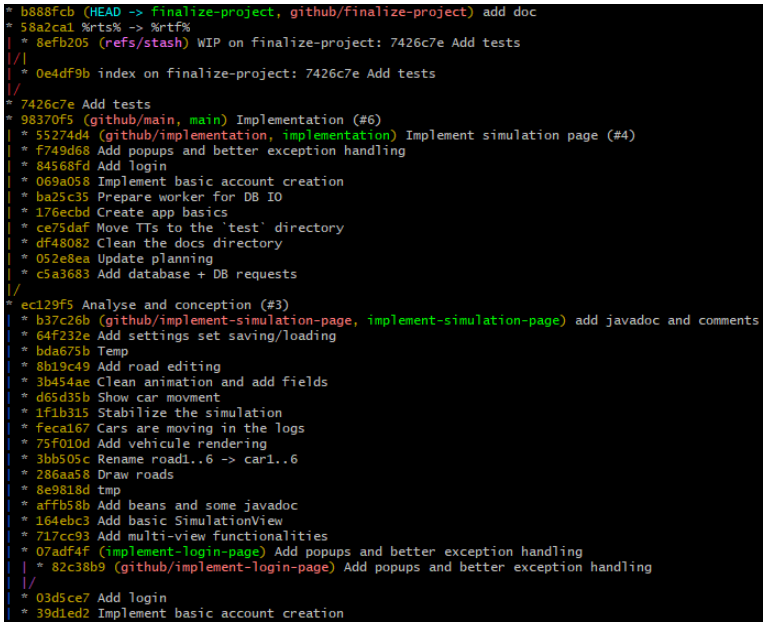
1.4.4 Sauvegarde et préservation

Le code et la documentation sont présents dans le Git du projet. Un `commit` est émis à chaque modification d'un élément dans le code et/ou la documentation.

Chaque groupe de `commit` majeur est édité sur sa propre branche et est merge sur la branche « `main` » une fois que le travail sur celle-ci est terminé.

Une fois que l'application atteint un objectif et est stable, une branche « `version_x.y.z` » est créer. Cette branche sert de backup pour rapidement revenir à la dernière version stable.

Pour revenir à un ancien commit, exécutez un « `git log` » pour voir l'identifiant d'un commit et un « `git reset IdDuCommit` » pour revenir à l'ancien commit.



Bien entendu, les `commits` sont journalièrement pousser vers un repository distant. Dans le cadre de ce projet, la repository initiale est https://github.com/alikadev/RoadTrafficSimulator. Ce repository est un repository personnelle sur lequel j'ai push les premiers `commits`. De plus, tous le code est envoyé vers https://gitlab.emf-infopro.ch/EMF-Informatique/tpi/tpi-2024-roadtrafficsimulation qui est le repository officiel.

Pour aller sur une branche, faites un « `git switch NomDeLaBranche` ». Si vous n'avez pas la branche en local, faites un « `git fetch --all` » suivi d'un « `git checkout NomDeLaBrancheDistante –detach` ».

2 Travaux préparatoires

2.1 Mise en place d'un projet JavaFX

Pour ce projet, NetBeans va être utilisé comme IDE. Vous pouvez installer NetBeans depuis le site web officiel.

Ensuite, pour créer un projet JavaFX, Allez dans « File > New Projet… » Et créer un projet « Java with Maven > FXML JavaFX Maven Archetype (Gluon) ». Maven va installer automatiquement JavaFX et mettre en place un projet de base. Vous pouvez ensuite lancer l'application et tout devrait fonctionner.

La raison pour laquelle nous allons utiliser Maven est car l'installation de JavaFX sera faite automatiquement. Additionnement, beaucoup plus de documentation est écrite sur les projets Maven JavaFX que Ant. Notez que l'architecture arm64 n'est pas supportée par Maven - JavaFX. JavaFX n'a pas un bon support de cette architecture et NetBeans causera de nombreuses difficultés durant la mise en place de l'application.

Vous pouvez aussi remarquer que Maven ajoute 1 fichier important : « module-info.java ». Ce fichier défini le module (l'application) et ses exports, imports, etc…

2.2 Création d'un projet JavaFX

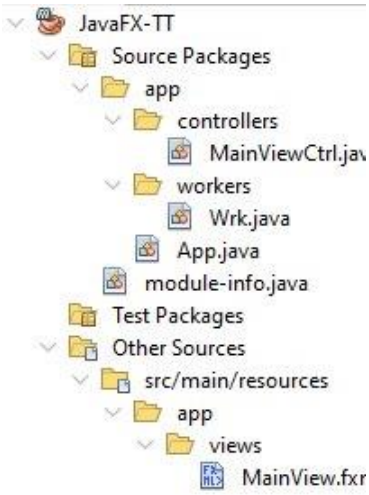
Le test technologique implémenté a pour but de tester la communication entre le Thread Java et le Thread JavaFX. Voici la liste des technologies qui vont être tester :

- Échange de données entre le `Thread JavaFX` et le `Main Thread`
- Expérimentation de `Platform.runLater`
- Expérimentation de `Binding`
- Expérimentation de `Task`

Le choix de sa technologie n'a pas été fait au hasard. La communication entre le « `backend` (`Worker`) » et le « `frontend` (`JavaFX`) » est une des parties les plus complexe de ce projet.

2.2.1 Structure MVC avec JavaFX

Le projet est nommé JavaFX-TT et sera une application `MVC (Model/Worker – View – Controller)` simple.



Les Views sont dans le `src/main/ressources/app/views` au format FXML. Leur location n'est pas aléatoire, « src/main/ressources » est le dosser depuis lequel les ressources de l'application sont copié avec Maven. Vu que les FXML ne sont pas compiler, ce sont des ressources.

Le contrôleur se trouve dans « `app/controller` » et le Worker se trouve dans « `app/workers` ».

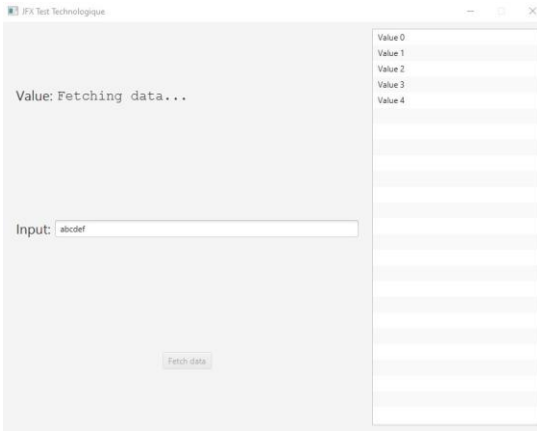
(Figure 1 : le projet sur NetBeans)

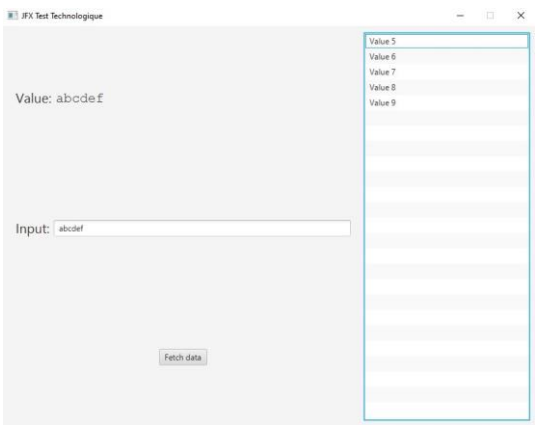
2.3 JavaFX-TT

2.3.1 Fonctionnement

Quand vous lancez l'application, vous allez voir un champ texte qui est lié à la « `value` » du label. Une fois que vous appuyez sur « `Fetch data` », le « `value` » affiche « `Fetching data...` » pendant 1 seconde et va charger des valeurs dans le tableau à droite.

Ce test n'est pas très utile mais permet de bien tester le fonctionnement des différentes technologies utile pour le projet





2.3.2 Points importants de l'implémentation

Je ne vais pas écrire tout le code de l'application ci-dessous mais uniquement les parties intéressantes.

```
/**
 * Update the list content
 * @param values The list of values to put in the list
 */
public void updateList(List<String> values) {
    // Use of Platform.runLater because it will be called by an external
thread.
    Platform.runLater(() -> listTask.getItems().setAll(values));
}
```

Lors de la mise à jour de la liste de l'application, l'utilisation du `Platform.runLater` permet à un second `thread` de modifier le contenu de la liste. Sans ce `Platform.runLater`, la mise à jour de la liste pourrai créer un `java.lang.IllegalStateException`. Cette erreur n'apparait pas à chaque clique mais arrivera au bout d'un moment.

```
/**
 * Bind the label to the input and enable the buttonRunTask.
 */
private void bindLabelToInputText() {
    label.textProperty().bind(textLabel.textProperty());
    buttonRunTask.setDisable(false);
}

/**
 * Unbind the label from the input and disable the buttonRunTask.
 */
private void unbindLabelToInputText() {
    label.textProperty().unbind();
    buttonRunTask.setDisable(true);
}
```

Ces 2 fonctions créer/retire le lien entre le texte du label « `value` » avec le champ texte (`textLabel`). Sans cette fonctionnalisée, un autre moyen serai d'utiliser un `callback` ce qui serai beaucoup moins efficient et dure à gérer.

```
// Create a task to set the new string value
Task<Void> fetchTask = new Task() {
    @Override
    protected Void call() throws Exception {
        // Simulate a long operation (1 second)
        wrk.fetchData();
        return null;
    }
};
```

```
// On task success, rebind the text
fetchTask.setOnSucceeded((WorkerStateEvent e) -> {
    bindLabelToInputText();
});

// Start the task
new Thread(fetchTask).start();
```

La création d'une tâche (`Task`) dans ce cas permet d'ajouter des `callbacks` pour l'exécution de la tâche. Ici, après que le `Worker` à récupérer les données, le texte est lié au champ de texte. Comme vous pouvez le voir, la tâche est lancée via la création et exécution d'un nouveau `thread`.

2.3.3 Récapitulatifs

- `Platform.runLater` permet d'interagir avec le `thread JavaFX`.
- Les `Bindings` permettent de lier les valeurs avec entre elle.
- Les `Task` permettent une gestion avancée des `Threads` et un meilleur contrôle de flux.

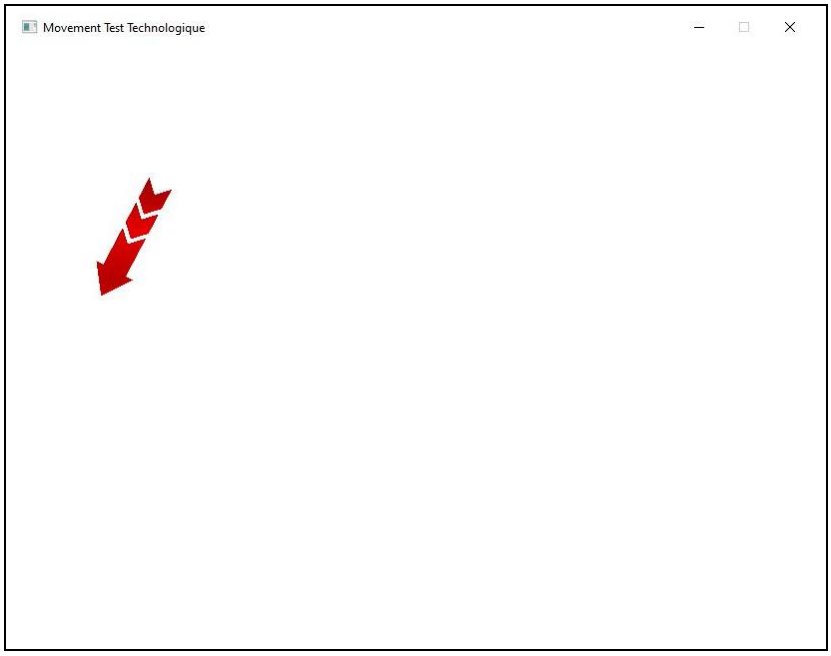
2.4 Moving-TT

Ce test technologique a pour but de tester le déplacement d'image sur l'écran. Ceci permet de mieux comprendre comme créer des animations et simulations sur le « `backend` » et les afficher sur le frontend.

2.4.1 Fonctionnement

Quand vous lancerez cette application, vous verrez une flèche tourner autour du centre de la fenêtre.

Cette application n'a aucune action intéressante mais permet de mieux comprendre comment JavaFX gère ses entités et comment les déplacer sur l'écran.



2.4.2 Points importants du code

```
/**
 * Set the arrow position.
 */
public void setArrowPosition(double x, double y) {
    arrow.setLayoutX(x);
    arrow.setLayoutY(y);
}

/**
 * Set the arrow rotation in degree.
 */
public void setArrowRotation(double angle) {
    arrow.setRotate(angle);
}
```

Ceci est le code permettant de déplacer et tourer une `ImageView` nommée `arrow`. La modification de ces valeurs ne demande pas de `Platform.runLater`.

```
// Update position
//  sin(x)        = SIN from -1 to 1
//  sin(x)+1      = SIN from  0 to 2
// (sin(x)+1)/2   = SIN from  0 to 1
// (sin(x)+1)/2*n = SIN from  0 to N
double angle = start * Math.PI * 2 / SEC_PER_ROT;
double xMult = ctrl.getLayerWidth() - ctrl.getArrowWidth();
double yMult = ctrl.getLayerHeight() - ctrl.getArrowHeight();
x = (Math.sin(angle) + 1) / 2 * xMult;
y = (Math.cos(angle) + 1) / 2 * yMult;
ctrl.setArrowPosition(x, y);
ctrl.setArrowRotation(-angle / Math.PI / 2 * 360);
```

Et voici comment le `Worker` modifie la position de la flèche. Ce n'est rien de plus qu'un peu de trigonométrie.

```
// Sleep to reach a FPS goal
double now = getTime();
long sleepMS = Math.round((1.0/FPS - (now - start)) * 1000);
if (sleepMS > 0) {
    try {
        Thread.sleep(sleepMS);
    } catch (InterruptedException e) {
        System.err.println("Unreachable");
    }
}
```

Et voici comment le nombre d'image par second est gérer.

2.4.3 Récapitulatifs

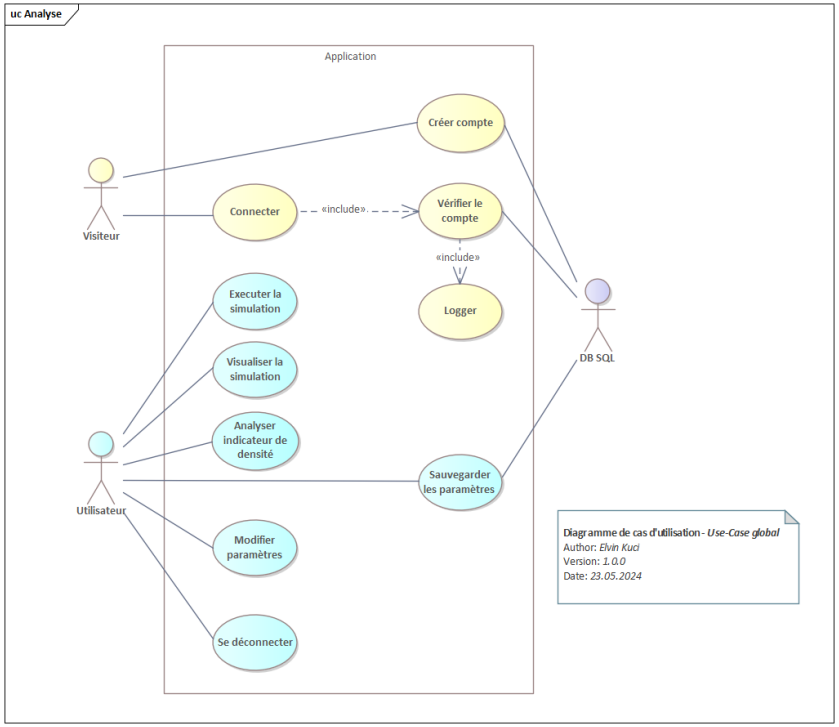
- `ImageView.setLayoutX/Y(double)` : Modifie la position d'une image
- `ImageView.setRotate(double)` : Modifie la direction de l'image en degré

3 Analyse

L'analyse du projet a été réalisé le premier jour de projet. Cette phase permet de visualiser les charges de travail ainsi que la direction générale du projet.

3.1 Analyse de l'état désiré

L'objectif du produit fini sera de simuler et visualiser le trafic d'un carrefour à 4 voies en fonctions des différents paramètres éditables. L'application final aura une interface intuitive permettant à l'utilisateur de comprendre visuellement en temps réel les effets de différents paramètres. Il sera ainsi en mesure d'optimiser la séquence des feux de signalisation.



Comme vous pouvez le voir sur la figure ci-présente, le diagramme de cas d'utilisation :

Un « visiteur » peut se connecter et créer un compte.

Un « utilisateur » connecté peut exécuter et visualiser la simulation, modifier et sauvegarder des jeux de paramètres et se déconnecter.

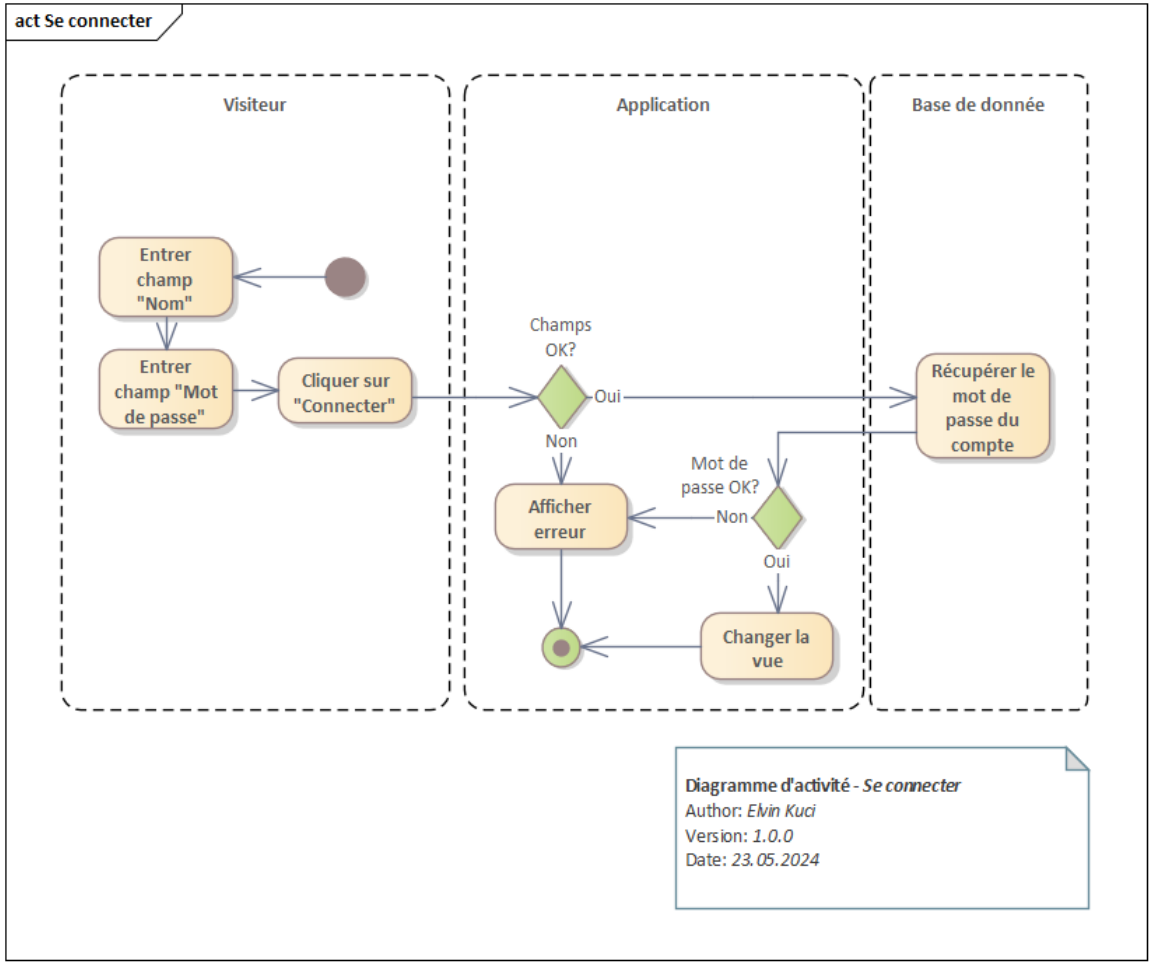
Et la base donnée ne travaille qu'avec le compte et les paramètres.

3.1.1 Fonctionnement de la connexion

Quand un visiteur arrive sur l'application, il sera invité à se connecter. Pour se faire, il va entrer son nom de compte, son mot de passe et appuyer sur « Se Connecter ».

Une fois que ceci est fait, l'application dois dévifier que les champs sont bel et bien entré.

Ensuite, l'application va récupérer le compte avec le même nom dans la base de donnée et vérifier si le mot de passe est bon. Si tout se passe bien, l'appliciation changera se vue pour entrer dans le mode de simulation.

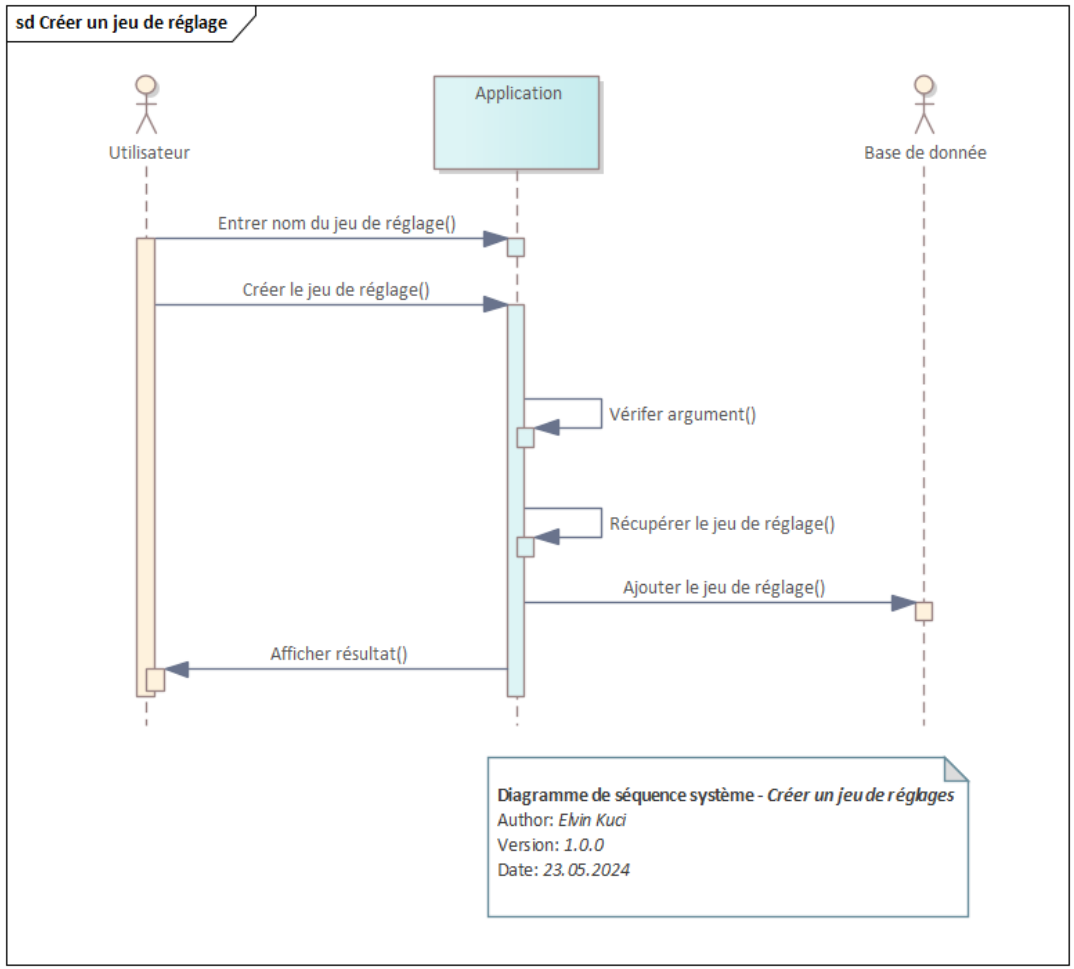


3.1.2 Création d'un jeu de réglage

Lors de la création d'un jeu de réglage, l'utilisateur devra entrer le nom du jeu de réglage et appuyer sur « Créer le jeu de réglage ».

À ce moment, l'application vérifiera l'argument et récupèrera les réglages en cours.

Une fois cela fait, le jeu de réglage sera envoyé à la base de données et affichera le résultat de l'action.



3.1.3 Sauvegarde de paramètres - détails

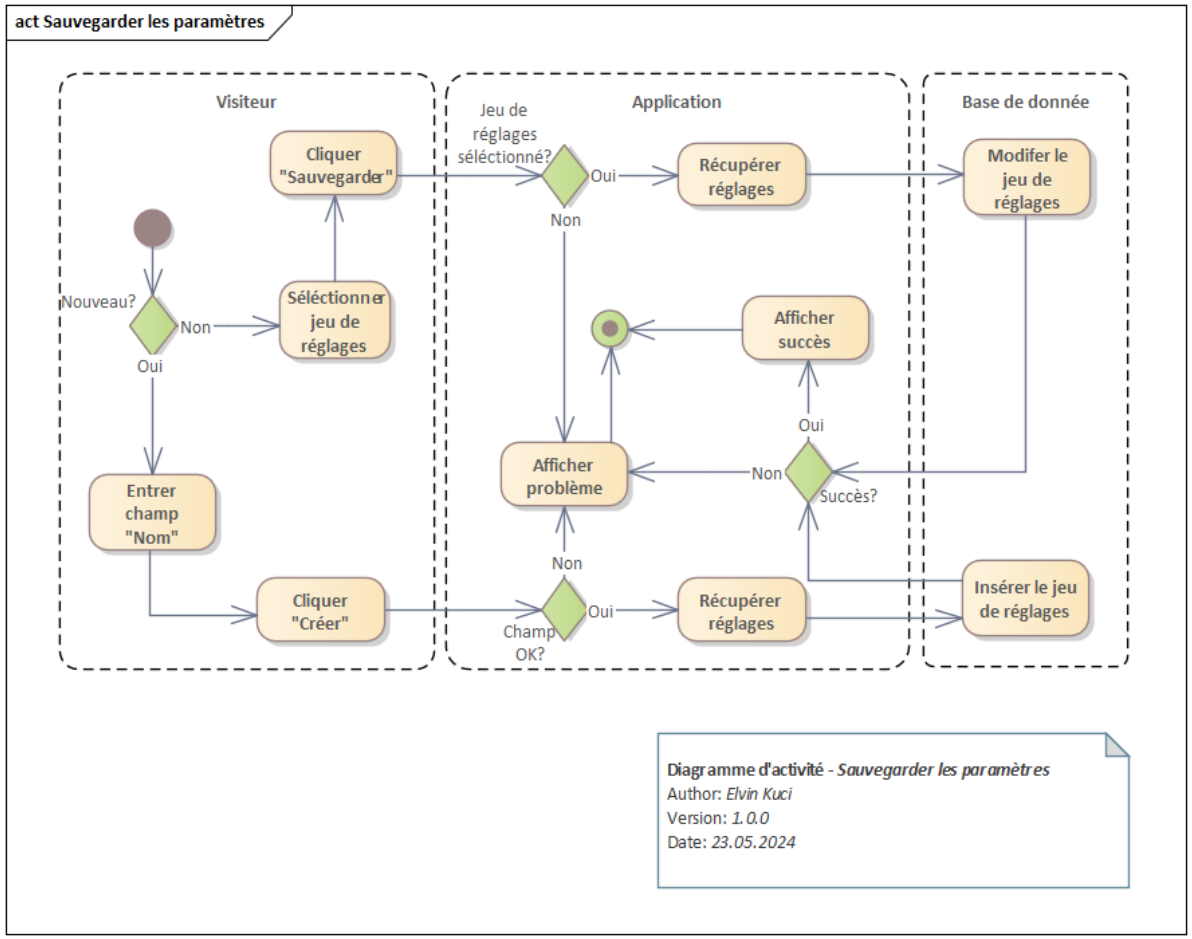
L'utilisateur n'est nullement obligé de créer un nouveau jeu de réglage à chaque modification. Il peut aussi sauvegarder de nouveaux réglages sur un jeu de réglage préexistant.

Le diagramme d'activité ci-présent démontre les deux cas et leurs fonctionnements.

Si l'utilisateur souhaite sauvegarder un jeu de réglages, il doit sélectionner le jeu dans la liste et cliquer sur sauvegarder.

Cependant, si l'utilisateur souhaite créer un nouveau jeu de réglages, il devra entrer le nom du jeu de réglage et cliquer sur créer.

Bien qu'il y ait 2 chemins différents dans ce diagramme, nous pouvons remarquer qu'ils ont un fonctionnement assez proche. Il sera donc probablement possible d'utiliser les mêmes fonctionnalités dans le code pour ces deux chemins !



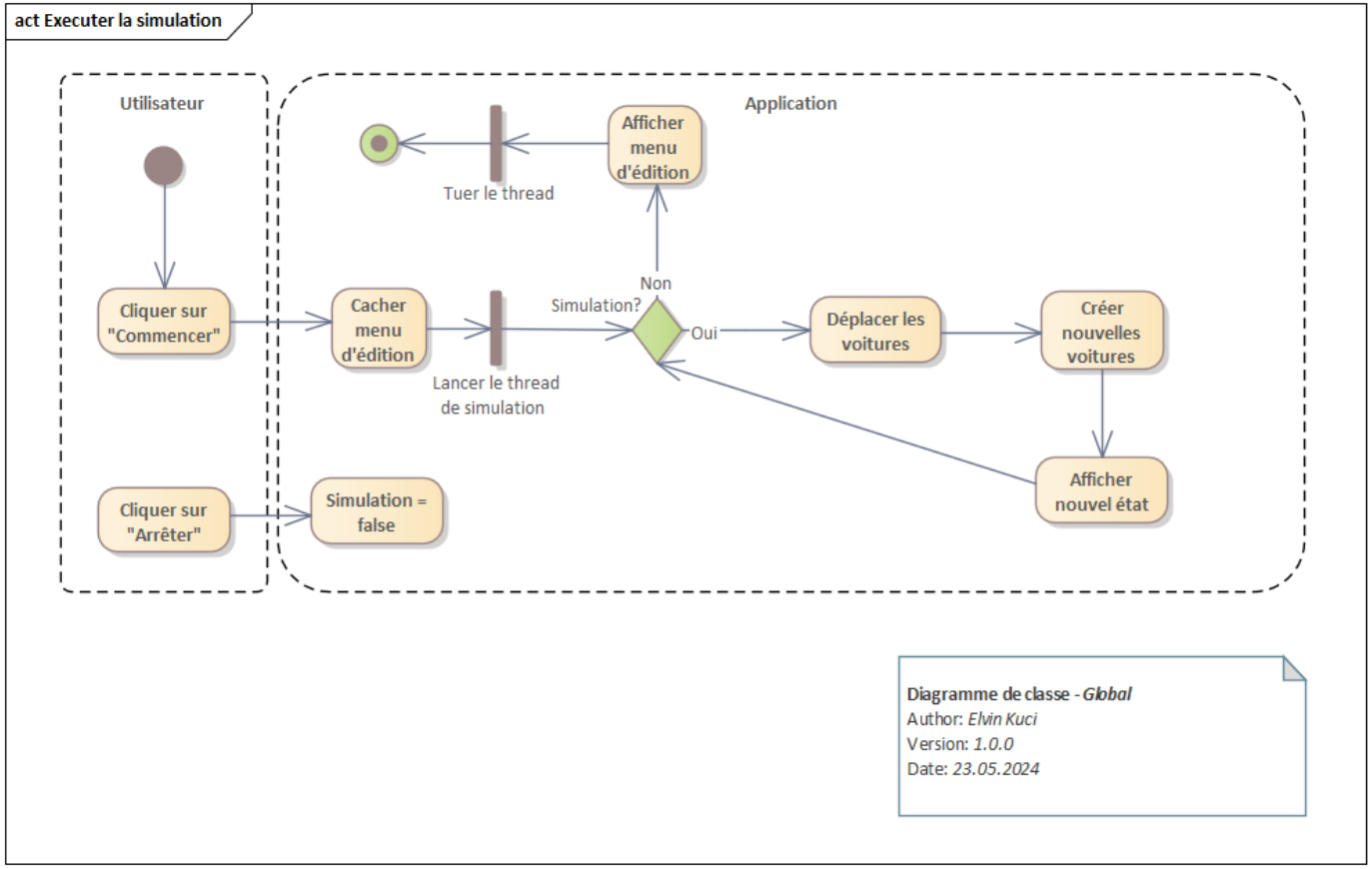
3.1.4 Exécution de la simulation

Pour débuter la simulation, l'utilisateur n'aura qu'à appuyer sur « Commencer ».

Ensuite, l'application cachera le menu d'édition et démarrera le thread de la simulation.

Celui-ci s'occupera de déplacer et créer les véhicules ainsi que d'afficher le nouvel état et vérifier s'il doit arrêter.

Quand l'utilisateur appuyer sur « Arrêter », la simulation s'arrêtera.



3.2 Variantes

Lors de la conception des structures de données et des flux de l'application, 2 méthodes de gestion des données intéressantes me sont venues en tête. La première est plus simple mais moins facile à généraliser et la deuxième est plus complexe mais simple à généraliser.

3.2.1 Variante 1

```
enum RoadType =
    STRAIGHT_ROAD
    CROSSROAD

type Road:
    getType(): RoadType

class StraightRoad impl Road;
    getType(): RoadType = return STRAIGHT_ROAD
    // ...

class Crossoad impl Road:
    getType(): RoadType = return CROSSROAD
    // ...

circuit: List<Road> = { ... }

for rd in circuit:
    if rd.getType() == CROSSROAD:
        rd = (Crossroad) rd;
        // Show crossroad
    elif rd.getType() == STRAIGHT_ROAD:
        rd = (StraightRoad) rd;
        // Show straight_road
```

`RoadType` est une constante permettant de différencier entre les différents types de routes.

`Road` est le type parent permettant de définir que les routes doivent implémenter `getType`.

`StraightRoad` et `Crossroad` sont des exemples de routes. Ils ont des propriétés différentes.

Donc dès qu'on utilise le `circuit`, nous devons récupérer le type de route et la ré-interprété en le bon type de route.

Cette solution est très simple à implémenter mais peut devenir très dure à agrandir. Dès le moment où nous voulons ajouter un nouveau type de route, nous devons ajouter une énumération, une classe et modifier tous les `if` et `switch-cases`.

3.2.2 Variante 2

```
type Road:
    getProps(): List<Pair<Label, Props>>

class StraightRoad impl Road:
    getProps(): List<Pair<Label, Props>> =
        return // Road props here...

class Crossroad impl Road:
    getProps(): List<Pair<Label, Props>> =
        return // Crossroad props here ...

circuit: List<Road> = { ... }

for rd in circuit:
    (label, property) = rd.getProps()

    JFX_HBox hbox = new JFX_HBox()
    hbox.add(new JFX_Label(label))
    hbox.add(new JFX_TextField().bindValue(property))

    editMenu.add(hbox)
```

`Road` est le type parent permettant de définir qu'une route implémentera `getProps`. Cette méthode est utilisée pour récupérer les propriétés d'une route.

`StraightRoad` and `Crossroad` sont des types de routes différent.

Dès le moment où nous devons afficher les champs permettant d'éditer une route, nous avons juste à récupérer les propriétés et les afficher. Le fait que ce soit une `Crossroad`, une `StraightRoad` ou un chemin de terre ne change rien, le code devient totalement réutilisable

3.2.3 Choix de variante

Pour décider quelle variante utiliser, nous pouvons dresser une liste des avantages et inconvenants avec une pondération et faire la somme des points de chaque variante. Plus la somme est haute, moins l'idée est intéressante dans ce cadre. Pour cette matrice, j'ai choisi 3 points majeurs :

- La difficulté initiale : Difficulté de la première implémentation

Ce n'est pas vraiment un problème au long terme, mais ça risque de rendre la première implémentation plus longue et fastidieuse. La pondération est donc de 1.

- Difficulté future : Difficulté d'agrandissement de la solution dans le futur.

C'est très important dans le cadre de ce mandat de faire une application qui sois extensible et améliorable. La pondération est donc de 3.

- Risque de bug : Risque de commettre une erreur et difficulté de debug

Le risque de bug peut causer des problèmes ennuyants et prendre du temps mais ce n'est en aucun cas autant important que l'extensibilité de l'application. Une pondération de 2 me semble honnête.

	Pondération	Variante 1	Variante 2
Difficulté initiale	x1 (au début)	1 (simple)	3 (complexe)
Difficulté future	x3 (extensible)	4 (immuable)	2 (constante – se facilite)
Risque de bug	x2 (risque + debug)	2 (normal)	1 (faible ou simple débugger)
Résultats		1 + 12 + 4 = **17**	3 + 6 + 2 = **11**

Malgré les quelques mots clé que j'ai mis dans la matrice ci-dessus, je pense que c'est une bonne idée de repasser sur chaque point ci-dessous. Mise à part les détails, nous pouvons voire que la deuxième variante est beaucoup plus intéressante dans ce cas !

3.2.3.1 Difficulté initiale

La première variante est probablement la plus simple possible. Une énumération et un « switch-case » permet d'implémenter cette variante. Une difficulté de 1 me parait bon.

La deuxième variante est plus complexe. L'abstraction du type de route, et même de ce qu'est une route en premier lieu rendra la première implémentation complexe. Au lieu de remplacer des bindings sur des onglets préfabriquer, il faudra générer les liens entre les objets manuellement. Une difficulté de 3 me semble correcte !

3.2.3.2 Difficulté future

La première option à un gros problème sur ce point. À chaque fois que nous voudrons ajouter un type de route, de devrons passer à travers tous le code et ajouter une condition pour ce nouveau type de route. C'est très facile d'oublier une condition et de causer une erreur qui peut être très difficile à trouver.

La deuxième option a un gros avantage sur ce point : Vu qu'une route est un concept abstrait de base, l'ajout d'un type de route ne consiste qu'à implémenter la route et la rajouter dans la simulation. De plus, au fil du développement, de plus en plus de type de route seront ajouter dans le code et l'ajout de nouvelles routes ne deviendra qu'un copier-coller de paterne qui ont déjà été créer auparavant. 2 est la difficulté que j'ai choisie.

3.2.3.3 Risque de bug

La première option est relativement standard. Il y a un risque de bug typique pour une application de cette nature et les résoudre ne sera pas trop complexe non plus. De simple solution sont simple à debugger. Le risque est de 2.

La deuxième option est encore plus simple à debugger. S'il manque un champ pour une route, c'est qu'il n'est pas retourné ou qu'il y a un problème d'affichage. C'est vraiment aussi simple que ça, 1 me semble correcte !

3.3 Sécurité de l'information et protection des données

Cette application subit un problème que toutes les applications avec la même fonctionnalisée ont : ils sauvegardent les identifiants et mots de passe de leurs utilisateurs.

Une personne malicieuse pourrait réussir à s'introduire dans la base de données et récupérer les mots de passe des utilisateurs. Le véritable risque est que certains utilisateurs vont utiliser les mêmes identifiants et mots de passe pour d'autres comptes.

Pour régler ce problème, nous allons alors hasher les mots de passes pour qu'ils ne puissent pas être retrouvé. Le hashage est un procédé informatique qui consiste à récupérer un texte en entré et à un sortir une emprunte unique. Le point intéressant est que la fonction de hash va toujours générer la même emprunte et le texte original ne peut pas être retrouvé avec l'emprunte. Nous pourrons alors comparer les empruntes sans divulguer le mot de passe de l'utilisateur !

4 Conception

4.1 Exigences du système

Les tronçons ainsi que le carrefour possèderont des identifiants unique et un système de connexion numérotées. La construction logique et visuelle se fera par le développeur mais dois être pensée générique.

Additionnellement, une gestion du flux et de la densité des véhicules par minutes pourra être affecté au tronçon. Le véhicule sera créé avec tous ses paramètres associés ainsi qu'une légère variabilité dans une plage de tolérance permettant de mieux simuler une route. Les véhicules réagiront aux comportements du véhicule suivant en essayant de maintenir une distance de sécurité.

De plus, l'utilisateur doit pouvoir se logger à sa session sur différents PC où l'application est installé et retrouver ses jeux de paramètres. Par conséquent, les données seront stockées sur une base de données hébergée sur emf-infopro-tpi.ch.

Finalement, l'interface graphique représentera de manière simple les tronçons de route et le carrefour avec les feux. L'accès aux paramètres devra être pensé dans une approche ergonomique et les éléments dynamique seront les véhicules et la couleur des feux de signalisation.

4.2 Architecture du système

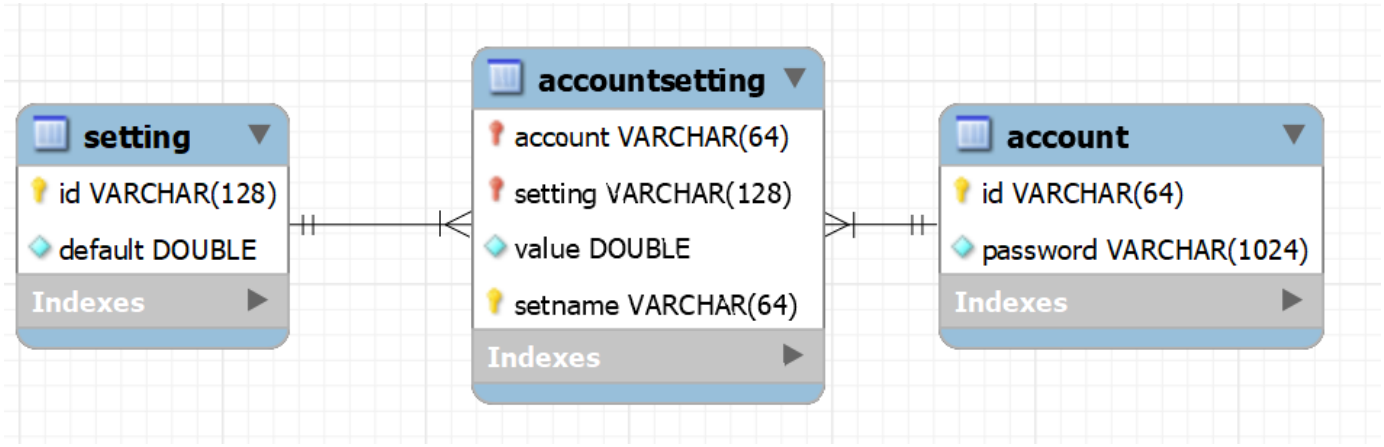
La structure du projet est relativement simple. L'application Java communique directement avec la base de données SQL distante.

4.2.1 L'application cliente

L'application cliente est développée en Java avec l'utilisation de la librairie graphique JavaFX. JavaFX est une bibliothèque de code développé par Oracle permettant de créer des interface utilisateur avec Java.

4.3 La base de données

Voici la représentation de la base de données :



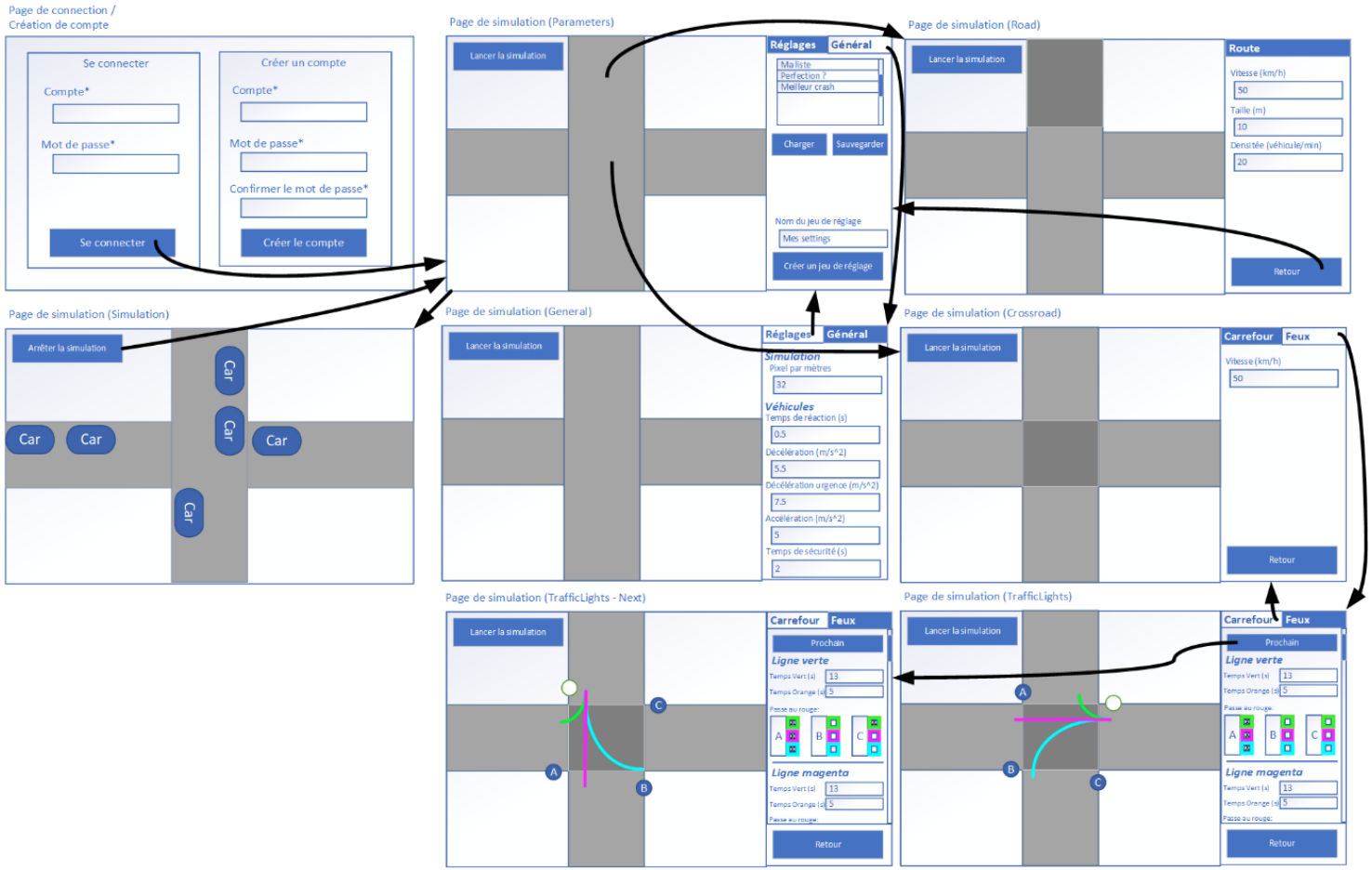
La table « setting » contient les paramètres existants. Chaque paramètre est défini par un identifiant (id) sous la forme d'une chaine de caractère de maximum 128 caractères.

La table « account » contient l'identifiant de chaque utilisateur et le mot de passe. Le champ « password » peut contenir jusqu'à 1024 caractères pour supporter n'importe quelle méthode de hashage.

La table « accountsetting » est une table de relation permettant de définir une valeur pour chaque réglage. Le « setname » est le nom du jeu de réglages. Bien que ça ne soit pas la technique la plus économe en place, ceci permet de facilement lire la base de données et de facilement SELECT tous les réglages d'un jeu de réglage…

4.4 Maquettes

La maquette ci-dessous est la maquette de l'application. Elle permet aussi de représenter le flux de l'application à travers les différentes cliques.



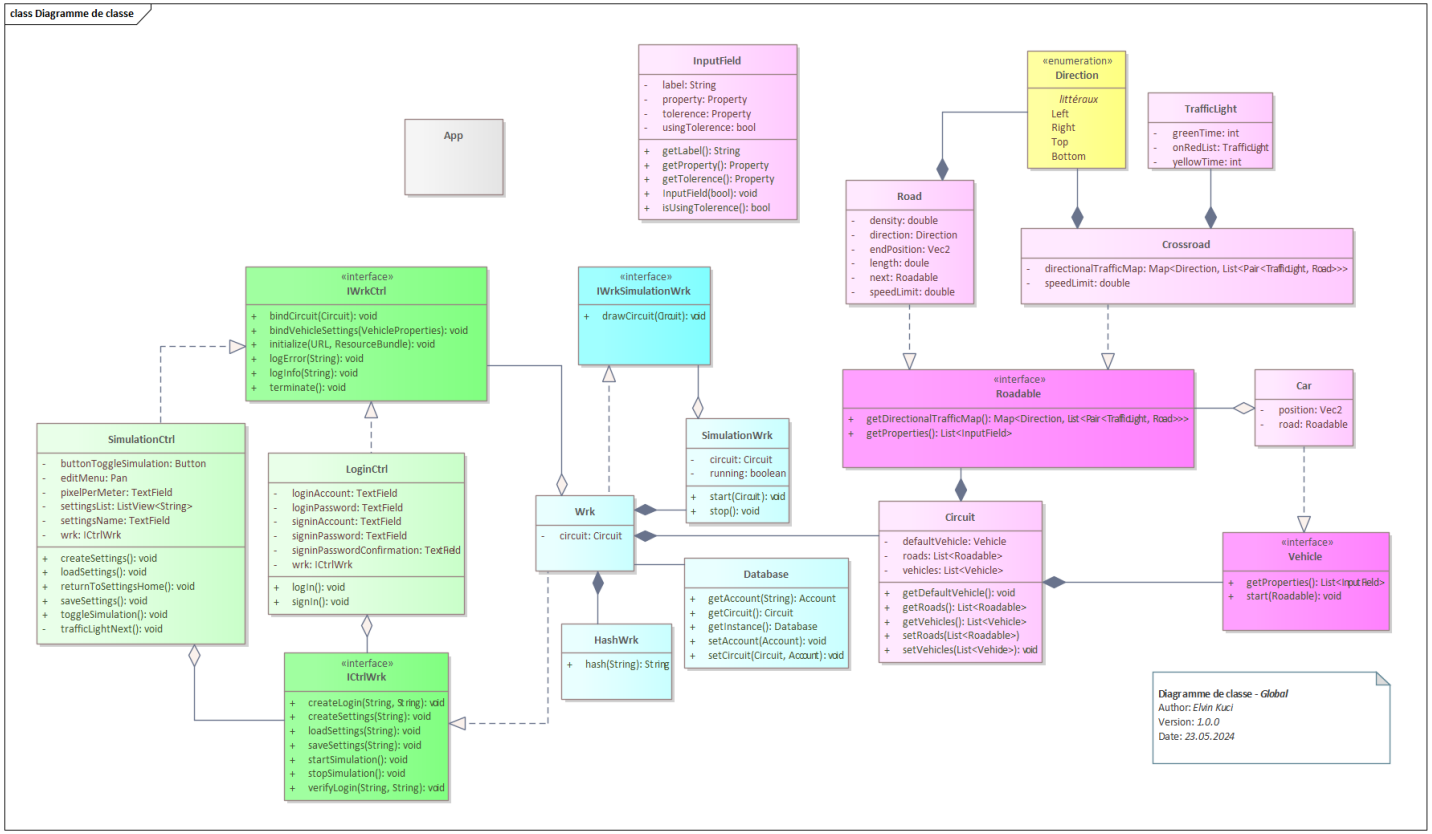
Tout commence en haut à gauche avec la page de connexion qui permet de se connecter.

Ensuite, la page de simulation permet d'éditer les paramètres généraux, gérer les jeux de réglages. Quand vous cliquez sur une route ou un carrefour, vous verrez ses paramètres que vous pourrez éditer.

Les carrefours ont aussi un onglet « Feux » permettant de controller la signalisation. Vous pourrez donc controller les feux de chaque direction ainsi que le temps au vert, orange et quels feux doivent être mis au vert quand celui-ci passe au rouge.

4.5 Diagramme de classe

Le diagramme de classe ci-dessous est celui qui a été créer durant la conception. Allez au point 6 pour retrouver le dernier diagramme de classe.



L'application est composée de contrôleurs (en vert) implémentant la même interface pour que le Worker (en bleu) puisse communiquer avec chaque contrôleur. De plus, les beans (en magenta) représente les différentes structures qui seront utilisé par l'application.

Le SimulationCtrl est le contrôleur de la page de simulation et le LoginCtrl est contrôleur de la page de login.

Les vues ne sont pas présentées sur ce schéma car ce sont des FXML et non des classes.

4.6 Diagrammes de séquence d'interaction

4.6.1 Se connecter

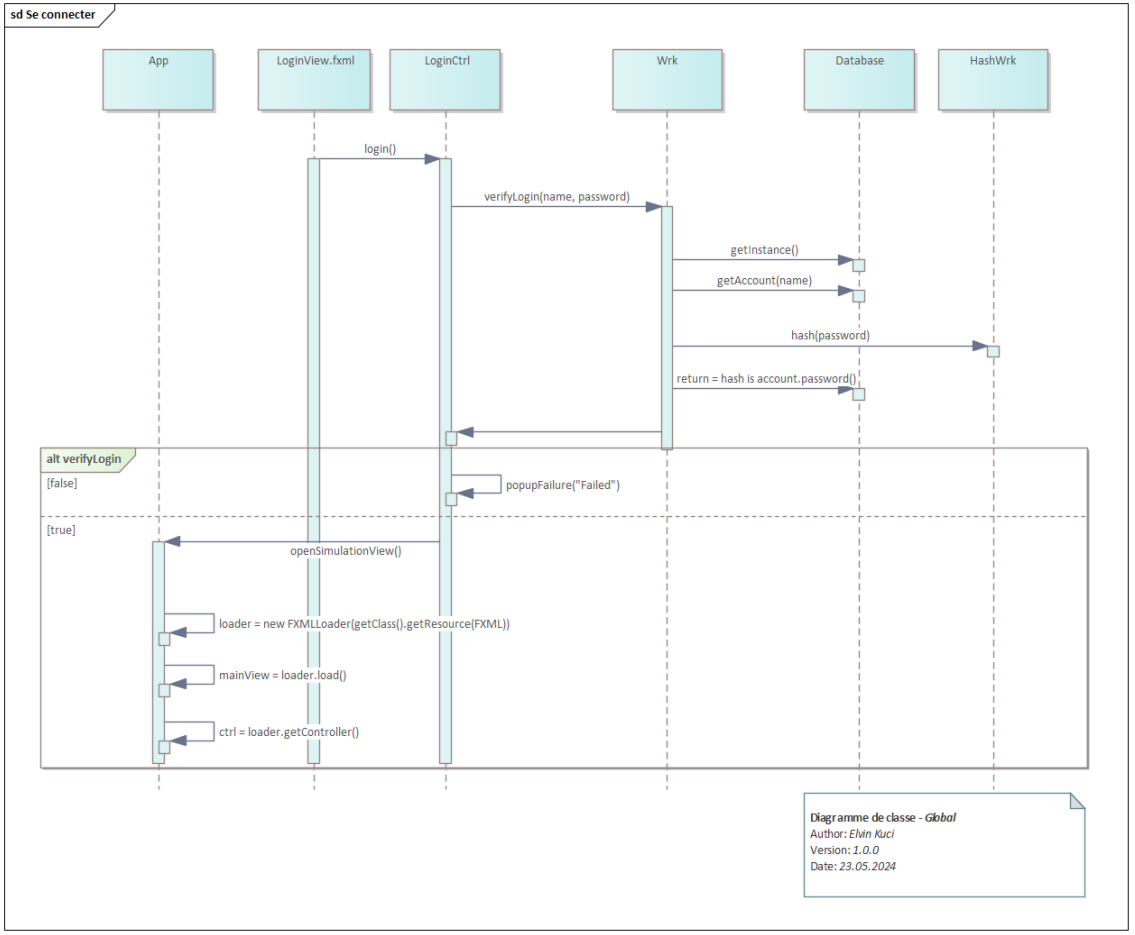
Quand l'utilisateur clique sur « se connecter », la requête de « login » va se lancer sur le LoginCtrl.

Ensuite, les identifiants sont envoyés vers le Wrk pour les vérifier.

Le Wrk va donc hasher le mot de passe et va le comparer avec celui de la base donnée.

Si la vérification n'est pas réussie, un message d'erreur est affiché.

Cependant, si la vérification est réussie, l'application s'occupera de charger la nouvelle vue.



4.6.2 Sauvegarder les paramètres

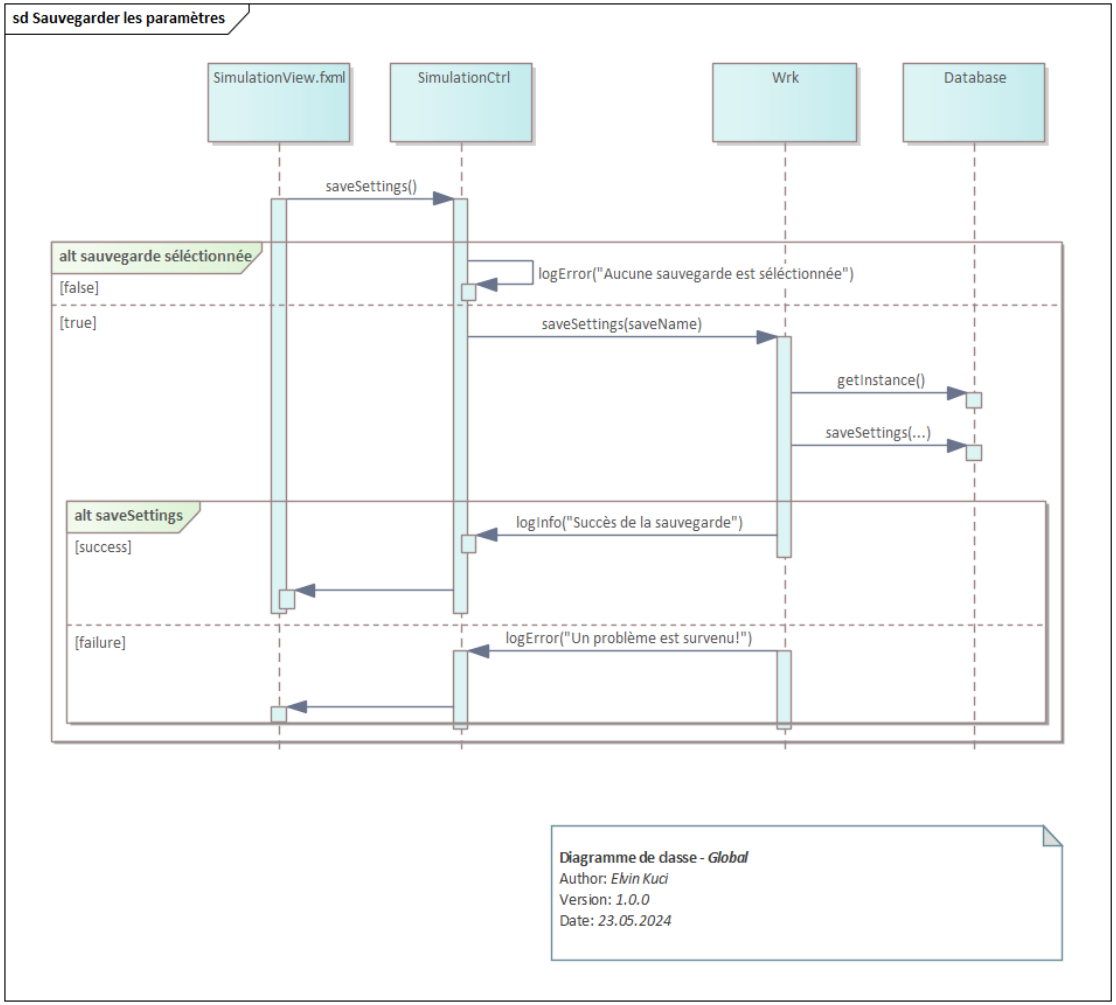
Lors de la sauvegarde de paramètres, la requête sera transmise au SimulationCtrl.

Si aucune sauvegarde n'est sélectionnée, une erreur devra être afficher.

Cependant, si une sauvegarde est sélectionnée, la requête sera transmise au Wrk qui sauvegardera les paramètres.

Si une erreur survient durant cette partie, une erreur sera affichée.

Autrement, le succès sera affiché.

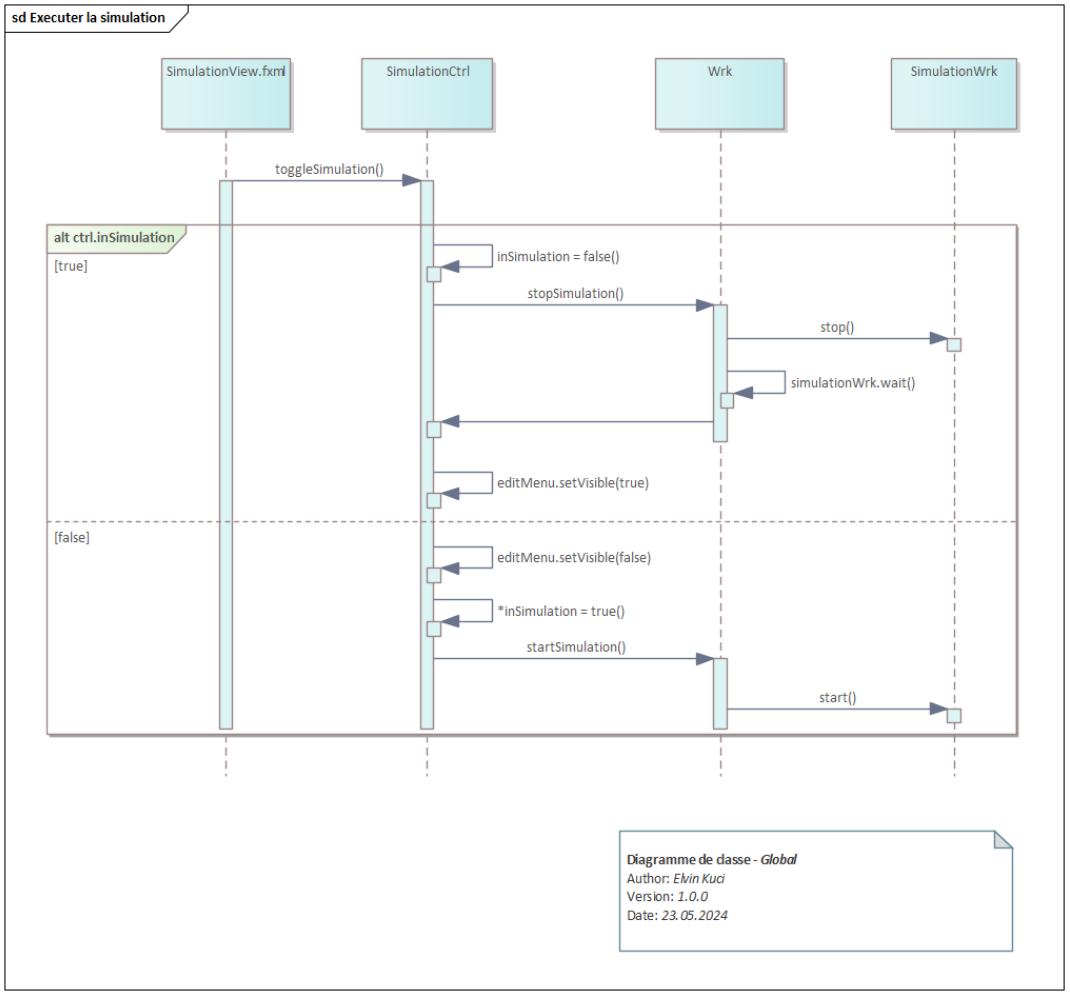


4.6.3 Exécuter la simulation

Quand utilisateur appuie sur le bouton « Lancer / Arrêter la simulation », la requête est transmise au SimulationWrk.

Si la simulation était en cours, une requête est transmise au Wrk. Celui-ci va arrêter le SimulationWrk et attendre que la simulation se sois bel et bien arrêter.

Cependant, si la simulation n'était pas en cours, la simulation est démarrée et le Wrk demandera au SimulationWrk de démarrer la simulation.



4.7 Concept de tests

Les tests qui seront réalisés sont des tests dit « Blackbox ». Ceci signifie que ce sont des tests que nous pouvons effectuer sans connaitre la structure interne de la solution / de l'application.

Page	Objet	Description du test	Résultat attendu
Login	Créer un compte	Créer un compte qui n'existe pas dans la base de données.	Le compte sera créé. La page change sur « Simulation ».
		Créer un compte qui existe déjà.	Le compte ne sera pas créé. Afficher une erreur.
		Créer un compte avec des champs vide.	Afficher une erreur.
		Créer un compte avec le champ « confirmation de mot de passe » qui est diffèrent du « mot de passe »	Afficher une erreur.
	Se logger	Se logger à un compte qui n'existe pas	Afficher une erreur.
		Se logger à un compte qui existe avec le mauvais mot de passe	Afficher une erreur.
		Se logger à un compte qui existe avec le bon mot de passe	La connection va fonctionner. La page change sur « Simulation ».
Simulation	Charger un jeu de réglages	Charger un jeu de réglages sans sélectionner un jeu	Afficher une erreur.
		Charger un jeu de réglages	Charger le jeu et afficher le succès
	Sauvegarder un jeu de réglages	Sauvegarder un jeu de réglages sans sélectionner un jeu	Afficher une erreur.
		Sauvegarder un jeu de réglages	Sauvegarder dans la base donnée et afficher le succès
	Créer un jeu de réglages	Créer un jeu de réglages sans nom	Afficher une erreur.
		Créer un jeu de réglages avec un nom qui existe déjà	Afficher une erreur.
		Créer un jeu de réglages avec un nom qui existe déjà mais qui a été ajouté après le lancement de l'application. Le jeu est donc dans la base de données mais pas la liste.	Afficher une erreur.
		Créer un nouveau jeu de réglages	Créer le jeu de réglages et afficher le succès
	Simulation	Lancer la simulation avec les réglages de départ	Le menu d'édition se ferme et la simulation fonctionne.
		Quitter la simulation.	La simulation s'arrête et le menu d'édition s'ouvre.
		Une fois que la simulation c'est stabilitée, vérifiez les statistiques sur la simulation.	La densité de la route équivaut au « In » et est le
	Réglage général	Changer le « Pixels par mètre » à une valeur inférieur à 0.	Afficher une erreur.

	Pixels par mètre	Changer le « Pixels par mètre » à une valeur entre 1 et 100.	Change la taille et fonctionne normalement.
	Réglage général Facteur de vitesse	Modifier le « Facteur de vitesse » à une valeur inférieur à 0.	Afficher une erreur.
		Modifier le « Facteur de vitesse » à une valeur entre 0.2 et 10. Une valeur plus grande pourrait causer un problème dans tous les cas.	Changer le facteur de vitesse.
	Réglage véhicule Décélération	Modifier cette valeur.	La valeur cause un changement lors des décélérations standard.
	Réglage véhicule Décélération %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.
		Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.
		Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.
	Réglage véhicule Décélération d'urgence	Modifier cette valeur.	La valeur cause un changement lors des décélérations d'urgence.
	Réglage véhicule Décélération d'urgence %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.
		Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.
		Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.
	Réglage véhicule Accélération	Modifier cette valeur.	La valeur cause un changement lors des décélérations d'urgence.
	Réglage véhicule Accélération %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.
		Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.
		Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.
	Réglage véhicule Temps de sécurité	Modifier cette valeur.	La valeur cause un changement de distance entre les véhicules
	Réglage véhicule Temps de sécurité %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.
		Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.
		Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.

	Réglage véhicule Temps de réaction	Modifier cette valeur.	La valeur cause un changement de temps de réaction
	Réglage véhicule Temps de réaction %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.
		Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.
		Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.
	Cliquer sur une route	Nous cliquons sur un tronçon de route standard	Le menu de la route s'ouvre.
	Cliquer sur un carrefour	Nous cliquons sur un carrefour.	Le menu de carrefour et le menu de feux de signalisation d'ouvre
	Réglage route Limitation de vitesse	Modifier cette valeur.	La valeur cause un changement de comportement des véhicules.
	Réglage route Limitation de vitesse %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.
		Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.
		Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.
	Réglage route Taille	Modifier cette valeur.	La valeur cause un changement visuel ainsi qu'un changement de comportement des véhicules.
	Réglage route Densité	Modifier cette valeur.	La valeur cause un changement de la vitesse d'apparition des véhicules
	Réglage route Densité %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.
		Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.
		Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.
	Réglage carrefour Limitation de vitesse %	Modifier cette valeur.	La valeur cause un changement de comportement des véhicules.
	Réglage carrefour Limitation de vitesse %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.
		Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.
		Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.

	Réglages feux Temps vert	Modifie cette valeur avec une valeur inférieur à 0. À tester sur les 4 différents feux.	Afficher une erreur.
		Modifie cette valeur avec une valeur différente de 0. À tester sur les 4 différents feux.	La valeur cause un changement de temps de vert à feux tester.
	Réglages feux Temps orange	Modifie cette valeur avec une valeur inférieur à 0. À tester sur les 4 différents feux.	Afficher une erreur.
		Modifie cette valeur avec une valeur différente de 0. À tester sur les 4 différents feux.	A valeur cause un changement de temps d'orange à feux tester.
	Réglages feux Passage au rouge	Modifie les « croix ». À tester pour toutes les lignes et tous les feux.	Lors d'un passage au rouge, les lignes sélectionnées passent au vert
	Réglages feux Prochain	Appuyer sur le bouton « Prochain »	Un changement de feu en cours s'opère dans une direction constante (horaire ou anti-horaire, pas de hasard !)

5 Réalisation

5.1 Type de projet

Ce projet est un projet Java Maven développé avec l'IDE Intellij IDEA Community Edition. Il utilise JavaFX pour l'interface utilisateur ainsi qu'afficher la simulation. Le driver MySQL est aussi utilisé pour les interactions avec la base de données.

5.2 Dépendances

Le projet repose sur plusieurs dépendances essentielles :

5.2.1 JavaFX

JavaFX est la bibliothèque utilisée pour gérer l'interface graphique de l'application. Conçue par Oracle, elle est fortement inspirée de son prédécesseur, Swing, et offre des fonctionnalités avancées pour le rendu et la gestion des fenêtres.

5.2.2 MySQL Driver

Le MySQL Driver est utilisé pour établir une connexion avec la base de données MySQL, permettant ainsi à l'application d'effectuer des opérations de lecture et d'écriture sur la base de données SQL.

5.3 Outils

L'IDE utilisé pour implémenter et exécuter ce projet est « IntelliJ IDEA 2024.1.2 (Community Edition) ».

La SDK utilisé est la « Eclipse Temurin 17.0.7 » avec un Language Level à 22

5.4 Structure du projet

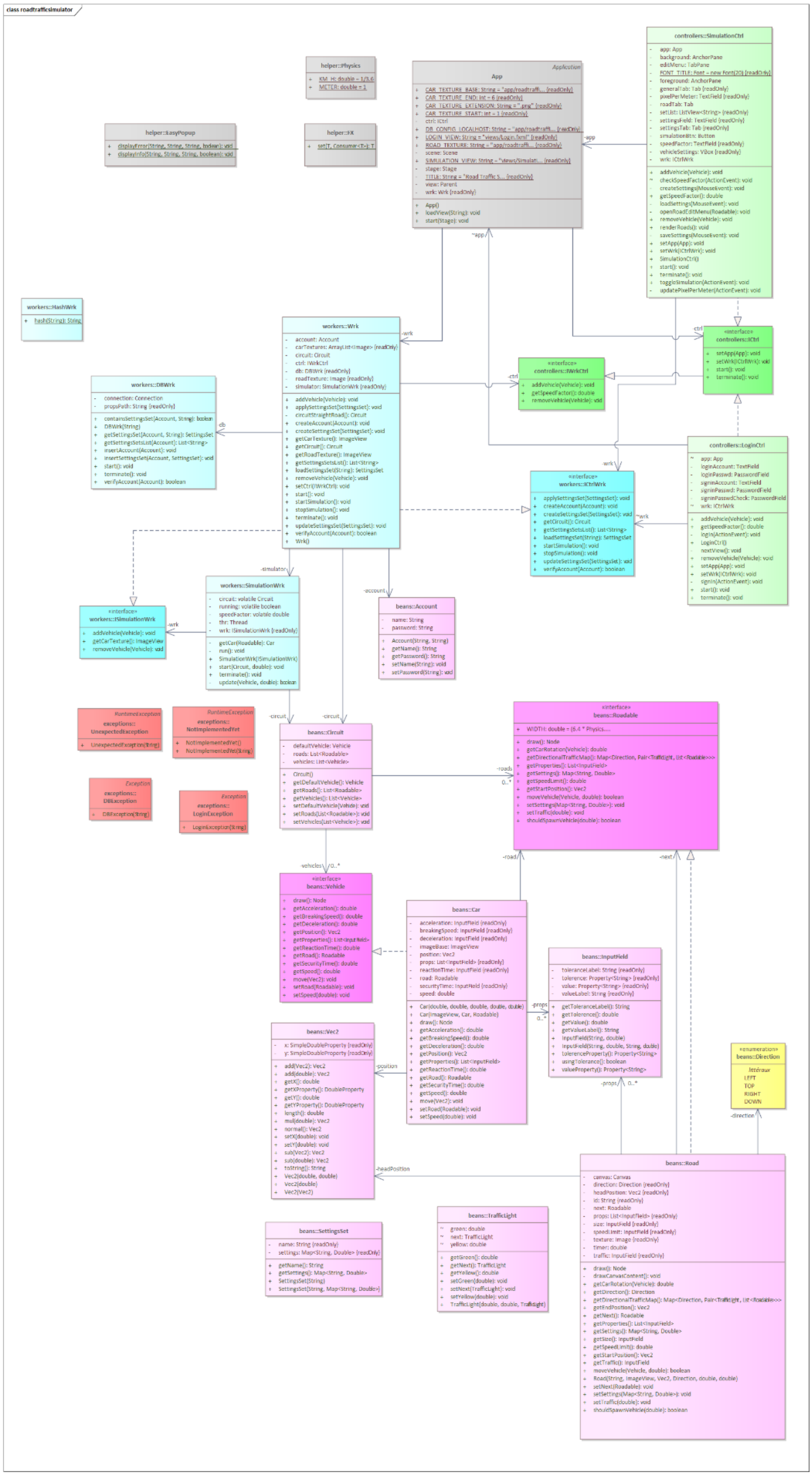
5.4.1 MVC2 avec FXML

La structure MVC2 (Model-View-Controller) est une architecture de conception logicielle utilisée pour séparer les préoccupations dans une application, facilitant ainsi la maintenance et l'évolution du code.

5.4.2 Diagramme de classe

Bien que ce diagramme soit plus complet que celui présent dans la phase de conception, il reste majoritairement similaire.

Une des différences majeures est le fait que cette version ne contient pas de classe Crossroad (Carrefour). Ceci est dû à un manque de temps pour l'implémentation. Certaines fonctionnalités ne sont pas encore implémentées…



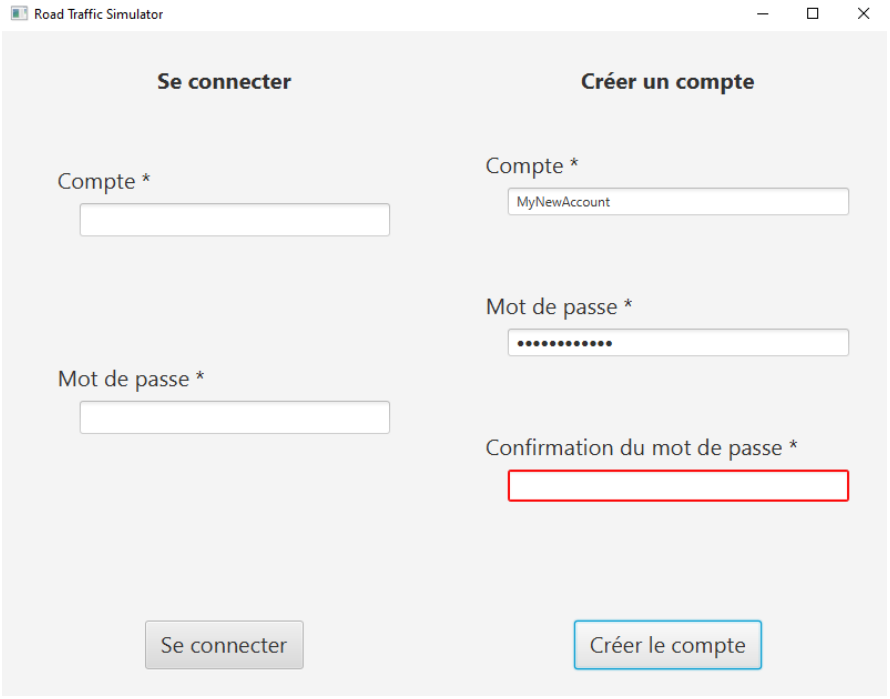
5.5 Interface utilisateur

Page de connexion

Il y a deux options : se connecter et créer un compte.

En cas d'oubli d'un champ, il deviendra rouge comme ci-présent.

En cas d'erreur de mot de passe ou de compte inexistant, un pop-up d'erreur s'affichera.

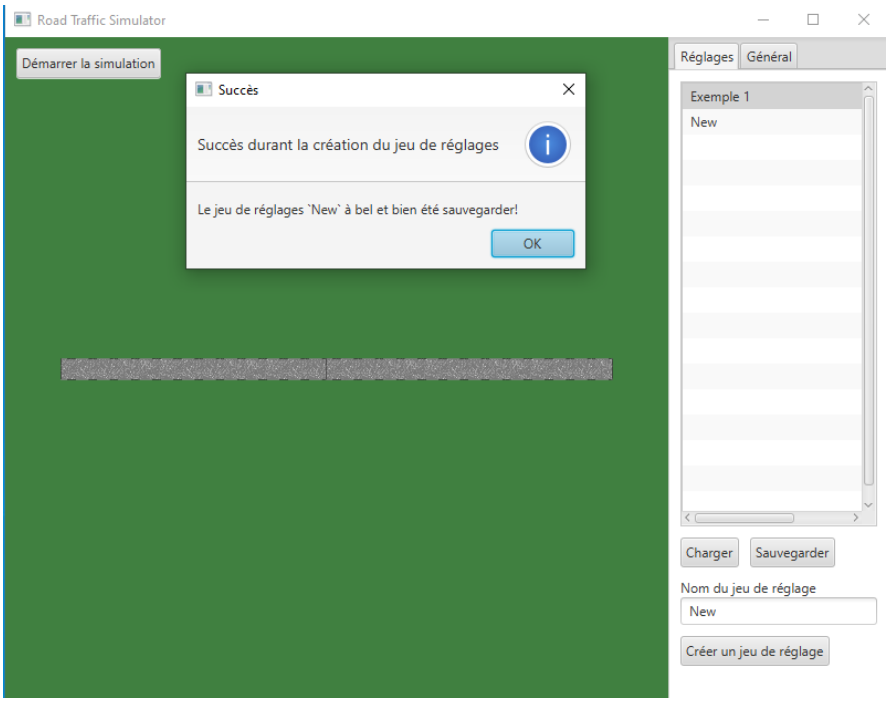


Gestion des réglages

Après vous être connecter, vous assez arriver sur le menu de réglages de la simulation.

Vous avez la possibilité de créer, charger et sauvegarder des jeux de réglages cet onglet.

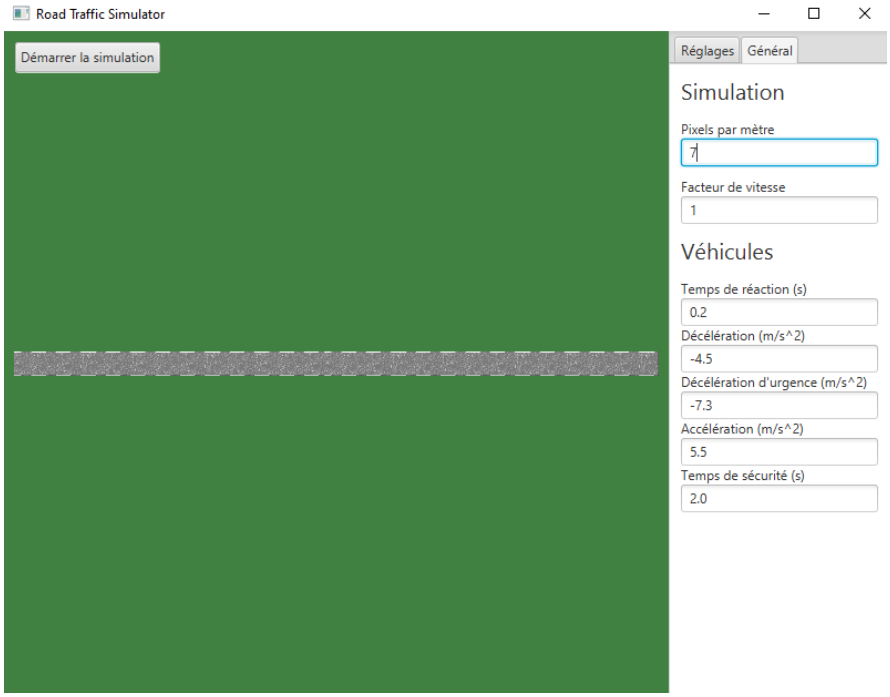
Vous avez aussi la possibilité de cliquer sur les autres onglets pour naviguer sur ceux-ci.



Régler de la simulation

Sur cet onglet, vous pouvez modifier les réglages généraux par rapport à la simulation.

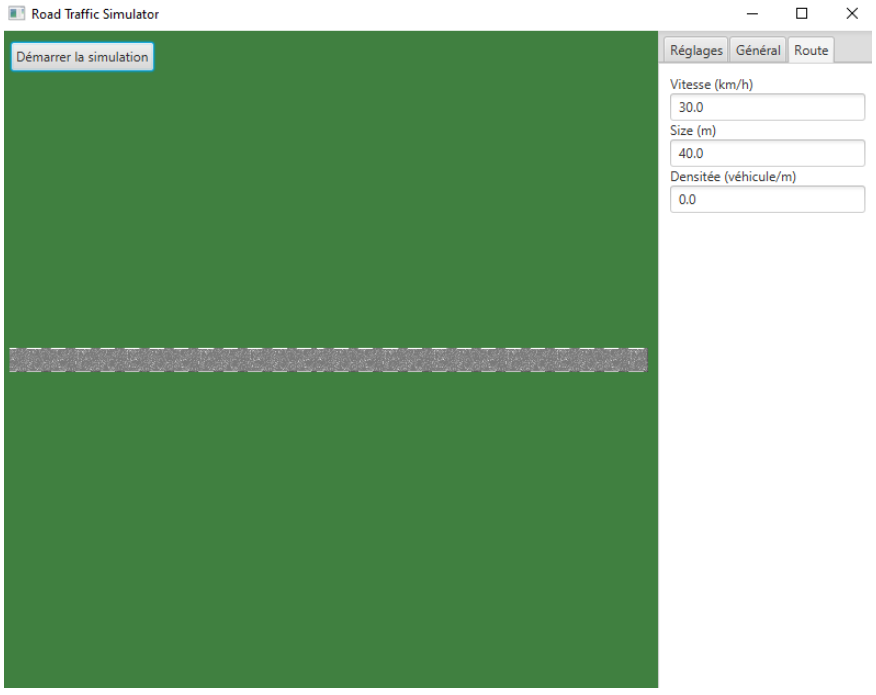
Ses réglages ne sont pas sauvegardés pour le moment mais ça serait rapide à mettre en place pour le prochain mandat.



Controller les routes

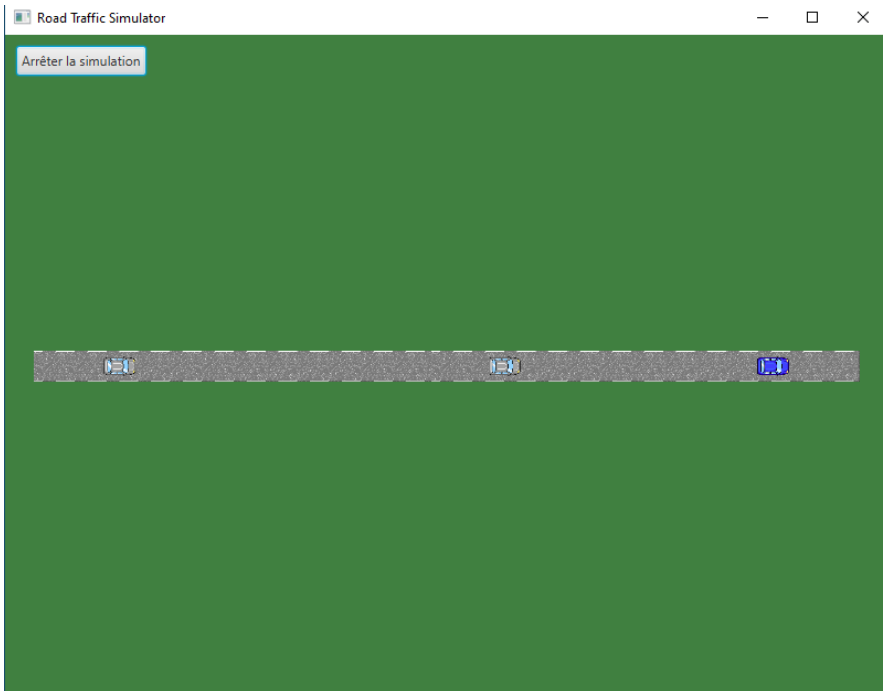
Si vous cliquez sur une route, un onglet « Route » s'ouvrira avec la configuration de la route. Vous pouvez éditer ses réglages et les sauvegarder dans un jeu de réglages plus tard.

Dès que vous cliquerez sur un nouvel onglet, celui-ci se fermera !



Simuler la route

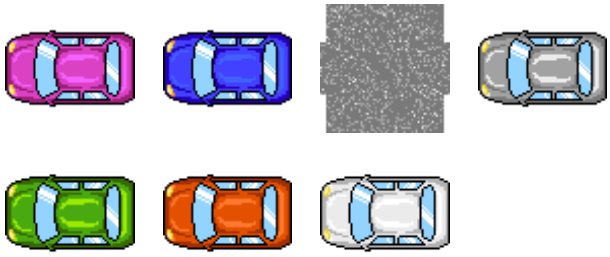
Si vous appuyer sur « Démarrer la simulation », la simulation se lancera et vous pourrez voir comment les véhicules interagissent avec votre tronçon de route.



5.6 Ressources additionnelles

Les textures de l'application ont été réalisé par moi via le logiciel « LibreSprite ».

Ceci est un travail personnel que j'ai réalisé chez moi uniquement par envie d'améliorer le projet. Concèderez ces textures comme des textures téléchargé depuis internet plutôt qu'un travail du projet.



Le fichier d'LibreSprite / Aseprite (sprites.ase) se trouve à côté des textures, dans les ressources de l'application (RoadTrafficSimulator\src\main\resources\app\roadtrafficsimulator\textures)

6 Différence entre la conception et la réalisation

6.1 Aspects du mandat non réalisés

Certains aspects du projet n'ont pu être achevés dans les délais impartis. Voici un aperçu de ces points :

6.1.1 Carrefours

L'implémentation de la gestion des carrefours n'a pas été effectuée. La complexité de la gestion des routes ainsi que leur affichage avec l'API JavaFX ont demandé un investissement important en temps, ne laissant pas suffisamment de ressources pour cette fonctionnalité.

6.1.2 Tolérance des paramètres

Bien que partiellement intégrée dans le code, la tolérance des paramètres n'est pas opérationnelle dans l'application. Ce choix découle d'un souci de qualité, privilégiant une implémentation complète et robuste des fonctionnalités, plutôt que des solutions temporaires et bricolées.

6.1.3 Gestion des distances

L'implémentation de la gestion des distances n'a pas été réalisée dans le projet.

6.1.4 Temps de réaction

Étant donné que ni la gestion des carrefours ni celle des distances n'ont été intégrées, le paramètre de temps de réaction n'est pas utilisé dans l'application. Cette décision découle de la nécessité de prioriser les fonctionnalités centrales du projet, en accord avec les contraintes de temps et de ressources.

7 Test

7.1 Procédure de test

Pour tester l'application, j'ai lancer l'application sur ma machine professionnel (x86-64 Windows 10) depuis Intellij IDEA Community Edition et j'ai réalisé les tests un par un en suivant les instructions. Pour reproduire ces tests, il vous faudra une machine (x86-64, Windows 10 / Debian) ainsi qu'une connexion réseau.

L'application ne fonctionnera pas sur l'architecture ARM car JavaFX ne fonctionne que sur x86-64. Je n'ai pas pu tester l'application sur un Mac (x86-64) car je n'en n'ai pas sous la main.

7.2 Protocol de test

Nr.	Objet testé	Description du test	Attente	Heure	Résultat	Visa
1	Créer un compte	Créer un compte qui n'existe pas dans la base de données.	Le compte sera créé. La page change sur « Simulation ».	**15:23**	Le compte à été créer. La page de simulation c'est ouvert.	**E.K.**
2	Créer un compte	Créer un compte qui existe déjà.	Le compte ne sera pas créé. Afficher une erreur.	**15:23**	Une erreur c'est afficher.	**E.K.**
3	Créer un compte	Créer un compte avec des champs vide.	Afficher une erreur.	**15:24**	Afficher une erreur – Les champs deviennent rouge	**E.K.**
4	Créer un compte	Créer un compte avec le champ « confirmation de mot de passe » qui est diffèrent du « mot de passe »	Afficher une erreur.	**15:24**	Afficher une erreur – Le champ deviens rouge	**E.K.**
5	Se logger	Se logger à un compte qui n'existe pas	Afficher une erreur.	**15:25**	Une erreur c'est afficher.	**E.K.**
6	Se logger	Se logger à un compte qui existe avec le mauvais mot de passe	Afficher une erreur.	**15:25**	Une erreur c'est afficher.	**E.K.**
7	Se logger	Se logger à un compte qui existe avec le bon mot de passe	La connexion va fonctionner. La page change sur « Simulation ».	**15:26**	La connexion fonctionne. La page de simulation s'ouvre.	**E.K.**
8	Charger un jeu de réglages	Charger un jeu de réglages sans sélectionner un jeu	Afficher une erreur.	**15:26**	Afficher une erreur.	**E.K.**
9	Charger un jeu de réglages	Charger un jeu de réglages	Charger le jeu et afficher le succès	**15:27**	Un message de succès s'affiche et les réglages changent	**E.K.**
10	Sauvegarder un jeu de réglages	Sauvegarder un jeu de réglages sans sélectionner un jeu	Afficher une erreur.	**15:28**	Afficher une erreur.	**E.K.**

11	Sauvegarder un jeu de réglages	Sauvegarder un jeu de réglages	Sauvegarder dans la base donnée et afficher le succès	**15:28**	Un message de succès s'affiche.	**E.K.**
12	Créer un jeu de réglages	Créer un jeu de réglages sans nom	Afficher une erreur.	**15:29**	Afficher une erreur – Le champ deviens rouge	**E.K.**
13	Créer un jeu de réglages	Créer un jeu de réglages avec un nom qui existe déjà	Afficher une erreur.	**15:29**	Afficher une erreur.	**E.K.**
14	Créer un jeu de réglages	Créer un jeu de réglages avec un nom qui existe déjà mais qui a été ajouté après le lancement de l'application. Le jeu est donc dans la base de données mais pas la liste.	Afficher une erreur.	**15:29**	Afficher une erreur.	**E.K.**
15	Créer un jeu de réglages	Créer un nouveau jeu de réglages	Créer le jeu de réglages et afficher le succès	**15:30**	Un message de succès s'affiche et le jeu de réglage apparait dans la liste.	**E.K.**
16	Simulation	Lancer la simulation avec les réglages de départ	Le menu d'édition se ferme et la simulation fonctionne.	**15:30**	Le menu d'édition c'est bien fermé et la simulation tourne sans problème	**E.K.**
17	Simulation	Quitter la simulation.	La simulation s'arrête et le menu d'édition s'ouvre.	**15:30**	La simulation s'arrête et le menu s'ouvre.	**E.K.**
18	Simulation	Une fois que la simulation c'est stabilitée, vérifiez les statistiques sur la simulation.	La densité de la route équivaut au « In » et est le	**-**	Pas implémenter, pas testable	**E.K.**
19	Réglage général Pixels par mètre	Changer le « Pixels par mètre » à une valeur inférieur à 0.	Afficher une erreur.	**15:31**	La simulation subie un effet miroir assez amusant. Vu que ceci ne cause aucun problème à l'utilisateur final et que je ne suis pas en avance, je laisserai la simulation dans cette état	**E.K.**
20	Réglage général Pixels par mètre	Changer le « Pixels par mètre » à une valeur entre 1 et 100.	Change la taille et fonctionne normalement.	**15:33**	Change la taille et fonctionne normalement.	**E.K.**
21	Réglage général Facteur de vitesse	Modifier le « Facteur de vitesse » à une valeur inférieur à 0.	Afficher une erreur.	**15:34**	Affiche une erreur et reset la valeur à 1.	**E.K.**
22	Réglage général Facteur de vitesse	Modifier le « Facteur de vitesse » à une valeur entre 0.1 et 10. Une valeur plus grande pourrait causer un problème dans tous les cas.	Changer le facteur de vitesse.	**15:35**	La vitesse change mais pas le résultat de la simulation.	**E.K.**

23	Réglage véhicule Décélération	Modifier cette valeur.	La valeur cause un changement lors des décélérations standard.	**15:35**	La valeur est prise en compte	**E.K.**
24	Réglage véhicule Décélération %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.	**-**	Pas implémenter, pas testable	**E.K.**
25	Réglage véhicule Décélération %	Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.	**-**	Pas implémenter, pas testable	**E.K.**
26	Réglage véhicule Décélération %	Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.	**-**	Pas implémenter, pas testable	**E.K.**
27	Réglage véhicule Décélération d'urgence	Modifier cette valeur.	La valeur cause un changement lors des décélérations d'urgence.	**15:36**	La valeur change mais elle n'est pas prise en compte vu qu'il n'y a pas de gestion des distances.	**E.K.**
28	Réglage véhicule Décélération d'urgence %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.	**-**	Pas implémenter, pas testable	**E.K.**
29	Réglage véhicule Décélération d'urgence %	Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.	**-**	Pas implémenter, pas testable	**E.K.**
30	Réglage véhicule Décélération d'urgence %	Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.	**-**	Pas implémenter, pas testable	**E.K.**
31	Réglage véhicule Accélération	Modifier cette valeur.	La valeur cause un changement lors des décélérations d'urgence.	**15:37**	La valeur est prise en compte	**E.K.**
32	Réglage véhicule Accélération %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.	**-**	Pas implémenter, pas testable	**E.K.**
33	Réglage véhicule Accélération %	Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.	**-**	Pas implémenter, pas testable	**E.K.**
34	Réglage véhicule Accélération %	Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.	**-**	Pas implémenter, pas testable	**E.K.**
35	Réglage véhicule Temps de sécurité	Modifier cette valeur.	La valeur cause un changement de distance entre les véhicules	**-**	La valeur change mais elle n'est pas prise en compte vu qu'il n'y a pas de gestion des distances.	**E.K.**

36	Réglage véhicule Temps de sécurité %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
37	Réglage véhicule Temps de sécurité %	Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
38	Réglage véhicule Temps de sécurité %	Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.	-	Pas implémenter, pas testable	**E.K.**
39	Réglage véhicule Temps de réaction	Modifier cette valeur.	La valeur cause un changement de temps de réaction	**15:41**	La valeur change mais elle n'est pas prise en compte vu qu'il n'y a pas de gestion des temps de réaction.	**E.K.**
40	Réglage véhicule Temps de réaction %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
41	Réglage véhicule Temps de réaction %	Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
42	Réglage véhicule Temps de réaction %	Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.	-	Pas implémenter, pas testable	**E.K.**
43	Cliquer sur une route	Nous cliquons sur un tronçon de route standard	Le menu de la route s'ouvre.	**15:43**	Le menu s'ouvre	**E.K.**
44	Cliquer sur un carrefour	Nous cliquons sur un carrefour.	Le menu de carrefour et le menu de feux de signalisation d'ouvre	-	Pas implémenter, pas testable	**E.K.**
45	Réglage route Limitation de vitesse	Modifier cette valeur.	La valeur cause un changement de comportement des véhicules.	**15:44**	La valeur change bel et bien le comportement des véhicules	**E.K.**
46	Réglage route Limitation de vitesse %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
47	Réglage route Limitation de vitesse %	Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
48	Réglage route Limitation de vitesse %	Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.	-	Pas implémenter, pas testable	**E.K.**

49	Réglage route Taille	Modifier cette valeur.	La valeur cause un changement visuel ainsi qu'un changement de comportement des véhicules.	**15:45**	La taille visuelle ne change pas mais le comportement des véhicules change !	**E.K.**
50	Réglage route Densité	Modifier cette valeur.	La valeur cause un changement de la vitesse d'apparition des véhicules	**15:47**	Le nombre de véhicule par seconde change bel et bien !	**E.K.**
51	Réglage route Densité %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
52	Réglage route Densité %	Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
53	Réglage route Densité %	Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.	-	Pas implémenter, pas testable	**E.K.**
54	Réglage carrefour Limitation de vitesse	Modifier cette valeur.	La valeur cause un changement de comportement des véhicules.	-	Pas implémenter, pas testable	**E.K.**
55	Réglage carrefour Limitation de vitesse %	Modifie cette valeur avec une valeur inférieur à 0.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
56	Réglage carrefour Limitation de vitesse %	Modifie cette valeur avec une valeur supérieur à 100.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
57	Réglage carrefour Limitation de vitesse %	Modifie cette valeur avec une valeur entre 0 et 100.	Les véhicules subissent des différences de comportement plus élevé plus la valeur est grande.	-	Pas implémenter, pas testable	**E.K.**
58	Réglages feux Temps vert	Modifie cette valeur avec une valeur inférieur à 0. À tester sur les 4 différents feux.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
59	Réglages feux Temps vert	Modifie cette valeur avec une valeur différente de 0. À tester sur les 4 différents feux.	La valeur cause un changement de temps de vert à feux tester.	-	Pas implémenter, pas testable	**E.K.**
60	Réglages feux Temps orange	Modifie cette valeur avec une valeur inférieur à 0. À tester sur les 4 différents feux.	Afficher une erreur.	-	Pas implémenter, pas testable	**E.K.**
61	Réglages feux Temps orange	Modifie cette valeur avec une valeur différente de 0. À tester sur les 4 différents feux.	A valeur cause un changement de temps d'orange à feux tester.	-	Pas implémenter, pas testable	**E.K.**
62	Réglages feux Passage au rouge	Modifie les « croix ». À tester pour toutes les lignes et tous les feux.	Lors d'un passage au rouge, les lignes sélectionnées passent au vert	-	Pas implémenter, pas testable	**E.K.**

63	Réglages feux Prochain	Appuyer sur le bouton « Prochain »	Un changement de feu en cours s'opère dans une direction constante (horaire ou anti-horaire, pas de hasard !)	-	Pas implémenter, pas testable	**E.K.**

7.3 Signature du protocole de test

Date	Nom	Signature
Le 5 juin 2024	Kuci Elvin	

8 Problèmes rencontrés

8.1 Maquette d'édition des feux

8.1.1 Problème

Je doutais fortement de comment réaliser une interface simple d'utilisation permettant de contrôler dynamiquement des feux de signalisations. J'ai déjà vu des jeux comme City Syklines qui ont un système complet de drag et de visualisation des directions sur les doutes mais ça reste complexe et l'implémentation de ceci en 1 jour est inimaginable.

8.1.2 Soliton

Après avoir fait un brainstorme chez moi, j'ai trouvé une idée permettant de gérer les feux de signalisations relativement simplement. Le code reste incroyablement complexe mais c'est faisable. Allez voir la maquette pour vous en rendre compte !

8.2 JDBC et NetBeans

8.2.1 Problème

Pour une raison que j'ignore, il était impossible de lier le projet Maven avec le Driver JDBC. NetBeans ne prenait pas en compte les nouvelles configurations et ne téléchargeait pas.

8.2.2 Solution

Un collègue, Filip Jastrzebski, m'a recommander d'essayer d'utiliser IntelliJ IDEA car il a un bon support de Maven. J'avais déjà utilisé IntelliJ IDEA pour des projets personnels mais je ne savais pas qu'il avait un bon support Maven. Après avoir installé la IntelliJ IDEA (Community Edition), tout à fonctionner et j'ai valider ce changement avec mon chef de projet.

8.3 JavaFX est lent !

8.3.1 Problème

Dès lors que 3 véhicules sont à l'écran en même temps, l'animation deviens très saccadée et certaines voitures arrêtent totalement de bouger. Mon estimation est que le Thread JavaFX n'arrive pas à suivre avec le nombre de changement qui sont causé par le backend.

8.3.2 Solution

En ajoutant un « Thread.sleep » de 16 millisecondes à la fin de la boucle (environ 60 mises à jour par secondes), la simulation devient fluide. Encore une fois je pense que c'est dû au Thread JavaFX qui est débordé.

9 Conclusion

Maintenant que ce mandat est conclu et que le projet le sera probablement aussi, il est temps d'analyser comment ce projet c'est dérouler.

D'après-moi, les objectifs majeurs du mandat ont été atteints mais un second mandat est obligatoire pour terminer l'implémentation. Bien que de grosses parties de l'application ne sois pas présentes, la base est là et le code est facilement extensible.

9.1 Améliorations possibles

Un support des routes à plusieurs voies, et donc aussi des carrefours à plusieurs feux.

Un système de création de circuit et la sauvegarde de ceux-ci dans la base de données.

9.2 Auto-évaluation

J'ai l'impression d'avoir bien travailler durant ce projet, même si l'application n'est pas terminée. Comme dit ci-dessus, c'est un gros projet et un petit mandat. De plus, j'ai essayé de respecter la méthodologie de travail au mieux pour montrer un professionnalisme. L'application utilise abondement de l'abstraction et polymorphisme pour la rendre facilement améliorable, un des grands points de l'application.

10 Bibliographie : liste des sources et références

- https://chatgpt.com
- https://gluonhq.com
- https://gaboweb.com/actualites/agence/methodes-traditionnelles-vs-methodes-agiles.html
- https://docs.github.com/fr
- https://sparxsystems.com/enterprise_architect_user_guide/16.1/welcome/index.html
- https://sparxsystems.com/enterprise_architect_user_guide/16.1/modeling_languages/fragment.html
- https://aseprite.org/
- https://www.w3schools.com/
- https://intellij-support.jetbrains.com
- https://www.jetbrains.com/help/idea
- https://en.wikipedia.org
- https://stackoverflow.com
- https://www.tabnine.com/code
- https://www.tutorialspoint.com
- https://gamedev.stackexchange.com
- https://www.reverso.net
- https://docs.oracle.com/javase/8/javafx/api/
- https://netbeans.apache.org/front/main/download/
- https://www.jetbrains.com/fr-fr/idea/
- https://ge.ch/grandconseil/data/texte/PL12730A.pdf
- https://dev.mysql.com/doc/connector-j/en/connector-j-installing.html

De plus, toutes les notions que j'ai appris dans les modules durant ma scolarité et mes connaissances personnel acquise de sources divers.

11 Glossaire

Terme	Signification
Base de données	Machine ou service permettant de sauvegarder des données dessus
Callback	Une fonction ou méthode qui sera appelé en retour
Classe / Class	Une méthodologie permettant de grouper code et données.
Debug / Débuguer	Résoudre un problème en informatique, un bug.
Driver / Un pilote	Logiciel permettant de communiquer avec un matériel ou périphérique.
Fonction	Une méthode avec une valeur de retour.
FXML	Un fichier JavaFX contenant une interface utilisateur.
IDE	Environnement de développement intégré, outil regroupant des fonctions pour coder, tester et debugger
Java	Un langage de programmation créer par Sun Microsystems.
JavaFX	Une Library Java permettant de faire des interfaces.
Library	Des fonctionnalités informatique pré-implémenter et réutilisable.
Méthode	Une procédure dans une classe
Model	Une classe exécutant
MVC	Méthodologie de développement (Model, View, Controller)
Table (Base de données)	Structure de donnée organisé en ligne et colonne.
Thread	Fil d'exécution permettant de réaliser plusieurs tâches à la fois.
PC	Ordinateur personnel.
SQL	Modèle et langage de base de données avec des liens forts.
View	Représentation d'une interface en développement.
Worker	Un Model en MVC.

12 Signatures

Je soussigné déclare que les informations contenues dans ce rapport de travail pratique individuel rendu ce jour le 06.06.2024 dans le cadre de la procédure de qualification de mon CFC d'informaticien, ne sont pas plagiées. Toutes les informations de sources extérieures ainsi que les informations fournies par des tiers durant le déroulement du travail sont consignées.

Date	Nom	Signature
Le 6 juin 2024	Elvin Kuci	

13 Annexes

Vous trouverez en annexes la totalité du contenu du Git et les accès à la base de données.

- Tests technologiques : `/tests/`
- Code source : `/RoadTrafficSimulator/`
- Planning et journal de travail : `/docs/`
- Copie des images de la documentation : `/docs/images/*/`
- Diagrammes et maquettes : `/docs/diagrams/`
- Configurations, requêtes et modèle de la base de données : `/docs/database/`
- Accès à la base de données : `/RoadTrafficSimulator/src/main/resources/app/roadtrafficsimulator/config/db/`