Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

ATL – Ateliers logiciels

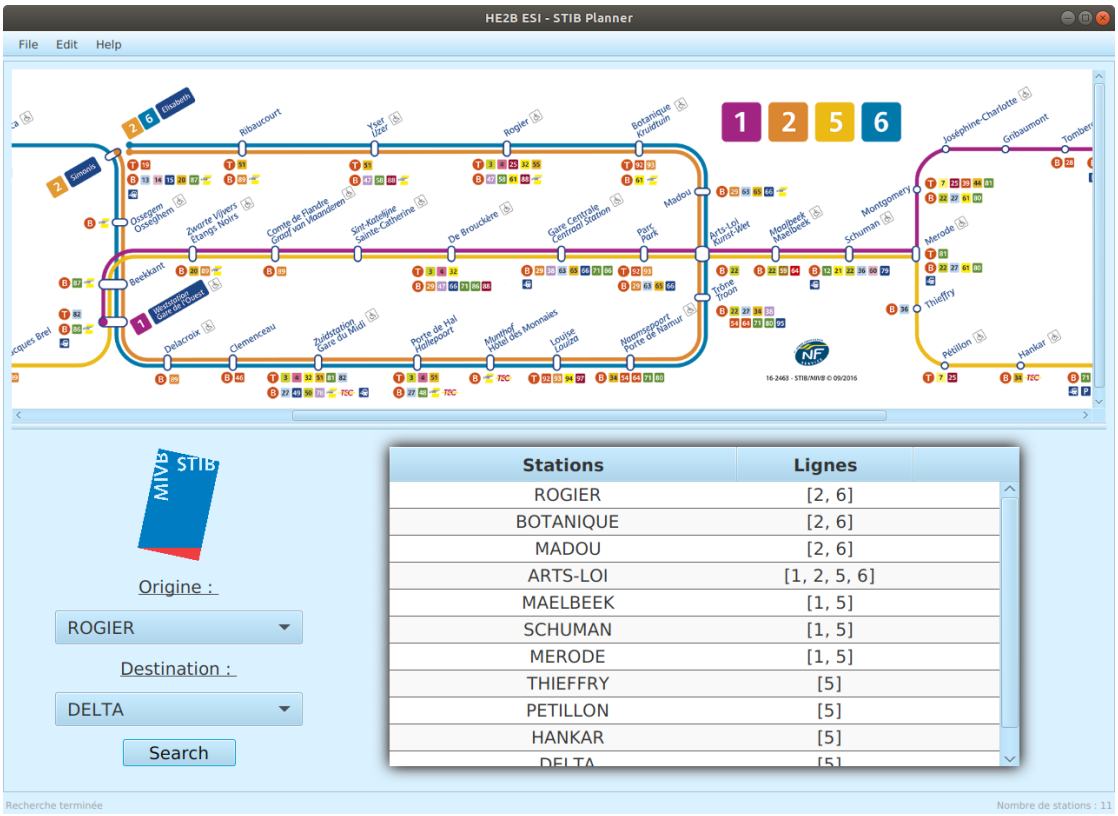
Mise en pratique JDBC

Recherche de chemin dans le réseau de métro de la STIB

> **Consignes**
> Dans cet exercice vous allez réaliser un outil permettant de rechercher le plus court chemin entre deux stations de métro du réseau de la STIB.

1 Présentation

La vue du logiciel a développer peut ressembler à ceci :



On distingue deux champs de recherches permettant d'entrer l'origine et la destination, un bouton pour lancer la recherche, un tableau pour afficher le résultat de la recherche et le plan du réseau dans lequel on peut naviguer.

Vous êtes libres de proposer une autre présentation, mais ces informations doivent être accessibles à l'utilisateur.

Version du document
23 mars 2021

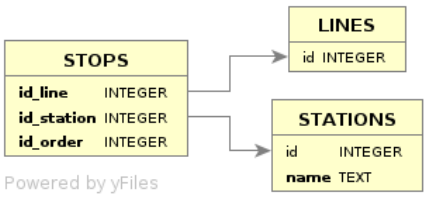
Commencez par créer le projet `maven StibRide`. Ajoutez ensuite les packages nécessaires à l'architecture `MVP`.

2 Base de données

La STIB met à disposition une API donnant accès aux données de son réseau. Vous pouvez y avoir accès en vous inscrivant sur https://opendata.stib-mivb.be/store/ si vous le souhaitez.

Ces données ont permis de construire la base de données dont le script de création est disponible sur `PoEsi`. Vous y trouverez les trois tables suivantes :

- `Lines` : contient le numéro de chaque ligne ;
- `Stations` : contient la liste de toutes les stations et leurs noms ;
- `Stops` : contient les arrêts de chaque ligne, un numéro d'ordre permet de connaître la position de la station dans la ligne.



Créez une base de données `SQLite` avec ces trois tables, et développez les classes `DTO`, `DAO` et `Repository` nécessaires à leurs utilisations.

Implémentation du repository

N'implémentez que les méthodes nécessaires pour la lecture des tables. Si les autres méthodes (`update`, `insert` et `delete`) sont nécessaires vous les ajouterez par la suite. Ajoutez les tests unitaires des méthodes développées afin de vérifier leurs fonctionnements.

3 Le modèle : recherche de chemin

Le modèle de l'application doit lire les informations enregistrées dans la base de données et constituer un graphe.

> Le réseau de métro de la STIB peut être représenté par un graphe non orienté et pondéré. Chaque arrête du graphe a une pondération valant 1 dans cette version de l'application.

Pour construire ce graphe vous pouvez lire les informations de la base de données via les requêtes implémentées par défaut ou ajouter des requêtes spécifiques. Ces nouvelles requêtes peuvent utiliser des jointures.

Pour rechercher un chemin dans ce graphe, plusieurs algorithmes sont imaginables. Dans le cadre de cet exercice implémentez l'algorithme de **Dijkstra**. Vous en trouverez une

description via https://www.baeldung.com/java-dijkstra. N'hésitez pas à vous écartez de l'implémentation.

Une méthode de votre modèle doit prendre en paramètre une station d'origine et une station de destination et rechercher le plus court chemin entre ces deux stations.

4 Vue

Développez via un (ou plusieurs) fichier(s) FXML la vue de votre logiciel.

Description de la vue

Cette vue doit permettre à l'utilisateur :

- ▷ de rechercher une station d'origine ;
- ▷ de rechercher une station de destination ;
- ▷ de démarrer la recherche du plus court chemin entre ces stations ;
- ▷ de consulter le chemin calculé ;
- ▷ de consulter le plan du réseau.

Afin de faciliter la recherche d'une station dans la liste fournie par la STIB, nous vous demandons d'utilisez un composant `JavaFX` qui n'est pas inclut dans la librairie standard, il s'agit du composant `SearchableComboBox`.

Une `SearchableComboBox` permet à l'utilisateur de rechercher dans une liste déroulante en tapant du texte. Jetez un œil sur sa documentation pour en savoir plus.

Afin d'utiliser cette nouvelle classe dans le contrôleur `FXML`, il faut commencer par ajouter la dépendance à cette nouvelle librairie dans le fichier `pom.xml`.

```
<dependency>
   <groupId>org.controlsfx</groupId>
   <artifactId>controlsfx</artifactId>
   <version>11.0.3</version>
</dependency>
```

Vous pourrez ensuite importer la librairie dans votre contrôleur `FXML` via

```
import org.controlsfx.control.SearchableComboBox
```

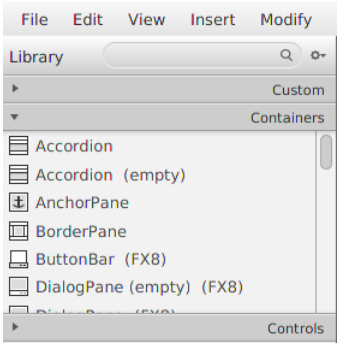
Composants et SceneBuilder

Pour qu'un composant soit utilisable avec `SceneBuilder`, il faut également donner le chemin vers ce composant à `SceneBuilder`.

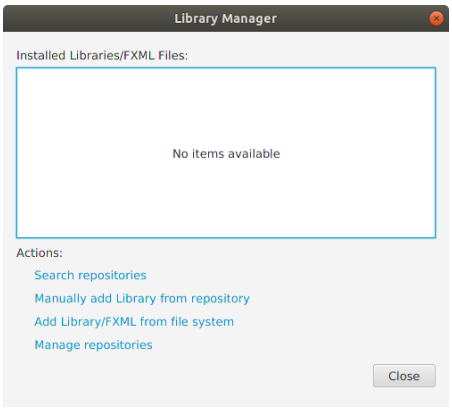
Au départ `SceneBuilder` ne connaît que les composants standards (`Label`, `TextField`, `Button`,...). Si de nouveaux composants sont utilisés, il faut renseigner le chemin vers ces nouveaux composants.

Tutoriel sur l'ajout d'une librairie dans `SceneBuilder`

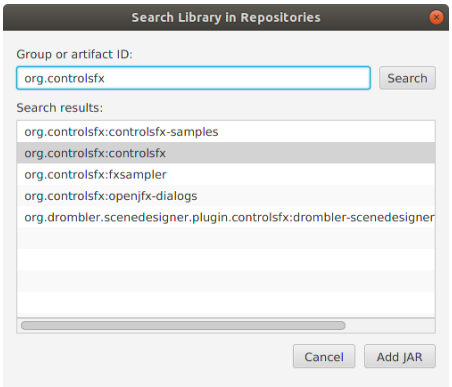
- Commencez par repérer les options de gestion de librairies via la roue dentée à droite du champs de recherche.



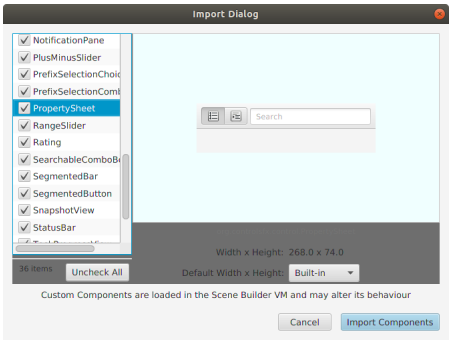
- Cliquez sur la roue dentée, et utilisez le menu `JAR/FXML Manager`.



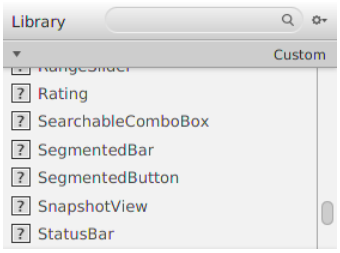
- L'option `Search repositories` permet de rechercher les composants fournis par `org.controlsfx`.



`SceneBuilder` affiche la liste des nouveaux composants graphiques auxquels vous avez dorénavant accès.



Tous ces composants sont accessibles via le menu `Custom`.



Un glisser-déposer permet de les utiliser comme les composants standards.

Vous trouverez d'autres composants en consultant la section `Community` de `https://openjfx.io/` : un calendrier, un formulaire, les css Boostrap ou encore des composants dédiés au jeu.

5 Sauvegarde de ses trajets fréquents

Pour que l'utilisateur puisse consulter facilement le chemin de ses trajets favoris, permettez à l'utilisateur de sauvegarder une liste de trajets favoris. Chaque favoris possédant un nom [1], une origine et une destination. **Modifiez** le schéma de la base de données suivant vos besoins. L'utilisateur peut **supprimer** ou **modifier** un trajet favori déjà enregistré. Modifiez également la vue pour permettre la gestion de ces trajets favoris.

6 Pour aller plus loin ....

> Cette section n'est pas obligatoire.

Afin d'améliorer cette application vous pouvez :

- ▷ permettre d'indiquer à l'utilisateur quand il doit changer de lignes ;
- ▷ mettre des pondérations différentes sur les arrêtes afin de représenter le temps de trajet ;
- ▷ consulter l'api de la STIB [2] pour obtenir l'heure d'arrivée du prochain métro [3].

1. `École-Domicile` par exemple
2. pour consulter une api vous pouvez utiliser `retrofit` `https://www.baeldung.com/retrofit`
3. attention plusieurs numéros de nœuds ont été modifiés, les détails sont sur PoEsi.