RAPPORT de PROJET

Design and make your own City Mapper Application

Encadré par
M.Youcef
M.Mustapha
M.Mirwasser

Soumia MEDDAS
Myriam MILHA
Thierry LAGUERRE

L3 informatique
Base de données
2022-2023

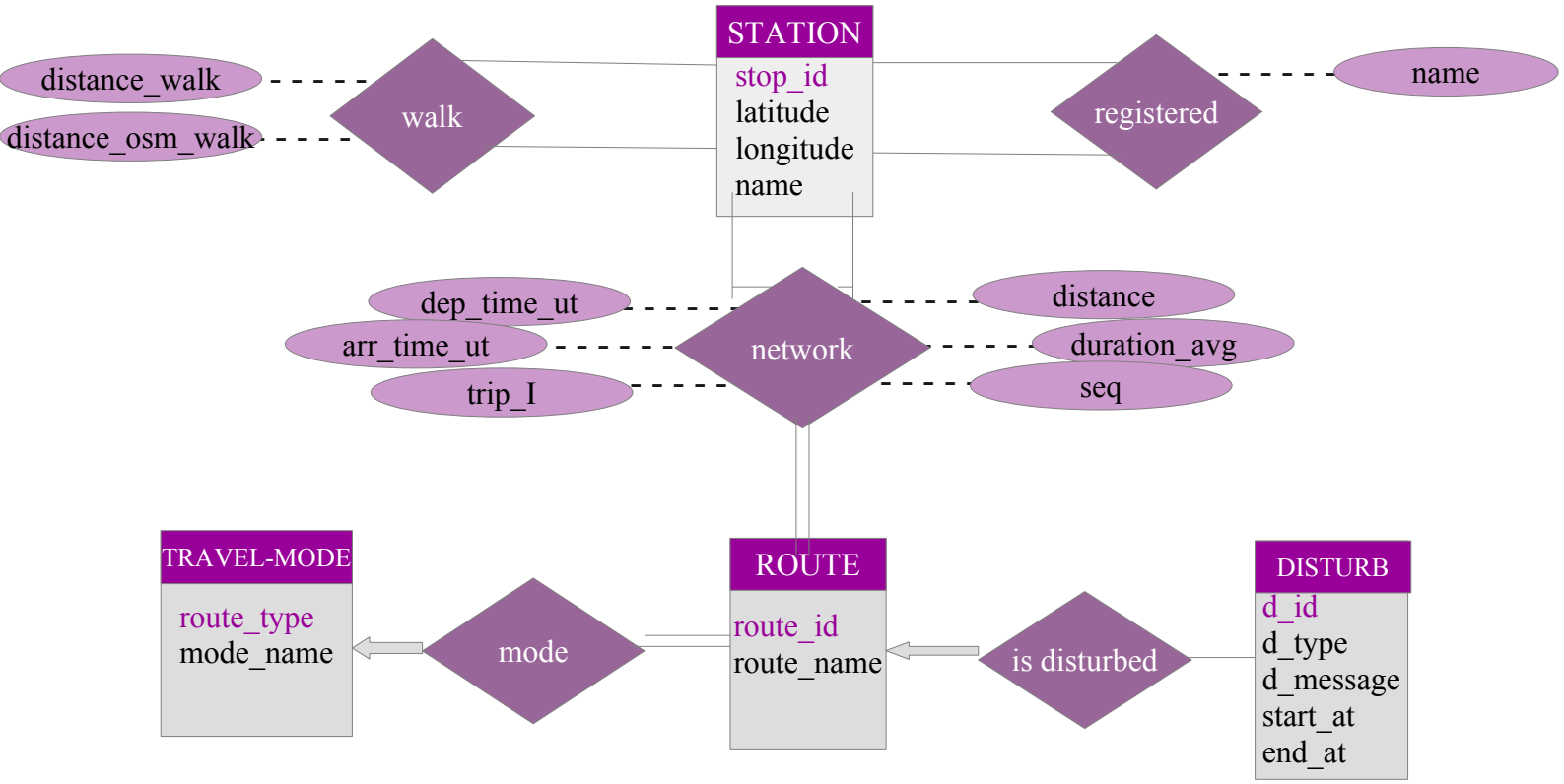
SOMMAIRE

Pour lancer l'application :

python3 connect.py

python3 main.py

➢ Le E-R diagram des données et explication E-R diagram



- On a choisi la ville Paris.

- Pour la database, on nous a donné des fichiers csv contenant les données.

- Grâce au fichier network_nodes.csv, on a trouvé les attributs de la table station.

- Grâche au fichier routes.geojson, on a sélectionné les attributs de la table route.

- « station » et « route » sont des entités fortes (strong entities).

- Les types de routes sont qualifiés par des nombres. On a trouvé une table associant chaque nombre de route_type un nom qui sont : bus, subway, tram, rail.

On a crée alors la table « travel_mode » qui est une entité faible (weak entity) de « route » qui associe un mode de transport à un type de route.

On a séparé la table « travel_mode » de la table « route » pour respecter la BCNF form.

- La table « walk » est une relation entre 2 stations.

- La table « network » est une relation entre 2 stations et une route.

- Et pour finir, on a crée les tables « disturb » pour notifier les perturbations pour une route spécifique et « registered » pour enregistrer les itinéraires de la maison, le travail ou l'université par exemple, pour ne pas à avoir à remplir le from-box et le to-box de nouveau pour des trajets récurrents.

- **Liste des tables avec leur description**

  La liste de toutes les dépendances entre les attributs

- station (stop_id, latitude, longitude, name)
- route (route_id, route_name, route_type)
- travel_mode (route_type, mode_name)
- walk(from_stop, to_stop, distance_walk, distance_osm_walk)
- network (from_stop, to_stop, dep_time_ut, arr_time_ut, route_type, trip_I, seq, route_id, distance, duration_avg)
- network_bis (from_stop, to_stop, dep_time_ut, arr_time_ut, route_type, trip_I, seq, route_id)
- disturb (d_id, d_type, d_message, start_at, end_at, route_id)
- registered(name, stop_id)

primary keys: purple

foreign keys:

- **station :**

Parsing à partir de station.py de « network_nodes.csv » pour obtenir station.sql en nous inspirant du fichier du TP.

- **route :**

Parsing à partir de route.py de « routes.geojson » pour avoir les données dans le fichier route.csv.

Parsing à partir de route2.py du fichier route.csv pour obtenir route.sql

- **travel_mode :**

Création du fichier travel_mode.csv (4 modes).

Parsing à partir de travel_mode.py du fichier travel_mode.csv pour obtenir travel_mode.sql

travel_mode.sql

```
INSERT INTO travel_mode VALUES ( '0', 'tram ');
INSERT INTO travel_mode VALUES ( '1', 'metro ');
INSERT INTO travel_mode VALUES ( '2', 'RER ');
INSERT INTO travel_mode VALUES ( '3', 'bus ');
```

- **walk :**

Parsing à partir de walk.py de « network_walk.csv » pour obtenir walk.sql

- **network :**

Parsing à partir de network.py de « network_temporal_week.csv » pour obtenir network.sql

Parsing à partir de network_update.py de « network_combined.csv » pour obtenir les attributs distance et duration_avg manquants dans le premier parsing pour obtenir le fichier complet network.sql

- **network _bis:**

Parsing à partir de network_bis.py de « network_temporal_day.csv » pour obtenir network_bis.sql

Parsing à partir de network_bis_update.py de « network_combined.csv » pour obtenir les attributs distance et duration_avg manquants dans le premier parsing pour obtenir le fichier complet network.sql

- **disturb :**

Créer le fichier disturb.csv en écrivant 2 perturbations possibles.

Créer le fichier disturb.py

Parsing de disturb.csv pour obtenir disturb.sql

disturb.sql

```
INSERT INTO disturb (route_id, d_type, d_message, start_at, end_at) VALUES
 ( 1068,'Mouvement social', 'Mouvement social à partir du 1er janvier jusqu au 6 janvier 2023', '
2023-01-01', '2023-01-06');
INSERT INTO disturb (route_id, d_type, d_message, start_at, end_at) VALUES
 ( 1055, 'Travaux nocturnes', 'En raison de travaux nocturnes, votre ligne sera pertubé à partir du
2 février jusqu au 15 février 2023', '2023-02-02', '2023-02-15');
```

- **Les requêtes SQL associées à chaque fonctionnalité de notre application**

Pour créer les tables dans psql, on se connecte à notre base de données psql et on lance la commande tables.sql qui contient le code sql suivant :

```
create table station(
   stop_id numeric(5,0),
   latitude TEXT ,
   longitude TEXT,
   name varchar(150),
   constraint PK_station primary key(stop_id)
);


create table registered(
  name varchar(50),
  from_station varchar(150),
  to_station varchar(150),
  constraint PK_registered primary key(from_station, to_station)
);


create table  travel_mode(
  route_type numeric(2,0),
  mode_name varchar(30),
   constraint PK_travel_mode primary key(route_type)
);


create table  route(
   route_id numeric(10,0),
   route_name  varchar(30),
   route_type  numeric(2,0),
   constraint PK_route  primary key(route_id),
   constraint FK_route_travel_mode foreign key(route_type) references travel_mode
);
```

```
create table  network(
   from_stop numeric(5,0),
   to_stop numeric(5,0),
   dep_time_ut  numeric(12,0),
   arr_time_ut numeric(12,0),
   route_type numeric(2,0),
   trip_I numeric(7,0),
   seq numeric(5),
   route_id numeric(10,0),
   distance numeric(10,5),
   duration_avg numeric(30,20),
   constraint PK_network primary key(from_stop,to_stop,dep_time_ut,arr_time_ut,route_id,trip_I),
   constraint FK_network_from_station foreign key(from_stop) references station,
   constraint FK_network_to_station foreign key(to_stop) references station,
   constraint FK_network_route foreign key(route_id) references route
);


create table  walk(
   from_stop numeric(5,0),
   to_stop numeric(5,0),
   distance_walk numeric(10,5),
   distance_osm_walk numeric(10,0),
   constraint PK_walk primary key(from_stop,to_stop),
   constraint FK_walk_from_station foreign key(from_stop) references station
   );


create table disturb(
  d_id serial primary key,
  d_type varchar(150),
  d_message TEXT,
  start_at date,
  end_at date,
```

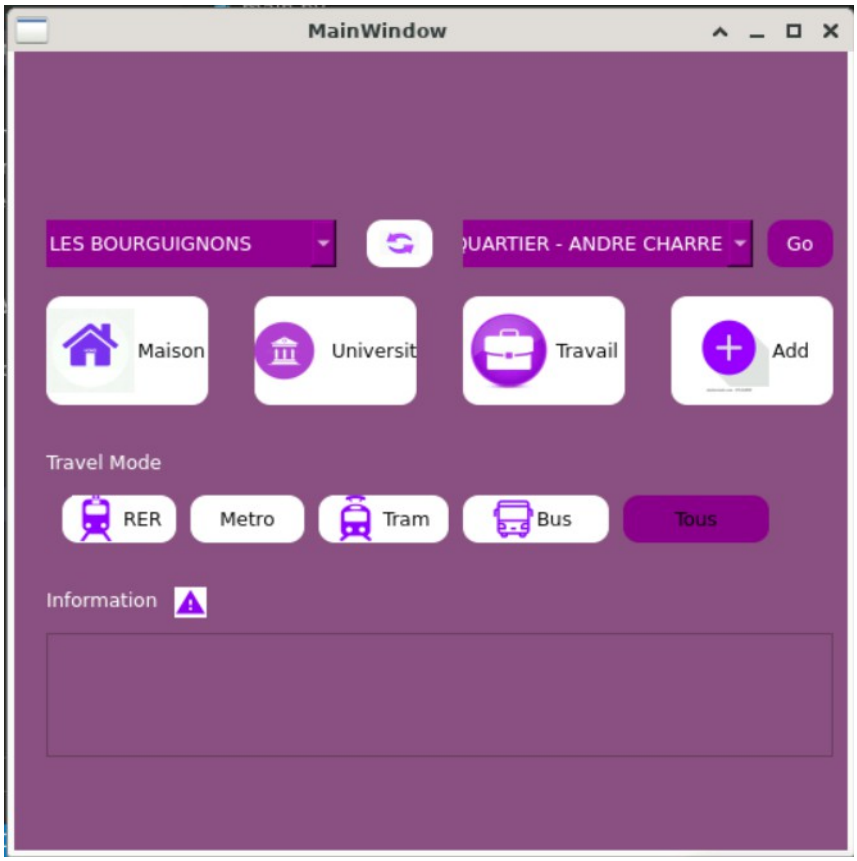
```
  route_id numeric(10,0),
  constraint FK_disturb_route foreign key(route_id) references route
);



create table network_bis(
    from_stop numeric(5,0),
    to_stop numeric(5,0),
    dep_time_ut  numeric(12,0),
    arr_time_ut numeric(12,0),
    route_type numeric(2,0),
    trip_I numeric(7,0),
    seq numeric(5),
    route_id numeric(10,0),
    constraint PK_network_bis primary key (from_stop, to_stop, dep_time_ut, arr_time_ut, route_id,
trip_I),
    constraint FK_network_bis_from_station foreign key(from_stop) references station,
    constraint FK_network_bis_to_station foreign key(to_stop) references station,
    constraint FK_network_bis_route foreign key(route_id) references route
);
```

Pour le design graphique de l'interface, on a utilisé PyQt5 Designer.

Pour interagir avec l'interface, le code est en python.

On a modifié le code pour se connecter à la base de données et effectuer des requêtes pour des fonctionnalités.

Voilà la page d'accueil :



- Pour enregistrer des itinéraires habituels, on clique sur le bouton Add.

Ici, on a déjà 3 itinéraires enregistrés : Maison, Université, Travail.

- Pour choisir le mode de transport, le choix se fait dans la section Travel Mode.

Les choix proposés sont : RER, Metro, Tram, Bus, Tous.

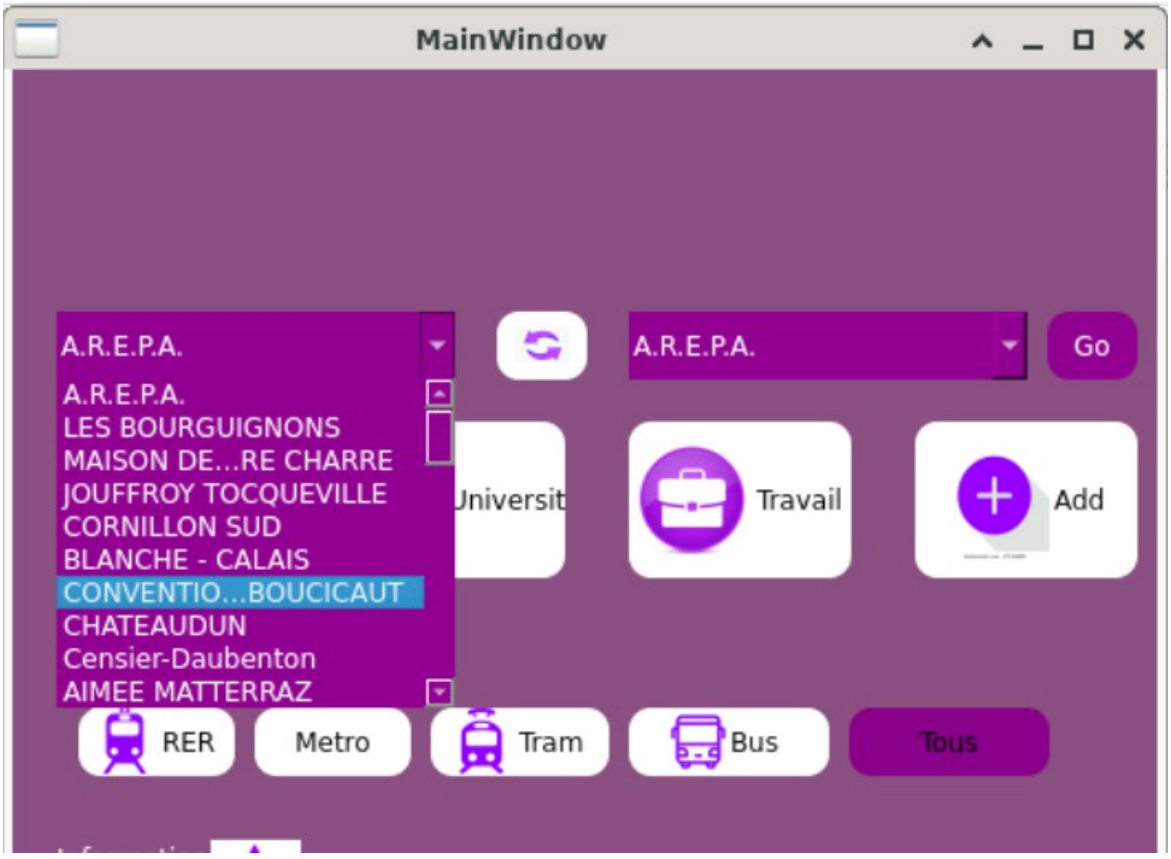
Par défaut le choix est sur « Tous ».

- Pour voir les perturbations d'itinéraires, il y a la section Information.

➢ Fonctionnalités conçus dans notre application

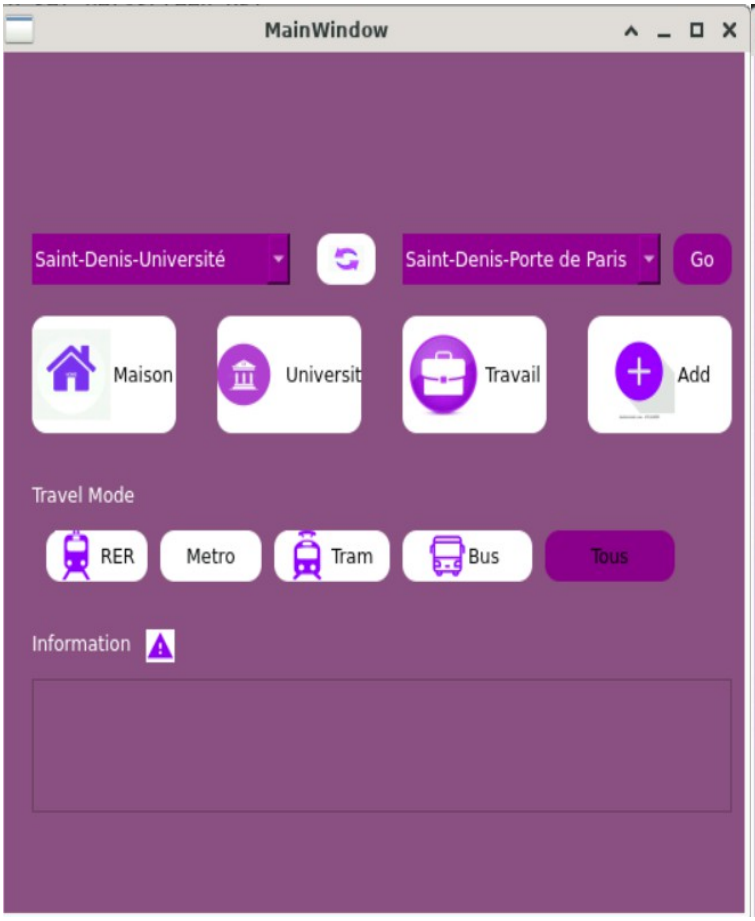
1. Montre les stations

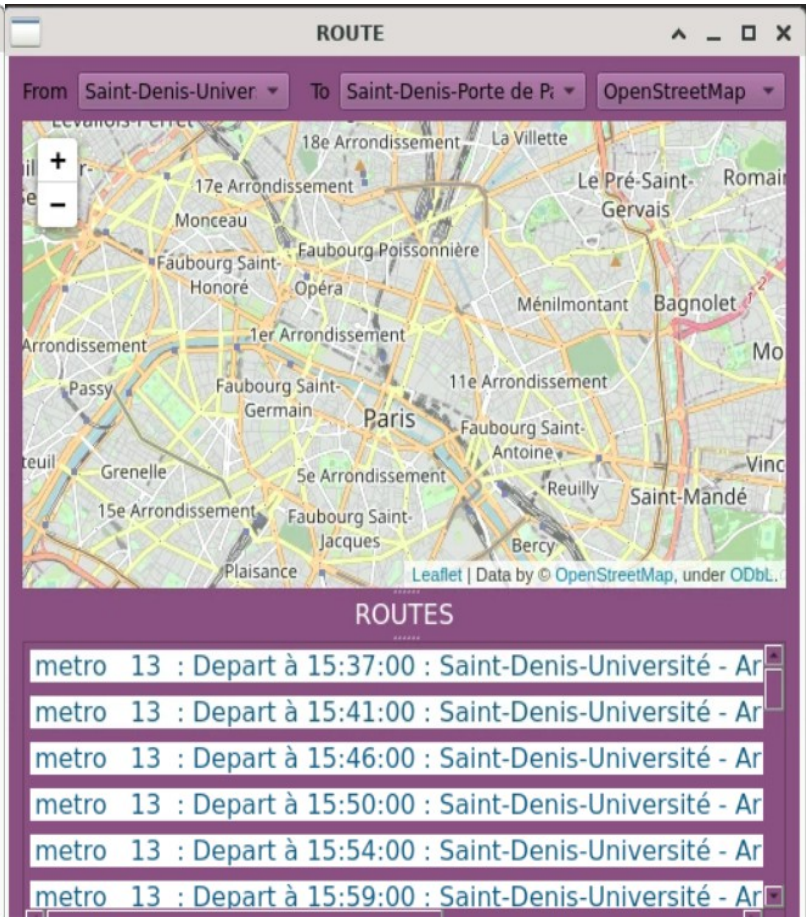
SELECT distinct name FROM station ;



2. Trouve les itinéraires possibles pour un hop (sans choisir un travel mode)

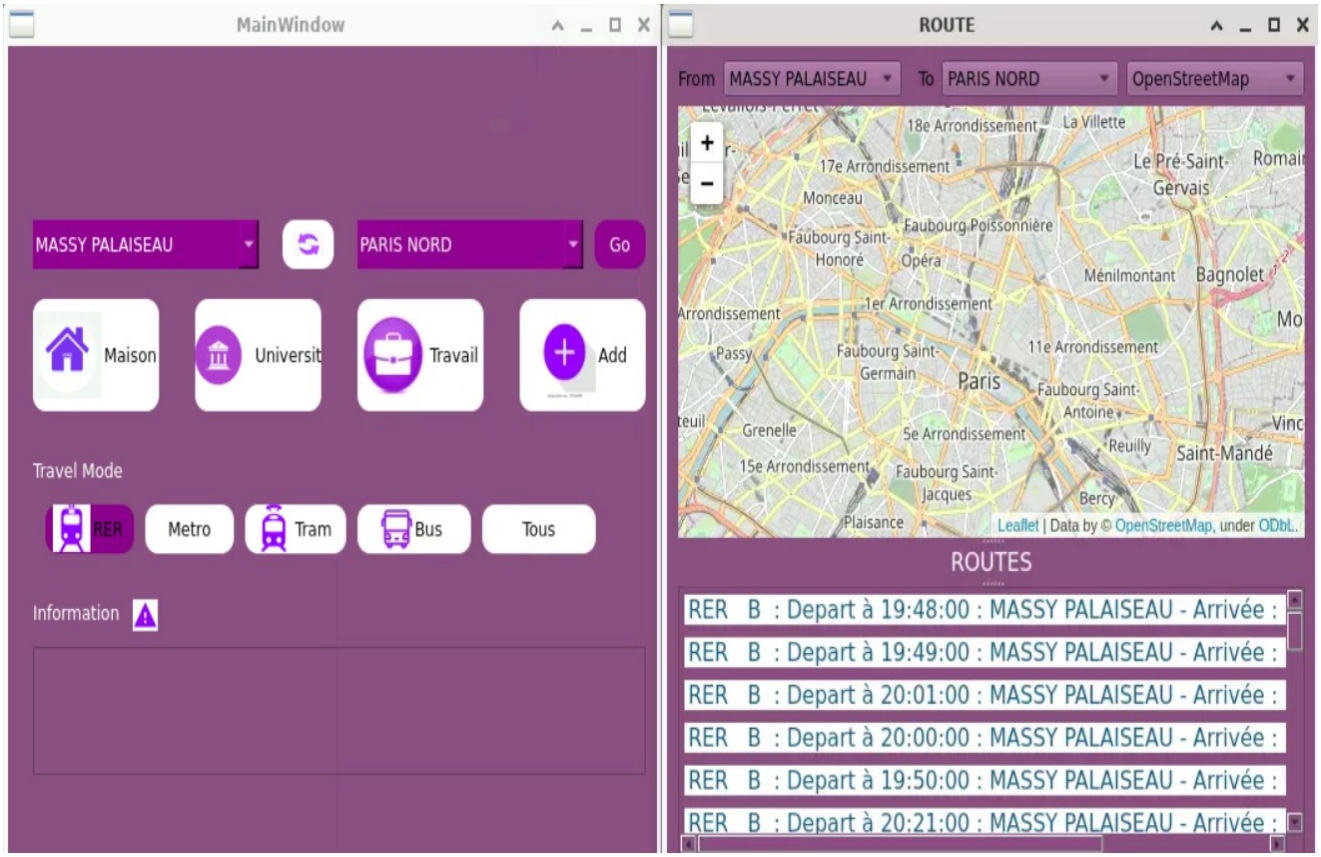
```
SELECT S.name, N.trip_I, N.arr_time_ut, R.route_name, T.mode_name, S.latitude, S.longitude
FROM network_bis N, station S, route R, travel_mode T
WHERE R.route_type = T.route_type and R.route_id = N.route_id and S.stop_id = N.from_stop
and trip_I IN
        ((SELECT distinct trip_I
        FROM network_bis N, station S
        WHERE S.stop_id = N.from_stop and S.name = $${from_stop}$$)

        INTERSECT

        (SELECT trip_I
        FROM network_bis N1, station S1
        WHERE S1.stop_id = N1.to_stop and  S1.name = $${to_stop}$$))
ORDER BY trip_I, N.route_id, N.arr_time_ut ;
```





3. Trouve les itinéraires possibles pour un hop (avec un mode de transport spécifique)

```
SELECT S.name, N.trip_I, N.arr_time_ut, R.route_name, T.mode_name, S.latitude, S.longitude
FROM network_bis N, station S, route R, travel_mode T
WHERE R.route_type = T.route_type and R.route_type = $${route_type}$$ and R.route_id =
N.route_id and S.stop_id = N.from_stop and trip_I IN
        ((SELECT distinct trip_I
        FROM network_bis N, station S
        WHERE S.stop_id = N.from_stop and S.name = $${from_stop}$$)

        INTERSECT

        (SELECT trip_I
        FROM network_bis N1, station S1
        WHERE S1.stop_id = N1.to_stop and  S1.name = $${to_stop}$$))
ORDER BY trip_I, N.route_id, N.arr_time_ut ;
```



4. Obtenir un itinéraire spécifique en cliquant sur un choix d'itinéraire de la liste

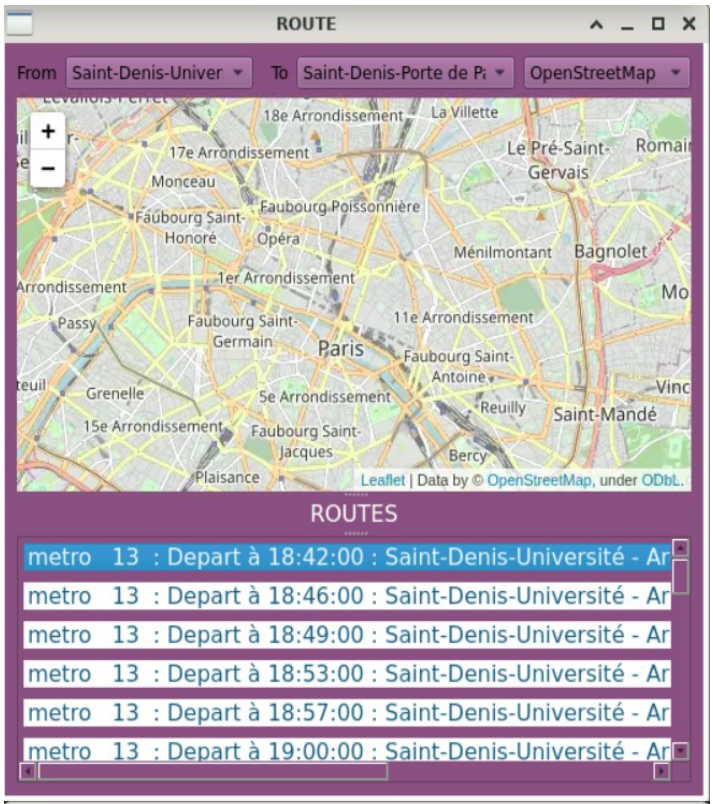
SELECT S.name, N.arr_time_ut, R.route_name, T.mode_name

FROM network_bis N, station S, route R, travel_mode T

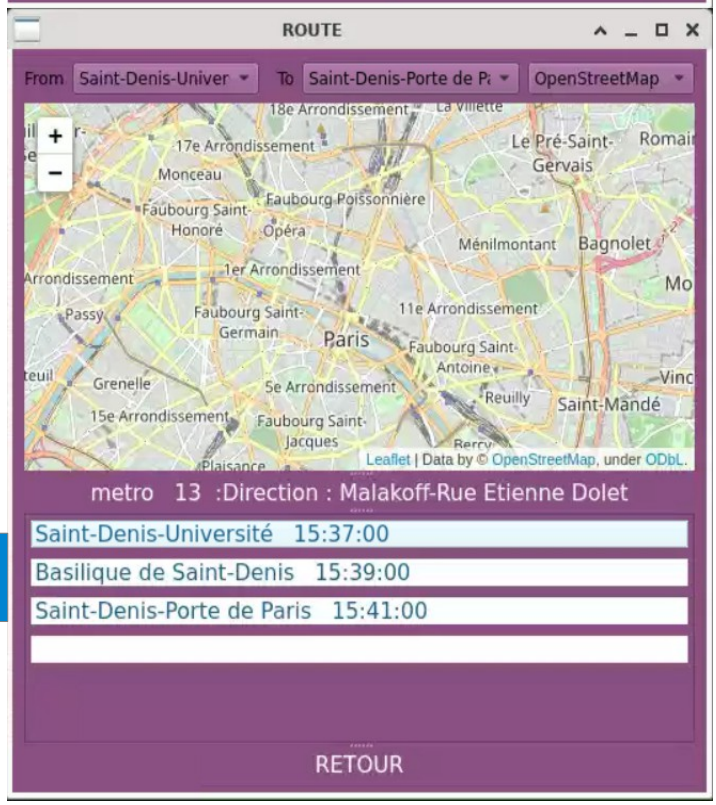
WHERE R.route_type = T.route_type and R.route_id = N.route_id and S.stop_id = N.from_stop and N.trip_I = $${trip_id}$$

ORDER BY N.arr_time_ut ;

CLICK



Après CLICK



5. Donne l'itinéraire enregistré demandé

SELECT from_station, to_station FROM registered WHERE name = $${hist_name}$$ LIMIT1 ;

6. insère dans la table l'itinéraire habituelle enregistré

Quand on clique sur le bouton Add après avoir rempli correctement from_box et to_box, on a la possibilité d'enregistrer l'itinéraire dans Maison, Travail ou Université. Cette requête sera effectuée :

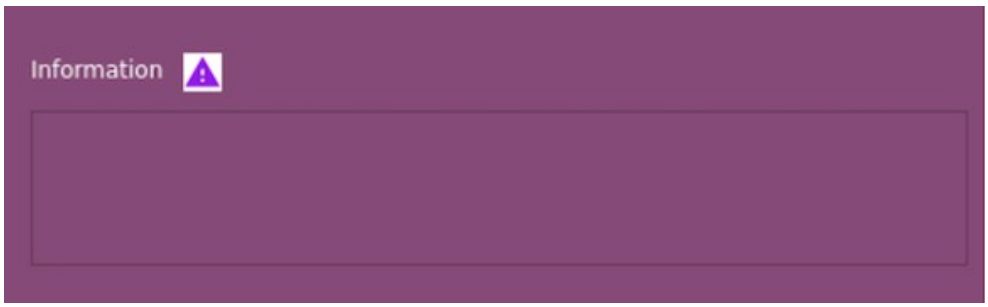
INSERT INTO registered VALUES ($${hist_name}$$, $${from_station}$$, $${to_station}$$

Et quand on clique sur Maison, Travail ou Université, on accède directement à l'itinéraire correspondant.

```
SELECT from_station, to_station
FROM registered
WHERE name = 'Maison'

SELECT from_station, to_station
FROM registered
WHERE name = 'Travail'

SELECT from_station, to_station
FROM registered
WHERE name = 'Université'
```

7. Informations de perturbations

SELECT mode_name, route_name, d_message, end_at FROM travel_mode T, route R, disturb D WHERE T.route_type = R.route_type and D.route_id = R.route_id;



Voici les informations de perturbations qui sont censées s'afficher :

```
l3info_20=> SELECT mode_name, route_name, d_message, end_at FROM travel_mode T, route R, disturb D WHERE T.route_type = R.route_type and D.route_id = R.route_id;
 mode_name | route_name |                                                  d_message                                                  |   end_at
-----------+------------+-------------------------------------------------------------------------------------------------------------+------------
 RER       | E          | Mouvement social à partir du 1er janvier jusqu au 6 janvier 2023                                            | 2023-01-06
 RER       | H          | En raison de travaux nocturnes, votre ligne sera pertubé à partir du 2 février jusqu au 15 février 2023 | 2023-02-15
(2 lignes)
```

- **Toutes les difficultés rencontrées lors de la réalisation du projet**

• Problèmes de réseau

• Compatibilité de nos emplois du temps

• Guacamole ne fonctionne pas sur l'ordinateur de Myriam, fonctionne lentement lorsqu'un partage d'écran est effectué et l'attente d'entrée dans guacamole est longue. La fac étant fermé pendant les vacances cela nous a handicapés.

• Installation des modules : QWebEnginePage et QwebEngineView

• Le problème d'espace disque nous a beaucoup embêté.

• Dans la section Information, les messages de perturbations ne s'affichent pas.

• On voulait réaliser la requête pour plus d'un hop mais on n'a pas eu le temps.

• Lorsqu'on enregistre un itinéraire, il s'enregistre seulement dans « Maison ».

➢ Contribution de chaque membre du groupe

On a fait tous les 3 les tâches suivantes, s'entraidant pendant les réunions sur Google Meet :

• Créer les tables en se connectant sur psql :

\i tables.sql

• Créer fichier python qui lit fichier CSV pour insérer les données dans chaque table dans un fichier <table>.sql

• Se connecter et entrer dans la base psql pour entrer les données dans chaque table :

\i station.sql
\i travel_mode.sql
\i walk.sql
\i route.sql
\i network.sql
\i network_bis.sql
\i disturb.sql

• Créer l'interface graphique :

fichier de connexion (connect.py)

fichier de l'interface de sauvegarde d'itinéraire (save_route.py)

fichier de l'interface ROUTE pour voir les itinéraires avec horaires (route.py)

fichier de l'interface MainWindow pour chercher l'itinéraire en choisissant le mode de voyage et avec les perturbations prévus visibles (Main.py)

• Rapport du projet