Visualisateur GTFS

**Application basée sur** : PostgreSQL/PostGIS, Ruby On Rails, MapboxGL

**Objectif** : Visualiser les tracés du réseau du RTL, visualiser les données de temps réel pendant les phases de développement, exporter les données en vue d’une utilisation pour des cartes animées, calculer le nombre de véhicules en service passant dans un corridor donné, exporter la forme géographique de tracés spécifiques.

Comme pour OVAP, la première étape consiste à procéder à l’installation des extensions et à mettre en place la base de données en accomplissant les étapes qui suivent

* Peupler le fichier database.yml avec les informations sur le serveur (un exemple est fourni)
* Peupler le fichier config.yml avec les informations de connexion à mapbox (un exemple est fourni)[[1]](#footnote-1)
* Préparer l’installation des extensions (

bundle install

* Créer la base de données

rake db:create && rake db:migrate

* Importer les données par défaut sur les municipalités

rake db:seed

Le visualisateur GTFS repose sur une commande principale, qui elle-même appelle de nombreuses méthodes. La tâche

rake import\_from\_csv[dossier\_contenant\_le\_gtfs]

contient l’ensemble des données nécessaires au peuplement de la base de données. La commande importera tous les fichiers du GTFS nécessaire à la visualisation dans le cas du RTL, soit les routes, les trips, les stops, les stop\_times et les shapes. La tâche créera en outre les informations précalculées nécessaires à la visualisation, notamment les regroupements sous forme de tracés et des arrêts formant les dits tracés, en plus de générer la forme géographique de chacun de ces tracés. Elle précalcule aussi les formes de chacun des trips individuels, facilitant les calculs de nombre de passage sur des corridors définis par l’utilisateur dans l’interface. À noter que lors de la création des tracés, l’application fusionne les allers et retour des boucles dans les tracés, tout en conservant les trips individuels. Cette situation permet de conserver l’information permettant de calculer le nombre de passage dans un corridor donné, tout en permettant de visualiser les boucles dans leur ensemble, ce qui est plus exact d’un point de vue de la réalité des opérations.

D’autres tâches permettent aussi de générer des informations de sortie. La tâche

rake export\_to\_deck\_json[nom\_du\_type\_de\_service]

, où type de service est le nom court du service, soit ‘SE’,’SA’, etc., permet d’exporter un fichier json pouvant être chargé dans la carte animée deck.gl, pour un service du choix de l’utilisateur. Si plusieurs json doivent être fusionnés, une autre tâche pouvant être appelée avec

rake merge\_deck\_files

fusionnera l’ensemble des fichiers json trouvés dans un dossier (le dossier est hard-codé comme c:\deck\trips\deck\_trips, mais peut être modifié à l’intérieur du code source dans lib\tasks\deck\_tools.rake). Finalement, une dernière tâche en lien avec la carte animée permet d’exporter le nombre de véhicules en service pour toute une journée de service à des intervalles d’une minute. Cette variable sera utilisée dans la carte animée pour afficher des statistiques en continu

rake export\_vehicles\_in\_service[nom\_du\_type\_de\_service]

Le visualisateur inclut aussi la possibilité d’obtenir des données de temps réel. La tâche

rake fetch\_vehicle\_positions

se charge d’obtenir ces données. À noter que cette tâche est une boucle sans fin, si bien qu’il sera nécessaire d’en arrêter l’exécution dans la console en utilisant la commande CTRL+C. Les données peuvent par la suite être visualisées en temps réel, ou a posteriori dans les pages /realtime et /realtime\_historic. À l’heure actuelle, les données de temps réel ne seront associées au service planifié qu’uniquement pour les jours de semaine puisque le calendrier n’est pas importé dans l’application.

L’application ne peut contenir qu’une assignation à la fois, et doit donc être réinitialisée périodiquement pour intégrer de nouvelles données GTFS. Il est toutefois possible de maintenir en fonction plusieurs instances de l’application, à condition d’exécuter les serveurs sur des ports distincts. La référence « Anatomie d’une application Rails » offre de l’information sur comment procéder. À noter que la version de MapboxGL dans cette application doit rester à 0.38, l’animation sur les lignes étant dépendante de cette version.

1. Le fichier config contient un token permettant d’obtenir les fonds de carte de MapboxGL, le chemin du style de carte à utiliser, ainsi que le router à utiliser pour la génération d’isochrone (voir fiche OpenTripPlanner pour plus de détails à ce sujet) [↑](#footnote-ref-1)