# OVAP

OVAP est l’acronyme pour Outil de Visualisation et d’Aide à la Planification. Elle se veut un visualisateur de différentes données d’achalandage et d’exploitation archivées, en premier lieu les données de carte à puces. Celle-ci sont utilisées afin de produire de générer des paires O-D entre arrêts pour une grande proportion des déplacements effectués sur le réseau (il est impossible de déterminer les lieux de sorties pour les paiements comptants et certains autres déplacements ne possédant pas d’équivalents pendulaires).

En ce moment, l’interface se résume à un visualisateur permettant de sélectionner un ou des arrêts, et de trouver l’origine des voyageurs y ayant fait leur descente ou la destination des voyageurs y ayant effectué leur montée. Les requêtes permettant de faire cet affichage sont paramétrables, notamment en fonction du type de jour, de la date, de l’heure de la journée et de la ligne utilisée.

## Algorithme de génération des paires OD

La figure suivante détaille la liste des sources de données permettant d’exécuter l’algorithme de génération de paires OD. À ce moment, l’algorithme est construit en Access/VBA, obtient les données des différentes sources, génère des paires OD avec une série de requêtes permettant de faire le match entre une validation et un arrêt d’embarquement et de débarquement selon une série d’imputations, avant de renvoyer les données sur le serveur MSSQL planif\_prod.

D:\schemas\ovap_db_schema\algo_cap.png

Jaune : Entrée de planif\_prod

Rouge : Entrée de planif\_works

Vert :Manipulation locale

Bleu : Sortie vers planif\_prod

Les grandes étapes de l’algorithme vont comme suit :

1-Gestion du calendrier. Inclut la gestion des assignations, en plus des relâches scolaires.

2-Obtention des données des tournées (stad\_ref\_course) et des assignations de véhicules (sar\_bus)

3-Obtention des données de sdap\_course pour assigner les embarquements ayant lieu avant les heures de départ. Les étapes 2 et 3 permettent de peupler la table bus\_course, qui sera utilisée dans la suite du processus

4-Union des tables GFI et OPUS (critère pour GFI pour une montée est de 1.65$)

5-Numérotation des validations avec un numéro séquentiel pour la journée

6-Premières et dernières validations se font donner un flag

7-Correction des correspondances et réparation des erreurs de lecteur

8-Filtre des données CAP selon le réseau d'exploitation

9-25 procédures de match en séquence, en ordre de qualité (voir complément de documentation sur les matchs); match de la validation et de la course

10-Application de la chaîne de validation pour les validations ayant un delta de moins de 12 secondes

11-Remplissage des autres tables temporaires (STAD/SDAP/Arrêts valides pour la période)

12-Match des arrêts d'embarquements

13-Match des arrêts de débarquements

## Backend

Les données produites étant stockée sur le serveur planif\_prod, la construction d’une interface de visualisation requiert de rapatrier les données sur la machine où sera installé le serveur hébergeant celle-ci. De plus, la structure des données sur planif\_prod est d’une grande table unique à « plat », c’est-à-dire sans références aux tables sources, par exemple celle contenant les informations sur les arrêts ou les lignes.

Afin de limiter la quantité d’information à stocker, il est plus efficace de redéfinir la structure des données sous la forme de tables relationnelles liées entre elles. Afin de simplifier la mise en place des relations, des identifiants uniques numériques sont créés afin de définir ces relations. Cela permet, par exemple, de lier la table des validations de cartes avec les arrêts uniquement en utilisant cet identifiant plutôt qu’en faisant le « join » sur deux colonnes (assignation et numéro d’arrêt).

Le backend du serveur est développé principalement le SGDBR PostgreSQL/PostGIS, alors que le serveur lui-même fonctionne sous Ruby on Rails. PostGIS présente l’avantage d’avoir de nombreuses fonctions internes permettant de manipuler les données géographiques. Entre autre, cela permet de stocker des objets géographiques comme position des arrêts et ainsi de faire des requêtes sur celles-ci, ou encore d’exporter les données directement en GeoJSON, le format standard de partage d’information géographique sur le web. Cette dernière fonction est d’autant plus appréciable du point de vue des performances qu’elle permet à l’information d’être passée directement du serveur de base de données à l’interface sans avoir à être réinterprétée par le serveur web. La structure complète de la base de données est présente en annexe à ce document.

La plateforme Rails permet quant à elle de programmer bon nombre de tâches de maintenance et de chargement/exportation de données. Entre autres, une tâche permet d’obtenir de planif\_prod des jours uniques ou une série de journées de données. L’exécution de ces processus s’accompagne aussi par la création de surcouches permettant un accès plus rapide aux données, notamment l’ajout d’une table déplacements en plus de la table segments, celle-ci faisant l’agrégation des différents segments pour montrer l’entrée/sortie du réseau d’une personne plutôt que l’entrée/sortie d’un véhicule.

## Interface

Une fois le backend construit, une interface a dû être développée afin de présenter l’information aux usagers. Les données à présenter étant somme toute importantes (>5000 entrées uniquement pour les arrêts), MapboxGL s’est présenté comme le choix le plus durable. La librairie inclut en outre des capacités de présentation pour différents type de couche avec de très bonnes performances, notamment des cartes de chaleur ainsi que la possibilité de présenter des informations sous la forme de carte animée.

Les différentes interactions utilisateurs et communications avec le serveur sont quant à elles gérées avec la bibliothèque jQuery, qui permet des raccourci notamment pour les appels au serveur (ajax) et pour les interactions entre les différents éléments de l’interface. De plus, un grand ensemble de plugin est disponible, simplifiant l’ajout de diverses fonctions (sélecteur de date, d’heure, etc.)

Le principe général de l’interface développée est de permettre l’exploration des paires O-D avec la possibilité de choisir un ou des arrêts d’embarquement ou de débarquement et d’indiquer respectivement les lieux de descente ou de montée pour les segments ou les déplacements. L’interface permet de filtrer les données à afficher selon différentes variables, notamment l’assignation, le type de service, la date et une période de la journée. En outre, une fois les données générées, le menu de droite sert à la fois d’élément de légende pour les graphiques, mais entraîne aussi une mise à jour de la carte, permettant ainsi de filtrer les destinations pour une ou plusieurs lignes.

Finalement, certains outils d’exportations sont aussi disponibles afin d’offrir un meilleur contrôle sur l’affichage de sortie. Il est en effet possible d’exporter les données de carte en format GeoJSON, qui est le standard en cartographie web, mais qui peut aussi aisément être importé dans des logiciels GIS de bureau. Il est aussi possible d’exporter une matrice O-D pour une journée unique dans un format compatible avec le logiciel de simulation Madigas à des fins de simulation de réseau. Finalement, les tableaux et graphiques sont conçus pour être aisément copié vers un tableur, ou encore sauvegardés en image.

## Perspectives

Dans sa forme actuelle, l’application se concentre sur la visualisation et l’interprétation des patrons de déplacements des usagers du RTL. Il est toutefois envisagé d’ajouter d’autres visualisations basées sur les mêmes données. Entre autres, il devrait être possible d’intégrer des visualisations plus axées sur le réseau, tel que des profils de charge. Dans le but de simplifier les processus de requête, il pourrait aussi être intéressant de présenter des outils plus simples pour indiquer des valeurs brutes de montées et de descente par arrêt. Il pourrait aussi être intéressant d’analyser les déplacements d’un point de vue de « chaîne », par exemple en s’attardant plus longuement aux correspondances dans le but d’étudier diverses améliorations pour les infrastructures réseau.

Finalement, il pourrait être intéressant d’intégrer de nouvelles sources de données. Une fois les étapes de visualisation complétées, certaines données obtenues de STAD pourraient en effet être intégrées pour permettre de réutiliser les visualisations et effectuer des comparaisons sur les profils de charges ou le nombre de montée/descente entre les deux sources de données.

