OVAP

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

**Objectif** : Étudier les patrons de déplacements des usagers; accéder rapidement et visuellement à des informations de paires O-D; stocker l’information de façon accessible pour permettre des requêtes plus complexes (profil de charge, correspondances)

**Fonctionnement** : L’application se connecte sur le serveur Planif\_prod pour obtenir les données traitées de carte à puce possédant des informations sur les arrêts de montée et descente. Ces données sont stockées dans un modèle relationnel, qui sert ensuite de base à une application web permettant de construire et d’afficher des requêtes. L’interface web envoie une série de paramètre au serveur (Rails) qui construit une requête (PostgreSQL), et retourne à l’interface des données permettant d’afficher une carte (MapboxGL) et des graphiques. L’interface web est conçue pour être intuitive, alors que le serveur possède un ensemble de commandes permettant de générer, stocker et organiser l’information.

## Commandes serveur (en ligne de commande):

Mettre en place une installation (après installation de msys et de ruby)

* Peupler le fichier database.yml avec les informations sur le serveur
* Peupler le fichier config.yml avec les informations sur le serveur (un exemple est fourni)
* Préparer l’installation des extensions (

bundle install

* Créer la base de données

rake db:create && rake db:migrate

Importer les plus récentes données sur les arrêts, assignations et lignes

rake db:seed

Importer des données du serveur pour une journée précise:

rake import\_cap\_depuis\_achalandage[AAAA-MM-JJ]

rake generer\_dates\_par\_type\_de\_service

Importer des données du serveur pour plusieurs journées consécutives (ou date\_debut et date\_fin répondent au format AAAA-MM-JJ) :

rake import\_cap\_depuis\_achalandage\_en\_lot[date\_debut,date\_fin]

Lancer le serveur ( -b 0.0.0.0 permet l’accès à distance; -p 3000 définit le port)

rails s -b 0.0.0.0 -p 3000

## Rapport d’achalandage

En plus des commandes permettant de générer les données pour la visualisation, l’application comporte aussi des utilitaires embryonnaires visant à produire les rapports d’achalandage mensuels. Il est important de noter que ces rapports ne dépendent pas des données locales, mais bien de celles situées sur le serveur Planif\_Prod. Cette situation implique qu’il n’est pas nécessaire d’exécuter les tâches d’importation pour produire les rapports, mais il est toutefois essentiel de générer les tables distantes avec l’utilitaire Access.

La procédure permet d’obtenir les données distantes, de les stocker localement et de produire les rapports dans un format approprié.

rake import\_mois\_pour\_achalandage[date\_debut,date\_fin]

où date\_debut et date\_fin sont les dates de début et de fin du mois pour lequel le rapport doit être généré, en format AAAA-MM-JJ. Une fois cette importation des données complétée, deux tâches intégrées permettent d’exporter les tableaux des rapports en format CSV

rake rapport\_achalandage\_pour[date]

où date est n’importe quelle date du mois pour laquelle le rapport doit être produit, en format AAAA-MM-JJ

rake rapport\_access\_65[date]

où date est n’importe quelle date du mois pour laquelle le rapport doit être produit, en format AAAA-MM-JJ. Certaines de ces tâches sont des tâches composites d’autres tâches. Celles-ci sont toutes disponibles dans le répertoire lib\tasks dans le fichier achalandage.rake.

Ces tâches restent embryonnaires et à développer. Elles sont dépendantes du fichier de titre dans le répertoire db\seeds\titres.csv, qui doit être mis à jour en début d’année pour refléter les changements sur les titres disponibles, leur tarif et la répartition des revenus entre les diverses agences pour les titres métropolitains.

## Page d’accueil

OVAP contient également une page d’accueil, permettant d’offrir un point d’entrée dirigé vers les autres utilitaires. Cette page est plutôt statique par nature, et ne devrait pas avoir besoin de maintenance régulière. Néanmoins, il est possible de devoir faire des changements mineurs à la vue contenant la définition des liens vers les autres pages. Cette vue, située dans app/views/index/index.html, contient en effet des liens (<a href=></a>) vers les autres pages, et certaines situations, comme par exemple un changement d’adresse IP du poste sur lequel l’application est exécutée, peuvent entraîner le besoin d’y faire des modifications. Dans son état actuel, la page auto-défini l’adresse du poste grâce à du code Ruby, ce qui devrait permettre de ne pas avoir à y effectuer de modifications tant que l’application n’est disponible que sur l’intranet.

La page d’accueil référençant plusieurs autres applications, les ports de ces dernières doivent être définies à même le code. La liste suivante illustre quels ports sont définis par défaut dans l’état actuel de la situation

|  |  |
| --- | --- |
| Application | Port |
| OVAP | 3000 |
| Visualisateur | 3010 |
| OpenTripPlanner | 8880 |
| Carte animée | 8082 |

## Requête complémentaires

L’interface d’OVAP ne fait qu’effleurer les possibilités en termes de visualisation et d’exploitation des données de carte à puce. Si l’interface ne va pas au-delà de la visualisation de paires OD entre arrêts, il reste possible d’effectuer des requêtes directement sur la base de données sur laquelle l’application s’appuie dans le cadre de besoins spécifiques. De tels besoins se sont présentés notamment dans le cadre d’analyses sur les correspondances. Un exemple de requête pour étudier le nombre de départ à un arrêt spécifique uniquement dans le cadre de correspondances :

select

date,

count(\*)

from

segments

where

assignation\_id = 4

and

arret\_depart\_id = 199076

and

sequence\_dans\_deplacement != 1

group by date

Cette commande SQL renvoie le nombre de montées pour l’arrêt possédant l’identifiant 199076, par jour, pour tous les déplacements n’étant pas la première montée dans la séquence.