OSM

La recherche

La recherche est sur `search.maps.ppsfleet.navy`, elle utilise Addok

https://addok.readthedocs.io/en/latest/

C'est installé dans `/srv/osm/env-addok`, le serveur utilise uwsgi dont la config est dans `/srv/osm/addok`. La config général est dans `/etc/addok`

Les données osm: `https://osm13.openstreetmap.fr/~cquest/osm_poi/`

Installation fixes

```
pip install setuptools==57.5.0
```

pour fonctionner avec python3.10, besoin de la dernière version de falcon (3.1.0) (cloner les sources et éditer le requirement.txt)

Le serveur de tuile

Servir les tuiles

Le serveur de tuile utilise openmaptile, installé dans `/srv/www/maps.ppsfleet.navy/tileserver.php`, il utilise des fichiers mbtiles, qui sont eux aussi à la racine.

Le thème

Les thème des cartes se trouvent dans `/srv/www/maps.ppsfleet.navy/tileserver-data`. Les deux utilisés sont:

- osm-bright-gl-style/style-cdn.json (pour /world)
- qwant-basic-gl-style-toulouse/built-style-debug.json (pour /toulouse)

C'est dans ces fichiers .json que le choix du fichier mbtile est definie.

- sources.basemap.tiles
- ___.poi.tiles

Pour eviter le cache, on renomme le fichier mbtile à chaque modif avec dd-mm-yy-increment, on doit donc éditer le style par la suite

Pour eviter des pbolème de mémoire, dans le docker compose, mettre sous postgres: `shm_size: 5gb`

Générer les tuiles

1. Télécharger un fichier osm.pbf depuis https://extract.bbbike.org/ pour une zone custom ou depuis http://download.geofabrik.de/ pour une region ou un pays
2. Editer le schema, openmaptiles.yaml et layers/ ou pas (par default c'est pas si mal)
3. Editer le .env, principalement le `MAX_ZOOM`. Apriori, inutile de le mettre à plus de 14, tout est présent à ce zoom. Et la génération du zoom 16,17,18 prend un temps montrueux (plusieurs jours pour la france)
4. Supprimer le dossier cache, build, clean le dossier data (ne laisser que le .pbf), supprimer `/var/lib/docker/volumes/openmaptiles_pgdata/`
5. Suivre le tuto de openmaptile https://github.com/openmaptiles/openmaptiles:
   - `make destroy-db` # supprime les volumes etc...
   - `make clean` # clean / remove existing build files
   - `make init-dirs` # ??
   - `make all` # ??
   - `make` # generate build files (pareil que make all ??)
   - `make start-db` # start up the database container.
   - `make import-data` # Import external data from OpenStreetMapData, Natural Earth and OpenStreetMap Lake Labels.
   - copy your pbf file to /data
   - `make import-osm` # import data into postgres
   - `make import-wikidata` # import Wikidata
   - `make import-sql` # create / import sql funtions
   - `make generate-bbox-file` # compute data bbox -- not needed for the whole planet
   - `make generate-tiles-pg` # generate tiles (le plus long)

Toulouse se fait en quelques minutes sur alshain

Midi pyrenées se fait en moins d'une heure sur alshain

La france se fait en un nombre certains d'heure, si le max_zoom est pas trop elevé (grand max 16). L'import OSM: 30/35 minutes, Import SQL: 2h30, Génération des tuiles: 17 heures zoom 14

6. Compiler le thème de qwant-maps (je sais plus comment j'avais fait)
   https://github.com/Qwant/qwant-basic-gl-style

Le front-end

Il est dans `/srv/www/maps.ppsfleet.navy/front/{world|toulouse}`, c'est basé sur https://github.com/tjiho/simplestreetmap, avec l'utilisation de maplibre.

TODO: responsive sur telephone

Les itineraires

Todo.

On pourrait se baser sur brouter. Voir aussi https://safecycle.atelier-des-communs.fr/?

ou https://navitia.io/

Ajouter des données

J'ai mis en place pg_tileserv. Il detecte automatiquement les tables postgis, et les transforme en tuile.

Il a son home dans /srv/pg-tileserv et sa conf dans /etc/pg-tileserv

Générer des tuiles "raster" en node-js

```
git clone https://github.com/maplibre/maplibre-gl-native
```

```
dnf groupinstall "Development Tools" "Development Libraries"
```

```
dnf install glfw-devel freeglut-devel libuv-devel libXrandr-devel libjpeg-devel zlib-devel libpng-devel g++ mesa-dri-drivers xorg-x11-server-Xvfb
```

A la racine du repo:

```
cmake . -B build
```

```
vim vendor/benchmark/src/benchmark_register.h
> #include <vector>
> #include <limits>

vim build/CMakeCache.txt
> CMAKE_CXX_FLAGS:STRING=-pthread
```

```
cmake --build build
```

ca va planter en essayant de compiler la partie nodejs (aux alentours de 90%), c'est pas grave tant que ca plante pas avant

Générer l'image

```
run.sh
> /srv/osm/maplibre-gl-native/build/bin/mbgl-render --style=https://maps.ppsfleet.navy/tileserver-data/qwant-basic-gl-style-toulouse/built-style-debug.json --output=test.png --width=512 --height=256 --lon=1.4436 --lat=43.6042 --zoom=13
```

```
xvfb-run ./run.sh
```

Idée theme

Révision #26
Créé 31 octobre 2021 23:11:35 par tjiho
Mis à jour 27 septembre 2023 15:35:48 par tjiho