GeoNature-citizen Documentation

Version dev

lpofredc

févr. 12, 2020

Contenu:

CHAPITRE 1

Installation de GeoNature-citizen

1.1 Prérequis

Application développée et installée sur un serveur Debian 9.

Ce serveur doit aussi disposer de :

- sudo (`apt-get install sudo`)
- un utilisateur (`monuser` dans cette documentation) appartenant au groupe `sudo` (pour pouvoir bénéficier des droits d'administrateur)

1. Créer un utilisateur Debian :

```
adduser nom_utilisateur (geonatadmin) entrez un mot de passe (****)
usermod -aG sudo nom_utilisateur (geonatadmin)
su - nom_utilisateur (geonatadmin)
```

2. Vérifier que l'utilisateur est correctement créé :

```
sudo -l (entrez un mot de passe) : vérifier que ALL
sudo whoami : ok si on peut faire un sudo
```

3. Changer la locale en fr (il faut être root) :

```
sudo dpkg-reconfigure locales
```

1.2 INSTALLATION DE TAXHUB

Pour plus de détails, lien officiel pour l'installation de TaxHub : https://taxhub.readthedocs.io/fr/latest/

Configurer le serveur : https://taxhub.readthedocs.io/fr/latest/serveur.html#installation-et-configuration-du-serveur

Configurer PostgreSQL : https://taxhub.readthedocs.io/fr/latest/serveur.html#installation-et-configuration-de-posgresql

Configuration et installation de l'application : https://taxhub.readthedocs.io/fr/latest/installation.html

notes

— Bien vérifier de ne pas être en `root` :

```
su - nom_utilisateur (geonatadmin)
```

— Pour avoir les caractéristiques de votre instance :

```
lsb_release -a
uname -a
```

1.3 INSTALLATION DE GEONATURE-CITIZEN

Etape 1 : Configurer PostgreSQL :

notes

— Cette étape n'est nécessaire que si TaxHub n'est pas installé

— voir init_launch_db.rst : https://github.com/PnX-SI/GeoNature-citizen/blob/master/docs/devs/init_launch_db.rst

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ stretch-pgdg main" >>/
↪etc/apt/sources.list.d/postgresql.list'
sudo wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-
↪key add
sudo apt update
sudo apt install postgresql-10 postgresql-10-postgis-2.5 postgresql-10-postgis-2.5-
↪scripts git
sudo -u postgres createuser -s -E -P dbuser (geonatadmin) (Entrez le password)
sudo -u postgres createdb -E -E UTF8 -O dbuser (geonatadmin) dbname (geonature2db)
```

notes

— ls /etc/init.d/ : pour lister les services

— sudo service restart postgresql : vérification

Etape 2 : Installer python3, pip et virtualenv :

```
python3 -m pip install --upgrade --user virtualenv
sudo apt install python3-pip
```

installer virtualenv :

```
export PATH=/home/username/.local/bin:$PATH (username = geonatadmin)
echo $PATH
virtualenv -p /usr/bin/python3 venv
source venv/bin/activate
```

Etape 3 : Installation du backend et de la base des données :

notes

— init_launch_backend.rst

— creation referentiel géo

— voir : https://github.com/PnX-SI/GeoNature-citizen/blob/master/docs/devs/init_launch_db.rst

Cloner le dépôt Github de GeoNature-citizen

```
sudo apt install git
git clone name (citizen)
git checkout branch_name
cd citizen/backend
```

Création du référentiel des géométries communales :

```
wget https://github.com/PnX-SI/GeoNature/raw/master/data/core/public.sql -P /tmp
wget https://github.com/PnX-SI/GeoNature/raw/master/data/core/ref_geo.sql -P /tmp
wget https://github.com/PnX-SI/GeoNature/raw/master/data/core/ref_geo_municipalities.
↪sql -P /tmp

psql -d geonature2db -h localhost -p 5432 -U geonatadmin -f /tmp/public.sql
sed 's/MYLOCALSRID/2154/g' /tmp/ref_geo.sql > /tmp/ref_geo_2154.sql
psql -d geonature2db -h localhost -p 5432 -U geonatadmin -f /tmp/ref_geo_2154.sql
```

Pour restaurer en cas de besoin :

```
psql -d geonature2db -h localhost -U geonatadmin -f ~/citizen_taxhub_l_areas_dump.sql
if [ ! -f '/tmp/communes_fr_admin_express_2019-01.zip' ]
then
   wget  --cache=off http://geonature.fr/data/ign/communes_fr_admin_express_2019-01.
↪zip -P /tmp
else
   echo "/tmp/communes_fr_admin_express_2019-01.zip already exist"
fi
unzip /tmp/communes_fr_admin_express_2019-01.zip -d /tmp/
psql -d geonature2db -h localhost -p 5432 -U geonatadmin -f /tmp/fr_municipalities.sql
psql -d geonature2db -h localhost -p 5432 -U geonatadmin -c "ALTER TABLE ref_geo.temp_
↪fr_municipalities
OWNER TO geonatadmin;"
psql -d geonature2db -h localhost -p 5432 -U geonatadmin -f /tmp/ref_geo_
↪municipalities.sql
psql -d geonature2db -h localhost -p 5432 -U geonatadmin -c "DROP TABLE ref_geo.temp_
↪fr_municipalities;"
```

Lancement du backend pour générer les schémas :

En mode debug :

```
export FLASK_ENV=development; export FLASK_DEBUG=1; export FLASK_RUN_PORT=5002;␣
↪export FLASK_APP=wsgi;
python -m flask run --host=0.0.0.0
```

Enregistrement du module principal :

```
insert into gnc_core.t_modules values (1, 'main', 'main', 'main', NULL, false, '2019-
↪05-26 09:38:39.389933', '2019-05-26 09:38:39.389933');
```

Enregistrement d'un programme exemple

```
psql -d geonature2db -h localhost -p 5432 -U geonatadmin -c "INSERT INTO gnc_core.t_␣
↪programs VALUES (1, 'Au 68', 'inventaire  du 68', 'desc', NULL,    NULL,   1,     ␣
↪1,      't',
↪'0106000020E6100000010000000103000000010000000500000010000070947C154042CA401665A545400
↪',    '2019-05-26 09:38:39.389933', '2019-05-26 09:38:39.389933');"
```

Etape 4 : éditer le fichier de config :

notes

— voir : https://github.com/PnX-SI/GeoNature-citizen/blob/dev/docs/devs/config_files.rst

```
cd ../config
editer les paramètres dans default_config.toml

-SQLALCHEMY_DATABASE_URI :"postgresql+psycopg2://
↪dbuser(geonatadmin):password(***)@127.0.0.1:5432/dbname(geonature2db)"
-URL_APPLICATION : 'https://ipserveur:4200/'
-API_ENDPOINT : 'https://ipserveur:5002/api'
-API_TAXHUB : 'http://ipserveur/taxhub/api/'

- Pour configurer du serveur Smtp renseigner les différents paramètres de votre␣
↪serveur
dans la partie [MAIL] ( MAIL_HOST,MAIL_PORT ..)
 # La partie [RESET_PASSWD] correspond à la configuration du texte du mail a envoyé␣
↪pour la restauration
  du mot de passe oublié
 # La partie [CONFIRM_EMAIL] correspond à la configuration du texte du mail a envoyé␣
↪pour l'activation
 du compte et la confirmation de l'adresse mail de l'utilisateur
```

Etape 5 : configuration des badges :

— voir : https://github.com/PnX-SI/GeoNature-citizen/blob/dev/docs/devs/badges.rst

Etape 6 : configuration du supervisor :

```
/etc/supervisor/conf.d/geonature-citizen-service.conf
[program:citizen]
command = /home/geonatadmin/citizen/backend/start_gunicorn.sh
autostart=true
autorestart=true
stdout_logfile = /var/log/supervisor/citizen.log
redirect_stderr = true
```

Etape 7 : Installation du frontend :

notes

— voir : https://github.com/PnX-SI/GeoNature-citizen/tree/dev/docs/devs/init_launch_frontend.rst

```
cd citizen/frontend/
nvm use --lts  # Now using node v10.16.0 (npm v6.9.0)
si pas installé : nvm install --lts (remplacer lts par la dernière version)
cp -v src/assets/badges_* ../media/
```

Editer la conf :

```
cp src/conf*.ts.sample src/conf/  # ajuster la conf
# copier le template css alternatif
cp src/custom/custom.css.template src/custom/custom.css
# Pour configurer le lien externe de la fiche détaillée de l'espèce, éditer l'entrée␣
↪suivante:
details_espece_url: "<url_inpn_or_atlas>/cd_nom/" // !! garder bien le cd_nom/ dans l
↪'url
```

Lancer le front :

```
npm run start -- --host=0.0.0.0
```

Ré-génération des locales après modification de l'UI :

```
for lang in 'fr' 'en'; do npm run -- ng xi18n --output-path locale --out-file _
↪messages.${lang}.xlf --i18n-locale ${lang}; done
```

CHAPITRE 2

Configuration des badges

Types de récompense :
- seniority : ancienneté d'inscription sur la plateforme
- all_attendance : nombre d'observations totales réalisées sur la plateforme
- program_Attendance : nombre d'observations totales réalisées par programme
- recognition : identification d'espèces

niveaux de récompense :
- min_obs : nombre d'observations minimum pour obtenir le badge
- min_date : date minimum pour obtenir le badge

Configuration de récompense :
- type : type de récompense (exemple Seniority)
- reward_label label de la récompense (a afficher du coté forntend)
- id_program : identifant du programme (0 pour les récompenses géneral)
- badges : tableau des badges par type de récompense

Pour les récompenses de type recognition il faut renseigner soit la classe ou la famille du taxref

Le fichier à configurer est le badges_config.py situé dans le dossier config

Ex : https://github.com/PnX-SI/GeoNature-citizen/blob/dev/config/badges_config.py

notes Pour plus d'informations voir : https://github.com/PnX-SI/GeoNature-citizen/issues/7

CHAPITRE 3

Installer GeoNature-citizen en mode développement

Avertissement : GeoNature-citizen nécessite l'installation préalable de TaxHub > https://taxhub.readthedocs.io/fr/latest/

3.1 Contribuer avec GitHub

Avertissement : Aucun commit n'est réalisé directement sur le dépot principal du projet (https://github.com/PnX-SI/GeoNature-citizen). Pour contribuer, il est nécessaire de faire un *fork* du projet, de travailler sur ce fork et de proposer des mises à jour du dépot principal par *pull request*.

3.1.1 Faire un fork du projet

Tout est ici

3.1.2 Cloner le projet

Dans un terminal :

```
$ git clone git@github.com:YOUR_NAME/GeoNature-citizen.git

Cloning into `GeoNature-citizen`...
remote: Counting objects: 10, done.
remote: Compressing objects: 100% (8/8), done.
remove: Total 10 (delta 1), reused 10 (delta 1)
Unpacking objects: 100% (10/10), done.
```

Récupérer les mises à jour du dépot principal

Dans un terminal, dans le dossier cloné :

```
$ git remote add upstream git@github.com:PnX-SI/GeoNature-citizen.git
```

Pour vérifier que votre clone local puisse suivre votre dépot (*origin*) et le dépot principal (*upstream*) :

```
$ git remove -v

origin      git@github.com:YOUR_NAME/GeoNature-citizen.git (fetch)
origin      git@github.com:YOUR_NAME/GeoNature-citizen.git (push)
upstream    git@github.com:PnX-SI/GeoNature-citizen.git (fetch)
upstream    git@github.com:PnX-SI/GeoNature-citizen.git (push)
```

Créer votre propre branche de développement

Pour créer votre branche de développement, dans un terminal :

```
$ git checkout -b dev_mabranche
```

3.2 Renseigner les fichiers de configuration

3.2.1 Modifier le fichier de configuration

Côté backend

Les fichiers de configuration sont dans le dossier `config`. Le fichier à modifier est default_config.toml. Le fichier utilisé par GeoNature-citizen est default_config.toml. Il peut-être créé en copiant le fichier `default_config.toml.example` vers `default_config.toml` :

```
$ cp default_config.toml.example default_config.toml
```

Editez alors les différents paramètres de ce fichier.

```
# Database
SQLALCHEMY_DATABASE_URI = "postgresql+psycopg2://geonatuser:monpassachanger@127.0.0.
↪1:5432/geonaturedb"
SQLALCHEMY_TRACK_MODIFICATIONS = false

# JWT Auth
JWT_SECRET_KEY = 'jwt-secret-string'
JWT_BLACKLIST_ENABLED = true
JWT_BLACKLIST_TOKEN_CHECKS = ['access', 'refresh']

# Application

appName = 'GeoNature-citizen'                                    # Application name in the␣
↪home page
DEFAULT_LANGUAGE = 'fr'

# Nom du zonage du portail
```

(suite sur la page suivante)

(suite de la page précédente)

```
PORTAL_AREA_NAME = 'zonage'

DEBUG = true

URL_APPLICATION = 'http://url.com/gncitizen'          # Replace my_url.com by your␣
↪domain or IP
API_ENDPOINT = 'http://url.com/gncitizen/api:API_PORT'         # Replace my_url.com by␣
↪your domain or IP
API_PORT = 5002 # 5000 déjà utilisé par taxhub
API_TAXHUB ='http://127.0.0.1:5000/api/'

SESSION_TYPE = 'filesystem'
SECRET_KEY = 'MyS3cr3tK3y'
COOKIE_EXPIRATION = 7200
COOKIE_AUTORENEW = true
TRAP_ALL_EXCEPTIONS = false
HTTPS = false
MEDIA_FOLDER = 'static/medias'
# File
# BASE_DIR = os.path.abspath(os.path.dirname(__file__))
UPLOAD_FOLDER = 'static/medias'

# Front end configuration
[FRONTEND]
    PROD_MOD = false
    DISPLAY_HEADER = false
    DISPLAY_FOOTER = false
    MULTILINGUAL = false

[MAILERROR]
    MAIL_ON_ERROR = false
    MAIL_HOST = 'host mail'
    HOST_PORT = host mail port
    MAIL_FROM = 'Email from'
    MAIL_USERNAME = 'email username'
    MAIL_PASS = 'email to'
    MAIL_TO = 'email to'


# API flasgger main config
[SWAGGER]
    title = 'GeoNature-Citizen API'
    version = 'x.x.x'
    produces = ["application/json"]
    consumes = ["application/json"]
```

Côté frontend

Le fichier de configuration du frontend se trouve dans le dossier `./frontend/src/conf`

Le fichier à créer est `app.config.ts` Il peut-être créé en copiant le fichier `app.config.ts.sample` vers `app.config.ts` :

```
$ cp default_config.toml.example default_config.toml
```

Editez alors les différents paramètres de ce fichier.

```
export const AppConfig = {
    "appName": "GeoNature-citizen",
    "API_ENDPOINT":"http://localhost:5002/api",
    "API_TAXHUB":"http://localhost:5000/api",
    "FRONTEND":{
        "PROD_MOD":true,
        "MULTILINGUAL":false,
        "DISPLAY_FOOTER": true,
        "DISPLAY_TOPBAR": false,
        "DISPLAy_SIDEBAR": true
    },
    "URL_APPLICATION":"http://localhost:4200"
}
```

3.3 Configurer et installer la base de données

GeoNature-citizen s'appuie sur le serveur de base de données spatiales PostgreSQL et son extension spatiale PostGIS.

3.3.1 Installer le serveur

Pour installer le serveur de base de données, suiviz les instructions du site officiel PostgreSQL Downloads :

Concrètement, sur Debian stretch :

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ stretch-pgdg main" >> /
↪etc/apt/sources.list.d/postgresql.list'
sudo wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-
↪key add -
sudo apt update
sudo apt install postgresql-10 postgresql-10-postgis-2.5 postgresql-10-postgis-2.5-
↪scripts git
```

3.3.2 Configurer la base de données

Création du role principal

Pour créer la base de données spatiale. On considèrera ici que l'utilisateur de la base de données sera `dbuser`, renseignez alors le mot de passe de l'utilisateur lorsqu'il vous sera demandé :

```
sudo -u postgres createuser -s -E -P dbuser
```

Créez la base de données, ici nommée `geonaturedb` appartenant à l'utilisateur `dbuser` :

Création de la base de données et des extensions

```
sudo -u postgres createdb -s -E UTF8 -O dbuser geonaturedb
```

Activez les extensions `postgis` pour la gestion des données spatiales et `uuid-ossp` pour la gestion des uuid. Seul un superutilisateur peut activer les extensions (ici, l'utilisateur `postgres`, installé par défaut) :

```
sudo -u postgres psql geonaturedb -c 'create extension postgis; create extension
↪"uuid-ossp";'
```

Votre serveur de base de données est maintenant opérationel.

3.4 Configurer et lancer le backend

3.4.1 Installer l'environnement virtuel python

La création de l'environnement virtuel python3 nécessite `virtualenv` ou `pyenv` ou tout autre outil équivalent (ex : pyenv) :

```
cd backend
sudo apt install python3-pip
python3 -m pip install --upgrade --user virtualenv
export PATH=/home/geonatadmin/.local/bin:$PATH
virtualenv -p /usr/bin/python3 venv
```

L'activation de cet environnement se fait avec la commande suivante :

```
source venv/bin/activate
```

Et l'installation des librairies nécessaires à GeoNature-citizen avec la commande suivante :

```
python3 -m pip install -r requirements.txt
```

3.4.2 Lancer le backend

Pour lancer l'application Backend, il suffit d'éxécuter les commandes suivantes depuis l'environnement virtuel python :

```
cd backend
source venv/bin/activate

cd ../config
cp default_config.toml.example default_config.toml

python -m wsgi.py
# debug mode
# export FLASK_ENV=development; export FLASK_DEBUG=1; export FLASK_RUN_PORT=5002;␣
↪export FLASK_APP=wsgi; python -m flask run --host=0.0.0.0
```

Vous pouvez alors aller sur la page de documentation de l'API à l'adresse suivant `http://VOTRE_HOTE:5002/apidocs`, en local, ce sera http://localhost:5002/apidocs.

3.5 Configurer et lancer le frontend

3.5.1 Installer l'environnement virtuel NodeJS avec nvm

L'installation de `nvm` se fait en suivant les instructions du dépot principal de l'outil nvm par creationix creationix/nvm.

Une fois l'environnement installé, installer la dernière version stable de `nodejs` :

```
nvm install v10.16
```

Pour utiliser cette version :

```
nvm use v10.16
```

Installer angular CLI (version LTS 6) et les dépendances requises :

```
npm install -g @angular/cli@v8-lts
npm install
```

3.5.2 Lancer du frontend

Vous pouvez lancer le frontend dans deux modes :

En mode développement et client-side rendering :

```
ng serve
```

En mode Server Side Rendering, optimisé pour le SEO et réservé aux robots d'indexation :

```
npm run build:ssr && npm run serve:ssr
```

3.5.3 Gestion du Server Side Rendering

Le SSR a été intégré au projet à partir de la commande :

```
npm run ng add @nguniversal/express-engine --clientProject frontend
```

NB : L'intégration Leaflet.MarkerCluster a nécessité de déclarer une variable globale `L` et d'y importer Leaflet ; c'est dans le script `server.ts`.

Les modules `BrowserTransferState` et `ServerTransferState` importés, nous avons créé un couple `{clé: valeur}` pour être transféré du serveur au client.

La clé est créée avec la fonction factory makeStateKey :

```
const PROGRAMS_KEY = makeStateKey("programs");
```

Le transfert d'état s'effectue avec accesseur et mutateur :

```
this.programs = this.state.get(PROGRAMS_KEY, null as any);
if (!this.programs) {
  /*
    code exécuté côté serveur Node, express
    qui effectue donc un appel à l'API de GN-Citizen
    et génère une capture d'état
  */
```

(suite sur la page suivante)

(suite de la page précédente)

```
  this.state.set(PROGRAMS_KEY, programs as any);
} else {
  /*
    code exécuté côté présentation qui consomme l'état "cristallisé"
    transféré depuis le serveur.
  */
}
```

La redirection de port pourrait se faire au niveau du serveur web / reverse proxy, avec un filtre sur l'entête de requête `User-Agent`

3.5.4 Gestion de l'internationalisation (i18n)

La fonctionnalité i18n a été intégrée selon la recette originale.

L'interface est paramétrée par défaut en langue française.

Si l'on souhaitait la servir en langue anglaise :

```
npm run ng serve -- --configuration=en
```

La stratégie en cas de traduction manquante est de faire remonter une erreur.

(Ré)génération des fichiers de traduction :

```
npm run -- ng xi18n --output-path locale --out-file _messages.fr.xlf --i18n-locale fr
```

```
npm run -- ng xi18n --output-path locale --out-file _messages.en.xlf --i18n-locale en
```

Les fichiers de traduction se retrouvent dans le répertoire `frontend/src/locale`.

Les copier en `messages.fr.xlf` et `messages.en.xlf` après édition (mon approche est de les mettre à jour depuis un éditeur de différence).

Génération du rendu SSR dans le context de l'i18n :

La commande suivante permet de générer un rendu SSR multilingue et le servir en langue française.

```
npm run build:i18n-ssr && npm run serve:ssr
```

3.5.5 Déploiement

Préparer la distribution avec :

```
npm run ng build -- --prod
```

ou :

```
npm run ng build -- --configuration=en --prod
```

pour une version en langue anglaise.

Tout est contenu dans le répertoire `frontend/dist`, qu'il faut copier sur la plateforme acceuillant le service.

3.5.6 Annexe :

Exemple de fichier de configuration serveur Apache2 :

`/etc/apache2/sites-enabled/citizen.conf`

```
# Configuration GeoNature-citizen
Alias /citizen /home/utilisateur/citizen/frontend/dist/browser

<Directory /home/utilisateur/citizen/frontend/dist/browser>
  Require all granted
  AllowOverride All

  <IfModule mod_rewrite.c>
      Options -MultiViews

      RewriteEngine On
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteCond %{REQUEST_FILENAME} !-f
          RewriteRule ".*" "index.html" [QSA,L]
  </IfModule>

</Directory>
<Location /citizen/api>
  ProxyPass http://127.0.0.1:5002/api
  ProxyPassReverse  http://127.0.0.1:5002/api
</Location>
```

Suivi des journaux d'évenements et d'erreurs :

Backend :

```
tail -f /var/log/supervisor/citizen.log
```

Gunicorn (option de gestion de processus pour lancer le backend) :

```
tail -f ~/citizen/var/log/gn_errors.log
```

Apache :

```
sudo tail -f /var/log/apache2/{error,access,other_vhosts_access}.log
```

Utiliser PgAdmin pour la gestion de la BDD distante (production) :

`~/.ssh/config`

```
Host nom_du_raccourci
Hostname son_addresse_ip
```

(suite sur la page suivante)

(suite de la page précédente)

```
User mon_user
LocalForward 5433 localhost:5432
```

Se logguer en SSH (`ssh nom_du_raccourci`) sur l'hôte distant va opérer une redirection de port et rendre la BDD distante accessible sur le port local `5433` pour un client PostgreSQL.

Il suffit alors d'ajuster les paramètres de `psql` en CLI ou ceux de l'assistant de configuration de PgAdmin pour son interface graphique.

CHAPITRE 4

Sommaire

— genindex
— modindex
— search