# HaitiWater Documentation

*Version beta 1*

**Céline Deknop, Adrien Hallet, Sébastien Strebelle**

**nov. 08, 2019**

# Table des matières

**1 Table des matières** **3**
1.1 Documentation destinée aux developpeurs . . . . . 3
1.2 Documentation destinée aux utilisateurs . . . . . 10

TODO Test du webhook 2

CHAPITRE 1

# Table des matières

## 1.1 Documentation destinée aux developpeurs

### 1.1.1 Démarrage rapide

#### Installation

**Les systèmes d'exploitation suivants ont été testés fonctionnels :**

- Windows 7, 10
- Ubuntu 16.04, 18.04

#### Prérequis

**Python 3.x**

**Windows**

- Télécharger la dernière version (3.x) de Python.
- Lancer l'exécutable
- Cocher la case : `Add Python 3.x to PATH`
- Lancer l'installation par défaut : `Install Now`

**Linux**

- Installer la dernière version de Python 3.x avec le gestionnaire de paquets de votre distribution
- Vérifier que la bonne version est installée :

```
$ python3 --version
```

**GDAL - Librairie géographique**

**Windows**

- Il existe de nombreuses manières d'installer GDAL. La plus simple est d'utiliser une distribution géospatiale comme OSGEO4W

**Linux**

— Téléchargez et installez les binaries GDAL pour votre distribution (Exemple Ubuntu)

**PostgreSQL 9.4 ou supérieure**

**Windows**

— Télécharger PostgreSQL (v9.4+).
— Lancer l'exécutable et suivre l'installation avec les valeurs par défaut jusque . . .
— Sélectionner un mot de passe (à retenir, important !)
— Continuer l'installation avec les valeurs par défaut
— Lancer « StackBuilder » à la fin de l'installation lorsqu'il le propose
— Sélectionner la version PostgreSQL que vous avez choisie
— Cocher `Categories>Spatial Extensions>PostGIS 2.x` (sélectionner la dernière version adaptée à votre installation PostgreSQL 32/64bits)
— Terminer l'installation avec les valeurs par défaut

**Linux**

— PostgreSQL est installé par défaut

**PostGIS**

— La méthode d'installation peut varier selon la version de PostgreSQL et le système d'exploitation. Si StackBuilder n'a pas été exécuté, visitez _PostGIS

**Environnement de développement python**

— Installer virtual environment via l'outil `pip` installé par défaut avec Python 3.x

```
$ pip install virtualenv
```

### Créer la base de données

— Vous avez besoin de créer un superutilisateur avec mot de passe. Référez-vous à des tutoriels pour savoir comment faire précisément sur votre système d'exploitation.
— Déclarez cette nouvelle base de données en tant que base de données géographique PostGIS

```
$ psql -U <nom_superutilisateur_choisi>
$ CREATE DATABASE haitiwater;
$ \connect haitiwater
$ CREATE EXTENSION postgis;
```

### Installer le projet

**Cloner le repository GitHub**

— Ouvrir un terminal (Windows : cmd.exe) à l'emplacement désiré pour le projet
— Cloner le repository

```
$ git clone https://github.com/AdrienHalletUCL/HaitiWater
```

— Naviguer dans l'arborescense jusque

```
../HaitiWater/code
```

— Créer un environnement virtuel Python (il permettra d'isoler l'installation, empêchant des conflits avec d'éventuels autres projets)

```
../HaitiWater/code $ virtualenv env
```

### Configurer le projet

Configurez l'accès à la base de données dans le fichier `HaitiWater/code/haitiwater/haitiwater/settings.py`

```
DATABASES = {
  'default': {
      'ENGINE': 'django.contrib.gis.db.backends.postgis',
      'NAME': 'haitiwater',
      'HOST': 'localhost',
      'PORT': '<PORT>',
      'USER': '<SUPERUSER>',
      'PASSWORD': '<PASSWORD>',
  }
}
```

**Où**

- <PORT> est le numéro de port utilisé par votre serveur PostgreSQL (5432 par défaut)
- <SUPERUSER> est le nom de superutilisateur choisi
- <PASSWORD> est le mot de passe choisi pour le superutilisateur

### Lancer le projet

**En vous positionnant au chemin (dernier laissé précédemment) `../HaitiWater/code` :**

- **Activer l'environnement virtuel Python** Windows

  ```
  env\Scripts\activate
  ```

  Linux

  ```
  $ source env/bin/activate
  ```

- Naviguer jusqu'au projet

  ```
  $ cd haitiwater
  ```

- Installer les dépendances

  ```
  $ pip install -r requirements.txt
  ```

  *Note : sous Windows, il est possible que la dépendance GDAL ne puisse s'installer. Il suffit de la supprimer de la liste et de l'installer manuellement (voir ci-dessus).*

- Exporter le schéma de la base de données

  ```
  $ python3 manage.py makemigrations
  $ python3 manage.py migrate
  ```

- Importer des données d'exemple (optionnel)

  ```
  $ python3 manage.py loaddata initial_data.json
  ```

- Créer les tables virtuelles dans la base de données. Uutiliser l'accès à la base de données décrit dans le fichier `settings.py` et rentrer le mot de passe <PASSWORD> si/quand il vous est demandé.

  ```
  $ psql -p <PORT> -U <SUPERUSER> -d haitiwater -f views.sql
  ```

- Lancer le serveur

```
$ python3 manage.py runserver
```

*Note : sous Windows, supprimez les "/" (slash) après "static-common/images", "static" et "static-common" aux lignes 140, 186 et 191 du fichier settings.py.*

### Structure du projet

HaïtiWater est un projet Django et utilise l'architecture MVC.

### Arborescence Générale

```
HaitiWater/code/haitiwater
├── apps
├── haitiwater
├── static-common
├── templates
├── initial_data.json
├── manage.py
└── requirements.txt
```

### haitiwater - Fichiers Django

`manage.py` est le fichier à partir duquel vous interagissez avec Django.

```
HaitiWater/code/haitiwater/haitiwater
├── settings.py
├── urls.py
└── wsgi.py
```

**Le dossier `haitiwater` contient les fichiers de base de Django, tout particulièrement :**

- `settings.py` : contient les informations de connexion aux services (base de données, serveur mail, serveur hôte, etc)
- `urls.py` : contient les règles de résolution URL de base. Les pages menant à des modules de l'application sont redirigées vers des fichiers `urls.py` propres à chaque module.

### apps - Modules

Les applications continennet les différents modules d'HaïtiWater.

```
Haitiwater/code/haitiwater/apps
├── administration
├── api (communications AJAX serveur/client)
├── authentication (utilisateurs)
├── consumers (consommateurs)
├── dashboard (page d'accueil)
├── financial (finances des consommateurs)
├── log (logging des actions)
├── offline (quand l'application est hors-ligne)
├── report (rapports mensuel et technique
├── utils (utilitaires du serveur)
├── water_network (réseau de distribution d'eau potable)
└── zone_management (gestion de zone)
```

Chaque module a une structure de type

```
module
├── classes
├── migrations
├── static
├── templates
├── test
├── __init__.py
├── admin.py
├── apps.py
├── models.py
├── tests.py
├── urls.py
└── views.py
```

Veuillez vous référer à la documentation Django pour comprendre la structure générale des applications [1]

### static-common - Fichiers statiques généraux

```
Haitiwater/code/haitiwater/apps
├── images
├── javascripts
├── stylesheets
└── vendor
```

- `images` contient les ressources graphiques du serveur (favicon, logo)
- `javascripts` contient les javascripts réutilisés à travers l'application. Les scripts utilisés par une application sont dans le dossier ´´static´´ de l'application (module) correspondante.
- `stylesheets` contient les fichiers CSS
- `vendor` contient les librairies utilisées par l'application en front-end.

*La totalité des librairies utilisées par l'application devrait être servie par le serveur et non pas par des CDN externes afin d'optimiser les téléchargements.*

### templates - Gabarits généraux

Le fichier `templates` contient les gabarits Django réutilisés à travers l'application. On y trouve les menus, graphes et le fichier `base.html` étendu par tous les modules de l'application.

### Librairies

HaïtiWater utilise plusieurs librairies pour gérer, traiter et afficher les données utilisées par l'application. Cette section vous décrit les librairies utilisées et leur(s) rôle(s) dans l'application.

### Front-End

Les librairies du front-end sont situées dans le dossier `HaitiWater/code/haitiwater/static-common/vendor`. Il est important d'importer les librairies localement (sans import provenant de CDNs). Cela permet d'optimiser la connexion en diminuant le nombre de résolutions DNS, de garantir la disponibilité des librairies et de fixer leur comportement.

1. https://docs.djangoproject.com/fr/2.1/intro/overview/

### Bootstrap 3

Bootstrap 3 est une librairie/framework HTML/CSS/JS destinée à la création de sites internets responsives et compatibles mobiles/desktops. Nous utilisons la version 3 pour sa grande compatibilité avec d'autres librairies, sa documentation complète et l'aide qui peut être aisément trouvée sur les forums d'aide.

### Bootstrap - Datepicker

Bootstrap Datepicker permet d'utiliser des fenêtres modales pour sélectionner une date. Datepicker est utilisé dans les tables utilisant la périodicité pour choisir le mois pour lequel les données sont affichées.

### Bootstrap - Multiselect

Bootstrap Multiselect permet de créer des composants de type « select » permettant la sélection multiple, la recherche, etc . . . *Note : Multiselect a été remplacé durant le développement par la librairie Select2, plus performante et permissive. Préférez son utilisation.*

### Bootstrap - Wizard

Bootstrap Wizard permet de gérer les pages de type « wizard » (step-by-step). L'application utilise ce wizard dans le rapport mensuel pour accompagner l'utilisateur et lui donner une vue plus simple du formulaire.

### FontAwesome

FontAwesome est utilisée dans sa version gratuite pour ses icônes SVG. L'application utilise ces icônes en tant que présentation / référence rapide de l'information (e.g. menus) ou en tant que boutons (e.g. tables)

### Fonts Google

Import de polices Google CSS pour l'application. La seule police d'écriture importée/utilisée est "Open Sans".

### JQuery

JQuery est une librairie JavaScript permettant la manipulation du DOM très aisément. JQuery devient de plus en plus obsolète par les nouvelles fonctionnalités des navigateurs récents (plus rapides), mais est utilisée dans notre application pour permettre la compatibilité avec plus de navigateurs et librairies.

### JQuery - DataTables

DataTables est une librairie permettant d'utiliser des tables dynamiques, qui ont une forte dépendance à JQuery. Les DataTables sont utilisées dans l'application pour présenter les données via des requêtes AJAX à l'API du serveur.

### Leaflet

Leaflet est une librairie permettant d'inclure une carte interactive que nous utilisons pour présenter l'information dans le module du Système d'Information Géographique.

#### Magnific Popup

Magnific Popup permet d'afficher des lightbox et boîtes de dialogue aisément.

#### Modernizr

Modernizr est une collection de features permettant d'augmenter la responsiveness de l'application et d'adapter l'interface de l'application au support.

#### Morris

Morris est une bibliothèque graphique permettant la réalisation de graphes simples.

#### NanoScroller

NanoScroller implémente une barre de navigation (type ascenseur) uniforme à travers les différents navigateurs.

#### PNotify

PNotify est utilisé pour les toasts (coin supérieur droit) notifiant des différentes actions effectuées par le serveur (ajout, erreur).

#### Select2

Select2 est une alternative à bootstrap-multiselect plus avancée et performante.

### Back-End

#### Django

Django est la librairie principale de l'application. Elle donne la structure du serveur et son fonctionnement principal. Nous utilisons plusieurs extensions à django :

- *auth* nous permet de gérer des groupes d'utilisateurs et leurs permissions.
- *sessions* nous permet de gérer des sessions d'utilisation de l'application.
- *messages* nous permet d'afficher des notifications à l'utilisateur.
- *staticfiles* nous permet de renvoyer des fichiers statiques par le serveur.
- *gis* nous permet de gérer des données géographiques.

#### Django REST framework

Django REST framework est une librairie permettant de créer une API web au sein de notre serveur.

#### Django compressor

Django compressor est une librairie permettant de compresser plusieurs fichiers statiques comme des scripts javascripts en un seul fichier.

### Django Bootstrap 3

Django Bootstrap 3 est une librairie permettant d'utiliser des fonctionnalités de bootstrap dans des templates django.

### DateUtil

DateUtil est une librairie rajoutant plusieurs fonctions utiles sur les dates. Elle est utilisée ici pour sa fonction relativedelta qui permet de faire des calculs avancés sur les dates.

### Tables - DataTables

**Les tables disposent de `handlers` (ou generator) situés dans le dossier `static` de leur module respectif. Chacune dispose d**

- Configuration : pour les actions autorisées sur la table (impression, tri, recherche, etc) et le format (responsive, etc)
- Initialisation : fonction de dessin appelée par le script de la page requérant la transformation de la table HTML en une DataTable. Cette fonction initialise l'appel à l'API et les différents listeners pour les actions.

Un `genericTableHandler` commun est utilisé par toutes les tables pour les fonctions communes (transactions au serveur, traduction française des boutons, etc).

### Fenêtres modales et formulaires

**Les modales/formulaires disposent de `handlers` situés dans le dossier `static` de leur module respectif. Chacune dispose d**

- Validation : pour vérifier localement les informations entrées dans les champs et préparer la requête d'envoi
- Initialisation : En ajout et en édition pour respectivement vider ou pré-remplir les champs en fonction des informations déjà disponibles

Un `genericModalHandler` commun est utilisé par toutes les modales pour les fonctions communes.

## 1.2 Documentation destinée aux utilisateurs