UCLouvain

epl

École polytechnique de Louvain

Développement d'une application Web pour la gestion des données pluviométriques

EPL21-232

Auteurs: **Florian Duprez, Nicolas Verbois**
Promoteurs: **Kim Mens, Sandra Soares-Frazao**
Lecteurs: **Marnik Vanclooster, Jean Vanderdonckt**
Année académique 2020–2021
Master [120] : ingénieur civil en informatique

Résumé

Ce mémoire est le fruit d'un travail réalisé durant notre dernière année de Master en Sciences Informatiques à l'École polytechnique de Louvain-La-Neuve, durant l'année académique 2020-2021.

Il trouve son origine en Haïti, pays en proie à divers bouleversements résultant de catastrophes naturelles auxquelles ce dernier a dû et doit encore faire face. Un des nombreux problèmes majeurs rencontré par ce pays est la gestion du réseau de données pluviométriques dont les données se retrouvent dispersées et pratiquement inexploitables du fait qu'il n'existe pas de format standardisé. Par conséquent, l'objectif principal de ce mémoire sera de fournir une solution sous forme d'outil permettant le stockage et la gestion des données pluviométriques.

Dans un premier temps, nous abordons le contexte haïtien, notamment les problématiques et enjeux qui y sont liés. Nous avons ainsi traité l'aspect technique de l'hydrologie, et nous avons mis en avant les quelques concepts clefs à la réalisation de notre travail. Dans un deuxième temps, afin de pouvoir rédiger notre cahier des charges, nous nous sommes basés sur deux applications existantes afin d'en observer les points forts.

Dans un troisième temps, après avoir rédigé notre cahier des charges détaillant les différents besoins fonctionnels et non-fonctionnels de notre solution, nous nous sommes lancés dans une phase d'implémentation, qui débute par une analyse technique. Les différentes solutions possibles qui s'offrent à nous y sont détaillées, et nous motivons finalement notre choix parmi celles-ci.

Ce n'est que par après que viendra la description de l'application web avec ses diverses pages, suivie d'une description technique de la manière dont nous avons réalisé l'implémentation des technologies présentées précédemment pour créer notre application. Nous aborderons notamment, les concepts clefs de Django, liés à la structure de notre base de données ainsi qu'aux technologies les plus importantes auxquelles nous avons eu recours.

La dernière partie de ce mémoire est consacrée à une phase de tests et de validation, dont une partie s'est déroulée au travers de tests unitaires automatiques, et l'autre en collaboration avec le client. Nous concluons ce mémoire en évoquant de futures améliorations possibles pour cette application web ainsi que des perspectives envisageables.

Vous pouvez retrouver la totalité du code informatique de notre travail dans un répertoire GitHub à l'adresse suivante :

`https://github.com/nverbois/TFE21-232`

Vous aurez ainsi la possibilité de tester notre prototype fonctionnel sur un serveur test de l'université à l'adresse internet suivante :

`http://tfe-verbois-duprez.info.ucl.ac.be/`

Si vous souhaitez profiter d'une expérience complète de l'application, et par exemple tester son interface administrative, vous pouvez vous connecter au site web avec les identifiants administrateurs suivants :

L'adresse email : admin@admin.com

Le mot de passe : Rek9086TFE

Tout d'abord, nous souhaitons remercier chaleureusement nos promoteurs, M. Mens et Mme Soares-Frazão, pour leur accompagnement et soutien au travers de la réalisation de ce mémoire.

Nous remercions aussi nos correspondants à l'Université d'état d'Haïti pour leur aide et leurs retours dans le développement de ce projet. Nous remercions Benoît Duhoux de nous avoir fourni et expliqué son script python servant à créer notre graphe de résultats SUS.

Nous tenons enfin à remercier nos familles et nos proches pour leur soutien indéfectible.

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Ce mémoire se situe dans un contexte propre à un développement durable, afin de faire avancer les pratiques liées aux technologies de l'information dans un pays en plein développement et sensible aux aléas de la nature.

En effet, Haïti est un pays extrêmement vulnérable aux catastrophes naturelles, principalement aux ouragans, aux inondations et aux tremblements de terre. Dans le courant de l'année 2010, le pays avait été frappé par un séisme de grande ampleur, causant de nombreuses pertes humaines et des dégâts catastrophiques. C'est à peine relevé de cette catastrophe qu'en 2016, la partie sud du pays fut ensuite ravagée par un ouragan. La tempête de catégorie 4 a causé d'importantes inondations et de graves dommages aux infrastructures du pays, ainsi que plusieurs centaines de morts.

On estime ainsi que plus de 90% de la population haïtienne serait exposée aux aléas naturels [1]. Et c'est au constat de cette situation précaire dans laquelle le pays se trouve que le projet de notre mémoire de fin d'études prend forme.

1.2 Historique

Avant toute chose, il est bon de souligner un point particulier quant à l'historique de ce travail de fin d'étude, étant donné qu'il ne s'agit pas de la première fois que le sujet est traité. Au contraire, une version précédente du travail existe déjà. Il s'agit des anciens mémorants Ansotte Gaëtan, Bailly Roland et Ricci Dorian qui ont été les premiers à avoir travaillé sur le sujet, et vous trouverez une référence vers leur mémoire dans la bibliographie [2].

Le travail de nos prédécesseurs nous a servi d'exemple au début de nos réflexions quant à la création de l'application web, mais nous avons fait le choix de ne pas nous en servir comme d'une base et de nous en détacher le plus possible. Nous préférons retravailler l'application depuis ses fondements, et ce pour plusieurs raisons, notamment afin de ne pas être contraints dans nos choix de technologies backend/frontend, mais aussi afin de nous laisser plus de liberté dans la conception de l'application. Une étude plus détaillée de cet ancien mémoire est réalisée dans le chapitre 3 : "Étude d'outils existants".

1.3 Problématique

A la lumière du contexte cité précédemment, les dégâts causés par les dernières catastrophes naturelles ont placé le pays et ses diverses institutions dans une situation délicate. Haïti tente actuellement de se reconstruire, mais le manque de moyens ralentit fortement le développement soutenable du pays. Certains services et projets s'en trouvent donc impactés, comme c'est le cas du Service National des Ressources en Eau de Haïti, ou SNRE.

En effet, une des missions du SNRE est la gestion des données pluviométriques au travers du pays. Ces différentes données ne sont malheureusement pas encore centralisées, mais au contraire éparpillées à travers le pays et ce, dans différentes institutions locales. La problématique pour le SNRE concerne donc le fait que regrouper toutes ces données risque de s'avérer être un travail assez colossal pour un service en peine de moyens suite aux crises qu'Haïti a traversées.

1.4 Objectifs

Suite à la problématique énoncée plus haut, l'objectif de ce travail de fin d'études sera donc de fournir une application web soutenue par une base de données qui servira d'outil permettant l'analyse des données pluviométriques en Haïti. Cette application aura pour but la visualisation et la représentation de données pluviométriques sous différentes formes, qu'il s'agisse de graphes, de tableaux ou même d'une carte. Nous pensons que cet outil jouera un rôle crucial dans le cadre du contexte pluvial particulier d'Haïti et de sa vulnérabilité aux aléas naturels. Il s'agira aussi d'imposer un canevas afin de standardiser le format des données pluviométriques au travers du pays.

Dans un premier temps, nous nous concentrerons sur les fonctionnalités que nous jugeons primordiales afin de fournir une version stable et opérationnelle de l'application. Ensuite, nous explorerons autant que possible l'implémentation de fonctionnalités plus approfondies en fonction du temps qu'il nous restera, étant donné la durée temporelle de réalisation de notre TFE nous limitant malheureusement dans le développement approfondi de celles-ci. L'objectif principal sera donc d'offrir en priorité un prototype stable de l'application web.

Nous pensons que ce travail apportera une aide considérable dans le domaine de l'hydrologie et au Service National des Ressources en Eau de Haïti. Il devrait permettre au processus de décision de certains projets en matière des Ressources en Eau de se réaliser plus rapidement, grâce à la centralisation des données et au traitement et transmission de ces informations. A plus grande échelle, nous pensons que l'application web pourrait aussi avoir un impact positif sur la population haïtienne ainsi que sur le maintien de la biodiversité.

1.5 Approche

L'approche que nous avons utilisée afin d'atteindre nos objectifs pourrait être qualifiée de "semi-agile". En effet, en ingénierie logicielle, une pratique agile met fortement en avant la collaboration avec le client, au travers de contacts réguliers dans le but d'être flexible face aux changements et de permettre une grande réactivité à ses demandes. Dans notre cas, nous n'avions pas contact directement avec "le client", mais nous avions des contacts assez réguliers avec nos professeurs qui allaient donc "prendre le rôle" de l'autre partie intervenante. C'est pour cela que

nous qualifierons notre approche de semi-agile. Nous pouvons aussi décomposer notre approche en différentes étapes.

Dans un premier temps, nous avons commencé à nous informer sur l'hydrologie, étant donné que nous n'avions jusqu'alors aucune notion en la matière. Nous avons pour cela reçu beaucoup d'aide de la part de notre promotrice Sandra Soares-Frazao. Ensuite, nous avons cherché à établir clairement les différents besoins fonctionnels pour la conception de l'application. Nous avons aussi choisi d'écrire différentes "user stories" ou "récits utilisateurs" afin de mieux saisir les attentes qui pourraient être exprimées par un utilisateur. Tout cela sera décrit plus amplement dans la troisième section de ce mémoire : "Analyse du problème".

Après cela, nous nous sommes penchés sur l'analyse technique, afin notamment de motiver notre choix de langage de programmation ou de technologies. Nous avons par après illustré l'application web en décrivant chaque interface du site et les différentes fonctionnalités. Ensuite, nous nous sommes lancés dans l'implémentation de l'application web dans une démarche semi-agile. Nous décrirons ensuite tous les tests et validations qui ont été réalisés en fin de projet. Et enfin, nous terminerons par un chapitre qui discute des différentes pistes d'exploitation suivi d'une conclusion au projet.

1.6 Contribution

Au travers de ce travail de fin d'études, nous pensons avoir permis aux clients d'être en possession d'un prototype fonctionnel de solution technologique qui leur permettra de mieux centraliser et répertorier les données pluviométriques du pays.

Nous espérons que cette contribution en la forme d'une application permettra aux acteurs en Haïti de mieux comprendre comment une telle solution peut les aider dans leurs travaux, qu'ils prendront conscience de l'importance des enjeux liés à l'intégration des technologies modernes et plus précisément celles de l'information, dans leurs projets.

Finalement, nous espérons de tout coeur avoir contribué d'une certaine manière à l'amélioration de la gestion des données pluviométriques au sein du pays. Nous sommes en tout cas certains que ce travail a grandement contribué à développer nos compétences techniques, notamment en matière de développement web au travers de frameworks comme Django et de librairies que nous avons manipulés pour le projet, mais aussi nos compétences en matière de travail d'équipe.

Chapitre 2

Hydrologie et pluviométrie

Avant de rentrer dans la partie plus technique de ce mémoire, nous estimons qu'il est utile de synthétiser en quelques pages les notions d'hydrologie que nous avons acquises avant de nous lancer dans l'implémentation de l'application.

2.1 Qu'est-ce que l'hydrologie

Nous commencerons par citer une définition de l'Encyclopædia Universalis : "Étymologiquement, l'hydrologie est la science de l'eau. Molécule, gaz, liquide ou solide, l'eau voit son étude ressortir à la physique et à la chimie. C'est à l'étude de l'eau dans la nature, où s'expriment évidemment ses propriétés physico-chimiques, qu'est consacrée l'hydrologie. L'eau apparaît dans la nature sous des formes et selon des rythmes extraordinairement diversifiés, mais toujours dans le cadre d'un milieu, souvent support d'un écosystème, où se déroulent de nombreux phénomènes physiques, chimiques et biologiques" [3].

L'hydrologie au sens large s'intéresse donc à toutes les étapes du cycle de l'eau, alors que dans notre cas, nous nous focaliserons surtout sur les précipitations et donc la pluviométrie, définie une fois encore par l'Encyclopædia Universalis comme étant "la mesure de la quantité d'eau de pluie tombée dans un lieu donné" [4].

2.2 Méthodes de mesure

Il existe différentes méthodes et instruments permettant de mesurer la quantité d'eau de pluie tombée dans un lieu donné. Dans le cadre de ce mémoire, nous allons nous concentrer sur les deux types pluviomètres les plus répandus en Haïti. Il s'agit des pluviomètres à lecture directe et des pluviomètres à auget basculeur. Les fonctionnements de ces dispositifs ont été décrits dans une vidéo de notre promotrice Sandra Soares-Frazao [5].

2.2.1 Pluviomètres à lecture directe

Le pluviomètre à lecture directe consiste en un simple récipient récoltant les précipitations. Le récipient possède une graduation permettant de connaître la quantité de précipitation récoltée sur une période précise. Il s'agit du type de pluviomètre le plus "simple" et le plus connu.

FIGURE 2.1 – Exemple d'un pluviomètre à lecture directe [6]

Souvent, la quantité d'eau en surplus due à de fortes abondances est récolté par des récipients secondaires. Un technicien doit régulièrement vider les récipients et enregistrer manuellement les valeurs obtenues.

2.2.2 Pluviomètres à auget basculeur

Ce pluviomètre utilise une petite balançoire composée de deux augets. Ces augets sont capables de contenir jusqu'à 0,5 mm d'eau en général. Les deux augets sont toujours en train d'exécuter deux tâches différentes en alternance.

FIGURE 2.2 – Exemple d'un pluviomètre à auget basculeur [7]

Un des augets, horizontal au sol, se remplit d'eau de pluie tandis que l'autre s'en vide entièrement. Une fois qu'un auget est rempli, il bascule permettant à l'autre auget de passer à l'horizontal et d'inverser les tâches. La quantité de précipitation est alors calculée par le nombre de basculements réalisé par les augets.

2.3 Traitement et représentations des données

2.3.1 Tableaux de données

Une manière de représenter les données pluviométriques d'une station sera de les afficher sous forme de tableaux. Pour chaque station, l'utilisateur va pouvoir entrer différents paramètres. Il choisira s'il veut afficher les données simples, les moyennes ou les intensités.

Pour ce qui est des données simples, l'utilisateur entrera une ou plusieurs date et intervalle de temps et pourra afficher les précipitations sur ces différents intervalles de temps et les comparer entre elles.

Date de la mesure	Heure de la mesure	Mesure [mm]
30-10-2014	17:00	0,000
01-11-2014	03:10	0,200
01-11-2014	06:33	0,400
03-11-2014	21:45	0,600
03-11-2014	21:46	0,800
03-11-2014	21:46	1,000
03-11-2014	21:47	1,200
03-11-2014	21:47	1,400
03-11-2014	21:50	1,600
03-11-2014	22:10	1,800

FIGURE 2.3 – Exemple d'un tableau de données bruts

En ce qui concerne les moyennes, l'utilisateur entrera une ou plusieurs dates journalières, hebdomadaires ou annuelles et pourra afficher les moyennes désirées pour chacune des dates sélectionnées.

Moyennes pluviométriques de la station

Moyennes journalières | Moyennes hebdomadaires | Moyennes annuelles

Afficher 10 éléments — Rechercher :

Date	Moyenne du jour [mm]	Maximum [mm]	Minimum [mm]
30-10-2014	0,000000	0,000000	0,000000
01-11-2014	0,300000	0,400000	0,200000
03-11-2014	1,200000	1,800000	0,600000
06-11-2014	2,400000	2,800000	2,000000
07-11-2014	6,800000	10,600000	3,000000
08-11-2014	11,100000	11,400000	10,800000
09-11-2014	65,000000	118,400000	11,600000
10-11-2014	148,400000	178,200000	118,600000
11-11-2014	180,200000	182,000000	178,400000
12-11-2014	186,600000	191,000000	182,200000

Affichage de l'élement 1 à 10 sur 36 éléments — Précédent 1 2 3 4 Suivant

FIGURE 2.4 – Exemple d'un tableau de moyennes pluviométriques

Enfin, un utilisateur pourra afficher les intensités d'une station, comme vous pouvez le constater à la figure 2.5

Jour	Durée de la mesure [min]	Maximum [mm]	Intensité [mm]
30-10-2014	60	0,000	0,000
30-10-2014	90	0,000	0,000
30-10-2014	50	0,000	0,000
30-10-2014	40	0,000	0,000
30-10-2014	30	0,000	0,000
30-10-2014	20	0,000	0,000
30-10-2014	15	0,000	0,000
30-10-2014	10	0,000	0,000
30-10-2014	180	0,000	0,000
30-10-2014	5	0,000	0,000

FIGURE 2.5 – Exemple d'un tableau d'intensités pluviométriques

L'intensité des précipitations est une mesure qui dépend d'une durée de temps précise. Elle détermine la quantité de précipitation maximale d'une journée mesurée sur la durée de temps sélectionnée et exprimée en mm/heure.

Un utilisateur sélectionnera donc une ou plusieurs dates de son choix et pour chaque date, le tableau affichera une série de périodes de différentes durées (10 min, 20 min, 30 min, 60 min, 90 min, 120 min et 180 min). Pour chaque période, l'utilisateur verra afficher la précipitation maximale récoltée sur cette période, les heures de début et fin de cet intervalle ainsi que l'intensité pluviométrique associée.

2.3.2 Graphes précipitations simples

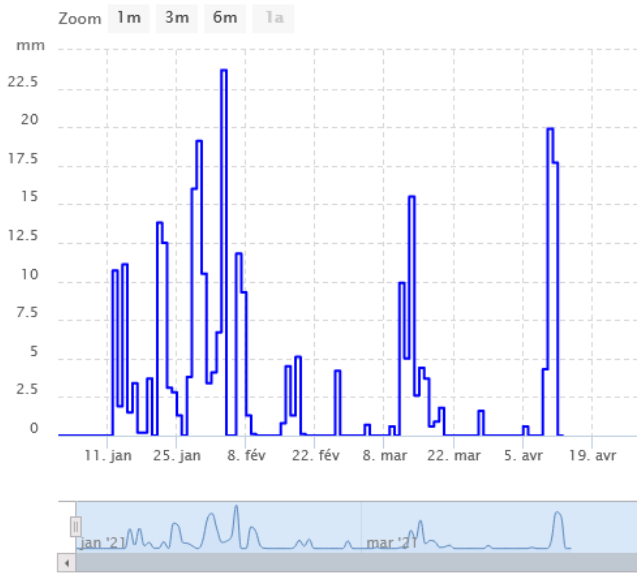


FIGURE 2.6 – Exemple d'un graphe simple de précipitations [8]

Les graphes représenteront les mêmes données que les tableaux précédents. Pour une station précise, l'utilisateur devra choisir une période précise sur laquelle il désire afficher les données. On peut observer ici soit les précipitations soit les moyennes journalières, hebdomadaires ou annuelles en fonction de la période désirée.

Il va être possible de pouvoir ajouter d'autres stations sur les graphes. Pour la même période sélectionnée, on peut comparer 2 régions différentes d'Haiti en observant les précipitations affichées.

2.3.3 Courbes IDF

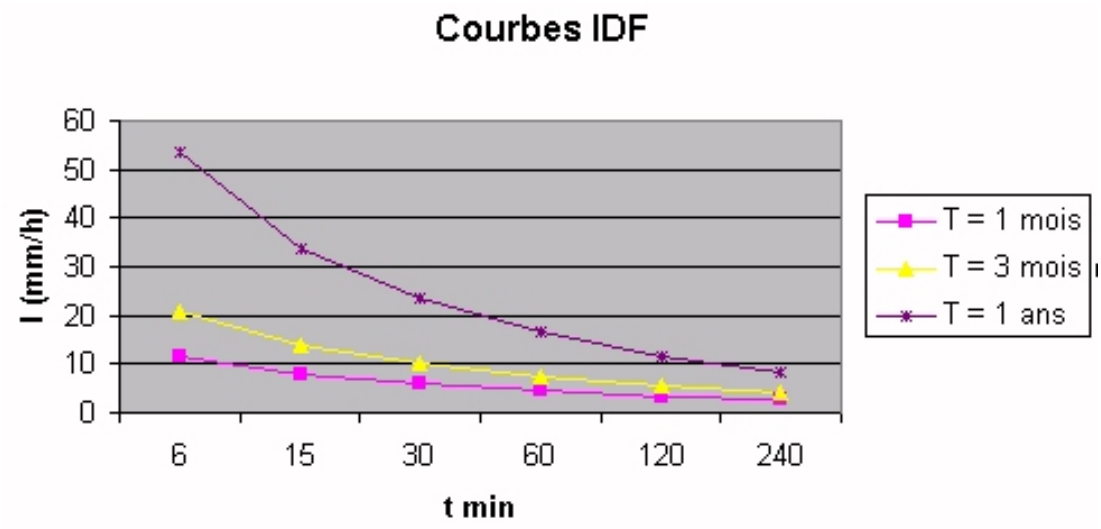


FIGURE 2.7 – Exemple de courbes IDF [9]

Pour les données d'intensités pluviométriques, le principe reste le même que pour les autres graphes. Il s'agit donc d'afficher les intensités de la même manière en fonction d'une période de référence où il sera aussi possible de les comparer à d'autres stations. Mais les intensités dépendent aussi de la durée de référence sur laquelle elles ont été calculées. L'utilisateur va devoir donc choisir en paramètre la durée de référence sur laquelle il souhaite afficher les intensités.

Un utilisateur aura la possibilité d'afficher et donc de comparer des intensités calculées sur des durées de référence différentes en plus des autres stations. Les intensités pluviométriques seront présentées sous forme de courbes IDF, ou courbes Intensité-Durée-Fréquence, comme vous pouvez le voir à la figure 2.7.

Chapitre 3

Étude d'outils existants

Après nous être appropriés ces notions d'hydrologie, il convient, avant de nous lancer dans le développement d'un projet tel que le développement de cette application, de nous renseigner et d'analyser différents exemples existants d'applications traitant du même sujet que nous, à savoir la pluviométrie.

Dans un premier temps, nous commencerons par discuter du travail effectué sur le sujet par nos prédécesseurs, et nous mettrons en lumière les points forts et les points faibles de ce qu'ils avaient réalisé. Ensuite, nous nous intéresserons à un site qui nous a été suggéré par nos promoteurs, à savoir le site wallon de la Direction générale opérationnelle de la Mobilité et des Voies hydrauliques.

3.1 Analyse du précédent mémoire

Comme nous l'avons expliqué dans la section "Historique" de notre introduction, ce sujet de mémoire n'en est pas à sa première itération. En effet, les anciens étudiants Ansotte Gaëtan, Bailly Roland et Ricci Dorian ont constitué le premier groupe de mémorants à s'intéresser au développement d'une application Web pour la gestion des données pluviométriques en Haïti, et ce durant l'année académique 2018-2019 [2].

Afin de réaliser une étude de leur travail, nous nous sommes procurés un exemplaire de leur mémoire. Dans celui-ci, nous pouvons voir que les auteurs nous proposent dans un premier temps de consulter l'application à une certaine adresse, comme nous pouvons le citer de leur mémoire : "**L'application est disponible sur un serveur de test à l'adresse suivante : `https://client-d4rk694.c9users.io/`**" [2].

Ensuite, les auteurs nous invitent aussi à consulter leur répertoire GitHub sur lequel se trouve tout le code source de leur application.

3.1.1 L'application web

Nous allons donc découper l'analyse de ce précédent travail en deux parties, et nous voulons pour commencer analyser l'application. Malheureusement pour nous, et comme vous pourrez le constater, cette adresse ne semble mener nulle part, et le serveur test de l'application semble avoir été coupé depuis un moment.

Cela est dû au fait que leur application était hébergée sur un serveur payant à leur frais, et non sur un serveur qui leur aurait été fourni par la faculté à l'époque, afin de leur permettre de faire tester le site web à des intervenants extérieurs. Ils n'avaient pas le choix étant donné le timing serré qu'ils ont eu en présentant leur mémoire en Janvier. Dès lors, il n'est pas étonnant qu'ils aient coupé leur serveur privé après leur présentation.

D'autre part, nous avions toujours espoir de pouvoir faire fonctionner leur application localement, car l'accès au code source au travers de leur répertoire GitHub nous était accessible, comme nous pouvons le citer de leur mémoire : "**Pour toute personne souhaitant reprendre le projet, celui-ci est disponible à cette adresse : `https://github.com/doricci/TFE4Haiti`**" [2].

En arrivant sur le répertoire GitHub, quelque chose d'assez frappant apparaît, il s'agit du manque important d'informations concernant l'installation de leur application. En effet, leur liste de "requirements", qui sont les logiciels et programmes nécessaires au bon fonctionnement d'une application, se limite simplement à nous demander d'installer ExpressJS[10] et MongoDB[11].

Or, le manque de précision concernant les versions des logiciels, et surtout concernant le type de déploiement pour MongoDB est assez consternant. En effet, comme vous pouvez le voir ci-dessous, MongoDB propose de nombreux types de déploiements différents, pas moins de 6 déploiements différents rien que pour les versions locales de MongoDB [11].

Face au cruel manque d'informations pour réaliser une installation correcte de l'application, ce qui génère des erreurs que nous n'avons pas spécialement le temps ni le devoir de chercher à corriger, et face au serveur de test qui ne semble plus être accessible en ligne, nous nous voyons donc, à notre grand regret, dans l'impossibilité de pouvoir réaliser des tests sur le travail effectué par nos prédécesseurs.

Cette constatation agira comme une des motivations principales à la création d'une nouvelle application. En effet, une ancienne application difficile à installer et faire fonctionner semble être un motif suffisant à la création d'une nouvelle version du projet, avec des outils permettant automatiquement l'installation de dépendances, comme nous le ferons avec Docker.

Cette motivation principale s'ajoutera aux autres motivations déjà citées dans la section 1.2 sur l'historique du travail, à savoir le fait de ne pas être contraints dans notre choix de technologies ou le fait d'avoir plus de liberté lors de la conception.

Enfin, un point important à noter dans leur code, bien qu'on ne puisse pas le faire fonctionner, concerne la manière dont ils avaient choisi de réaliser les calculs de statistiques pluviométriques. En effet, il s'agit selon nous d'un point critique concernant l'application, puisque tout le développement tourne autour du stockage des données pluviométriques et du calcul de statistiques sur celles-ci. Et un choix de design marquant dans leur développement est le choix pour les calculs sur ces données d'être effectués au moment où l'utilisateur demande à consulter une page de statistiques sur leur site web.

Pour nous, il s'agit là d'un choix de design qui n'est pas optimal du tout, surtout pour le déploiement d'une application dont la base de données pourrait contenir des centaines de milliers voire des millions de mesures pluviométriques, dont les calculs statistiques peuvent

prendre un certain temps selon la puissance du serveur. Nous pensons alors qu'il faudrait changer complètement la manière dont les calculs se feront sur les données, de sorte à ce qu'ils soient par exemple réalisés au moment de l'ajout de mesures dans la base de données, pour ensuite stocker les résultats dans celle-ci. Tout cela sera discuté plus en détails lors de la description de notre phase d'implémentation.

3.1.2 Le mémoire

Malgré l'impossibilité à laquelle nous avons fait face quant à l'accès à la version précédente de l'application, il nous est toujours possible de lire le rapport écrit de ce travail de fin d'études et d'en tirer des conclusions intéressantes.

Pour cela, nous allons nous concentrer sur l'analyse de la section intitulée "Produit final" de leur rapport de mémoire [2], et plus précisément de quelques points qui ont tout particulièrement retenu notre attention.

Carte interactive Dans leur barre de navigation, la fonctionnalité de la carte est la première accessible aux utilisateurs en dehors de la page d'accueil. Sur cette carte interactive centrée sur Haïti, ils ont permis l'affichage des différentes stations pluviométriques enregistrées au sein de la base de données sur la carte, représentées par des indicateurs de position verts.

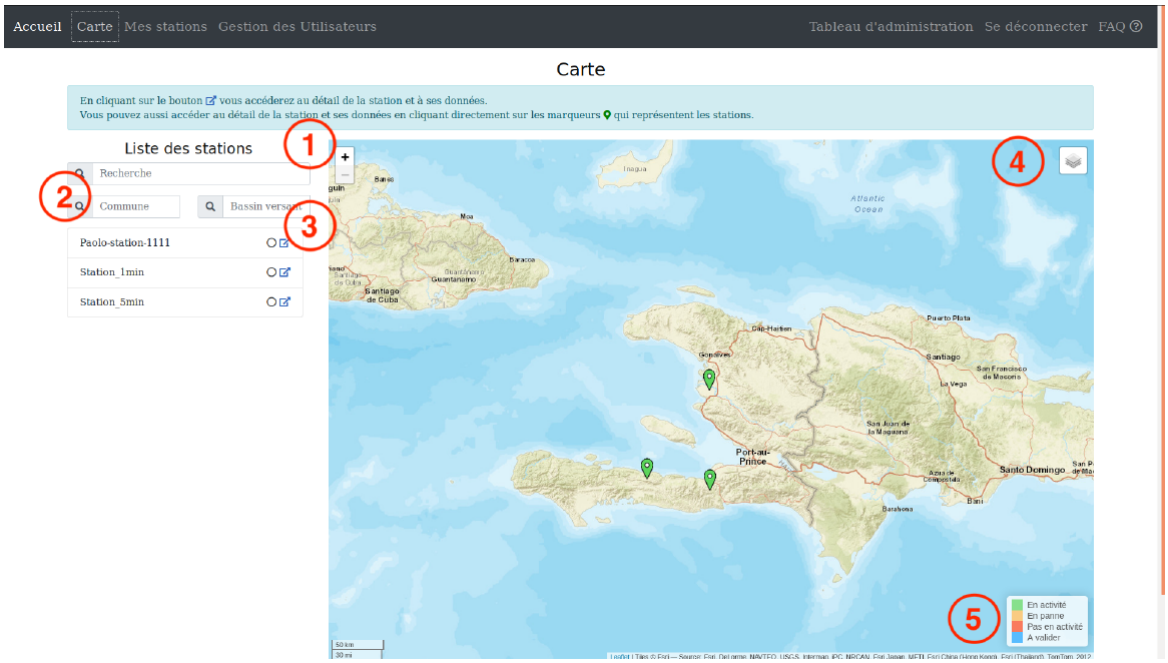


FIGURE 3.1 – Image de la carte interactive du précédent TFE, tirée de leur mémoire [2]

Cette fonctionnalité, qui est un point commun avec le prochain outil existant que nous analyserons au point 3.2, est un des premiers points que nous avons jugé bon d'intégrer à nos besoins fonctionnels. En effet, une carte interactive telle que celle-ci est intuitive et simple à prendre en main pour les utilisateurs, une qualité que nous recherchons pour développer une application ergonomique.

Informations d'une station Ensuite, il est possible pour un utilisateur de leur site web d'accéder aux données de la station, sur une page qu'ils intitulent "Informations d'une station". Un utilisateur pourra alors consulter les détails de la station elle-même, comme son statut d'activité ou sa date d'installation. Cet onglet n'est pas un des plus primordiaux à nos yeux.

Cependant, notre attention a été retenue par les onglets intitulés "Graphiques" et "Tableaux". Le premier permet donc de visualiser les données pluviométriques de la station sous la forme d'un graphique. L'interface est très intéressante, elle offre directement sur cette page un bouton permettant d'importer des données pluviométriques. L'utilisateur peut aussi changer le type d'intervalle (mensuel, annuel ou journalier), les dates affichées dans l'intervalle et même le type du graphe (bâtonnets ou lignes).

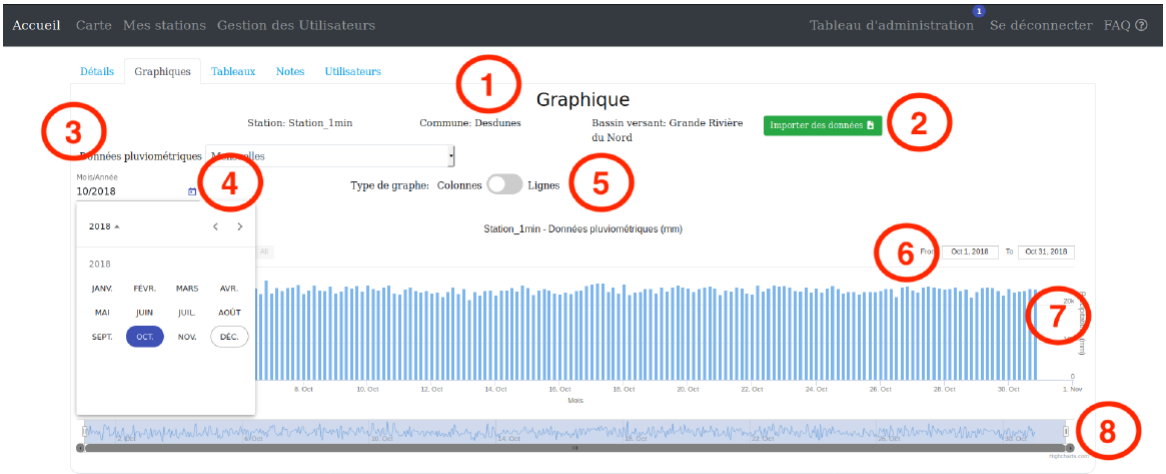


FIGURE 3.2 – Image des graphes de données du précédent TFE, tirée de leur mémoire [2]

Le second onglet permet quant à lui un affichage des mêmes données mais sous forme de tableaux plutôt que sous forme de graphes. L'interface contient par exemple :

- Choisir le format d'intervalle pour la représentation des données, à savoir annuel, mensuel ou journalier.
- Un champs permettant de choisir la période temporelle à afficher.
- Un tableau contenant les données pluviométriques à "l'état brut" de la période temporelle choisie, affichée selon le format d'intervalle temporel sélectionné.
- Un second tableau contenant des statistiques sur ces données, comme les minima, maxima et moyennes pluviométriques.

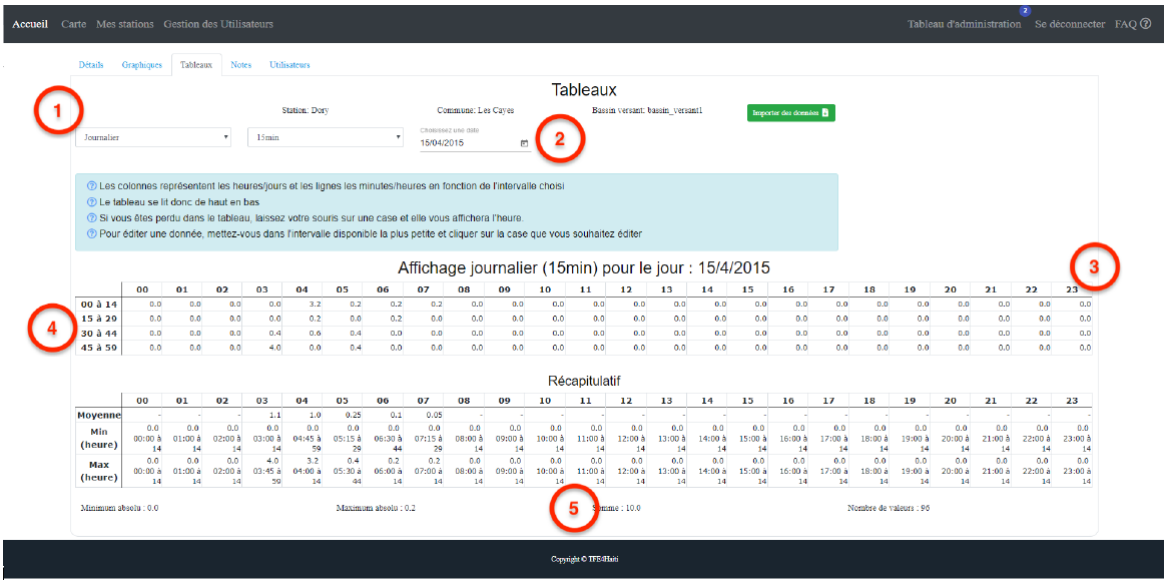


FIGURE 3.3 – Image des tableaux de données du précédent TFE, tirée de leur mémoire [2]

Les formats des tableaux et des graphes sont très similaires à ceux de l'application wallonne, à savoir le "Site wallon de la Direction générale opérationnelle de la Mobilité et des Voies hydrauliques", qui sera décrit juste après au point 3.2.

Comme nous l'avons évoqué juste avant, c'est donc lors de la requête web d'affichage de ces onglets que les calculs statistiques sont seulement effectués sur les données, un choix qui n'est pas optimal selon nous. Cela avait aussi été confirmé par quelques utilisateurs qui avaient pu tester l'application précédente ; ceux-ci avaient relevé des temps de chargement trop longs. De plus, en prenant en compte la contrainte de vitesse de connexion internet, il n'est pas désirable de perdre encore plus de temps à attendre que les calculs statistiques soient effectués au moment de la requête.

Importation de données L'importation de données dans la base de données pluviométriques est une des fonctionnalités les plus importantes du site web. Ici, ils avaient décidé d'accepter des fichiers au format CSV uniquement, ou alors d'effectuer une importation manuelle de données.

Administration Il est enfin nécessaire, selon nous, pour un tel site web de posséder une bonne console d'administration afin d'aisément gérer les stations, les utilisateurs et surtout les données précédemment importées. Cela semblait être le cas pour l'ancienne application, et c'est donc quelque chose que nous aimerions garder dans notre itération du projet.

3.2 Site wallon de la Direction générale opérationnelle de la Mobilité et des Voies hydrauliques

Dans un second temps, après avoir analysé l'ancien mémoire traitant de notre sujet et comme nous sommes venus à la conclusion qu'il serait plus intéressant pour nous de recommencer le travail sur une nouvelle base, nous nous sommes intéressés à un autre site existant qui touche à un sujet similaire au nôtre.

Ce site n'est autre que celui de la " Direction générale opérationnelle de la Mobilité et des Voies hydrauliques "[12], un site web belge du service public de la Wallonie, qui nous permettra d'avoir une meilleure vue d'ensemble du type d'interface attendu pour une application traitant de la pluviométrie comme la nôtre.

Carte interactive Lorsque l'on accède à la section 'Annuaires et statistiques' de la partie 'Données archivées et statistiques' de la rubrique 'Hydrologie', nous faisons face à la première interface qui a retenu notre attention, à savoir aussi une carte interactive.

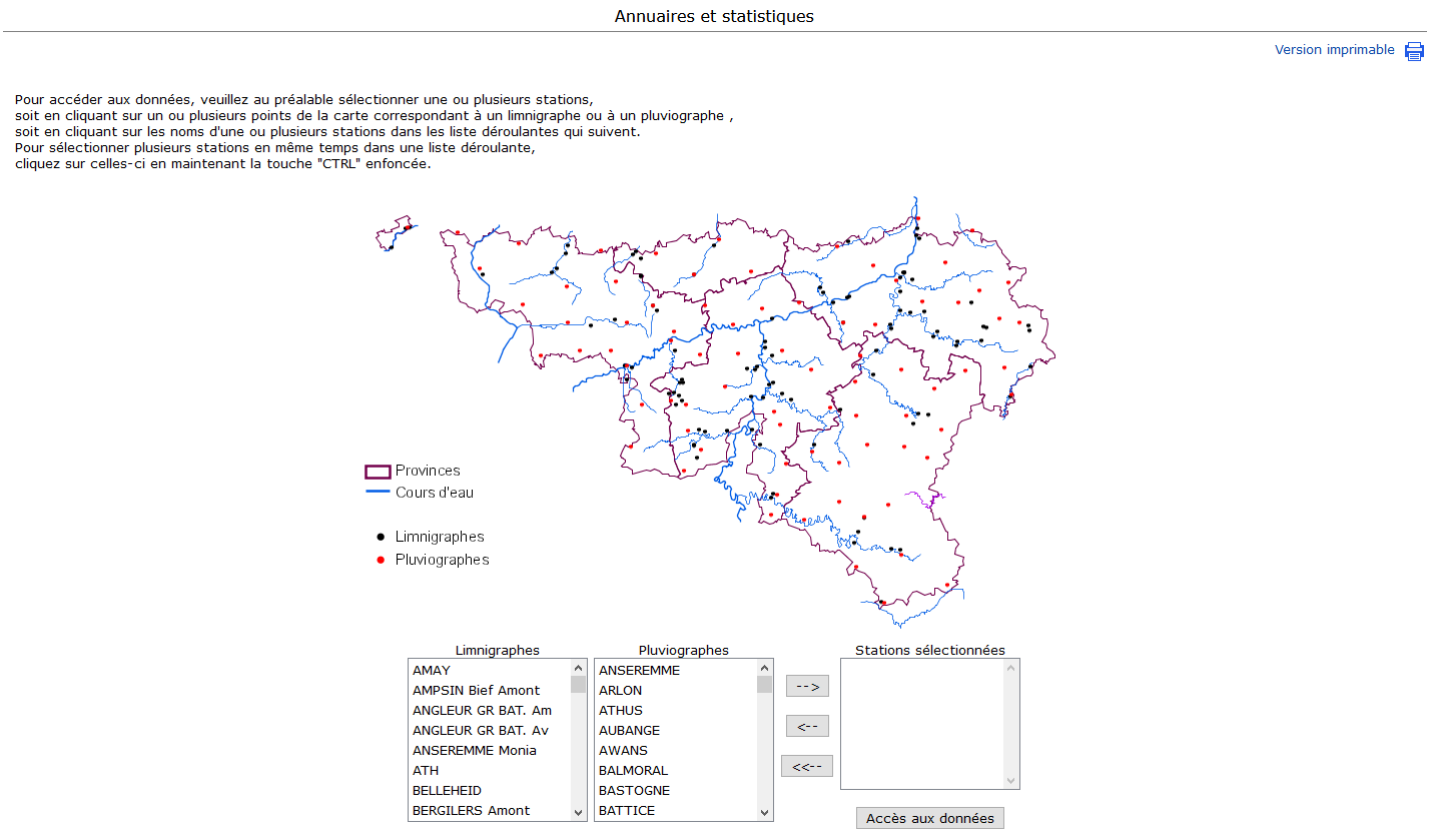


FIGURE 3.4 – Carte interactive de l'application web wallonne [12]

La carte en elle-même affiche différentes informations importantes, telles que :

— Les provinces belges
— Les cours d'eau traversant le pays
— L'emplacement des pluviographes, qui sont des pluviomètres reliés à des enregistreurs en continu de la hauteur d'eau des précipitations.

— L'emplacement d'autres appareils de mesure ne nous intéressant pas dans notre cas, à savoir des limnigraphes pour les cours d'eau.

Ensuite, en-dessous de la carte se trouve une interface composée de plusieurs listes à sélections multiples, permettant de choisir une ou plusieurs stations, de type pluviographes ou limnigraphes, à consulter. Pour cela, il suffit à l'utilisateur de cliquer sur une des stations à gauche, et d'ensuite utiliser le bouton en forme de flèche pour le faire passer dans la liste des stations sélectionnées. Après cela, l'utilisateur peut cliquer sur le bouton d'accès aux données pour être emmené sur l'interface suivante.

Nous avons trouvé le principe d'une interface incluant une carte interactive affichant les stations pluviométriques au travers d'un pays comme étant particulièrement intéressant à inclure à notre future application web. C'est d'ailleurs le choix que les précédents mémorants avaient aussi effectué.

Inventaire des données L'interface suivante est intitulée "Inventaire" et présente aux utilisateurs du site web un Tableau du nom de "Précipitations (mm)". Ce tableau sert principalement à présenter la moyenne pluviométrique de la station sur une période de référence définie, ainsi que les maxima et minima de cette période de référence. Il est aussi possible d'accéder aux informations de la station sélectionnée.

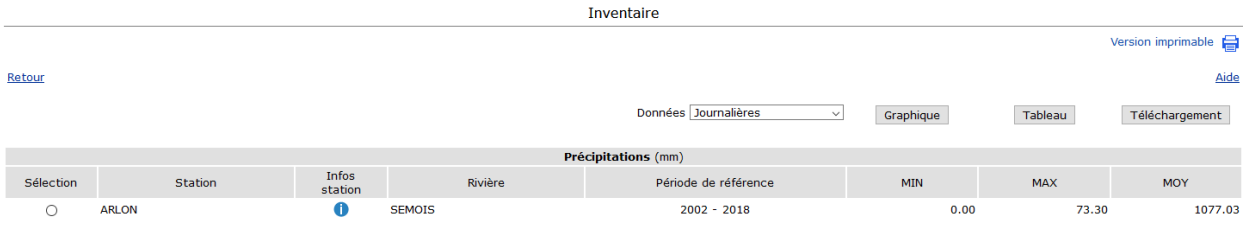


FIGURE 3.5 – Résumé de données de la (les) station(s) [12]

Tableaux des valeurs journalières Lorsque l'on clique sur le bouton intitulé "Graphique" en haut à droite de la page, l'utilisateur est alors amené sur une page contenant des tableaux. Cette interface comporte donc :

— Un grand tableau, contenant l'ensemble des données de précipitations pluviométriques rassemblées par jour.
— Un autre tableau au-dessus du précédent, ressemblant des données récapitulatives pour chaque mois, comme par exemple le minimum ou maximum du mois, la somme des mesures de précipitations en mm du mois, etc.

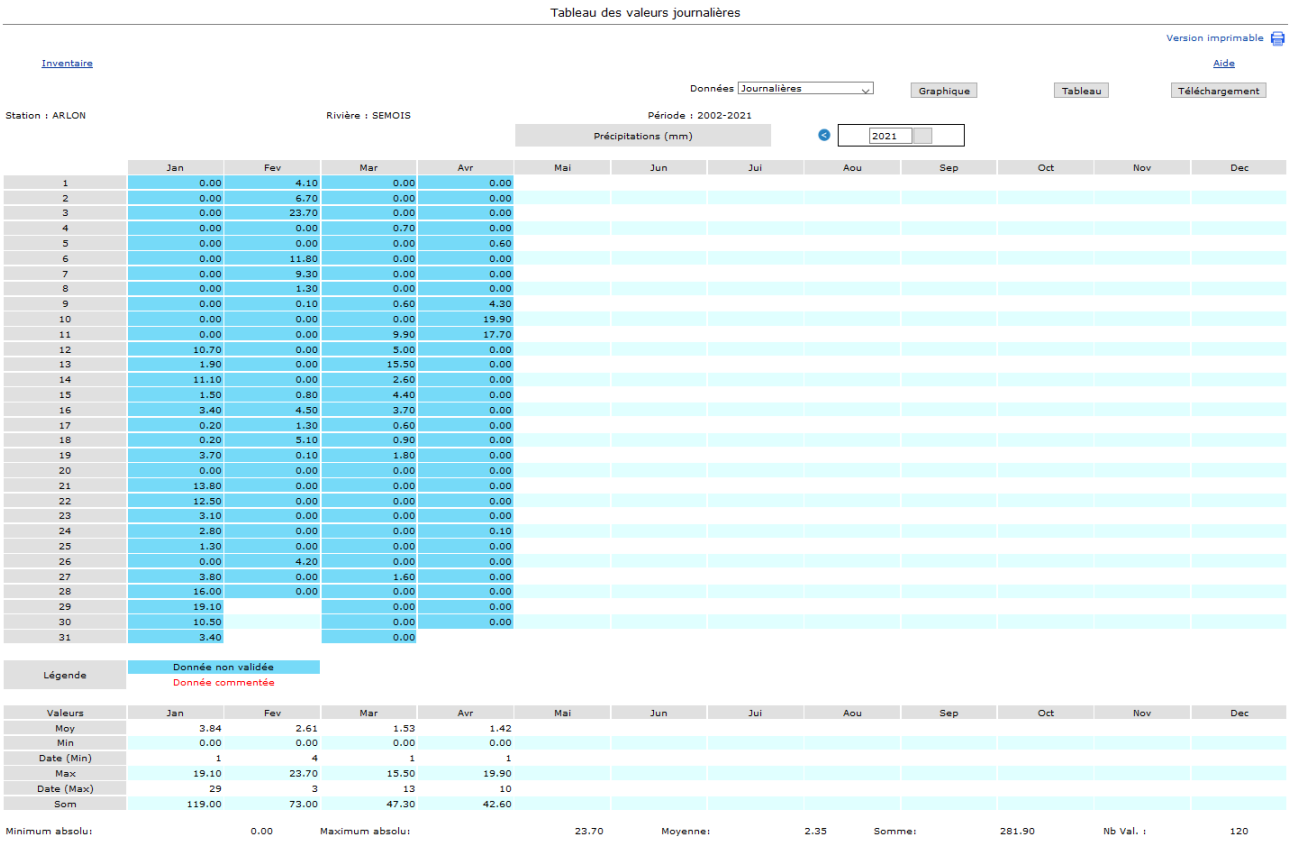


Tableau des valeurs journalières

Version imprimable

Inventaire

Aide

Données Journalières | Graphique | Tableau | Téléchargement

Station : ARLON | Rivière : SEMOIS | Période : 2002-2021

Précipitations (mm) | 2021

	Jan	Fev	Mar	Avr	Mai	Jun	Jui	Aou	Sep	Oct	Nov	Dec
1	0.00	4.10	0.00	0.00								
2	0.00	6.70	0.00	0.00								
3	0.00	23.70	0.00	0.00								
4	0.00	0.00	0.70	0.00								
5	0.00	0.00	0.00	0.60								
6	0.00	11.80	0.00	0.00								
7	0.00	9.30	0.00	0.00								
8	0.00	1.30	0.00	0.00								
9	0.00	0.10	0.60	4.30								
10	0.00	0.00	0.00	19.90								
11	0.00	0.00	9.90	17.70								
12	10.70	0.00	5.00	0.00								
13	1.90	0.00	15.50	0.00								
14	11.10	0.00	2.60	0.00								
15	1.50	0.80	4.40	0.00								
16	3.40	4.50	3.70	0.00								
17	0.20	1.30	0.60	0.00								
18	0.20	5.10	0.90	0.00								
19	3.70	0.10	1.80	0.00								
20	0.00	0.00	0.00	0.00								
21	13.80	0.00	0.00	0.00								
22	12.50	0.00	0.00	0.00								
23	3.10	0.00	0.00	0.00								
24	2.80	0.00	0.00	0.10								
25	1.30	0.00	0.00	0.00								
26	0.00	4.20	0.00	0.00								
27	3.80	0.00	1.60	0.00								
28	16.00	0.00	0.00	0.00								
29	19.10		0.00	0.00								
30	10.50		0.00	0.00								
31	3.40		0.00									

Légende : Donnée non validée / Donnée commentée

Valeurs	Jan	Fev	Mar	Avr	Mai	Jun	Jui	Aou	Sep	Oct	Nov	Dec
Moy	3.84	2.61	1.53	1.42								
Min	0.00	0.00	0.00	0.00								
Date (Min)	1	4	1	1								
Max	19.10	23.70	15.50	19.90								
Date (Max)	29	3	13	10								
Som	119.00	73.00	47.30	42.60								

Minimum absolu: 0.00 | Maximum absolu: 23.70 | Moyenne: 2.35 | Somme: 281.90 | Nb Val. : 120

FIGURE 3.6 – Tableaux de données de la station [12]

Graphique des valeurs journalières Ensuite, lorsque l'on clique sur le bouton intitulé "Graphes" en haut à droite de la page, l'utilisateur est amené sur une page contenant un graphe. Le graphe de type "bâtonnets" affiche les mesures journalières pour la station en millimètre de pluie. Les jours affichés sont ceux compris dans une période de référence d'un mois, que l'utilisateur peut changer si il le souhaite dans les cases prévues à cet effet au dessus du graphe.

On peut aussi constater au dessous du graphe une barre interactive qui permet de gérer le niveau de "zoom" sur le graphe.

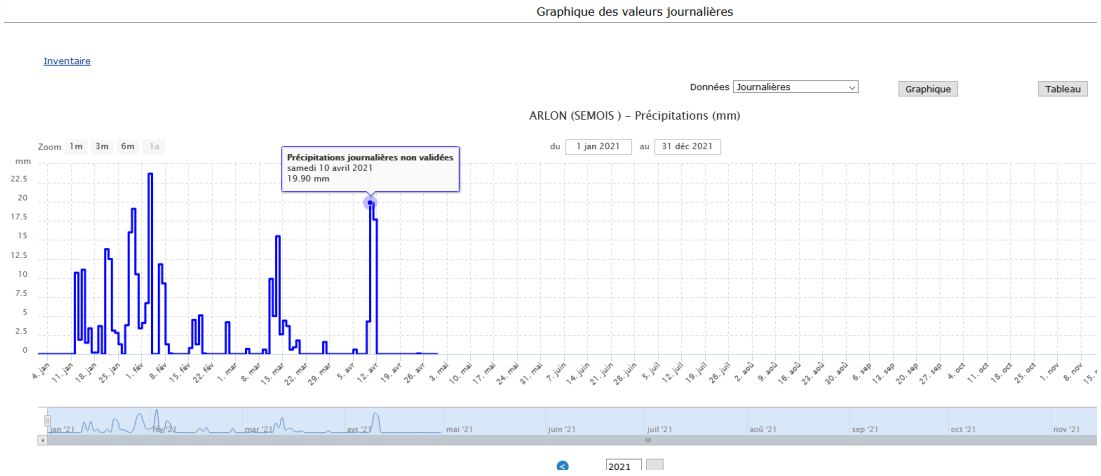


FIGURE 3.7 – Graphes de précipitations de la station [12]

3.3 Conclusion de l'étude des outils

Pour conclure notre analyse de ces deux outils, nous allons vous présenter ci-dessous, sous forme de tableau, les différentes fonctionnalités et idées que nous souhaitons intégrer dans notre application, celles que nous ne retiendrons pas telles quelles et voudrions modifier, ainsi que celles que nous allons ajouter.

Fonctionnalités à conserver	Fonctionnalités à modifier	Fonctionnalités à ajouter
Carte interactive	Importation des données	Calcul des intensités pluviométriques
Graphes de données	Calcul des statistiques	Déploiement simple de l'application
Tableaux de données		

Table 3.1 – Tableau récapitulatif des fonctionnalités

Parmi les **fonctionnalités à conserver** pour notre application, on retrouve celles qui étaient présentes sur les deux outils que nous avons étudiés et similaires entre elles, à savoir :

1. une carte interactive du pays avec les différentes stations affichées sur celle-ci, cela nous semble être une manière simple, directe et intuitive de représenter les stations pluviométriques.
2. un affichage des données et statistiques sous forme de tableaux,
3. un affichage des données et statistiques sous forme de plusieurs graphiques.

En effet, ces fonctionnalités nous semblent essentielles à une application telle que celle que nous allons développer, et il sera donc primordial de les ajouter dès le début de la phase d'implémentation.

Parmi les **fonctionnalités à modifier** pour notre application, on retrouve des fonctionnalités que nous avons trouvé intéressantes sur l'ancienne application mais qui devraient être remaniées :

1. L'importation de données est une fonctionnalité très importante, vu la taille des fichiers de mesures pluviométriques, il est évident qu'il faudra permettre aux utilisateurs d'importer un fichier plutôt que d'encoder chaque mesure à la main.
   Nous considérons cependant que la version de l'importateur de l'ancienne application était un peu restrictive en ne permettant que de l'import au format CSV, et souhaiterions plutôt nous tourner vers un import au format XLS ou XSLX, étant donné que les données semblaient plutôt être stockées sous ces formats là du côté du client.
   De plus, nous pensons qu'il serait plus judicieux de permettre l'importation directement dans la console d'administration et non pas de laisser un bouton sur les pages de statistiques, cela semble plus sûr à nos yeux d'un point de vue de sécurité.

2. Le calcul des statistiques quant à lui sera modifié dans le sens où, comme nous l'avons expliqué plus haut, nous souhaiterions qu'il ne soit plus effectué au moment de la requête de ces statistiques mais en amont, dès leur insertion après l'importation.

Enfin, les **fonctionnalités à ajouter** viennent en réponse à des constations de fonctionnalités manquantes ou simplement de choses qui faisaient que l'ancienne application ne fonctionnait pas correctement et que nous voulons rejeter, à savoir :

1. le calcul d'intensités pluviométriques en plus des autres statistiques comme les moyennes, maxima et minima,
2. la possibilité de déployer simplement l'application, comme ça n'avait pas été le cas pour nous avec l'ancienne version du mémoire. Nous voulons simplifier au maximum ce déploiement, qu'il puisse se réaliser en un minimum d'étapes et sans encombre, et ce sur tout type de machine.

Une dernière note, avant de conclure cette partie, concerne certaines fonctionnalités que nous ne rejetons pas, mais que nous ne considérons pas comme prioritaires et qui ne seront donc peut-être pas reprises des outils existants. Par exemple, nous pensons à certaines pages de l'application comme celle affichant les détails d'une station avec sa photo.

Chapitre 4

Analyse du problème

Cette section est équivalente à un cahier des charges, où nous allons commencer par identifier les différents besoins fonctionnels et non-fonctionnels. Pour ce faire, nous définirons une méthode d'analyse qui déterminera les différentes tâches à réaliser et l'importance consacrée à chacune d'entre elles. Ensuite nous discuterons des contraintes imposées au projet et comment faire fonctionner notre travail sous ces contraintes. Et pour finir, nous discuterons des différents rôles utilisateurs du site et de leur différentes fonctions dans l'application.

4.1 Besoins fonctionnels

Sur base de l'étude de l'existant réalisée au chapitre précédent ainsi que sur base de discussions avec nos promoteurs, nous avons pu identifier différents besoins fonctionnels pour notre application. Nous allons diviser les besoins fonctionnels en différentes catégories en nous basant sur la méthode MoSCoW. Nous définirons ainsi certaines limites au travail que nous allons effectuer.

Les lettres majuscules de l'acronyme MoSCoW signifient ceci (en anglais) :

— **M** : must have this, c'est-à-dire 'doit être fait' (vital).
— **S** : should have this if at all possible, c'est-à-dire 'devrait être fait dans la mesure du possible' (essentiel).
— **C** : could have this if it does not affect anything else, 'pourrait être fait dans la mesure où cela n'a pas d'impact sur les autres tâches' (confort).
— **W** : won't have this time but would like in the future, 'ne sera pas fait cette fois mais sera fait plus tard' (c'est plus un luxe, correspondant à notre zone d'optimisation).

Fonctionnalités M

1. Présenter aux utilisateurs une carte interactive affichant les différentes stations pluviométriques au travers du pays. Cette fonctionnalité est vitale car il s'agit de pouvoir informer les utilisateurs de la situation géographique des différentes stations du pays. Comprendre d'où proviennent les différentes données pluviométriques du pays pourra aider les utilisateurs à éventuellement déterminer quelles sont les régions où les précipitations sont les plus élevées.

2. Sélectionner une station sur la carte afin d'afficher différentes données et statistiques à son sujet :
   — Le nom de la station sera affiché lorsque l'on passe dessus.
   — Une fois une station sélectionnée (en cliquant dessus), une page s'ouvrira avec les graphiques et tables de données de cette station.
   — Les données seront affichées de manière journalière, hebdomadaire, ou annuelle en fonction des choix de l'utilisateur.

   Cette fonctionnalité permettra d'accéder directement aux données d'une station particulière. Il s'agit de la fonctionnalité la plus vitale de l'application.
3. Permettre aux administrateurs de créer des comptes pour les autres administrateurs et gestionnaires de stations. Cela va leur permettre de s'identifier sur l'application web et d'accéder à des fonctionnalités supplémentaires au travers d'une console d'administration. Cette fonction est vitale pour permettre la bonne gestion de l'application et des données.
4. Permettre aux gestionnaires de stations d'encoder leurs données pluviométriques sur le site web. Un canevas Excel sera imposé pour faciliter la généralisation des données provenant des différentes stations. Cette fonction est importante car elle permet d'enregistrer les données en ligne et d'y donner accès à n'importe quel utilisateur.
5. Créer une base de données pour enregistrer les données de chacune des stations dans différentes tables, dont le contenu devra être visible par les administrateurs. C'est une fonction nécessaire pour encoder les données de façon efficace pour y accéder facilement.

Fonctionnalités S

1. Filtrer les données et statistiques d'une station sélectionnée sur base de critères tels que :
   — Afficher dans un tableau des données filtrées selon un certain seuil défini par l'utilisateur, comme par exemple une date ou une valeur pluviométrique.
   — Afficher sur un graphique les donnée d'une certaine période définie par l'utilisateur.

   Il s'agit d'une fonctionnalité essentielle car elle permettrait à un utilisateur de réaliser des analyses plus poussées.

Fonctionnalités C

1. Permettre à un utilisateur d'envoyer un message de demande d'aide via le site web directement aux administrateurs des stations ou aux techniciens du site web en cas de problème technique.
2. Permettre la comparaison des graphiques d'une station avec d'autres stations (définies par l'utilisateur dans un menu déroulant).
3. Permettre la comparaison des tableaux de données d'une station avec d'autres stations ou différentes périodes (définies par l'utilisateur dans un menu déroulant).
4. Filtrer les différentes stations sur la carte sur base de critères tels que :
   — La localisation dans une certaine région
   — Le type de station
   — La comparaison de la moyenne, du minimum ou du maximum des données des stations avec un certain seuil sur une période définie par l'utilisateur.

5. Filtrer les données et statistiques d'une station sélectionnée sur base de critères plus poussés tels que l'affichage sur un tableau des données supérieures ou inférieures à un seuil.

Fonctionnalités W

1. La capacité d'exporter les données sous forme de tableaux ou de graphiques dans un document PDF, pour les utilisateurs désirant obtenir une version imprimable de ces formats de données.
2. Un système de messagerie interne entre administrateurs.

4.2 Besoins non-fonctionnels

Après avoir énuméré les besoins fonctionnels, nous allons discuter des besoins liés à l'implémentation de l'application. Il semble nécessaire d'énoncer ces besoins non-fonctionnels si on souhaite réaliser une application de qualité.

4.2.1 Flexibilité du projet

Notre travail étant un travail étudiant, il pourra potentiellement être ré-utilisé comme une base pour de futurs travaux d'étudiants dans les années à venir. Il est donc nécessaire pour nous d'utiliser des technologies faciles d'accès et simples d'utilisation. L'objectif est donc que notre travail soit le plus lisible et le plus maniable possible.

4.2.2 Indépendance du système d'exploitation et de l'environnement

Une propriété qui rendrait l'application plus fiable et facile d'utilisation serait de faire fonctionner l'application indépendamment du système d'exploitation et de l'environnement. Pour y parvenir nous avons décidé d'utiliser Docker. Docker est une technologie permettant de lancer des applications au sein de conteneurs virtuels, une technologie qui sera décrite plus en détails au travers de la section "5.3 Utilisation de Docker".

4.2.3 Interface utilisateur

Dans un premier temps, il est nécessaire que l'interface utilisateur soit facile d'accès et intuitive. Il est donc fortement souhaitable qu'un utilisateur comprenne rapidement comment fonctionne l'application et comment accéder aux données requises par cette dernière. Dans un second temps, il est important que chaque requête du client se suive d'une réponse rapide pour assurer une meilleure performance de l'application. Pour y parvenir, l'objectif est d'assurer que le transfert de données de la base de données vers l'interface utilisateur se réalise rapidement. Et pour terminer, l'objectif premier de l'application reste bien sûr l'accès à des données pluviométriques affichées sous forme de tableaux ou de graphiques. Il est donc sensé d'imposer que les graphes et tableaux de données doivent pouvoir être lisibles et compréhensibles pour les utilisateurs. Il a donc été impératif pour nous de trouver une solution répondant à ce besoin.

4.3 Contraintes

Après avoir établi les besoins fonctionnels et non fonctionnels, nous allons nous intéresser aux contraintes liées au projet.

Une première contrainte imposée est la gestion de la qualité des connexions Internet d'Haïti. Nous allons devoir prendre en compte des connexions possiblement plus faibles en adaptant les priorités des différentes fonctionnalités. Il se trouve que le plus lourd à faire charger est la carte interactive, et il sera donc peut-être nécessaire de permettre à un utilisateur n'arrivant pas à afficher la carte interactive de pouvoir malgré tout accéder aux données d'une station sans forcement afficher la carte. Dès lors, nous devrions permettre d'afficher la liste des stations en priorité avant de pouvoir afficher complètement la carte si celle-ci venait à s'avérer être trop lourde.

En ce qui concerne les graphiques, nous devrons probablement afficher les données sous forme de tableaux avant de pouvoir afficher les graphiques qui seront sans doute plus difficiles à afficher.

Durant la réalisation de l'application, nous simulerons avec un outil adéquat une qualité d'Internet semblable à celle d'Haïti pour pouvoir tester les cas les plus contraignants, et présenterons le résultat de cette simulation à la section 8.4.

4.4 Récits utilisateurs

Pour l'analyse du problème, il est important de pouvoir déterminer avec exactitude les besoins d'un utilisateur de l'application. Pour y parvenir, nous allons décrire ici un ensemble de récits utilisateurs, ou user stories, afin de décrire de manière simple les attentes et besoins de différents types d'utilisateurs.

Un récit utilisateur est une phrase courte décrivant un utilisateur exécutant une fonctionnalité de l'application. Ces récits permettent de mieux visualiser les besoins d'un utilisateur réel et donc des différentes fonctionnalités à implémenter.

Pour cela, nous commençons par décrire ce que nous considérerons être les 4 différents types d'utilisateurs qui vont interagir avec l'application web :

Utilisateur : Le simple visiteur qui veut se renseigner sur les données pluviométriques.

Responsable de station(s) : Le responsable de station(s) pouvant modifier les données de la base de données. Nous y ferons aussi référence au travers du mémoire avec le terme équivalent "Gestionnaire de station(s)".

Administrateur : La personne qui modère l'application web dans sa globalité.

Technicien informatique : Celui qui s'occupe de l'entretien technique de l'application web. Les techniciens informatiques auront accès au code du projet.

Utilisateur

1. En tant qu'utilisateur, je peux afficher une carte d'Haïti sur laquelle sont répertoriées les différentes stations.
2. En tant qu'utilisateur, je peux faire apparaître une section d'informations sur laquelle apparaît le nom de la station en cliquant dessus.
3. En tant qu'utilisateur, je peux cliquer sur un nom de station pour accéder à ses données pluviométriques.
4. En tant qu'utilisateur, je peux afficher les moyennes, maximums et minimums des données, ainsi que les intensités pluviométriques spécifiques à une station sous forme de graphiques.
5. En tant qu'utilisateur, je peux afficher les moyennes, maximums et minimums des données, ainsi que les intensités pluviométriques spécifiques à une station sous forme de tableaux.
6. En tant qu'utilisateur, je peux sélectionner la période (jour, semaine ou année) de données pluviométriques que je souhaite prendre en compte dans l'affichage de données et statistiques.
7. En tant qu'utilisateur, je peux envoyer un message d'aide à un technicien informatique en cas de problème technique avec le site web.

Responsable de station

1. En tant que responsable de station, je peux réaliser les mêmes tâches qu'un utilisateur.
2. En tant que responsable de station, je peux afficher les données brutes spécifiques à une station sous forme de tableaux.
3. En tant que responsable de station je peux me connecter sur le site web.
4. En tant que responsable de station, je peux modifier les stations affiliées sur mon compte.
5. En tant que responsable de station, je peux importer de nouvelles données relatives à mes stations affiliées à la base de données, notamment au format xlsx.
6. En tant que responsable de station, je peux modifier ou supprimer des données relatives à une station

Administrateur

1. En tant qu'administrateur, je peux réaliser les mêmes tâches qu'un utilisateur.
2. En tant qu'administrateur, je peux afficher les données brutes spécifiques à une station sous forme de tableaux.
3. En tant qu'administrateur, je peux créer un compte pour un autre administrateur, technicien ou responsable de station.
4. En tant qu'administrateur, je peux éditer le rôle d'un membre du site en lui attribuant son nom, son rôle et ses éventuelles stations affiliées.
5. En tant qu'administrateur, je peux me connecter sur le site web.
6. En tant qu'administrateur, je peux ajouter une station ou la supprimer.

7. En tant qu'administrateur, je peux éditer les informations d'une station pluviométrique.
8. En tant qu'administrateur, je peux importer des données pluviométriques relatives à une station dans la base de données, notamment au format xlsx.
9. En tant qu'administrateur, je peux supprimer des données pluviométriques relatives à une station de la base de données.
10. En tant qu'administrateur, je peux modifier des données pluviométriques relatives à une station de la base de données.
11. En tant qu'administrateur, je peux rechercher certaines données pluviométriques spécifiques dans la base de données selon des critères tels que le nom de la station ou la date de mesure.

Technicien informatique

1. En tant que technicien informatique, je peux réaliser les mêmes tâches qu'un utilisateur.
2. En tant que technicien informatique, je peux recevoir et lire les messages d'aide envoyés par n'importe quel utilisateur.
3. En tant que technicien informatique, je peux modifier les données de la base de données avec l'accord d'un administrateur.
4. En tant que technicien informatique, je peux supprimer les données de la base de données avec l'accord d'un administrateur.
5. En tant que technicien informatique, je peux ajouter des données dans la base de données avec l'accord d'un administrateur.
6. En tant que technicien informatique, je peux accéder au code du site web et le mettre à jour avec l'accord d'un administrateur.

Chapitre 5

Analyse technique

Avant de nous lancer dans l'implémentation, il nous a fallu faire le choix des technologies qui allaient servir à créer l'application web. Pour ce faire, nous nous sommes penchés sur plusieurs frameworks web populaires, et avons cherché à sélectionner un framework qui répondrait à nos besoins tout en alliant la caractéristique d'être simple à prendre en main, que ce soit pour nous ou pour les futures personnes qui travailleront sur l'application web.

5.1 Choix du framework

Commençons par définir ce qu'est un framework, qui peut se traduire en français par structure logicielle ou infrastructure de développement mais dont nous préférerons l'appellation anglaise au travers de ce mémoire. Un framework se définit de la manière suivante : "Ensemble d'outils constituant les fondations d'un logiciel informatique ou d'applications web, et destiné autant à faciliter le travail qu'à augmenter la productivité du programmateur qui l'utilisera." [13].

On peut d'ailleurs diviser les frameworks en deux catégories, "frontend" et "backend". De manière générale, la partie d'une application ou d'un site web avec laquelle un utilisateur va interagir est ce que l'on nomme frontend ou "frontal" en français. L'autre partie d'une application ou d'un site web est donc le backend ou "arrière-plan" en français, concernant tout ce qui s'occupe de stocker et de gérer les données [14]. On peut donc choisir de faire du développement backend ou frontend selon le framework qu'on choisira. Mais il s'agit au final des deux aspects différents d'une même situation.

L'utilisation d'un framework va s'imposer comme une évidence, puisque l'ensemble d'outils et de librairies fourni par celui-ci permettra d'améliorer significativement notre productivité lors du développement. Il convient donc de choisir un framework adapté à notre projet. De nombreuses possibilités s'offrent à nous, et nous allons détailler ci-dessous les quelques frameworks qui ont attiré notre attention lors de nos recherches.

5.1.1 Frameworks frontends

Dans un premier temps, nous nous sommes penchés sur les différentes alternatives possibles parmi les frameworks frontends. Nous allons analyser ci-dessous trois des plus populaires à l'heure actuelle, selon certains comparatifs [15] [16].

AngularJS Tout d'abord, il y a AngularJS, un framework JavaScript open-source très répandu et populaire développé par Google en 2009 [17]. Grâce à sa popularité, un des premiers attraits d'AngularJS est la quantité d'informations disponibles sur le net du fait de sa communauté active.

Étant un framework JavaScript, AngularJS est donc par extension un framework orienté frontend, qui se spécialise dans la création d'applications dites "monopages", dont l'objectif est d'éviter le chargement d'une nouvelle page après une interaction de l'utilisateur. Le framework utilise une combinaison de codes HTML et TypeScript [18] pour créer les pages web. TypeScript étant ce qu'on appelle un sur-ensemble syntaxique de Javascript permettant notamment de sécuriser la conception de code JavaScript [19].

D'un côté, AngularJS peut présenter des avantages : par exemple, certains mettent en avant son approche plus structurée à base de composants, et sa façon plus claire d'échanger les données entre les composants [20]. De l'autre, un des premiers désavantages majeur d'AngularJS est sa taille comparée à d'autres frameworks JavaScript, comme ReactJS ou VueJS [21]. Aussi, les développeurs web doivent apprendre et devenir familier avec TypeScript [18] avant toute chose, et aucun de nous deux ne connaît le langage. Ce dernier point s'avère assez important dans le choix de notre framework comme vous le verrez au cours du comparatif, car nous souhaitions perdre le moins de temps possible avant d'entamer la programmation du site.

ReactJS Ensuite, nous pouvons parler de ReactJS, un librairie JavaScript open-source développé par Facebook en 2013 [22]. ReactJS est aussi très populaire et répandu, et bénéficie donc tout autant qu'AngularJS d'une communauté très active, mettant à notre disposition beaucoup d'informations [23].

Tout comme AngularJS, la spécialité de ReactJS est le développement frontend. Aussi, il utilise le langage JSX [24], qui est une extension syntaxique de JavaScript uniquement là pour faciliter l'expérience de développement, tout en se reposant aussi sur la notion de composants comme AngularJS présenté plus tôt, ou VueJS que nous allons présenter dans la section suivante [25]. ReactJS étant même le premier à adopter cette architecture basée sur les composants. Enfin, un point intéressant à souligner est que ReactJS est une libraire, et non pas un framework à proprement parler, bien qu'on l'y compare souvent.

Une librairie, aussi appelée bibliothèque logicielle, est considérée comme "un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire. Les fonctions sont regroupées de par leur appartenance à un même domaine conceptuel (mathématique, graphique, tris, etc)." [26]

Bien que les frameworks et les librairies semblent très similaires, de par le fait que les deux sont constitués de code écrit par d'autres personnes qui est utilisé pour aider à résoudre des problèmes communs. La différence majeure d'une libraire avec un framework vient du fait que

lorsque l'on utilise une librairie logicielle, nous décidons quand et où appeler les fonctions de la librairie. Alors que lorsqu'on utilise un framework, celui-ci sera plus restrictif en fournissant des emplacements précis où notre code doit être "branché", et c'est lui qui appellera le code que nous avons ajouté au besoin.

D'un côté, ReactJS peut présenter certains avantages, comme le fait qu'il se concentre sur le fait d'afficher des "vues" rapidement. Le fait que la librairie ReactJS aide à la génération complète du code des côtés aussi bien serveur que client est un point fort, contrairement à AngularJS qui ne s'occupe que du côté client. De l'autre, un des désavantages majeur de ReactJS est, aussi bien que pour AngularJS, l'utilisation d'un langage de programmation que nous ne maîtrisons pas plus que ça, à savoir le JSX. Nous avons aussi lu que sa prise en main peut s'avérer assez difficile et coûteuse en termes de temps. Enfin, il ne s'agit que d'une libraire et non d'un framework, et donc ReactJS manque par exemple d'un modèle de gestion routeur, ce qui n'est pas le cas de VueJS ou AngularJS [27].

VueJS VueJS est le troisième framework JavaScript open-source basé sur l'utilisation de composants que nous présenterons ici [25]. Il a été développé en 2014 par Evan You. Des trois options JavaScript, VueJs est celle qui a le plus retenu notre attention. en effet, le framework est un peu considéré comme une étoile montante et un des framework JS les plus populaires.

Pour citer plusieurs avantages de VueJS, on peut commencer par souligner une courbe d'apprentissage jugée plus rapide que ReactJS, notamment de par le fait que VueJS utilise comme langage JavaScript et non une sur-couche syntaxique comme JSX pour ReactJS ou TypeScript pour AnuglarJS. Un autre avantage fort de VueJS concerne les performances, puisque le framework est avant tout de très petite taille [21] [25], et qu'il se vante de requérir des efforts d'optimisation minimum [25].

Concernant ses désavantages, le plus gros que nous pouvons retenir contre VueJS est sa "nouveauté", ce qui fait qu'il n'y a pas encore beaucoup de documentation ou de "plugins" disponibles. Aussi, bien que nous ayons des bases en JavaScript, il ne s'agit toujours pas là d'un de nos langages de prédilection.

5.1.2 Frameworks backends

Dans un second temps, nous nous sommes ensuite intéressés aux frameworks orientés backend. Ceux-ci présentent des avantages différents et sont souvent aussi populaires et répandus que les précédents.

Ruby on Rails Ruby on Rails est le premier framework backend sur lequel nous nous sommes penchés. En effet, ce framework backend open-source est de plus en plus en vogue [23], tout en étant réputé pour sa structure permettant un développement plus rapide et intuitif. Ce framework est de type modèle-vue-contrôleur (MVC) [28] [29], et ce type de framework est plutôt idéal pour commencer facilement le développement web.

Une architecture Modèle-Vue-Contrôleur est un patron de conception orienté pour les interfaces graphiques [30]. L'objectif de cette architecture logicielle est de séparer la logique du code en trois parties distinctes, à savoir [31] :

— Modèle : Il s'agit de la partie qui gère les données de l'application. Elle a pour objectif de permettre d'interagir avec la base de données, et de par exemple faciliter l'extraction, l'insertion ou l'organisation et le tri de ces données.
— Vue : Il s'agit de la partie qui gère l'affichage des données. La vue a pour rôle principal de récupérer les données depuis le modèle pour les variables qui devront être affichées.
— Contrôleur : Il s'agit de la partie qui fait le lien entre le modèle et la vue, le contrôleur va contenir la logique du code. le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue.

Cette séparation en trois parties va aider à savoir quels fichiers créer et comment correctement les définir.

Comme nous l'avons dit, un des avantages de Ruby on Rails est encore une fois sa popularité, donnant donc accès à de nombreuses ressources et informations sur le net concernant le framework. Le framework a par exemple été utilisé pour des sites tels que Airbnb, GitHub, Twitch ou Soundcloud. De plus, le framework est considéré comme très simple à prendre en main, ce qui permet de démarrer le développement web rapidement pour une petite équipe comme la nôtre.

D'autres avantages au framework peuvent être soulignés, comme sa très grande flexibilité ou les nombreuses libraires ou "gems" disponibles pour développer l'application. D'autre part, nous avons été refroidis dans notre choix par le simple fait que Ruby est une fois encore un langage de programmation dans lequel nous n'avons pas encore fait nos marques. Aussi, nous avons lu que Ruby ne semblait pas proposer certaines fonctionnalités, comme le multithreading qui permet d'exécuter plusieurs fils d'exécutions en même temps, qui pourraient s'avérer intéressantes pour les calculs du projet.

Django Finalement, nos recherches finirent par aboutir sur Django. Tout comme Ruby on Rails, Django est un framework backend open-source [32]. Dans le cas de Django, il se base sur le langage Python.

Tout d'abord, il présente aussi l'avantage d'être un framework de type modèle-vue-contrôleur (MVC), que l'on qualifie même généralement plutôt de MVT, ou Modèle-Vue-Template, une variante du MVC qui sera expliquée dans la partie 7 sur l'implémentation.

De plus, parmi tous les frameworks présentés, Django est le plus ancien d'entre eux. Créé en 2003 par Adrian Holovaty et Simon Willison, et ensuite paru officiellement en 2005, le framework a donc aujourd'hui déjà 16 années derrière lui [32]. Cela fait de Django un des frameworks qui a connu le plus de développement de la part de la Django Software Foundation, et en fait un framework très bien établi à l'heure actuelle. Cela peut se remarquer en termes du nombre de recherches effectuées parmi les différents frameworks sur Google, en termes de questions liées à Django sur Stackoverflow ou en termes de répertoires liés à Django sur Github [16].

Il a par exemple été utilisé par de grands noms tels que Google, Youtube ou Instagram. Sa popularité est donc aussi très grande, et donc la quantité de ressources disponibles pour le framework est élevée. Il supporte les schémas ORM de base de données et fournit un système simple de gestion de base de données. De plus, le framework est très personnalisable comparé à d'autres.

L'ORM ou Object-Relational Mapping se définit comme suit :

> " Il s'agit d'une technique de programmation informatique qui permet de simplifier l'accès à une base de données en proposant à l'informaticien des « objets » plutôt que d'accéder directement à des données relationnelles. "[34]

Enfin, Django a l'avantage de proposer beaucoup de fonctionnalités très utiles "out of the box", comme un système d'authentification efficace ou un système de messagerie.

5.1.3 Technologie retenue

Avant de motiver notre choix de technologie retenue, voici ci-dessous un résumé comparatif des différentes technologies présentées. Les critères sont les suivants :

- **MVC :** On regarde si le framework utilise une architecture Modèle-Vue-Contrôleur, car ce critère nous semble important. En effet, nous sommes plus familiers avec ce genre de structure.
- **Difficulté d'utilisation et courbe d'apprentissage :** Notée de simple à élevée, la difficulté perçue du framework par ses utilisateurs va jouer un rôle important dans la décision.
- **Popularité :** Enfin, la popularité du framework est importante à prendre en compte si l'on souhaite utiliser des libraires et plugins. Comme vous le verrez, ils sont tous très populaires (une note de A indiquant une très bonne valeur dans le système de notation anglo-saxon), à l'exception de ReactJS qui semble présenter une popularité encore un peu plus élevée que les autres (d'où sa note de S) [23]

Framework	Langage	MVC	Difficulté	Popularité
AngularJS	TypeScript	non (MVVM)	élevée	A
ReactJS	JavaScript, JSX	non	moyenne	S
VueJS	JavaScript	non	simple	A
Ruby on Rails	Ruby	oui	élevée	A
Django	Python	oui	moyenne	A

Table 5.1 – Résumé comparatif des technologies

En définitive, nous nous sommes donc tournés vers Django comme choix de framework pour le développement de notre application web. Le framework backend semblait présenter tous les points forts que nous recherchions pour atteindre nos objectifs. Notamment au travers de toutes les libraires et plugins disponibles de par sa popularité, mais aussi grâce au format MVC. De plus, le fait que Django utilise Python a été décisif dans notre choix.

5.2 Choix du SGBD

Un SGDB, ou Système de gestion de base de données, est un logiciel générique permettant de réaliser facilement des bases de données, d'accéder aux informations, d'assurer la sécurité, la confidentialité, l'archivage ou la restauration des informations [35] [36]. Il sera indispensable au bon fonctionnement de notre application web. En effet, une quantité importante de données se verra être stockée dans la base de données et sera sollicitée pour les différentes statistiques. Il convient donc de choisir un SGBD performant pour gérer nos données.

5.2.1 Relationnel ou non-relationnel

Les systèmes de gestion de base de données sont à séparer en deux catégories ; il existe actuellement en développement web les bases de données relationnelles et non-relationnelles. Ce qui différencie principalement les deux est le type de structure utilisée.

Dans un cas, les bases de données relationnelles utilisent des tables dans lesquelles les données sont stockées, et ces tables sont reliées les unes aux autres, d'où le nom de "base de données relationnelle". Alors que dans l'autre cas, les bases de données non-relationnelles sont basées autour de "documents" ou "fichiers". Ces bases de données offrent plus de liberté quant à la forme des documents stockés, car contrairement aux tables utilisées par les bases de données relationnelles qui n'acceptent qu'une seule forme de données, les bases de données non-relationnelles peuvent stocker les informations sous différentes catégories.

Nous allons ci-dessous détailler les points forts et les points faibles des deux types de SGBD, afin de motiver notre choix.

5.2.2 SGBD relationnel

Commençons par nous pencher sur les SGBD relationnels. Ceux-ci sont indéniablement les plus répandus, et ce d'après le classement effectué par DB-Engines se reposant sur de nombreux indicateurs [37]. Nous pouvons alors en lister quelques-uns considérés comme les plus populaires, comme présenté dans un article les comparant sur plusieurs critères [38] :

— Oracle
— MySQL
— Microsoft SQL Server
— PostGreSQL
— SQLite
— MariaDB

Avantages

Les systèmes de bases de données relationnelles possèdent de nombreux avantages que nous allons détailler ci-dessous, selon diverses références [35] [39] [40] :

— **Modèle simple :** La structure d'une telle base de données est généralement simple et uniforme aussi bien pour son développement, puisqu'elle ne requiert pas une architecture fastidieuse ou complexe, que pour son utilisation, car les requêtes sont simples à écrire de part la syntaxe de SQL qui est standardisée. C'est une caractéristique plus que souhaitable étant donné que nous développons cette application web pour d'autres, nous

voulons donc une structure de bases de données aussi simple et facile à comprendre que possible.

— **Sécurité :** Dans un modèle relationnel, il est possible de gérer simplement les niveaux de sécurité des différentes tables, par exemple en décidant quels types d'utilisateurs auront accès à quelles tables ou en décidant quelles tables seront confidentielles. C'est une propriété intéressante puisque nous voulons différencier un visiteur "lambda" d'un gestionnaire de station pluviométrique ou d'un administrateur de l'application.

— **Intégrité des données :** Un autre avantage des bases de données relationnelles est qu'elles vont systématiquement vérifier la validité des données qui seront ajoutées. Puisque les données ne sont pas indépendantes mais qu'elles obéissent à des règles sémantiques, un SGBD relationnel va contrôler qu'aucune contrainte n'a été violée après chaque ajout de données. C'est une fois encore plus que souhaitable dans notre cas.

— **Pas de duplication :** Une telle base de données n'autorise simplement pas de doublons, ce qui rend le tout plus efficace et rapide. On ne perdra pas de temps sur des données stockées en double.

— **Flexibilité :** Ils sera facile à l'avenir de créer de nouvelles relations entre des tables existantes, par exemple si l'on cherchait à calculer et stocker de nouvelles statistiques.

Désavantage(s)

Un des seuls désavantages qui semble apparent est le suivant :

— **S'adapte mal aux données "complexes" :** Un schéma relationnel n'est souvent pas le choix le plus adapté pour stocker des données complexes en terme de structure. En effet, il est parfois très difficile d'efficacement représenter des données par des tables, et on risque alors par exemple de perdre grandement en performance.

5.2.3 SGBD non-relationnel

Avantages

Contrairement au base de données "SQL", les bases de données non-relationnelles ou "no-SQL" ne dépendent pas de la complexité des données et présentent donc l'avantage de pouvoir stocker un peu tout "format" de données dans ce qu'on appelle des documents. Dès lors, choisir un SGBD non-relationnel s'avère être un choix plus adapté lorsque l'on souhaite stocker des données aux formats plus ouverts/flexibles.

On peut aussi citer la lisibilité accrue des données, puisqu'elles sont stockées dans des documents et non plus séparées dans de multiples tables différentes.

Désavantages

Contrairement aux SGBD relationnels qui utilisent tous le même langage, à savoir SQL, ce n'est pas le cas des SGBD non-relationnels et il faudra donc se plier au langage utilisé par le SGBD qui serait choisi. De plus, étant donné que les SGBD SQL sont là depuis beaucoup plus longtemps que les bases de données non-SQL, on peut donc constater que ces dernières sont "moins matures" et moins bien finies en terme de fonctionnalités que ces premières.

5.2.4 Technologie retenue

Tout d'abord, il est important de noter que notre choix entre SGBD relationnel et non-relationnel a été en partie conditionné par notre choix précédent de framework. En effet, les bases de données non-SQL ne sont pas officiellement supportées par Django, celui-ci se basant uniquement sur des bases de données relationnelles.

Toutefois, il est intéressant de souligner qu'il n'est pas impossible du tout d'utiliser des bases de données non-relationnelles avec Django, car il existe déjà certains projets résolvant ce problème de compatibilité et permettant des fonctionnalités non-SQL au sein de Django. Le projet "Django non-rel" en est un très bon exemple [41].

En-dehors de cela, notre choix s'est quoi qu'il arrive porté en faveur d'une base de données relationnelles. Pour raisons principales, les avantages cités précédemment sont exactement les propriétés qui conviendraient aux types de données que nous voulons stocker. Les forces principales des bases de données relationnelles étant leur efficacité, leur conformité aux propriétés ACID [1].

De plus, un point important est que les données pluviométriques ne requièrent pas du tout de format particulier pour être stockées, ce qui fait perdre son intérêt à l'utilisation d'un schéma non-relationnel. Au contraire, elles se prêtent parfaitement au stockage dans des tables puisque de telles données seront constituées de champs simples tels que :

— Une date
— Une heure
— Un nom de station
— Une mesure en millimètres
— etc

Maintenant que nous sommes fixés sur le type de SGBD, nous avons le choix entre de nombreux SGBDR, dont par exemple les cinq que nous avions cités plus haut qui représentent les plus populaires. Ceux-ci ont l'avantage d'être tous officiellement supportés par Django "out of the box".

Un d'entre eux a particulièrement retenu notre attention, et il s'agit de PostGreSQL. En effet, nous avons pu constater par exemple que grâce à sa capacité de processus parallèle supérieure, PostGreSQL sort vainqueur en termes de vitesse par exemple dans des applications analytiques [43] [44]. Ce genre de propriété a beaucoup retenu notre attention, étant donné que notre application web sera quasiment entièrement tournée vers du traitement analytique et statistique. De plus, il est à noter que PostGreSQL est open source.

1. En informatique, les propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité) sont un ensemble de propriétés qui garantissent qu'une transaction de données se déroule de manière fiable. [42]

5.3 Utilisation de Docker

Au travers de ce mémoire, nous avons décidé de nous reposer sur Docker, une technologie logicielle permettant de lancer des "applications" au sein de "conteneurs" [45]. Un conteneur consiste en l'ensemble d'un logiciel avec ses dépendances, comme par exemple des librairies, que l'on aurait regroupées. En ce sens, les conteneurs peuvent être vus comme des machines virtuelles améliorées.

De manière générale, nous pensons que l'ajout de Docker au projet sera bénéfique car son plus gros avantage est que nous pourrons ainsi nous assurer que l'application que nous développons **fonctionnera indépendamment aussi bien du système d'exploitation que de l'environnement**, puisque tout cela est géré dans les "conteneurs".

Il y a de nombreux autres avantages à l'utilisation de Docker et de ces conteneurs, et pour n'en citer que quelques-uns qui ont motivé notre décision à utiliser la technologie [46] [47] :

1. **Léger :** Au contraire des Machines Virtuelles, la majeure partie de "l'overhead" du système d'exploitation est enlevé, rendant la technologie plus légère mais aussi plus rapide qu'une VM puisqu'il n'y a pas d'OS à "proprement" faire tourner. En effet, Docker va partager le kernel du système d'exploitation déjà présent.
2. **Extensibilité :** Docker est favorable à l'extensibilité ou "scalabilité" de l'application, puisque d'une part il est aisé de fixer les ressources qu'utilise un conteneur et de les augmenter si besoin, par exemple en termes de puissance de calcul CPU ou de RAM disponible, et d'autre part la taille même du conteneur peut être changée et agrandie si l'application venait à prendre plus d'ampleur.

5.4 Licence

Il est enfin important que nous nous penchions sur la licence qui va encadrer notre code, afin de définir les conditions dans lesquelles notre application pourra être utilisée.

Tout d'abord, ce projet utilisera une licence open-source et ce, d'une part car une telle licence nous semble être le choix le plus cohérent à faire, étant donné qu'il s'agit d'un travail universitaire qui va probablement connaître encore plusieurs itérations avant d'arriver à terme. De plus, étant donné que ce travail est en collaboration avec l'université d'État d'Haïti, il nous semble logique de viser à leur laisser un maximum de liberté quant à ce travail.

D'autre part, comme vous le verrez dans le chapitre 7 sur l'implémentation, nous allons avoir recours à plusieurs technologies et librairies elles-mêmes open-sources, et qui se verront intégrées dans le travail.

Ces différentes technologies et libraires open-source quant à elles ont choisi différents types de licences, certaines ont opté pour une licence MIT comme Chart.JS [66] ou DataTables [60], et d'autres ont opté pour FreeBSD (aussi connu sous Licence 2-clause BSD) comme OpenLayers [59]. Tout cela sera donc détaillé plus loin dans la section 7.2. Mais une des technologies que nous allons utilisé, à savoir Django-Jet [67], est un plugin open-source sous licence AGPLv3.

Là où d'une part, l'utilisation d'un logiciel sous licence "2-clause BSD" est complètement libre :

" Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met :

1. *Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.*
2. *Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.[48]*

D'autre part, l'utilisation d'un logiciel sous licence AGPLv3 peut quant à elle s'avérer un peu différente. En effet, la section 13 de la licence stipule que si le code open-source est modifié, alors il doit être accessible et rendu publique :

" Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. " [68]

Mais dans notre cas, étant donné que le seul code open-source sous licence AGPLv3 que nous utiliserons est Django-Jet[67] et que nous ne modifierons en aucun cas le code de ce plugin, nous ne sommes alors pas contraints par cette section de la licence.

Dès lors, nous sommes libres de choisir le type de licence open-source qui nous conviendrait le mieux et qui serait le plus bénéfique au projet. C'est pour cela que nous avons opté pour une licence MIT.

En effet, la licence MIT est un licence très courante parmi les logiciels open-sources, et elle va droit au but de ce que nous voudrions ; une licence simple qui permettra à d'autres personnes de faire à peu près ce qu'ils veulent du projet, comme l'améliorer pour en faire des versions à code source fermé à l'avenir.

Cette licence très permissive sera donc parfaite pour toute personne souhaitant reprendre et améliorer le projet. Voici un extrait du texte de cette licence :

" Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software ..." [50]

Chapitre 6

Application web

6.1 Présentation des différentes pages

Au travers du chapitre qui va suivre, nous allons discuter du site web et plus particulièrement des interfaces utilisateurs. Nous allons illustrer toutes les différentes interfaces du site web avec des photos et décrire chacune d'entre elles.

6.1.1 Page d'accueil

FIGURE 6.1 – Page d'accueil

Sur la figure 6.1 se trouve la page d'accueil de l'application web. On y retrouve tout d'abord un petit texte de présentation et d'aide. Au dessus de la page se trouve une barre de navigation qui permet à un utilisateur de parcourir les différentes pages du site web. Les différents onglets et leurs contenus seront décrits dans les sous-sections suivantes.

6.1.2 Carte interactive d'Haïti

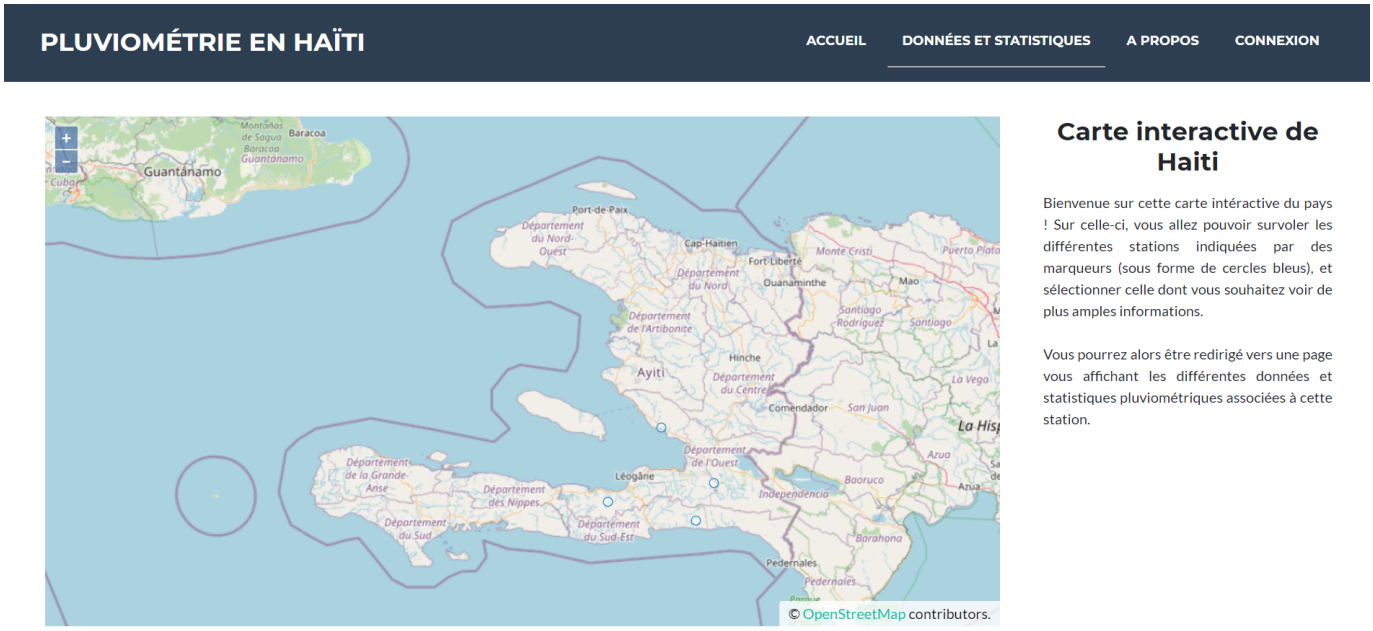


FIGURE 6.2 – Page web de la carte d'Haïti

Si un utilisateur sélectionne l'onglet "Données et statistiques", il découvrira la page représentée à la figure 6.2. Il s'agit ici de la carte interactive d'Haïti. On retrouve à droite une petite description de la page. La carte est directement centrée sur le pays concerné. Pour chaque station enregistrée dans la base de données, un cercle bleu est représenté sur la carte aux coordonnées adéquates.

6.1.3 Analyse des données

Quand un utilisateur clique sur un des points affichés, il sera directement dirigé vers la page illustrée à la figure 6.3.

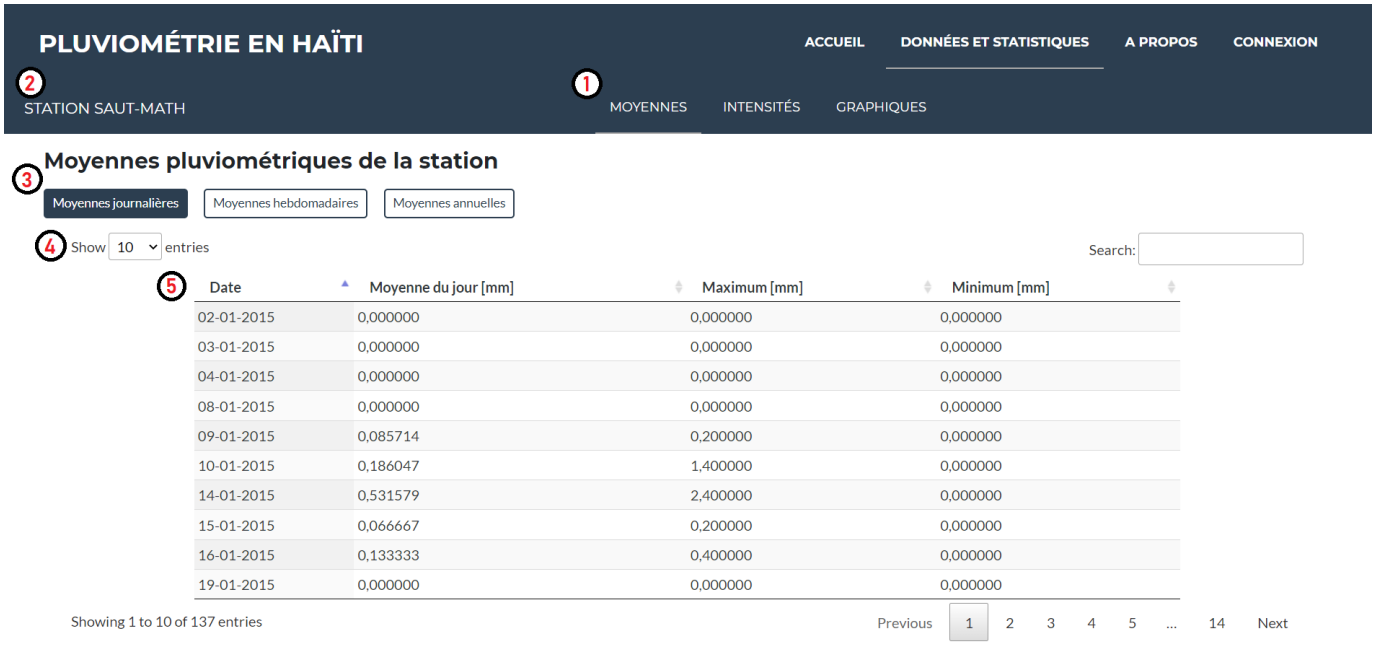


FIGURE 6.3 – Page web de la carte d'Haïti

Il s'agit de la page permettant d'afficher toutes les données relatives à une station particulière. Sur cette page on retrouve plusieurs éléments :

1. Tout d'abord, au point numéro 1 se trouve une nouvelle barre de navigation. Cette barre de navigation permet de naviguer parmi les différents modèles d'analyses. On retrouve deux types de modèles. Tout d'abord les tableaux de données et ensuite les graphiques de données. L'onglet des moyennes affiche les tableaux de moyennes des données sur 3 types de données différentes décrites au point 3. Dans l'onglet des intensités on retrouve un tableau regroupant toutes les intensités enregistrées pour la station. Le dernier onglet "Graphiques" dirige vers les graphiques des données qui seront décrits plus loin.

2. Au point numéro 2, on retrouve le nom de la station qui a été sélectionnée par l'utilisateur.

3. On retrouve une série de boutons au point 3. Ces boutons sont spécifiques à l'onglet des moyennes défini au premier point. Ils permettent de naviguer entre les différents tableaux des moyennes calculées pour la station sélectionnée. Il y a 3 types de tableaux de moyennes. Le tableau des moyennes quotidiennes, le tableau des moyennes hebdomadaires et le tableau des moyennes annuelles.

4. Au point 4 se trouve un champ qui permet d'afficher un nombre précis de données à la fois dans chaque tableau.

5. Et enfin au point 5, on retrouve le tableau de données sélectionné par l'utilisateur via la barre de navigation décrite au point 1. On peut donc y voir les données enregistrées dans la base de données relative à la station sélectionnée. Un champ de recherche en haut du tableau permet de filtrer dynamiquement les données pour pouvoir analyser des données particulières. On peut filtrer les données en fonction de n'importe quelle critère.

Portons maintenant un intérêt à la page des graphiques illustrée à la figure 6.4. Un utilisateur peut y trouver plusieurs choses :

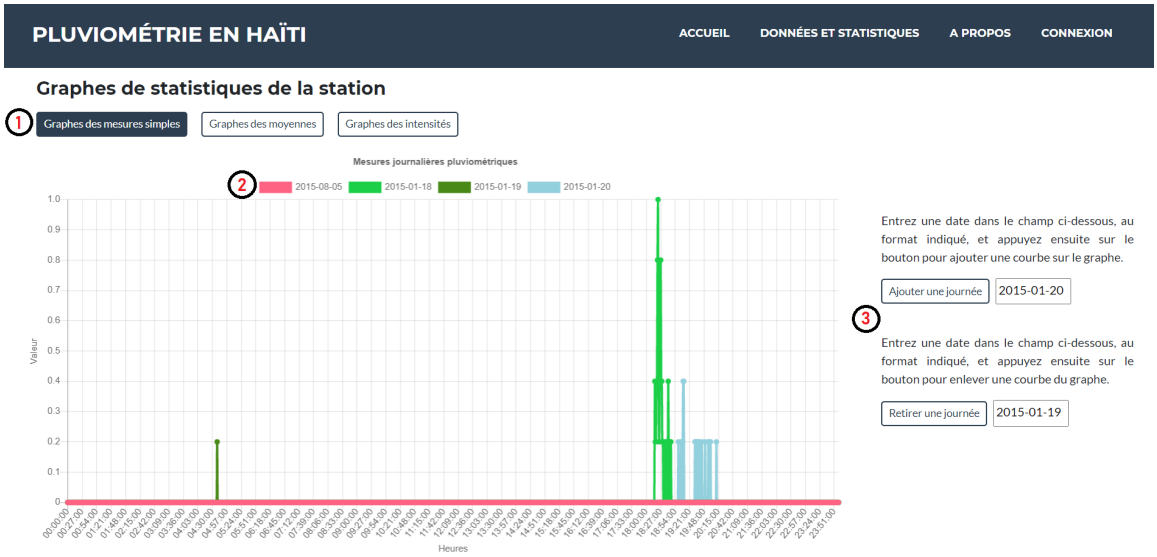


FIGURE 6.4 – Page du graphe des données quotidiennes

1. Premièrement, on retrouve au point 1, une nouvelle série de boutons propres aux graphes. Ils disparaissent si l'utilisateur retourne sur les tableaux de valeurs. Ils renvoient chacun vers un graphe différent. Le premier bouton renvoie vers le graphique des données brutes. Le graphique va afficher les données pour une journée précise. Lors du chargement de la page, les données de la dernière journée enregistrée dans la base de données seront automatiquement affichées sur le graphe. Le 2ème bouton renvoie vers les graphes des moyennes tandis que le dernier bouton renvoie vers le graphe des données d'intensité.
2. Ensuite, au 2ème point on retrouve le graphique sélectionné via les boutons décrits au premier point. Il s'agit ici sur la figure 6.4 du graphique des données brutes. Les autres graphiques seront décrits juste en dessous. Le graphique des données quotidiennes va donc placer en abscisse la totalité des minutes de la journée et afficher les valeurs correspondantes pour chaque abscisse. Les sets de données de chaque journée sont définis juste au dessus du graphe.

3. Enfin au point 3, on retrouve les boutons d'interactions avec la base de données pour ajouter les données à afficher ou les enlever. L'utilisateur peut soit ajouter une date en écrivant simplement celle ci dans le champs adéquat et en appuyant sur le bouton "ajouter une journée" ce qui fera apparaître tout le set de données correspondant sur le graphe. En effet, vu que le graphe de données affiche les données pour l'intégralité de la journée, ajouter une date va afficher sur le graphique les données de 0h00 à 23h59 de cette journée. Il peut ensuite choisir de retirer un set de données en réalisant la même action dans les champs prévus à cet effet.

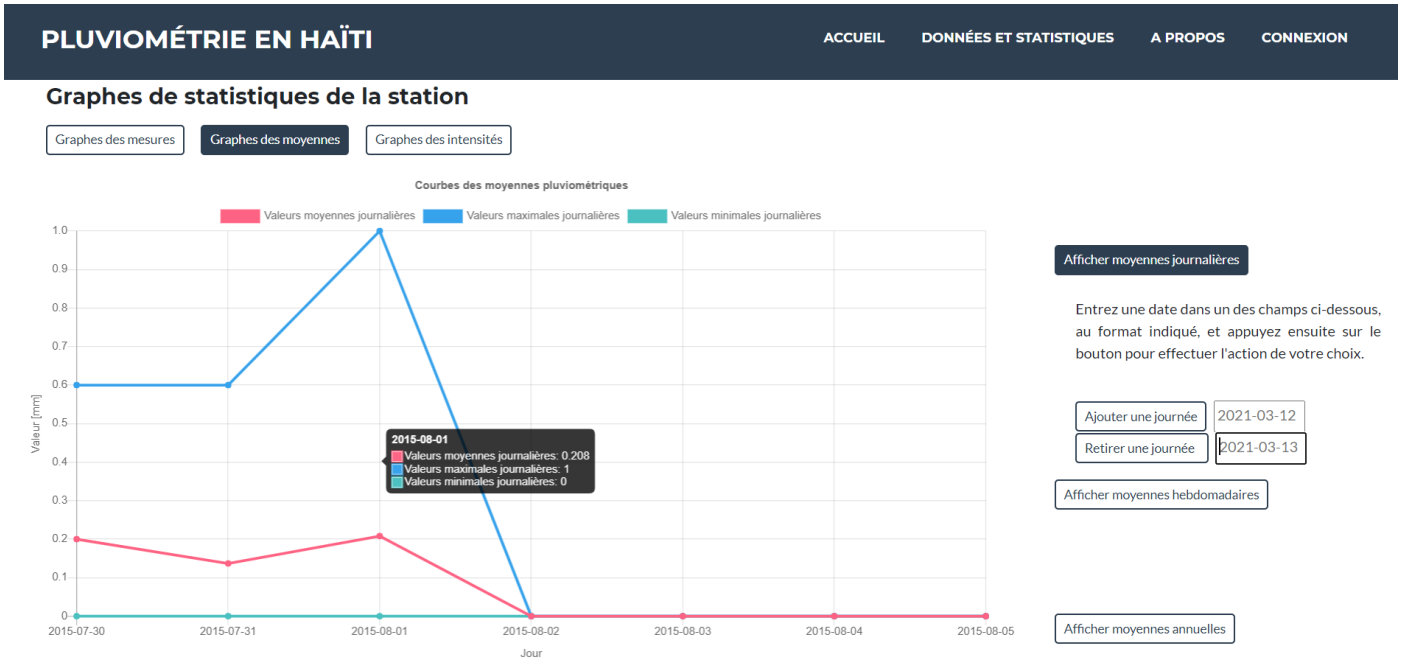


FIGURE 6.5 – Page du graphe des moyennes

4. Sur la figure 6.5, on peut voir le graphe des moyennes. A l'aide de boutons d'interactions, il y a la possibilité de passer d'un graphe de moyenne à l'autre. Les 3 types de graphes sont le graphe des moyennes quotidiennes, le graphe des moyennes hebdomadaires et le graphe des moyennes annuelles. Sur chacun des graphes, on ne retrouve que 3 sets de données exactement, il s'agit de la moyenne, de la donnée maximale et de la donnée minimale de la période de temps sélectionnée. Les périodes sont affichées en abscisses. Un utilisateur peut encore une fois afficher la journée de son choix dans les champs adéquats et appuyer sur le bouton "ajouter une journée" pour afficher les données de la date voulue sur le graphique.

   Aussi, sur ce graphe comme sur tous les graphes, quand un utilisateur survole un graphique avec la souris, les données de l'abscisse sélectionnée apparaissent dans une fenêtre d'information

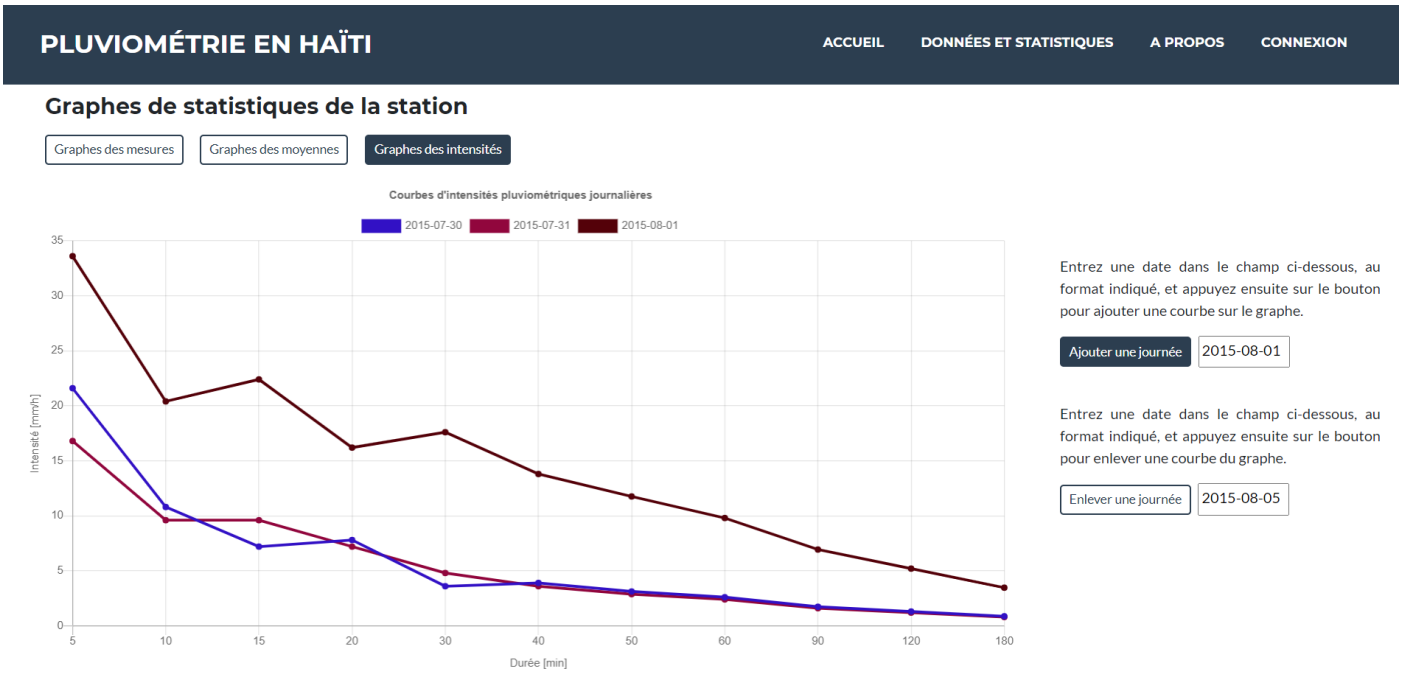


FIGURE 6.6 – Page du graphe des intensités

5. Le dernier type de graphe est le graphe des intensités. Sur l'axe des abscisses on retrouve une série de durées de temps représentant les différents temps de calculs d'intensités. Un utilisateur peut à nouveau sélectionner un jour à afficher dynamiquement sur le graphe ou à retirer à l'aide des champs et boutons d'interactions. Le bouton "ajouter une journée" permet d'afficher un set de données supplémentaires sur le graphe. les données de ce set représente les différentes intensités quotidiennes pour chaque durée prédéfinie pour le jour sélectionné. Le bouton "enlever une journée " permet de retirer du graphe l'entièreté des données spécifiques à la journée sélectionnée. Ces boutons d'interactions permettront à l'utilisateur de comparer les intensités de journées différentes pour une même station sur un graphe.

6.1.4 A propos

FIGURE 6.7 – Page à propos

Dans l'onglet "à propos" on retrouve une page d'information sur l'application web regroupant entre autre le contexte du projet ainsi que quelques explications sur le fonctionnement du site.

6.1.5 Contact

FIGURE 6.8 – Barre de bas de page

En bas de toutes les pages présentées jusqu'à présent, on retrouve une barre d'informations illustrée à la figure 6.8 contenant l'adresse de la FAMV (Faculté d'agronomie et de Médecine vétérinaire) mais aussi un moyen de contacter les administrateurs du site web.

FIGURE 6.9 – Page de contact

Une fois qu'un utilisateur sélectionne le bouton " Contact " il est automatiquement envoyé vers la page des contacts illustrée à la figure 6.9. Sur cette page, il est possible pour un utilisateur de contacter un administrateur de l'application pour signaler un problème ou pour poser des questions. Il suffit de remplir les champs à disposition et d'appuyer sur un bouton d'envoi. Le mail actuellement utilisé est celui d'un membre des développeurs. Mais l'adresse par défaut peut être modifiée dans le fichier de paramètres. Une meilleure description a été réalisée dans le chapitre sur l'implémentation.

6.1.6 Connexion

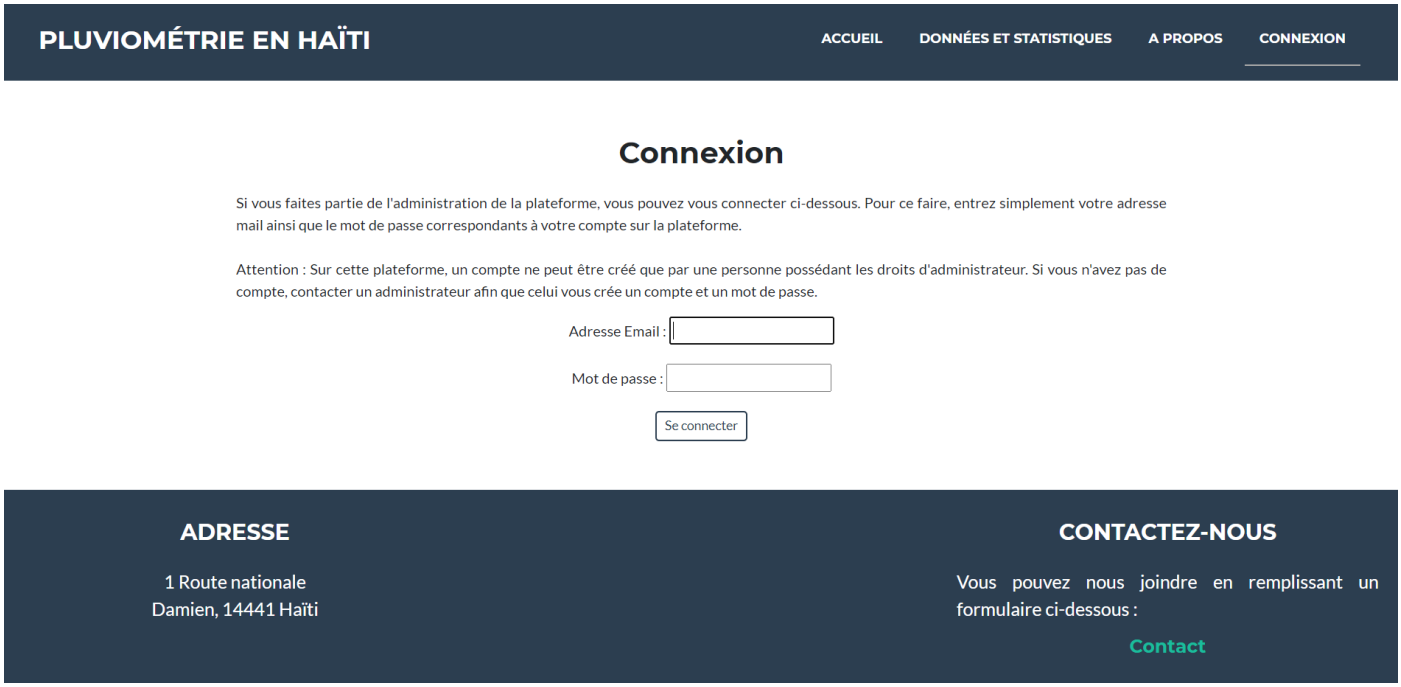


FIGURE 6.10 – Page de connexion

Le dernier élément de la barre de navigation conduit à une page de connexion où seuls les gestionnaires de stations ou administrateurs peuvent se connecter. Cette page se trouve à la figure 6.10. Il n y a pas de champ d'inscription. Les administrateurs devront attribuer un compte à tous les autres nouveaux administrateurs ou responsables de stations.

FIGURE 6.11 – Page d'accueil

Comme on peut le voir à la figure 6.11, une fois qu'un administrateur ou un gestionnaire de station se connecte, il est automatiquement envoyé à la page d'accueil. Cependant la barre de navigation a été modifiée. Il y a 2 nouveaux onglets qui remplacent l'onglet "connexion". Il s'agit des deux onglets encadrés en rouge à savoir "Utilisateur" et "Déconnexion".

L'onglet de déconnexion va comme son nom l'indique déconnecter l'administrateur et va le ramener sur la page d'accueil normale.

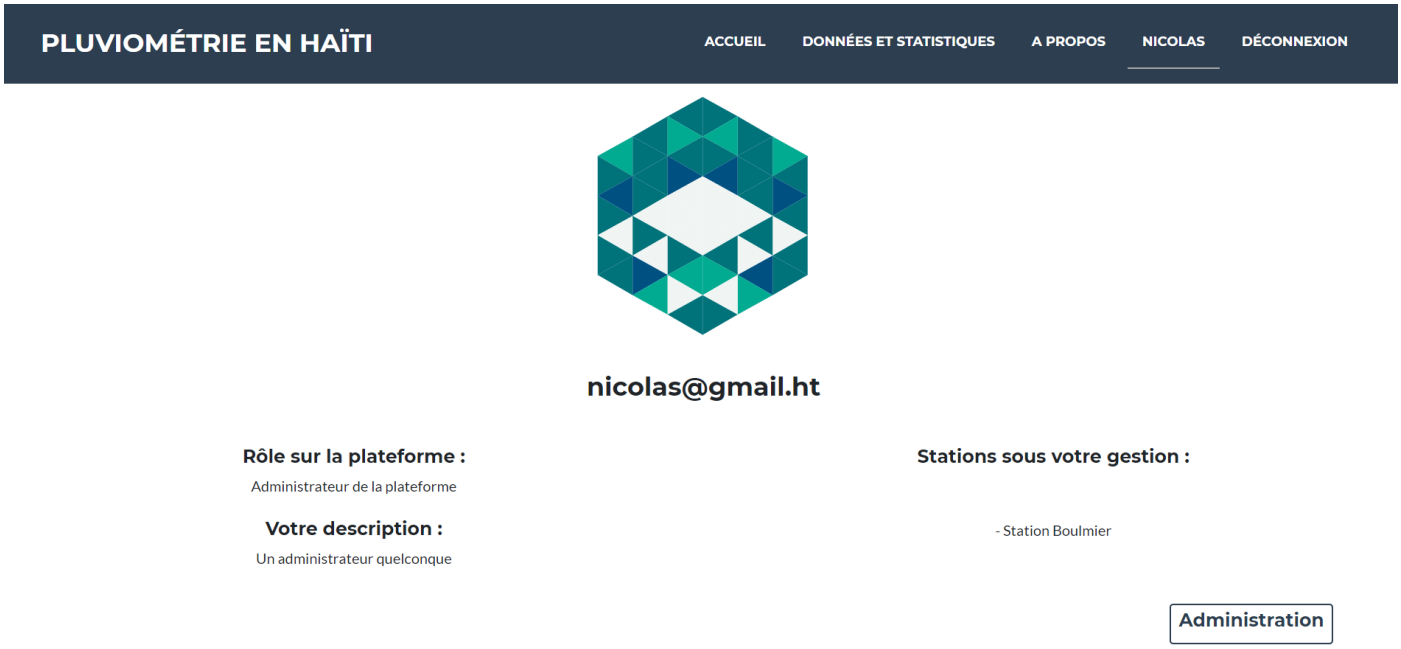


FIGURE 6.12 – Page de l'utilisateur

L'onglet utilisateur renvoie la page affichée sur la figure 6.12. Quand le profil n'est pas édité, il affiche le message que l'on peut voir actuellement sur la figure. Autrement il affiche les informations de l'utilisateur. Le nom de l'onglet change aussi en remplaçant le mot "utilisateur" avec le pseudonyme de l'utilisateur.
Le bouton "Administration" permet aux administrateurs et gestionnaires de stations de rejoindre la console d'administration où ils pourront gérer les données et les utilisateurs. Ces deux parties seront décrites dans les sections 6.2 et 6.3.

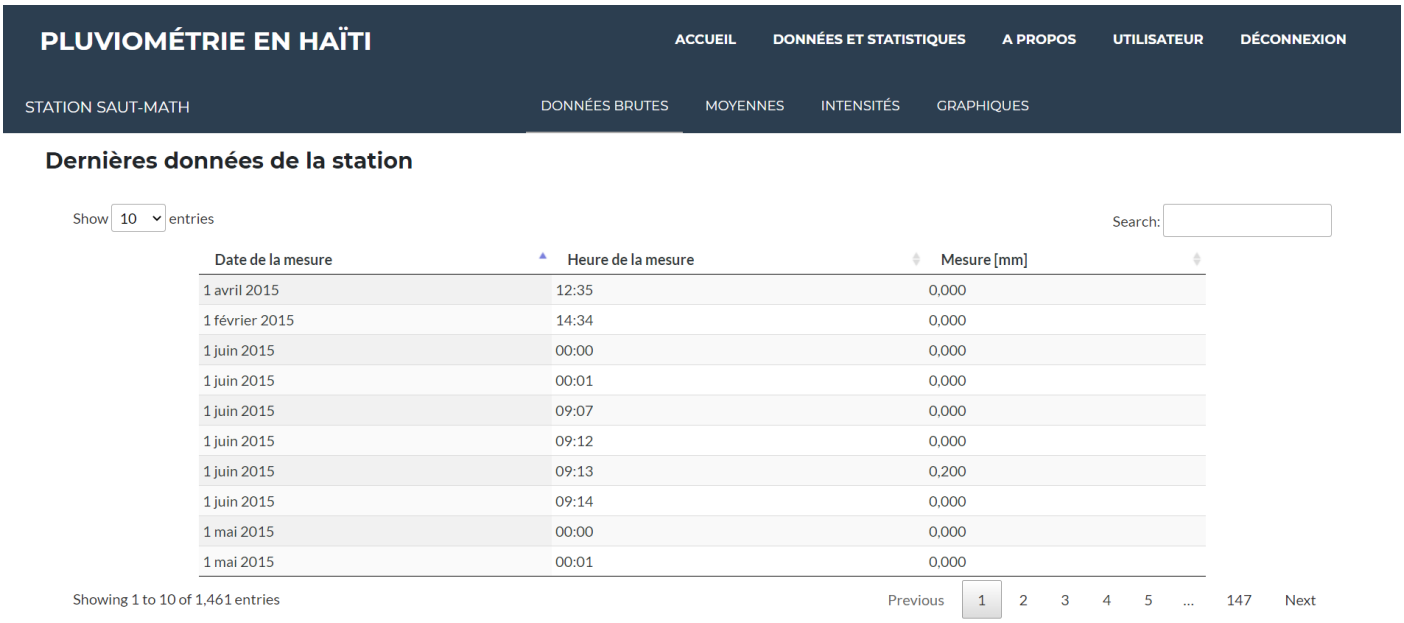


FIGURE 6.13 – Tableau de données brutes

Aussi, lorsqu'un administrateur ou gestionnaire de station est connecté, il a accès au tableau de données que l'on peut apercevoir sur la figure 6.13. Il s'agit du tableau des données brutes enregistrées dans la base de données.

6.2 Traitement des données

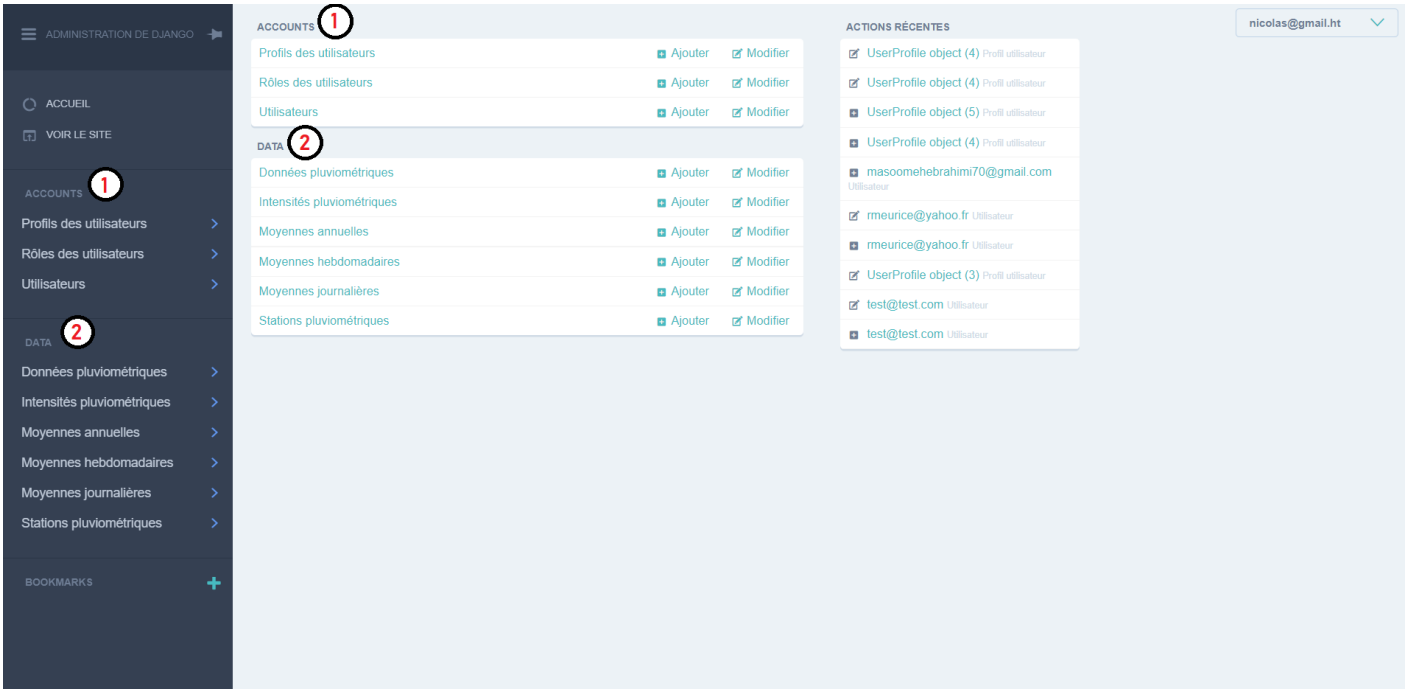


FIGURE 6.14 – Console d'administrateur

Sur la figure 6.14 se trouve la console d'administrateur. Dans cette console, un gestionnaire de station ou administrateur va pouvoir gérer les données pluviométriques.

Dans le menu d'accueil, on distingue 2 sections à savoir "Data" et "Account". Cette section sera consacrée à la partie "Data".
Dans cette section on retrouve des tables dans lesquelles sont enregistrées les données pluviométriques ainsi que les données pré calculées. Chaque sous section représente une table en particulier. Voici la liste de ces tables :

- La table "Données pluviométriques" dans laquelle on enregistre directement les données brutes.
- La table "Intensités pluviométriques" est une table qui pré-calcule pour chaque jour et pour chaque durée de pluie considérée les intensités à partir des données de la table "Données pluviométriques".
- La table "Moyenne annuelle" qui pré-calcule pour chaque année enregistrée dans la table "Données pluviométriques" la moyenne annuelle.
- La table "Moyenne hebdomadaire" qui pré-calcule pour chaque année enregistrée dans la table "Données pluviométriques" la moyenne hebdomadaire.
- La table "Moyenne quotidienne" qui pré-calcule pour chaque année enregistrée dans la table "Données pluviométriques" la moyenne quotidienne.
- La table "Station pluviométrique" dans laquelle sont enregistrées les différentes stations pluviométriques d'Haïti. Les administrateurs les rajoutent directement dans la base de données avec leurs coordonnées géographiques.

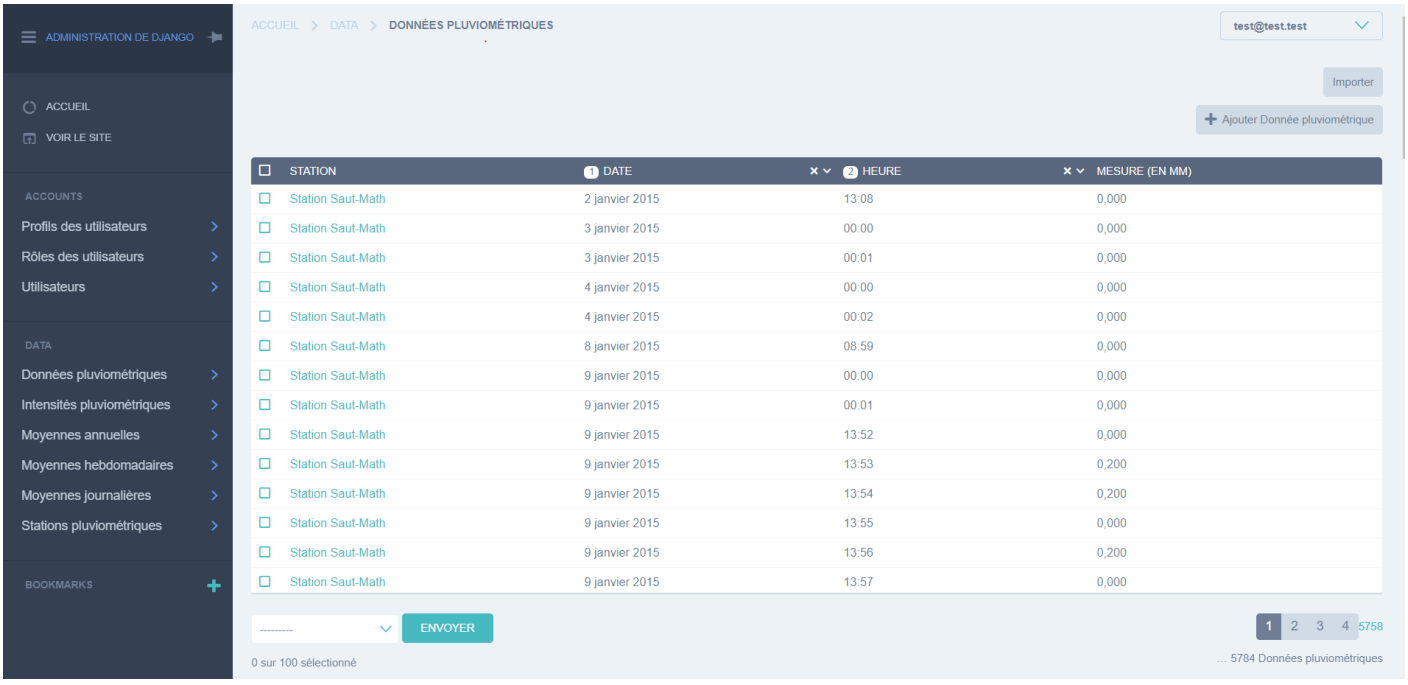


FIGURE 6.15 – Données pluviométriques

Sur la figure 6.15, on retrouve par exemple la table de données "Données pluviométriques". Toutes les tables sont similaires à celle-ci. Pour ajouter de nouvelles données, le gestionnaire de

station peut appuyer sur le bouton "importer" où il peut soumettre un fichier de données.

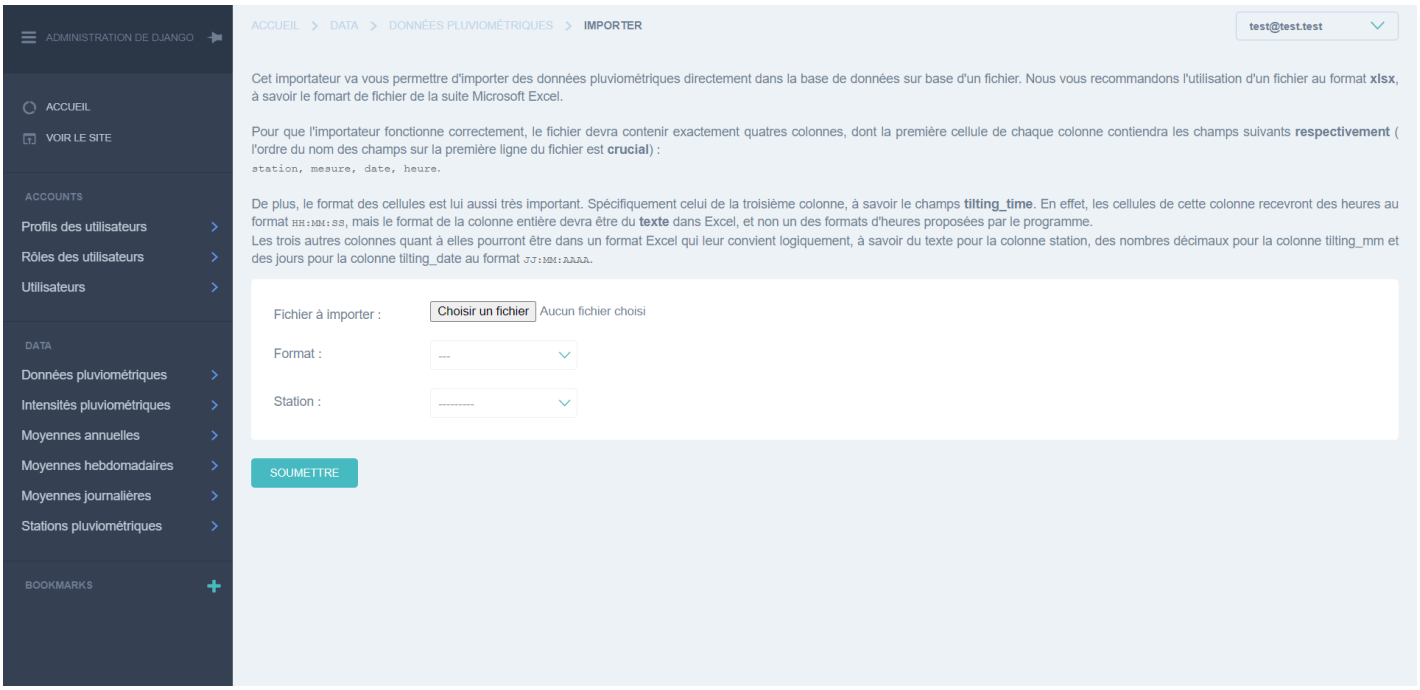


FIGURE 6.16 – Page d'importation

Pour importer il suffit de remplir les champs qui se trouvent sur la page à la figure 6.16 et de soumettre. Il faut entrer le fichier contenant les données en respectant le canevas de la table de données brutes, le format du fichier ainsi que la station des données. Une fois soumis et confirmé, le gestionnaire de station est directement reconduit sur la page de la table des données pluviométriques. Un message apparaîtra en haut de la page pour confirmer la bonne importation des données.

6.3 Gestion des utilisateurs

Pour finir, on discutera de la section "Account" de la console d'administration où les administrateurs peuvent gérer les utilisateurs. Comme on peut le voir à la figure 6.14, il y a 3 tables dans cette section :

— La table "Profils des utilisateurs" dans laquelle on crée un profil pour chaque utilisateur du site web. Un administrateur peut lui attribuer tous les rôles adéquats. C'est lorsqu'un profil est créé pour un utilisateur que l'onglet "Utilisateur " est modifié sur la figure 6.11.

— La table "Rôles des utilisateurs" est une table qui enregistre les différents rôles possibles d'un membre de l'application web.

— La table "Utilisateurs" possède la liste de tous les utilisateurs enregistrés sur le site web.

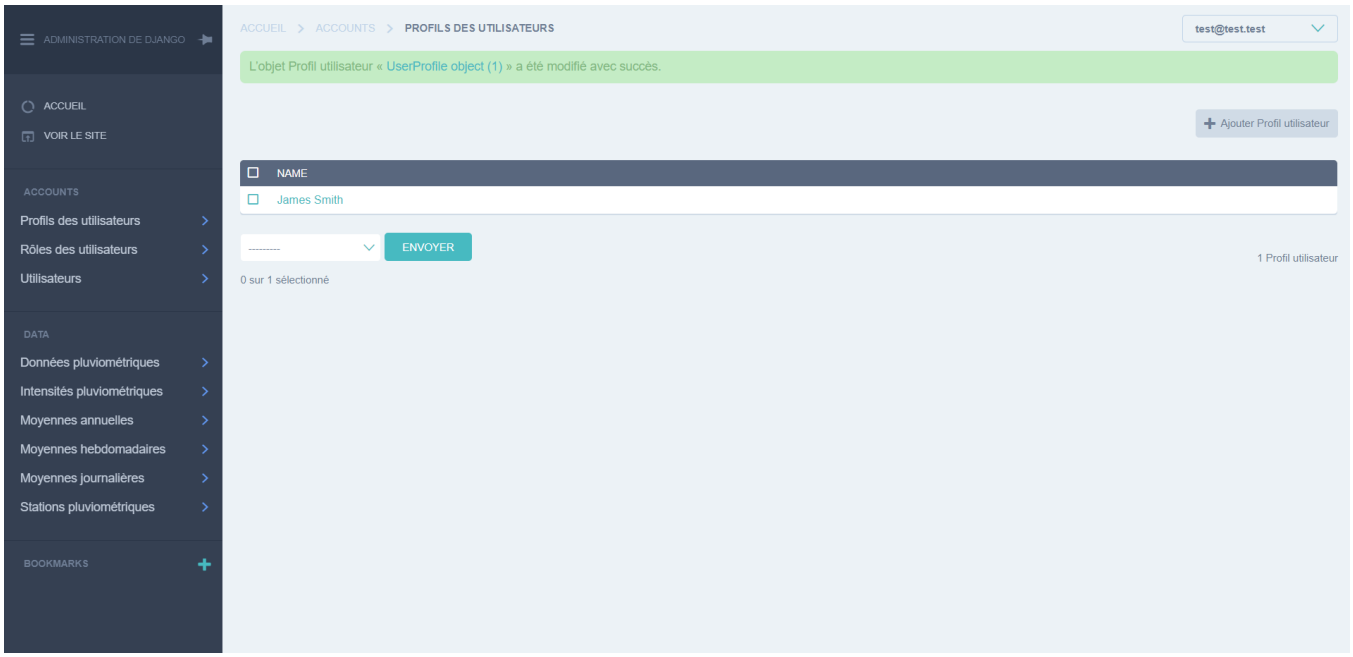


FIGURE 6.17 – Page de profil

Sur la figure 6.17 se trouve la table de "Profils des utilisateurs". Toutes les autres tables sont très similaires à cette table-ci. Si un Administrateur désire ajouter un nouveau profil, il peut simplement appuyer sur le bouton "ajouter Profil utilisateur".

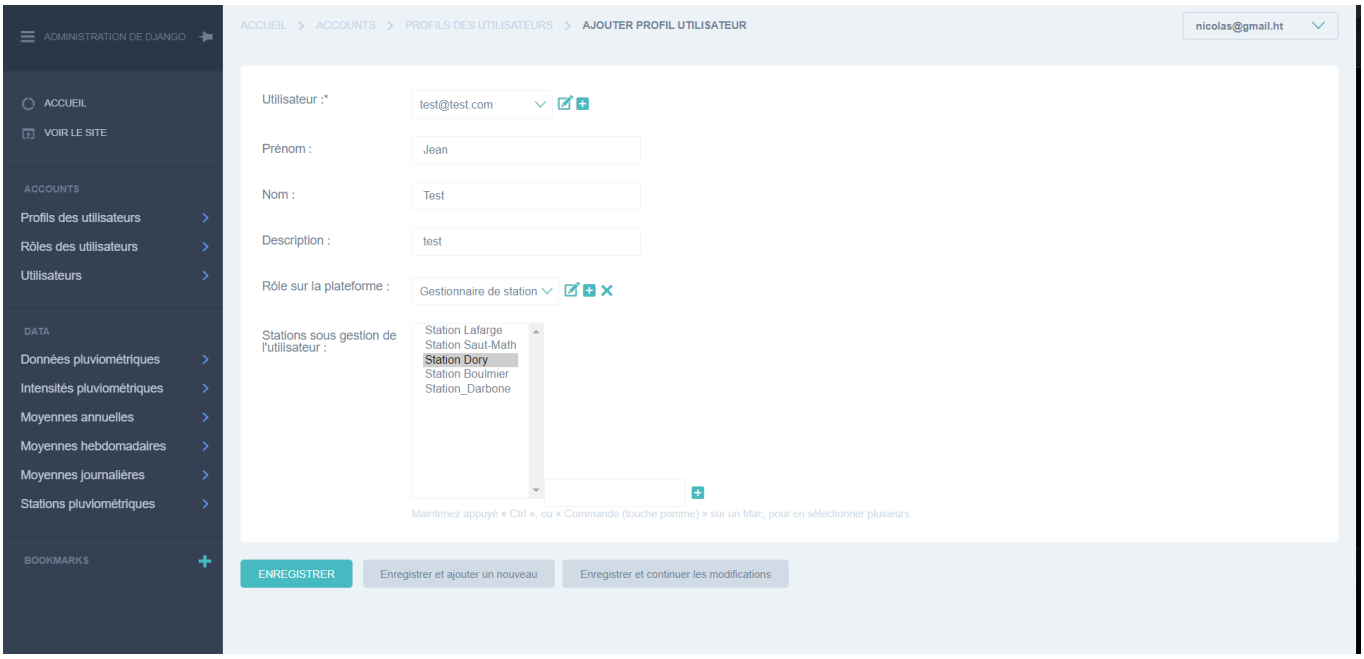


FIGURE 6.18 – Nouveau profil

Sur la figure 6.18, on peut voir la démarche pour créer un nouveau profil. Il suffit de sélectionner un utilisateur, lui écrire son nom et son prénom et lui attribuer un rôle et une ou plusieurs stations.

Dans ce chapitre, nous avons donc décrit les différentes pages web qu'un utilisateur peut rencontrer et comment réagir avec les différentes interfaces à sa disposition.

Chapitre 7

Implementation

La présentation de notre application web, de ses différentes pages et interfaces ayant été réalisée, nous allons maintenant nous pencher sur l'implémentation qui soutient cette application. Nous allons notamment vous décrire la structure générale des parties frontend et backend, ainsi que différentes libraires ou plugins auxquels nous avons pu avoir recours.

7.1 Structure générale

La structure générale d'une application web Django peut-être décrite de manière très simplifiée comme sur la figure 7.1 ci-dessous.

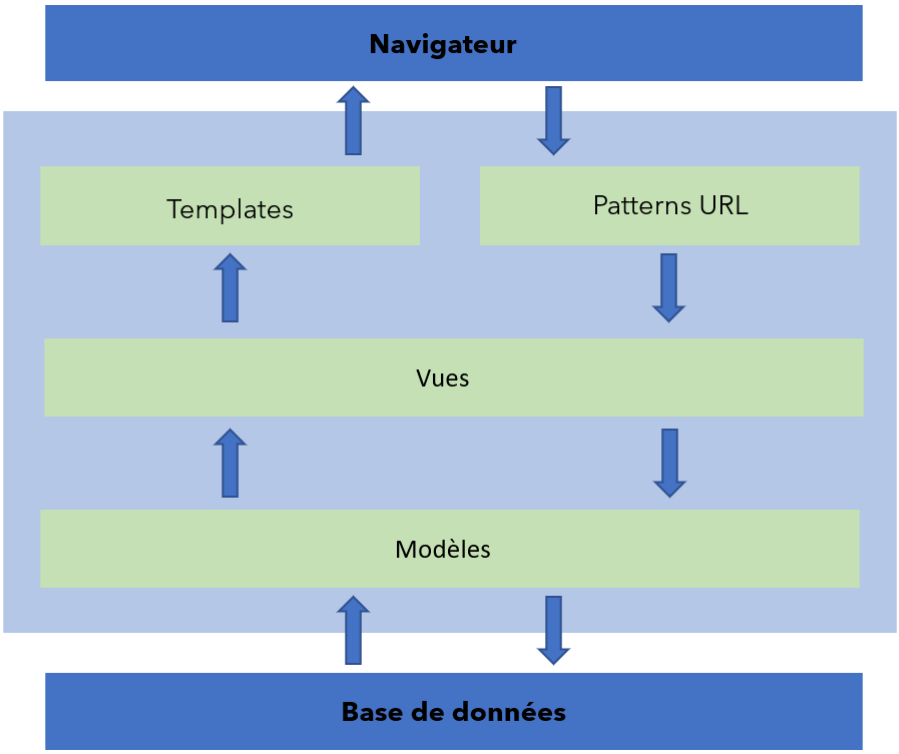


FIGURE 7.1 – Schéma simplifié des interactions pour l'architecture MVC de Django

Comme nous l'avions introduit lors de la section 5.1.2, Django est basé sur une architecture MVC, une abréviation pour Modèle-Vue-Contrôleur. Dans le cas de Django, il s'agit même plutôt d'une architecture MVT ou Modèle-Vue-Template. Les modèles, vues et templates sont les trois blocs les plus importants, apparaissant dans le schéma 7.1, qui constituent le coeur de la structure d'un site web Django, décrite ci-dessous.

Lorsqu'un utilisateur introduira une requête web HTTP depuis son **navigateur**, par exemple une requête de demande de ressources "GET", l'adresse URL demandée sera analysée par ce qu'on appelle les **Patterns URL** sur le schéma. Django va ensuite analyser l'adresse demandée et la comparer aux chemins URL définis dans un fichier python prévu à cet effet, à savoir un fichier "urls.py", afin de savoir à quelle **vue** il va devoir faire référence.

Dans le contexte de Django, une **vue** agit ici comme l'ensemble de la logique responsable du traitement des requêtes des utilisateurs, comme la requête "GET" énoncée avant, et aussi du renvoi des réponses. Elle interagit donc avec à la fois les **templates** (ou gabarits en français), qui sont la partie affichée au client et qui sera décrite un plus loin, et aussi avec les **modèles**, à savoir la partie serveur. Pour cela, la vue demandée va typiquement aller chercher des données stockées en passant par les modèles, qui seront ensuite envoyées vers le gabarit pour être affichées sur le navigateur à l'utilisateur qui les a demandées.

Quant à elle, la couche des **templates** ou "gabarits" fournira une syntaxe adaptée aux développeurs web pour le rendu et l'affichage d'informations à présenter aux utilisateurs [51]. Django présente aussi une couche d'abstraction appelée **modèles** pour structurer et manipuler les données d'une application web qui sont stockées dans une base de données de notre choix [51].

7.2 Schéma global des technologies utilisées

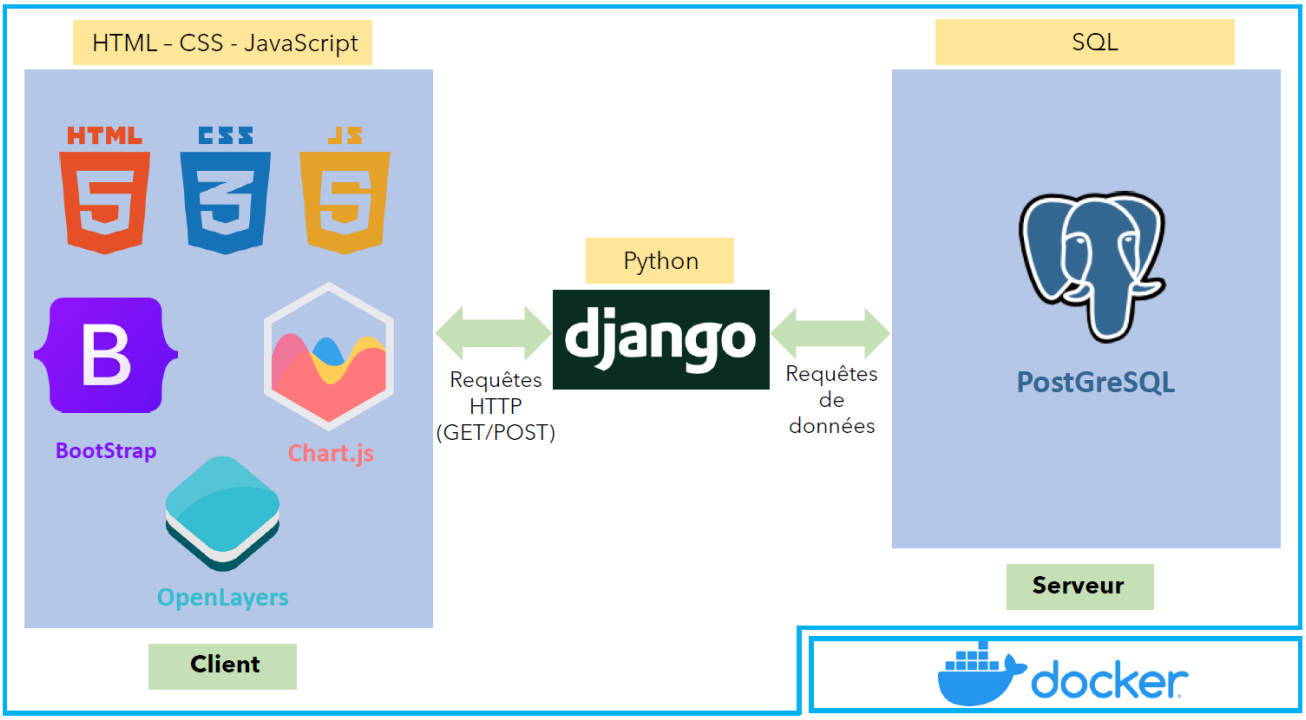


FIGURE 7.2 – Schéma simplifié des technologies interagissant avec Django

Vous trouverez ci-dessus au niveau de la figure 7.2 un schéma non-exhaustif des différentes technologies, pour le côté client (frontend) et pour le côté serveur (backend), que nous avons utilisées en conjonction avec Django. Nous avons aussi indiqué Docker sur le graphe, la technologie qui permet de faire tourner nos applications au sein de "conteneurs" virtuels. Vous pouvez notamment y voir les différents langages informatiques utilisés, et ces technologies vont être décrites dans les deux sections qui suivent, sections 7.3 à 7.5.

7.3 Frontend

Tout d'abord, comme vous avez pu le voir sur le schéma 7.1, le plus gros du travail frontend se fait aux travers des templates lorsque l'on utilise Django. Ceux-ci représentent en quelque sorte toute l'interface graphique destinée aux utilisateurs.Ils définissent donc les éléments qui seront affichés à l'écran.

Et comme vous pouvez le voir sur la figure 7.2, le code qui va constituer les templates est majoritairement écrit en **HTML, CSS et JavaScript**. Ces trois langages informatiques constituent donc le noyau permettant de construire l'interface de l'application.

Mais ce n'est pas tout, Django ajoute aussi du code python dans ce qu'il appelle le "langage de gabarit", et ce au travers de **balises et variables**. Comme il est expliqué dans la documentation de Django : "Un gabarit contient des variables qui sont remplacées par des valeurs lorsque le gabarit est évalué, ainsi que des balises qui contrôlent la logique du gabarit" [53].

La figure 7.3 donne un exemple de gabarit contenant du code HTML, comme par exemple les balises **"<h1> ... </h1>"**, ainsi que beaucoup de code python propre à Django. Nous avons alors d'une part les balises Django, sous la forme de deux sigles "pour cents" : **{% ... %}**, qui permettent de définir des boucles itératives ou "boucle for", que nous utiliserons souvent au travers de notre code. On retrouve aussi d'autre part des variables Django, sous la forme de : **{{*variable*}}** , qui permettent par exemple de récupérer des valeurs depuis les modèles et de les afficher dans le gabarit. Un exemple concret de la figure serait **{{*story.headline*}}** où "headline" est un attribut d'un objet story dans le modèle correspondant. Cela permet donc de récupérer la valeur "headline" stockée dans la table intitulée "story" en base de données.

Concernant les variables, on peut voir plus en détails à la figure 7.3 quelques exemples concrets utilisant des filtres Django. Les filtres transforment les valeurs de variables et les paramètres de balises [53]. Ils utilisent pour cela le symbole "|", et se présentent sous la forme : **{{variable|filtre}}** . Parmi les exemples de la figure, on a **{{story.headline|upper}}** , où "upper" agit comme un filtre qui va faire en sorte que la variable soit affichée en lettres majuscules, à savoir la headline de la story dans ce cas.

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

FIGURE 7.3 – Exemple de code d'un gabarit Django [53]

Maintenant, nous allons détailler un à un les différents plugins, frameworks et librairies frontends que nous avons utilisés afin de réaliser ce projet.

7.3.1 Apparence générale de l'application : BootStrap

Tout d'abord, nous avons utilisé BootStrap pour le développement frontend, ou "côté client" [54]. Il s'agit d'un framework incontournable pour le développement du design et donc de la partie frontend de sites webs de nos jours, comme cela peut se voir sur GitHub ou il détient à ce jour, en Mai 2021, le rang de dixième dépôt (ou répertoire) le plus populaire de la plateforme toute entière [55].

BootStrap apporte directement des outils sous forme de code HTML et CSS afin d'afficher facilement différents éléments communs de pages webs comme des boutons, outils de navigation et autres éléments interactifs, et tout cela de manière simple et élégante. Dans notre cas, nous avons utilisé la version 4 de BootStrap que nous importons dans notre code au travers d'un CDN, à savoir un Content Delivery Network ou Réseau de diffusion de contenu en français, afin de rendre l'application plus légère.

Ensuite, bien que BootStrap mette tous ces outils à notre disposition, il incombe toujours aux développeurs utilisant le framework de s'en servir pour designer le site web. Certaines plateformes, comme *Start BootStrap* [56], proposent ce que l'on appelle des "thèmes BootStrap" open-source et gratuits, pour éviter de devoir tout créer de zéro.

Nous nous sommes tournés vers un thème intitulé "Freelancer" pour le développement du site, car celui-ci nous paraissait simple et épuré. Cela semblait donc répondre à nos attentes pour une telle plateforme.

Enfin, aussi bien BootStrap que le thème que nous avons choisi sont open-source et gratuits, sous licence MIT.

7.3.2 Carte interactive : OpenLayers

La carte de Haïti est le premier élément du site avec lequel un utilisateur va réellement interagir. Il s'agit là d'un élément central pour nous, car c'est cette carte qui va permettre à un utilisateur d'être redirigé vers la page de données et statistiques liées à la station de son choix sur cette carte.

Nous avons alors cherché une technologie permettant d'aisément intégrer une carte dynamique dans un site web, et nous voulions idéalement que cette solution technologique soit libre de droit et gratuite.

Au début de nos recherches, nous avions considéré plusieurs technologies, comme l'API Google Map [57] dans un premier temps ou MapBox [58] dans un second temps. Le problème avec les deux était le même au final, puisque ces solutions requièrent toutes deux de souscrire à une forme d'abonnement pour payer le service.

En cherchant un peu plus loin, nous sommes alors tombés sur le projet OpenLayers [59]. Il s'agit d'un projet libre de librairie JavaScript, aussi dite librairie open-source, permettant l'affichage de cartes dans un navigateur web et ce sous licence FreeBSD (aussi connu sous 2-clause BSD License), une licence permettant d'utiliser le logiciel dans notre application sans restriction.

En-dehors de l'aspect gratuit et complètement libre d'OpenLayers, d'autres points forts sont aussi à mentionner, comme [59] :

1. La librairie est très **légère, et donc rapide à charger** dans la page web. C'est un point important, étant donné que l'affichage d'une carte dynamique peut rapidement demander de charger une quantité de ressources non-négligeable sur le réseau et que nous voulons minimiser le poids de l'application au maximum. Nous pouvons ainsi l'importer grâce à un CDN, tout comme BootStrap.
2. La librairie est **versatile**, puisqu'elle peut fonctionner avec différents types de formats de "couches" pour les cartes, comme d'une part les couches de type "carrelées", ou Tiled layers, comme un utilisateur pourrait vouloir intégrer depuis OSM, Bing ou MapBox, et d'autre part les couches de type vectorielles, ou Vector layers, comme un utilisateur pourrait vouloir intégrer depuis GeoJSON par exemple.
3. La librairie est aussi **facile à customiser ou étendre** avec du code CSS.

7.3.3 Tableaux : DataTables

Ensuite, passée la sélection d'une station pluviométrique sur la carte interactive d'Haïti affichée avec OpenLayers, l'utilisateur sera amené sur la page associée aux données et statistiques de cette station. L'utilisateur du site web se trouvera alors dans un premier temps face aux tables de données de cette station, telles que les tables de moyennes journalières, hebdomadaires et annuelles ou les tables d'intensités pluviométriques.

Afin d'éviter de ré-inventer la roue, nous avons cherché une solution open-source et gratuite afin de pouvoir afficher de telles tables de manières simples et ergonomiques, et c'est exactement ce que DataTables nous apporte.

DataTables est ce qu'on appelle un plug-in, que nous pouvons définir comme "un petit logiciel qui se greffe à un programme principal pour lui conférer de nouvelles fonctionnalités" [61], destiné à la librairie JavaScript JQuery [60]. Quant à JQuery, il s'agit donc d'une librairie JavaScript libre qui a pour but de rendre plus simple l'écriture de scripts dans le code HTML des pages web. Jquery permet notamment de rendre l'utiliastion d'Ajax beaucoup plus aisée, grâce à une API facile à prendre en main qui fonctionne dans tous types de navigateurs [62]. Enfin, Ajax, qui est l'acronyme de "Asynchronous JavaScript and XML", est un ensemble de techniques de programmation utilisant différentes technologies web pour créer des applications web asynchrones, ce qui permet par exemple de modifier une partie d'une page web affichée chez l'utilisateur sans avoir à afficher complètement une nouvelle page [63] [64].

Pour revenir à DataTables, c'est un outil performant qui offre différentes fonctionnalités sur les tables directement "out of the box", en-dehors d'un plus bel affichage des tables :

1. Pagination
2. Recherche instantanée
3. Sélection du nombre de données affichées par page
4. Tri des colonnes
5. "Customisation" du thème de la table

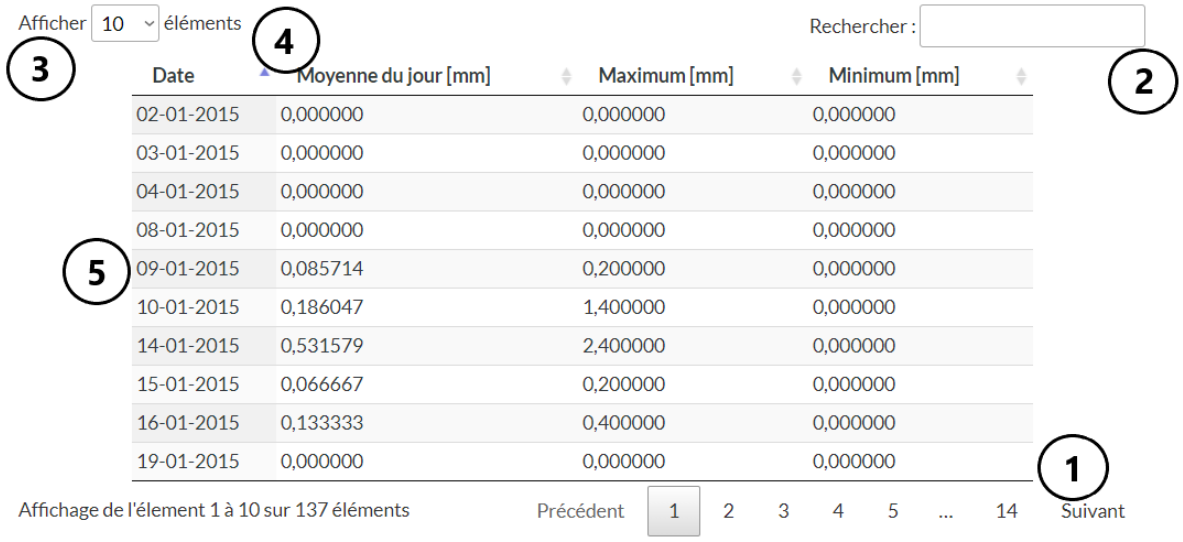


FIGURE 7.4 – Fonctionnalités fournies par DataTables

Enfin, DataTables est avant tout open source et gratuit, sous licence MIT. Cela signifie que nous sommes libres d'utiliser DataTables comme nous le souhaitons, ce qui comprend la modification et la redistribution du code [60].

7.3.4 Graphes : Chart.JS

Après les tables, nous voulions fournir aux utilisateurs de la plateforme la possibilité d'observer les données et les statistiques calculées sur ces dernières sous forme de graphes. En effet, lorsque l'on est face à de grandes quantités d'informations comme c'est le cas pour notre

application, il est important de pouvoir les visualiser correctement afin de pouvoir en tirer des conclusions, ce qui est souvent plus difficile avec uniquement des tableaux.

Pour ce faire, nous avons encore une fois cherché une librairie open-source et gratuite pour notre implémentation. Lors de nos recherches parmi d'éventuelles librairies JavaScript qui pourraient subvenir à nos besoins, nous nous sommes rapidement intéressés à Chart.JS.

Chart.JS est une librairie codée en JavaScript comme son nom l'indique. Il s'agit d'un projet open-source et gratuit, sous licence MIT, tout comme DataTables précédemment [66].

En plus de cela, Chart.JS est populaire avec une communauté assez active, et présente différentes fonctionnalités et avantages, comme le sont cités sur leur page web [66] :

1. Il est possible de choisir entre 8 types de graphes différents, ce qui comblera plus que nécessaire nos besoins qui se résument majoritairement à des graphes linéaires ou en bâtonnets.
2. Leurs graphes peuvent facilement être rendus "responsives", c'est-à-dire que leur taille peut facilement s'adapter au type d'écran ou d'interface sur laquelle la page web sera affichée.
3. Une compatibilité revendiquée parfaite avec les navigateurs modernes, de part leurs canvas en HTML 5.

Nous en avons donc conclu que Chart.JS était un candidat parfait pour venir en solution à notre souhait d'afficher des graphes. La librairie s'est avérée être aisée à mettre en oeuvre, et cela en nous offrant un résultat bien fini au niveau graphique.

7.3.5 Administration : Django Jet

En dehors de l'expérience des utilisateurs avec les graphes et tableaux sur les pages de "Données et Statistiques", il est aussi important de fournir une interface aux administrateurs de la plateforme, notamment afin que ceux-ci puissent avoir un meilleur aperçu du contenu de la base de données, et aussi afin qu'ils puissent importer des données pluviométriques.

Heureusement pour nous, Django fournit automatiquement une console d'administration lorsqu'on l'on crée une application web. Mais cette console d'administration basique ne correspondant selon nous pas à certains critères d'ergonomie et d'apparence que nous aurions aimé avoir, nous avons alors commencé à rechercher comment changer et améliorer celle-ci simplement.

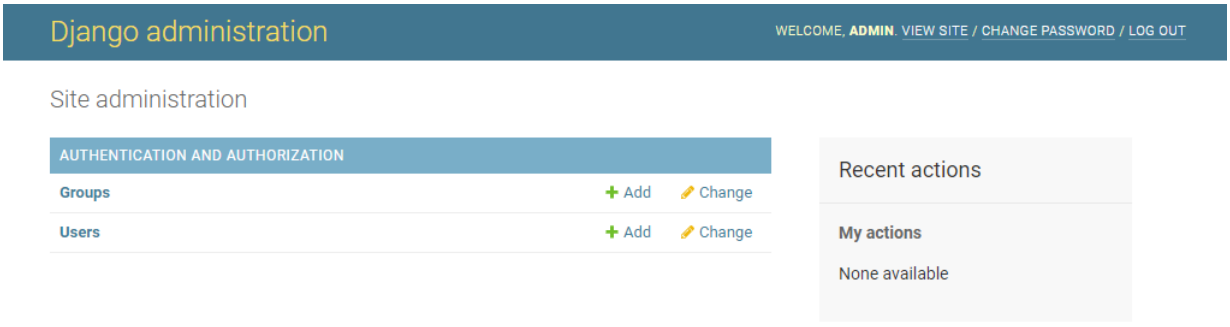


FIGURE 7.5 – Une console d'administration "basique"

Django-Jet propose alors une solution parfaite à ce que nous recherchions. Il s'agit d'un projet de plug-in open-source sous licence AGPLv3. Il donne à l'administration Django une nouvelle apparence plus ergonomique, et permet aussi de rendre "responsive" les interfaces mobiles. La page d'accueil de l'administration devient aussi plus claire, et peut être personnalisée pour comporter des informations plus utiles pour les administrateurs, comme des graphes grâce à certains widgets [67].

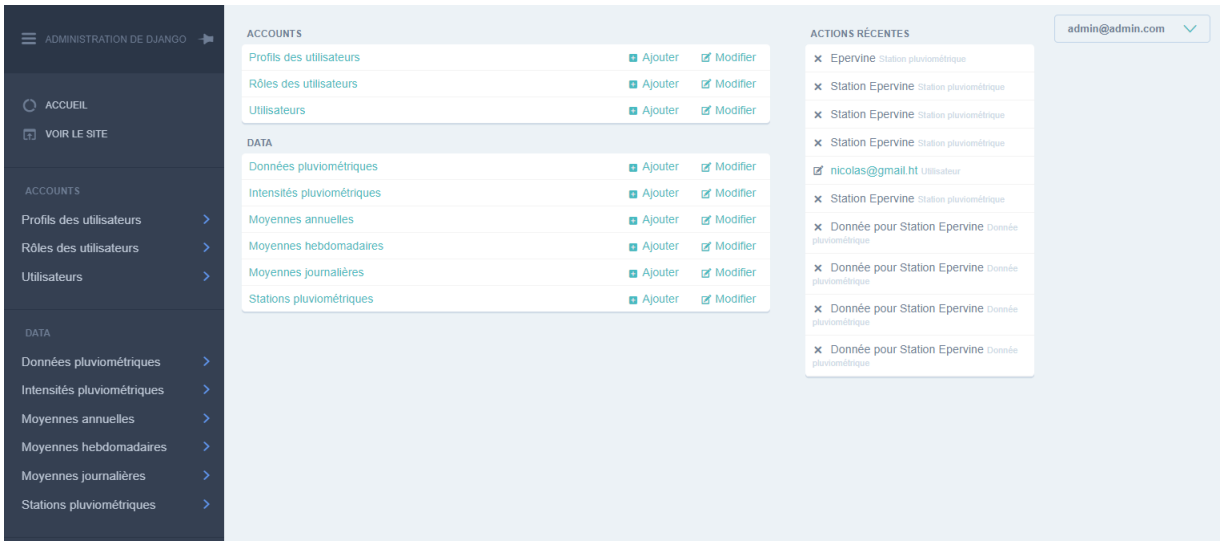


FIGURE 7.6 – Apparence de la console d'administration avec Django-Jet (Voir annexe 11.5)

Contrairement aux librairies et plug-ins précédents qui étaient sous licence MIT ou freeBSD, et donc totalement libre, Django Jet quant à lui est un projet open-source mais sous licence AGPLv3 [67]. Étant donné la nature de la licence AGPLv3, abréviation de GNU Affero General Public License version 3, abrégée AGPL, qui est ce qu'on appelle une licence libre copyleft, ayant pour but d'obliger les services accessibles par le réseau de publier leur code source, cela pouvait signifier que notre projet devait donc lui aussi se conformer à ce type de licence [68]. Mais tout cela a été discuté et analysé dans la section 5.4.

7.3.6 Importation : Django Import/Export

Enfin, comme nous l'avons mentionné à plusieurs reprises, nous avions besoin pour notre application de fournir un moyen aisé d'importer des données pluviométriques au sein de la base de données.

Pour ce faire, et une fois encore dans une démarche nous permettant d'éviter de ré-inventer la roue, nous nous sommes penchés sur le projet "Django Import/Export". Il s'agit d'une application et librairie pour Django permettant d'importer et d'exporter des données, et ce avec une intégration directement dans la console d'administration [69]. Le projet est une fois encore open-source, sous licence FreeBSD (ou BDS 2-clause) tout comme OpenLayers. L'intégration de cette application s'est faite aisément dans notre projet.

7.4 Backend

Il convient maintenant de discuter de notre architecture serveur dite architecture backend. Nous commencerons dans un premier temps par vous présenter notre schéma relationnel de la base de données, sous forme de schéma ERD.

Ce schéma est la représentation directe de ce que nos "modèles" Django vont contenir. Ensuite, nous pourrons discuter de technologies qui sont intervenues dans l'implémentation du backend.

7.4.1 Structure de la base de données

Comme vous pouvez le voir à la figure 7.7, nous avons représenté la structure de notre base de données par un schéma relationnel. Nous avons choisi d'utiliser le schéma **erd** (Entity Relationship Diagram) pour sa simplicité de conception et de représentation des bases de données[33]. Nous avions d'abord pensé représenter la base de donnée avec un schémas ORM [34] mais nous avons abandonné cette idée pour l'unique raison que les relations étaient difficilement lisibles. Notre base de données est constituée de 5 tables pour les données pluviométriques et leurs statistiques, 1 table pour les stations et trois tables pour la gestion des utilisateurs. Ces 9 tables au total sont représentées respectivement par 9 classes écrites en python dans nos modèles Django. Nous allons alors pouvoir vous décrire chacune de ces tables ou modèles.

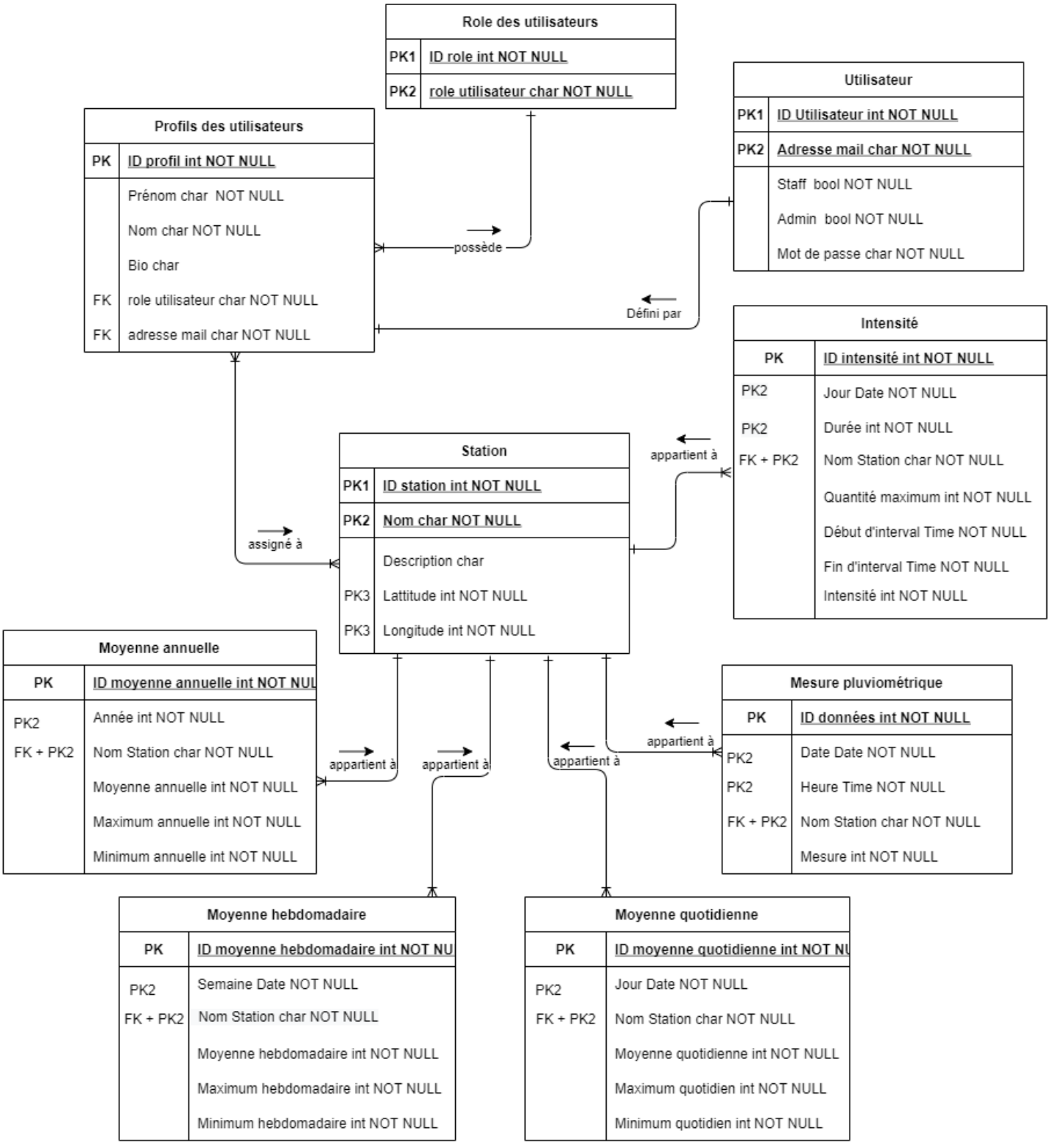
Stations Pour commencer, prenons la table **Station** qui est représentée dans Django par notre classe "Station" dans le fichier des modèles des données. Cette table va donc bien évidemment représenter les stations, et nous avons fait le choix qu'une station devrait obligatoirement avoir un nom, ainsi qu'une longitude et une latitude.

Le nom d'une station sera quant à lui unique, il nous semble en effet logique que plusieurs stations ne puissent posséder le même nom. Une station pourra aussi avoir éventuellement une description, bien que cela ne soit pas obligatoire, afin par exemple de décrire l'emplacement auquel la station se trouve pour faciliter sa localisation sur le terrain.

Mesures pluviométriques Ensuite, nous pouvons considérer la table **Mesure pluviométrique**, qui est est représentée dans Django par notre classe "Data" dans le fichier des modèles des données. C'est dans cette table que nous allons enregistrer l'ensemble des données qui seront importées à l'état brut sur l'application web. Il va de soi que cette table sera de loin la plus grande de toutes. Les champs de cette table sont relativement directs à comprendre ; nous avons décidé qu'une donnée serait donc représentée par :

— la station à laquelle la mesure a été prise, d'où le champ **Nom Station**.
— le moment où la mesure a été prise, séparé en deux champs **Date** et **Heure**.
— ainsi que la **Mesure** en elle-même

Nous avons aussi ajouté une contrainte d'unicité sur le triplet constitué des champs **station**, **date** et **heure**, afin de spécifier à la base de données qu'il était impossible pour une même station d'avoir deux mesures différentes exactement au même moment. Cela permettra d'assurer notamment l'unicité des mesures pluviométriques, et l'intégrité des données en général.



Légende :

Relation 1 : 1 — **PK**(gras) : Clé primaire

Relation M : 1 — PK(suivit d'un nombre) : Contrainte d'unicité

Relation 1 : N — FK : Clé étrangère

Relation M : N

FIGURE 7.7 – Schéma ERD de notre base de données

Moyennes Concernant les données statistiques que nous calculons, nous pouvons parler des moyennes. Nous avons décidé de subdiviser les moyennes en trois tables différentes, étant donné qu'on calculera trois types de moyennes : journalières, hebdomadaires et annuelles. Nous pouvons analyser une seule des trois tables, étant donné qu'elles sont tout à fait similaires.

Prenons par exemple la table **Moyenne quotidienne** qui est représentée dans Django par notre modèle ou classe "MeanDay" dans le fichier des modèles des données. C'est dans cette table que les moyennes journalières sont calculées sur nos données pluviométriques. Mais ce n'est pas tout, nous enregistrons aussi pour chaque moyenne journalière le minima et le maxima en chute de pluie de la journée. Ce raisonnement est valable pour les tables de moyennes hebdomadaires et annuelles.

Une moyenne pluviométrique journalière sera donc représentée par :

— la station pour laquelle la moyenne est calculée, d'où le champ **station**.
— la journée de la moyenne en question, d'où le champ **mean_day**.
— la valeur moyenne calculée pour cette journée, représentée par le champ **mean_per_day**.
— le maxima et le minima en mm de pluie mesuré pour la journée en question, donnant respectivement les champs **max_per_day** et **min_per_day**.

Une fois encore, nous avons ajouté une contrainte d'unicité sur la table, tout comme pour celle des données pluviométriques. Ici, la contrainte stipule simplement qu'il ne peut y avoir qu'une seule moyenne journalière par station et par jour, car elle porte sur le doublon **station** et **mean_day**.

Intensités La dernière table représentée par un modèle dans notre fichier de modèles des données est la table intitulée **Intensité**. Cette table va contenir les valeurs calculées automatiquement des intensités pluviométriques pour les différentes stations, qui serviront pour les courbes Intensité-Durée-Fréquence ou IDF. Les intensités pluviométriques sont un peu différentes à représenter des moyennes dans une table, étant donné qu'une valeur d'intensité est propre à une journée et une station, mais aussi et surtout relative à une certaine **durée**. Il s'agit bien de calculer la valeur de chute de pluie maximale (**l'intensité**) sur un certain intervalle (**la durée**) et ce pour une journée.

Nous avons donc décidé qu'une valeur d'intensité pluviométrique serait représentée comme suit :

— comme pour les moyennes, la valeur fera référence à la station pour laquelle elle est calculée, d'où le champ **station**.
— la journée pour laquelle l'intensité en question a été calculée, **intensity_day**.
— la durée temporelle de l'intensité, ce qui donne le champ **duration**.
— le début et la fin de l'intervalle de durée qui a eu en moyenne l'intensité de pluie la plus élevée, ce que représentent respectivement les champs **start_interval** et **end_interval**.
— la mesure maximale de chute de pluie pendant cet intervalle **max_amount**.
— enfin, n'oublions pas la valeur d'intensité pour la durée choisie, représentée par le champ **intensity_value**.

La relation d'unicité pour cette table porte donc sur le triplet constitué des champs **station**, **intensity_day** et **duration**, permettant ainsi d'assurer qu'il n'y ait bien qu'une seule valeur d'intensité pluviométrique par station, journée et durée.

Utilisateurs Il reste maintenant à discuter des tables pour la gestion des utilisateurs de la plateforme.

Nous avons alors trois tables différentes pour représenter les utilisateurs. La première et plus importante est la table **Utilisateur**, qui est est représentée dans Django par notre classe "User" dans le fichier de modèles des comptes.

Profils des utilisateurs Ensuite, nous avons la table **Profils des utilisateurs**, qui permet de faire référence à un élément de la table **Utilisateur** afin de lui lier un profil. Cette table est représentée par la classe "UserProfile" dans le fichier de modèles des comptes.

Nous avons fait le choix qu'un profil pourrait contenir les informations suivantes à propos d'un utilisateur :

- Le prénom de l'utilisateur.
- Le nom de famille de l'utilisateur.
- Le rôle sur la plateforme de l'utilisateur. Ce rôle doit être un de ceux définis dans la dernière table dont nous allons parler.
- Les stations sous la gestion de cet utilisateur.
- Une description facultative de l'utilisateur, si certaines informations supplémentaires à son sujet devaient être répertoriées.

Rôles des utilisateurs Enfin, nous avons créé une dernière table, **Rôle des utilisateurs**, pour encoder les différents rôles des gestionnaires de l'application web, et qui servent donc à être affichés dans les profils. Cette table est représentée par la classe "UserRole" dans le fichier des modèles des comptes. Un rôle sera donc composé d'un nom pour ce rôle et d'une description.

7.4.2 Les calculs statistiques

Maintenant que nous avons présenté comment les données et statistiques seront sauvegardées et maintenues dans la base de données, il convient à présent d'expliquer comment nous calculons les différentes statistiques, et tout particulièrement les intensités pluviométriques qui utilisent une formule mathématique non-triviale.

Moyennes pluviométriques A chaque importation de données, l'application va calculer les moyennes, maxima et minima journaliers pour chaque date enregistrée dans la base données et en faire de même pour les moyennes, maxima et minima hebdomadaires de chaque semaine enregistrée ainsi que pour chaque année enregistrée.

Intensités pluviométriques Le modèle de calcul va donc tout d'abord définir les différents intervalles de temps sur lesquels on calcule les intensités. Les durées ont été définies comme ceci : [5,10,15,20,30,40,50,60,90,120,180]. Ces valeurs sont toutes exprimées en minutes. Le modèle va itérer sur cette liste de durée pour calculer les intensités. C'est à dire qu'à chaque itération,

chaque journée enregistrée dans la base de données sera divisée en plusieurs intervalles de temps par ces différentes durées.

L'intervalle de temps sur lequel on retrouve la quantité de chute de pluie maximum sera l'intervalle qui déterminera l'intensité pour le jour assigné et la durée assignée. L'intensité est simplement la quantité de chute maximale exprimée en heures. [65]

Le modèle de calcul va donc pour chaque paire de station et de jour enregistrés dans la base de données, calculer 11 entrées de table.

7.4.3 PostGreSQL

Comme nous l'avons introduit lors du chapitre intitulé "Analyse technique" au point 5.2 - "Choix du SGBD", PostGreSQL est un système de gestion de bases de données relationnelles libre sous licence BSD. Comme il est défini sur la page officielle du projet PostGreSQL, on peut traduire ceci :

> " PostgreSQL est un puissant système de base de données relationnelles open source avec plus de 30 ans de développement actif qui lui a valu une solide réputation de fiabilité, de robustesse des fonctionnalités et de performances. "[70]

Pour gérer une base de données relationnelles avec les tables définies plus haut, PostGreSQL nous semble donc être un très bon choix. Les détails des quelques avantages qui ont fait pencher notre choix en faveur de ce SGBD ont été expliqués au point 5.2.

Quant à l'implémentation même, le côté pratique de Django est qu'il s'occupe entièrement de gérer et de communiquer avec PostGreSQL. Nous écrivons nos requêtes sous forme de code Python portant sur les classes définies dans les modèles, et Django se charge automatiquement de faire le lien entre ces classes et les tables correspondantes en base de données, ainsi que de traduire nos requêtes écrites en python en requêtes SQL.

Par exemple, une commande python dans notre code telle que :

Data.objects.filter(station="Station Epervine")

sera traduite en une requête SQL par Django vers PostGreSQL par :

SELECT * FROM data WHERE station="Station Epervine"

Grâce à l'abstraction apportée par Django, nous n'avons pour ainsi dire jamais du recourir à écrire des commandes SQL directement, qui auraient pu s'avérer très complexes pour des requêtes plus élaborées, mais nous avons pu nous simplifier la tâche en écrivant du code Python.

7.4.4 Service de livraison d'e-mail et SMTP

Dans le cadre de la partie "Contact" de notre application, une fonctionnalité nécessaire pour le site est de pouvoir automatiquement envoyer un mail contenant les messages, questions ou requêtes d'utilisateurs sur une boîte mail administrative qui est définie dans les paramètres de l'application web.

Pour ce faire et pouvoir envoyer des e-mails réels, il est obligatoire de passer par ce qu'on appelle un service SMTP tel que SendGrid[71], Mailgun[72] ou SES[73]. Heureusement, Django rend assez aisé le passage à ces services.

SMPT est l'acronyme pour **Simple Mail Transfer Protocol**. Comme défini dans "SIMPLE MAIL TRANSFER PROTOCOL" par Jonathan B. Postel en 1982, smtp est donc un protocole de communication utilisé pour transférer le courrier électronique vers les serveurs de messagerie électronique [74]. On peut notamment citer :

> The objective of Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently.[74]

Mais SMTP n'est que le protocole, et nous avons donc encore besoin de passer par un des services cités précédemment pour accéder à un serveur qui s'occupera d'envoyer nos mails. Nous nous sommes alors tournés vers Sendgrid, car il semblait fournir un service SMTP efficace et fiable et gratuitement, tout en ayant le bénéfice d'offrir une intégration assez rapide dans notre application.

Et comme nous l'avons dit, Django permet aisément d'intégrer un tel service à notre application en modifiant directement quelques lignes de code python dans le fichier de paramétrages du site web, à savoir le fichier "settings.py" :

```
EMAIL_BACKEND = "django.core.mail.backends.smtp.EmailBackend"
DEFAULT_FROM_EMAIL = "nicolas.verbois@student.uclouvain.be"
EMAIL_HOST = 'smtp.sendgrid.net'
EMAIL_HOST_USER = 'apikey' # this is exactly the value 'api_key'
EMAIL_HOST_PASSWORD = SENDGRID_API_KEY #api_key
EMAIL_PORT = 587
EMAIL_USE_TLS = True
```

FIGURE 7.8 – Code pour l'intégration d'un service SMTP dans notre application

Les deux seuls champs à réellement retenir et modifier par la suite si vous souhaitez utiliser l'application sont les champs **DEFAULT_FROM_EMAIL** et **EMAIL_HOST_PASSWORD**.

- **DEFAULT_FROM_EMAIL** est le paramètre qui va définir la boîte mail de réception, à savoir donc la boîte d'administration de l'application.
- **EMAIL_HOST_PASSWORD** contient non pas un mot de passe mais en réalité la clef d'API du service SMTP choisi, dans notre cas celle de Sendgrid.

7.5 Docker

Enfin, comme nous l'avons mentionné au point 5.3 de l'analyse technique, nous avons choisi d'opter pour l'intégration de Docker dans notre projet.

Nous avons choisi d'utiliser un outil de Docker intitulé "Docker Compose", qui se caractérise comme étant un outil permettant de définir et d'exécuter des applications Docker "multi-conteneurs" qui peut aussi bien tourner sur Linux, MacOS ou Windows [88]. Le fonctionnement de Docker Compose repose lui-même sur Docker Engine, qui doit être installé au préalable [87].

L'avantage comme nous l'avons expliqué au point 5.3 de l'analyse technique, est qu'une fois Docker Compose et Docker Engine installé sur la machine, dont d'ailleurs le processus d'installation est assez simple et extrêmement bien détaillé dans la documentation Docker, il va nous être alors très aisé de prendre notre application et de la lancer au sein d'un conteneur Docker.

En effet, il suffit de configurer deux fichiers dans le code de notre projet, que vous pouvez retrouver à la racine de l'arborescence aux noms de **Dockerfile** et **docker-compose.yml**. C'est au travers de ces deux fichiers que toute la configuration du conteneur Docker va se faire.

Dockerfile Un fichier Docker ou Dockerfile est un document texte qui va contenir toutes les commandes et instructions qu'un utilisateur pourrait appeler afin d'assembler l'image de notre application.

Par exemple, dans notre cas, notre fichier Dockerfile va se charger de lancer l'installation des fichiers de "requirements", qui sont les logiciels et programmes nécessaires au bon fonctionnement d'une application. Ensuite, le Dockerfile nous sert aussi à exposer le port qui va servir à l'application pour communiquer sur le web, à savoir le port 80 qui est le port classique pour le service HTTP.

docker-compose.yml Ensuite, le fichier YAML quant à lui est propre à Docker Compose, et l'utilisation de ce fichier permet de configurer les services de l'application au sein de Docker Compose. Par exemple, dans le cas de notre application, nous avons définis 3 services différents pour Docker compose, à savoir un service pour la base de données PostGreSQL intitulé "postgres", un service pour l'exécution des migrations, répondant à la commande python *python manage.py migrate*, intitulé "migrate" et enfin un service pour lancer l'application web intitulé "website" et répondant à la commande python *python manage.py runserver 0.0.0.0 :80*.

A l'aide du Dockerfile et du fichier YAML docker-compose.yml, nous pourrons très simplement lancer tous les services définis de notre application en une seule commande. Celle-ci est typiquement la commande *docker-compose up* pour lancer les services de l'application, et au contraire *docker-compose down* pour les couper. Mais dans notre cas, comme vous pourrez le voir au prochain point, nous avons décidé de créer un Makefile, un fichier , afin de redéfinir ces commandes.

7.6 Makefile

Un fichier "Makefile" est un fichier utilisé par le logiciel Make pour ordonner la séquence d'une compilation de fichiers. En d'autres termes, le Makefile est utilisé pour organiser la procédure de compilation du code.

Make est particulièrement utilisé sous les plateformes UNIX, comme GNU/Linux et MacOS, pour lesquelles il est très aisé de l'installer. D'ailleurs, la plupart des serveurs ayant pour système d'exploitation une distribution Linux, le choix d'utiliser un Makefile va s'avérer utile.

Quant à Windows, il est tout à fait possible d'installer Make en passant par des utilitaires tel que Scoop [86], qui est un installateur en ligne de commandes pour Windows.

Dans notre cas, nous avons créé un petit fichier makefile afin de de rendre l'exécution des commandes Docker plus intuitives et courtes, comme vous pouvez le voir à la figure 7.9.

Makefile

```
1   build:
2       docker build --force-rm $(options) -t epl21232:latest .
3
4   build-prod:
5       $(MAKE) build options="--target production"
6
7   compose-start:
8       docker-compose up --remove-orphans $(options)
9
10  compose-stop:
11      docker-compose down --remove-orphans $(options)
12
13  compose-manage-py:
14      docker-compose run --rm $(options) website python manage.py $(cmd)
```

FIGURE 7.9 – Différentes commandes du Makefile

- **build** La commande *make build* va donc permettre de construire directement "l'image Docker" de notre application qui tournera dans un conteneur par la suite.

- **build-prod** La commande *make build-prod* est similaire à la précédente, sauf que dans la première, seuls les requirements de développement sont installés lors de la construction de l'image, alors qu'avec cette commande, on installera aussi les requirements de production.

  La distinction entre production et développement est majoritairement une question de bonne pratique dans le monde du développement d'application, bien qu'ici cette distinction soit minime voire invisible [78].
- **compose-start** La commande *make compose-start* va permettre de lancer les services de l'application au sein du conteneur Docker.

- **compose-stop** La commande *make compose-stop* va permettre de stopper les services de l'application au sein du conteneur Docker.

- **compose-manage-py** Enfin, La commande *make compose-manage-py* est un peu plus particulière. En effet, cette commande prend un argument sous la forme du string passé dans la variable *cmd*. De cette manière, on peut appliquer les différentes commandes classiques de gestion du site web Django en python à notre service "website" qui tourne dans un conteneur Docker, au travers de cette commande. Par exemple :

  — Appliquer la commande typique pour créer les migrations d'un site web Django *python manage.py makemigrations* devient *make compose-manage-py cmd="makemigrations"* afin de correctement l'appliquer au service dans le conteneur.

  — Ce principe s'applique pour toutes les autres commandes python pour gérer le site Django, comme *python manage.py migrate* qui applique les migrations créées par la commande précédente, et qui devient *make compose-manage-py cmd="migrate"*.

  — Une autre commande utile est celle permettant de créer ce qu'on appelle un "super-utilisateur", typiquement un administrateur, dont la commande python est normalement *python manage.py createsuperuser* et devient ici *make compose-manage-py cmd="createsuperuser"*.

C'est avec ces exemples que nous pouvons conclure la présentation de l'implémentation de l'application web. Au travers de ce chapitre, nous avons pu discuter tout d'abord de la structure générale où nous avons pu définir les différentes relations entre le côté client et la base de données. Nous avons par après décrit les différentes technologies de l'application, ce qu'elles apportent à celle-ci, le fonctionnement de la base de données ainsi que les moyens mis en place pour faire fonctionner l'application web.

Chapitre 8

Tests et validation

Pour pouvoir nous assurer du bon fonctionnement de l'application, il nous est nécessaire de devoir la tester. Dans ce chapitre nous détaillerons les différents tests qui ont été réalisés. Nous avons dans un premier temps réalisé des tests unitaires et dans un second temps réalisé des tests de validation par les clients. Nous aborderons également le cas d'une démonstration réalisée devant les clients ainsi que d'un test réalisé pour analyser le problème de la contrainte de connexion internet en Haïti.

8.1 Tests unitaires

Comme expliqué dans les précédents chapitres de la conception de l'application, nous avons intégré "Django" et "PostgreSQL" dans ce projet.

Par principe, il n'est pas nécessaire de vérifier si les champs sont correctement enregistrés par "Django" car lors de la conception, nous avons logiquement déjà vérifier si la base de données se remplit correctement.

Les tests unitaires vont surtout se focaliser sur les fonctionnalités implémentées autour de l'importation de données. En effet lors de l'importation de données dans l'application, des calculs supplémentaires sont réalisés pour déterminer les moyennes et intensités pluviométriques. Une cinquantaine de tests ont été créés pour déterminer si ces modèles de calcul fonctionnaient correctement.

Pour pouvoir lancer ces tests, nous avons fait appel à la classe "TestCase" qui permet de créer temporairement une petite base de données fictive qui se supprime juste après la réalisation des différents tests. Nous avons donc créé de fausses valeurs que nous ajoutions dans la base de donnée fictive et réutilisé des valeurs pour tester chaque modèle de calcul. Suite à la série de tests nous avons pu déterminer que les modèles de calculs fonctionnent correctement.

8.2 Démonstration devant les clients

Le 27 avril 2021, nous avons eu une réunion avec les clients pour leur présenter l'application. Nous avons, à cette occasion, réalisé une démonstration des fonctionnalités du site web actuellement implémenté et avons décrit celles qui ne l'étaient pas encore. L'objectif de cette réunion

était d'avoir des feedbacks des clients afin de determiner les modifications ou améliorations à apporter aux fonctionnalités actuelles du site web. Ils ont bien posé quelques questions sur les différentes fonctionnalités mais n'ont pas demandé de modifications substantielles. Les clients ont semblé satisfaits de l'application.

8.3 Validation par les clients

Au terme du développement de la première version du prototype de notre application pluviométrique, nous souhaitions faire valider le prototype et recevoir un feedback de la part des clients sur celui-ci.

Pour ce faire, nous avons alors créé un ensemble de scénarios à exécuter sur le site. L'objectif était de simuler toutes les situations auxquelles un utilisateur pourrait être confronté. Dans un deuxième temps, il était prévu de coupler des scénarios à un questionnaire simple de dix questions sur une échelle de Likert [79], un type de questionnaire que l'on appelle une "System Usability Scale" ou SUS en ingénierie des systèmes logiciels [80].

8.3.1 System Usability Scale

La "System Usability Scale" a été inventée en 1986 par John Brooke qui cherchait à créer une échelle de cotation rapide afin de pouvoir évaluer pratiquement tout type de système. Il s'agit donc, comme mentionné avant, d'une échelle de Likert qui comprend 10 questions auxquelles les utilisateurs du système ou de l'application sont invtés à répondre. Ces 10 questions, écrites donc à l'origine en anglais, sont les suivantes :

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Pour notre enquête, pour laquelle nous avons traduit les questions en français, cela signifiera que chaque question placée à côté de l'échelle de Likert ressemblera à ceci :

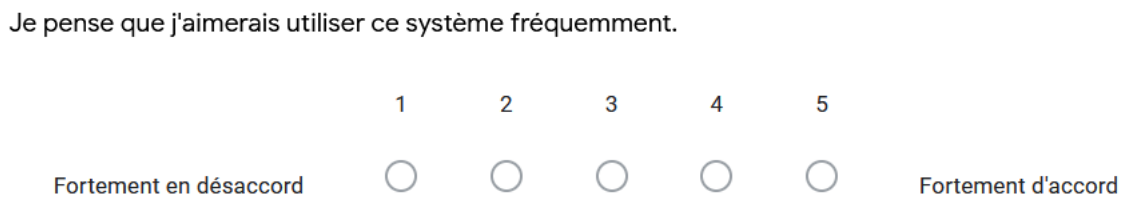


FIGURE 8.1 – Première question de notre enquête SUS

Vous pourrez retrouver en Annexe, à la section 11.5.1, l'ensemble du Google Form qui anous a servi pour avoir des retours sur l'utilisation de l'application avec nos scénarios. Vous pourrez remarquer que certaines "questions" sont intentionnellement posées de façon négative, et donc pour ce genre de phrases, être en désaccord avec celle-ci implique donc bien être d'accord avec la version positive de la phrase. Ces phrases négatives sont généralement posées une fois sur deux, après une question positive.

8.3.2 Scénarios

Concernant les scénarios, ceux-ci ont été créés afin de tester 6 fonctionnalités différentes, et sont entièrement disponibles en Annexe, à la section 11.5.2. Il s'agit de :

- Créer un nouveau compte
- Attribuer un profil à un nouvel utilisateur
- Ajouter une nouvelle station
- Importer de nouvelles données
- Comparer graphiquement les données des moyennes quotidiennes de plusieurs jours
- Envoyer un message d'aide

8.3.3 Analyse des résultats

Voici maintenant la partie la plus impactante de cette section, concernant les résultats de notre enquête. Ceux-ci sont disponibles en annexe au format "brut" dans un tableau Excel, à la section 11.5.3.

L'analyse des résultats va se dérouler en deux temps ; nous commencerons d'abord par afficher sous forme de graphes les résultats afin d'avoir une vue d'ensemble des réponses. Ensuite, nous calculerons ce que l'on appelle le "score" du "System Usability Scale" [80].

La figure 8.2 reprend, sous forme de graphique à barres empilées divergentes, les valeurs en pourcentage des différentes réponses possibles sur l'échelle de Likert pour chaque question.

Nous avons pu utiliser un script de Benoît Duhoux pour générer ce graphe, cela nous a grandement faciliter la tâche mais a pour conséquences que le graphe est en anglais et non en français.

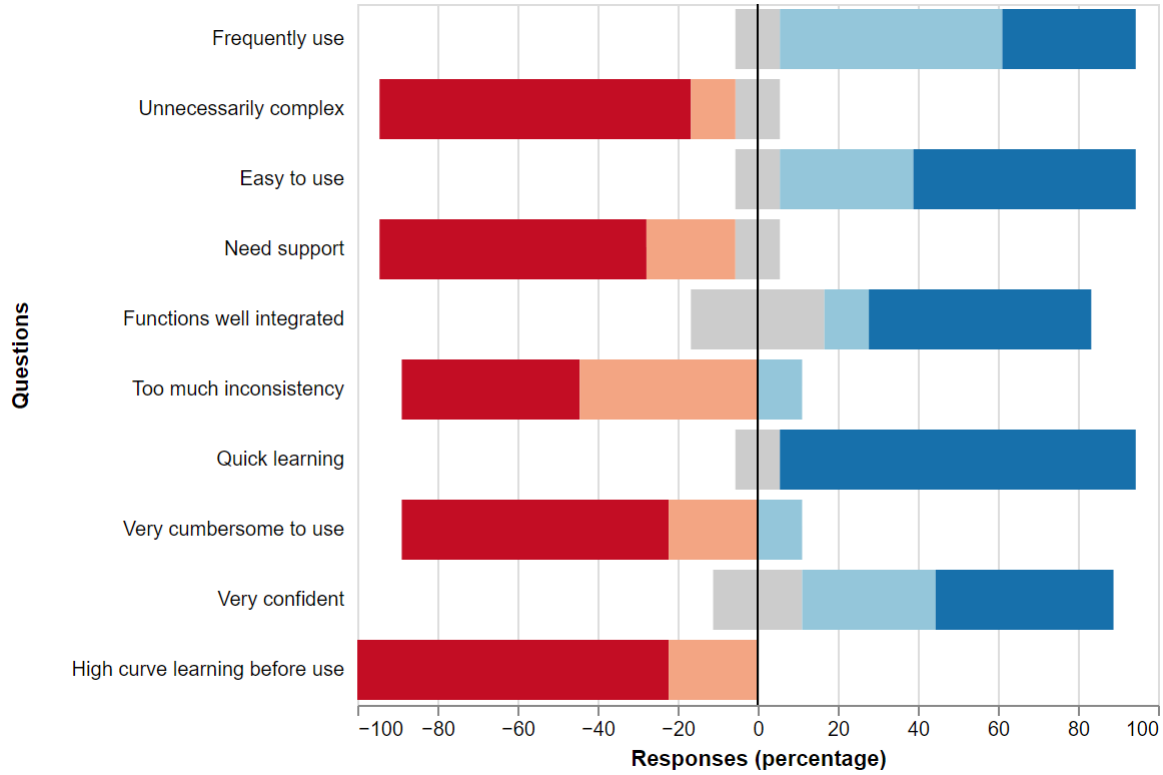


FIGURE 8.2 – Graphique à barres empilées divergentes des réponses aux questions

On peut constater que les résultats sont, à la grande majorité, positifs, situés en moyenne entre ce que l'on pourrait qualifier de "Bien" à "Très bien" pour chaque question.

La question ayant rencontré l'avis le plus mitigé est la question 6, à savoir : "Je pensais qu'il y avait trop d'incohérences dans ce système". Elle a reçu un avis plus négatif sur l'échelle en moyenne par rapport aux autres. Cela nous indique qu'il y a encore du travail à faire pour améliorer l'ergonomie du site, comme par exemple la manipulation des graphes par exemple, afin de la rendre plus cohérente.

D'autre part, les questions 7 et 10, à savoir respectivement : "J'imagine que la plupart des gens apprendraient à utiliser ce système très rapidement." et "J'avais besoin d'apprendre beaucoup de choses avant de pouvoir utiliser ce système." ont toutes deux d'excellentes moyennes, nous indiquant que notre application web semble assez rapide à prendre en main et à comprendre.

Ensuite, nous pouvons calculer le score de notre enquête, en utilisant une formule qui est expliquée au travers de diverses sources [80][81][82]. La formule que nous avons appliquée fonctionne comme suit :

Les utilisateurs auront donc noté chacune des 10 questions ci-dessus entre 1 et 5. Ensuite, pour chacune des questions impaires ou questions dites "positives", on soustrait 1 au score moyen de la question. Pour chacune des questions paires ou questions dites "négatives", on soustrait leur score moyen à 5. Pour finir, il suffit d'additionner le total de ces nouvelles valeurs pour chaque question, et de multiplier le résultat par 2,5. On trouve alors le score SUS, que nous pourrons ensuite comparer à la figure 8.3.

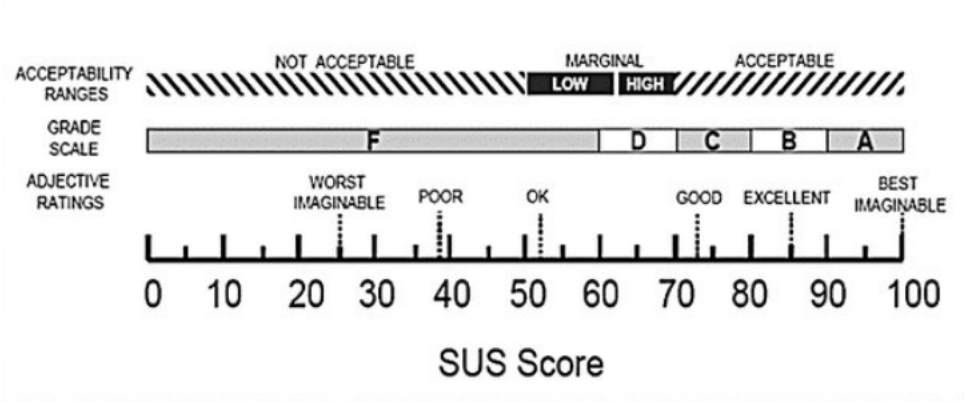


FIGURE 8.3 – Échelle du score pour notre enquête SUS [80]

En appliquant la formule décrite au-dessus aux résultats extraits du tableau Excel en annexe, on obtient ceci :

Numéro de question	1	2	3	4	5	6	7	8	9	10
Valeur moyenne	4,2222	1,3333	4,4444	1,4444	4,2222	1,7778	4,7778	1,5556	4,2222	1,2222

TABLE 8.1 – Moyennes pour chacune des différentes questions

Sur base de ces moyennes, on peut appliquer les soustraction :

Numéro de question	1	2	3	4	5	6	7	8	9	10
Nouvelle valeur moyenne	3,2222	3,6666	3,4444	3,5556	3,2222	3,2222	3,7778	3,44444	3,2222	3,7778

TABLE 8.2 – Nouvelles valeurs avec la formule décrite plus haut

Enfin, on peut alors calculer le score total en sommant les valeurs de la tables 8.2, puis en les multipliant par 2,5, comme suit :

$$34,55555554 * 2,5 = 86,38888885 \approx 86,4$$

Notre score final est donc de 86.4, un score qui selon l'échelle plus haut à la figure 8.3 pourrait être qualifié d'excellent [80]. Mais, il est nécessaire de prendre ce résultat avec le recul qui s'impose, car l'enquête n'a malheureusement été réalisée que par une dizaine de personnes seulement. Conséquemment, cet échantillon est très, voire trop, petit que pour tirer de réelles conclusions statistiques.Néanmoins, le fait que l'application ait été appréciée par les utilisateurs et le client, représente pour nous, un bon indicateur relatif à l'efficacité de l'application.

8.4 Contrainte de connexion internet

Une contrainte imposée dans ce projet était de s'assurer que le site web pouvait être exploité sous un wifi de moins bonne qualité. En effet le réseau en Haïti est de moins bonne qualité que la moyenne.

Pour pouvoir vérifier si cette contrainte est respectée, nous allons utiliser un outil pour simuler une connexion internet semblable à celle présente à Haïti. Nous avons utilisé l'outil de réseau de Google chrome [83]. Avec cet outil il est possible d'assigner des valeurs à des paramètres pour modifier la bande passante via le navigateur chrome.

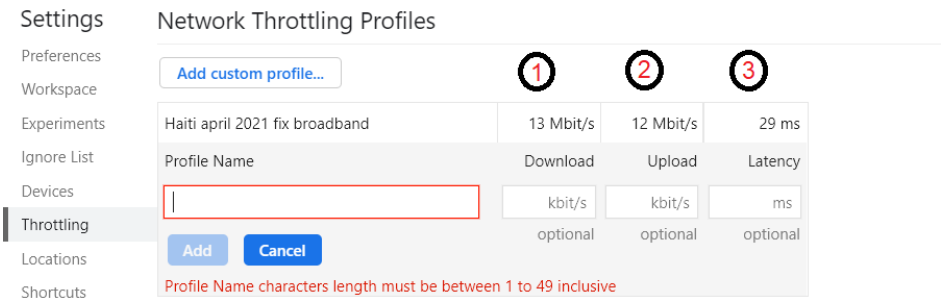


FIGURE 8.4 – Création d'une limite de bande passante sur Google Chrome

Comme on le voit sur la figure 8.4, nous créons une bande passante basée sur 3 paramètres. Ces paramètres sont le nombre de byte de de données téléchargées par secondes (au point 1), le nombre de byte de données chargées par secondes (au point 2) et le temps de latence (au point 3).

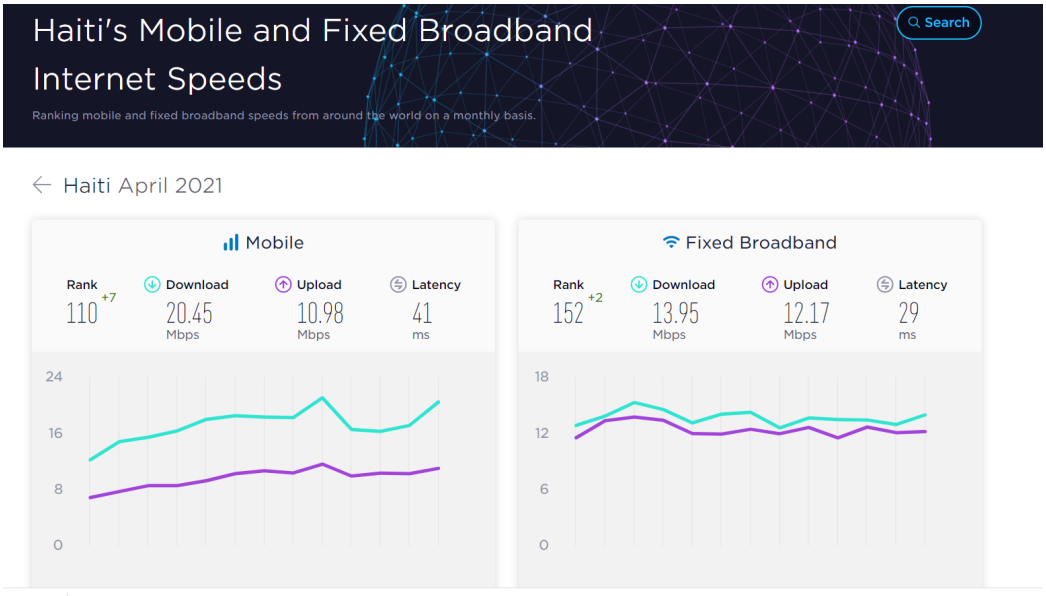


FIGURE 8.5 – Qualité wifi d'Haiti en Avril 2021

Pour savoir quelles valeurs assigner à ces paramètres, nous sommes allés sur le site web "speedtest" qui est un site sur lequel on retrouve pour chaque pays les moyennes des différentes qualités de bandes passantes enregistrées [84]. Nous avons définis nos paramètres basés sur les valeurs qui se trouvent à la figure 8.5. Sur cette figure, on retrouve les paramètres moyens des bandes passantes d'Haiti en avril 2021. Nous avons donc utilisé l'application web avec et sans cette bande passante pour comparer les temps de différences :

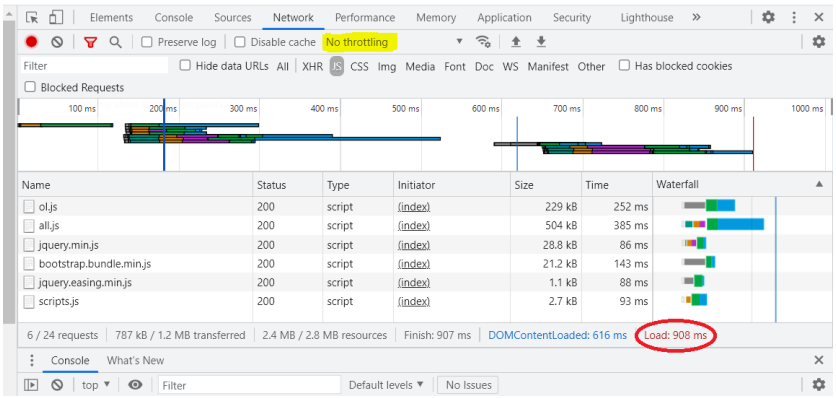


FIGURE 8.6 – Temps de chargement sans bande passante

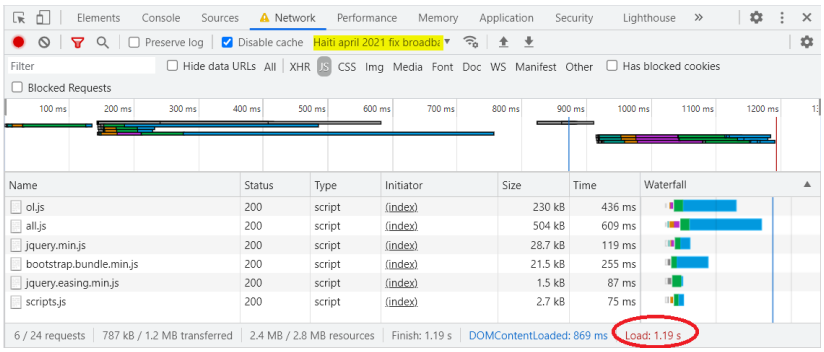


FIGURE 8.7 – Temps de chargement avec bande passante

Sur les figures 8.6 et 8.7, nous avons surligné les limites des bandes passantes testées. Sur la figure 8.6 il s'agit du cas où il n y a pas de limites imposées tandis que sur la figure 8.7 il s'agit du cas où il y a la limite de bande passante.

Le test a été réalisé sur chargement de la page contenant la carte interactive d'Haïti. Nous avons choisit de prendre cette tâche car il s'agit d'une des tâches qui demande beaucoup plus de ressource que les autres tâches. On remarque que la différence entre les deux temps qui sont de 908 ms pour le premier cas et 1,19 s est plutôt négligeable. Ce test a été réalisé plusieurs fois et chacun de ces tests affichent des résultats très similaires.

Les différents tests réalisés sont repris à la fin du projet. A l'analyse de ces tests, que ce soit pour les test unitaires ou pour les tests manuels, nous pouvons conclure que les résultats sont satisfaisants et, par le dernier test réalisé, nous pouvons affirmer que l'application respecte bien la contrainte de mauvais qualité internet qui nous avait été imposée.

Chapitre 9

Pistes d'améliorations

Nous avons jusqu'à présent exposé les différentes facettes du projet, à savoir les problématiques, l'analyse du problème, les fonctionnalités du site, les technologies utilisées ainsi que les tests réalisés pour démontrer le bon fonctionnement de l'application. Bien que les tests se soient montrés positifs, l'application peut être améliorée de plusieurs façons et ce chapitre va se consacrer aux différentes pistes pour améliorer le site.

https ://hydro-blog.com/logiciel-traitement-donnees-hydrologiques-gratuit/

9.1 Calcul des statistiques pluviométriques

Le calcul actuel des intensités et moyennes peut être optimisé. En effet, actuellement, à chaque importation de fichiers de données, le modèle de calcul va calculer de nouveau les intensités et moyennes pour chaque station et date répertoriées dans la base de données. Une amélioration possible serait de limiter le calcul des données uniquement à la station et aux dates répertoriées sur le fichier de données, ce afin d'éviter d'augmenter exponentiellement le temps de calcul à chaque importation de données.

9.2 Apparence générale du site web et ergonomie

Une autre amélioration envisageable serait l'apparence générale du site web. On peut envisager une meilleure structure, ainsi qu'une représentation différente des graphiques et tables. Actuellement, le graphe des intensités représente des données calculées quotidiennement. En pluviométrie, cela peut être intéressant d'afficher, pour chaque durée, les moyennes d'intensités sur différentes années. Aussi, les graphes ne permettent que d'ajouter les données d'une seule date à la fois. On peut optimiser cette fonction en ajoutant la possibilité d'ajouter en une seule fois toutes les données d'une période présentent entre deux dates.

9.3 Frontend responsive

Le frontend peut aussi être amélioré et rendu responsive. Les éléments du frontend s'agencent différemment en fonction de la taille de l'écran. Pour optimiser l'application, une piste serait

d'ajouter la possibilité pour l'application d'adapter l'échelle des composants du frontend en fonction de la taille de l'écran.

9.4 Fonctionnalités hors connexion de l'application

Le développement d'une fonctionnalité qui permettrait de stocker en cache des informations ou données qu'un utilisateur voudrait importer, le temps qu'il retrouve une connexion ; ou le fait que cette fonctionnalité permette aussi de constamment chercher à charger des éléments du site lorsque la connexion est assez instable, se verrait être une idée efficace à implémenter en Haïti.

9.5 Carte interactive

En ce qui concerne la carte interactive, un utilisateur peut seulement survoler des points sur la carte. Une amélioration intéressante serait d'ajouter une liste des différentes stations, et idéalement une division entre différentes régions du pays pour retrouver plus facilement une zone ou station particulière sur la carte. En plus de cette liste, on pourrait ajouter un outil de filtrage des stations pour retrouver directement une station sur la carte.

9.6 Gestion des différents types d'utilisateurs

Nous avons défini différents rôles lors de la conception du projet à savoir les administrateurs, les gestionnaires de stations, les techniciens et les utilisateurs "basiques". Les différences entre certains rôles peuvent être beaucoup plus prononcées et mieux gérées dans la console d'administration.

Chapitre 10

Conclusion

Nous conclurons ce projet de fin d'études, qui a été pour nous une expérience constructive et particulièrement enrichissante, par un bilan technique et humain.

Au départ, nous avions pour objectif la réalisation d'une application web dans le cadre d'un projet humanitaire. Ce projet avait pour but de créer un moyen d'enregistrer et de traiter les données pluviométriques d'Haiti, ainsi que de représenter ces données sous différentes formes accessibles à tous les citoyens de ce pays afin que ces derniers puissent les analyser et en tirer des conclusions bénéfiques.

Après un an d'organisation et de travail, nous avons fini par fournir un résultat concret et exploitable. Pour s'assurer que le résultat obtenu répondait bien aux besoins des clients, une démonstration du produit final a pu être réalisée en leur présence. Nous avons fait tester notre application par des utilisateurs extérieurs qui ont appliqué des scénarios écrits par nos soins dans le but de recueillir un maximum d'avis constructifs. Suite aux avis positifs émis par les utilisateurs qui ont pu tester l'application et ceux des clients qui se sont montrés enthousiastes après l'expérience, nous pensons objectivement que l'objectif de départ a bien été atteint.

En effet, il est maintenant possible pour un utilisateur de sélectionner une station sur une carte interactive pour se rendre sur une page affichant les données propres à cette station. De plus, ces données peuvent être facilement représentées par des graphiques ou par des tableaux. Les données affichées sur les tableaux peuvent être filtrées et traitées dynamiquement en fonctions de dates ou de périodes souhaitées. Et pour finir, toutes ces données peuvent être facilement importées par un gestionnaire de station.

Lors de l'importation de données depuis un fichier, les moyennes ainsi que les intensités sont directement et automatiquement calculées ainsi que stockées dans la base de données. Grâce à ces modèles de calcul, l'application web est performante et rapide lors de l'affichage de données par un utilisateur, étant donné que les valeurs sont déjà calculées. Aussi, un canevas Excel est disponible pour les gestionnaires de stations afin de permettre une meilleure gestion de l'importation des données.

Pour parvenir à ces résultats, nous avons utilisé des technologies puissantes et faciles d'accès. Le but étant de permettre à de futurs étudiants de reprendre le projet et de manipuler ces technologies facilement, ainsi que d'assurer des performances correctes du site web. Parmi ces

technologies, on a utilisé « Docker » pour gérer les différents environnements de l'application web, « Django » et « PostGreSQL » pour gérer la console d'administrateur et la base de données. Nous avons également utilisé « Javascript en frontend » et « Python en backend » qui sont 2 langages connus et faciles à appréhender.

Le projet a été structuré et documenté de façon à ce qu'il soit facilement repris par de futurs étudiants.

Cependant, au terme des tests, certaines remarques intéressantes nous ont été communiquées quant à de possibles améliorations à apporter dans l'application pour la rendre plus intuitive à l'utilisation. Par exemple, l'optimisation du filtre de données sur les graphiques. Certaines remarques ont été facilement corrigées pendant la séance de test mais, par manque de temps, les autres remarques seront retenues comme pistes d'exploitation futures.

Lors de ce projet, nous avons non seulement appris beaucoup de choses concernant la conception d'une application web du début à la fin, que ce soit sur l'étude ou l'utilisation de technologies pour sa réalisation, mais aussi sur les méthodologies de travail adéquates liées à un tel projet. C'est en effet la première fois que nous travaillions sur un projet d'une telle envergure, demandant autant d'investissement et de temps.

Au niveau humain, ce projet a également été un succès dans notre apprentissage professionnel. Dès le début, nous avons pris l'habitude, de travailler régulièrement sur le projet, de nous coordonner et de nous réunir chaque semaine avec les promoteurs pour discuter des avancées accomplies. Grâce aux conseils ciblés et judicieux qui nous ont été donnés, nous avons pu progresser régulièrement dans la bonne direction et ainsi atteindre l'objectif dans les délais prescrits.

L'expérience acquise tout au long de ce projet au niveau organisation de travail, coordination, gestion du temps ainsi qu'au niveau du travail en équipe s'est traduite pour nous par une plus-value humaine qui nous sera bien utile dans la réalisation de nos futurs projets.

Chapitre 11

Annexe

11.1 Manuel

Afin de vous permettre soit d'utiliser notre projet directement, soit de travailler dessus et d'en améliorer son développement, vous trouverez ici un manuel contenant les instructions pour l'installation et le bon fonctionnement de l'application web.

11.2 Installation

Afin de pouvoir faire fonctionner l'application, le plus gros du travail va donc être d'installer Docker Engin et Docker Compose, deux outils de Docker.

Vous devrez aussi vous assurer de pouvoir utiliser Make sur votre machine. Assurez-vous donc que celui-ci soit bien installé.

Installer Make

MacOS Si vous utilisez une des versions du système d'exploitation d'Apple, nous vous recommandons vivement d'installer la dernière version de Make avec Homebrew.

Homebrew est un gestionnaire de paquets pour MacOS, et il fonctionne aussi sous Linux [85]. Il vous suffit alors de simplement vous rendre à l'adresse suivante : `https://brew.sh/index_fr` et de suivre les instructions pour installer Homebrew. Ensuite, vous pourrez écrier la commande suivante dans votre terminal de commande, et Make sera installé automatiquement :

```
brew install make
```

Linux Si vous utilisez une des distributions de Linux, comme par exemple sur un serveur sur lequel vous souhaiteriez faire tourner l'application web, ce sera d'autant plus facile car Make est bien souvent installé de base avec le système d'exploitation.

Quand bien même ce ne serait pas le cas, vous pouvez aussi recourir à Homebrew pour installer la dernière version de Make, étant donné que le programme est aussi bien compatible avec MacOS et Linux. Référez-vous donc au paragraphe précédent. Vous êtes aussi libres d'installer Make avec le gestionnaire de paquets de votre choix.

Windows Concernant Windows, c'est le chemin que vous emprunterez pour installer l'application si vous souhaitez plutôt la faire tourner en local afin de faire du développement et modifier l'application. En effet, les serveurs se reposant sur Windows sont rares.

Pour installer Make dans l'invite de commandes Powershell, nous vous conseillons de passer par Scoop [86]. Vous pouvez trouver leur site web à l'adresse suivante : https ://scoop.sh/. Il s'agit d'un installateur de ligne de commande qui fonctionne de manière similaire à Homebrew, mais pour Windows.

Une fois que Scoop est installé en suivant les indications sur leur site officiel, nous vous invitons à installer Make en ligne de commandes pour Windows comme suit :

scoop install make

Installer Docker Engine

Sur la page d'installation de Docker Engine, qui se trouve à l'adresse `https://docs.docker.com/engine/install/`, vous pourrez retrouver ceci :

Windows / MacOS Si vous utilisez un de ces deux systèmes d'exploitation, il vous suffira simplement d'installer Docker Desktop. Docker Engine vient directement inclus avec cette interface graphique pour Docker.

Linux Si vous utilisez une distribution Linux, il apparaît aussi des liens sur la page vers des tutoriels d'installation précis en fonction de la distribution de Linux que vous utilisez. Docker fournit en effet des fichiers **.deb** ou **.rpm** à cette fin.

Installer Docker Compose

Nous y sommes presque, maintenant que Make et Docker Engine sont installés sur votre machine, il ne vous reste plus qu'à installer Docker Compose et vous pourrez ensuite commencer à utiliser l'application.

Pour cela, je vous invite à vous rendre à l'adresse suivante : `https://docs.docker.com/compose/install/`, il s'agit de l'adresse officielle du guide d'installation de Docker Compose selon votre système d'exploitation.

Cloner le répertoire GitHub

Maintenant, il vous reste à cloner notre code depuis le répertoire GitHub. Notre répertoire GitHub se trouve à l'adresse `https://github.com/nverbois/TFE21-232`.

Vous pouvez par exemple utiliser git en ligne de commande et cloner notre répertoire à l'aide de la commande :

git clone https ://github.com/nverbois/TFE21-232.git

Vous pouvez tout aussi bien le cloner depuis une application de bureau comme GitHub Desktop, ou encore simplement télécharger directement une archive contenant le code.

Important : Une fois que vous avez téléchargé notre code dans un dossier, ouvrez une invite de commandes et placez-vous à la racine du fichier.

Build le code

Finalement, il ne vous reste plus qu'à build notre code. Vous allez pour cela recourir à Make qui a été installé plus haut. Utilisez la commande suivante dans votre invite de commandes pour construire l'image Docker, cela devrait prendre quelques dizaines de secondes tout au plus.

```
Make build
```

Attention : Cette commande ne fonctionnera que si Docker Engine est lancé sur votre machine

11.3 Utilisation

A ce stade, si vous avez suivi le tutoriel précédent étape par étape et que vous avez correctement réussi à construire (build) l'application, nous allons pouvoir maintenant vous guider au travers des quelques lignes de commandes restantes pour gérer le site web. Celles-ci ont en grande partie déjà été introduites à la section **7.6 - Makefile**. De plus, le chapitre **6 - Application web** contient déjà une description détaillée du fonctionnement de chaque interface de l'application.

Allumer et éteindre le serveur

Lorsque Docker Engine est bien allumé sur votre machine et que l'application a été build, il vous suffit de faire une des deux commandes suivantes pour respectivement allumer ou éteindre l'application web :

```
make compose-start
make compose-stop
```

Créer un "super utilisateur"

Ensuite, d'autres commandes vous serons utiles pour la gestion du site, comme la commande permettant de créer un super utilisateur. Il est nécessaire d'utiliser cette commande au moins une fois au début pour pouvoir accéder à l'administration du site web :

```
make compose-manage-py cmd="createsuperuser"
```

Faire des migrations pour la base de données

Si jamais vous modifiez le code des modèles et donc par conséquent les relations des tables stockées en base de données, vous devrez effectuer des migrations. Pour se faire, exécutez d'abord cette commande :

```
make compose-manage-py cmd="makemigrations"
```

Ensuite, si la commande s'est exécutée avec succès, vous pourrez utiliser la commande suivante :

make compose-manage-py cmd="migrate"

Ainsi, vos migrations se seront réalisées avec succès et la base de données sera à jour par rapport à vos modifications sur les tables.

Réinitialiser la base de données au travers de Docker

Si jamais pour une quelconque raison vous deviez recourir à effacer la base de données et la remettre à zéro, vous pouvez utiliser ces commandes dans l'invite de commandes :

docker volume ls

Cette commande listera les différents "volumes" Docker sur votre machine, un volume étant l'équivalent d'une base de données. Dans cette liste, vous devriez apercevoir un volume nommé **tfe21-232_db-data**.

docker volume rm tfe21-232_db-data

Cette commande ensuite retirera le volume, et votre application web devra recréer une nouvelle base de données au prochain démarrage. Assurez-vous que l'application soit bien coupée avant de la lancer, et une fois qu'elle aura été exécutée, l'utilisation des commandes de migration sera peut-être nécessaire.

Exécuter la suite de tests

Si vous souhaitez exécuter la suite de tests que nous avons écrite pour vérifier le bon fonctionnement de notre code et principalement des modèles, vous pouvez utiliser la commande suivante :

make compose-manage-py cmd="test EPL21232.apps.data.tests"

Importer des données

L'importateur de données pluviométriques, se situe dans la console d'administration. Pour y accéder, vous devrez suivre le chemin suivant :

Administration > Données Pluviométriques > Importer

Cet importateur va vous permettre d'importer des données pluviométriques directement dans la base de données sur base d'un fichier. Nous vous recommandons l'utilisation d'un fichier au format xlsx, à savoir le format de fichier de la suite Microsoft Excel.

Pour que l'importateur fonctionne correctement, le fichier devra contenir exactement quatre colonnes, dont la première cellule de chaque colonne contiendra les champs suivants respectivement (l'ordre du nom des champs sur la première ligne du fichier est crucial) :

station, mesure, date, heure.

De plus, le format des cellules est lui aussi très important. Spécifiquement celui de la troisième colonne, à savoir le champ **heure**. En effet, les cellules de cette colonne recevront des heures au format HH :MM :SS, mais le format de la colonne entière devra être du texte dans Excel, et non un des formats d'heure proposés par le programme.

Les trois autres colonnes quant à elles pourront être dans un format Excel qui leur convient logiquement, à savoir du texte pour la colonne **station**, des nombres décimaux pour la colonne **mesure** et des jours pour la colonne **date** au format JJ :MM :AAAA.

Deux fichiers Excel sont disponibles sur le répertoire GitHub pour vous aider, à savoir : un fichier vide servant de canevas, ainsi qu'un fichier rempli de données servant d'exemple.

Clef d'API d'un fournisseur SMTP

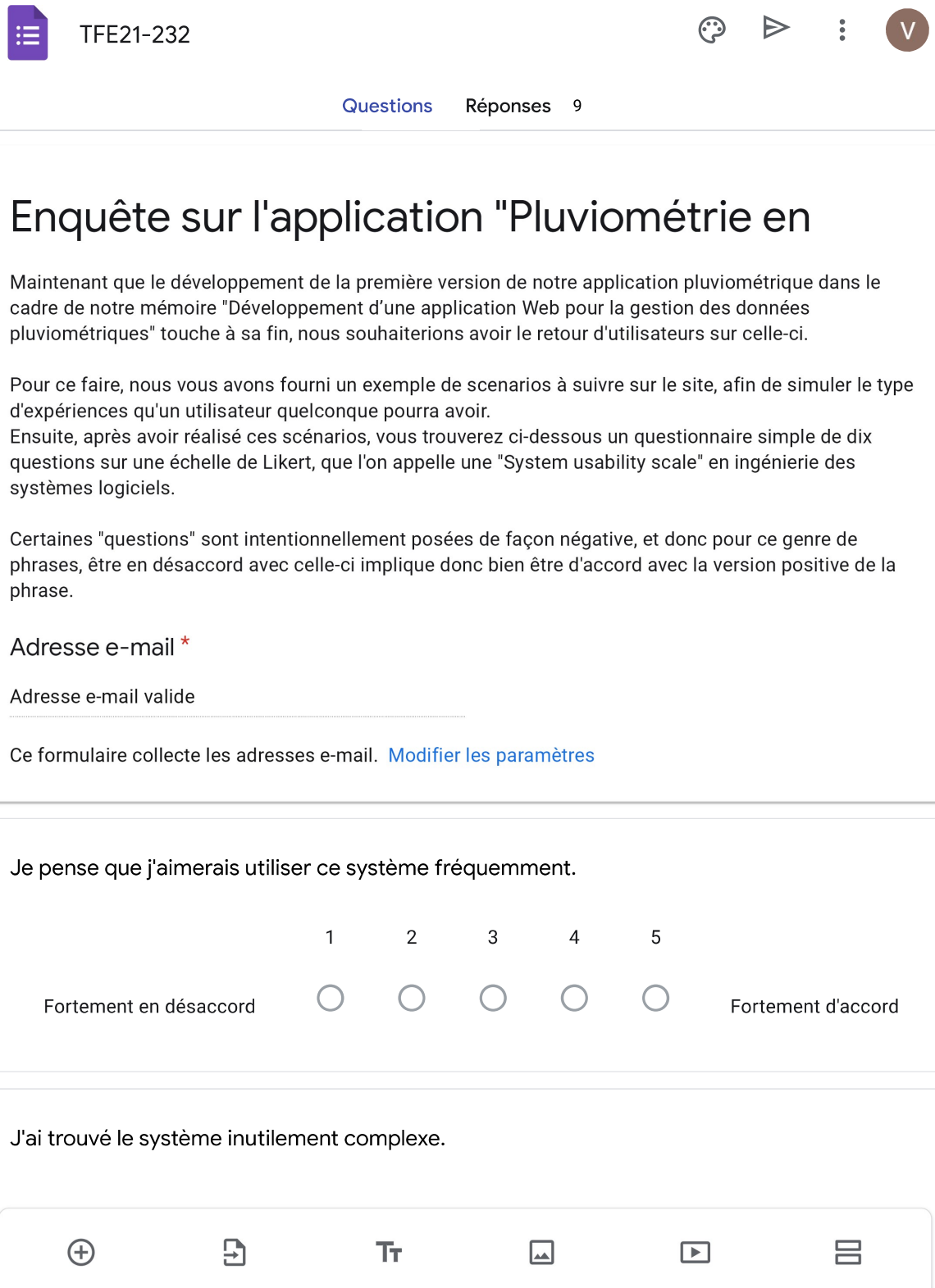
Afin de pouvoir utiliser la fonctionnalité de contact de l'application, il vous faudra inclure au projet une clef d'API d'un fournisseur SMTP. Actuellement, une clef générée par un des mémorants se trouve dans le projet. Vous devrez générer votre propre clef et l'inclure dans les paramètres du site web.

Pour ce faire, rendez-vous par exemple sur SendGrid afin de vous créer gratuitement une telle clef d'API, à l'adresse : `https://sendgrid.com/docs/ui/account-and-settings/api-keys/`.

Ensuite, rendez-vous dans le fichier "settings.py" de notre code pour entrer la valeur de votre clef dans le champ "SENDGRID_API_KEY", voir section 7.4.4.

11.4 System Usability Scale

11.4.1 Formulaire de l'enquête



TFE21-232

Questions Réponses 9

Enquête sur l'application "Pluviométrie en

Maintenant que le développement de la première version de notre application pluviométrique dans le cadre de notre mémoire "Développement d'une application Web pour la gestion des données pluviométriques" touche à sa fin, nous souhaiterions avoir le retour d'utilisateurs sur celle-ci.

Pour ce faire, nous vous avons fourni un exemple de scenarios à suivre sur le site, afin de simuler le type d'expériences qu'un utilisateur quelconque pourra avoir.
Ensuite, après avoir réalisé ces scénarios, vous trouverez ci-dessous un questionnaire simple de dix questions sur une échelle de Likert, que l'on appelle une "System usability scale" en ingénierie des systèmes logiciels.

Certaines "questions" sont intentionnellement posées de façon négative, et donc pour ce genre de phrases, être en désaccord avec celle-ci implique donc bien être d'accord avec la version positive de la phrase.

Adresse e-mail *

Adresse e-mail valide

Ce formulaire collecte les adresses e-mail. Modifier les paramètres

Je pense que j'aimerais utiliser ce système fréquemment.

	1	2	3	4	5	
Fortement en désaccord	○	○	○	○	○	Fortement d'accord

J'ai trouvé le système inutilement complexe.


Questions Réponses 9

	1	2	3	4	5	
Fortement en désaccord	○	○	○	○	○	Fortement d'accord

Je pense que j'aurais besoin du soutien d'une personne technique pour pouvoir utiliser ce

	1	2	3	4	5	
Fortement en désaccord	○	○	○	○	○	Fortement d'accord

J'ai trouvé que les différentes fonctions de ce système étaient bien intégrées.

	1	2	3	4	5	
Fortement en désaccord	○	○	○	○	○	Fortement d'accord

Je pensais qu'il y avait trop d'incohérences dans ce système.

	1	2	3	4	5	
Fortement en désaccord	○	○	○	○	○	Fortement d'accord

J'imagine que la plupart des gens apprendraient à utiliser ce système très rapidement.







Questions Réponses 9

J'ai trouvé le système très lourd ou compliqué à utiliser.

	1	2	3	4	5	
Fortement en désaccord	○	○	○	○	○	Fortement d'accord

Je me suis senti très à l'aise en utilisant le système.

	1	2	3	4	5	
Fortement en désaccord	○	○	○	○	○	Fortement d'accord

J'avais besoin d'apprendre beaucoup de choses avant de pouvoir utiliser ce système.

	1	2	3	4	5	
Fortement en désaccord	○	○	○	○	○	Fortement d'accord






11.4.2 Scénarios

Scénario 1 : Créer un nouveau compte

Rôle : Administrateur

Objectif : Attribuer un nouveau compte à un gestionnaire de station via la console d'administration.

Contexte : Vous êtes Nicolas, un administrateur de l'application web. Un nouveau gestionnaire de station, responsable de la station Epervine, demande à rejoindre l'application en tant que gestionnaire d'application.

Tâches :

1. Connectez-vous à l'application avec vote compte d'administrateur.
2. Rendez-vous dans la console d'administrateur.
3. Créez un nouvel utilisateur pour le responsable de station dans la table d'utilisateurs avec ses identifiants.

Informations nécessaires :

- Votre adresse mail d'administrateur : **nicolas@gmail.ht**
- Votre mot de passe d'administrateur : **NicolasMDP**
- L'adresse mail du nouveau gestionnaire : **adresse mail de votre choix**
- Le mot de passe de son nouveau compte : **mot de passe de votre choix**

Scénario 2 : Ajouter une nouvelle station

Rôle : Responsable de station

Objectif : Ajouter de nouvelles données pluviométriques à une station dans la base de données en utilisant la fonction d'importation de fichiers.

Contexte : Vous êtes un nouveau responsable de station de l'application. Vos accès ont récemment été créés par un administrateur de la plateforme, Nicolas, et vous pouvez maintenant accéder à la console d'administration de l'application. Vous souhaitez alors ajouter une de vos stations à l'application, afin de pouvoir amener des données de cette station sur le site par après.

Tâches :

1. Connectez-vous sur l'application avec le compte que vous avez créé lors de la première étape.
2. Accédez à votre profil utilisateur en cliquant sur votre nom dans la barre de navigation.
3. Accédez ensuite à l'administration.
4. Dans le menu gauche, vous pouvez accéder à la page des "Stations pluviométriques".
5. **Si la station existe déjà, passez à l'étape suivante**. Sinon, cliquez ensuite sur le bouton intitulé "Ajouter Station pluviométrique".
6. Remplissez les champs adéquats selon les informations données juste ci-dessous. Le champ de description peut être laissé vide.

Info nécessaire :

- Nom de la station à ajouter : **Station Epervine**
- Ses coordonnées géographiques : **–72,1536 en longitude**, et **18.8574 en latitude**
- Son nom normalisé : **epervine_station**

Scénario 3 : Attribuer un profil à un nouvel utilisateur

Rôle : Administrateur

Objectif : Attribuer le rôle et la station adéquats à un gestionnaire de station via la console d'administration en lui assignant un nouveau profil.

Contexte : Vous êtes Nicolas, un administrateur de l'application web. Le gestionnaire de station responsable de la station Epervine a rejoint récemment le site web et ajouté sa station, il a maintenant besoin d'un nouveau profil.

Tâches :

1. Connectez-vous à l'application avec vote compte d''administrateur.
2. Rendez-vous dans la console d'administrateur, depuis votre page de profil.
3. Assignez un nouveau profil au gestionnaire de station et assignez-lui le rôle et la station adéquats, à savoir respectivement "gestionnaire de station" et "Station Epervine".

Informations nécessaires :

- Votre adresse mail d'administrateur : **nicolas@gmail.ht**
- Votre mot de passe d'administrateur : **NicolasMDP**
- Adresse mail du gestionnaire : **l'adresse mail de votre compte gestionnaire, à l'étape précédente**

Scénario 4 : Importer de nouvelles données

Rôle : Responsable de station

Objectif : Ajouter de nouvelles données pluviométriques à une station dans la base de données en utilisant la fonction d'importation de fichiers.

Contexte : Vous êtes un responsable de station de l'application. Vos accès ont récemment été créés par un administrateur de la plateforme, et vous pouvez maintenant accéder à la console d'administration de l'application. Vous souhaitez alors ajouter les données pluviométriques de votre station, que vous venez de créer sur l'application, dans la base de données.

Tâches :

1. Connectez-vous sur l'application avec le compte que vous avez créé lors de la première étape.
2. Accédez à votre profil, et accédez ensuite à l'administration.
3. Dans le menu gauche, vous pouvez accéder à la page des "Données pluviométriques".
4. De là, cliquez sur le bouton "Importer".
5. Choisissez comme fichier celui que nous vous avons fourni.
6. Sélectionnez ensuite le format de fichier adéquat, à savoir **xslx**.
7. Choisissez la station parmi le menu déroulant.
8. Lancez l'importation, confirmez que les données à importer vous semblent correctes et attendez la validation de l'importation.

Informations nécessaires : Vous devrez utiliser le fichier Excel qui a été fourni avec ce document, intitulé **Station Epervine.xslx**, afin d'effectuer cette tâche.

Scénario 5 : Comparer graphiquement les données des moyennes quotidiennes de plusieurs jours

Rôle : Utilisateur

Objectif : Utilisation correcte des graphiques de données.

Contexte : Vous êtes Vincent, un étudiant analysant les données pluviométriques d'Haïti. Vous désirez connaitre les différentes précipitations moyennes quotidiennes de la région autour de la station Saut-math. Pour ce faire vous décidez d'afficher les moyennes, maximums et minimums journaliers de la semaine du 22 mars au 28 mars 2015.

Tâche :

1. Retrouvez la station de donnée Saut-math sur la carte.
2. Accédez à sa page de données.
3. Rendez-vous sur le graphique des moyennes journalières.
4. Supprimez toutes les données inutiles à votre recherche.
5. Ajoutez toutes les données utiles à votre recherche.

Information nécessaire :

Aucune information particulière nécessaire pour réaliser ce scénario.

Scénario 6 : Envoyer un message d’aide

Rôle : Utilisateur

Objectif : Utiliser la fonctionnalité de contact pour envoyer un message d’aide

Contexte : Vous êtes Vincent, un étudiant analysant les données pluviométriques d’Haïti. Vous souhaiteriez avoir accès aux données brutes d’une station sous forme de tableau. Mais en tant que simple utilisateur, vous n’y avez pas accès. Vous décidez de demander aux administrateurs du site si vous pouvez y avoir accès.

Tâche :

1. Rendez-vous sur la page de contact.
2. Remplissez les champs requis et écrivez votre message d’aide.

Information nécessaire :

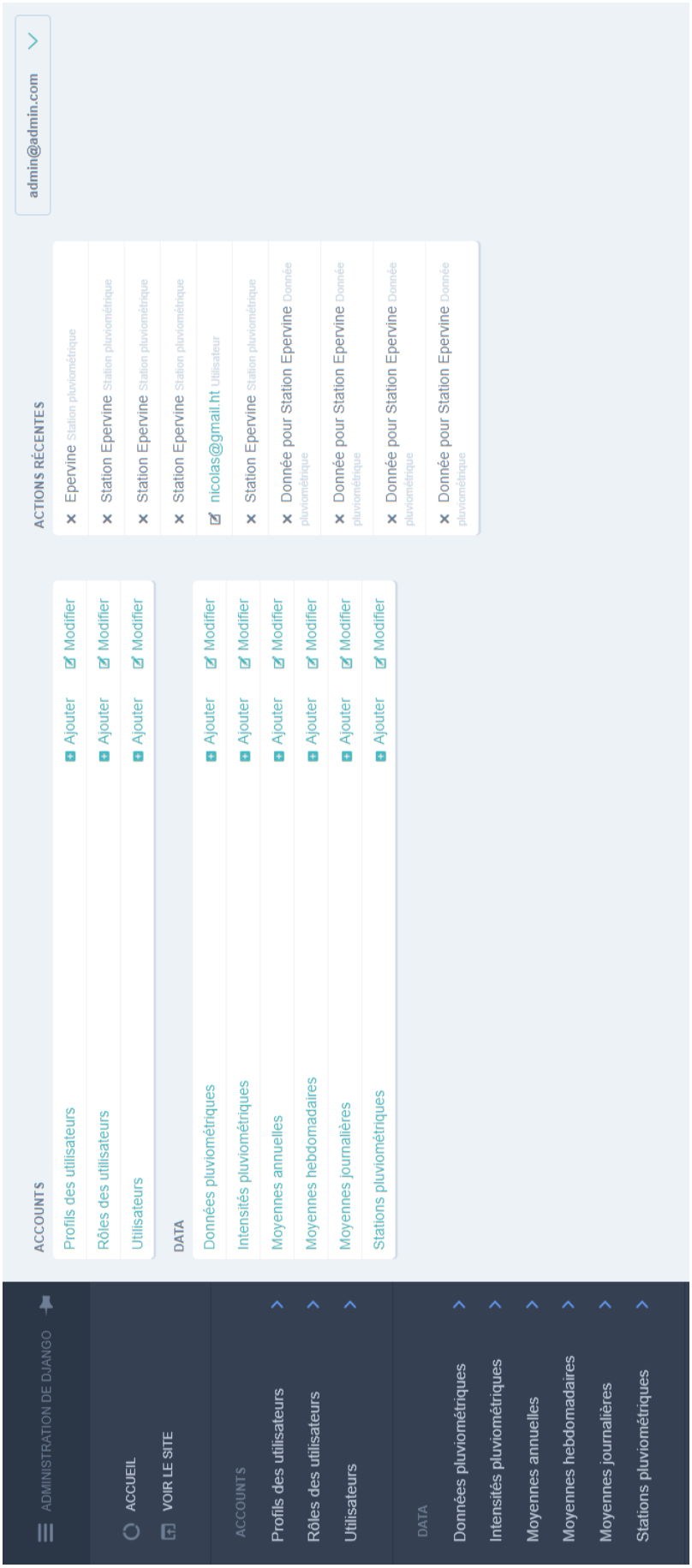
- Votre adresse mail : **vincent@gmail.ht**

11.4.3 Résultats bruts de l'enquête

	A	B	C	D	E	F	G	H	I	J	K	L
1	Horodateur	Adresse e-mail	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10
2	5/21/2021 14:31:46		4	3	3	2	4	2	3	4	3	1
3	5/26/2021 9:39:32		3	1	4	1	3	2	5	2	4	1
4	5/26/2021 12:06:02		4	1	5	1	5	1	5	1	5	1
5	5/27/2021 17:18:38		4	2	5	1	5	4	5	1	5	1
6	5/28/2021 4:28:48		5	1	5	2	5	1	5	1	5	1
7	5/28/2021 16:14:15	gethro.dauphin1(	4	1	5	1	3	2	5	1	5	1
8	5/28/2021 19:15:43	fredgeffrard01@	5	1	5	1	5	1	5	1	4	2
9	5/29/2021 12:34:03	dieuvenson105@	5	1	4	3	5	1	5	2	4	2
10	5/31/2021 20:29:36	sandra.soares-fi	4	1	4	1	3	2	5	1	3	1

FIGURE 11.1 – Tableau des réponses reçues à notre enquête

11.5 Apparence de la console d'administration avec Django-Jet



Glossaire

Application :	Selon le dictionnaire Larousse, il s'agit d'un "programme ou ensemble de programmes destiné à aider l'utilisateur d'un ordinateur pour le traitement d'une tâche précise". Dans notre cas, il s'agit plus précisément de l'ensemble des pages de l'application web ainsi que de la base de données, constituant l'outil informatique pour la gestion des données pluviométriques.
Backend:	Aussi appelé "Côté serveur" en français, il s'agit de la partie d'une application ou d'un site web concernant tout ce qui s'occupe de stocker et de gérer les données [14].
Base de données:	Il s'agit d'une structure informatique permettant d'enregistrer une collection d'informations organisées, dans le but de pouvoir accéder facilement à ces informations, ou données, et de les restituer. Elle fait donc partie du backend.
Framework:	"Ensemble d'outils constituant les fondations d'un logiciel informatique ou d' une application web, et destiné autant à faciliter le travail qu'à augmenter la productivité du programmateur qui l'utilisera" [13].
Frontend:	Aussi appelé "Côté client" en français, il s'agit de la partie d'une application ou d'un site web avec laquelle un utilisateur va interagir [14].
Interface :	Selon le dictionnaire Larousse, il s'agit d'un "module matériel ou logiciel permettant la communication d'un système avec l'extérieur". En pratique dans notre cas, les interfaces seront graphiques sous la forme de pages web contenant des boutons, graphes, etc. Les interfaces font donc partie du frontend.

Librairie :	"Aussi appelée "bibliothèque logicielle", ce que l'on qualifie de librairie au travers de ce mémoire est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire. Les fonctions sont regroupées de par leur appartenance à un même domaine conceptuel (mathématique, graphique, tris, etc)" [26].
Multithreading :	Capacité d'un programme à pouvoir exécuter plusieurs fils d'exécution à la fois ou à répondre à plusieurs requêtes à la fois.
Patron de conception	En développement logiciel, un patron de conception est un plan ou un schéma qui présente une solution à un problème commun de conception d'un logiciel. Il s'agit donc d'une solution "standard" que nous pouvons modifier selon nos besoins afin de résoudre un problème.
Plug-in (ou plugin) :	"Un petit logiciel qui se greffe à un programme principal pour lui conférer de nouvelles fonctionnalités" [61].
Requirements:	Ensemble des logiciels et programmes nécessaires au bon fonctionnement d'un logiciel ou d'une application.
Widget :	Selon le dictionnaire Le Robert, un widget est une "application interactive qui permet l'affichage d'informations variées (calendrier, météo...) ou l'accès à des services (actualité, liens...)".

Bibliographie

[1] La Banque Mondiale, 2021, *La Banque mondiale en Haïti*, La Banque Mondiale, consulté le 20 Février 2021, <https://www.banquemondiale.org/fr/country/haiti/overview>

[2] Ansotte Gaëtan, Bailly Roland, Ricci Dorian, 2019, *Développement d'une application Web pour la gestion des données pluviométriques*, Ecole polytechnique de Louvain, Université catholique de Louvain, Prom. : Mens Kim, Soares Frazao Sandra, <http://hdl.handle.net/2078.1/thesis:17745>

[3] Pierre Hubert, Gaston Réméniéras, 2021, *Hydrologie*, Encyclopædia Universalis, consulté le 28 Février 2021, <https://www.universalis.fr/encyclopedie/hydrologie/>

[4] Encyclopædia Universalis, 2021, *Pluviométrie*, Encyclopædia Universalis, consulté le 28 Février 2021, <https://www.universalis.fr/dictionnaire/pluviometrie/>

[5] Soares-Frazao Sandra, 2020, *LGCIV2051 HY2 - Measurement and analysis of rain*, Youtube, consulté le 29 Février 2021, <https://www.youtube.com/watch?v=xBaGe7UN0U0>

[6] Wikipédia (User :LoKiLeCh), 20 décembre 2005, *Pluviométrie : pluviomètre à lecture directe*, consulté le 30 Février 2021, <https://fr.wikipedia.org/wiki/Pluviom%C3%A8tre#/media/Fichier:Regenmesser.jpg>

[7] Wikipédia (User :CambridgeBayWeather), 26 juillet 2005, *Pluviométrie : mécanisme d'un pluviomètre à augets basculants*, consulté le 30 Février 2021, <https://fr.wikipedia.org/wiki/Pluviom\%C3\%A8tre#/media/Fichier:Interior_tipping_bucket.JPG>

[8] Service public de Wallonie, 2021, *Direction générale opérationnelle de la Mobilité et des Voies hydrauliques : Graphique des valeurs journalières*, consulté le 5 Mars 2021, <http://voies-hydrauliques.wallonie.be/opencms/opencms/fr/hydro/Archive/annuaires/statjourgraph.jsp?code=95960015&PG=Graphique>

[9] Bounahed Robert, Savatier Jérémy, 13 Janvier 2003, *REJETS URBAINS PAR TEMPS DE PLUIE (RUTP)*, consulté le 6 Mars 2021, <http://hmf.enseeiht.fr/travaux/CD0203/travaux/optsee/bei/7/pics/idfqa.jpg>

[10] ExpressJS, 2021, *Express : Infrastructure Web minimaliste, souple et rapide pour Node.js*, consulté le 1er Mars 2021, <https://expressjs.com/fr/>

[11] MongoDB, 2021, *The database for modern applications*, consulté le 1er Mars 2021, <https://www.mongodb.com/>

[12] Service public de Wallonie, 2021, *Direction générale opérationnelle de la Mobilité et des*

Voies hydrauliques : Annuaires et statistiques, Service public de Wallonie, consulté le 5 Mars 2021, <http://voies-hydrauliques.wallonie.be/opencms/opencms/fr/hydro/Archive/annuaires/index.html>

[13] linternaute, 2021, *Définition du mot framework*, linternaute, consulté le 02 mars 2021, <https://www.linternaute.fr/dictionnaire/fr/definition/framework/>

[14] geeksforgeeks, 09 Mar, 2021 , *frontend vs backend*, geeksforgeeks, consulté le 12 mars 2021 <https://www.geeksforgeeks.org/frontend-vs-backend/>

[15] Hiren Dhaduk, January 5, 2021, *Best Frontend Frameworks of 2021 for Web Development*, simform, consulté le 03 mars 2021, <https://www.simform.com/best-frontend-frameworks/>

[16] Lal Verma, Jul 12, 2020, *Most Popular FrontEnd Frameworks, lalverma*, consulté le 28 février 2021, <https://lalverma.medium.com/most-popular-frontend-frameworks-51441986395>

[17] angular,2010-2021, *The modern web developer's platform*, angular,consulté le 28 février 2021 <https://angular.io/>

[18] angular,2010-2021, *TypeScript configuration*, angular,consulté le 17 novembre 2020 <https://angular.io/guide/typescript-configuration>

[19] typescriptlang,2012-2021,*Typed JavaScript at Any Scale, typescriptlang*, consulté le 29 octobre 2020 <https://www.typescriptlang.org/>

[20] Younes Jaaidi,2021,*Pourquoi Angular ?*, guide-angular, consulté le 28 février 2021 <https://guide-angular.wishtack.io/pourquoi-angular>

[21] Anton Vynogradenko, May 2021, *framework-sizes*, github, consulté le 03 mars 2021 <https://gist.github.com/Restuta/cda69e50a853aa64912d>

[22] React, 2021, *A JavaScript library for building user interfaces*, React , consulté le 03 mars 2021 <https://reactjs.org/>

[23] hotframeworks, 2021,*Find your new favorite web framework*, hotframeworks, consulté le 02 mars 2021 <https://hotframeworks.com/>

[24] React, 2021, Introducing JSX,React, consulté 2021, <https://reactjs.org/docs/introducing-jsx.html>

[25] vuejs,2014-2021, *Vue.js : The Progressive JavaScript Framework*, vuejs, consulté le 29 novembre 2020, <https://vuejs.org/>

[26] Techno-Science,23 décembre 2019,*Bibliothèque logicielle - Définition et Explications*, Techno-Science, consulté le 29 novembre 2020 <https://www.techno-science.net/definition/1470.html>

[27] Hiren Dhaduk, January 5, 2021, *Best Frontend Frameworks of 2021 for Web Development*,simform, consulté le 14 mars 2021 <http://www.referencement-auto.com/developper-son-appli-en-react-avantages-et-inconvenients/>

[28] rubyonrails,2008-2021,*Ruby on Rails Guides*,rubyonrails, consulté le 17 novembre 2020, <https://guides.rubyonrails.org/>

[29] Sonoo Jaiswal,2011-2021,*Ruby on Rails MVC*,javatpoint, consulté le 2 décembre 2020 <https://www.javatpoint.com/ruby-on-rails-mvc>

[30] rj45, 8 Apr 2008,*Simple Example of MVC (Model View Controller) Design Pattern for Abstraction*,codeproject,consulté le 2 décembre 2020, <https://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design>

[31] Mathieu Nebra, 2020,*Comment fonctionne une architecture MVC ?*, openclassrooms,consulté le 30 mars 2021 <https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php/4678736-comment-fonctionne-une-architecture-mvc>

[32] Django Software Foundation , 2005-2021, *Django makes it easier to build better Web apps more quickly and with less code*, Django Software Foundation, consulté le 02 decembre 2020 <https://www.djangoproject.com/>

[33] smartdraw, 1994-2021,*Entity Relationship Diagram*, smartdraw, consulté le 30 mai 2021 <https://www.smartdraw.com/entity-relationship-diagram/>

[34] Christophe de Base de données, 8 mars 2018,*ORM*, Base de données, consulté le 17 avril 2021 <https://www.base-de-donnees.com/orm/>

[35] Nijssen Siegfried, 2020,*Base de données cours 1*, UCL-moodle, consulté le 12 avril 2021 <LSINF1225-Cours1-BasededonnÃĭes>

[36] Bases de données, J.-L. Hainaut, 2009, *Bases de données*, Dunod, consulté le 12 avril 2021 <BasesdedonnÃĭes,J.-L.Hainaut,2009,Dunod>

[37] solid IT, 2020-2021, *DB-Engines Ranking*, solid IT, consulté le 17 novembre 2020 <https://db-engines.com/en/ranking>

[38] Akademily, Jul 24, 2020, *The most popular DBMS (database management systems) in the world in 2020*, paggyru-medium, consulté le 02 mars 2021 <https://paggyru.medium.com/the-most-popular-dbms-database-management-systems-in-the-world-in-2020-14ec2ebe8dc7>

[39] Priya Pedamkar, 2020-2021,*Introduction to Relational Database Advantages*, educba, consulté le 07 mars 2020 <https://www.educba.com/relational-database-advantages/>

[40] Jamsheer K, 26 FEB 2020,*The advantages and disadvantages of RDBMS*, acodez, consulté le 07 mars 2020 <https://acodez.in/advantages-disadvantages-rdbms/>

[41] Django,2005-2021, *Django non-rel, Django*, consulté le 28 février 2021 <https://django-nonrel.org/>

[42] Haerder, T. ; Reuter, A. (1983). *"Principles of transaction-oriented database recovery"*. ACM Computing Surveys. 15 (4) : 287. consulté le 17 mars 2021

[43] EnterpriseDB,(2016, Sep 29), *New community version of PostgreSQL 9.6 boosts performance with parallel query and vertical and horizontal scalability features*, PR Newswire, consulté le 10 mars 2021 <https://www.prnewswire.com/news-releases/

new-community-version-of-postgresql-96-boosts-performance-with-parallel-query-and-verti
html>

[44] Asaf Yigal Nov 8, 2018 , *Sqlite vs. MySQL vs. PostgreSQL : A Comparison of Relational Databases*, logz.io, consulté le 28 février 2021 <https://logz.io/blog/relational-database-comparison/>

[45] docker, 2013-2021, *docker site web*, docker, consulté le 04 decembre 202<https://www.docker.com/>

[46] Sarabjeet Singh, 2020, *Advantages of Docker*,Educba, consulté le 4 décembre 2020 <https://www.educba.com/advantages-of-docker/>

[47] Jennifer Seaton, Oct 09, 2020, *6 Reasons to Use Docker Virtualization Software*, makeuseof, consulté le 25 février 2021 <https://www.makeuseof.com/reasons-to-use-docker/>

[48] Open Source Initiative, 1998-2021, *2-Clause BSD License*, Open Source Initiative , consulté le 29 mai 2021 <https://opensource.org/licenses/BSD-2-Clause>

[49] Free Software Foundation, Inc, 19 Novembre, 2007, *Licence publique générale GNU Affero*, gnu.org,consulté le 02 juin 2021 <https://www.gnu.org/licenses/agpl-3.0.fr.html>

[50] Open Source Initiative, 1998-2021, *The MIT License*, Open Source Initiative, consulté le 30 mai 2021 <https://opensource.org/licenses/MIT>

[51] Django Software Foundation, 2005-2021, *Documentation de Django*, Django Software Foundation, consulté le 27 février 2021 2021<https://docs.djangoproject.com/fr/3.2/>

[52] Django Software Foundation, 2005-2021,*Balises et filtres de gabarit intégrés*, Django Software Foundation, consulté le 06 mars 2021 <https://docs.djangoproject.com/fr/3.2/ref/templates/builtins/>

[53] Django Software Foundation, 2005-2021,*Le langage de gabarit de Django*, Django Software Foundation, consulté le 06 mars 2021 <https://docs.djangoproject.com/fr/3.2/ref/templates/language/>

[54] Bootstrap team, 2011-2021, *Build fast, responsive sites with Bootstrap*, Bootstrap team, consulté le 18 mars 2021<https://getbootstrap.com/>

[55] github, 2021,*Resultat de recherche de répertoires contenant le mot 'stars'*, github, consulté le 18 mars 2021. <https://github.com/search?o=desc&q=stars\%3A\%3E1&s=stars&type=Repositories>

[56] Start Bootstrap team, 2021, *Bootstrap themes, templates, and UI tools to help you start your next project !*, Start Bootstrap LLC, consulté le 18 mars 2021 <https://startbootstrap.com/>

[57] Google, 2020-2021,*Google Maps Platform*, Google Developers, consulté le 26 février 2021 <https://developers.google.com/maps/>

[58] Mapbox, 2010-2021, *Maps and location for developers*, Mapbox, consulté le 03 mars 2021<https://www.mapbox.com/>

[59] MetaCarta, 2006-2021, *Openlayers website*, Open Source Geospatial Foundation, consulté le 17 mars 2021 <https://openlayers.org/>

[60] SpryMedia Ltd, 2007-2021, *Add advanced interaction controlsto your HTML tables the free easy way*, SpryMedia Ltd, consulté le 30 mars 2021 <https://www.datatables.net/>

[61] futuratech, 2001-2021, *Plug-in : qu'est-ce que c'est ?*, futura science, consulté le 04 juin 2021 <https://www.futura-sciences.com/tech/definitions/informatique-plug-in-4235/>

[62] OpenJS Foundation and jQuery, 2006-2021, *jQuery website*,OpenJS Foundation and jQuery, consulté le 16 mars 2021 <https://jquery.com/>

[63] Mozilla, 2005-2021, *AJAX Documentation*, Mozilla, consulté le 23 avril 2021 <https://developer.mozilla.org/fr/docs/Web/Guide/AJAX>

[64] Refsnes Data, 1999-2021, *AJAX Introduction*, w3schools, consulté le 23 avril 2021, <https://www.w3schools.com/js/js_ajax_intro.asp>

[65] Soares-Frazao Sandra, 2020, *LGCIV2051 HY2 - Measurement and analysis of rain*, Youtube, consulté le 29 Février 2021, <youtube.com/watch?v=xBaGe7UN0U0>

[66] Individual contributors, 2020-2021, *Chartjs website*, chartjs, consulté le 12 avril 2021 <https://www.chartjs.org/>

[67] Django Jet, 20215-2021, *The missing Django admin interface for modern professionals*, Django Jet, consulté le 02 avril 2021 <http://jet.geex-arts.com/>

[68] Free Software Foundation, Inc, 31 mai 2021, *Licence publique générale GNU Affero*,Free Software Foundation, Inc, consulté le 01 juin 2021 <http://www.gnu.org/licenses/agpl-3.0.html>

[69] Individual contributors, 2020-2021, django-import-export, github, consulté le 19 mars 2021 <https://github.com/django-import-export/django-import-export/>

[70] The PostgreSQL Global Development Group, 1996-2021 , *PostgreSQL : The World's Most Advanced Open Source Relational Database*, The PostgreSQL Global Development Group, consulté le 09 mars 2021 <https://www.postgresql.org/>

[71] Twilio SendGrid, 2021, *Sendgrid Documentation*, SendGrid, consulté le 27 mars 2021 <https://sendgrid.com/docs/>

[72] Mailgun Technologies, Inc., 2021, *mailgun*, mailgun , consulté le 15 mars 2021<https://www.mailgun.com/>

[73] Amazon Web Services, Inc., 2021, *Amazon Simple Email Service*, Amazon, consulté le 15 mars 2021 <https://docs.aws.amazon.com/fr_fr/ses/latest/DeveloperGuide/send-email-smtp.html>

[74] Jonathan B. Postel, 1982, *SIMPLE MAIL TRANSFER PROTOCOL*, Information Sciences Institute - University of Southern California, consulté le 14 Mai 2021, <https://richardg.users.greyc.fr/fr/ens/Res1-TD2_rfc788.pdf>

[75] Docker Inc.,2013-2021,*Overview of Docker Compose*,docker docs, consulté le 01 mars 2021<https://docs.docker.com/compose/>

[76] Docker Inc.,2013-2021,*Docker Engine overview*,docker docs, consulté 1 mars 2021 <https://docs.docker.com/engine/>

[77] lukesampson, 2020-2021, *Scoop website*, github, consulté le 27 février 2021<https://scoop.sh/>

[78] Docker Inc.,2013-2021,*Docker development best practices*,docker docs, consulté le 27 février 2021 <https://docs.docker.com/develop/dev-best-practices/>

[79] Trochim, William M. K. (October 20, 2006). *"Likert Scaling"*. Research Methods Knowledge Base, 2nd Edition. consulté le 30 avril, 2021.

[80] J. Brooke,1996, *Sus : a quick and dirty usability,* Usability evaluation in industry, vol. 189. consulté le 30 avril 2021

[81] J. Sauro,2011, *A practical guide to the system usability scale : Background, benchmarks best practices.* Measuring Usability LLC. consulté le 12 mai 2021

[82] Nathan Thomas, 2011-2020, *How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website*, usabilitygeek, consulté le 12 mai 2021 <https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/>

[83] Morgan Fuchs, 2018-2020, *Comment simuler une bande passante pour tester un site internet*, Morgan FUCHS,consulté le 20 mai 2021 <https://www.morganfuchs.fr/comment-simuler-une-bande-passante-pour-tester-un-site-internet/>

[84] Ookla, Avril 2021, *Haiti's Mobile and Fixed Broadband Internet Speeds*, speedtest, consulté le 20 mai 2021, 2021<https://www.speedtest.net/global-index/haiti#mobile>

[85] Max Howell, 2009-2021, *Homebrew website*, Homebrew, consulté le 13 mars 2021 <https://brew.sh/index_fr>

[86] lukesampson, 2020-2021, *Scoop website* , github, consulté le 13 mars 2021 <https://scoop.sh/>

[87] Docker Inc.,2013-2021,*Install Docker Engine*,docker docs, consulté le 07 mars 2021 <https://docs.docker.com/engine/install/>

[88] Docker Inc.,2013-2021,*Install Docker Compose,docker docs*, consulté le 07 mars 2021 <https://docs.docker.com/compose/install/>

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**