Nom : **FLEURY CALAIS**
Prénom : **Pierre**

Année : **2021 - 2022**

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

IUT DE VÉLIZY

Rapport de stage de deuxième année d'IUT Informatique

Stage de développement web au laboratoire DAVID de l'UVSQ

4 avril 2022 - 15 juin 2022

Tutrice de stage : **ZEITOUNI Karine**
Professeur encadrant : **HOGUIN Fabrice**

Remerciement

Alors que le stage au laboratoire DAVID se termine, je tiens à remercier Karine ZEITOUNI pour avoir proposé ce stage et pour la confiance dont elle a fait preuve à mon égard pendant toute la période où j'ai travaillé à ses côtés. Je veux également remercier l'ensemble des membres du laboratoire DAVID pour m'avoir accueilli et pour tous les moments que l'on a passés ensemble.

Sommaire

Résumé / Summary

Je suis rentré au laboratoire DAVID de l'Université de Versailles-Saint-Quentin-en-Yvelines dans le cadre de mon stage de fin d'étude à l'IUT Informatique de Vélizy. Le stage a pour sujet Développement d'interfaces de chargement, de prétraitement et de visualisation de données issues d'un capteur et d'une application mobile et a commencé le 4 avril 2022. Il devait initialement se terminer le 27 mai mais il a été prolongé jusqu'au 15 juin pour me permettre d'achever pleinement mon travail. Durant cette période j'ai participé au projet de recherche Polluscope en créant un programme pour générer des rapports de façon automatique et en refactorisant la plateforme de données du projet Polluscope en langage Python Flask. Plus qu'une expérience dans le milieu professionnel en tant que membre actif du laboratoire, ce stage a été riche en apprentissage sur le plan technique et m'a permis de découvrir le monde de la recherche en rencontrant des personnes aux parcours variés mais partageant la même passion pour l'informatique

I entered in the DAVID laboratory of the Université de Versailles-Saint-Quentin-en-Yvelines in the context of my internship of end studies in computer sciences at the IUT of Velizy. The internship subject is Development of loading, pretreatement and visualization interface for data from sensor and of mobile application and began the April 4th 2022. Initially it was suppose to last until the May 27th but was extended to the June 15th to allow me to fully achieve my work. During this time, I participated to the research project Polluscope by creating a program of auto report generation and by refactoring the Polluscope project's data platform in the Python Flask language. More than an experience in a professionnal environment as an active laboratory's member, this internship was rich in learning on the technical aspect and allowed me to discover the research world by meeting peoples with varied career history but sharing the same passion for computer sciences.

Introduction

Le présent rapport a pour fonction de présenter l'entreprise ayant accepté de m'embaucher pour mon stage de fin d'étude en IUT informatique. L'ensemble des éléments de ce rapport sont accessibles via le dépôt Github présent en sitographie.
J'ai été accepté au **laboratoire DAVID de l'Université de Versailles-Saint-Quentin-en-Yvelines** en répondant à l'offre de stage Développement d'interfaces de chargement, de prétraitement et de visualisation de données issues d'un capteur et d'une application mobile de l'équipe ADAM.
Ce stage a débuté le 4 avril 2022 et a duré deux mois. Ce stage constitue ma première expérience dans le milieu professionnel en tant que membre actif d'une organisation et a pour objectif de valider mes deux ans d'étude à l'IUT informatique de Vélizy et d'obtenir le Diplôme Universitaire de Technologie en Informatique.
Dans un premier temps nous verrons une présentation du laboratoire DAVID de l'UVSQ avant de présenter le travail effectué durant la période de stage puis de finir par une conclusion récapitulant ce que le stage m'a appris et apporté.

I. Présentation de l'organisme d'accueil

Dans cette partie, nous allons voir une présentation générale du laboratoire avant d'évoquer sa dimension juridique suivie de sa structure organisationnelle, d'une présentation de l'équipe avec laquelle j'ai effectué le stage pour finir avec une étude de l'environnement économique du laboratoire.

I.A Historique

Le **laboratoire DAVID de l'Université de Versailles-Saint-Quentin-en-Yvelines** (Données et Algorithme pour une Ville Intelligente et Durable) se situe dans les bâtiments Descartes et Buffon du campus de l'Unité de Formation et de Recherche (UFR) des Sciences de la ville de Versailles (78000). Il a été fondé en juillet 2015 à l'initiative des membres du laboratoire PRISM pour créer un laboratoire spécialisé en algorithmique et en science de la donnée dont le principal objectif est de concevoir une ville écologiquement responsable, durable et dotée d'infrastructures autonomes et intelligentes.

I.B Présentation juridique de l'organisme d'accueil

Le laboratoire regroupe 60 chercheurs et doctorants spécialisés dans la recherche en informatique et plus particulièrement dans les domaines du big data, de la sécurité des données, de l'algorithmique, des réseaux de télécommunication et travail en interdisciplinarité avec la fédération **SIHS** (Sciences Informatiques, Humaines et Sociales) du CNRS dans l'objectif de concevoir une ville durable et intelligente.
La majorité des ressources financières du laboratoire proviennent de l'État et varient en fonction de la période ou du nombre de projets en cours et du coût de ceux-ci. Dans le cas du projet de recherche Polluscope (voir Présentation du projet), celui-ci à généré une aide financière de 172 721 € provenant de l'État et des financements 694 988 € provenant des partenaires du laboratoire dans le cadre du projet. La majorité des produits du laboratoire sont les publications scientifiques liées aux projets de recherches menés par les différentes équipes.

I.C Structure de l'organisme d'accueil

La direction du laboratoire est assurée par le professeur **BARTH Dominique**, **KEDAD Zoubida** et **DUCOIN Chantal** ayant respectivement les postes de directeur du laboratoire, directrice adjointe et responsable administrative. La gestion des équipes de recherches du laboratoire est répartie entre **BARTH Dominique** pour l'équipe **ALMOST**, **ANCIAUX Nicolas** pour l'équipe **PETRUS** et **ZEITOUNI Karine** pour l'équipe **ADAM**.
L'organigramme du laboratoire est disponible en annexe et provient du site internet de ce dernier (voir sitographie).

I.D Présentation de l'équipe de travail

Le laboratoire regroupe les équipes de recherches **ALMOST** (Algorithms and Stochastic Models), **PETRUS** (PErsonnal and TRUSted cloud), **NGN** (Next Generation Network) et **ADAM** (Ambient Data Access and Mining) avec laquelle j'ai réalisé le stage. Celles-ci sont majoritairement constituées de chercheurs ou d'étudiants doctorants et ont respectivement pour spécialités la résolution algorithmique et l'évaluation de performance, la structure, la gestion et la sécurisation de données personnelles, les réseaux et l'internet des objets et la modélisation de données hétérogènes.

Dans le cadre de mon stage, j'ai travaillé au sein de l'équipe ALMOST sous la direction de **ZEITOUNI Karine**, la cheffe du groupe et également ma tutrice de stage ainsi qu'une des professeurs rencontrées durant mes études à l'IUT informatique de Vélizy.

Elle fait partie des membres permanents de l'équipe au même titre **TAHER Yehia** (Maître de conférence), **YEH Laurent** (Maître de conférence). J'ai remarqué que parmi les membres permanents de l'équipe ADAM se trouvent **PREDA Nicoleta** et **LOYER Yann**, deux professeurs rencontrés durant mes études à l'IUT informatique de Vélizy.

Au sein de l'équipe, se trouvent également des étudiants doctorant ou post doctorant dont **BOUHAMOUM Redouane** un autre professeur de l'IUT de Vélizy. Par ailleurs, pendant la période de mon stage, deux membres du groupe de doctorants, **EL HAFYANI Hafsa** et **ZUO Jingwei**, ont soutenu leurs thèses devant un jury et obtenu le titre de Docteur en informatique. Quand le sujet de thèse est attribué à l'étudiant préparant un doctorat, il rejoint un des projets dont l'équipe est en charge. Celui-ci dépend du sujet de la thèse, de sa compatibilité avec celle-ci, et ce que l'étudiant peut apporter au projet avec ses travaux de recherche.

Aussi, dans le cadre de certains projets ou pour des exigences précises, le laboratoire recrute des étudiants devant réaliser des stages en entreprises dans le cadre de leurs études. J'ai fait partie de cette catégorie en rejoignant le laboratoire, au même titre que **BADRA Riham**, une étudiante ayant rejoint l'équipe dans le cadre de son Master en Informatique. Étant donné que le laboratoire est en relation avec des universités à l'échelle mondiale, cela lui permet d'accueillir des étudiants stagiaires et doctorants venus du monde entier.

I.E Étude de l'environnement économique de l'organisme d'accueil

Cette partie est entièrement dédiée à la présentation de l'environnement économique du laboratoire DAVID.

1) Analyse PESTEL de l'environnement du laboratoire DAVID

L'ensemble des opportunités et menaces pesant sur le laboratoire DAVID sont résumées dans la matrice PESTEL ci-dessous.
PESTEL : Politique, Économique, Socioculturel, Technologique, Environnemental, Légal

Dimension	Opportunité	Menace
Politique	- la loi de programmation de la recherche promet 3% du PIB consacré à la recherche et une meilleure organisation de celle-ci	- depuis mai 2021, le Plan National de Recherche sélectionne les proposition de projet de recherche en fonction de leurs impacts sociétaux, ce qui réduit le nombre projets financés
Économique		- pénurie mondiale de composants informatiques en raison de la crise du Covid-19 - les fortes dettes de l'État réduisent les possibilités de financement
Socio-culturel	- prise de conscience de l'importance des principes du développement durable - prise de conscience des enjeux environnementaux	
Technologique	- amélioration constante des technologies de l'information pour le public et les entreprises - la quantité de donnée augmente constamment et des travaux de recherche sont nécessaire pour optimiser leur traitement - de plus en plus d'appareils interagissent grâce à internet	- pénurie mondiale de composants informatiques en raison de la crise du Covid-19
Environnemental	- la dégradation de l'environnement nécessite des recherche pour en connaître les conséquences et les moyens de les diminuer	- on sait depuis peu de temps que le stockage massif de données est facteur d'augmentation de la pollution - la production d'ordinateur a des conséquences néfastes sur l'environnement
Légale		

Tableau 1 : Matrice PESTEL de l'environnement du laboratoire DAVID

L'actualité sur le plan technologique est globalement favorable au bon fonctionnement du laboratoire car l'utilisation massive des réseaux de télécommunication et des bases de données nécessite des travaux de recherche pour les rendre toujours plus performants. Néanmoins, d'un point de vue économique, étant donné les fortes dettes de l'État français, celui-ci possède moins d'argent pouvant être investi dans la recherche scientifique. De plus, les enjeux écologiques actuels nécessitent des travaux de recherche pour prévenir les conséquences de la pollution de l'air ou pour réduire l'impact des nouvelles technologies sur l'environnement.

2) Analyse SWOT du laboratoire DAVID

SWOT : Strengths, Weaknesses, Opportunities, Threats (Forces, Faiblesses, Opportunités, Menaces) L'ensemble des forces, faiblesses, opportunités et menaces du laboratoire DAVID sont résumés dans le tableau suivant et proviennent majoritairement du document d'évaluation HCERES de 2018 (voir Bibliographie).

Aspect	Description
Forces	- spécialisation du laboratoire dans le domaine de l'informatique - le laboratoire est en lien avec plusieurs centres de recherches de l'UVSQ et du territoire français - le laboratoire est en lien avec des universités à l'échelle international (Liban, Chine ...) et accueil des étudiants et doctorants venus du monde entier - les activités de recherche du laboratoire se basent sur plusieurs thèmes liés à des défis sociétaux majeurs - collaboration avec des acteurs territoriaux au niveau des communautés d'agglomération, du département des Yvelines et de la région Ile-de-France
Faiblesses	- effectif réduit par rapport à d'autres laboratoires de recherches - peu de moyens et absence de soutien politique de la part de l'UVSQ - difficulté pour les membres du laboratoire à investir dans des projets de recherche
Opportunités	- le laboratoire de recherche **NIWC** (Naval Information Warafare Center) des États-Unis est à la recherches d'étudiants, doctorants et chercheurs compétents dans les domaines de l'informatiques, essentiellement dans le big data - la loi de programmation de la recherche promet 3% du PIB consacré à la recherche et une meilleure organisation de celle-ci
Menaces	- depuis mai 2021 et l'apparition du Plan National de Recherche, le gouvernement sélectionne les proposition de projet de recherche en fonction de leurs impacts sociétaux, ce qui réduit le nombre projets financés par l'État - baisse de motivation des membres du laboratoire dans l'investissement pour l'obtention des moyens de recherche

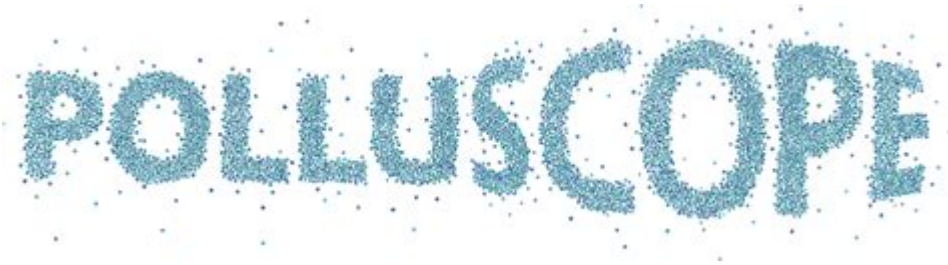
En France, les principaux organismes de recherche sont l'**INRA** (Institut National de la Recherche Agronomique), l'**INSERM** (Institut National de la Santé Et de la Recherche Médicale) et le **CNRS** (Centre National de la Recherche Scientifique) reconnu par son nombre de publications scientifiques comme le premier organisme de recherche au monde. Ceux-ci sont les principaux concurrents du laboratoire DAVID dans la mesure où ils exercent tous dans le domaine de la recherche et aussi les principaux associés car certains projets de recherche, tel que le projet Polluscope, se déroulent en collaboration avec les laboratoires évoqués précédemment.

II. Présentation du travail accompli durant le stage

Dans cette partie, nous allons évoqué le travail effectué durant le stage en commençant par une description du projet, suivi d'une description des tâches réalisées, d'une mise en évidence des principaux problèmes rencontrés, d'une présentation des outils utilisés, des méthodes et formations suivies pour finir par l'analyse des résultat de chaque tâches.

II.A Présentation du Projet

Le stage s'est déroulé dans le cadre du projet de recherche **Polluscope** (voir sitographie) dont l'équipe ADAM du laboratoire DAVID est en charge depuis le 1er septembre 2016 jusqu'à la date limite prévue, le 31 décembre 2022. Réalisé en collaboration avec AirParis, Cerema Ile-de-France, Versailles Grand Parc, Sorbonne Médecine, Inserm, LSCE, EIVP et l'Agence Nationale de la Recherche, ce projet a pour objectif de mesurer l'exposition à la pollution de la population française au quotidien.



Pour ce faire, des campagnes de recensement sont régulièrement réalisées en région parisienne et marseillaise avec la participation de volontaires. Pour les besoins de l'expérience, chaque participant reçoit un ou plusieurs capteurs de pollution et une tablette dans certains cas pour relever leurs activités ainsi que leur position géographique. Les capteurs sont conçus pour mesurer le taux de particules fines de différentes tailles dans l'air ambiant et transférer les données sur une application. Celles-ci peuvent ensuite être téléchargées et déposées sur le site internet du projet pour être analysées et mises sous différentes formes. Dans certaines conditions, des stations fixes sont utilisées pour mesurer la pollution de l'air ambiant. Toutefois, dans la mesure où celles-ci sont difficiles à déplacer, elles sont surtout utilisées comme témoins de la pollution dans des zones rurales ou peu urbanisées.

II.B Description du travail réalisé

Mon premier travail durant le stage est de créer le programme permettant la génération des rapports. La création des cartes et des graphes de visualisation de données ayant été réalisée en langage Python par **EL HAFYANI Hafsa**, je dois utiliser ce même langage pour créer des rapports de façon automatique. J'ai également pour contrainte de générer une première version des rapports en Markdown avant de les convertir en PDF.
Une autre partie de mon travail est de créer une nouvelle version de la plateforme de données Polluscope permettant le dépôt des données de pollution, le téléchargement de fichiers de données et le téléchargement des rapports. Une première version du site à été codée en PHP par **TAHER Yehia**. Cependant, pour répondre à un besoin d'homogénéisation des langages utilisés, notamment par rapport à la génération automatique des rapport, il faut que je créer une version du site internet en utilisant les langages Python et HTML ainsi que le framework Flask et la base de données Postgres du laboratoire sur laquelle sont stockées les données de pollution. Il faut également que je déploie cette nouvelle version de la plateforme sur le serveur du laboratoire en utilisant la technologie Docker.

II.C Description des problèmes à traiter et des solutions

Cette section du rapport est entièrement consacrée à la description détaillée des problèmes rencontrés durant la refactorisation de la plateforme de données Polluscope qui constitue mon travail le plus important durant le stage. Cette partie présente également les solutions découvertes ou mises au point pour résoudre ces problèmes.

Dans cette partie, les références aux portions code présents en annexes sont entre parenthèses.

Étape 1 : Comprendre la structure de la plateforme

Étant donné qu'une première version de la plateforme à été créée en PHP par **TAHER Yehia** et **Ahmad Ktaish**, il fallait que je respecte la structure de l'application mais aussi sa charte graphique. Pour visualiser la structure du site internet et mieux comprendre son fonctionnement, j'ai créé un schéma comportant des balises représentant les différentes pages de la plateforme ainsi que les liens entre celles-ci. Ici le terme de structure est utilisé pour évoquer le comportement de la plateforme en fonction des actions de l'utilisateur et des pages sur lesquelles elles sont réalisées.

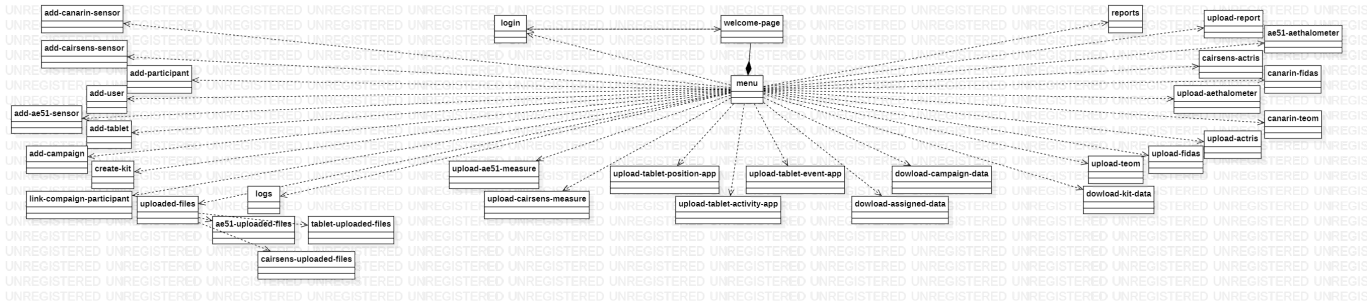


Schéma 1 : Structure de la plateforme de données Polluscope

Ce schéma est également disponible en annexe.

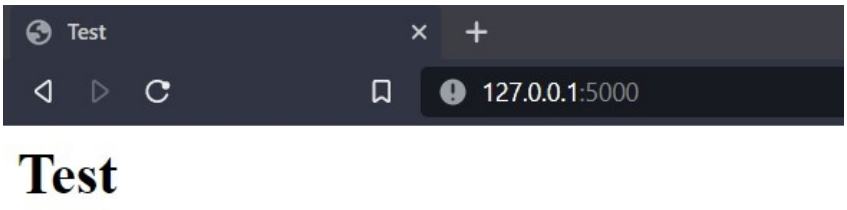
En examinant le code source de la plateforme, on constate qu'il est nécessaire d'avoir un compte enregistré dans la base de données pour utiliser les services du site internet. On peut également remarquer qu'un utilisateur peut avoir soit un rôle de participant, soit un rôle d'administrateur et que cela influe sur les actions réalisables par la personne qui se connecte à la plateforme et donc sur les pages qui lui sont accessibles. Dans le schéma précédent, les pages à gauche sont accessibles pour les participants, celles en bas au centre sont utilisables uniquement par les administrateurs et les autres sont accessibles pour tous les utilisateurs. Sur cette plateforme, seule la page `login` est accessible pour le grand public.

Étape 2 : Comprendre le fonctionnement basique de Python Flask

Afin d'apprendre à créer un site internet en Python Flask j'ai recherché sur internet des contenus expliquant la structure et le fonctionnement de ce dernier.
La liste des liens dont je me suis servi pour cette auto formation sont présents dans la sitographie.

D'après mes recherches, la technologie Flask a un fonctionnement semblable au Design Pattern **Modele-View-Controller** (Modèle-Vue-Contrôleur) utilisé dans les langages de programmation orientés objets. En effet, l'affichage des pages codées en HTML ainsi que les actions de l'utilisateur sur celles-ci sont ou peuvent être entièrement ou partiellement gérées par un seul fichier codé en Python.

Les deux portions de code *2.1.1* et *2.1.2* présentes en annexe, permettent l'affichage d'une pages web identique à celle de l'image suivante :



Page Web 1 : Page web basique avec Python Flask

Pour cela, le fichier `index.html` (Code *2.1.1*) doit être placé dans un répertoire **templates** avant d'exécuter le fichier `app.py` (Code *2.1.2*). Cela permet d'ouvrir une fenêtre de terminal comportant les lignes suivantes :

```
* Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

Pour afficher la page web il faut entrer l'URL fourni (ici `http://127.0.0.1:5000`) dans un moteur de recherche.

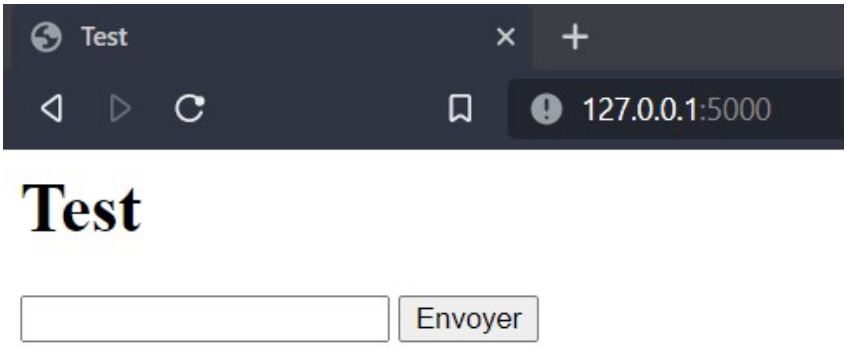
En sachant cela, j'ai pu créer la page d'accueil de la plateforme du projet.

Page Web 2 : Page d'accueil de la plateforme de données Polluscope

Étape 3 : L'implémentation de la méthode POST avec Python Flask

D'après l'analyse de la plateforme, celle-ci comporte de nombreux formulaires nécessaires aux dépôt des données de pollution. Tout comme dans la première version du site internet, celui que j'ai créé doit permettre la transmission d'éléments insérés dans des formulaires au moyen de la méthode POST.
En reprenant le code HTML vu précédemment on peut rajouter un formulaire (Code *2.2.1* #1) constitué d'une zone de texte (Code *2.2.1* #2) et d'un bouton de validation (Code *2.2.1* #3). La méthode `main()` du fichier `app.py` (Code *2.2.2*) qui permet l'affichage de la page web est par défaut à l'écoute des actions réalisées sur ce formulaire.

Avec le nouveau fichier `index.html` (Code *2.2.1*), on obtient une page web identique à celle-ci :



Page Web 3 : Page web avec un formulaire pour afficher du texte dans une fenêtre de terminal

Afin que le programme en Python puisse recevoir des éléments avec la méthode POST, celle-ci doit être spécifiée grâce à la méthode `@app.route()` (Code *2.2.2 #1*). Pour récupérer les éléments insérés dans le formulaire, il faut utiliser l'attribut `form` de l'objet `request` en spécifiant le nom de l'élément (Code *2.2.2 #2*) après avoir vérifié si la méthode POST est utilisée (Code *2.2.2 #3*).

Avec les deux dernières portions de code présentées précédemment, on peut entrer du texte dans la zone dédiée et l'envoyer pour que celui-ci soit affiché dans un terminal.

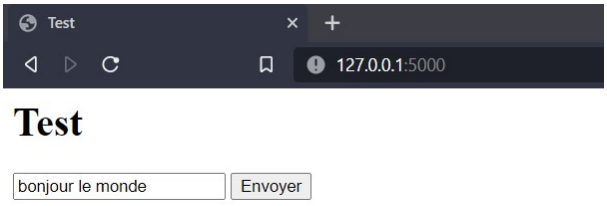
Comprendre le fonctionnement de la méthode POST m'a permis de d'implémenter une première version de la page de connexion sans utilisation de la base de données.

Étape 4 : Modifier la page web en fonction des actions de l'utilisateur

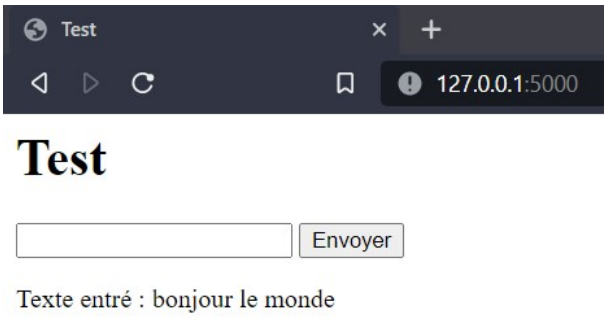
Durant l'analyse de la première version du site internet, j'avais constaté que le programme en PHP permet l'affichage de texte sur les pages web pour indiquer si des erreurs ont eu lieu ou si une action s'est achevée correctement. Afin que cela soit possible dans la nouvelle version de la plateforme je devais trouver un moyen pour passer des éléments en paramètre des fonctions d'affichage des pages web.
Pour ce faire, j'ai utilisé des éléments de syntaxe **Jinja** dans les pages web. Cela permet de créer des boucles ou des action conditionnelles dans des pages codées en html.
Dans le code *2.3.1*, si la fonction d'affichage de la page comporte un paramètre `posts` portant le label 'text' (Code *2.3.1 #1*) alors celui-ci est affiché sur la page web (Code *2.3.1 #2*).

Pour pouvoir passer un paramètre à la fonction d'affichage, un dictionnaire doit être instancié dans le script python (Code *2.3.2 #1*) et les éléments à passer en paramètre doivent être associés à la bonne clé (Code *2.3.2 #2*). Par défaut, l'élément associé à la clé doit être nul (Code *2.3.2 #3*) pour éviter des erreurs. Le dictionnaire doit ensuite être placé en paramètre de la méthode `render_template()` avec le nom approprié (Code *2.3.2 #4*).

Le code ainsi modifié permet d'afficher une page web similaire à la précédente mais à présent lorsque du texte est entré dans la zone appropriée et envoyé (#1), celui-ci est affiché dans le navigateur (#2).



#1 **Page Web 4 :** Page web avec un formulaire permettant d'entrer du texte



#2 **Page Web 5 :** Page web affichant le texte entré dans le formulaire

L'affichage conditionnel sur des pages web m'a permis de modifier les pages accessibles aux utilisateurs en fonction de leurs rôle de participant ou d'administrateur ou d'afficher des messages d'erreur ou de réussite de certaines actions.

Page Web 6 : Page d'accueil de la plateforme Polluscope après une erreur de connexion

Étape 5 : Permettre le changement de page par l'utilisation de liens

La plateforme doit comporter un menu permettant l'accès à différentes pages en fonction des besoins de l'utilisateur et de ses actions à réaliser. Dans la version que j'ai créé, j'ai dû trouver une méthode permettant l'utilisation des liens spécifiés pour chaque fonctions rattachées à chaque page.
Pour ce faire, il faut utiliser un élément de syntaxe **Jinja** associé à la méthode `url_for()` (Code *2.4.1* #1). Celle-ci doit avoir l'URL de la page souhaitée en paramètre.

Pour qu'une autre page HTML (ici celle du Code *2.4.2*) soit accessible par un lien, celle-ci doit être rattachée à une fonction (Code *2.4.3* #1) et à un chemin d'accès (Code *2.4.3* #2).

Les trois fichier permettent d'afficher une page comportant un titre et un lien.

Page Web 7 : Première page web avec un lien vers la deuxième page

Cliquer sur le lien permet d'afficher la deuxième page.

Ceci est la deuxième page

Page Web 8 : Deuxième page web accessible via la première

Avec ces éléments, j'ai pu créer le menu contenant les liens vers les différentes pages.

Page Web 9 : Exemple de menu de la plateforme Polluscope

Étape 6 : Permettre la connexion et les interactions avec la base de données PostgreSQL

Tout comme la première version de la plateforme, celle que j'ai créé doit pouvoir insérer, lire et modifier des éléments contenus dans une base de données PostgreSQL.
Pour établir la connexion avec la base de données, il est possible d'utiliser la commande python suivante avec le nom de la base de données, le nom d'utilisateur, le mot de passe de l'utilisateur et l'adresse IP du serveur hébergeant le système de gestion de base de données.

```
import psycopg2

conn = psycopg2.connect(database="nom_base_de_données", user="nom_utilisateur",
password="mot_de_passe", host="adresse_ip")
```

Code 2.5.1 : Commande de connection à la base de données PostrgreSQL

Pour exécuter une commande sur la base de données, il faut écrire cette dernière dans un objet de type String (#1) et le passer en paramètre de la méthode `execute()` du curseur (#2), attribut de la connection précédemment établie (#3).

```
import psycopg2

conn = psycopg2.connect(database="nom_base_de_données", user="nom_utilisateur",
password="mot_de_passe", host="adresse_ip")

query = "select * from test" #1
cur  = conn.cursor() #3
cur.execute(query) #2
```

Code 2.5.2 : Script Python permettant l'exécution d'une commande SQL sur une base de données PostgreSQL

Les résultats des requêtes SQL sont stockés dans la variable curseur sous forme de tableau. Pour visualiser les ligne générées par la requête, il faut donc afficher chaque élément de la variable curseur (#1) grâce à une boucle (#2).

```
import psycopg2

conn = psycopg2.connect(database="nom_base_de_données", user="nom_utilisateur",
password="mot_de_passe", host="adresse_ip", port= "port")

query = "select * from test"
cur  = conn.cursor()
cur.execute(query)

for row in cur: #2
    print(row) #1
```

Code 2.5.3 : Script Python permettant l'exécution d'une requête SQL et l'affichage des lignes générées

```
Output :
['test1']
['test2']
['test3']
```

Pouvoir interagir avec la base de données m'a permis de créer une nouvelle version de la page de connexion en utilisant les logins et mots de passe stockés dans la base de données. J'ai également pu créer des scripts Python pour enregistrer les données de pollution mais aussi ajouter des utilisateurs, des capteurs, des kits ou des campagnes.

Étape 7 : Sécuriser les interactions avec la base de données

D'après mon analyse du code source de la première version de la plateforme Polluscope, les requêtes utilisées pour insérer des éléments dans la base de données étaient exécutées sans être préparées. En tant que passionné de cybersécurité, je sais qu'une requête SQL non préparée est une faille de sécurité pouvant être utilisée à des fins malveillantes pour compromettre une base de données grâce à des **injections SQL**. Il était donc primordiale que je corrige cette faille dans la nouvelle version du site internet.

Normalement, quand le client d'une page web insère un élément dans un champ prévu à cet effet tel qu'une zone de texte, celui-ci est récupéré et inséré dans la variable de type String qui constitue la commande SQL. Cette commande est ensuite exécutée sur la base de données.
Dans les exemples suivants nous utiliserons la table `personnes` présentée ci-dessous :

```
 id |   nom
----+---------
  1 | Camille
  2 | Bruce
  3 | Selina
```

Les deux portions de code *2.6.1.1* et *2.6.1.2* présents en annexes, permettent l'affichage d'une page web avec un formulaire composé d'un champ texte. L'utilisateur doit entrer l'ID d'une personne dans le champ texte pour que le nom de cette personne s'affiche sur la page web ().

Le script Python établit la connexion avec la base de donnée (Code *2.6.1.2 #1*) si il y a un élément transmis par la méthode POST (Code *2.6.1.2 #2*). Une commande est ensuite créée avec l'id de la personne dont l'utilisateur veut afficher le nom (Code *2.6.1.2 #3*). Une fois la commande exécutée (Code *2.6.1.2 #4*), les résultats de la commande sont récupérés dans la variable `out` (Code *2.6.1.2 #5*) et envoyés vers la page web par la méthode POST (Code *2.6.1.2 #6*).

Si l'utilisateur entre l'id `1` alors le nom de `Camille` est affiché.

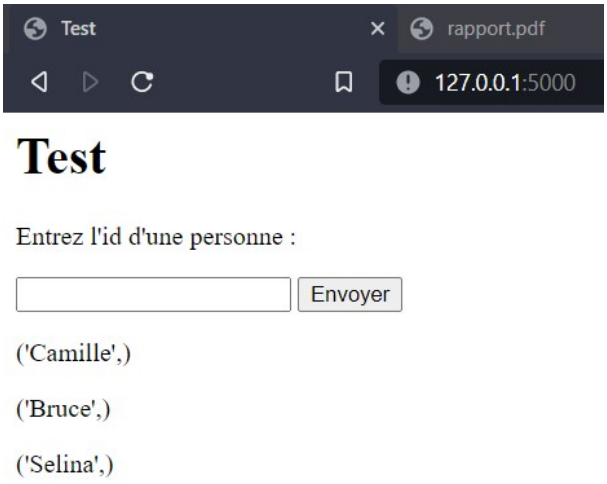
Page Web 10 : L'identifiant 1 est entré dans le formulaire

Page Web 11 : Le nom de Cammile est affiché sur la page

Cependant, il est possible d'afficher tous les noms des personnes en utilisant une injection SQL. Pour cela, il suffit de rajouter une condition après l'identifiant. Pour fonctionner, cette condition doit être vraie pour toutes les lignes de la table. Il peut aussi être nécessaire de rajouter des tirêts pour que la suite de la commande devienne un commentaire et ne soit pas pris en compte lors de l'exécution. Par exemple, le texte suivant est une injection SQL.

```
1 or 1=1; -- -
```

Si ce texte est inséré et envoyé, on obtient la page suivante.



Page Web 12 : Tout les noms sont affichés grâce à l'injection SQL

Dans l'exemple précédent, l'injection SQL permet l'affichage de plusieurs éléments alors que le programme utilisé normalement n'en affiche qu'un seul. Cependant, entrer le texte suivant aura pour conséquence, la suppression de la table `personnes`.

```
1; delete from personnes where 1=1 -- -
```

Dans le cadre de travaux de recherche, la suppression des données peut avoir de graves conséquences sur le projet.
Pour éviter cela, j'ai dû rajouter un traitement des éléments transmis par l'utilisateur pour que les requêtes ne contiennent pas de caractères permettant les injections SQL.

La plupart des langages utilisés dans la programmation web possèdent une fonction destinée à effacer les injections SQL des éléments insérés par le client d'un site web. Etant donné que je n'ai pas trouvé ce type de fonction dans le langage python, j'ai dû trouver un autre moyen.
Dans l'exemple précédent l'injection SQL est permise par l'élément de comparaison `or` envoyé par le client de la page web. Utiliser la fonction `replace()` de la classe String avec les paramètre `"or"` et `""` permet de le supprimer de l'élément inséré par le client (Code *2.6.2.1* #1).

Avec la modification du script Python, entrer l'injection SQL génère une erreur mais n'endommage pas la base de données.
Concernant la plateforme Polluscope, le nombre de caractères à supprimer est plus important pour écarter la plupart des possibilités d'injection SQL.

Étape 8 : Vérifier si les données insérée par le client sont déjà dans la base de données

Dans la mesure où les travaux de recherche nécessitent des mesures précises pour être fiables, j'ai considéré que l'insertion de doublons de données de pollution dans la base de données mettrait en péril les résultats des campagnes de recensement de la pollution. Je suis donc parti du principe que les participants des campagnes pouvaient faire l'erreur d'insérer plusieurs fois un même fichier de données et que si cela se produit alors les scripts Python doivent le savoir et ne pas insérer les doublons dans la base de données. C'est un constat qui a été approuvé par mes encadrants.

J'ai considéré que le nom du fichier inséré et contenant les données de pollution n'est pas un élément de décision fiable car celui-ci peut être facilement modifié par le participant.
Étant donnée que les fichier inséré par les clients de la plateforme sont enregistrés sur le serveur du laboratoire, dans une première version du programme, pour tester l'existence d'une ligne dans la base de donnée, le script Python testait si elle se trouvait parmis les lignes des fichiers précédemment insérés et stockés sur le serveur. Le pseudo code Python suivant illustre

cette première méthode de vérification. La fonction prend en paramètre la ligne à insérer et une liste contenant toutes les lignes enregistrées (#1). Si la nouvelle ligne est présente dans la liste alors la fonction renvoie la valeur booléenne `False` (#2) et la valeur `True` dans le cas contraire (#3).

```
def testNew(nouvelleLigne, toutesLesLignes): #1
    if nouvelleLigne in toutesLesLignes:
        return False #2
    return True #3
```

Code 2.7.1.1 : fonction `testNew()`

Avec la fonction précédente, il est possible de placer l'instruction d'insertion dans une boucle conditionnelle (#1). L'insertion ne sera effectuée que si la nouvelle ligne n'est pas enregistrée.

```
if testNew(nouvelleLigne, toutesLesLignes) : #1
    inserer(nouvelleLigne)
```

Code 2.7.1.2 : boucle d'insertion conditionnelle

Néanmoins, comme l'ont remarqué mes encadrants, cette méthode est très coûteuse en mémoire pour l'ordinateur réalisant l'opération et l'utilisation de cette ressource augmente avec le nombre de ligne à comparer. Étant donné que, dans le cadre du projet Polluscope, ce nombre peut s'élever à plusieurs millions, cette méthode n'est pas applicable pour ce projet. J'ai donc dû trouver une autre méthode.

Mes encadrant m'ont alors suggéré d'utiliser les dates d'utilisation du capteur comme moyen de vérification. En effet, l'ensemble des capteurs, en plus de relever des données de pollution, mémorisent les dates auxquelles les données sont enregistrées. En ayant connaissance de l'identifiant du capteur, de l'intervalle pendant lequel il a été utilisé et en sachant qu'un capteur ne peut être utilisé que par un seul participant à la fois, il est possible de savoir si une ligne à déjà été insérée.
Pour ce faire, le script Python récupère grâce à une requête SQL l'intervalle d'utilisation d'un capteur en fonction de son identifiant.

```
select min(time), max(time) from sensor_measure where id='sensorId';
```

Code 2.7.2.1 : Requête permettant de connaître l'intervalle d'utilisation d'un capteur

On peut alors modifier la fonction `testNew()` pour que celle-ci vérifie si la date de la ligne à insérer est comprise dans cet intervalle (#1). Si tel est le cas alors la fonction renvoie la valeur `False` (#2) et la valeur `True` dans le cas contraire (#3).

```
def testNew(dateNouvelleLigne, intervalleMin, intervalleMax):
    if intervalleMin < dateNouvelleLigne and dateNouvelleLigne < intervalleMax :
#1
        return False #2
    return True #3
```

Code 2.7.2.1 : nouvelle version de la méthode `testNew()`

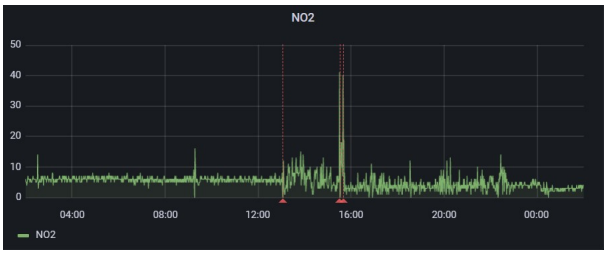
Il s'agit d'une méthode de vérification nécessitant moins de ressources de l'ordinateur, ce qui permet d'accélérer l'insertion de nouvelles lignes dans la base de données.

Étape 9 : Créer et ajouter des graphes de visualisation des données

Peu avant la fin du stage, il m'a été demandé de rajouter des graphes et des cartes, créés avec le logiciel Grafana, à la plateforme Polluscope pour permettre la visualisation de données. Le programme de création des graphes avait été créé par **ABBOUD Mohammad** et des étudiant de l'université. Mon travail dans cette étape a donc été d'intégrer ce programme dans le code source de la plateforme Polluscope en faisant les modifications nécessaires.

Dans cette partie du travail la principale difficulté a été d'apporter des modifications au code déjà créé et d'en comprendre les erreurs. En effet, pour interagir avec le logiciel Grafana, le programme utilise le package `requests` pour envoyer des requêtes vers l'ordinateur sur lequel ce premier est installé et en obtenir les réponses. Il s'agit d'une librairie que je n'avais alors jamais utilisé et dont les erreurs peuvent être incomprises et mener la personne en charge du développement vers de fausses réflexions.

Aussi, les données utilisées pour générer les graphes proviennent de requêtes SQL transmises au logiciel par l'intermédiaire d'un ou plusieurs fichiers JSON. Or ces fichier ont été écrits avec des requêtes permettant l'extraction d'informations d'une base de données selon un modèle créé au début du projet. Celui-ci a ensuite été mis à jour, ce qui rend les fichiers JSON inutilisables. Il a donc été nécessaire que je modifie les requêtes SQL de ces fichier pour rendre possible les interations avec la base de données et donc l'affichage de graphes.



Page Web 13 : Exemple de graphe généré sur Grafana

A la demande de ZEITOUNI Karine, la génération des graphes m'a permis de réaliser un contrôle de la qualité des données recueillies par les capteurs PMScan / Diams lors de la campagne de recesement de pollution réalisée durant le mois de mai. Cela a pour objectif de mettre en évidence des disfonctionnements sur certains capteurs. Ceux-ci sont censés fonctionner et recueillir des données de pollution pendant sept jours. Ainsi, des absences de données peuvent être décelées grâces aux graphes et imposé une réparation ou un remplacement de matériel.

Le rapport sur la qualité des données des capteurs PMScan est disponible dans le répertoire `quality_check` sur le dépôt Github du présent rapport (voir sitographie).

II.D Les outils informatiques et techniques utilisés

Dans le cadre de mon travail durant ce stage, l'ordinateur personnel que je possède et utilise au quotidien pour mes études et mes activités dans le domaine de l'informatique est devenu mon principal outils de travail. Étant déjà familiarisé avec les environnements de développement créer par **JetBrains** tel que PyCharm mon choix s'est porté sur ceux-ci pour produire les logiciels et applications durant le stage.

Pendant toute la durée du stage, j'ai utilisé Github pour la gestion et le partage de versions des applications avec les membres de l'équipe car ceux-ci l'avait déjà utilisé auparavant. L'historique des implémentation est disponible ici en annexes.

Comme évoqué dans la partie précédente, l'ensemble des données recueillies lors des campagnes de recensement sont stockées sur une base de données PostgreSQL appartenant au laboratoire. Afin de ne pas endommager, effacer ou modifier ces données lors du développement de la nouvelle version de la plateforme Polluscope, j'ai installé ce même système de gestion de base de données sur mon ordinateur personnel. Celle-ci a été utilisée pendant toute la durée de la création du nouveau site web du projet.

Tel que spécifié dans le sujet du stage, la plateforme Polluscope doit fonctionner avec le langage Python. Par conséquent, il a été nécessaire d'utiliser le framework Flask. Celui-ci permet le déploiement d'une application web sur un ordinateur en ayant seulement recours au langage Python et ne nécessite pas l'utilisation de logiciels d'hébergement de site internet tel que Apache pour fonctionner. De plus pour rendre les données recueillies visibles et compréhensibes, il m'était demandé que j'utilise l'outil de visualisation de données Grafana. Cela a permis la création de graphes ou de cartes avec les données de pollution relevées ainsi que leurs coordonnées.
Aussi, pour faciliter le déploiement de l'application sur le serveur du laboratoire, l'équipe a exigé que l'acquisition des packages nécessaires et la mises en marche de la plateforme se fasse par l'utilisation de la technologie Docker. Cela permet de faire fonctionner le site internet sans avoir à installer les paquets de façon définitive sur le serveur et donc d'économiser de l'espace sur le disque dur de celui-ci.


II.E Méthodes, formations et autoformations suivies pour l'obtention de la solution

Comme évoquer auparavant ma première tâche à été de créer un programme générant automatiquement des rapports en Markdown avant de les convertir au format PDF. Étant déjà familiarisée avec ce premier je n'ai pas eu besoin d'apprendre la structure d'un fichier Markdown. En revanche, il a été nécessaire que je trouve des fonctions en Python permettant l'exportation du fichier au format souhaité.

Dans le cadre de la refactorisation de la plateforme Polluscope, je n'avais jusqu'alors jamais créé de site internet avec le framework Flask. Il a donc été nécessaire que j'apprenne et comprenne son fonctionnement avant de commencer à produire le code de la plateforme. Sachant cela et étant donnée que mes seules contraintes étaient l'utilisation d'un langage spécifique et l'achèvement de ce travail avant la fin du stage, j'ai utilisé une méthode de travail semblable à la méthode agile. Cela signifie les membres de l'équipe encadrant mon travail me demandaient régulièrement des retours sur l'avancée de mon travail en me suggérant des modifications ou des améliorations. Par exemple, cette méthode m'a permis de recevoir les retours de mes encadrants concernant une méthode de vérification d'insertion de données que j'avais créé et qui nécessitait des modification (voir Étape 8 du développement de la plateforme).
J'ai donc débuté mon travail de développement en contruisant des pages simples et en ajoutant des fonctionnalités au fur et à mesure de mon apprentissage. Toutefois, ayant déjà travaillé avec le langage Python, créer des applications web et étant désireux d'apprendre à utiliser de nouvelles technologies et préparé à cette éventualité, cette tâche s'est avéré être passionnante.

Une autre partie de mon travail a été de déployer la plateforme de données Polluscope sur le serveur du laboratoire en utilisant la technologie **Docker**. Pour comprendre son fonctionnement, j'ai eu recours aux lien présents en sitographie et j'ai été aidé par TAHER Yehia. J'ai également eu recours à ma machine virtuelle et à ma clé bootable, toutes deux sous Linux, pour tester le fonctionnement des conteneurs ainsi créés.

On peut constater que la plupart des tâches qui m'ont été confiée nécessite l'utilisation de technologies nouvelles ou peu utilisées. Toutefois, ma formation à l'IUT étant axée sur la pratique de la programmation par le développement et la recherche de documentation, l'utilisation de nouveaux langages ou de technologies de programmation spécifiques ne m'a pas posé de difficultés.

II.F Résultats

Concernant le programme de génération automatique de rapport, celui-ci a pu être terminé au bout d'une semaine de travail ce qui m'a permis d'entamer très vite la refactorisation de la plateforme Polluscope.
Cela a constitué la partie la plus importante de mon travail et également la plus longue. La refactorisation de la plateforme ainsi que l'ajout des pages permettant le chargement des données des nouveaux capteurs a commencé le 11 avril, au début de la deuxième semaine de stage, et s'est déroulé jusqu'à la fin du stage. La création des fichiers permettant le fonctionnement d'une image Docker hébergeant la plateforme a nécessité deux jours de travail durant la semaine de l'Ascension. Néanmoins, ayant peu de connaissances concerant la mise en place de cette technologie, c'est une tâche qui a été remprise par TAHER Yehia. Il m'était également demandé d'intégrer la génération de rapport dans la plateforme de données. Néanmoins, en raison du départ de EL HAFYANI Hafsa et de la complexité de son code intégré dans les scripts Python que j'ai créé, c'est une tâche que je n'ai pu accomplir.
Pour des raisons de santé et d'obligations liées à la poursuite d'étude j'ai été absent pendant cinq jours durant le stage. Mon stage a donc été prolongé d'une semaine en premier lieu, puis de quinze jour. Cela m'a permis d'apporter de nouvelles fonctionnalités à la plateforme et également de corriger des disfonctionnements. La plateforme Polluscope sera sûrement amenée à évoluer pendant la suite du projet de recherche et je reste par conséquent à disposition des membres de l'équipe ADAM pour leur apporter mon aide dans l'amélioration de la plateforme Polluscope

Conclusion

Mon choix de stage en entreprise s'est porté sur l'offre proposée par ZEITOUNI Karine car l'utilisation de technologies ou de langages peu connus dans le domaine de l'informatique faisait partie intégrante du sujet du stage et du travail à réaliser durant celui-ci. J'y ai vu une opportunité pour continuer d'acquérir des compétences en informatique tout en participant à un projet de recherche. Il s'agit d'une opportunité dont je n'aurais peu être pas bénéficié dans une autre entreprise.
L'apprentissage de nouvelles technologies a rajouté une difficulé au travail à réaliser car il fallu que je me forme a leurs utilisations. Toutefois, étant avide d'acquérir de nouvelles connaissances et compétences, cette tâche ne m'a pas posé de grandes difficultés et s'est avérée être passionnante.
Le principal objectif du stage était de refactoriser la plateforme de données Polluscope en utilisant la technologie Flask et le langage Python. Celui-ci a été accompli au même titre que le programme permettant la génération automatique des rapport sur l'expostion à la pollution.

Annexes

Organigramme 1 : Organigramme du laboratoire DAVID

Code 2.1.1 : fichier `templates/index.html`

```
<html>
    <head>
        <title>Test</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Test</h1>
    </body>
</html>
```

Code 2.1.2 : fichier `app.py`

```
from flask import *

app = Flask(__name__)

@app.route("/")
def main():
    return render_template("index.html")

if __name__ == "__main__" :
    app.run()
```

Code 2.2.1 : fichier `templates/index.html` avec un formulaire

```
<html>
    <head>
        <title>Test</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Test</h1>
        <form action="" method="post"> #1 un formulaire
            <input type="text" name="text"> #2 avec une zone de texte
            <input type="submit" name="submit"> #3 et un bouton de validation
        </form>
    </body>
</html>
```

Code 2.2.2 : fichier `app.py` permettant de recevoir des éléments avec la méthode POST

```
from flask import *

app = Flask(__name__)

@app.route("/", methods=("GET", "POST")) #1 La fonction peut utiliser les méthodes
GET et POST
def main():
    if request.method == "POST": #3 Si la méthode post est utilisée
        text = request.form["text"] #2 le texte inséré dans le formulaire est
récupéré
        print(text)
    return render_template("index.html")

if __name__ == "__main__" :
    app.run()
```

Code 2.3.1 : fichier `templates/index.html` avec un affichage conditionnel

```
<html>
    <head>
        <title>Test</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Test</h1>
        <form action="" method="post">
            <input type="text" name="text">
            <input type="submit" name="submit">
        </form>

        {% block content %}

        {% if posts['text'] %} #1 Si il y a un paramètre post avec le label 'text'
        <p>Texte entré : {{posts['text']}}</p> #2 alors celui-ci est affiché sur
la page
        {% endif %}

        {% endblock %}
    </body>
</html>
```

Code 2.3.2 : fichier `app.py` avec passage de paramètre par le dictionnaire `post`

```
from flask import *

app = Flask(__name__)

@app.route("/", methods=("GET", "POST"))
def main():
    post = {} #1 Un dictionnaire est instancié
    post['text'] = None #3 Par défaut l'élément associé à la clé est nul
    if request.method == "POST":
        text = request.form["text"]
        post["text"] = text #2 Les éléments a passer en paramètre doivent être
associés à la bonne clé
    return render_template("index.html", posts = post) #4 dictionnaire doit
ensuite être placé en paramètre de la fonction d'affichage

if __name__ == "__main__" :
    app.run()
```

Code 2.4.1 : fichier `index.html` avec un lien vers la page comportant l'URL `page2`

```
<html>
    <head>
        <title>Test</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Ceci est la première page</h1>
        <li>
            <a href="{{ url_for('page2') }}"> #1 élément de synthaxe Jinja associé
à la méthode url_for()
                lien vers la deuxième
            </a>
        </li>
    </body>
</html>
```

Code 2.4.2 : fichier `page2.html` accessible par un lien

```
<html>
    <head>
        <title>Test</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Ceci est la deuxième page</h1>
    </body>
</html>
```

Code 2.4.3 : fichier `app.py` avec la fonction `page2` permettant l'affichage de la page du fichier `page2.html`

```
from flask import *

app = Flask(__name__)

@app.route("/")
def main():
    return render_template("index.html")

@app.route("/page2") #2 le chemin d'accès vers la deuxième page
def page2(): #1 la fonction permettant l'affichage de la deuxième page
    return render_template("page2.html")

if __name__ == "__main__" :
    app.run()
```

Code 2.6.1.1 : fichier `index.html`

```
<html>
    <head>
        <title>Test</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Test</h1>
        <form action="" method="post">
            <p>Entrez l'id d'une personne : </p>
            <input type="text" name="text">
            <input type="submit" name="submit">
        </form>

        {% block content %}

        {% for n in posts['name'] %}
        <p>{{n}}</p>
        {% endfor %}

        {% endblock %}
    </body>
</html>
```

Code 2.6.1.2 : fichier `app.py`

```
import psycopg2
from flask import *

app = Flask(__name__)

@app.route("/", methods=("GET", "POST"))
def main():
    post = {}
    if request.method == "POST": #2 si la méthode POST est utilisée
        conn = psycopg2.connect(database="X", user="X", password="X", host="X") #1
une connexion est établie avec la base de données
        input = request.form["text"]
        query = "select nom from personnes where id="+input #3 la requête
selectionne le nom de la personne en fonction de son identifiant
        cur = conn.cursor()
        cur.execute(query) #4 la requête est exécutée ...
        out = []
        for row in cur:
            out.append(row) #5 ... et son résultat est stocké dans la variable out
...
        post['name'] = out
    return render_template("index.html", posts=post) #6 ... et envoyé vers la page
web
```

```
if __name__ == "__main__" :
    app.run()
```

Code 2.6.2.1 : fichier `app.py` avec les éléments pour empêcher une injection SQL

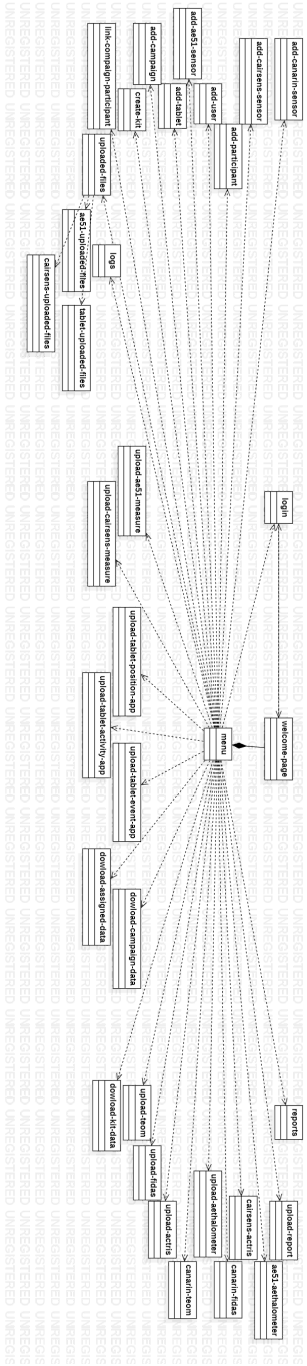
```
import psycopg2
from flask import *

app = Flask(__name__)

@app.route("/", methods=("GET", "POST"))
def main():
    post = {}
    if request.method == "POST":
        conn = psycopg2.connect(database="X", user="X", password="X", host="X")
        input = request.form["text"]
        query = "select nom from personnes where id="+str(input).replace("or", "")
#1 la fonction replace permet de supprimer la chaîne de caractère "or" de
l'élément envoyé par l'utilisateur
        print(query)
        cur = conn.cursor()
        cur.execute(query)
        out = []
        for row in cur:
            out.append(row)
        post['name'] = out
    return render_template("index.html", posts=post)

if __name__ == "__main__" :
    app.run()
```

Schéma 1 : Structure de la plateforme de données Polluscope



Fichier 1 : Fichier README du dépôt github contenant le code source de la plateforme Polluscope

````
# Polluscope

The code in the main branch can be used on any device with python3 and all
packages installed.

**Required Packages :**
- flask
- flask_session
- pathlib
- pandas
- werkzeug.utils
- datetime
- pyscopg2
- configparser
- gunicorn

It's important to modifie lines in the file ```database.ini``` to make possible a
connection to your own PostgreSQL database. SQL queries in the file
```creation.sql``` will create all necessary tables in your data base. Use queries
in ```delete.sql``` to erase all tables and in ```insert_basic.sql``` to create an
admin user for the platform.

Before launch the application for the first time, be sure that all data
repositories in ```data``` folder are empty. Before launch the app check the
```data``` repository contains all repositories in the following list and create
them else :
- actris
- ae51
- aethalometer
- cairsens
- canarin
- diams
- flow
- report
- tablet_activity
- tablet_event
- tablet_gps
- teom

To launch the web application on the computer in development mod you have to run
the python file ```app.py```.

To launch the web application on a server for deployement, you have to use the
command ```gunicorn --bind 0.0.0.0:5000 app:app```
````

```
**Unfonctional pages :**
- canarinTeom
- canarinFidas
- cairsensActris
- ae51Aethalometer
```

Journal 1 : Historique des implémentations de la plateforme de données Polluscope

commit c843939f0204418f3522f618a776b8a3aa3f8cad (HEAD -> main, origin/main, origin/HEAD) Author: yureiiko pierrefcz@gmail.com Date: Fri Jun 3 15:41:24 2022 +0200

```
UPDATES :
Dashboard creation in progress.
```

commit 6dfcaf7219951344c5d7f77bbe8b77d06cda9860 Author: yureiiko pierrefcz@gmail.com Date: Fri Jun 3 12:05:04 2022 +0200

```
UPDATES :
Dashboard creation in progress.
```

commit d6074c14d7b8c46d476e73ab66b4686c51a7c4a9 Author: yureiiko pierrefcz@gmail.com Date: Thu Jun 2 17:12:56 2022 +0200

```
UPDATES :
Add the package to create DashBoards with Grafana.
```

commit aa1dace405619332f076107ac77568104f89aea1 Merge: 1e8dc75 fd306d8 Author: yureiiko pierrefcz@gmail.com Date: Thu Jun 2 10:41:30 2022 +0200

```
Merge remote-tracking branch 'origin/main'
```

commit 1e8dc758de6ee3e2b37984e1d872aa05d4fdc16f Author: yureiiko pierrefcz@gmail.com Date: Thu Jun 2 10:41:16 2022 +0200

```
UPDATES :
New documentation.
```

commit fd306d84486bc3fa437211d7b0e5b14a562fe0d3 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Wed Jun 1 17:22:21 2022 +0200

```
Update README.md
```

commit 0bc6fa9c8eb9fcd70c558b21db53db3fb903a88a Merge: 4874a42 8581bee Author: yureiiko pierrefcz@gmail.com Date: Wed Jun 1 17:20:40 2022 +0200

```
Merge remote-tracking branch 'origin/main'
```

commit 4874a42f108cdec16f635772eb647744aa208dbf Author: yureiiko pierrefcz@gmail.com Date: Wed Jun 1 17:20:20 2022 +0200

```
UPDATES :
Add a page to load data form the GPS Logger app on the participant personal phone.
```

commit 8581bee6305059e23fe7740285928063d0d365e8 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Wed Jun 1 10:12:51 2022 +0200

```
Update README.md
```

commit 9ddb35776d793ee9c8a56e75953c23a9f68a1c40 Author: yureiiko pierrefcz@gmail.com Date: Tue May 31 15:52:01 2022 +0200

```
UPDATES :
Modification to raise an error if there's a problem with the database.
```

commit fbd9d68559828b34ea561624636c3ec53a73da6a Author: yureiiko pierrefcz@gmail.com Date: Mon May 30 16:00:49 2022 +0200

```
UPDATES :
Minor modification.
```

commit b59bf52a9009eacd7b5dfb303ab26a733f382527 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 30 14:35:30 2022 +0200

```
Update README.md
```

commit 882406a4a46c631699a10427d92d1fe6020d21eb Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 30 14:35:03 2022 +0200

```
Update README.md
```

commit 6904707234d45b22cb003f25f1076f79219f8c94 Author: yureiiko pierrefcz@gmail.com Date: Mon May 30 14:26:40 2022 +0200

```
UPDATES :
Addition of SQL script to insert an admin user in the database and a script to
drop all tables.
```

commit 5618cdd24d92fabbcf620fc06f89df6937713ba1 Author: yureiiko pierrefcz@gmail.com Date: Mon May 30 12:19:57 2022 +0200

```
UPDATES :
Addition of SQL script to create tables.
```

commit c2a0f5253e36eba67e20e2e3c11b270703756dd1 Merge: 508b9ee 7743b0a Author: yureiiko pierrefcz@gmail.com Date: Mon May 30 10:47:39 2022 +0200

```
Merge remote-tracking branch 'origin/main'
```

commit 508b9ee5b76801ed903543cff07e452d8793d0f5 Author: yureiiko pierrefcz@gmail.com Date: Mon May 30 10:47:24 2022 +0200

```
UPDATES :
Some commentaries were delete.
```

commit 7743b0ad6bda831bafa3863a8a419d9b2404d7d4 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Wed May 25 14:03:07 2022 +0000

```
Update table_creation_sql
```

commit 09193840a6eb70a465c3a84160875d9ca87802e9 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 23 16:12:59 2022 +0200

```
Update README.md
```

commit d9a49006f2fa127e1d6176e5cb25a3fac8bbecc7 Merge: 42debd1 99a7269 Author: yureiiko pierrefcz@gmail.com Date: Mon May 23 16:07:51 2022 +0200

```
Merge remote-tracking branch 'origin/main'
```

commit 42debd11e9a3e1ebb6e9a477b9954595f81d23dd Author: yureiiko pierrefcz@gmail.com Date: Mon May 23 16:07:36 2022 +0200

```
UPDATES :
All pages are now up to upload Flow data and record it.
New pages :
- ```uploadFlowMeasure```
```

commit 99a72692c637125ad7f1fd12b1e01d3ad554aa94 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 23 12:07:32 2022 +0200

```
Update README.md
```

commit c0250aaf7279460eb2633a72635305004bbd7cae Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 23 12:06:43 2022 +0200

```
Update README.md
```

commit 370c195ed5a79b7ddc4f956ca8e1fe4ff5fc1d91 Merge: d21ae7b 5f8227d Author: yureiiko pierrefcz@gmail.com Date: Mon May 23 12:01:09 2022 +0200

```
Merge remote-tracking branch 'origin/main'
```

commit d21ae7b565be4921c4dd1584f6f6a81a1b910d44 Author: yureiiko pierrefcz@gmail.com Date: Mon May 23 12:00:53 2022 +0200

```
UPDATES :
First modifications to add a flow sensor insert it in a kit and link this kit with
a campaign and a participant.
New pages :
- ```addFlowSensor.html```
```

commit 5f8227d2e4856033807992abb5bdad06841a2b2e Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri May 20 10:55:45 2022 +0200

```
Update wsgi.py
```

commit 52cc4dfaf9bd6ba02c0e1892b5f2a32ef4c3db37 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri May 20 10:53:02 2022 +0200

```
Update README.md
```

commit d14b5c43f7045d86c6a9d661fcca1c749e112455 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri May 20 10:52:30 2022 +0200

```
Update README.md
```

commit 9c46ebe0ec9a3da36e3f0f31f60cf4c62af00676 Merge: f3a5b2d 28e4205 Author: yureiiko pierrefcz@gmail.com Date: Fri May 20 10:27:08 2022 +0200

```
Merge remote-tracking branch 'origin/main'
```

commit f3a5b2db5d2eafa04af5e6914d0dd01cc7c6f6d9 Author: yureiiko pierrefcz@gmail.com Date: Fri May 20 10:26:44 2022 +0200

```
UPDATES :
Add the file ```wsgi.py```. That's useful to deploy the web application.
```

commit 28e420507fdfa44d4808a46c79ef690944f52c10 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri May 20 10:11:10 2022 +0200

```
Update README.md
```

commit 319b5aa955a57c9b8a7989ddc5d6f2f289b27857 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 19 15:00:40 2022 +0200

```
Update README.md
```

commit b58b634d18e0ed949765b3ff499b213e557a3fe7 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 19 14:18:20 2022 +0200

```
Update README.md
```

commit 1bb6eeee81813aff19ccc57bbd40db0980a0cac6 Merge: 978274e 9de06f3 Author: yureiiko pierrefcz@gmail.com Date: Thu May 19 14:14:55 2022 +0200

```
Merge remote-tracking branch 'origin/main'
```

commit 978274e53bb1d50ad87712283036093ad37b244c Author: yureiiko pierrefcz@gmail.com Date: Thu May 19 14:14:25 2022 +0200

```
UPDATES :
Minor problem correction.
```

commit 9de06f34c6e752da12d3bbd8f82b68a80926b030 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 19 12:01:37 2022 +0200

```
Update README.md
```

commit d14f5815377b4706eea4ed32122d3f1cbc145e98 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 19 12:01:12 2022 +0200

```
Update README.md
```

commit a5ae3af39471c6a13e2b611784954c650eb0250e Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Wed May 18 17:28:27 2022 +0200

```
Update README.md
```

commit 932ea94e03c32135beff9a6d30064cec52956768 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Wed May 18 12:51:57 2022 +0000

```
Delete requirement.txt
```

commit 734a3bf7c37a9c0125bbe0fb3eb0d0fb93bcbcd2 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Wed May 18 12:51:44 2022 +0000

```
Delete DockerFile
```

commit d2caa193d0557e089b6b0b2387a1a7321e4bfe5f Merge: 178435e e65c939 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Wed May 18 12:47:24 2022 +0000

```
Merge pull request #1 from yureiiko/docker

Add files via upload
```

commit e65c939ba16b0f2a59f37636195406f7c989f4e6 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Wed May 18 12:46:54 2022 +0000

```
Add files via upload

First version of docker.
Not complete.
```

commit 178435ef86259261af418fc2119bdcc03fb7b2e1 Merge: 15c275d 577f917 Author: yureiiko pierrefcz@gmail.com Date: Mon May 16 11:13:26 2022 +0200

```
Merge remote-tracking branch 'origin/main'
```

commit 15c275dec590bf56b413dc5be79421407b5da889 Author: yureiiko pierrefcz@gmail.com Date: Mon May 16 11:11:51 2022 +0200

```
UPDATES :
Addition of the page ```canarinFidas.html```. This page isn't working.
Modifiation of all page's header.
```

commit 577f9173e9e3239bcead009c5e3de49c9ad0e87a Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri May 13 11:35:56 2022 +0200

```
Update README.md
```

commit b844daeeb08c43e040fcb228dca7be77ee90e489 Author: yureiiko pierrefcz@gmail.com Date: Fri May 13 11:33:46 2022 +0200

```
UPDATES :
Addition of the fonctional page ```uploadFidas.html```
```

commit ee638a00b0615215f5cf793a6cf9e90a1c6b6a1d Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 12 11:42:02 2022 +0200

```
Update README.md
```

commit e86f45c37cc002c5ec3f30d0953f249f89075e57 Author: yureiiko pierrefcz@gmail.com Date: Thu May 12 11:41:25 2022 +0200

```
UPDATES :
Addition of the fonctional page ```uploadTeom.html```
```

commit c04c95fbcf3d06a4823708ee6d749374a798933b Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Tue May 10 11:32:28 2022 +0200

```
Update README.md
```

commit e71c28c67f4d31cf1507ca4e273f88851ae97b92 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Tue May 10 11:31:08 2022 +0200

```
Update README.md
```

commit b413aa08ddfb8e8915b53eb73f5fe3dd868d4202 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Tue May 10 11:15:05 2022 +0200

```
Update README.md
```

commit 347520624453265a36c182d4cb4c7cc9f46e3485 Author: yureiiko pierrefcz@gmail.com Date: Tue May 10 11:14:08 2022 +0200

```
UPDATES :
Addition of the fonctional page ```downloadKitData.html```
```

commit b2555dddc115cbfcc3c86acac32510eacd555524 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 9 16:22:18 2022 +0200

```
Update README.md
```

commit 21589f36cc7404e8475d3f9a31eecab22cbc6b4e Author: yureiiko pierrefcz@gmail.com Date: Mon May 9 16:21:14 2022 +0200

```
UPDATES :
Addition of the fonctional page ```downloadCampaignData.html```
```

commit a75faaf2c57ce109ddb39399e181d947db841840 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 9 14:58:01 2022 +0200

```
Update README.md
```

commit 821a6e42cf7e0cdce3bf7b82c0657660b1d716f7 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri May 6 16:41:28 2022 +0200

```
Update README.md
```

commit d2e4e88e4dbb892c003c7fd4d39d6c9f13da9d55 Author: yureiiko pierrefcz@gmail.com Date: Fri May 6 16:39:56 2022 +0200

```
UPDATES :
It's nom possible to use the ```downloadAssignedData.html``` page.
```

commit c9ae60910febb3e647440caeceb4ad4dabc867b8 Author: yureiiko pierrefcz@gmail.com Date: Fri May 6 12:32:46 2022 +0200

```
UPDATES :
Minor modifications to fix errors.
```

commit de5fc91cd81a80a06dce367c17432e0a74b67bec Author: yureiiko pierrefcz@gmail.com Date: Thu May 5 12:01:50 2022 +0200

```
UPDATES :
Modifications on ```downloadAssignedData.html```.
```

commit 030f6f5afcd0cdc65f96680ed0232f605289f6dd Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 5 11:37:47 2022 +0200

```
Update README.md
```

commit 4cf594f5affb2fd1d0cb88946a51247ddfbd55a7 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 5 11:10:43 2022 +0200

```
Update README.md
```

commit 2ba62dca514a828a5a305842a525cf2c1d74c8c2 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 5 11:08:26 2022 +0200

```
Add files via upload
```

commit adbc5a91d2e59bc65d065d01819f44ed78e2e128 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Thu May 5 11:06:52 2022 +0200

```
Add files via upload

UPDATE :
The way to check if a row was already inserted change. Now the program check in
fonction of the device's id and of the date.
It's now possible to upload aethalometer, actris, report and to check the history
of report uploading.
```

commit 572ca0e80cb7298128d506449200db9abe14d504 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 16:35:04 2022 +0200

```
Add files via upload
```

commit 2e847c0528d27a356fea61d06dd840f416458fa0 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 16:34:31 2022 +0200

```
Add files via upload
```

commit a285b3017d2d1c538e0f4f289d1df3764f897a93 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 16:33:21 2022 +0200

```
Update README.md
```

commit 9d434b36975f26a4f1e09b07f1c4f890891a6843 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 16:32:07 2022 +0200

```
Add files via upload

UPDATE
It's now possible to load reports, tablet's activity and tablet's events. All
history for this loads can be seen by an admin.
```

commit e9d4405f6b1bd582b243782e4e6edcfef356ec78 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 16:29:30 2022 +0200

```
Add files via upload
```

commit 885e53518d98a412b0d9a1829428c00fcbcf7718 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 16:28:20 2022 +0200

```
Add files via upload

Update
new data folders
```

commit 9d17074065e3ad05fa1bf50717637f298ddf3766 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 14:33:13 2022 +0200

```
Update README.md
```

commit bc5dcfdadab917203a61ca6b3e428535079f9cc1 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 14:32:55 2022 +0200

```
Delete scriptPython directory
```

commit c1436feacb570ad9e3b3a22917722c6cb4a62bfa Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Mon May 2 14:21:26 2022 +0200

```
Update README.md
```

commit 999905ac6ff31a5778cb79f412018484bdbd1ed5 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri Apr 29 17:37:54 2022 +0200

```
Update README.md
```

commit f8759e0e11e6e561271cca86eb392cbd389b18bf Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri Apr 29 17:37:29 2022 +0200

```
Create README.md
```

commit 310e40980ea4cc5bb0b0358d21f631ca0221573f Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Fri Apr 29 17:33:44 2022 +0200

```
Add files via upload

Updates
Kits can be created.
It's now possible to create link between campaign, participant and kit. When we
associate a kit to a participant the old kit association is erased.
The device id seeking is now based on this link.
It's also possible to see the history of file uploading.
```

commit 58d107b8f3dfc7bfc778d2aa3390df5f23852a5d Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Tue Apr 26 16:42:52 2022 +0200

```
Add files via upload

New functionnalities
It's possible to add a user, a participant and to upload gps tablet files.
```

commit 1786bb3707a3c428d0e0e40a0f9cf3e3c9d7c5c4 Author: yureiiko 101016323+yureiiko@users.noreply.github.com Date: Sun Apr 24 20:58:14 2022 +0200

```
Add files via upload

First Commit
It's possible for the user to upload Diams, Cairsens and AE51 measures. The
administrator can download uploaded files.
```

Glossaire

Déploiement : Installation des composants d'une ou plusieurs applications dans un environnement de production.

Plateforme (informatique) : Environnement servant d'interface pour l'utilisation ou la gestion de services.

Injection SQL : Type d'attaque sur une application reliée à une base de données permettant à l'attaquant d'insérer et d'exécuter des commande SQL.

SQLi : Faille de sécurité permettant l'injection de commandes SQL.

Sitographie

Site Internet du laboratoire DAVID : https://www.david.uvsq.fr/accueil/

Site de présentation du projet Polluscope : http://polluscope.uvsq.fr/index.php/fr/

Dépot Github de la plateforme Polluscope : https://github.com/yureiiko/pollu.git (pour des questions de confidentialité, ce dépôt est privé)

Dépôt Github du rapport de stage : https://github.com/yureiiko/david_internship_report.git (ce dépôt deviendra privé après la soutenance de stage)

Liens utilisés pour l'apprentisage de **Python Flask** :

- https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3
- https://pypi.org/project/Flask/
- https://openclassrooms.com/fr/courses/4425066-concevez-un-site-avec-flask/4525776-installez-flask
- https://sysadmin.cyklodev.com/deployer-une-application-flask/
- https://pythondex.com/convert-gpx-to-csv-python

Liens utilisés pour l'apprentissage de la technologie **Docker** :

- https://www.digitalocean.com/community/tutorials/how-to-build-and-deploy-a-flask-application-using-docker-on-ubuntu-20-04
- https://www.youtube.com/watch?v=SXB6KJ4u5vg
- https://www.youtube.com/watch?v=cWkmqZPWwiw
- https://linuxhint.com/install_docker_kali_linux/

Bibliographie

- Évaluation HCERES DAVID UVSQ 25 Mai 2018