**Projet CUBE**

Station météo

« Ce qui se conçoit bien s’énonce clairement, et les mots pour le dire arrivent aisément » N. Boisleau.

Avant de dérouler le fil du projet il convient de bien le concevoir et en déterminer les contours afin d’au mieux répondre au cahier des charges. Le projet consiste alors à produire une station météorologique miniature au moyen d’un Raspberry Pi. Il s’agira alors de récupérer les informations du Raspberry au moyen d’une API connectée à une BDD. L’API distribue alors les données sur un site internet qui les affiche sur un graphique et spécifie la localité de la station.

**Problématique :**

L’entreprise Lepetit à fait l’acquisition d’Atmos qui commercialise des stations météorologiques et stocke ses données sur un serveur cloud. La problématique est de changer de modèle afin de réduire les couts de l’application. Ainsi nous avons été sollicités pour refonder l’application afin de l’héberger désormais sur un serveur local, et de libérer le code source.

Sommaire

[I- Organisation 3](#_Toc73544979)

[II- Elaboration de la Base de Donnée 4](#_Toc73544980)

[III- Conception de l’API 5](#_Toc73544981)

[IV- Conception du Front 6](#_Toc73544982)

[V- Les problèmes rencontrés. 9](#_Toc73544983)

[VI- Axes D’Améliorations 11](#_Toc73544984)

[Conclusion 12](#_Toc73544985)

# Organisation

Nous nous sommes organisés au moyen d’un Trello qui nous a permis de définir plusieurs objectifs afin d’être plus efficace sur le projet. Nous avons réalisé les différentes tâches tous ensemble.

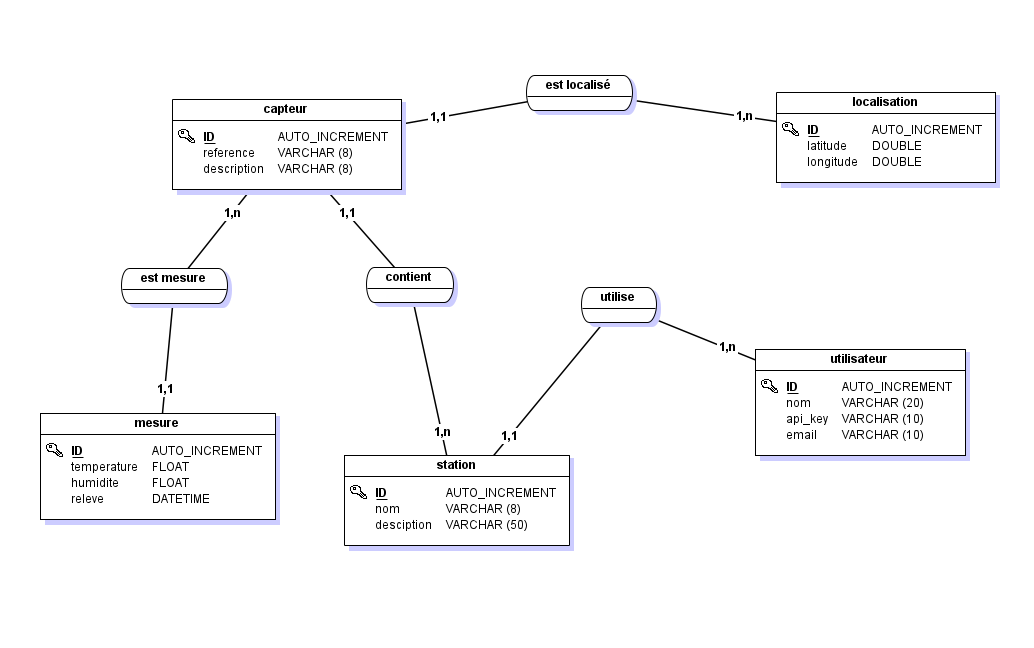
Pour les besoins de la présentation nous avons créé un serveur en Local avec Xamp, pour héberger le site : un serveur Apache et mySQL.

Nous avons créé un dépôt Git, nous avons donné les accès Git à tout le groupe de travail afin d’au mieux pourvoir collaborer. Dans ce dépôt nous avons mis tout le code de l’Api ainsi que du site internet.

Faute de moyens de moyens technique nous n’avons pas pu assembler le Raspberry Pi, mais nous avons générer des données directement dans la base de données.

# Elaboration de la Base de Donnée

Nous avons élaboré un MCD afin d’au mieux cadrer le projet et de stocker les informations dont nous avons besoin pour avoir une Base de données le plus proche du fonctionnel :



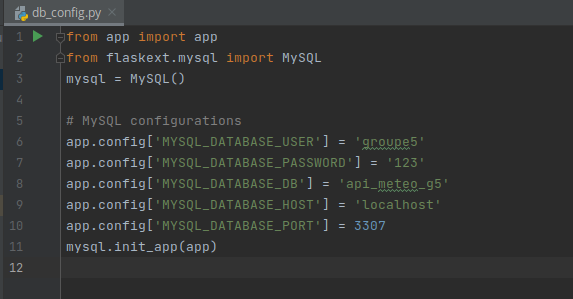
Nous sommes partis sur quatre entités : utilisateur, station, capteur, mesure et localisation.

Ensuite nous avons générer un script SQL que nous avons importé dans la Base de Données de notre serveur mySQL.

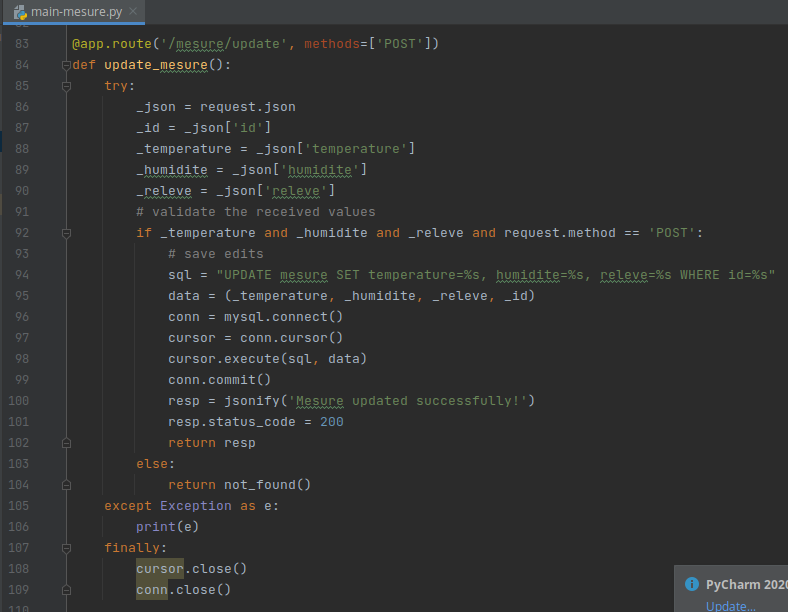
# Conception de l’API

Nous avons ensuite réalisé une API en python car c’est un langage facile d’accès que nous avons utilisé à plusieurs reprises, ce dernier, est normalement bien interfacé avec le Raspberry.

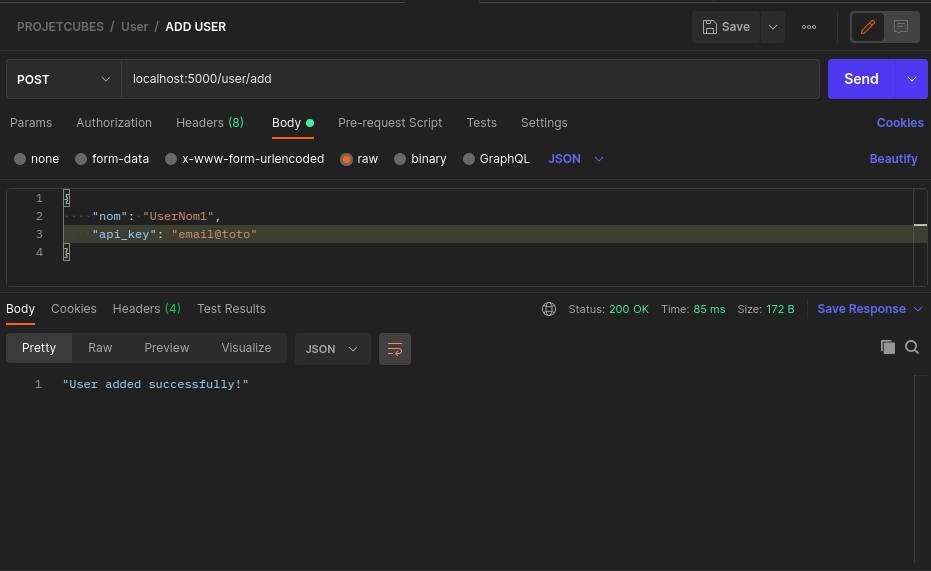
Nous avons créé un fichier db\_config.py afin de relier la base de données à l’API, il contient les accès à la base en local. Puis les fichiers pour le CRUD qui permettront de créer, récupérer, mettre à jour ou supprimer les informations de la base. Chacune de ces opérations ont une route url qui permet d’y accéder.



Nous avons ensuite paramétré nos différentes routes pour les données des tables dont nous avons besoin :



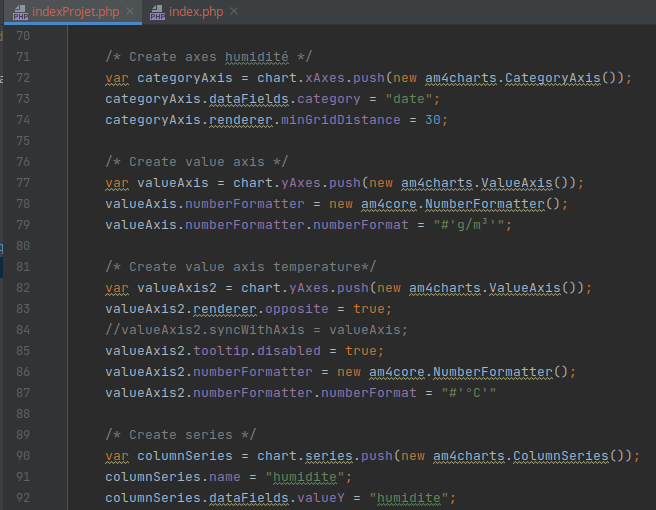
Nous avons testé nos routes au moyen de Postman :

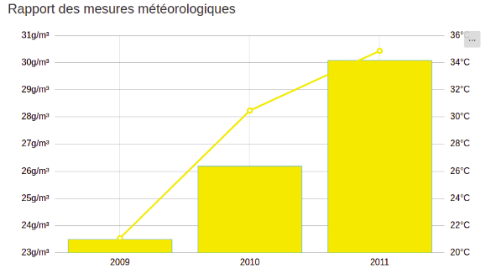


# Conception du Front

Objectif du front : ergonomie, confidentialité

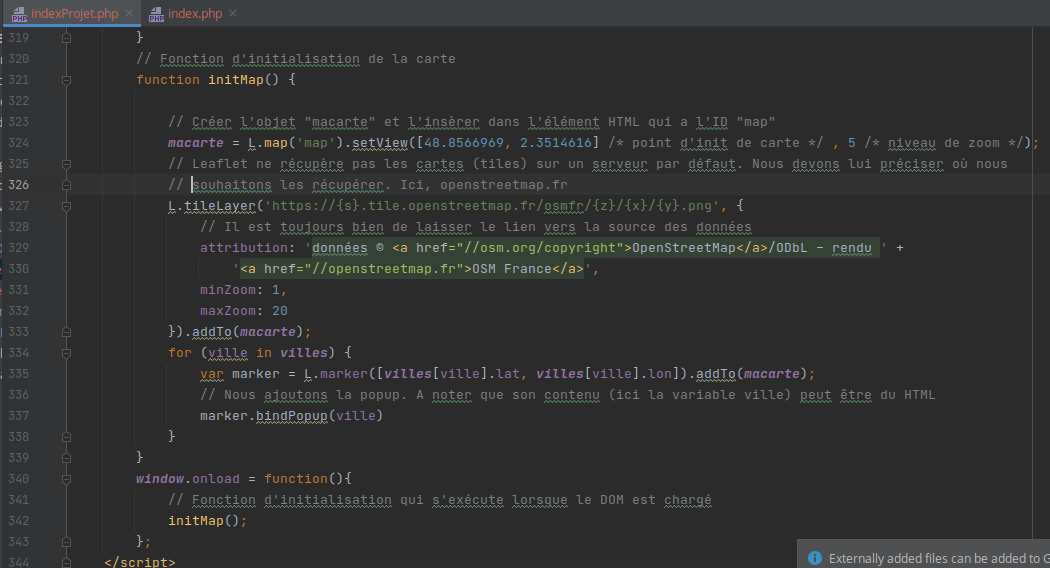
Nous avons ensuite réalisé un site web nous nous sommes aidés de plusieurs Framework que nous avons trouvés sur le net. Notre site internet est donc en HTML, CSS, JavaScript et un peu de PHP. Notre graphique a été réalisé en HTML et JavaScript avec AmCharts 4 :

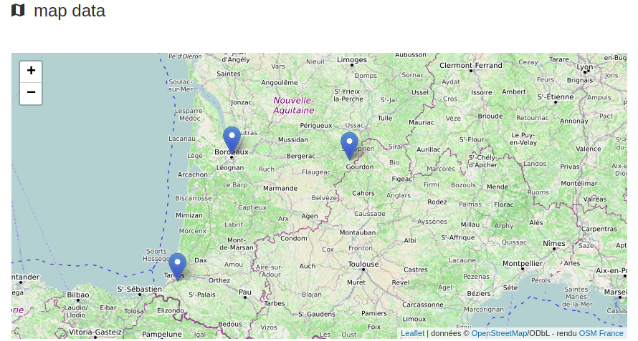




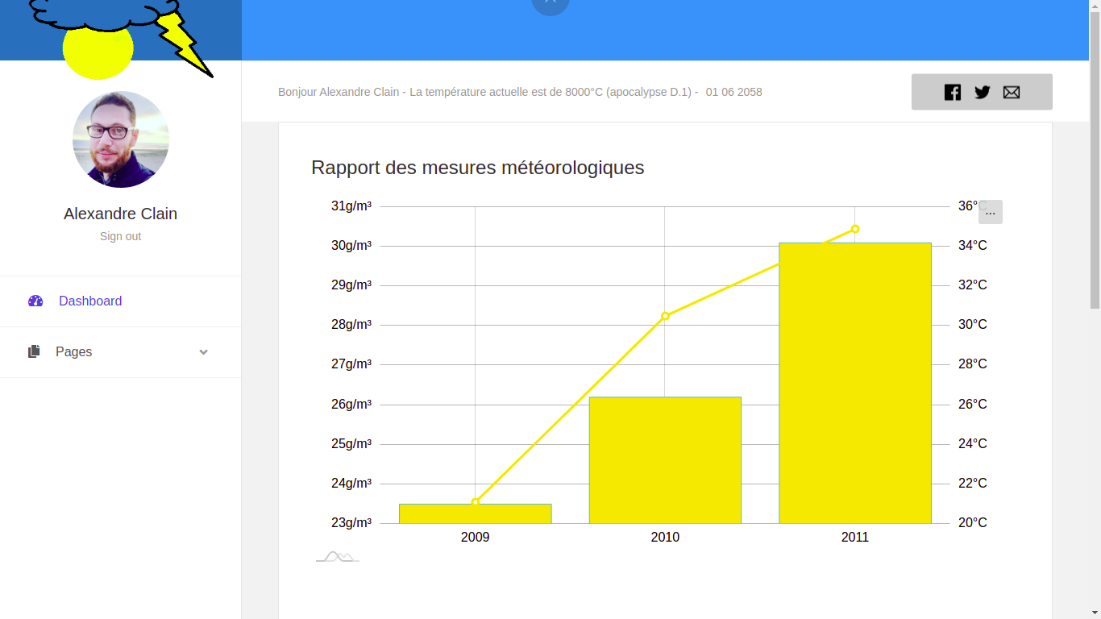
Nous avons privilégié ce langage a cause sa fonction Chart qui est déjà implémentée et facile d’utilisation, mais aussi à cause de sa responsivité.

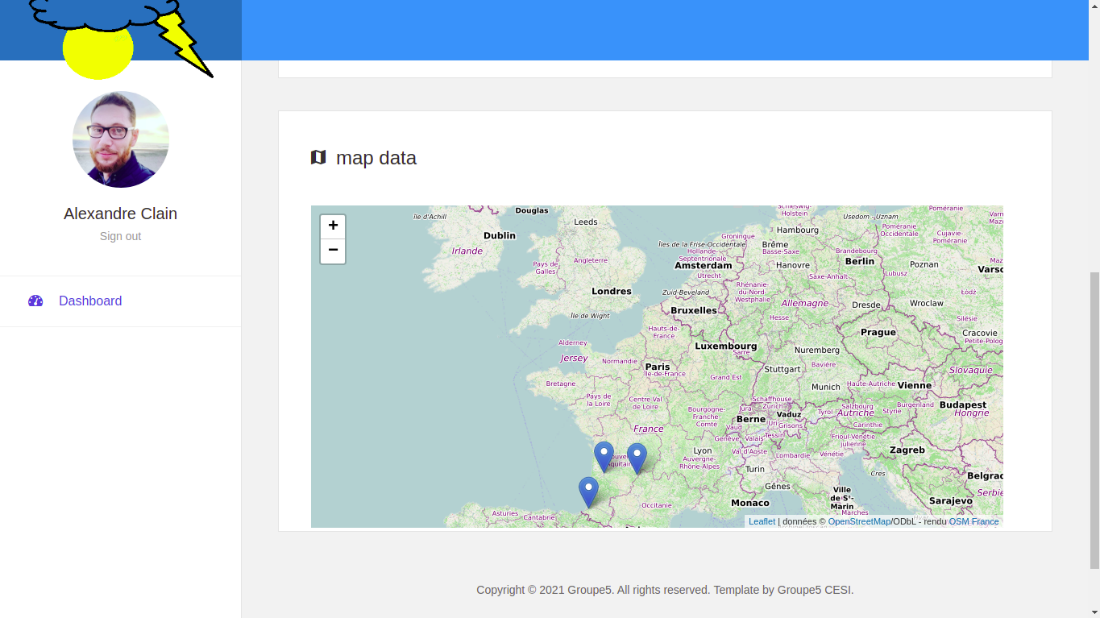
Et notre carte également réalisé en Javascript nous l’avons réalisé au moyen d’openstreet map grâce à ses fonctions intégrées cela nous a permis d’importer directement le script JavaScript et de l’implémenter dans notre site :





Voici le rendu général de notre site :





Au moyen d’un script PHP nous avons relié notre front à notre Api pour capter les différentes informations de notre Base de données :

Nous avons également pris soin de mettre la possibilité de partage sur les réseaux sociaux afin de pouvoir communiquer les relevés graphiques.

Les données entrées en base de données apparaissent bien sur notre site internet hébergé en local.

Le contrat est alors rempli notre API est fonctionnelle, notre base de données est bien implémentée, il ne manque plus que notre Raspberry Pi qui pourra nous transmettre des données enregistrées par notre Api.

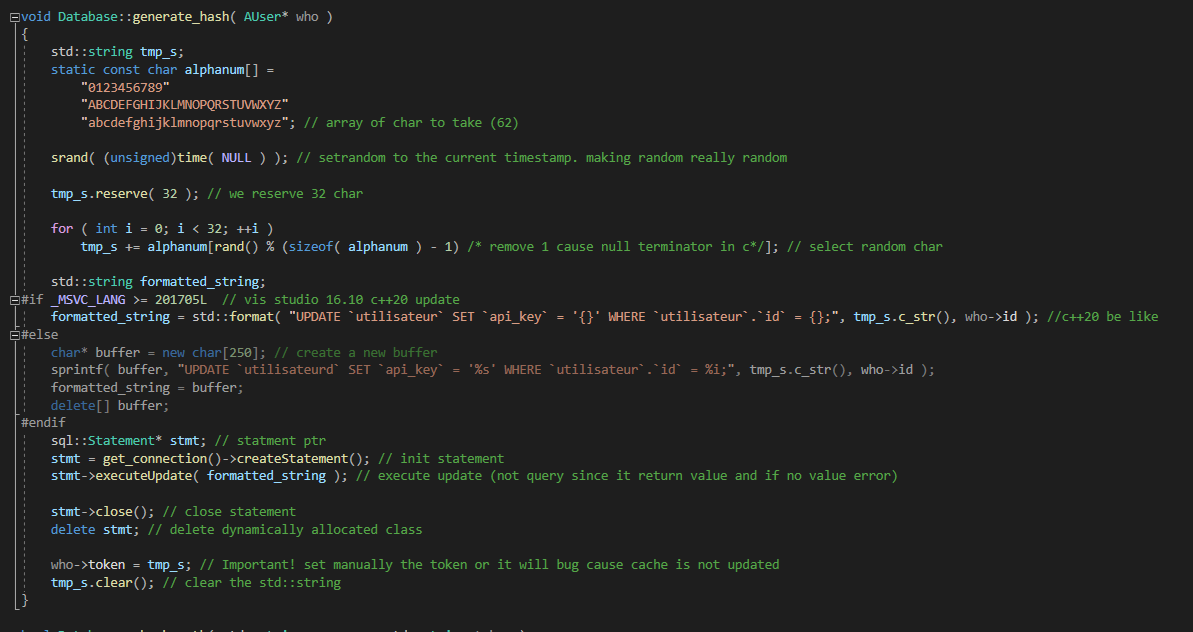
# Les problèmes rencontrés.

Le principal problème rencontré pour le développement du projet est lié au contexte particulier dans lequel nous avons développé le projet. En effet les circonstances nous ont imposés de devoir tout concevoir en distanciel qui peut avoir des avantages mais qui freine une certaine dynamique de groupe. Un autre problème qui est lié au précédant est le fait de ne pas avoir pu opérer de véritables mesures au moyen du Raspberry, même si in fine notre Api possède des routes claires et faciles à utilisés, nous n’avons pas pu manipuler l’objet.

Nous avions également commencé initialement une API en C++, suite à la sortie de visual studio 16.10, et du c++20. Nous avons donc commencé par utiliser siesta, une librairie qui est utile pour faire du post/get en c++. Mais l’api manquait d’une chose qui est pourtant très simple, les headers. Nous avons donc fait un fork (<https://github.com/Jerem584/siesta>) et nous l’avons utilisé.

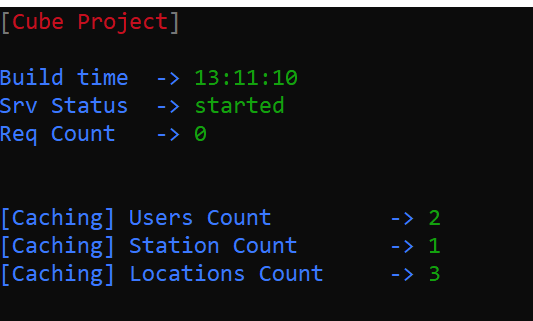
Suite à cela, nous avons mis en place une solution de caching, qui récupérer les utilisateurs de la base tous les x temps.

Suite à cela, nous avons mis en place un token d’authentification, la solution, qui peut paraitre assez compliqué au premier abord, mais qui est en fait très simple de compréhension.



On a fait une route /user/auth, qui avait 2 besoins (deux headers (username & password)) dans ce cas-là, notre rest api recherche dans le cache, s'il trouve une correspondance, il génère un token, le met dans la base de données, et force le rafraichissement du caching, puis retourne le token.

Suite à cela, l’api privé peut être contacté en utilisant un header nommé token.



Mais cela nous a semblé complexe, en effet le Python est beaucoup plus adapté pour concevoir une API, d’autant que plusieurs personnes dans notre groupe ne sont pas très à l’aise avec le C++.

Un autre problème est celui de la connexion sécurisée que nous n’avons pas pris en compte pour notre Api et notre site.

# Axes D’Améliorations

Nous aurions pu, si nous avions eu plus de temps, plus pousser tout l’aspect sécurisation de l’API et du site internet grâce à un système d’authentification. Nous aurions pu également améliorer l’accessibilité du site afin de permettre au public handicaper de pouvoir utiliser notre interface.

Le matériel Raspberry nous étant pas fournis nous n’avons pas pu tester certaines fonctionnalités de notre Api pour voir si cette dernière peut fonctionner.

# Conclusion

Ainsi nous avons pu élaborer une Api fonctionnelle, connectée a une base de données, un site web hébergé en local et connecté à cette Api. Nous avons rempli le cahier des charges, nous avons refondé la station météo à moindre cout, notre hébergement étant local.

Nous espérons avoir participer a la prospérité de l’entreprise Lepetit et espérons que vous referez appel à nos services pour vos futurs projets.