IUT DE COLMAR

SAE 24

PROJET INTÉGRATIF

Partie Web - Base de données

RT11
Martin Baumgaertner

RT12
Mehdi Rehm

RT11
Sâji Doghmane

24 juin 2022

Table des matières

Table des codes

Table des figures

1 Introduction

Le but de cette dernière partie est de pouvoir afficher sur une page web que nous développons en Django les valeurs de températures avec plusieurs types de complications. Par exemple, nous devons être capable de lister les données de températures en fonction d'un capteur, ou de proposer des modifications de champs.

2 Mise en place du projet Django

2.1 Connexion du projet Django à la base de données

Pour pouvoir connecter notre projet à la base de données que nous avions remplie dans la partie précédente avec le script de récupération des données, nous devons modifier le fichier `settings.py` de notre projet Django.

```
DATABASES = {
        'default': {
                'ENGINE': 'django.db.backends.mysql',
                'NAME': 'temp',
                'USER': 'martin',
                'HOST': '10.37.129.3',
                'PORT': '3306',
                'PASSWORD': 'martin',
        }
}
```

Code 1 – Configuration de `settings.py`

Nous mettons l'adresse `10.37.129.3`, qui correspond à l'adresse de ma machine windows où est herbergée notre base de données. Puis, le nom de notre base de données est `temp`. Enfin, `martin` correspond au nom utilisateur avec le mot de passe `martin`. J'ai au préalable créé cet utilisateur et autorisé la connexion depuis la machine où est mon projet Django.

L'adresse IP `10.37.129.4` correspond à l'adresse de ma machine où je fais mon projet Django. Pour créer cet utlisateur, lui donner tous les droits et l'autoriser à se connecter il faut utiliser les commandes suivantes :

```
CREATE USER 'martin'@'10.37.129.4' IDENTIFIED BY 'martin';
GRANT ALL PRIVILEGES ON *.* TO 'martin'@'10.37.129.4';
FLUSH PRIVILEGES;
```

Code 2 – Création de l'utilisateur

2.2 Récupération des tables

Nous pouvons donc désormais nous connecter à la base de données depuis le projet Django, et faire un `python3 manage.py makemigrations` et `python3 manage.py migrate` pour que les données soient migréées. Toutes les tables sont sauvegardées et nous observons que les tables que Django créee lors de la migrations du projet sont bien présentes :

```
mysql> show tables;
+----------------------------+
| Tables_in_temp             |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
| sensors                    |
| sensors_data               |
+----------------------------+
12 rows in set (0.05 sec)
```

FIGURE 1 – Les tables créées par Django

3 Développement

3.1 Création du fichier `models.py` à partir de la base de données

Ensuite, nous allons créer le fichier `models.py` qui va permettre de créer les modèles nécessaire pour la suite du projet. Pour ce faire, nous allons utiliser la commande suivante :

```
python3 manage.py inspectdb > models.py
```

Code 3 – Création du fichier `models.py`

Suite à cela, notre fichier est bien créé et nous pouvons commencer le développement.

3.2 Affichage des 2 capteurs sur une page d'accueil

Nous nous retrouvons donc avec les 2 capteurs, je veux les afficher sur une page d'accueil. Pour ensuite permettre à l'utilisateur de naviguer à travers les données des capteurs en fonction du capteur qu'il sélectionne. Premièrement je fais une "vue all" pour récupérer les informations des deux capteurs :

```
def index(request):
    sensors = Sensors.objects.all()
    sensorsdata = SensorsData.objects.all()
    return render(request, 'index.html', {'sensors': sensors})
```

Code 4 – Création de la vue `all`

Avec un peu de développement HTML et CSS nous obtenons ce style de page, qui nous permet d'avoir les deux capteurs sur une seule page :

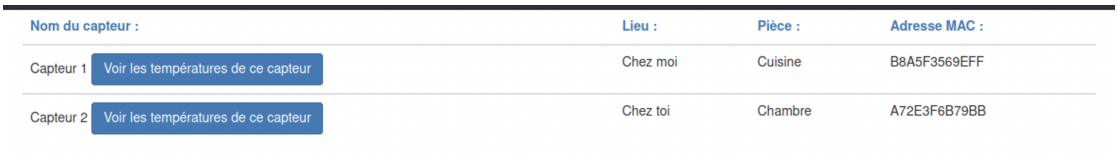


FIGURE 2 – Page d'accueil

Avoir fait le choix d'avoir uniquement quelques éléments sur la page d'accueil, donc ici les 2 capteurs permet de ne pas surcharger la page d'index et d'avoir un rendu minimaliste que j'apprécie tout particulièrement.

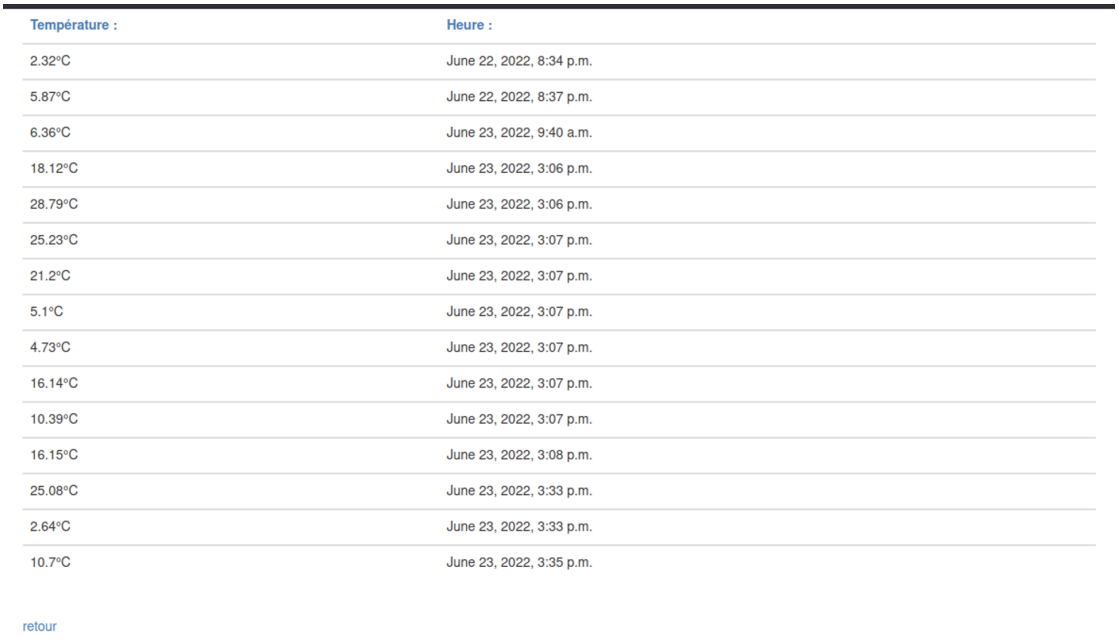
3.3 Affichage des données en fonction d'un capteur

Pour pouvoir afficher les valeurs d'un capteur en fonction de celui que l'on choisit nous mettre en place un filtre qui sélectionne les données du capteur que l'on veut afficher. Pour ce faire, je vais créer cette vue :

```
def temp(request, id):
        temp = SensorsData.objects.filter(sensor__id=id)
        return render(request, 'data.html', {'temp': temp})
```

Code 5 – View pour afficher les données d'un capteur

Encore une fois, avec un peu de HTML et de CSS, nous arrivons assez facilement à générer ce genre de tableau pour lister les données du capteur de température choisi :



Température :	Heure :
2.32°C	June 22, 2022, 8:34 p.m.
5.87°C	June 22, 2022, 8:37 p.m.
6.36°C	June 23, 2022, 9:40 a.m.
18.12°C	June 23, 2022, 3:06 p.m.
28.79°C	June 23, 2022, 3:06 p.m.
25.23°C	June 23, 2022, 3:07 p.m.
21.2°C	June 23, 2022, 3:07 p.m.
5.1°C	June 23, 2022, 3:07 p.m.
4.73°C	June 23, 2022, 3:07 p.m.
16.14°C	June 23, 2022, 3:07 p.m.
10.39°C	June 23, 2022, 3:07 p.m.
16.15°C	June 23, 2022, 3:08 p.m.
25.08°C	June 23, 2022, 3:33 p.m.
2.64°C	June 23, 2022, 3:33 p.m.
10.7°C	June 23, 2022, 3:35 p.m.

retour

FIGURE 3 – Page de données

3.4 Mise à jour automatique de la page

Pour que nous puissions laisser tourner notre script qui récupère les données, et pouvoir laisser la page des données affiché, nous devons écrire un petit script en javascript pour que la page se mette à jour automatiquent. Voici le script que j'ai utilisé :

```
setTimeout(function(){
        window.location.reload(1);
    }, 2000);
```

Code 6 – Script pour mettre à jour la page

4 Conclusion

En fin de compte, nous avons vu comment créer un projet Django, dépandant d'une base de données, elle-même dépendante d'un script python retransmettant des données transmises en MQTT. Pour finir, cet SAE fût pour nous un gain de connaissances énormes, en réseau, téléphone et programmation. Elle nous a apporté les savoirs qu'il fallait pour pouvoir mener à bien un projet complet et fonctionnel. La compléxité de la tâche ainsi que le temps qui nous était accordé (1 jour par partie), nous a appris la gestion du temps et la répartition des tâches. Le fait que nous étions seuls nous a permis d'apprendre à se débrouiller seul, sans forcément dépendre de quelqu'un.