24 JUIN 2022

SAE24
PROJET INTÉGRATIF

PARTIE WEB/BDD

RAPPORT TECHNIQUE

GROUPE 1
JULES BRUTSCHY – ERNEST HALBOUT – BAHIR BOUDOUMA
Groupe C

Table des matières

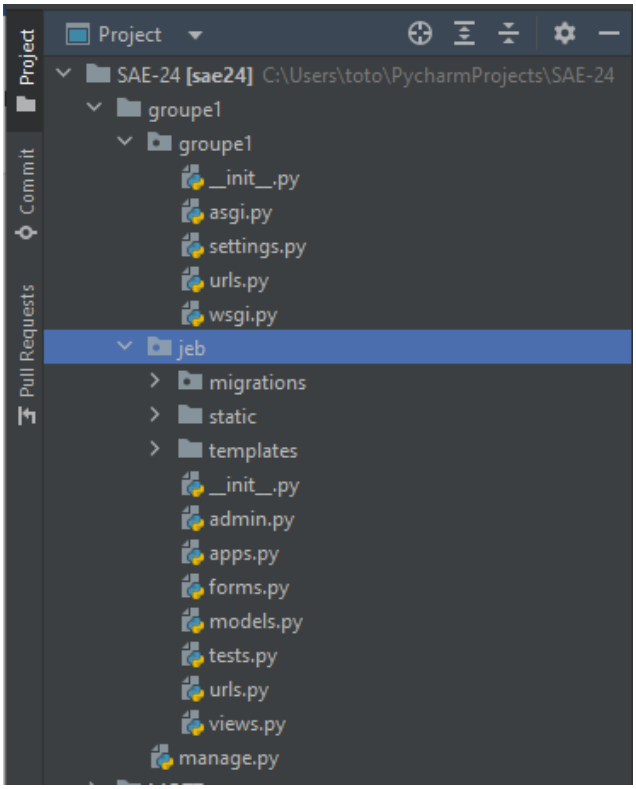
1. Introduction :

Dans le cadre de notre formation en BUT Réseaux et Télécommunications nous avons eu la possibilité de travailler sur un projet qui a eu pour but de mobiliser et valider toutes les compétences acquises au sein des ressources précédentes (faire un site web, utiliser une VM, gérer un OS , y installer des services, travailler en groupe, être capable de trouver des solutions à vos problèmes en autonomie, …) cette SAÉ 24 nous a permis de renforcer nos différentes compétences techniques qui se traduisent par la création d'un site web dynamique avec un serveur local à l'aide différents éléments tel Django et python ainsi que la création d'une base de donnés externe relié au site web et une machine virtuel qui collecte les données et transfère ces informations dans la base de données. A l'issue de cette expérience nous avons acquis diverses aptitudes externes comme la recherche en autonomie de documentation des multiples procédures techniques requises pour l'avancée de notre projet.

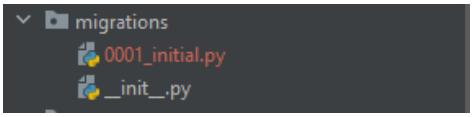
2. Mise en place du site web

a) Création de l'application

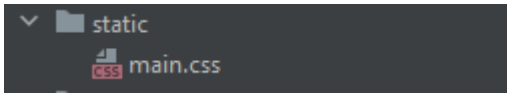
Nous avons commencé par la création de l'application « jeb » et le projet « groupe1 » dans lequel on va créer nos différents programme python afin de gérer au mieux la base de données externe et la manipulation de ces données.



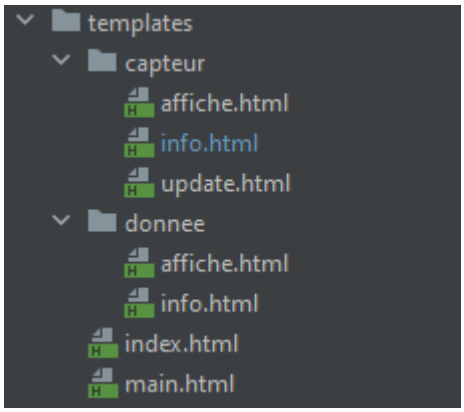
Dans le dossier migration on retrouve les données de la base de données.



Dans static on peut voir le fichier CSS qui habille nos pages HTML.



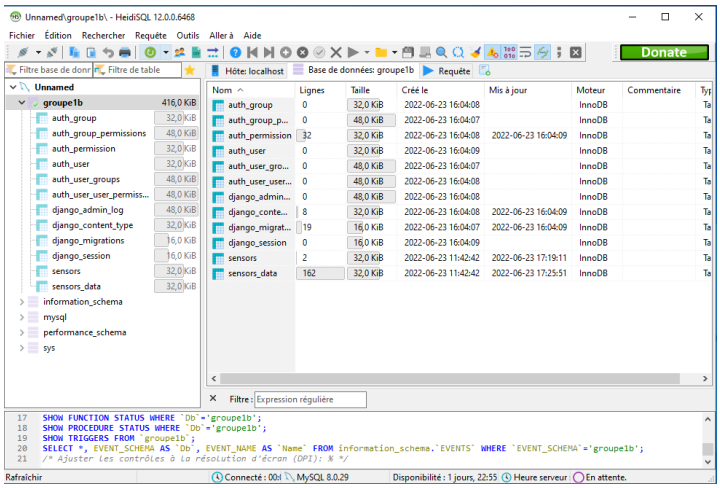
Pour finir dans le dossier Templates on retrouve nos différente pages HTML.



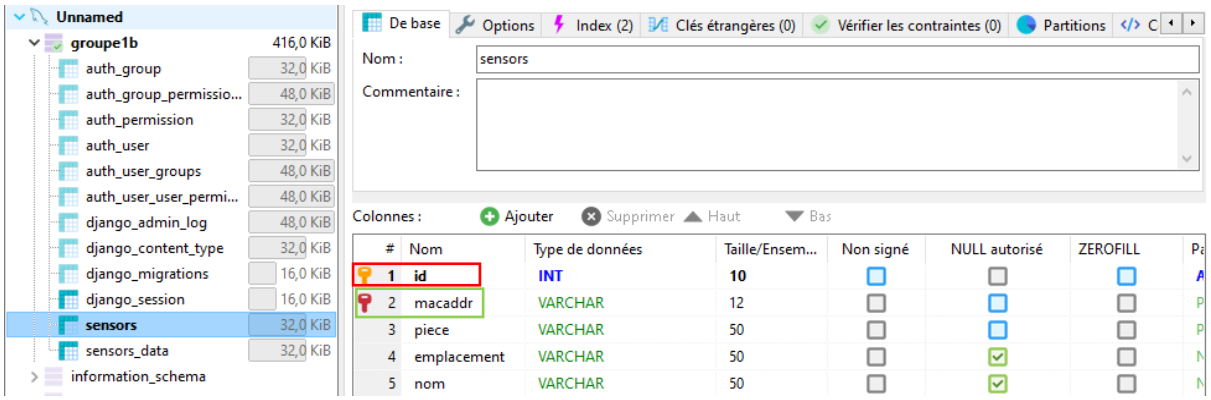
Au niveau des configurations de l'application, on renseigne nos informations de la base de données qui pour l moment est en localhost mais configuration déploiement du serveur web sur une machine linux à la suite

b) Base de données

Notre base de données à été mise en place sur le logiciel « Heidimysql »



On retrouve nos différentes tables, « sensors » et « sensors_data », et leurs caractéristiques au niveau de leur champs, car il faut respecter les demandes de l'enseignant et donc prendre en compte la clé primaire et la clé étrangère qui permet de faire le lien entre deux tables.



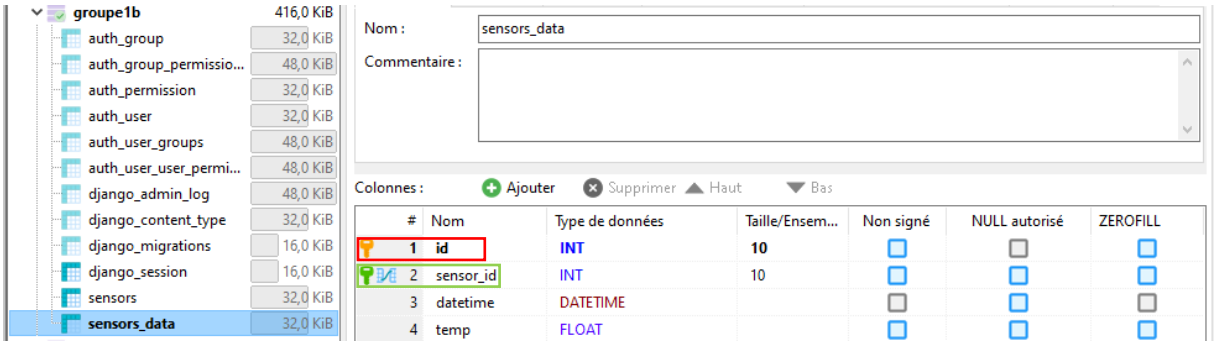
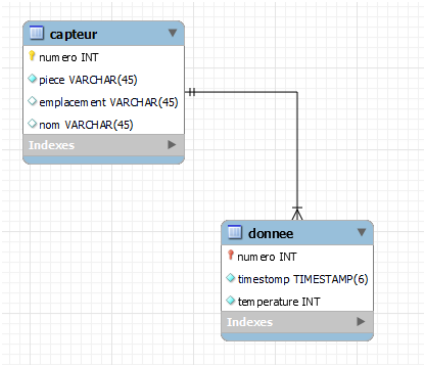
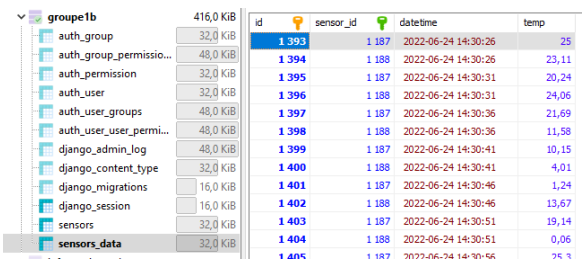
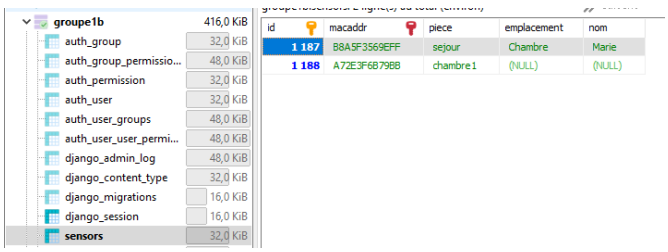


Schéma des liens :



On retrouve les données dans la base de données envoyés depuis le script python qui permet de transférer les informations du Topic MQTT directement vers la base de données





c) Création des programmes python

Afin de mieux gérer la manipulation des données on met en place ces fonctions qui nous permettent d'avoir une visualisation des données de la base de données.

```python
def index(request):
    return render(request, 'index.html')

def capteur(request):
    sensors = sensors1.objects.all()
    return render(request, 'capteur/info.html', {'sensors': sensors})

def donnee(request):
    data = sensors_data.objects.all()
    return render(request, 'donnee/info.html', {'data': data})
```

ID	Mac	Nom	Date	Pièce	Emplacement	Temp
1187	B8A5F3569EFF	Marie	June 24, 2022, 2:30 p.m.	sejour	Chambre	25.0
1188	A72E3F6B79BB	None	June 24, 2022, 2:30 p.m.	chambre1	None	23.11
1187	B8A5F3569EFF	Marie	June 24, 2022, 2:30 p.m.	sejour	Chambre	20.24
1188	A72E3F6B79BB	None	June 24, 2022, 2:30 p.m.	chambre1	None	24.06
1187	B8A5F3569EFF	Marie	June 24, 2022, 2:30 p.m.	sejour	Chambre	21.69

MAC	Pièce	Emplacement	Nom	Mise a jour	
B8A5F3569EFF	sejour	Chambre	Marie	Mise à jour	Trier1
A72E3F6B79BB	chambre1	None	None	Mise à jour	Trier1

Fonction table Trier par ID.

ID	Mac	Nom	Date	Pièce	Emplacement	Temp
1187	B8A5F3569EFF	Marie	June 24, 2022, 2:30 p.m.	sejour	Chambre	25.0
1187	B8A5F3569EFF	Marie	June 24, 2022, 2:30 p.m.	sejour	Chambre	20.24
1187	B8A5F3569EFF	Marie	June 24, 2022, 2:30 p.m.	sejour	Chambre	21.69
1187	B8A5F3569EFF	Marie	June 24, 2022, 2:30 p.m.	sejour	Chambre	10.15

Ensuite ces programmes nous permettent de mettre à jour les données correspondantes à l'emplacement et le nom :

```python
def update(request, id):
    sensors = sensors1.objects.get(pk=id)
    form = SensorsForm(model_to_dict(sensors))
    if request.method == "POST":
        form = SensorsForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect("/capteur/info")
    else:
        return render(request, "capteur/update.html", {"form": form, "id": id})


def updatetraitement(request, id):
    sensors = SensorsForm(request.POST)
    bak = sensors1.objects.get(id=id)
    if sensors.is_valid():
        sensors = sensors.save(commit=False)
        sensors.id = id
        sensors.macaddr = bak.macaddr
        sensors.piece = bak.piece
        sensors.save()
        return HttpResponseRedirect("/capteur/info.html")
    else:
        return render(request, "capteur/info.html", {"form": sensors, "id": id})
```

Mise à jour

- Emplacement: Chambre
- Nom: Marie

Envoyer les données

Le programme python url.py nous permet d'identifier nos différents programmes vers le lien ciblé.

```python
urlpatterns = [
    path('', views.index),
    path("capteur/info.html", views.capteur),
    path('capteur/update/<int:id>', views.update),
    path('capteur/info/<int:id>', views.updatetraitement),
    path("donnee/info.html", views.donnee),
    path('capteur/affiche/<int:id>', views.filtreid),
]
```

d) Création des pages HTML

Afin d'afficher la visualisation précédemment citée à l'aide de la fonction sur python, nous devons également créer une page HTML qui permet d'afficher chaque élément de la liste.

```html
<table>
    <tr>
        <th>MAC</th>
        <th>Pièce</th>
        <th>Emplacement</th>
        <th>Nom</th>
        <th>Mise a jour</th>
    </tr>
    {% for sensor in sensors %}
    <tbody>
    <tr>
        <td>{{sensor.macaddr}} </td>
        <td>{{sensor.piece}}</td>
        <td>{{sensor.emplacement}} </td>
        <td>{{sensor.nom}} </td>
        <td><a href="/capteur/update/{{sensor.id}}">Mise à jour</a></td>
        <td> <a href="/capteur/affiche/{{sensor.id}}">Trier1</a></td>
    </tr>
    {% endfor %}
```

3. Déploiement sur machine virtuelle

a) Configuration de la Collecte sur la VM

On transfere notre prgramme python mqtt qui envoie les données à la base de données sur notre machine virtuel linux.

On installe les paquets python et mysql afin de pouvoir run le script mqtt dans la console linux sans probleme et on oublie de changer l'adresse Ip dans le fichier pour mettre l'adresse ip du pc dans laquelle la base de donhnées est placée.

```
try:
    db=_mysql.connect("10.129.4.155","tata","tata", "groupe1B")
except OperationalError:
    db=_mysql.connect("10.129.4.155","tata","tata")
    db.query("CREATE DATABASE groupe1B")
    db.query("USE groupe1B")
```

Ensjuite on peut run le script :

```
root@serveur:/home/toto/Téléchargements# nano main.py
root@serveur:/home/toto/Téléchargements# python3 main.py
Connected with result code 0
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1187, '2022-06-24 1
4:02:19', 26.54)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1188, '2022-06-24 1
4:02:19', 17.78)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1187, '2022-06-24 1
4:02:24', 17.41)
INSERT INTO sensors_data (sensor_id, datetime, temp) VALUES (1188, '2022-06-24 1
4:02:24', 24.42)
```

b) Configuration du serveur et host sur la VM

On commence par installer une nouvelle machine virtuel linux dans laquelle on va permettre l'host de notre site web. Dans la console on installe les paquets suivants.

apt-get install python3-pip apache2 libapache2-mod-wsgi-py3
apt-get install default-mysql-server
apt-get install default-mysql-client

Afin transférer notre projet sur la machine linux, on fait un git clone de notre projet sur GitHub.

On met en place un environnement virtuel dans le dossier dans lequel on a mis notre projet afin de pouvoir installer les paquets nécessaires des programmes python.

```
root@serveur:/home/toto/django-app# virtualenv django-appenv
created virtual environment CPython3.9.2.final.0-32 in 754ms
  creator CPython3Posix(dest=/home/toto/django-app/django-appenv, clear=False, n
o_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle
, via=copy, app_data_dir=/root/.local/share/virtualenv)
    added seed packages: Django==4.0.5, asgiref==3.5.2, mysqlclient==2.1.1, pip=
=22.1.2, setuptools==62.3.4, sqlparse==0.4.2, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerS
hellActivator,PythonActivator
root@serveur:/home/toto/django-app# source /home/toto/django-app/django-appenv/b
in/activate
(django-appenv) root@serveur:/home/toto/django-app#
```

A l'intérieur de celui-ci on se rend dans le fichier settings.py afin de changer quelques paramètres. Premièrement les adresses IP qu'on permet à accéder aux site web du serveur.

```
ALLOWED_HOSTS = ['127.0.0.1',
                 '10.129.4.161', '10.129.4.157']
```

Ensuite on configure la connexion à la base de données, en mettant en place l'adresse IP et les identifiants.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'groupe1b',
        'USER': 'tata',
        'HOST': '172.11.30.2',
        'PASSWORD': 'tata',
    }
}
```

On installe le paquet ufw afin d'autoriser les connexons au port 80.

```
ufw allow 80/tcp
```

Désormais on peut lancer notre serveur dans la machine virtuelle dans toutes circonstances avec la commande « python manage.py runserver 0.0.0.0 :8000 » la machine nous affiche notre site.

```
(django-appenv) root@serveur:/home/toto/django-app# ./manage.py runserver 0.0.0.
0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
```

On finit par configurer le fichier 000_default.conf ce script qui stipule nos différents dossier et fichiers

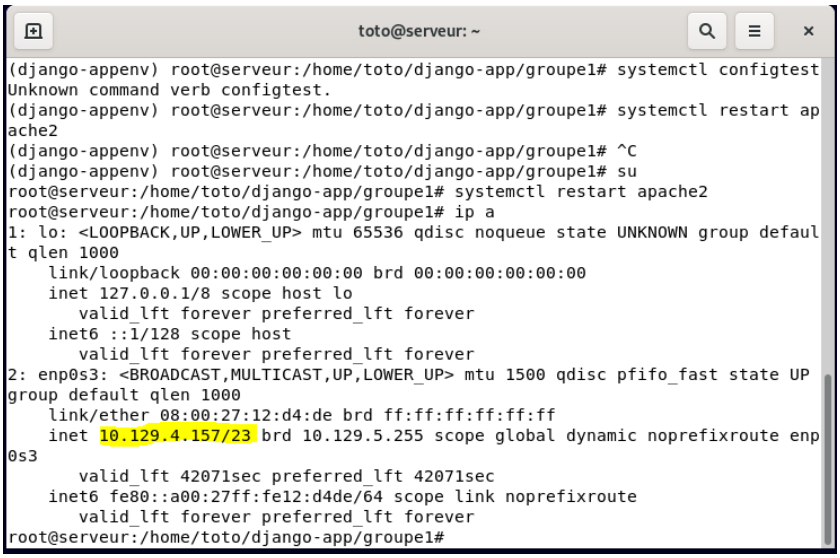
```
root@serveur:/home/toto/django-app/groupe1# nano /etc/apache2/sites-available/00
0-default.conf
```

```
    Alias /static /home/toto/django-app/static
    <Directory /home/toto/django-app/static>
        Require all granted
    </Directory>
    <Directory /home/toto/django-app/groupe1>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>
    WSGIDaemonProcess django-app python-home=/home/toto/django-app/django-appen>
    WSGIProcessGroup django-app
    WSGIScriptAlias / /home/toto/django-app/groupe1/wsgi.py

</VirtualHost>
```

sudo apache2ctl configtest
sudo systemctl restart apache2

On verfie le fichier s'il est bien correcte et on relance notre system apache2 et on arrive a se connecter a notre site en mettant l'adresse ip de la vm serveur.



Liste des données Liste des capteurs