
RAPPORT SAE 23 – Partie Web

WAGNER Nicolas, LOSSER Julien et DESCAMPS Nathan

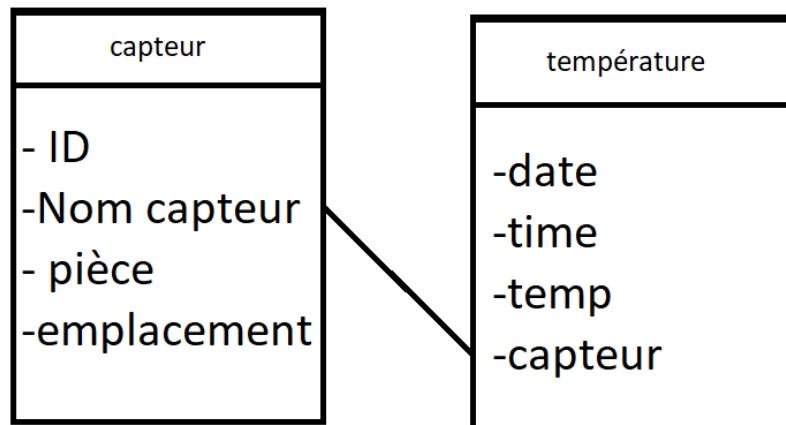
Table des Matières

Table des Matières	1
1) Introduction	2
2) Création Django et CRUDs.....	2
a) Form.py.....	2
b) Urls.py.....	3
c) Views.py.....	3
3) Mise en place databases mysql	4
a) Installation des paquets	4
b) Mise en place de la database	5
c) Exportation de la database vers django	5
d) Résultats dans le prompt MySQL	5
4) Aspect graphique	6
a) HTML	6
b) CSS	7

1) Introduction

Dans ce projet, nous devons mettre en place une base de données avec MySQL sous Windows possédant 2 tables (ID, nom du capteur, pièce et emplacement du capteur). Nous devons ensuite migrer notre projet sous Ubuntu. Nous finirons par afficher les données avec un filtre.

Tel que :



2) Création Django et CRUDs

Nous avons créé à l'aide de pycharm des Crud avec le paquet Django, nous avons suivi le cours vu dans les ressources web dynamique précédentes notamment la ressource R209.

a) Form.py

```
from django.forms import ModelForm 1
from . import models

class CapteursForm(ModelForm): 2
    class Meta:
        model = models.Capteurs
        fields = ('nom', 'piece', 'mac')
        labels = {
            'nom': ('Nom du capteur'), 3
            'piece': ('Pièce du capteur'),
            'mac': ('Adresse MAC du capteur'),
        }
```

1 : On importe de ModelForm la `django.forms` ce qui nous permet d'appliquer une première structure à notre form.

2 : On crée la classe `CapteursForm` dans laquelle on va définir les différents champs et l'affichage que l'on va leur attribuer. Les champs et les labels.

3 : On crée 3 labels nom, pièce et mac qui renverront le nom du capteur, la pièce où il se trouve et l'adresse mac de celui-ci.

b) Urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.index),
    path('accueil/update/<int:id>', views.update),
    path('updatetraitement/<int:id>', views.updatetraitement),
    path('filter/', views.filter),
    path('export/', views.export),
    path('graph/', views.graph),
    path('image/', views.image),
    path('page/', views.page),
]
```

1 : Dans la première partie on peut importer `path` de `django.urls` pour créer les différents chemins, on y importe également notre `views`.

2 : On crée nos chemins courants à partir de nos fonctions dans les `views`, soit `l'update`, `l'index`, `le traitement`, etc... On notera un système d'ID qui permet de repérer les valeurs dans les pages correspondantes.

c) Views.py

```
def accueil(request):
    temp = models.Temperatures.objects.all()
    capteurs = models.Capteurs.objects.all()
    return render(request, "accueil.html", {"data": capteurs, "temp": temp})
```

Ici on a la fonction `accueil` qui retourne l'accueil, avec les valeurs des températures et des capteurs qui sont redirigés vers `accueil.html`.

```
def update(request, id):  
    content = models.Capteurs.objects.get(pk=id)  
    form = CapteursForm()  
    form.fields['mac'].widget.attrs['readonly'] = True  
    form.fields['mac'].initial = content.mac  
    return render(request, "update.html", {"form": form, "id": id})
```

Ici on a la fonction d'update avec la form qui est définie sur la form du capteur et donc qui renvoie l'id et la form définie plus tôt dans le rapport sur « CapteursForm ».

3) Mise en place databases mysql

a) Installation des paquets

Sur le terminal pycharm on installe le paquet MySQL client :

```
pip install mysqlclient
```

Cela nous permet de lier la base de données réalisée avec MySQL command prompt sur windows, là où l'on crée notre database Mysql.

Au préalable pour récupérer l'application MySQL command prompt on doit aller sur internet et installer Mysql



Pour cela on suit l'installation et on choisit l'option « full installation », une fois cela fait on a accès à notre prompt Mysql :


```
Enter password: *****  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 9  
Server version: 8.0.28 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2022, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql>
```

On rentre le mot de passe initialisé plus tôt sous le nom de « frangipane »

b) Mise en place de la database

```
create database adminserver;  
use adminserver  
CREATE USER "adminserver"@"localhost" IDENTIFIED BY "toto";  
GRANT ALL PRIVILEGES ON *.* TO "adminserver"@"localhost";  
FLUSH PRIVILEGES;
```



Ici sur MySQL prompt  On crée la base de données avec les commandes précédentes et on se donne les privilèges sur un utilisateur crée précédemment.

c) Exportation de la database vers django

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'sae24',  
        'USER': 'root',  
        'PASSWORD': 'frangipane',  
        'HOST': 'localhost',  
        'PORT': '3306',  
        'OPTIONS': {  
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"  
        }  
    }  
}
```

Ici on a lié la base de données databases sae24 crée précédemment sur MySQL avec le nom d'user 'root', le mot de passe initialisé sur 'frangipane' et le localhost sur le localhost crée précédemment dans MySQL :

```
CREATE USER "adminserver"@"localhost"
```

Et on y entre le port 3306.

d) Résultats dans le prompt MySQL

Dans le prompt MySQL on voit qu'après avoir fait un show TABLES on a la liste des tables qui

```
mysql> SHOW TABLES;
+-----+
| Tables_in_sae24 |
+-----+
| 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   |
| recuperation_temp_app_capteurs |
| recuperation_temp_app_temperatures |
+-----+
```

apparaît :

Et avec la commande `SELECT * FROM recuperation_temp_app_temperatures` on récupère les données listées dans un tableau en ASCII des différentes données récupéré par le broker et Mqtt.

```
mysql> SELECT * FROM recuperation_temp_app_temperatures
-> ;
+----+-----+-----+-----+-----+
| id | date       | time           | temp    | capteur_id |
+----+-----+-----+-----+-----+
| 1  | 2022-06-23 | 13:24:02.000000 | 24.25   | 1           |
| 2  | 2022-06-23 | 13:24:02.000000 | 19.65   | 2           |
| 3  | 2022-06-23 | 13:24:07.000000 | 5.29    | 1           |
| 4  | 2022-06-23 | 13:24:07.000000 | 2.94    | 2           |
| 5  | 2022-06-23 | 13:24:12.000000 | 0.58    | 1           |
| 6  | 2022-06-23 | 13:24:12.000000 | 17.09   | 2           |
+----+-----+-----+-----+-----+
```

4) Aspect graphique

Lors de la partie graphique nous avons utilisé pycharm pour la création d'un site qui permet de visualisé les valeurs et de voir les différentes températures sur les deux capteurs.

a) HTML

Ici un exemple de notre fichier accueil.html qui es la page principale de notre projet où sont affichés les valeurs remarquables :

b) CSS

```
c) body {
    background: #222;
    font-family: Raleway, Helvetica, sans-serif;
    position: relative;
}

/*Navigation */

/* Position */
.side-nav {
    position: fixed;
    height: 100vh;
    left: 0;
    top: 0;
}

/* Pour masquer la sous-navigation */
.wrapper {
    background: #333;
    height: 100vh;
    width: 75px;
}

.nav-bloc {
    padding: 20px 0;
    display: flex;
    justify-content: center;
    cursor: pointer;
    border-bottom: 1px solid #f2f2f2le;
```

```
}  
.nav-bloc:hover {  
  background: rgb(106, 146, 196);  
}  
.nav-bloc:hover .sub-nav {  
  transform: translateX(75px);  
}  
.nav-bloc img {  
  width: 45px;  
  transition: 0.2s ease-in-out ;  
}  
  
.sub-nav {  
  padding: 0px;  
  width: 200px;  
  height: 100vh;  
  position: absolute;  
  top: 0;  
  left: 0;  
  background: #333;  
  z-index: -1;  
  color: white;
```