Procédure d'installation et de déploiement de l'application «Gestion d'absences»

En quoi consiste le projet ?

Une application web

Créer une application web à l'aide de l'outil Django qui permet la saisie d'absences d'élèves dans une base de données. Il faut également pouvoir importer un fichier avec plusieurs absences qui les implémentera directement dans la base de données.

Gestion de la base de données

La base de données doit être hébergée sur un autre serveur qui sera dédié à cela, nous avons choisi une machine virtuelle de Windows 10 et l'outil MySQL.

Travail collaboratif

Tout ce travail devant être réalisé en groupe, il faut donc également mettre en place des outils de travail collaboratif. Github nous permet de travailler sur le code à plusieurs. Nous avons aussi crée un diagram de Gantt en ligne sous Google Sheet pour y accéder à plusieurs.

Table des matières

Création de l'application

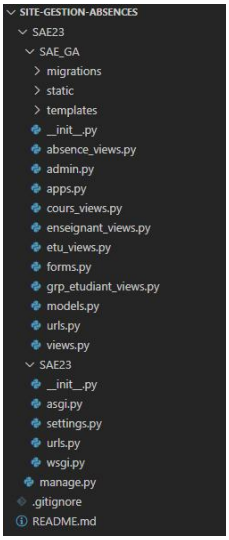
Nous devions donc créer un projet Django avec le logiciel de programmation au choix, j'ai utilisé Visual Studio Code.

Afin de commencer le projet, il est recommandé de créer un environnement virtuel pour ce dernier avec à l'intérieur tous les modules nécessaires au projet :

```
py -3 -m venv .venv
.\venv\Scripts\Activate.ps1
py -m pip install --upgrade pip
py -m pip install django
py -m pip install pillow
```

Architecture du projet :

Nous avons donc notre dossier de projet qui contient les «static» qui sont nos images et feuilles de style CSS. Le dossier template contenant toutes les pages du site ainsi qu'une page principale appelé le «main» qui hérite sa structure aux autres. Chaque donnée du site a plusieurs informations. Pour les créer et modifier il nous fallait créer ce que l'on appel des «views» et des routes pour celles-ci. Chaque instance avait son propre CRUD : Groupe, Etudiants, Enseignants, Cours et Absence.



Modèle des absences sur lequel va se baser le formulaire :

```python
class Absence(models.Model):
    #etudiant=models.CharField(max_length=100)
    etudiant=models.ForeignKey("etudiant", on_delete=models.CASCADE, default=None)
    #cours=models.CharField(max_length=100)
    cours=models.ForeignKey("cours", on_delete=models.CASCADE, default=None)
    accepte=models.BooleanField()
    justification=models.TextField(null = True, blank = False)

    def __str__(self):
        if self.accepte == True:
            chaine = f"Absence de {self.etudiant}  au cours de {self.cours}. Absence acceptée."
        else :
            chaine = f"Absence de {self.etudiant}  au cours de {self.cours}. Absence refusée car justification non valable : {self.justification}"
        return chaine
```

La vue va donc créer un formulaire d'ajout à l'aide d'un autre fichier encore pour ajouter ou modifier les informations de l'instance au bon format. Les vues appellent en suite des routes pour afficher les pages et les routes, elles donnent le chemin de ces dernières dans l'arborescence.

Modèle de formulaire :

```python
class AbsenceForm(ModelForm):
    class Meta:
        model = models.Absence
        fields = ('etudiant', 'cours', 'accepte', 'justification')
        labels = {
            'etudiant' : _('Etudiant'),
            'cours' : _('Cours'),
            'accepte' : _('L\'absence est-elle acceptée :'),
            'justification' : _('Justification')
        }
```

```html
{% extends "SAE_GA/main.html" %}

{% block title %} Formulaire de saisie d'une absence{% endblock %} {% block content %}
<article>
        <h1>Saisie d'un nouvelle absense :</h1>
        {% if id is None %}
                <form action="traitement" method="post">
        {% else %}
                <form method="post" action="/SAE_GA/Absence/updatetraitement/{{id}}/">
        {% endif %}
        {% csrf_token %}
        {{form.as_ul}}
        <input type="submit" value="envoyer les données de l'absence">
                </form>
        <footer class="crud"><a href="/SAE_GA/Absence/accueil">Liste des absences</a></footer>
</article>

{% endblock %}
```

Les pages html contiennent du Python pour interagir avec nos vues justement. Ici on regarde si l'identifiant de l'instance existe déjà pour utiliser la vue traitement ou updatetraitement qui va modifier une instance selon son id au lieu d'en créer une.

Installation de l'application sous Linux

Installation d'Apache2

Commencer par installer le service apache2 et mod_wsgi dont on aura besoin : **apt install apache2 libapache2-mod-wsgi-py3**

Créer la bonne arborescence de fichiers :

/projetdjango/
|--- dosser-du-site
| |-- logs
| |-- public
| |-- media
| |-- static
|--- src
| |-- fichiers-de-l'app-django
|--- venv

Clone du projet Github

Le projet étant hébergé sous Github, il nécessite d'être téléchargé sur le poste où l'on veut lancer l'application. La méthode que je conseille est celle de cloner directement le dossier du projet avec l'outil git. Ce dernier s'installe via la commande : **apt install git**

Une fois git téléchargé, vous pouvez vous rendre dans le dossier destination et coller la commande suivante :
git clone https://github.com/FritschyMatheo/Site-Gestion-Absences

```
toto@debian:~/Documents/PROJET$ git clone https://github.com/FritschyMatheo/Site-Gestion-Absences
```

Les fichiers sont en suite automatiquement copié dans votre dossier.

Il faut installer python et Django avant de continuer avec : **apt install python3 python3-django**
Ici nous travaillons a nouveau dans un environnement virtuel (**python3 -m venv venv**) que nous pouvons activer avec : **source venv/bin/activate** pour bien toujours avoir nos dépendances qui sont comme vu page 2 : Django, pip et pillow.

Configuration d'Apache2

Fichier 000-default.conf

Il faut à présent modifier le fichier **/etc/apache2/sites-available/000-default.conf** avec la commande :

nano /etc/apache2/sites-available/000-default.conf

Éditer le fichier pour qu'il contienne ceci :

```
 GNU nano 7.2                                                    /etc/apache2/sites-available/000-default.conf *
<VirtualHost *:80>

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog /home/toto/Documents/Site-Gestion-Absences/site/logs/error.log
        CustomLog /home/toto/Documents/Site-Gestion-Absences/site/logs/access.log combined

        Alias /static /home/toto/Documents/Site-Gestion-Absences/site/public/static
        <Directory /home/toto/Documents/Site-Gestion-Absences/site/public/static>
            Require all granted
        </Directory>

        Alias /media /home/toto/Documents/Site-Gestion-Absences/site/public/media
        <Directory /home/toto/Documents/Site-Gestion-Absences/site/public/media>
            Require all granted
        </Directory>


        <Directory /home/toto/Documents/Site-Gestion-Absences/SAE23>
        <Files wsgi.py>
                Require all granted
            </Files>
        </Directory>

        WSGIDaemonProcess Site-Gestion-Absences python-home=/home/toto/Documents/Site-Gestion-Absences/env python-path=/home/toto/documents/Site-Gestion-Absences/SAE23
        WSGIProcessGroup Site-Gestion-Absences
        WSGIScriptAlias / /home/toto/Documents/Site-Gestion-Absences/SAE23/SAE23/wsgi.py

</VirtualHost>
```

Fichier settings.py

Modifier avec nano le fichier /Site-Gestion-Absences/SAE23/SAE23/settings.py comme ceci :

Indiquer où sont les images et le CSS :

STATIC_URL = 'static/'

STATIC_ROOT = '/Site-Gestion-Absences/site/public/static'

MEDIA_ROOT = '/Site-Gestion-Absences/site/public/media'

Il faut ensuite faire : **python manage.py collectstatic** pour collecter ces fichiers static.

Pour finir, redémarrer le service Apache2 avec la commande :

systemctl restart apache2

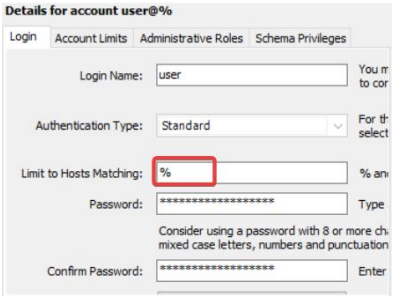
Base de donnée SQL

Nous avons donc utilisé l'outil de gestion de bases de données MySQL, voici comment procéder pour créer la base de donnée et configurer les accès.

Installation

Lien d'installation de MySQL pour Windows : https://dev.mysql.com/downloads/installer/

Lors de l'installation, créez un utilisateur tel que «user», mettez lui un mot de passe et un % à Host Matching :



Création de la base de données

Dans MySQL Command Line Client, connectez vous et faites : **create database sae;**
On pourra les voir avec une fois que l'on aura migré la base : **use sae; show tables;**

Connexion avec le serveur Linux

Installer les paquets requis : **apt install python3-dev** et **pip install mysqlclient**
Il faut à nouveau modifier settings.py pour lui dire où se connecter et avec quels identifiant pour la base de données : **nano SAE23/SAE23/settings.py**
Dans la partie DATABASE, remplir ces informations :

Une fois le fichier enregistré, il ne reste plus qu'à migrer la base de données depuis le Linux vers le Windows avec la commande : **python3 manage.py migrate**

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'nom de la base de données',
        'USER': 'user',
        'PASSWORD': 'password',
        'HOST': '@ip de la machine Windows',
        'PORT': '3306' #Port de MySQL
    }
}
```

Conclusion

Avec cette procédure, vous pouvez donc déployer l'application de gestion d'absence sur un serveur Linux avec la base de donnée MySQL sur un serveur Windows à part.