SAE 2.04 – Mettre en place une solution informatique pour l'entreprise

Fiche de procédure

GARRONE Gabriel - RT122

2022-2023

1) Objectif du projet

Le projet consiste à développer un site web permettant la gestion des absences. L'objectif principal est de permettre aux gestionnaires de saisir les absences pour différents cours et d'ajouter des justifications par la suite. Le schéma de données comprend les entités suivantes :

- Des groupes d'étudiants (id, nom)
- Des étudiants (id, nom, prénom, email, groupe, photo)
- Des enseignants (id, nom, prénom, email)
- Des cours (id, titre du cours, date, enseignant, durée, groupe)
- Des absences aux cours (étudiants, cours, justifié ou non, justification)

Pour ce projet, nous devrons mettre en place un CRUD (Create, Read, Update, Delete) pour chaque type de données mentionné ci-dessus. Nous devrons préparer et remplir la base de données avec des groupes, des étudiants et des enseignants.

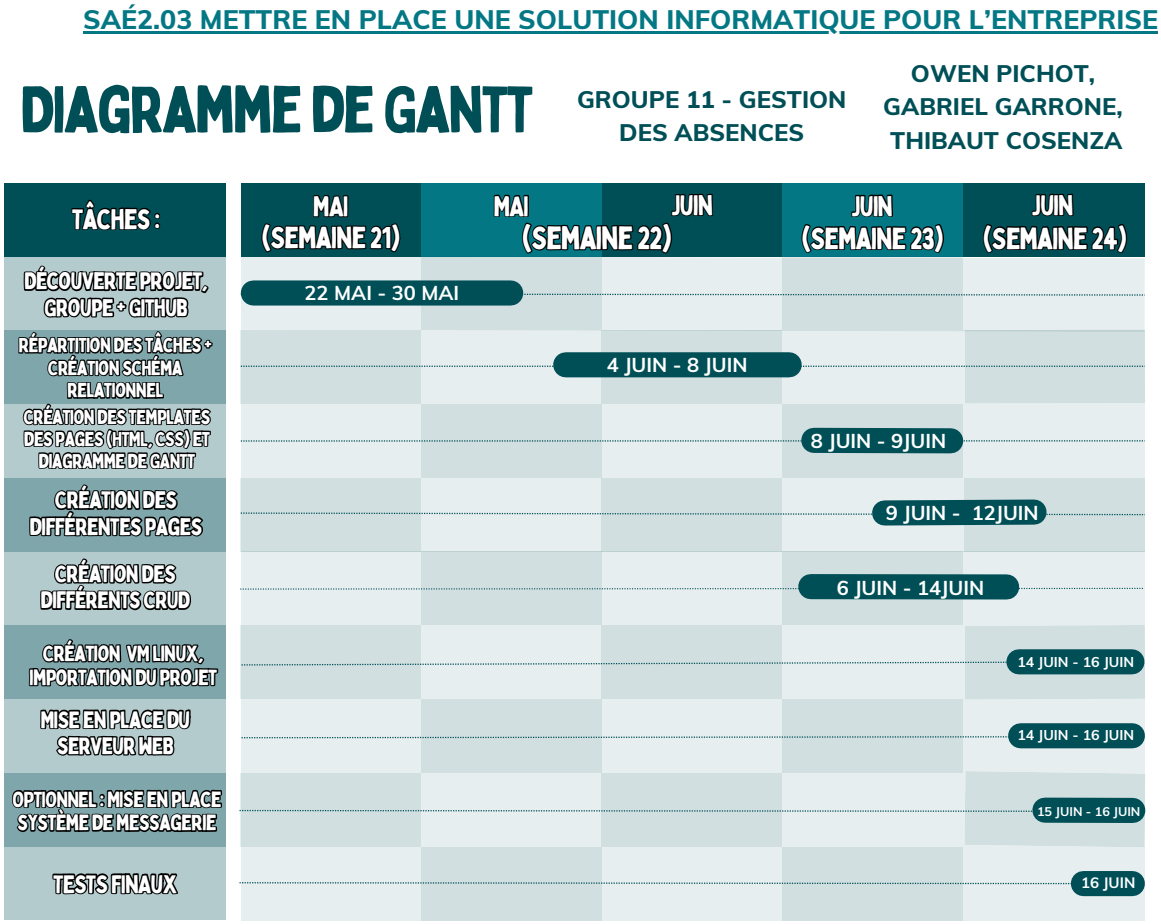
Notre site web devra permettre la saisie de nouveaux cours et l'enregistrement des absences pour ces cours. Nous devrons également pouvoir valider une absence et ajouter des justificatifs. Une fonctionnalité importante sera la possibilité d'importer les absences d'un cours à partir d'un fichier dont la structure devra être clairement définie.

Il sera nécessaire de pouvoir générer une fiche récapitulative des absences pour un cours donné. De plus, nous devrons pouvoir générer une liste des absences d'un étudiant avec le total des absences justifiées et non justifiées.

En résumé, notre site web de gestion des absences devra permettre la saisie, la modification, la suppression et la consultation des données relatives aux groupes, étudiants, enseignants, cours et absences. Via un sytème de connexion, les enseignants et les étudiants pourront interagir indépendamment sur les absences.

2) Diagramme de Gant et Planning

L'objectif principal de la création d'un diagramme de Gantt pour ce projet était de planifier et organiser les différentes tâches et activités à réaliser, en établissant une séquence chronologique et en déterminant les dépendances entre elles, afin d'assurer une gestion efficace du temps et des ressources. Cela nous permettait de visualiser et suivre le déroulement du projet, d'identifier les éventuels retards ou problèmes, et de prendre des mesures correctives en conséquence.



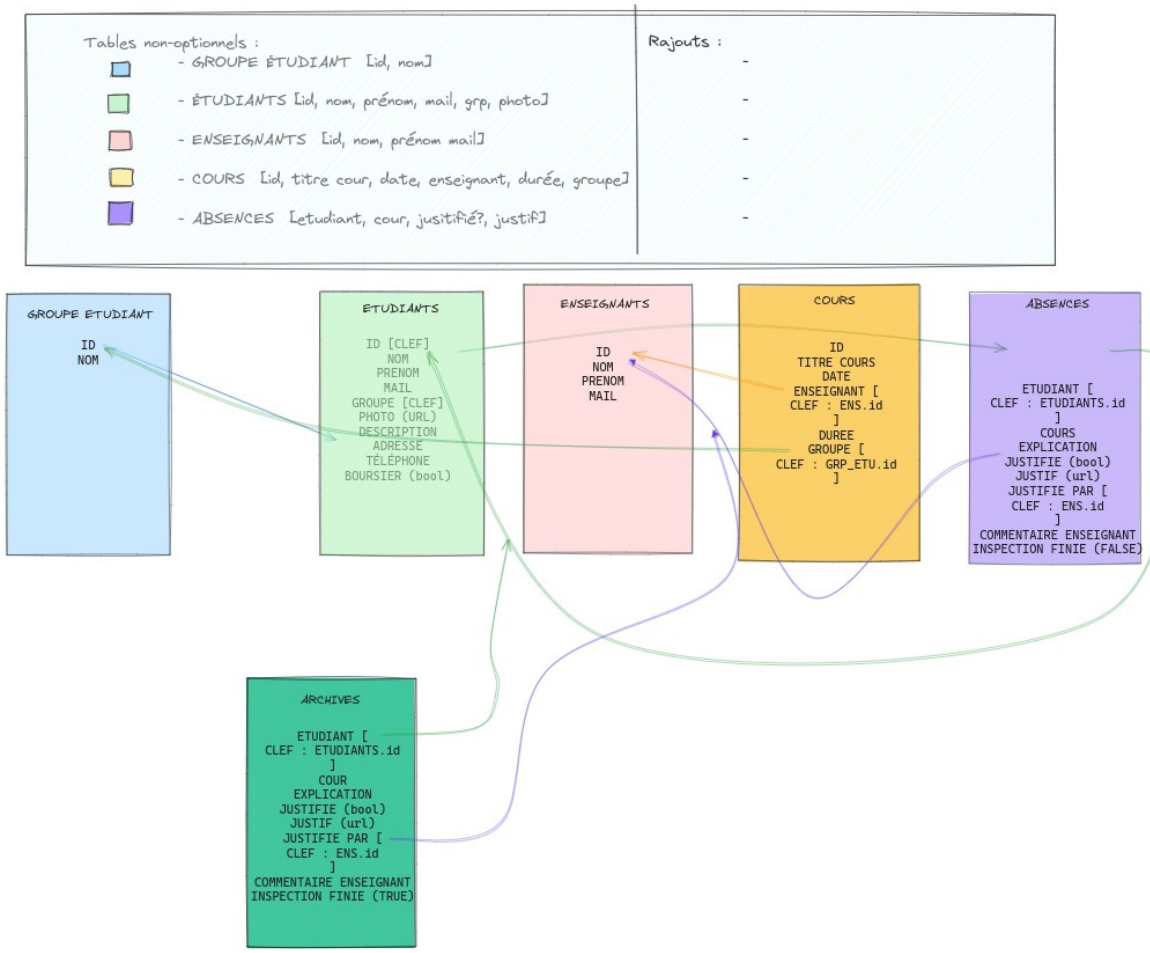
Ici une représentation nette de notre répartition des tâches, qui a évoluée au long du projet.

3) Schéma relationnel du projet

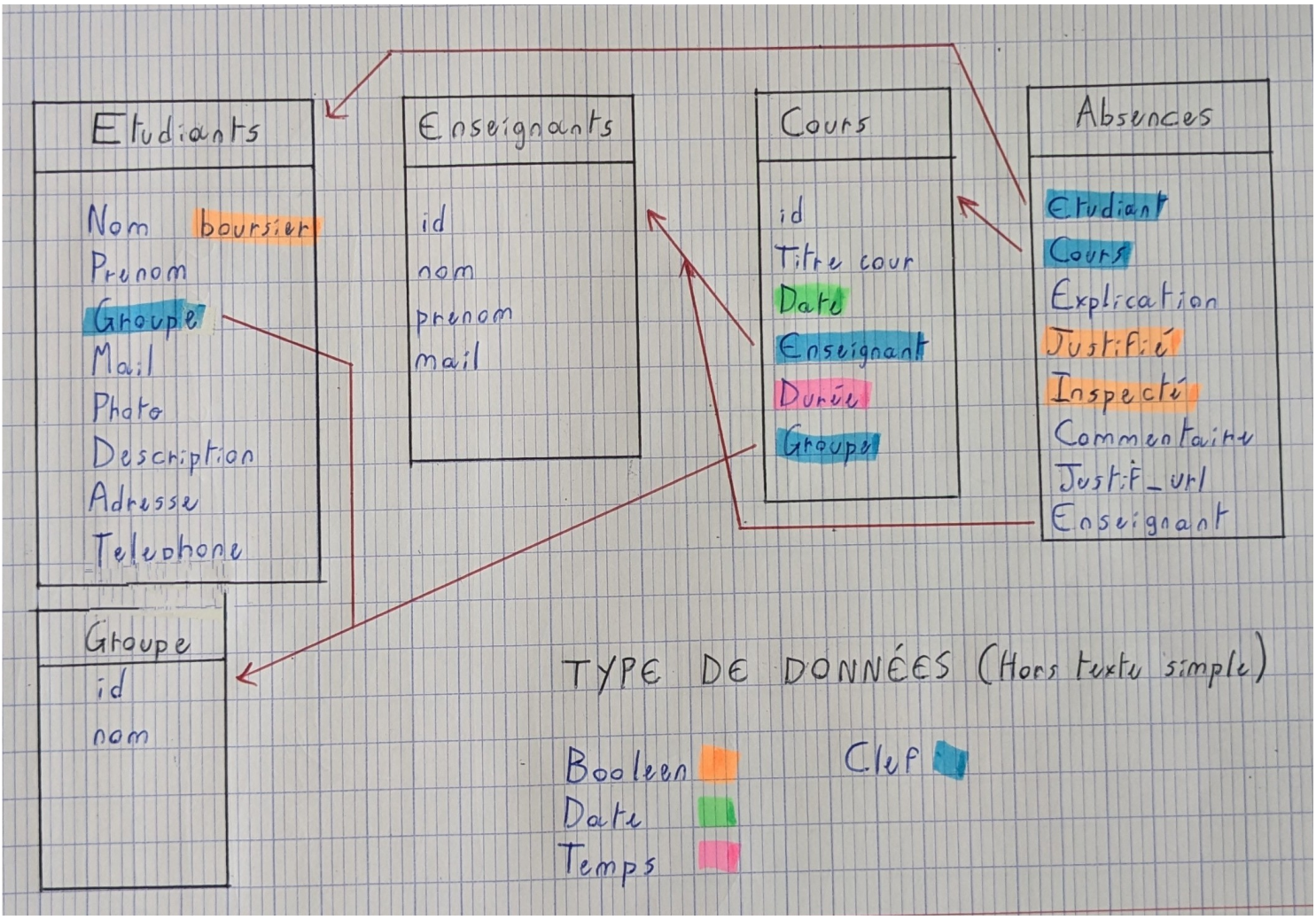
L'objectif principal de la création d'un schéma relationnel pour notre projet de base de données était de définir la structure des tables et les relations entre elles, afin de garantir l'intégrité des données et la cohérence de l'ensemble du système. Cela nous permettait également de représenter de manière claire et compréhensible les entités, les attributs et les contraintes, facilitant ainsi la conception, la maintenance et la manipulation des données.

Voici la première version faites sur Excalidraw. (voir le schéma entier)
https://excalidraw.com/#room=22fb39ca8e7b237d88e9,t2-EOKdqlGOkowkR7Qwt3A

Nous avions rajouter la table Archives qui n'a finalement pas été implantée.



Voici la version finale du schéma relationnel, que nous avons utilisé comme base pour créé toute la base de données, avec les relations qui sont très importantes.



4) Création du projet Django

L'une des étapes les plus importantes du projet a été la création d'un site web attrayant en HTML et CSS, associé à une gestion en Python à l'aide de Django pour notre base de données. Pour accomplir cela, il a été nécessaire de mettre en place un projet Django en suivant les commandes appropriées suivantes :

• *pip install django*
permet installer django sur notre pc

• *django-admin startproject nom_du_projet*
permet de créer notre projet django

• *python manage.py startapp nom_application*
permet de créer l'application

• *'nom_app.apps.Nom_appConfig'*
information à ajouter dans settings.py dans INSTALLED_APPS (projet)

• *'python manage.py runserver'*
permet de lancer le seveur

• *'python manage.py makemigrations' & 'python manage.py migrate'*
permet d'effectuer les migrations lorsque l'on a effectué une modification dans les models

Ainsi, nous avons pu amorcer notre projet Django, et comme mentionné précédemment, des détails plus approfondis sont disponibles sur mon portfolio.

Notre projet a été intégralement réalisé sur Ubuntu sur Visual Studio Code.

5) Base de données MySQL

Pour commencer, il est nécessaire de préciser que Owen Pichot s'est occupé de créer la base de données et les tables au niveau de MySQL, pour autant les captures d'écran sont issus de mon système Ubuntu, puisque en important le projet sur mon PC, j'ai du réaliser ces manipulations moi aussi pour accéder à la BdD.

Pour créer la base de données, il est nécessaire d'installer mysql et les paquets qui en dépendent. Tout d'abord on lance mysql

```
frigiel@fluffy:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2023, 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.
```

D'abord on accède à la base de données

```
mysql> show DATABASES;
+--------------------+
| Database           |
+--------------------+
| DBABSENCES         |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0,00 sec)
```

On remarque qu'il y a une base de données « DBABSENCES »
Cette base de données a été créé dans le fichier settings.py de notre projet, avec quelques informations sur lesquels nous reviendront.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql'
        'NAME': 'DBABSENCES',
        'USER': 'gab',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '3306',
        'OPTIONS': {
            'autocommit': True,
        },
    }
}
```

Ici on peut voir toutes les tables créées, ainsi que celles mises en place par Django.

```
mysql> use DBABSENCES;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------------+
| Tables_in_DBABSENCES       |
+----------------------------+
| ABSENCES_absences          |
| ABSENCES_archives          |
| ABSENCES_cours             |
| ABSENCES_cours_Groupes     |
| ABSENCES_enseignant        |
| ABSENCES_etudiants         |
| ABSENCES_groupeetu         |
| 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             |
+----------------------------+
17 rows in set (0,00 sec)
```

Ensuite je crée un utilisateur possédant tous les droits, qui pourra ce connecter grâce à ses privilèges administrateurs.

```
mysql> create user 'gab'@'localhost';
Query OK, 0 rows affected (0,01 sec)
```

```
mysql> GRANT ALL PRIVILEGES on DBABSENCES.* to 'gab'@'localhost';
Query OK, 0 rows affected (0,01 sec)
```

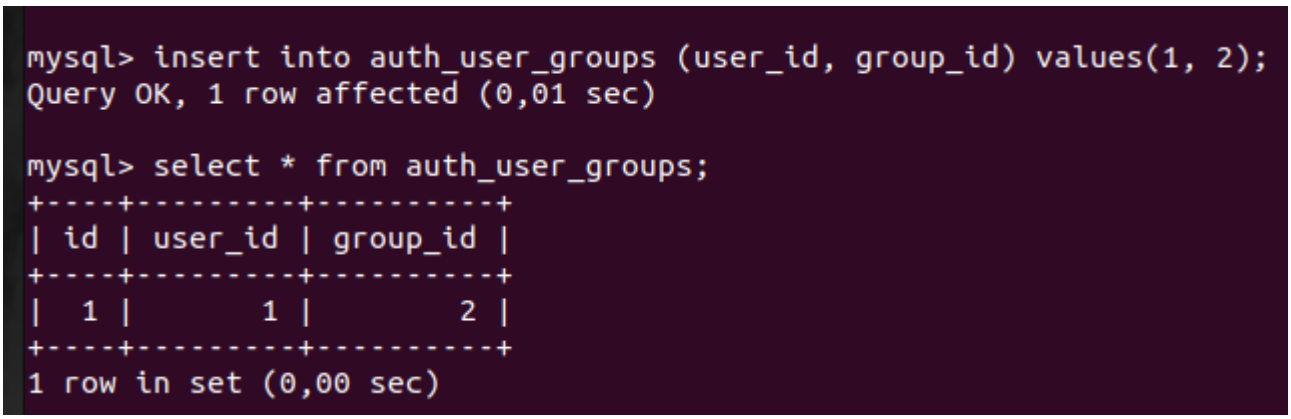
Pour autant, il sera nécessaire de créer un utilisateur « enseignant » et un utilisateur « étudiant » pour les biens du projet, qu'il sera nécessaire de relier aux différentes tables suivant leur relations.

D'abord je vais créer le groupe « ETUDIANTS » et le groupe « ENSEIGNANTS » qui auront pour id 1 et 2. L'id est très important car il permettra de relier les utilisateurs à ces nouveaux groupes.

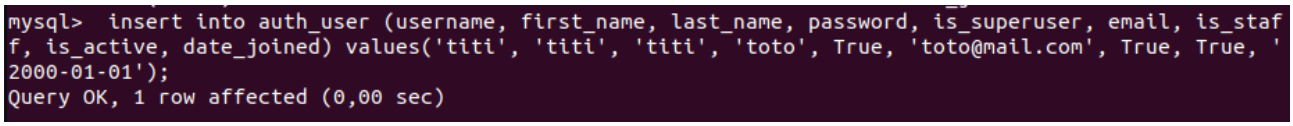
```
mysql> insert into auth_group (name) values('ETUDIANTS');
Query OK, 1 row affected (0,01 sec)

mysql> select * from auth_group;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | ETUDIANTS |
+----+-----------+
1 row in set (0,00 sec)

mysql> insert into auth_group (name) values('ENSEIGNANTS');
Query OK, 1 row affected (0,03 sec)

mysql> select * from auth_group;
+----+-------------+
| id | name        |
+----+-------------+
|  2 | ENSEIGNANTS |
|  1 | ETUDIANTS   |
+----+-------------+
2 rows in set (0,00 sec)
```

Ici on peut voir que le user gabriel que j'ai créé dans le terminal avec la commande

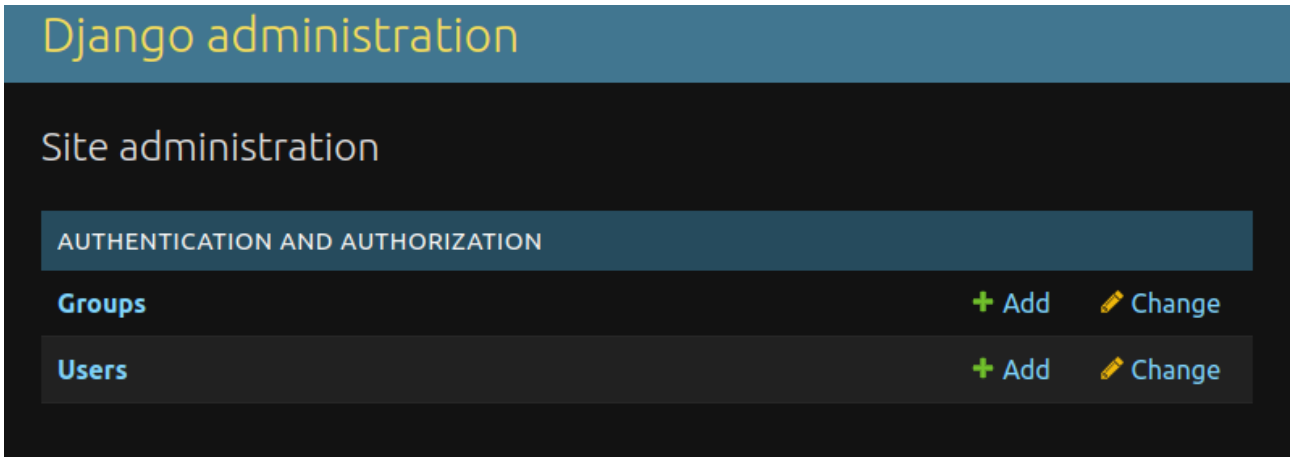
On remarque que ce user a pour id « 1 », et on souhaite qu'il soit enseignant, qui a pour rappel id « 2 ». On va donc faire le lien entre eux de cette manière.



On peut aussi créé directement le user dans la table auth_user comme je l'ai fait avec toto. Chaque champ que j'ai du remplir est obligatoire.



On peut aussi procéder à ces manipulations dans le /admin de django
Pour y accéder je peux me log avec « gabriel » et le mdp « toto »
On pouvait voir ce user précédemment dans la table « auth_user ».



Ceci fait, le réel objectif est en fait de créer des users dans les tables que nous avons créé pour les biens du projet. Ici je crée un enseignant avec les champs que nous avons défini dans la table ABSENCES_enseignant que nous avons déjà créé.

```
mysql> insert into ABSENCES_enseignant (nom, prenom, mail) values('toto', 'toto', 'toto@mail.com');
Query OK, 1 row affected (0,02 sec)

mysql> select * from ABSENCES_enseignant;
+----+------+--------+---------------+
| id | nom  | prenom | mail          |
+----+------+--------+---------------+
|  1 | toto | toto   | toto@mail.com |
+----+------+--------+---------------+
1 row in set (0,00 sec)
```

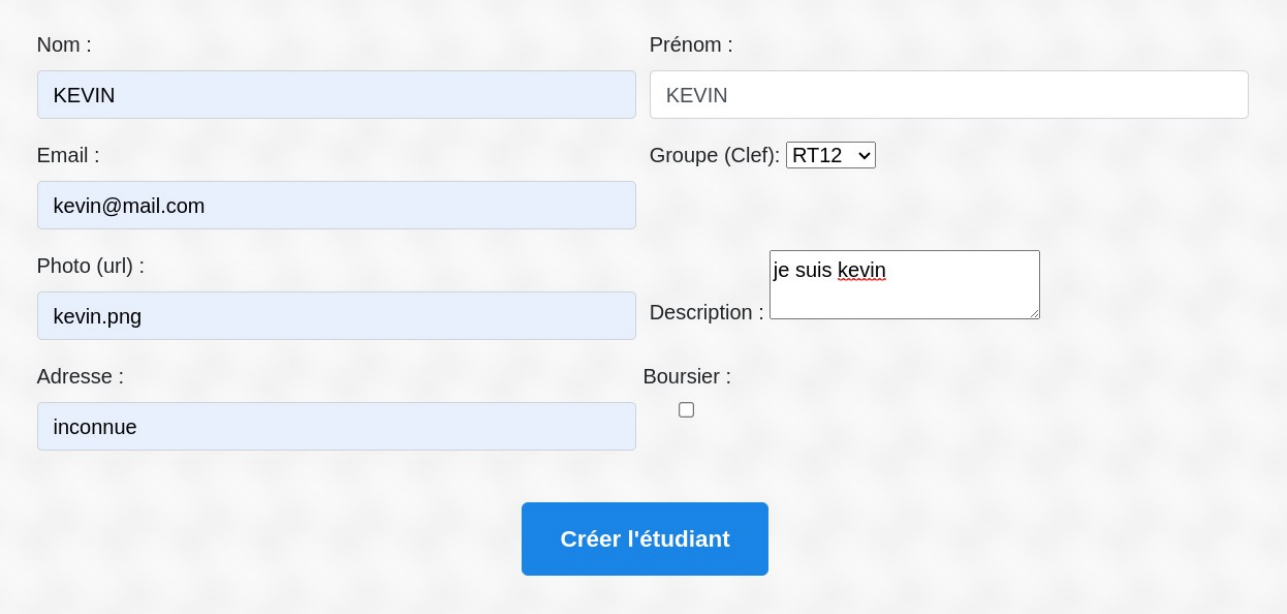
Enfin il sera nécessaire de le relier avec tables groupes et cours, suivant ses relations, manipulation que j'ai pu faire précédemment pour l'utilisateur gabriel avec comme exemple les tables « auth_user_groups » et « auth_user ».

Dans le cas où l'on a créé un utilisateur élève, dans la table ABSENCES_etudiants, il faudra procéder aux même manipulations suivant ses relations propres groupe et cours.

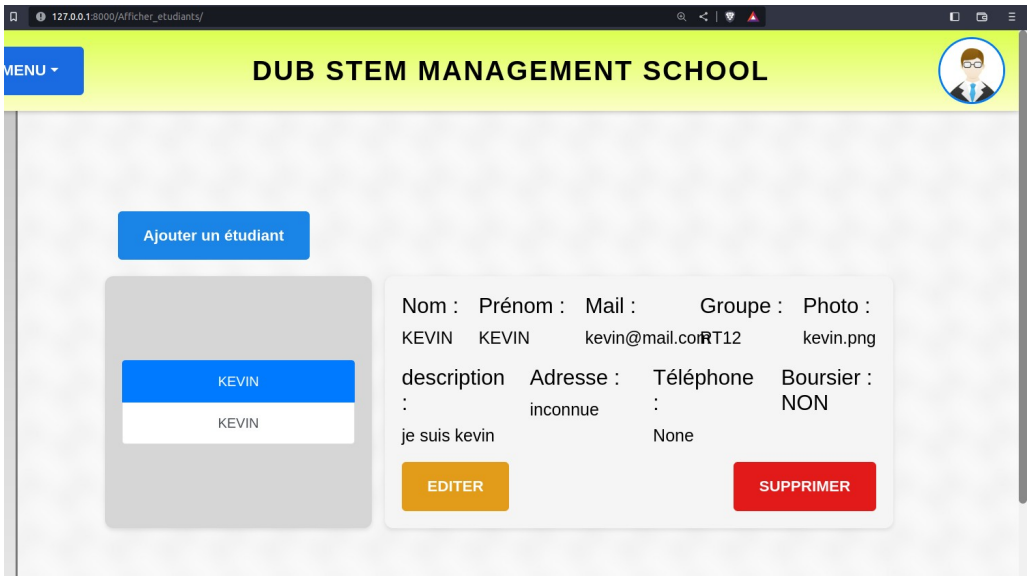
Rappel des tables existantes pour la compréhension :

```
mysql> use DBABSENCES;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------------+
| Tables_in_DBABSENCES      |
+---------------------------+
| ABSENCES_absences         |
| ABSENCES_archives         |
| ABSENCES_cours            |
| ABSENCES_cours_Groupes    |
| ABSENCES_enseignant       |
| ABSENCES_etudiants        |
| ABSENCES_groupeetu        |
| 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            |
+---------------------------+
17 rows in set (0,00 sec)
```

Enfin de manière vraiment applicable, l'idée sera de créer des utilisateurs directement depuis le site Ici je vais créer l'utilisateur Kevin depuis «Ajout_etudiants ».



Ici on voit qu'il a bien été ajouté dans mon onglet affichage :



Enfin on peut observer l'ajout dans la table donc dans la base de données
Je rappelle que ceci a été effectué depuis l'interface accessible depuis les enseignants, qui peuvent donc créé directement les utilisateurs étudiants, enseignants, mais aussi les cours, les groupes etc.

C'est donc de cette manière que n'importe quel utilisateur est possible d'être créé depuis le site.

```
mysql> select * from ABSENCES_etudiants;
+----+-------+--------+----------------+-----------+--------------+----------+-----------+----------
-----------+
| id | nom   | prenom | mail           | photo     | description  | adresse  | telephone | boursier
 groupe_id |
+----+-------+--------+----------------+-----------+--------------+----------+-----------+----------
-----------+
|  1 | KEVIN | KEVIN  | kevin@mail.com | kevin.png | je suis kevin | inconnue |      NULL |        0
         1 |
+----+-------+--------+----------------+-----------+--------------+----------+-----------+----------
```

6) Déploiement

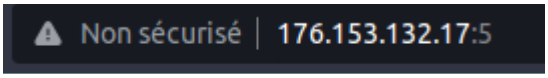
Pour déployer le site localement ou non, il est nécessaire d'aller dans settings.py pour modifier ces 2 objets.

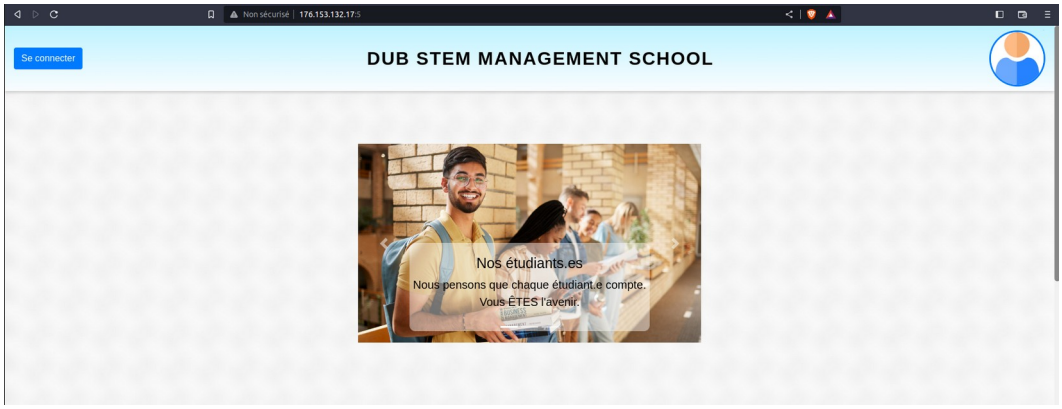
```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'DBABSENCES',
        'USER': 'gabriel',
        'PASSWORD': 'j38fqt',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
            'autocommit': True,
        },
    }
}
```

```python
ALLOWED_HOSTS = [
    '127.0.0.1',
    '192.168.1.10', #this pc (ethernet)
    '192.168.1.1', #this pc (wi-fi)
    '176.153.132.17',
    '192.168.1.106', #mon pc


]
```

Premièrement dans « ALLOWED_HOSTS » on retrouve les adresses possibles d'accès du site. Ici il y a l'adresse de bouclage 127.0.0.1 que nous utilisons en local, les adresses d'Owen Pichot sur ces différents interfaces, et l'indresse de mon PC en wi-fi.
Deuxièmement dans « DATABASES » on retrouve l'hôte que l'on a ici mis en local « localhost », avec la redirection du port.
Pour héberger sur son adresse personnel il est nécessaire de modifier dans les paramètres de sa boxe la redirection du port. Ici pour notre projet nous avons décider de rediriger tout sur le port 5.

Ici accès au site hebergé sur le système ubuntu de Owen.





L'objectif en tant que tel reste de pouvoir ajouter des absences depuis un profil étudiant, puis de pouvoir les accepter ou non pour un enseignant, après l'avoir inspecté en détail.
L'étudiant lui reçoit un retour sur l'évolution de son justificatif d'abense.

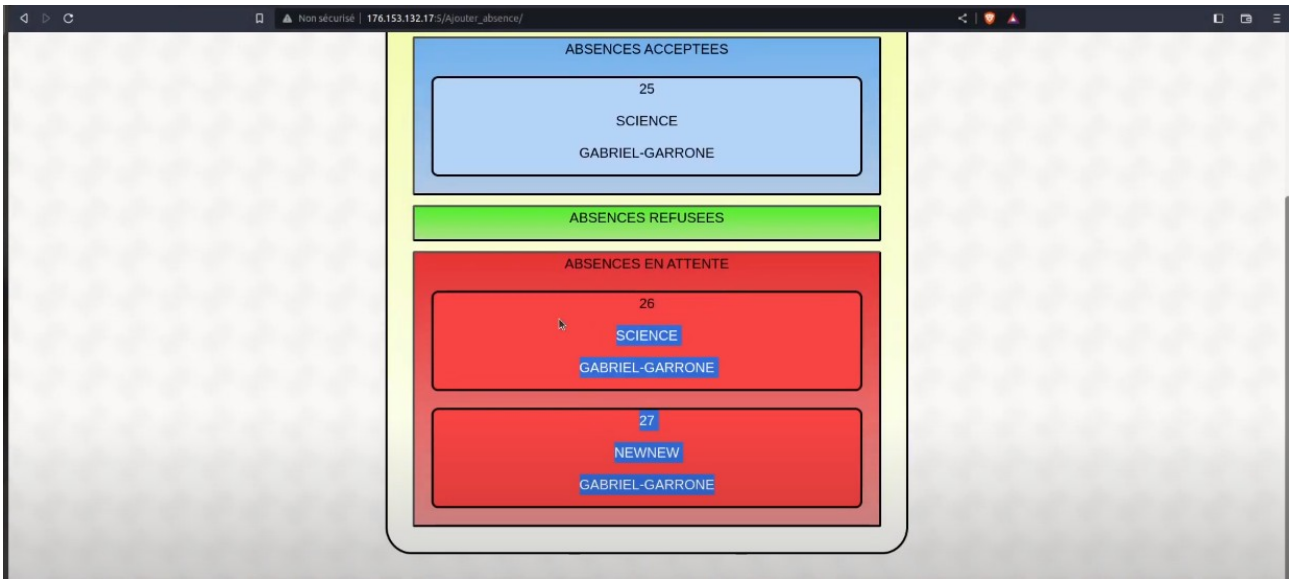
Dans cette courte vidéo présente aussi dans mon portfolio, je montre cette fonctionnalité.

- *https://youtu.be/HonorB5bTv4*

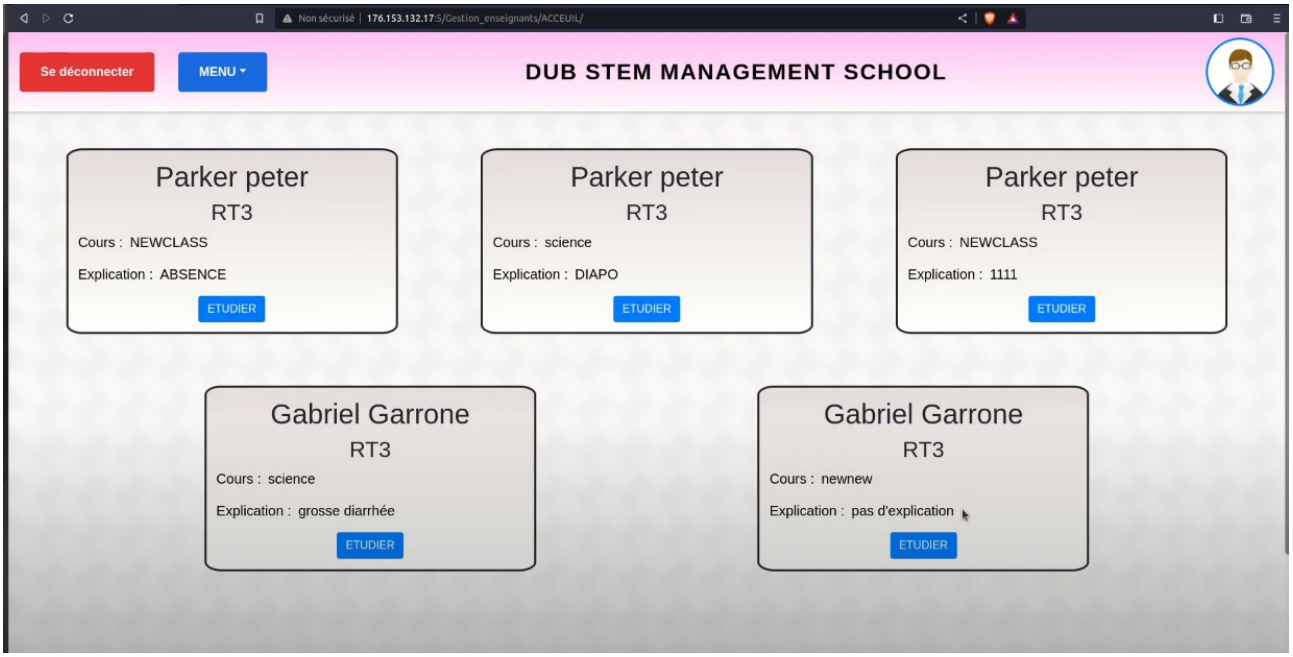
Quelques captures d'écran récapitulatives

Ajout d'un absence chez l'étudiant (ici GARRONE Gabriel)

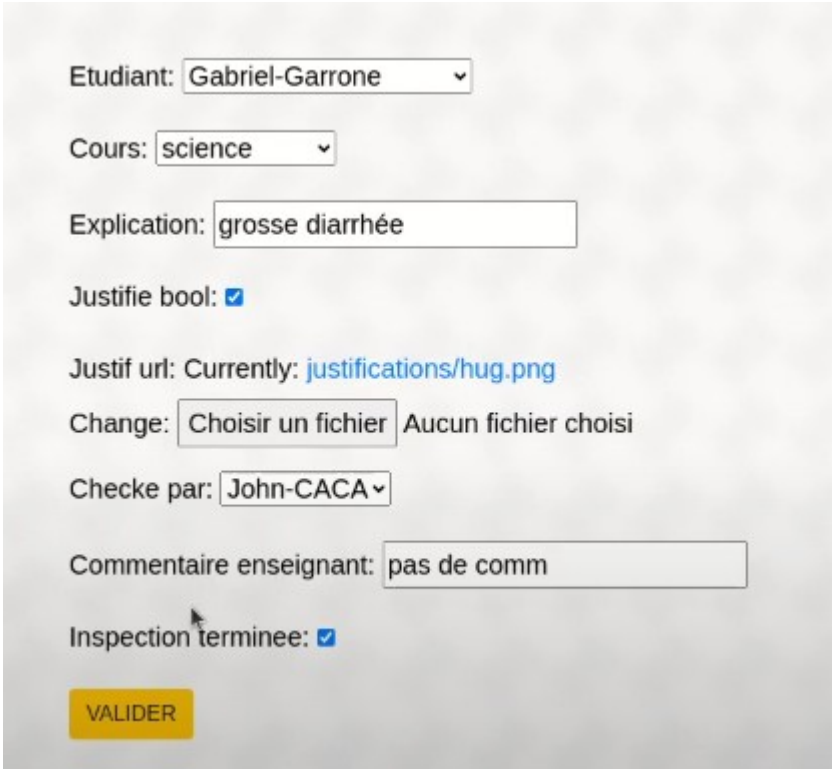
Fenêtre de visualisation d'absence pour l'étudiant avec leur état (en attente, refusé, accepté)



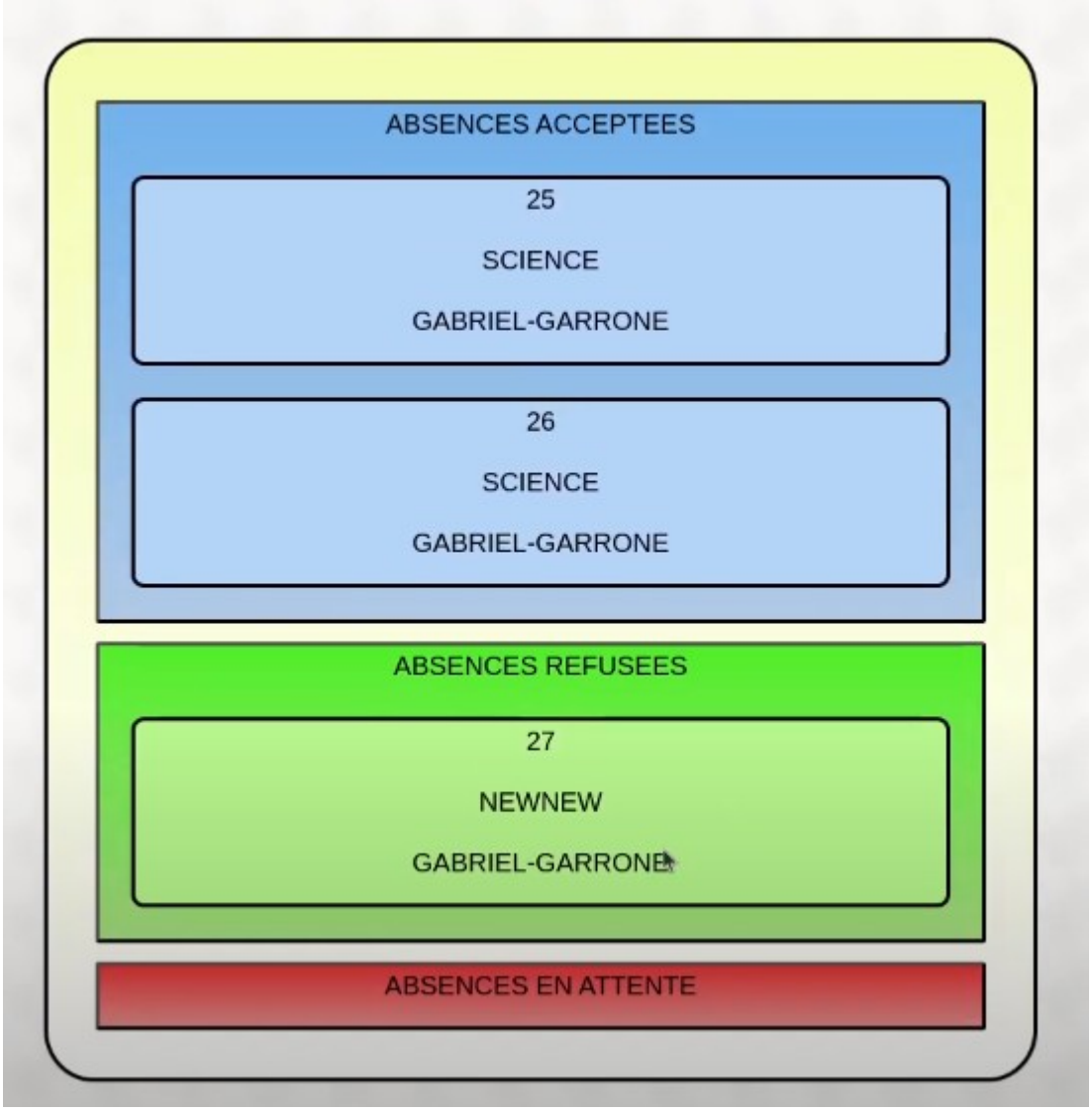
Visualisation des absences à étudier de TOUS les étudiants existants. Ici on remarque l'apparition des absences précédemment ajoutées par l'utilisateur GARRONE Gabriel.



En cochant la case « Justifie bool », l'enseignant accepte une des absences, il refusera l'autre



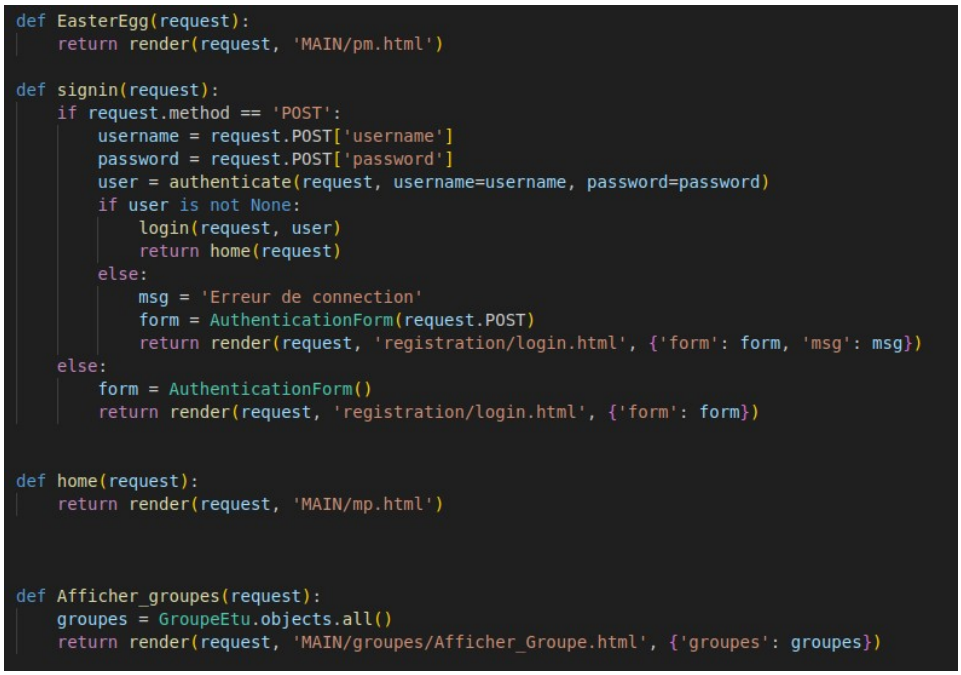
Enfin retour du point de vue de l'étudiant, on remarque que l'absence acceptée est mise dans la case en question, et de même pour celle refusée.



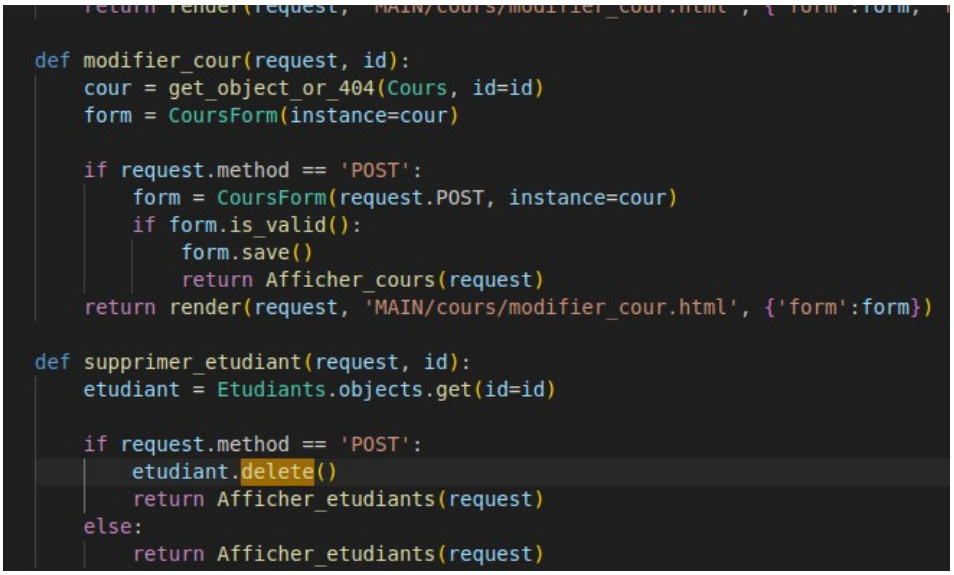
Je précise que la vidéo revient plus en détail sur cette fonctionnalité.

7) Fonctionnement

Pour accéder à tous ces éléments, il est impératif de créer des fonctions dans le fichier views.py pour chaque fonctionnalité proposée par notre projet. En voici quelques exemples d'affichage, d'ajout, de modification, de suppression et simplement pour accéder à une page html.



```python
def EasterEgg(request):
    return render(request, 'MAIN/pm.html')

def signin(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return home(request)
        else:
            msg = 'Erreur de connection'
            form = AuthenticationForm(request.POST)
            return render(request, 'registration/login.html', {'form': form, 'msg': msg})
    else:
        form = AuthenticationForm()
        return render(request, 'registration/login.html', {'form': form})


def home(request):
    return render(request, 'MAIN/mp.html')



def Afficher_groupes(request):
    groupes = GroupeEtu.objects.all()
    return render(request, 'MAIN/groupes/Afficher_Groupe.html', {'groupes': groupes})
```



```python
def modifier_cour(request, id):
    cour = get_object_or_404(Cours, id=id)
    form = CoursForm(instance=cour)

    if request.method == 'POST':
        form = CoursForm(request.POST, instance=cour)
        if form.is_valid():
            form.save()
            return Afficher_cours(request)
    return render(request, 'MAIN/cours/modifier_cour.html', {'form':form})

def supprimer_etudiant(request, id):
    etudiant = Etudiants.objects.get(id=id)

    if request.method == 'POST':
        etudiant.delete()
        return Afficher_etudiants(request)
    else:
        return Afficher_etudiants(request)
```

Il sera aussi nécessaire de mettre chaque URL porssible pour l'accès au page.

Pour préciser, la section entre guillemets après « path » et la section que l'on rentre dans l'url de navigateur après l'adresse d'hébergement pour accéder à une page.

Chacun des chemins appel la fonction qui permet d'accéder au html comme par exemple « home » montré dans la capture d'écran précédemment.

Lorsque l'on affiche une page correspondant à l'id d'un groupe, d'un étudiant, ou d'un cours, il est nécessaire de rajouter « <int:id> » pour que la page s'affiche suivant l'id

```
urlpatterns = [
    path('home/', views.home, name='home'),

    path('EasterEgg/', views.EasterEgg, name='EasterEgg'),

    path('signin/',views.signin, name='signin'),

    #////////////////////////////////////////////etudiant////////////////////////////////////////////////////////

    path('gestion_etudiants/<str:arg1>/', views.gestion_etudiants, name='gestion_etudiants'),
    path('acceuil_etu/<int:id>/', views.acceuil_etu, name='acceuil_etu'),
    path('Ajouter_absence/', views.Ajouter_absence, name='Ajouter_absence'),
    path('afficher_etude_absence/<int:id>/', views.afficher_etude_absence, name='afficher_etude_absence'),
    path('etudier_absence/<int:id>/', views.etudier_absence, name='etudier_absence'),


    #////////////////////////////////////////////enseignant////////////////////////////////////////////////////////

    path('Afficher_etudiants/', views.Afficher_etudiants, name='Afficher_etudiants'),
    path('Ajouter_etudiants/', views.Ajouter_etudiants, name='Ajouter_etudiants'),
    path('supprimer_etudiant/<int:id>/', views.supprimer_etudiant, name='supprimer_etudiant'),
    path('modifier_etudiant/<int:id>/', views.modifier_etudiant, name='modifier_etudiant'),
    path('afficher_modif_form/<int:id>/', views.afficher_modif_form, name='afficher_modif_form'),


    path('profil/update_mail_ens/<int:id>/', views.update_mail_ens, name='update_mail_ens'),

    path('Gestion_enseignants/<str:arg1>/', views.Gestion_enseignants, name='Gestion_enseignants'),

    path('Afficher_groupes/', views.Afficher_groupes, name='Afficher_groupes'),
    path('ajout_groupe/', views.ajout_groupe, name='ajout_groupe'),
    path('suppression_groupe/<int:id>/', views.suppression_groupe, name='suppression_groupe'),
    path('modification_groupe/<int:id>/', views.modification_groupe, name='modification_groupe'),

    path('Ajouter_cour/', views.Ajouter_cour, name='Ajouter_cour'),
    path('Afficher_cours/', views.Afficher_cours, name='Afficher_cours'),
    path('Supprimer_cour/<int:id>/', views.Supprimer_cour, name='Supprimer_cour'),
    path('afficher_modif_form_cour/<int:id>/', views.afficher_modif_form_cour, name='afficher_modif_form_cour'),
    path('modifier_cour/<int:id>/', views.modifier_cour, name='modifier_cour'),


    path('Afficher_enseignants/', views.Afficher_enseignants, name='Afficher_enseignants'),
    path('Ajouter_enseignants/', views.Ajouter_enseignants, name='Ajouter_enseignants'),
    path('Supprimer_enseignant/<int:id>/', views.Supprimer_enseignant, name='Supprimer_enseignant'),
    path('afficher_modif_form_ens/<int:id>/', views.afficher_modif_form_ens, name='afficher_modif_form_ens'),
    path('modifier_enseignant/<int:id>/', views.modifier_enseignant, name='modifier_enseignant'),

]
```

Il est aussi nécessaire de remplir le fichier forms.py et models.py pour le bon fonctionnement des champs de chaque table, sans oublier d'ensuite faire les migrations.

```python
from django import forms
from .models import GroupeEtu, Cours, Etudiants, Enseignant, Absences

class GroupeEtuForm(forms.ModelForm):
    class Meta:
        model = GroupeEtu
        fields = ['nom']


class CoursForm(forms.ModelForm):
    Groupes = forms.ModelChoiceField(queryset=GroupeEtu.objects.all())
    enseignant = forms.ModelChoiceField(queryset=Enseignant.objects.all())

    class Meta:
        model = Cours
        fields = [
            'Titre_cours',
            'enseignant',
            'Groupes',
            'Date',
            'Duree',
            ]

class EtudiantsForm(forms.ModelForm):
    groupe = forms.ModelChoiceField(queryset=GroupeEtu.objects.all())

    class Meta:
        model = Etudiants
        fields = ['nom', 'prenom', 'mail', 'groupe', 'photo', 'description', 'adresse', 'telephone', 'boursier']



class EnseignantForm(forms.ModelForm):
    class Meta:
        model = Enseignant
        fields = ['nom', 'prenom', 'mail']


class AbsenceForm(forms.ModelForm):
    Checke_par = forms.ModelChoiceField(queryset=Enseignant.objects.all(), required=False)

    class Meta:
        model = Absences
        fields = ['Etudiant', 'Cours', 'Explication', 'Justifie_bool', 'Justif_url', 'Checke_par', 'Commentaire_enseignant', 'Inspection_terminee']
```

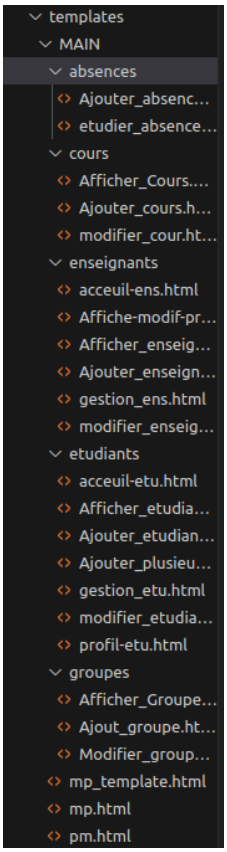
```python
from django.db import models

# Create your models here.

class Enseignant(models.Model):
    nom = models.CharField(max_length=50, blank=False, null=False)
    prenom = models.CharField(max_length=50, blank=False, null=False)
    mail = models.CharField(max_length=50, blank=False, null=False)

    def __str__(self):
        return f"{self.nom}-{self.prenom}"

class GroupeEtu(models.Model):
    nom = models.CharField(max_length=50, blank=False, null=False)

    def __str__(self):
        return f"{self.nom}"

class Cours(models.Model):
    Titre_cours = models.CharField(max_length=50, blank=False, null=False)
    enseignant = models.ForeignKey("Enseignant",on_delete=models.CASCADE, default=None, blank=True, null=True)
    Groupes = models.ForeignKey("GroupeEtu", on_delete=models.DO_NOTHING, default=None)
    Date = models.DateTimeField(blank=True, null=True)
    Duree = models.TimeField(blank=True, null=True)

    def __str__(self):
        return f"{self.Titre_cours}"

class Etudiants(models.Model):
    nom = models.CharField(max_length=50, blank=False, null=False)
    prenom = models.CharField(max_length=50, blank=False, null=False)
    mail = models.CharField(max_length=50, blank=False, null=False)
    groupe = models.ForeignKey("GroupeEtu", on_delete=models.CASCADE, default=None)
    photo = models.CharField(max_length=100, blank=True, null=True)
    description = models.CharField(max_length=300, blank=True, null=True)
    adresse = models.CharField(max_length=100, blank=True, null=True)
    telephone = models.IntegerField(blank=True, null=True)
    boursier = models.BooleanField()


    def __str__(self):
        return f"{self.nom}-{self.prenom}"

class Absences(models.Model):
    Etudiant = models.ForeignKey("Etudiants", on_delete=models.CASCADE, default=None)
    Cours = models.ForeignKey("Cours", on_delete=models.CASCADE, default=None)
    Explication = models.CharField(max_length=200, blank=False, null=False)
    Justifie_bool = models.BooleanField(blank=False, null=False)
```

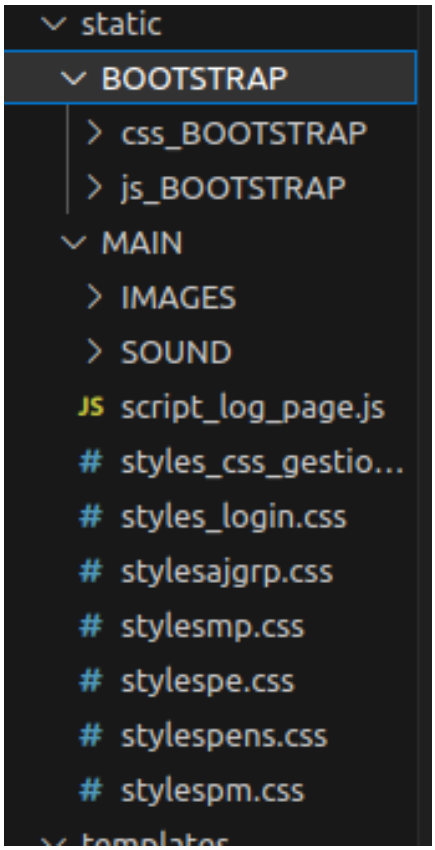
8) Apparence et style – HTML & CSS

Dans le cadre de ce projet, notre objectif principal était de développer une plateforme moderne similaire à Moodle, mais spécifiquement adaptée aux besoins d'une école privée de Management.

Je me suis occupé du développement des styles, et en bonne partie du développent des pages, mais en raison de contraintes de temps et de la nécessité de travailler sur deux projets distincts simultanément, nous n'avons pas pu mettre en place toutes les fonctionnalités souhaitées.

Cela a également entraîné de légères différences entre certaines pages par soucis d'optimisation. Malgré cela, je suis satisfait du résultat esthétique, puisque parfaitement fonctionnel.



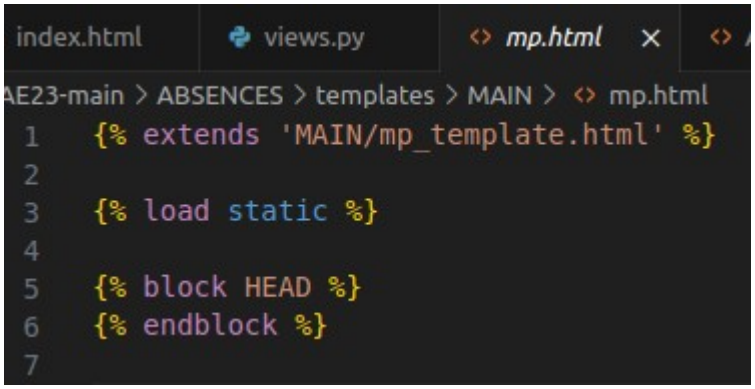


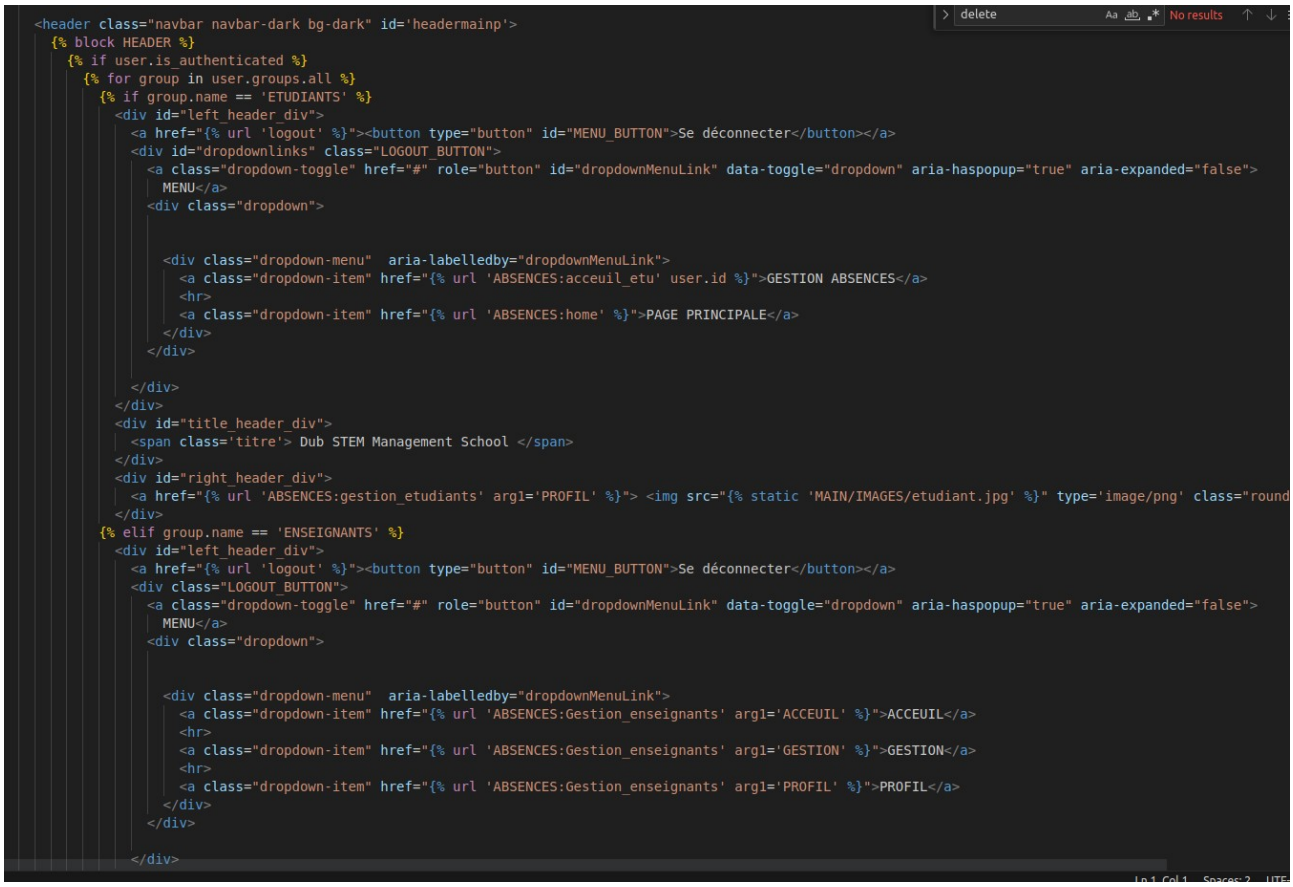
Nous avons créé un nombre de pages très important, mais nous avons réutilisé un maximum nos fichiers css et nos styles déjà existants par soucis d'optimisation encore une fois.

Enfin, nous avons mis en place un squelette notamment pour le header.

Ici on peut voir en python « load static » qui permet de charger nos fichiers statiques, que ce soient des styles, des images, des polices, etc.

Ensuite les autres lignes permettent de récupérer le header mis dans mp_template.html.





9) Conclusion

Ce projet de développement d'un site web de gestion des absences a été une expérience enrichissante pour moi à plusieurs niveaux. Tout d'abord, il m'a permis d'approfondir mes connaissances en programmation web, notamment en utilisant Django et en travaillant avec les langages HTML, CSS et Python. J'ai pu mettre en pratique les concepts et les techniques appris auparavant et les appliquer à un projet concret.

En réalisant ce projet, j'ai également développé mes compétences en gestion de projet. Nous avons dû planifier les différentes tâches, organiser notre temps et nos ressources, et suivre un diagramme de Gantt pour assurer une progression régulière. J'ai également du faire face aux imprévus et à trouver des solutions pour résoudre les problèmes rencontrés, et donc à m'adapter.

Travailler sur ce projet m'a également permis de renforcer ma compréhension des bases de données, notamment en utilisant MySQL pour la création et la manipulation des tables. Nous avons réutilisé nos connaissances acquises dans un précédent module pour concevoir un schéma relationnel, et établir les relations entre les différentes entités. De plus, j'ai travaillé en étroite collaboration avec Owen Pichot et Thibaut Cosenza, et cela m'a permis de développer mes compétences en communication, en partage d'idées et en résolution de problèmes en groupe.

Cependant, j'ai ressenti une certaine frustration de ne pas avoir eu suffisamment de temps pour explorer toutes les fonctionnalités que je souhaitais mettre en place. En raison de contraintes de temps et de la nécessité de travailler sur d'autres aspects du projet simultanément, nous avons dû faire des compromis et nous concentrer sur les fonctionnalités essentielles. J'aurais aimé avoir plus de temps pour développer davantage les pages et les fonctionnalités, afin de rendre le site encore plus complet et plus attrayant.

Dans l'ensemble, ce projet m'a apporté une grande satisfaction personnelle. J'ai pu concrétiser mes connaissances et mes compétences dans un projet concret et fonctionnel. J'ai apprécié chaque étape du processus de développement, depuis la planification jusqu'au déploiement du site web. Ce projet m'a donné une meilleure compréhension de la réalité du développement web et m'a motivé à continuer à approfondir mes compétences dans ce domaine.