SAE 23 - Mettre en place une solution informatique pour l'entreprise

Fiche de procédure des parties importantes de la mise en place de notre application

Martin Baumgaertner

16 juin 2022

Table des matières

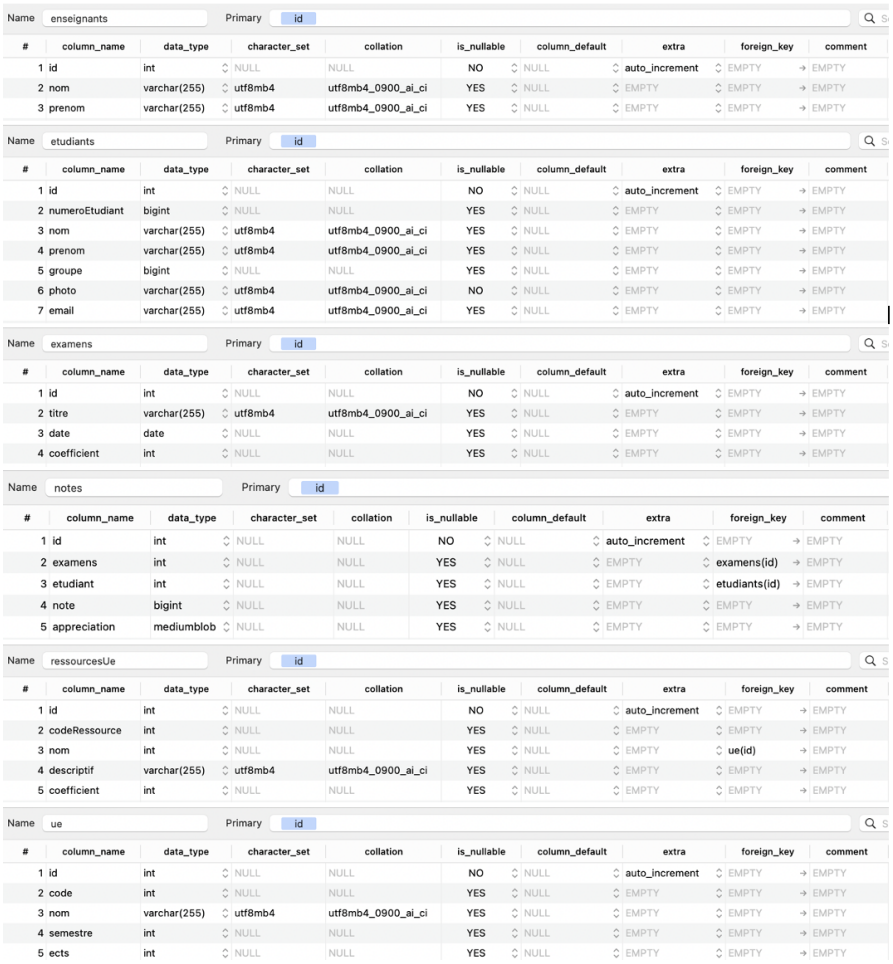
1 Introduction

Pour pouvoir mettre à bien notre SAE, il va falloir d'abord comprendre ce que le sujet veut nous faire comprendre. Nous devons réaliser une interface graphique sur l'internet qui permet la gestion de notes. Nous possèdons plusieurs domaines que nous devons relier à des sous-domaines. Je vais les lister ci-après sous forme de tableau :

Étudiant	Numéro étudiant Nom Prénom Groupe Photo E-mail
Unité d'enseignement	Code Nom Semestre ECTS
Resource associé aux UEs	Code ressource Nom Descriptif Coefficient
Enseignants	Nom Prénom
Examens	ID Titre Date Coefficient
Notes	Examen Étudiant Note Appréciation

1.1 Création de la base de données

J'ai donc créé la base de données comme demandé à partir de TablePlus, un éditeur/créateur de base de données. J'ai choisi d'utiliser TablePlus pour sa facilité d'utilisation et son design innovateur. On comprend rapidement comment utiliser l'éditeur à l'inverse de phpMyAdmin par exemple. J'ai créé les tables en veillant à ajouter les bonnes clés étrangères et les bons paramètres. Voici les tables créees :



Name enseignants — Primary id

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	EMPTY
2	nom	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
3	prenom	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY

Name etudiants — Primary id

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	EMPTY
2	numeroEtudiant	bigint	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
3	nom	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
4	prenom	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
5	groupe	bigint	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
6	photo	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	NO	NULL	EMPTY	EMPTY	EMPTY
7	email	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY

Name examens — Primary id

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	EMPTY
2	titre	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
3	date	date	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
4	coefficient	int	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY

Name notes — Primary id

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	EMPTY
2	examens	int	NULL	NULL	YES	NULL	EMPTY	examens(id)	EMPTY
3	etudiant	int	NULL	NULL	YES	NULL	EMPTY	etudiants(id)	EMPTY
4	note	bigint	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
5	appreciation	mediumblob	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY

Name ressourcesUe — Primary id

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	EMPTY
2	codeRessource	int	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
3	nom	int	NULL	NULL	YES	NULL	EMPTY	ue(id)	EMPTY
4	descriptif	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
5	coefficient	int	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY

Name ue — Primary id

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	EMPTY
2	code	int	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
3	nom	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY	EMPTY
4	semestre	int	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
5	ects	int	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY

FIGURE 1 – Base de données

1.2 Import de la base de données dans Django

Pour commencer, il faut donc lier notre base de données, pour ce faire nous devons éditer notre fichier settings.py dans les paramètres « database » :

```
DATABASES = {
    'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'grade',
    'USER': 'root',
    'HOST': 'localhost',
    'PORT': '3306',
    'PASSWORD': 'webgradedatabase',
    }
}
```

Code 1 – code settings.py

Ensuite j'ai donc pu connecter ma base de données à mon projet Django.
J'ai fait la commande « python3 manage.py makemigrations » pour pouvoir faire les migrations, puis je les ai appliqués avec la commande «python3 manage.py migrate » :

Par la suite, j'ai voulu donc crée un fichier models.py à partir de la base de données que je viens de créer en SQL :

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

Suite à cela j'ai donc bien obtenu mon fichier models.py avec toutes les "class" qui ont bien été créees.

2 Développement Backend

2.1 Gestion des CRUDs

2.1.1 Démarrage de la création des CRUDs

Après avoir donc récupéré mes différents « models » j'ai pu commencer le développement de la partie backend. J'ai configuré mes vues, mes templates pour obtenir d'abord des formulaires fonctionnels pour tous les CRUDs. Ensuite j'ai directement créé un .html squelette car je me suis vite rendu compte que j'allais avoir besoin de beaucoup de fichier html. J'ai mis dedans une barre de recherche qui relie tous les formulaires et les liens importants.

Pour pouvoir ajouter des images aux élèves il a fallu au préalable bien éditer la base de données. Au début mon champ image dans MySQL était de type « blob » car je pensais que c'était le paramètre adéquat. Mais en fin de compte lorsque je voulais récupérer les images pour les afficher, j'avais le chemin de destination de l'image qui s'affichait et non pas l'image. Donc après plusieurs recherches je me suis rendu compte que le bon paramètre MySQL pour les images était de type « Varchar ». En plus de cela il nous faut ajouter dans le fichier « settings.py » ces lignes de code :

```
STATIC_URL = '/static/'
MEDIA_URL = '/media/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)
```

Code 2 – code settings.py

On indique ici à notre projet d'aller chercher dans le fichier /media pour retrouver les images. Puis, dans le fichier « urls.py » de notre projet on va venir rajouter à la fin que nous voulons ajouter la possibilité de pouvoir charger des images :

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',include('notes.urls')),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

FIGURE 2 – Code URL du projet

Après avoir correctement éditer notre fichier « views.py » en demandant bien à notre fonction de récupérer les images comme ici :

```
form = EtudiantsForm(request.POST, request.FILES)
```

Nous avons notre projet qui fonctionne avec des images.

2.2 Fonction recherche étudiant

2.2.1 Views.py

Nous devons premièrement créer une vue dans notre fichier views.py qui consistera à dire que si l'on recherche un étudiant depuis un prénom, views.py ira chercher dans notre fichier models.py tous les prénoms des étudiants matchant avec notre recherche. S'il y a des résultats à afficher, alors, les résultats iront dans la variable « etudiants » pour qu'on puisse afficher les résultats dans une page html. Voici donc ci-dessous ce que j'ai développé pour avoir la fonction recherche fonctionnelle.

```
def recherche(request):
    if request.method == "POST":
        rechercher = request.POST['rechercher']
        etudiants = Etudiants.objects.filter(prenom__contains=rechercher)
        return render(request, 'recherche.html', {'rechercher':rechercher, 'etudiants': etudiants})
    else:
        return render(request, 'recherche.html')
```

FIGURE 3 – Code du fichier Views.py

2.2.2 html

Pour pouvoir afficher les valeurs que nous retourne notre vue, il faut donc créer un fichier html. Dedans on va donc définir que s'il y a une recherche AVEC résultat, nous irons rechercher parmi tous les étudiants le nom et l'étudiant ainsi que son prénom. Et, si la recherche est vide et qu'on ne recherche rien, un texte disant « vous n'avez pas recherché » apparaît chez l'utilisateur.

```
{% if rechercher %}

  {% for etudiant in etudiants %}
      {{etudiant.nom}} {{etudiant.prenom}}
  {% endfor %}

{% else %}
    <h1>Vous n'avez pas recherché...</h1>
{% endif %}
```

FIGURE 4 – Code HTML

3 Mise en place de notre application sur serveur linux

3.1 Configuration de la machine

3.1.1 Installation de base

Pour commencer, j'ai choisi de mettre en place mon serveur sur une machine sous Debian11, je l'ai installé en mode console uniquement. Puis, j'ai récupéré mon projet depuis GitHub en effectuant la commande « git clone https ://martinbaumg/SAE-23 ».

3.1.2 Création de l'environnement virtuel

Une fois fait j'ai créé un environnement virtuel que j'ai nommé « venv » Dans cet environnement virtuel j'ai installé pour les prérequis pour l'application, donc Django, Pip, et Pillow qui sert à la mise en place des éléments photos.

3.2 Mise en place de la base de données

Pour mettre en place la base de données j'ai installé MariaDB-server et client ainsi que MySQL. J'ai exporté la base de données que j'avais créé sur ma machine hôte et je l'ai transféré sur ma machine Debian. Ensuite, j'ai donc ouvert MariaDB et créée une base de données que j'ai nommé « grade » à l'aide de la commande suivante :

```
CREATE DATABASE grade;
```

Puis, j'ai importé ma base de données déjà existante dans la nouvelle base de données que je viens de créer dans MariaDB avec cette commande, depuis le terminal linux :

```
mysql -u root -p grade < grade.sql
```

3.3 Modification du fichier settings.py

3.3.1 Adresse IP

J'ai modifié certains champs du fichier settings du projet pour que tout fonctionne, premièrement, cette ligne, dans laquelle j'ai ajouté mon adresse IP :

```
ALLOWED_HOSTS = ['192.168.64.6']
```

Code 3 – Adresse IP

3.3.2 Base de données

Pour pouvoir mettre en place la base de données avec le projet j'ai dû me connecter à la base de données que j'ai créée avec MariaDB, pour ce faire j'ai créé un utilisateur depuis MariaDB, et je lui ai donnée tous les droits, à l'aide des commandes suivantes :

```
CREATE USER 'django'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON grade.* TO  'django'@'localhost';
```

Code 4 – code sql

3.3.3 Deployment de l'application

J'ai donc fait la commande « python3 manage.py runserver 0.0.0.0 :8000 ».
On indique à notre serveur de démarrer le projet sur l'adresse IP de la machine sur le port 8000. Enfin, en écrivant donc l'adresse IP de la machine suivis du port 8000 nous retrouvons donc localement sur notre machine l'application.

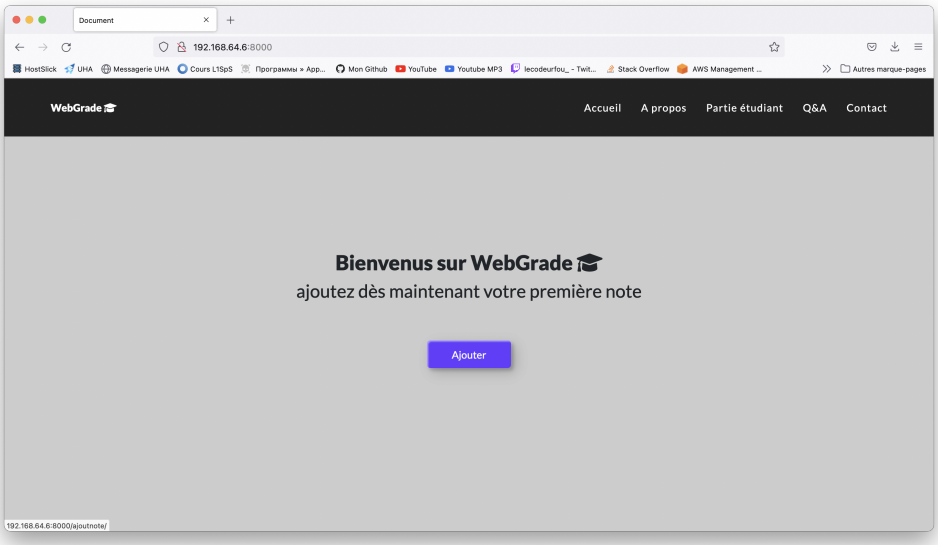


FIGURE 5 – Notre application deployée