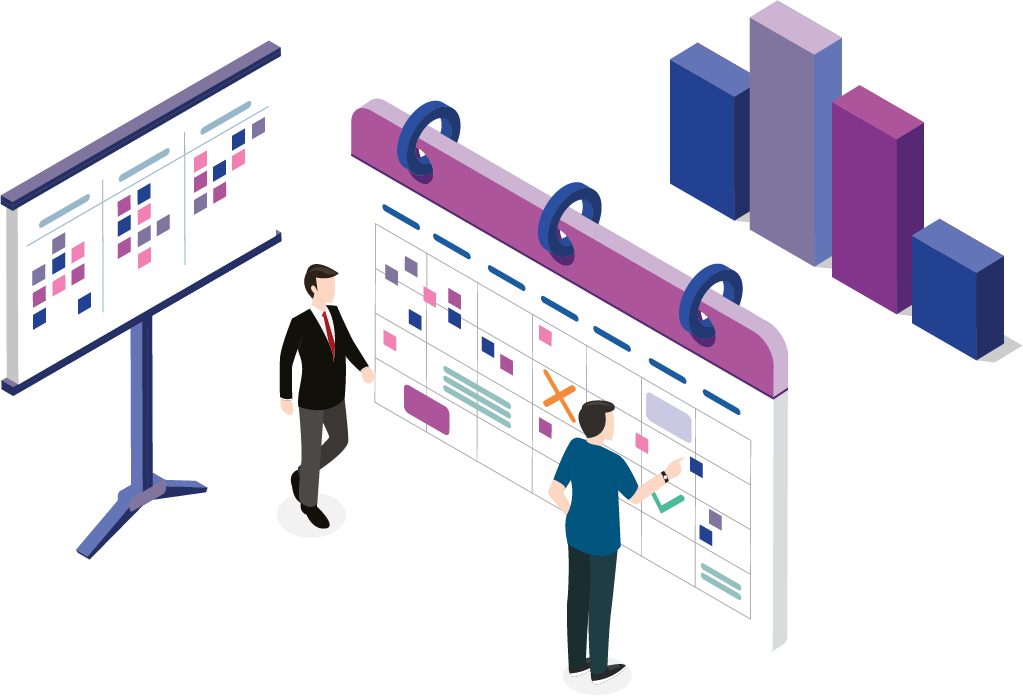
Onglet 1

**Fiche de Procédure** - Saé 23 - Mettre en place une solution informatique pour l’entreprise

**Projet de Gestion des Absences**



**Sommaire :**

1. **Présentation du projet**
2. **Technologies et ressources utilisées**
3. **Pré-requis techniques**
4. **Étapes d’installation détaillées**
5. **Fonctionnalités principales de l’application**
6. **Organisation du travail en équipe**
7. **Architecture du projet Django & structure du code**
8. **Configuration de la machine virtuelle (VM) et des services**
9. **Problèmes rencontrés et solutions apportées**
10. **Conclusion**
11. **Présentation du projet**

Ce projet a pour objectif de développer une application web permettant de **gérer les absences des étudiants** d’un département. Il permet :

* L’enregistrement des absences à des cours spécifiques
* La justification des absences
* L’import d’absences via un fichier CSV
* L’édition de rapports par cours ou par étudiant

Le site repose sur un serveur web et une base de données, installés dans une machine virtuelle Linux.

1. **Technologies utilisées**

* **Linux Debian 12** (machine virtuelle)
* **Python 5.15.11**
* **Django 5.2.3** (framework web en Python)
* **MariaDB 11.7.2** (base de données légère intégrée)
* **HTML/CSS** (interface utilisateur)
* **Git / GitHub** (collaboration et versionnage)
* **CSV** (format d’import d’absences)

1. **Pré-requis techniques**

Avant de commencer, vous devez disposer :

* **D’un poste avec Linux installé** (idéalement Debian 12)
* **D’un accès Internet**
* **D’un terminal** (accès en ligne de commande)
* **Des droits administrateur** (sudo)

1. **Étapes d’installation détaillées**

**Étape 1 : Ouvrir un terminal**

Sous Debian ou Ubuntu, faites Ctrl + Alt + T pour ouvrir le terminal.

**Étape 2 : Mettre à jour le système**

**sudo apt update && sudo apt upgrade -y**

Met à jour les paquets de votre système. Entrez votre mot de passe si demandé.

**Étape 3 : Installer Python 3 et pip**

**sudo apt install python3 python3-pip python3-venv git -y**

Cette commande installe Python, son gestionnaire de paquets pip, les environnements virtuels et Git.

**Étape 4 : Créer un environnement virtuel**

**python3 -m venv venv**

**source venv/bin/activate**

Crée un environnement isolé pour installer les bibliothèques sans polluer le système.

**Étape 5 : Installer les dépendances**

**pip install -r requirements.txt**

Cette commande installe toutes les bibliothèques nécessaires, listées dans requirements.txt.

**Étape 6 : Initialiser la base de données**

**python manage.py migrate**

Cela crée automatiquement les tables de la base de données MariaDB.

**Étape 7 : Créer un compte administrateur**

**python manage.py createsuperuser**

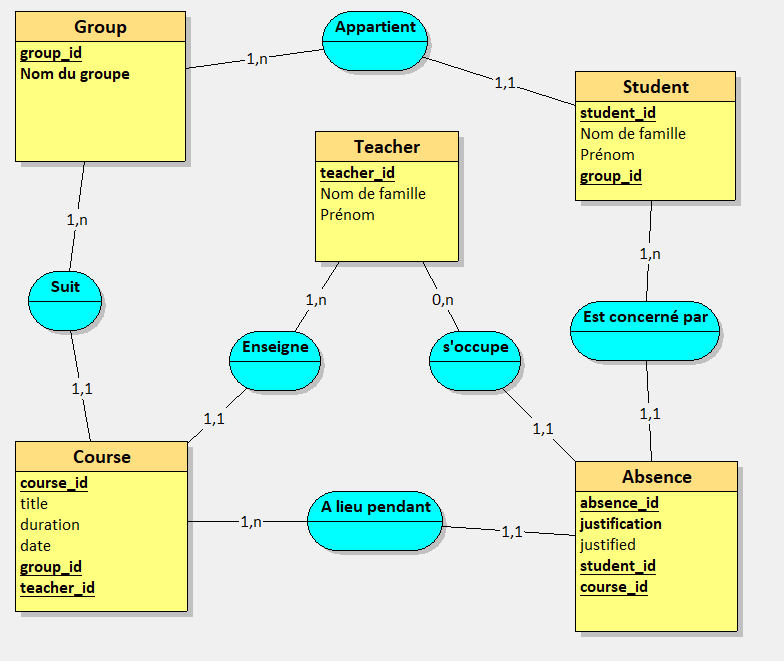
Entrez un nom d’utilisateur, un mot de passe et un email. Ce compte vous permettra de gérer le site depuis l’interface d’administration (/admin).

**Étape 8 : Lancer le serveur**

**python manage.py runserver**

Puis, ouvrez votre navigateur et tapez : **http://127.0.0.1:8000/**

1. **Fonctionnalités de l’application**



**Gestion des données**



**Pour la réalisation de ce schéma entité-relation, nous avons utilisé le logiciel Looping :** [**https://www.looping-mcd.fr/**](https://www.looping-mcd.fr/) **permettant de réaliser de la modélisation conceptuelle de données totalement Open Source.**

**La réalisation de ce schéma a été fait grâce au fichier de configuration “models.py” … à compléter !!!!!**

**Groupes :** ajout/modification des groupes d’étudiants

**Étudiants :** informations personnelles, groupe

**Enseignants :** informations personnelles

**Cours :** titre, date, enseignant, durée, groupe associé

**Absences :** création manuelle ou import via fichier

**Import CSV**

Vous pouvez importer des absences à partir d’un fichier .csv.

**Exemple de fichier CSV :**

**A compléter !!!!!!**

1. **Organisation du travail en équipe**

**Répartitions des rôles :**

La répartition des tâches s’est faite à parts égales entre les membres du groupe.

La création d’un diagramme de Gantt a été nécessaire pour organiser efficacement le projet, visualiser les étapes, et suivre l’avancement. Il détaille clairement qui fait quoi et à quel moment.

**Outils collaboratifs :**

**GitHub :** suivi du code

**Google Drive :** documents partagés

**Diagramme de Gantt :** planification des livrables

**Vous pouvez consulter notre Gantt ici :** [**https://docs.google.com/spreadsheets/d/1lEZbBS\_p8gX7BIr-TPMM-xg0pseG6wq6GKmb7ieiznQ/edit?gid=1408921046#gid=1408921046**](https://docs.google.com/spreadsheets/d/1lEZbBS_p8gX7BIr-TPMM-xg0pseG6wq6GKmb7ieiznQ/edit?gid=1408921046#gid=1408921046)

1. **Architecture du projet Django & structure du code**

L’application suit l’architecture classique de Django, organisée en modules :

**Ajouter l’image de l'arborescence !!!!!!!**

**Le projet suit l’organisation classique Django :**

* **models.py : définition des modèles pour les entités** (Étudiants, Enseignants, Cours, Absences…)
* **views.py : logique métier, traitement des requêtes, appels aux modèles**
* **forms.py : création des formulaires pour les saisies utilisateurs**
* **urls.py : routage des pages de l’application**
* **templates/ : pages HTML avec intégration dynamique des données**
* **static/ : fichiers CSS, JS et images**

Tous les fichiers, codes sources, structures de répertoires et indications nécessaires au développement de l’application sont disponibles sur ce dépot GitHub :

**METTRE LE LIEN GITHUB !!!!!!!!!!!!!!!!!!!!!!!!!!**

1. **Configuration de la machine virtuelle (VM) et des services**

Le projet a été développé et déployé dans une machine virtuelle Linux Debian 12, qui regroupe tous les services nécessaires :

* **Django pour le site web**
* **MariaDB pour la base de données**
* **Environnement Python avec tous les modules requis (django, mysqlclient, etc.)**

**Connexion Django ↔ MariaDB**

**Django est relié à MariaDB via le fichier settings.py, avec le connecteur mysqlclient.**

**Extrait de configuration : CODE !!!!!!!!!!!!!!!!!!!!!!!!**

**Exemple de commandes utilisées :**

**sudo mysql -u root -p**

**CREATE DATABASE absences\_db;**

**CREATE USER 'admin'@'localhost' IDENTIFIED BY 'motdepasse';**

**GRANT ALL PRIVILEGES ON absences\_db.\* TO 'admin'@'localhost';**

**La VM joue donc le rôle de serveur tout-en-un (**web + BDD + backend Python**), simplifiant le développement et les tests.**

1. **Problèmes rencontrés et solutions apportées**

* **Problème de connexion entre Django et MariaDB**
* **Erreurs lors de la migration des modèles Django**
* **Mauvais affichage ou non-chargement des fichiers CSS/JS**
* **Problèmes d’importation d’absences via fichier CSV**

**Elles ont été résolues en installant les bons outils, en corrigeant la structure du code, en configurant correctement Django.**

1. **Conclusion**

**Ce projet nous a permis de mettre en pratique l’ensemble des compétences liées au développement web, à la gestion de bases de données, et à l’administration d’un environnement Linux.**

**Grâce à une organisation rigoureuse et un travail collaboratif efficace, nous avons conçu et déployé une application fonctionnelle répondant au cahier des charges.**

**Ce travail nous prépare concrètement aux réalités professionnelles de la conception et du déploiement de solutions web d’entreprise.**

code et explication à ajouter

Version [models.py](http://models.py) dans /gestionabsences/gestionapp/models.py

from django.db import models

class Group(models.Model):

name = models.CharField(max\_length=100)

number\_students = models.PositiveIntegerField(default=0)

def \_\_str\_\_(self):

return self.name

def number\_students(self):

return self.students.count()

class Student(models.Model):

last\_name = models.CharField(max\_length=100)

first\_name = models.CharField(max\_length=100)

group = models.ForeignKey(Group, on\_delete=models.CASCADE, related\_name='students')

def \_\_str\_\_(self):

return f"{self.first\_name} {self.last\_name}"

class Teacher(models.Model):

last\_name = models.CharField(max\_length=100)

first\_name = models.CharField(max\_length=100)

def \_\_str\_\_(self):

return f"{self.first\_name} {self.last\_name}"

class Course(models.Model):

name = models.CharField(max\_length=200)

date = models.DateField()

teacher = models.ForeignKey(Teacher, on\_delete=models.CASCADE, related\_name='courses')

duration = models.DurationField()

group = models.ForeignKey(Group, on\_delete=models.CASCADE, related\_name='courses')

def \_\_str\_\_(self):

return f"{self.title} on {self.date}"

class Absence(models.Model):

student = models.ForeignKey(Student, on\_delete=models.CASCADE, related\_name='absences')

course = models.ForeignKey(Course, on\_delete=models.CASCADE, related\_name='absences')

justified = models.BooleanField(default=False)

justification = models.TextField(blank=True, null=True)

class Meta:

unique\_together = ('student', 'course')

def \_\_str\_\_(self):

return f"Absence of {self.student} for {self.course} - Justified: {self.justified}"