

SAE23 METTRE EN PLACE UNE SOLUTION INFORMATIQUE POUR L'ENTREPRISE

sujet7: Gestion de trafic aérien

Groupe :

- FAHMI Chahinez
- GRIES-HUFFSCHMITT Pierre
- BRODIN Mathias



INTRODUCTION

Sommaire

- Objectif de la SAE.
- Ressources utilisés.
- Répartition du travail.
- Schéma relationnel de la base de donnée.
- Organisation des 6 CRUD.
- Création des formulaires et de leur style.
- Configuration des VM
- Gestion de la base données.
- Création du serveur Django.





OBJECTIF DE CETTE SAE

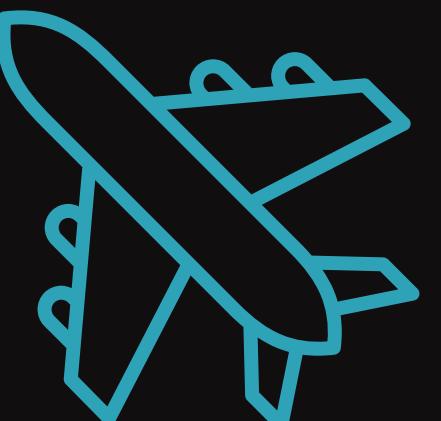
Développer un outil sur le Web permettant gérer le trafic aérien.

Intuitif

Accessible

fonctionnel

beau



RESSOURCES UTILISÉES

Semestre 1

- R1.02 Principe et architecture des réseaux
- R1.07 Programmation en Python
- R1.08 Base des systèmes d'exploitation
- R1.09 Technologie du WEB
- R1.15 Gestion de Projet

Semestre 2

- R2.07 Base de données
- R2.08 Analyse et traitement des données structurées
- R2.09 WEB dynamique

RÉPARTITION

Schema relationnel

- Mathias

Diagramme de GANT

- Mathias, Pierre

Installation des VM

- Mathias, Chahinez

Python

- Mathias, Pierre

Configuration des serveurs

- Mathias

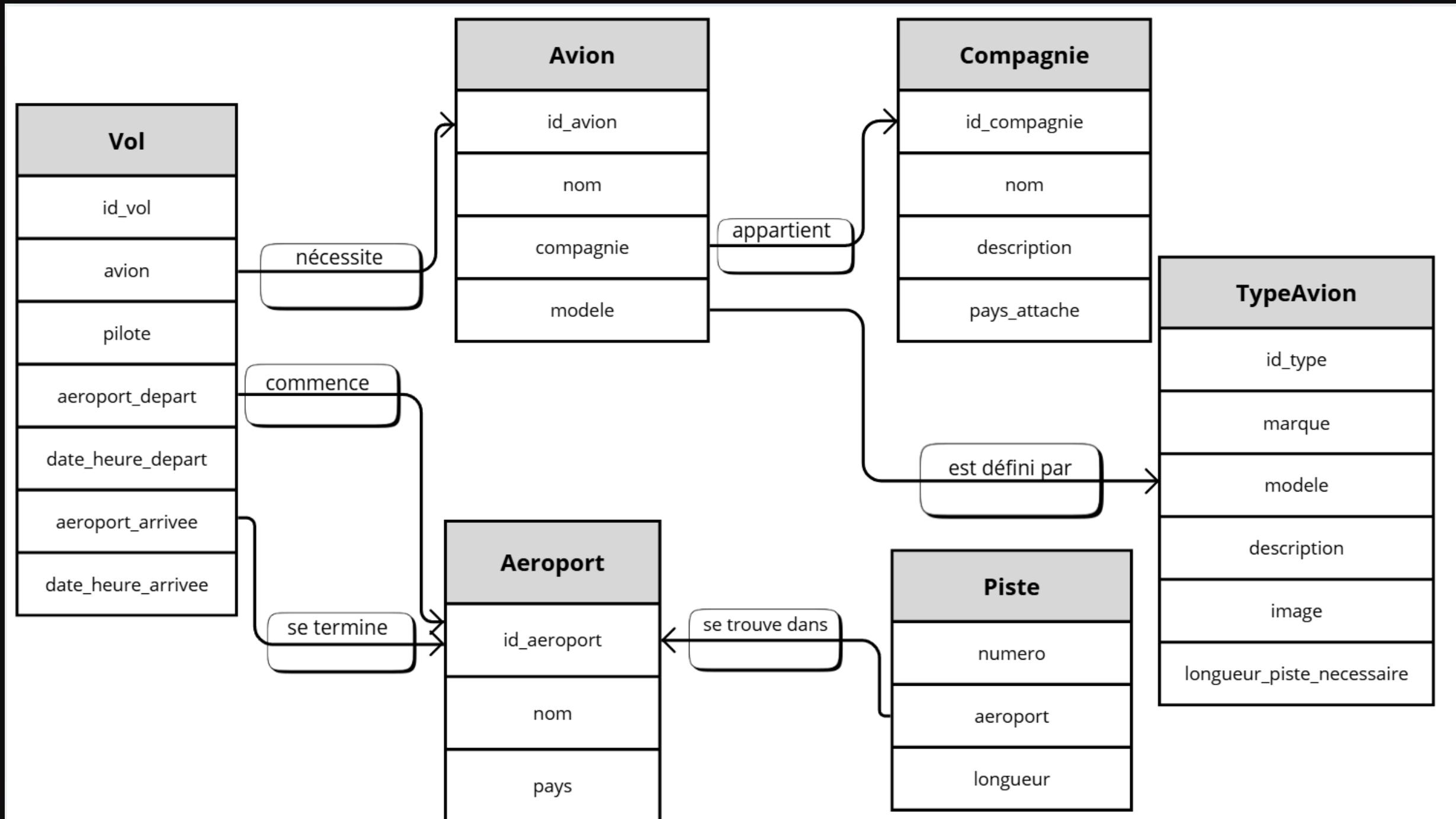
HTML CSS BOOSTRAP

- Chahinez, Pierre,

SCHÉMA RELATIONNEL

Tables de la base de données :

- Avion
- Vol
- Aeroport
- Piste
- TypeAvion
- Compagnie



ORGANISATION DES 6 CRUD

- Création du forms.py et du models.py avec les attributs des tables.
- 1 seul urls.py dans l'application
- 1 fichier views.py pour chaque tables.
- 4 formulaires pour chaque CRUD et un formulaire de page d'accueil.
- 1 fichier style.css pour tous les formulaires

aeroport_views

types_views

compagnies_views

avion_views

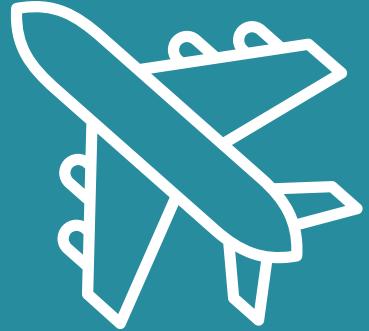
vol_views

piste_views



Structure du urls.py

```
1  from django.urls import path
2  from . import aeroport_views, piste_views, avion_views, compagnie_views, type_views, vol_views
3
4  urlpatterns=[
5      # page pour les aeroports
6      path("create/", aeroport_views.create),
7      path("traitement/", aeroport_views.traitement),
8      path("index/", aeroport_views.index),
9      path("affiche/<int:id>/", aeroport_views.affiche),
10     path("update/<int:id>/", aeroport_views.update),
11     path("updatetraitement/<int:id>/", aeroport_views.updatetraitement),| # Line 11
12     path("delete/<int:id>/", aeroport_views.delete),
13     path("", aeroport_views.home),
14     # page pour les pistes
15     path("create2/", piste_views.create2),
16     path("traitement2/", piste_views.traitement2),
17     path("index2/", piste_views.index2),
18     path("affiche2/<int:id>/", piste_views.affiche2),
19     path("update2/<int:id>/", piste_views.update2),
20     path("updatetraitement2/<int:id>/", piste_views.updatetraitement2),
21     path("delete2/<int:id>/", piste_views.delete2),
```



Structure des views.py

```
> ➜ vol_views.py > updatetraitement6
from django.shortcuts import render, HttpResponseRedirect
from .forms import VolForm
from . import models

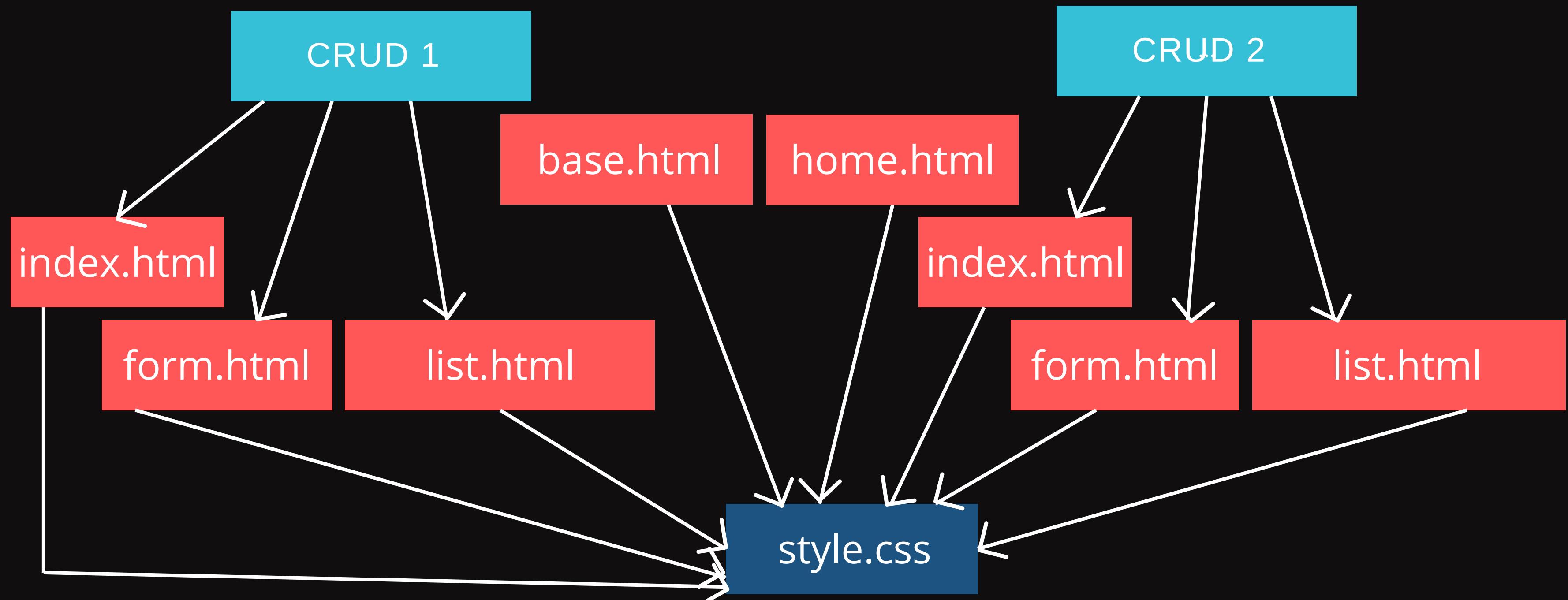
def create6(request):
    if request.method == "POST":
        form = VolForm(request)
        return render(request, "app/vol/create.html", {"form": form})
    else:
        form = VolForm()
        return render(request, "app/vol/create.html", {"form": form})

def traitement6(request):
    lform = VolForm(request.POST)
    if lform.is_valid():
        vol = lform.save()
        return HttpResponseRedirect("/app/index/")
    else:
        return render(request, "app/vol/create.html", {"form": lform})

def index6(request):
    liste = list(models.Vol.objects.all())
    return render(request, "app/vol/index.html", {"liste": liste})

25  def affiche6(request, id):
26      vol = models.Vol.objects.get(pk = id)
27      return render(request, "app/vol/traitement.html", {"vol" : vol})
28
29  def update6(request, id):
30      vol = models.Vol.objects.get(pk=id)
31      form = VolForm(vol.dico())
32      return render(request, "app/vol/update.html", {"form": form, "id" : id})
33
34  def updatetraitement6(request, id):
35      lform = VolForm(request.POST)
36      if lform.is_valid():
37          vol = lform.save(commit = False)
38          vol.id = id
39          vol.save()
40          return HttpResponseRedirect("/app/index/")
41      else:
42          return render(request, "app/vol/update.html", {"form": lform, "id" : id})
43
44  def delete6(request, id):
45      vol =models.Vol.objects.get(pk = id)
46      vol.delete()
47      return HttpResponseRedirect("/app/index")
```

CRÉATION DES FORMULAIRES ET DE LEUR STYLE



CONFIGURATION DES VMs

- Usage de 2 VMs (Serveur SQL, Serveur Django)
- Le serveur SQL héberge la base de donnée.
- Le serveur Django conserve le code python, HTML et CSS.
- Les 2 VMs ont 2 interfaces chacunes (SQL : 1 en NAT et 1 en static) (Django : 1 en bridge et 1 en static).



GESTION DE LA BASE DE DONNÉES

- IP : 192.168.100.10
- Service déployé : MySQL avec Mariadb
- Nom de la base de donnée : gestion_trafic
- Usage de commandes MySQL (create database..., create table..., id_... INT NOT NULL PRIMARY KEY)



CRÉATION DU SERVEUR DJANGO

- IP static: 192.168.100.11
- Installer les paquets python nginx
gunicorn
- Installer l'environnement virtuel
Python
- Dans l'environnement virtuel, installer
django et mysqlclient.
- Modifier le fichier settings.py pour lier
la base de données et gérer l'accès à la
base de données.
- Effectuer un makemigrations puis
migrate.
- Démarrer le serveur avec runserver.



source : digitalOcean

DÉMONSTRATION

PROBLÈMES RENCONTRÉS

- Impossible d'installer mysqlclient sur le serveur django
- Des templates qui n'étaient pas détectées
- problèmes d'interfaces réseaux

A photograph of a man sitting in an airport terminal. He is wearing a brown denim vest over a light-colored shirt and jeans. His feet are propped up on a dark green suitcase. He is looking out of a large window at an airplane taking off from a runway. The word "CONCLUSION" is overlaid on the image in a white, sans-serif font.

CONCLUSION