**Fiche Creation Projet**

**I. Creation Environnement Virtuel**

Se placer sur le bon repertoire avec cd

pip install virtualenv

python -m virtualenv monenv

Windows 🡺 ./monenv/Scripts/activate

Mac 🡺 source ./monenv/activate

Desactivate : deactivate

**II. Creation projet Django + fichier back**

**Etape1 : Ligne commande**

1.pip install django djangorestframework

2. $ python -m django startproject test\_project

3. cd test\_project

4. python -m django startapp backend

**Etape 2 : Fichier Settings**

Dans le fichier setting du projet sur installed\_apps rajouter :

    'backend.apps.BackendConfig',

    'rest\_framework',

    'corsheaders',

**Etape 3 : Test avec construction d’une view dans l’app:**

1. Dans le dossier back créer un fichier url.py

2.Sur le fichier url du projet :

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',include('backend.url'))

]

3. sur fichier views dans backend et le coder le fichier

from django.shortcuts import render

from django.http import HttpResponse

def main(*request*):

    return HttpResponse("Hello")

4.Coder le fichier url de backend qui va appeler la fonction main. Ce fichier URL est appelé dans urls du projet

from django.urls import path

from .views import main

urlpatterns = [

    path('', main),]

**Etape 4 : Lancer le projet (a faire des que changement sur les modèles ou database)**

Sur le terminal taper :

Nage.py migrate

puis

python manage.py migrate

puis lancer le serveur

python manage.py runserver

On peut aller sur le lien et voir le site

**III. Creation front react**

**Etape 1 : ligne commande**

npx create-react-app frontend

cd frontend

npm i react-router-dom sass

npm i axios => pour gérer data (fecth)

**Etape 2 : Nettoyage fichiers Front**

**Public** : garder ca et supprimer commentaires index.html :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <title>Cours React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>

**Src :**

**-**Garder que App.js et index.js

-Rajouter dossier styles

-Supprimer Contenu App.js

-index.js garder que ca :

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

**-Rajouter dossier pages et components**

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

**importer style dans index.js**

import "./styles/index.scss";

**Etape 4 : Creation liaisons**

<https://www.youtube.com/watch?v=FhkqMHxchZ8>

1.Settings

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [ os.path.join(BASE\_DIR, 'frontend/build')],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'corsheaders.middleware.CorsMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

*# L'origine de votre application React]*

CORS\_ALLOWED\_ORIGINS = ["http://localhost:3000",

                        "http://localhost:8000", "http://127.0.0.1:8000"]

*# Autorisez tous les domaines (utilisez True si nécessaire)*

CORS\_ALLOW\_ALL\_ORIGINS = False

STATIC\_URL = '/static/'

*# Ajoutez le chemin vers le dossier "public" de votre application React*

STATICFILES\_DIRS = [

    os.path.join(BASE\_DIR, 'frontend/build/static'),

]

2. créer un fichier views dans le dossier test\_project

from django.shortcuts import render

*# Create your views here.*

def index(*request*,\**args*, \*\**kwargs*):

    return render(*request*,'index.html')

3. dans le fichier du projet ajouter dans les urls le path du fichier url crée dans front

from django.contrib import admin

from django.urls import path,include

from django.views.generic import TemplateView

from . import views

urlpatterns = [

    path('admin/', admin.site.urls),

*# path('',include('backend.urls')),*

    path('', views.index,*name*='index'),

*# path('',include('frontend.urls'))*

]

**Etape 5 : Lancement**

Se placer sur frontend et npm run build

Cd ..

python manage.py runserver

**IV. Creation et lisaison de la BDD**

**Etape 1 : Creation Model**

from django.db import models

class Eleve(models.Model):

    nom = models.CharField(*max\_length*=20, *default*="", *unique*=True)

    prenom = models.CharField(*max\_length*=20, *default*="", *unique*=True)

    age = models.IntegerField()

    SEXE\_CHOICES = (

        ('homme', 'Homme'),

        ('femme', 'Femme'),

    )

    sexe = models.CharField(

*max\_length*=6, *choices*=SEXE\_CHOICES, *default*='homme')

**Etape 2 : Créer le fichier serializers.py**

from rest\_framework import serializers

from .models import Eleve

class EleveSerializer(serializers.ModelSerializer):

    class Meta:

        model = Eleve

        fields = ('nom', 'prenom', 'age', 'sexe')

**Etape 3 : Créer le fichier views.py**

from django.shortcuts import render

from rest\_framework import generics, status

from rest\_framework.response import Response

from .serializers import EleveSerializer

from .models import Eleve

*#View qui sert a afficher les eleves*

class EleveView(generics.ListAPIView):

    queryset = Eleve.objects.all()

    serializer\_class = EleveSerializer

*#View pour creer un eleve*

class EleveCreateView(generics.CreateAPIView):

    queryset = Eleve.objects.all()

    serializer\_class = EleveSerializer

    def create(*self*, *request*, \**args*, \*\**kwargs*):

        serializer = *self*.get\_serializer(*data*=*request*.data)

        if serializer.is\_valid():

*self*.perform\_create(serializer)

            return Response({"message": "Élève créé avec succès"}, *status*=status.HTTP\_201\_CREATED)

        return Response(serializer.errors, *status*=status.HTTP\_400\_BAD\_REQUEST)

**Etape 4 : URLS de backend**

from django.urls import path

from .views import EleveView

from .views import EleveCreateView

urlpatterns = [

    path('api', EleveView.as\_view()),

    path('api/create', EleveCreateView.as\_view(), *name*='eleve-create'),

]

**Etape 5 : Liaison sur le front**

import React from "react";

import { useEffect, useState } from "react";

import Navigation from "../components/Navigation";

import axios from "axios";

import Student from "../components/Student";

const Histo = () => {

  const [dataStudent, setDataStudent] = useState([]);

  const getData = () => {

    const init = {

      method: "GET", *// Get =>Apporte moi données // POST (incremente donées), PUT (MAJ), DELETE (Supprimer données)*

      headers: { "Content-Type": "application/json" },

      mode: "cors",

      cache: "default",

    };

    axios.get("http://localhost:8000/back/api", init).then((*response*) => {

      setDataStudent(*response*.data);

      console.log(*response*.data);

    });

  };

  useEffect(() => {

    getData();

  }, []);

  return (

    <div>

      <Navigation />

      <div *className*="eleves-container">

        {dataStudent.map((*student*, *index*) => (

          <Student *key*={*index*} *stu*={*student*} />

        ))}

      </div>

    </div>

  );

};

export default Histo;

import React from "react";

import Navigation from "../components/Navigation";

const Home = () => {

  return (

    <div>

      <Navigation />

      <h1>Coucou Cyrikl</h1>

    </div>

  );

};

export default Home;

**Etape 6 : Lancement**

Se placer sur frontend 🡺 npm run build

Cd ..

Faire les migration pour models

Python manage.py runserver

Apres ca se lance et on peut travailler sur le serveur npm pour avoir les données et le front.