**Exercice Euronews**

Une image contenant texte, homme, costume, capture d’écran

Description générée automatiquement

Surligner en vert : le nom du fichier  
Surligner en jeune : morceaux de code expliquer  
Texte en orange : commande python   
Texte en bleu : bout de code en rapport avec les requêtes

Explication partie Django :

models.py  
Je commence par définir un model pour mes articles que j’appelle « Article ». Cela représente une table dans une bdd.

Pour ce faire j’ai dû utiliser la commande « python manage.py startapp Article ». Cette commande va créer une nouvelle application pour définir mon modèle « Article »

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

**class** **Article**(models.Model):  
J’indique à ma class « Article » qu’elle hérite de « models.Model », ce qui signifie qu’elle est un modèle de la bdd.

Ensuite je définie les champs du modèle :   
title\_fr = models.CharField(max\_length=200)  
title\_en = models.CharField(max\_length=200)  
image = models.ImageField(upload\_to='images')  
description\_fr = models.TextField()  
description\_en = models.TextField()

Pour l’image je la sauvegarde dans le dossier « images »

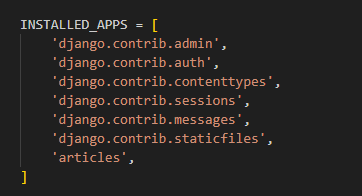
Et je fini par indiquer que lors qu’un article est afficher, il affichera le titre en français « title\_fr »  
def \_\_str\_\_(self):  
 return self.title\_fr

Forms.py   
Ensuite je défini un formulaire basé sur le modèle « Article » Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

class ArticleForm(forms.ModelForm):  
ArticleForm est une classe qui hérite de forms.ModelForm, ce qui signifie qu'elle est un formulaire lié à un modèle de base de données. Cela permet de créer facilement un formulaire pour ajouter ou modifier des instances du modèle Article.

class Meta:  
 model = Article  
 fields = ['title\_fr', 'title\_en', 'image', 'description\_fr', 'description\_en']  
Je crée une classe interne “Meta” qui fournit des métadonnées pour le formulaire.  
 - J’indique que ce formulaire est lié au modèle « Article ».  
 - j’indique avec fields, les champs qui doit être inclus dans le formulaire.

Settings.py  
Dans mon fichier settings.py dans « INSTALLED\_APP » j’ajoute mon application à la liste.  


Ensuite je fais la commande « python manage.py makemigrations » pour créer les migrations pour ce modèle.

Et j’applique les migrations à la base de données avec « python manage.py migrate »

view.py  
Je crée un fichier qui va me permettre de gérer les différentes vues de mon application Django.

Pour la vue « add\_article » :  
Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Si la méthode de la requête est POST, elle crée un formulaire ArticleForm avec les données soumises. Si le formulaire est valide, l'article est enregistré dans la base de données et l'utilisateur est redirigé vers la page d'accueil (home).

Si la méthode de la requête n'est pas POST, un formulaire vide est créé et affiché.

La vue renvoie un rendu du template add\_article.html, avec le formulaire passé en contexte.

Pour la vue « articles\_json » :  
Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Cette vue permet de récupérer tous les articles de la bdd et les retourner au format JSON.

* Je commence par récupérer tous les articles « articles = Article.objects.all() ».
* Ensuite, je construis une liste de dictionnaires contenant les données de chaque article.
* Et la fonction renvoie une réponse JSON contenant les données des articles.

Il n’y a pas de vue pour supprimer un article car on peut le faire directement avec Django admin

Et pour la partie HTML :

Dans mon fichier « add\_article.html » qui permet d’ajouter un article, il y a un formulaire. Ce formulaire est afficher grâce à la ligne « {{ form.as\_p }} » qui va chercher les champs qui ont été défini dans le back-end.

Superutilisateur

Pour finir je vais créer un superutilisateur, pour accéder à l’interface d’administration, avec la commande « python manage.py createsuperuser ». Cela me permettra de créer, modifier et supprimer des articles facilement avec l’interface de Django.

Pour accéder à cette page l’adresse sera « http://127.0.0.1:8000/admin »

Explication partie Node.js :

Je commence par créer une route « api/articles » dans mon serveur Node.js qui récupère les articles depuis mon API Django. J’utilise « axios » pour effectuer des requêtes http.

Une image contenant texte, capture d’écran

Description générée automatiquement

app.**use**(**cors**());  
Cela permet à mon serveur node d’accepter des requêtes provenant d’autres origines. Dans notre cas, cela permet d’appeler l’API Django

app.**get**('/api/articles', **async** (req, res) => {  
Je définie une route get à l’URL « /api/articles ». Quand cette route est appelée, la fonction asynchrone est exécutée.

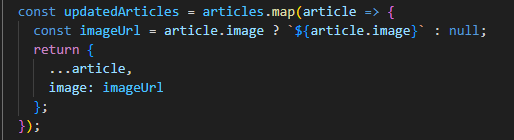
J’utilise une fonction asynchrone car elle permet d'effectuer des appels réseau (comme la requête à l'API externe http://127.0.0.1:8000/api/articles/) sans bloquer le serveur node. Cela signifie que pendant que le serveur node attend la réponse de l'API, il peut continuer à traiter d'autres requêtes.

**const** response = **await** axios.**get**('http://127.0.0.1:8000/api/articles/');  
Ensuite, j’utilise axios pour faire une requête GET à une autre API. Cela me retourne la liste de tous les articles qui sont dans l’API django

Dans notre projet le but est d’aller chercher les informations que l’on a dans la base de données de Django. Pour ce faire je vais utiliser le lien Django où sont renseigner toutes les informations « http://127.0.0.1:8000/api/articles/ ».   
Mon serveur Node.js va chercher les informations des articles dans le serveur de Django avec « const response = await axios.get('http://127.0.0.1:8000/api/articles/'); », ce qui va permettre à Django de lui envoyer.

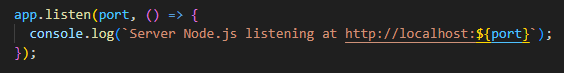
Une image contenant texte, capture d’écran, conception

Description générée automatiquement

  
On vérifie si chaque article récupéré à une image. Si oui, il la garde, sinon il la met à « null »

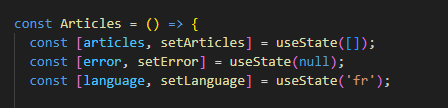
  
Ensuite si une image est bien présente dans l’article, le code essaye de faire une requête GET à l’URL de l’image pour vérifier si elle est accessible. Si elle est accessible il l’affiche dans la console sinon il affiche son erreur.

res.**json**(updatedArticles);  
Une fois que tous les articles ont été traités, le serveur node envoie la liste mise à jour des articles en réponse à la requête.

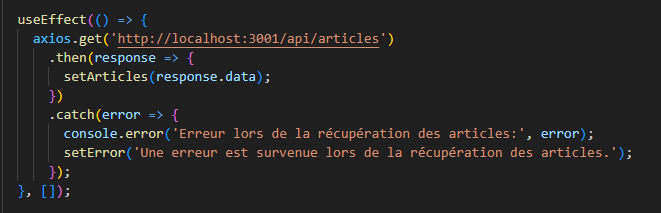
  
Le serveur node commence à écouter sur le port défini (3001) et affiche un message dans la console pour indiquer qu'il est en cours d'exécution.

Explication partie react :

Pour finir avec react je vais récupérer les données de l’API de node.

  
Je commence par ajouter la fonction Article pour pouvoir l’utilisé où je veux sur mon site.

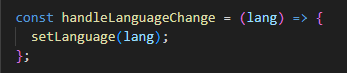
Ensuite je crée   
- un tableau pour récupérer les articles « const [articles, setArticles] = useState([]); »,   
- un état pour gérer les erreurs « const [error, setError] = useState(null); »,  
- une fonction pour mettre à jour les changement de langue « const [language, setLanguage] = useState('fr'); »

  
Je fais un hook qui va me permettre de récupérer les données. Dans ce hook :

- avec axios je fais une requête GET à l’URL de l’api de node « http://localhost:3001/api/articles » pour récupérer les articles traiter. Ce qui me permettra par la suite de pouvoir faire la mise en page des articles sur la page.  
Dans mon projet, react va demander les articles au serveur de Node.js et celui-ci va les lui transmettre.  
Une image contenant texte, capture d’écran, ordinateur, conception

Description générée automatiquement

- ensuite je mets à jour l’état « articles » avec les données récupérées. « setArticles(response.data); »  
- Et je rajoute catch pour gérer les erreur qui peuvent arrivé pendant la récupération des articles. « .catch(error => {«

Ensuite je fais une fonction pour le changement de langue.  
  
handleLanguageChange est une fonction qui met à jour l'état language en fonction de la langue sélectionnée (français ou anglais).

  
Je retourne le titre de la page, deux boutons pour choir la langue et une liste d’article où chaque article est affiché avec un titre, une image et une description, en fonction de la langue sélectionnée.

Annex

Schéma de transmission d’information en fonction de la demande :   
Une image contenant texte, capture d’écran, conception

Description générée automatiquement

Django :

Settings.py :   
INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'articles',

]

Models.py :  
from django.db import models

class Article(models.Model):

    TAG\_CHOICES = [

        ('écologie', 'Écologie'),

        ('politique', 'Politique'),

        ('numérique', 'Numérique'),

        ('santé', 'Santé'),

    ]

    title\_fr = models.CharField(max\_length=200)

    title\_en = models.CharField(max\_length=200)

    image = models.ImageField(upload\_to='images/', null=True, blank=True)

    description\_fr = models.TextField()

    description\_en = models.TextField()

    tags = models.CharField(max\_length=50, choices=TAG\_CHOICES, default='écologie')

    def \_\_str\_\_(self):

        return self.title\_fr

forms.py :   
from django import forms

from .models import Article

class ArticleForm(forms.ModelForm):

    class Meta:

        model = Article

        fields = ['title\_fr', 'title\_en', 'image', 'description\_fr', 'description\_en', 'tags']

views.py  
from django.shortcuts import render, redirect

from django.http import JsonResponse

from .forms import ArticleForm

from .models import Article

# Vue pour ajouter un nouvel article

def add\_article(request):

    if request.method == 'POST':

        form = ArticleForm(request.POST, request.FILES)

        if form.is\_valid():

            form.save()

            return redirect('home')

    else:

        form = ArticleForm()

    return render(request, 'articles/add\_article.html', {'form': form})

# Vue pour récupérer les articles au format JSON

def articles\_json(request):

    articles = Article.objects.all()

    base\_url = request.build\_absolute\_uri('/')

    data = [

        {

            'id': article.id,

            'title\_fr': article.title\_fr,

            'title\_en': article.title\_en,

            'image': f"{base\_url}{article.image}" if article.image else None,  # Concaténation de l'URL de base avec le chemin de l'image

            'description\_fr': article.description\_fr,

            'description\_en': article.description\_en,

            'tags' : article.tags

        }

        for article in articles

    ]

    return JsonResponse(data, safe=False)

# Vue pour supprimer un nouvel article

def supp\_article(request):

    if request.method == 'POST':

        form = ArticleForm(request.POST, request.FILES)

        if form.is\_valid():

            form.save()

            return redirect('home')

    else:

        form = ArticleForm()

    return render(request, 'articles/supp\_article.html', {'form': form})

# Vue pour l'accueil

def home(request):

    articles = Article.objects.all()

    return render(request, 'articles/home.html', {'articles': articles})

add\_article.html  
{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <link rel="stylesheet" href="{% static 'css/styles.css' %}">

    <title>Ajouter un article</title>

</head>

<body>

    <h1>Ajouter un nouvel article</h1>

    <form method="post" enctype="multipart/form-data">

        {% csrf\_token %}

        {{ form.as\_p }}

        <button type="submit">Ajouter</button>

        <a href="{% url 'home' %}" class="btn btn-primary">Retour à l'accueil</a>

    </form>

</body>

</html>

Une image contenant capture d’écran, texte

Description générée automatiquement

Node

Server.js  
const express = require('express');

const axios = require('axios');

const cors = require('cors');

const app = express();

const port = 3001;

app.use(cors());

app.get('/api/articles', async (req, res) => {

  try {

    const response = await axios.get('http://127.0.0.1:8000/api/articles/');

    console.log(response.data);

    const articles = response.data;

    // Vérifiez si chaque article à une image

    const updatedArticles = articles.map(article => {

      const imageUrl = article.image ? `${article.image}` : null;

      return {

        ...article,

        image: imageUrl

      };

    });

    // Vérifiez si chaque image est accessible

    for (const article of updatedArticles) {

      if (article.image) {

        try {

          await axios.get(article.image);

          console.log(`Image accessible: ${article.image}`);

        } catch (error) {

          console.error(`Image n'est pas accésible: ${article.image}`, error.message);

        }

      }

    }

    res.json(updatedArticles);

  } catch (error) {

    console.error(error);

    res.status(500).send('Erreur lors de la récupération des articles');

  }

});

app.listen(port, () => {

  console.log(`Server Node.js listening at http://localhost:${port}`);

});

Package.json  
{

  "name": "server",

  "version": "1.0.0",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "description": "",

  "dependencies": {

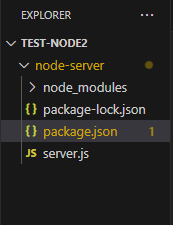
    "axios": "^1.7.9",

    "cors": "^2.8.5",

    "express": "^4.21.2"

  }

}



React

Articles.js  
import React, { useEffect, useState } from 'react';

import axios from 'axios';

import './css/Articles.css';

const Articles = () => {

    const [articles, setArticles] = useState([]);

    const [error, setError] = useState(null);

    const [language, setLanguage] = useState('fr');

    const [currentIndex, setCurrentIndex] = useState(0);

    useEffect(() => {

        axios.get('http://localhost:3001/api/articles')

            .then(response => {

                setArticles(response.data);

            })

            .catch(error => {

                console.error('Erreur lors de la récupération des articles:', error);

                setError('Une erreur est survenue lors de la récupération des articles.');

            });

    }, []);

    const handleLanguageChange = (lang) => {

        setLanguage(lang);

    };

    const nextArticle = () => {

        setCurrentIndex((prevIndex) => (prevIndex + 1) % 3);

    };

    const prevArticle = () => {

        setCurrentIndex((prevIndex) => (prevIndex - 1 + 3) % 3);

    };

    useEffect(() => {

        const interval = setInterval(() => {

            nextArticle();

        }, 5000);

        return () => clearInterval(interval);

    }, []);

    return (

        <div>

            <header>

                <h1 id='titre'>Bienvenue sur Euronews</h1>

                <div id='bouton'>

                    <button className="button" onClick={() => handleLanguageChange('fr')}>Français</button>

                    <button className="button" onClick={() => handleLanguageChange('en')}>Anglais</button>

                </div>

            </header>

            <h1 id='Articles'>A la une</h1>

            <body>

                {error && <p style={{ color: 'red' }}>{error}</p>}

                <div id='allarticles'>

                    <h1 id='troisnews'>Nos 3 plus anciennes news</h1>

                    {/\* caroussel \*/}

                    {articles.length > 0 && (

                        <div className="carousel">

                            <button onClick={prevArticle}>&lt;</button>

                            <div className="carousel-content">

                                {articles.slice(0, 3).map((article, index) => (

                                    <div

                                        key={index}

                                        className={`carousel-item ${index === currentIndex ? 'active' : ''}`}

                                        style={{ display: index === currentIndex ? 'block' : 'none' }}>

                                        <div className="carousel-container">

                                            <img id='imagecaroussel' src={article.image} alt={article.title}></img>

                                            <div className="text-container">

                                                <h2 id='titrecaroussel'>{language === 'fr' ? article.title\_fr : article.title\_en}</h2>

                                                <p id='textcaroussel'>{language === 'fr' ? article.description\_fr : article.description\_en}</p>

                                            </div>

                                        </div>

                                    </div>

                                ))}

                            </div>

                            <button onClick={nextArticle}>&gt;</button>

                        </div>

                    )}

                    <br /><br /><br /><br />

                    {/\* liste des articles \*/}

                    <ul className="articles-container">

                        {articles.slice(3).reverse().map((article, index) => (

                            <li key={index} className={index === 0 ? 'first-article' : 'other-articles'}>

                                {index === 0 && <h3 id='importante'>Dernière minute</h3>}

                                <div id='article'>

                                    <h2>{language === 'fr' ? article.title\_fr : article.title\_en}</h2>

                                    <img src={article.image} alt={article.title}></img>

                                    <p>{language === 'fr' ? article.description\_fr : article.description\_en}</p>

                                </div>

                            </li>

                        ))}

                    </ul>

                </div>

            </body>

        </div>

    );

};

export default Articles;

css :   
body {

  font-family: 'Helvetica', sans-serif;

  margin: 0;

  padding: 0;

}

header {

  display: flex;

  background-color: #0172f0;

}

#titre {

  color: #ffffff;

  margin-left: 15px;

}

.button {

  background-color: white;

  color: #0172f0;

  border: none;

  border-radius: 10px;

  padding: 10px 20px;

  font-size: 16px;

  cursor: pointer;

  transition: background-color 0.3s;

  margin-left: 15px;

  margin-top: 18px;

}

#Articles {

  margin-left: 15px;

}

.first-article {

  background-color: #0172f0;

  color: white;

  padding: 20px;

  border-radius: 15px;

}

#allarticles {

  margin-left: 15px;

  margin-right: 15px;

}

#article h2 {

  height: 58px;

  overflow: hidden;

  text-overflow: ellipsis;

  display: -webkit-box;

  -webkit-box-orient: vertical;

  -webkit-line-clamp: 2;

}

.articles-container {

    display: flex;

    flex-wrap: wrap;

    padding: 0;

    margin: 0;

    list-style-type: none;

  }

  li {

    box-sizing: border-box;

    padding: 10px;

  }

  .first-article {

    flex: 0 0 50%;

  }

  .other-articles {

    flex: 0 0 25%;

  }

    /\* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

  ARTICLE

  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \*/

  #article img {

    width: 100%;

    height: 300px;

    object-fit: cover;

  }

  #article p {

    height: 56px;

    overflow: hidden;

    text-overflow: ellipsis;

    display: -webkit-box;

    -webkit-box-orient: vertical;

    -webkit-line-clamp: 3;

  }

#importante {

    color: red;

}

  /\* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

  CAROUSEL

  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \*/

  #troisnews {

    display: flex;

    align-items: center;

    justify-content: center;

  }

  .carousel {

    display: flex;

    align-items: center;

    justify-content: center;

}

.carousel-content {

    display: flex;

    overflow: hidden;

    width: 80%;

}

.carousel-item {

    min-width: 100%;

    transition: transform 0.5s ease;

    opacity: 0;

}

.carousel-item.active {

    opacity: 1;

    background-color: #0172f0;

}

.carousel button {

    background-color: #0172f0;

    color: white;

    border: none;

    border-radius: 5px;

    padding: 10px;

    cursor: pointer;

    margin: 0 10px;

}

/\* mise en forme .carousel \*/

.carousel-container {

  display: flex;

  align-items: center;

}

#imagecaroussel {

  width: 40%;

  height: auto;

}

.text-container {

  padding-left: 20px;

  flex: 1;

}

#titrecaroussel {

  font-size: 24px;

  margin: 0;

  color: #ffffff;

  margin-right: 15%;

}

#textcaroussel {

  font-size: 16px;

  margin: 10px 0 0 0;

  color: #ffffff;

  margin-right: 15%;

}

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

Description générée automatiquement