Projet DEP (module déploiement de logiciel)

Table des matières

1. Présentation du projet

Ce document présente le projet de développement d'une application Django avec un frontend en HTML, CSS, et JavaScript. Le projet implique la gestion d'une base de données PostgreSQL pour stocker des articles générés, ainsi que l'intégration d'une API d'intelligence artificielle via Ollama pour générer du contenu.

2. Objectifs du projet

L'objectif principal de ce projet est de créer une plateforme permettant de générer des articles en utilisant l'intelligence artificielle, de les stocker dans une base de données, et de les afficher sur une interface web. L'application doit permettre à l'utilisateur de consulter les derniers articles générés, ainsi que d'en générer de nouveaux.

3. Description du projet

3.1 Technologies utilisées

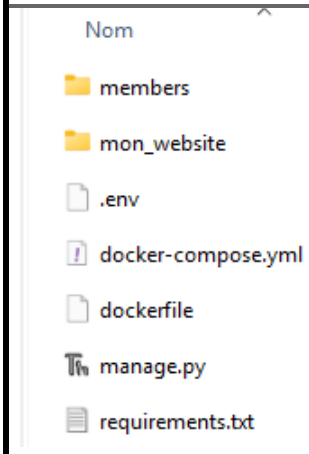
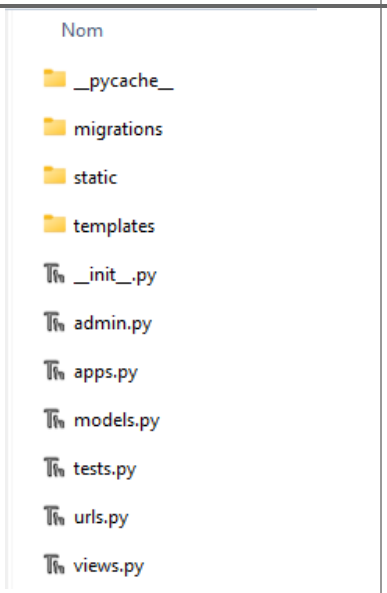
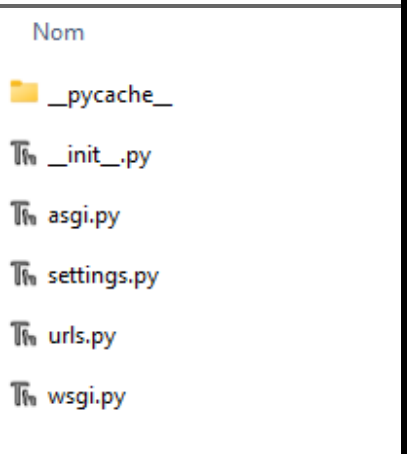
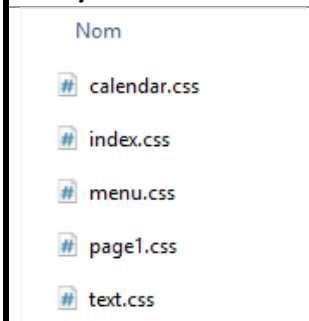
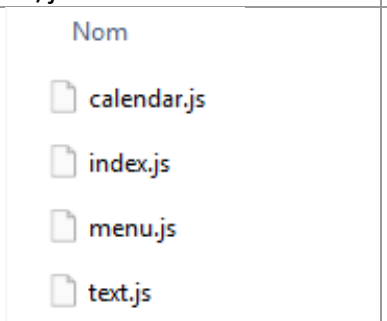
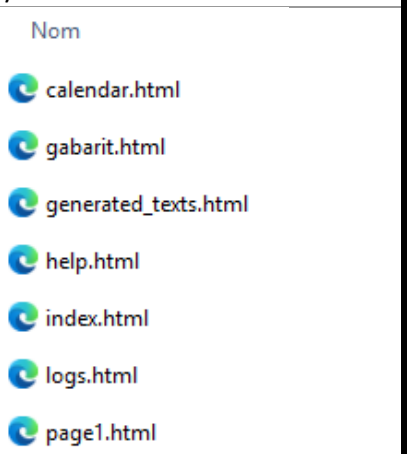
1. **Django (5.1.7)**: Framework Python utilisé pour le développement de l'application web. Il gère les vues, les modèles, et les interactions avec la base de données.

2. **PostgreSQL (15.12)** : Système de gestion de base de données utilisé pour stocker les articles générés. Chaque article est sauvegardé avec un titre, un sous-titre, une question, et divers mots-clés et thèmes.

3. **Ollama (0.5.13)** : API d'intelligence artificielle utilisée pour générer des articles en fonction des prompts fournis par l'utilisateur.

4. **Docker (27.4.0)** : Outil de conteneurisation qui permet d'encapsuler l'application et ses dépendances dans des conteneurs afin de faciliter le déploiement.

3.2 Structure du projet

mon_website/	**mon_website/members/**	**mon_website/mon_website/**
		
mon_website/members/static/css	mon_website/members/static/js	mon_website/members/templates/
		

3.3 Docker-Compose

Le fichier docker-compose.yml définit les services nécessaires à l'application, en gérant les environnements d'Ollama, PostgreSQL et Django.

```yaml
services:
  ollama_server:
    image: ollama/ollama
    container_name: ollama_server
    volumes:
      - ollama_data:/root/.ollama

  db:
    image: postgres:15
    container_name: postgres_db
    restart: always
    volumes:
      - postgres_data:/var/lib/postgresql/data
    env_file:
      - .env

  web:
    build: .
    container_name: django_app
    restart: always
    depends_on:
      - db
    volumes:
      - .:/app
    env_file:
      - .env
    command: >
      sh -c "python manage.py migrate &&
             python manage.py runserver 0.0.0.0:8000"

volumes:
  postgres_data:
  ollama_data:
```

ollama_server : Ce service utilise l'image Docker officielle d'Ollama (version 0.5.13) pour exécuter l'API d'intelligence artificielle.

db : Ce service utilise l'image postgres:15 de PostGreSQL. Le volume postgres_data permet de persister les données de la base de données. Le conteneur est configuré pour redémarrer automatiquement en cas d'échec.

web : Ce service est dédié à Django. Il est construit à partir du Dockerfile et est configuré pour redémarrer automatiquement. Il dépend du service de base de données pour fonctionner correctement.

3.4 Dockerfile

Le fichier dockerfile configure l'environnement de l'application django. Il installe les prérequis puis installe le client postgresql nécessaire pour communiquer avec la base de donnée.

```dockerfile
# Utilisation de l'image officielle Python
FROM python:3.11

# Définition du répertoire de travail
WORKDIR /app

# Copie des fichiers dans le conteneur
COPY requirements.txt .

# Installer les dépendances nécessaires
RUN apt-get update && \
    apt-get install -y postgresql-client

RUN pip install --no-cache-dir -r requirements.txt

# Copie du projet
COPY . .

# Définition de la commande de lancement
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

```

3.5 Variables d'environnement

Variables d'environnements présents dans .env :

```
POSTGRES_DB=mydatabase
POSTGRES_USER=postgres
POSTGRES_PASSWORD=secret
DB_HOST=db
DB_PORT=5432
OLLAMA_API_URL=http://ollama_server:11434
```

Ces variables d'environnements sont utilisés dans settings.py :

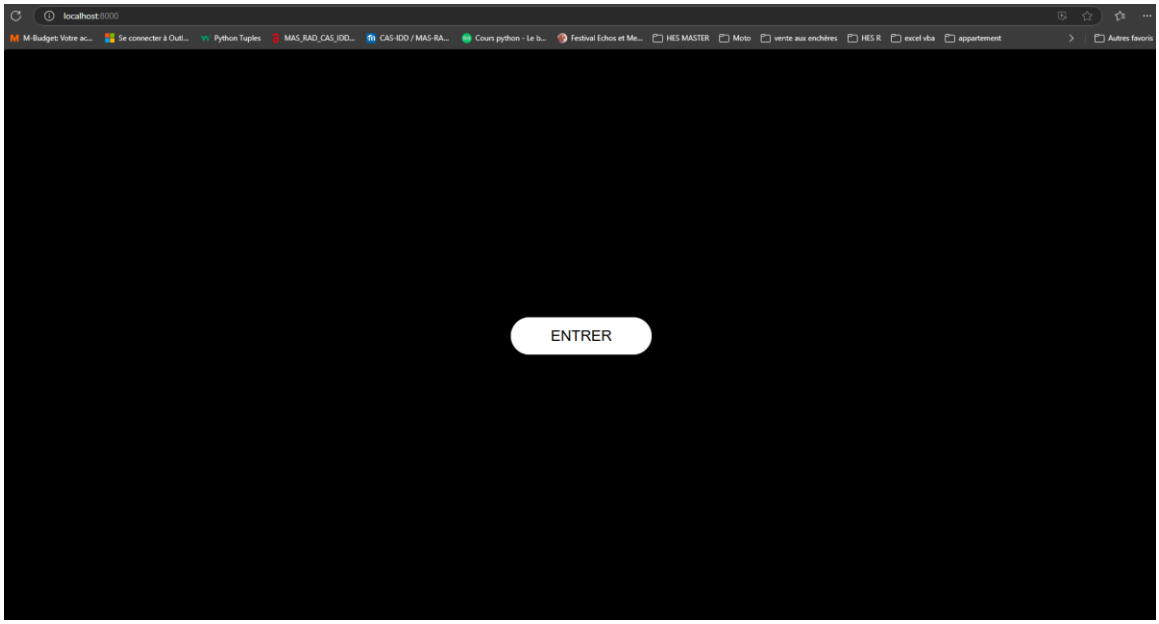
```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': get_env_variable('POSTGRES_DB', 'mydatabase'),  # Valeur par défaut 'mydatabase' si non définie
        'USER': get_env_variable('POSTGRES_USER', 'postgres'),  # Valeur par défaut 'postgres'
        'PASSWORD': get_env_variable('POSTGRES_PASSWORD', 'secret'),  # Valeur par défaut
        'HOST': get_env_variable('DB_HOST', 'db'),  # Valeur par défaut 'db' pour correspondre à docker-compose
        'PORT': get_env_variable('DB_PORT', '5432'),  # Valeur par défaut '5432'
    }
}
```

Ainsi que dans views.py :

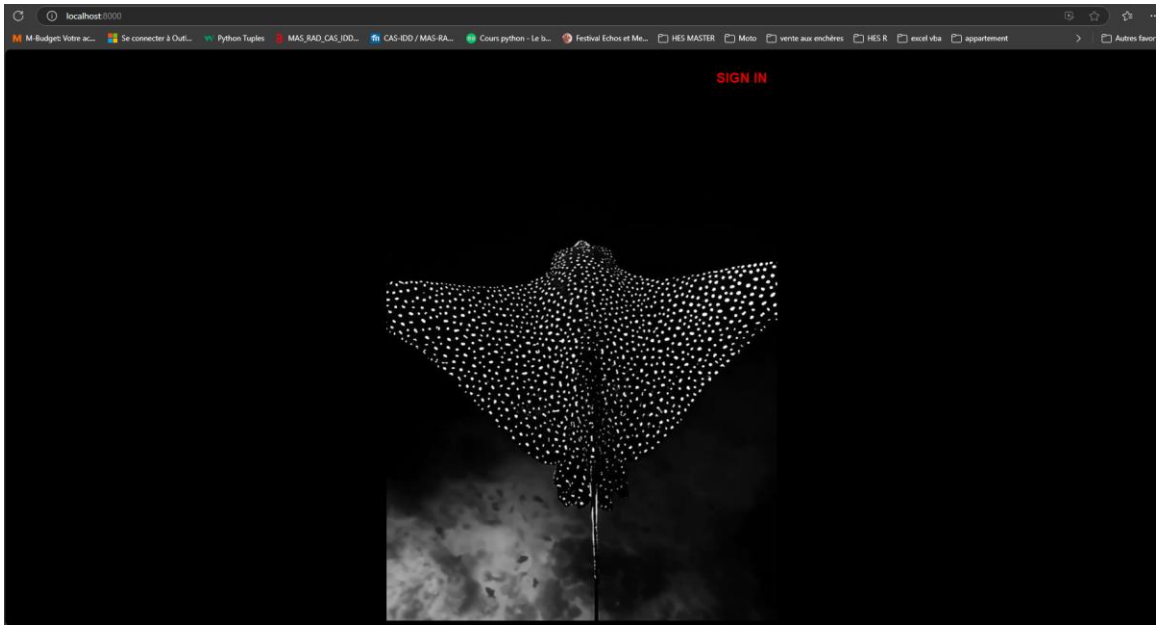
```python
OLLAMA_API_URL = os.getenv("OLLAMA_API_URL", "http://ollama_server:11434")
```

3.6 Fonctionnalités

Le site possède un menu de 6 onglets dont 4 sont utilisés. L'utilisateur doit d'abord appuyer sur entrer :



Une fois entré, les menus ne s'affichent pas directement. Ils défilent les uns après les autres à intervalles réguliers et ne deviennent visible que si la souris passe dessus. Sur le plan principal, une video montre une raie sombre entrain de nager dans la pénombre.

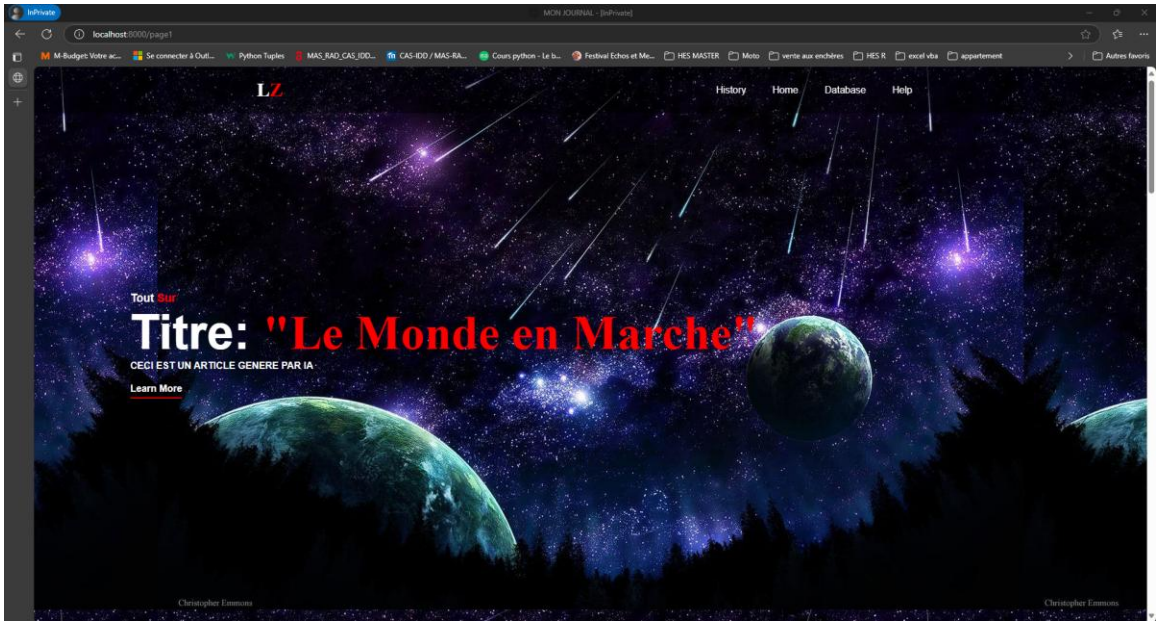


1. Onglet : « **New Ollama** » : Cet onglet permet de générer un article html dont les éléments suivants sont générés par l'intelligence articielle :
   - ai_title : Titre de l'article
   - *subtitle :* Sous-titre de l'article
   - *question :* Question en lien avec l'article
   - *com1 :* Commentaire en lien avec l'article

- *com2 :* Commentaire en lien avec l'article
- *com3 :* Commentaire en lien avec l'article
- *theme1 :* Thème en lien avec l'article
- *theme2 :* Thème en lien avec l'article
- *theme3 :* Thème en lien avec l'article

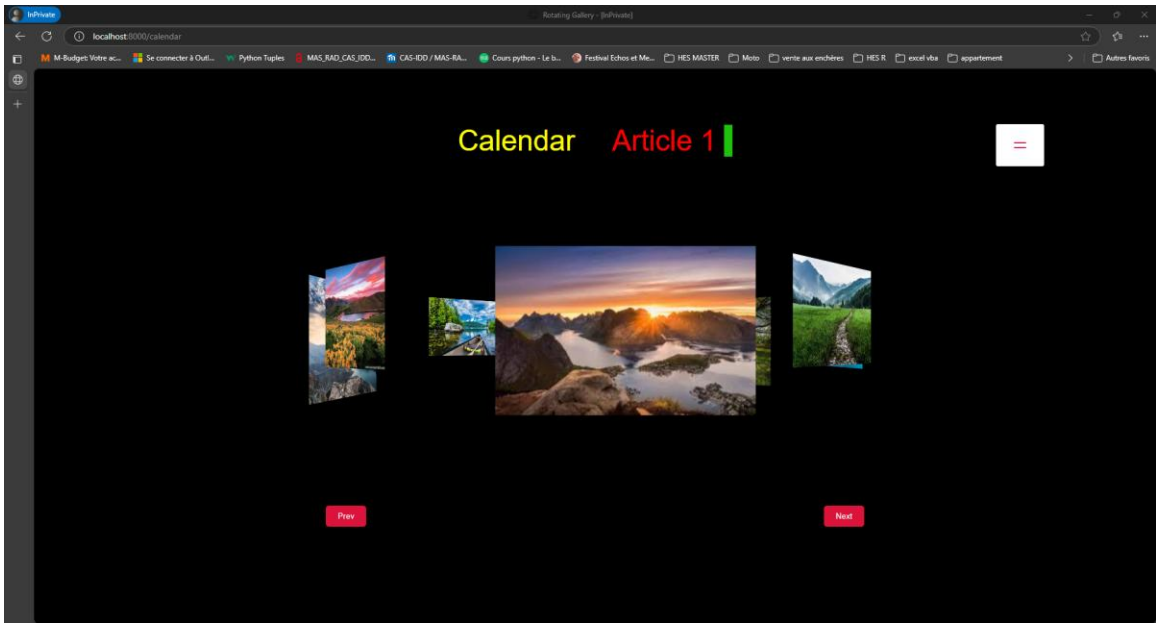
Remarque : La génération d'un nouvel article est lente (environ 3 minutes). Cette lenteur peut s'expliquer par le fait que le LLM prends du temps à démarrer. Aussi, chaque demande effectuée dans le tableau ci-dessus prends environ 10 à 20 secondes avant de recevoir une réponse.

Une fois généré, l'article prends la forme suivante :



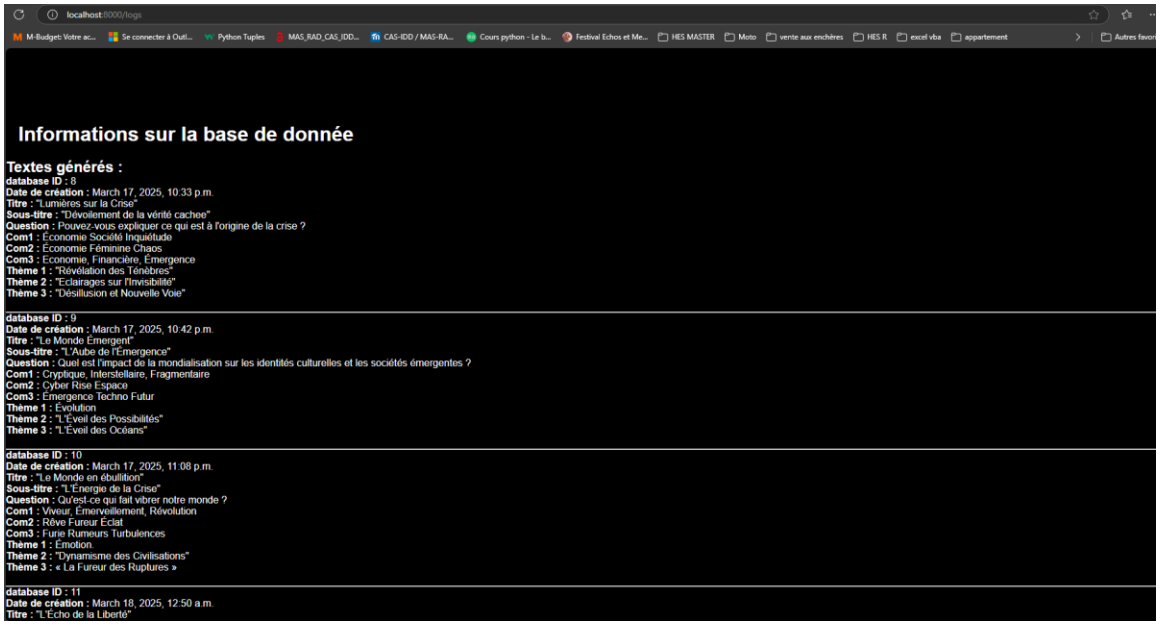
Il est toujours possible de revenir en arrière en utilisant les options du menu (« Home », « History », Database », « Help »).

2. Onglet « **History** » : Cet onglet affiche un carrousel avec des images de fleurs et un texte dynamique en javascript qui numérote les 8 derniers articles générés :

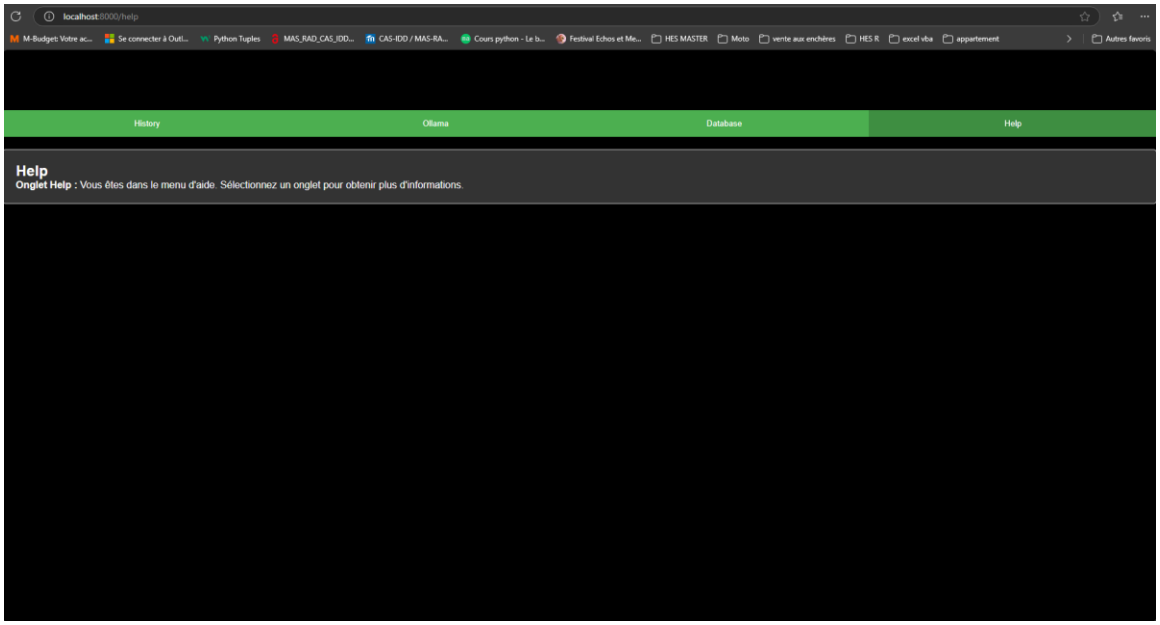


En cliquant sur une image de fleur, on ouvre un ancien article. Les articles sont affichés par ordre chronologique, les plus récents apparaissant en premiers. Si un nouvel article est généré, il écrase la plus ancienne entrée de la database. En cliquant sur le carré blanc avec les barres horizontales, les choix du menus sont proposés.

3. Onglet : « **Database** » : Cet onglet donne les informations présente dans la base de donnée.



4. Onglet : « **Help** » : Cet onglet représente l'aide utilisateur du site. En cliquant dans les onglets des sous-menus, il est possible d'obtenir de l'aide utilisateur.



5. Onglet : « **Sign In** » : Non attribué. Une future implémentation est en vue.
6. Onglet : « **Contact** » : Non attribué. Une future implémentation est en vue.

4. Conclusion

Ce projet a permis d'intégrer les concepts de containerisation avec docker, de développement web MVC (Modèle, Vue, Contrôle) avec Django, ainsi que l'utilisation de la base de donnée afin de sauver des informations de manière permanente et sécurisée.