Projet personnel

Language web
Site web
Rapport

Multilangue_site

Redigé par :
Haroun Bachar Amine

Table des matières

1 Introduction

1.1 Presentation du projet

Ce rapport présente le développement et le déploiement d'un site web Django multilingue intégrant des applications de modèles de langage (LLM) et une fonctionnalité de recherche augmentée par intelligence artificielle (RAG). Le projet a été conçu pour démontrer l'utilisation de Django pour créer une application web multilingue simple, avec la capacité d'intégrer des chatbots et des fonctionnalités de recherche avancées. Le site est déployé sur Render, une plateforme d'hébergement d'applications web.

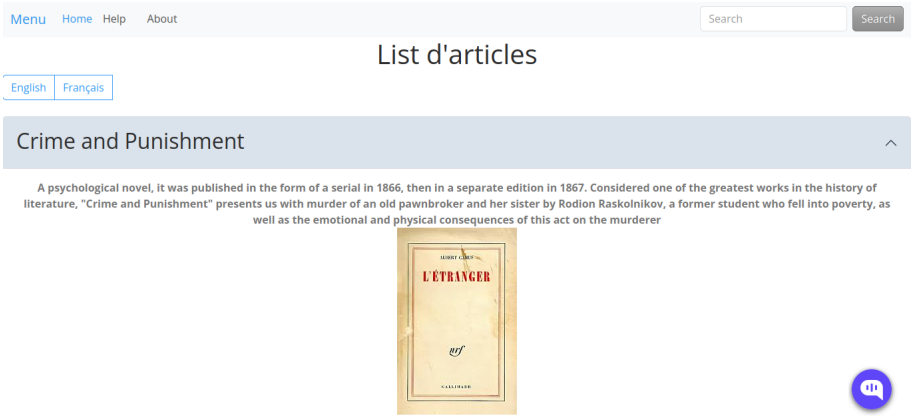


Figure 1 – Interface du projet

2 Technologies Utilisées

Django : Framework web pour construire l'application .
HTML/CSS : Pour la structure et le style des pages web .

JavaScript : Pour les interactions dynamiques et le changement de langue .
Dialogflow Kommunicate : Pour l'intégration du chatbot .
Docker : Pour la containerisation de l'application .
Render.com : Pour le déploiement de l'application .

2.1 Base de données

Figure 2 – BDD PostgreSQL

La base de données PostgreSQL est configurée pour interagir avec l'application, facilitant la gestion des entités .Ce choix etait fait car render propose des bases maintenus par le site gratuitement.

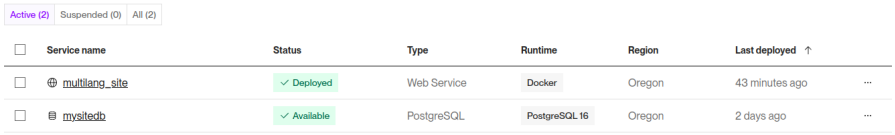


Figure 3 – interface render

2.2 Modèles et Vue de Base

Modèle pour les Articles de Blog : Un modèle Article a été créé dans main/models.py avec les champs title, content, et publication_date.

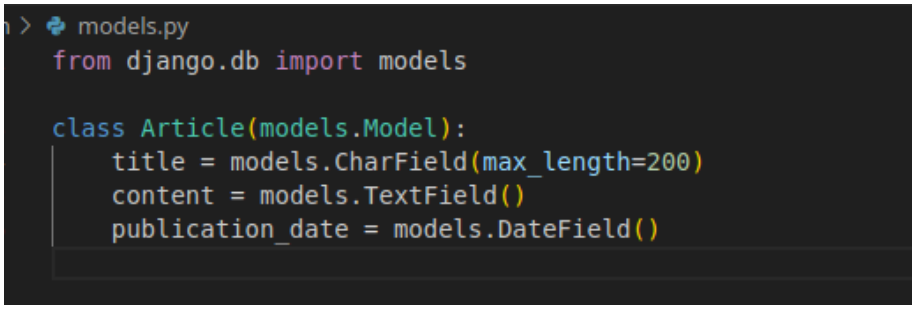


Figure 4 – modèle_article

Vue pour Afficher les Articles : Une vue article_list a été ajoutée dans main/views.py pour afficher une liste d'articles.

```python
def article_list(request):
    articles = Article.objects.all()
    return render(request, 'article_list.html', {'articles': articles})
```

Figure 5 – Article_list

3 Internationalisation

Configuration de l'Internationalisation (i18n) : Le projet est configuré pour supporter l'internationalisation (le francais et l'anglais pour le moment) en ajoutant les paramètres nécessaires dans settings.py.

```python
# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

LANGUAGES = [
    ('en', 'English'),
    ('fr', 'French'),
]
USE_I18N = True  # Active l'internationalisation
USE_L10N = True  # Active la localisation

LOCALE_PATHS = (
    os.path.join(BASE_DIR, 'locale'),
)
```

Figure 6 – Internationalisation

Traduction des Éléments Statistiques : Les éléments statiques de l'application ont été traduits en ajoutant des fichiers de traduction et en utilisant le balisage {% trans %} dans les templates.

4 Interface utilisateur

Templates pour les Articles de Blog : Des templates ont été créés pour afficher les articles de blog. Par exemple, article_list.html pour la liste des articles.

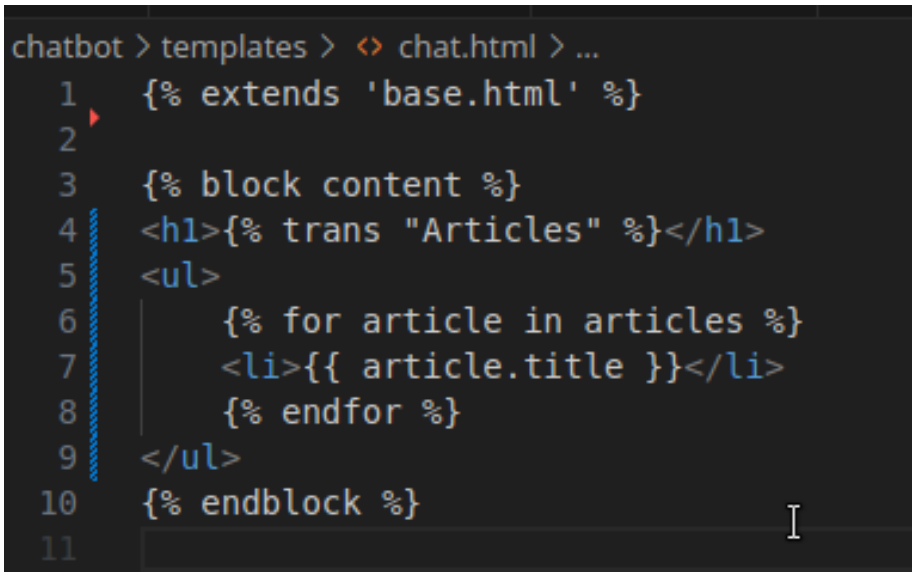


Figure 7 – blog d'articles

Changement de Langue : : Une fonctionnalité permettant aux utilisateurs de changer la langue de l'interface a été ajoutée dans le template base.html.

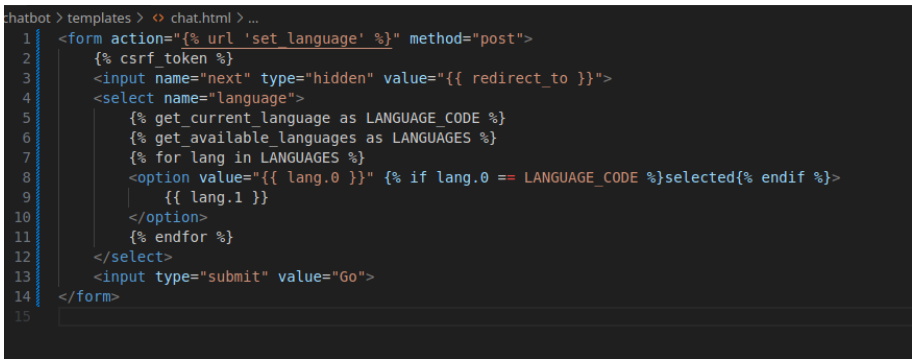


Figure 8 – changement de langues

4.1 Page d'acceuil

HomeController : Affiche la page d'accueil du projet, incluant le nom du projet, la version, les membres de l'équipe, et le logo de l'Université..

4.2 Page d'aide

HelpController : Fournit une page d'aide qui liste les opérations disponibles dans le service REST, avec des détails sur les URL, les méthodes attendues et un résumé des opérations.

5 Utilisation de Modèles de Langage (LLM) et RAG

Intégration d’un Chatbot :

Figure 9 – DialogueFLow et communicate

Initialement, une API GPT a été utilisée pour le chatbot, mais en raison des coûts, elle a été remplacée par DialogFlow, qui a été intégré avec Kommunicate pour la gestion des conversations,pour fournir des réponses basées sur un corpus de documents le chatbot est entrainer à repondre sur un echantillons limité.

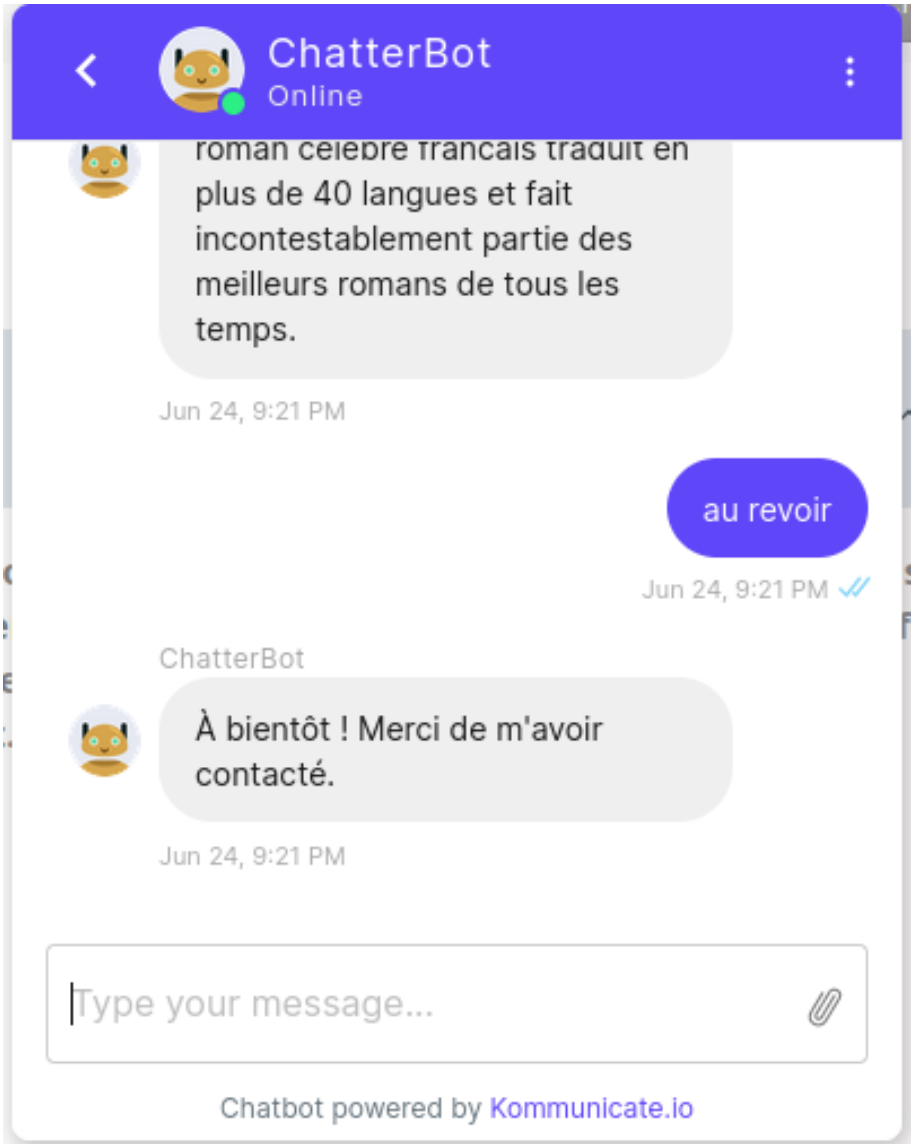


Figure 10 – chatterBot

6 Déploiement et Maintenance

Pour déployer l'application, nous utilisons Render.com. Voici les fichiers de configuration nécessaires.

Fichier render.yaml : Le fichier render.yaml configure le service web pour déployer l'application sur Render. Il spécifie les paramètres tels que le type de service, l'environnement et les commandes de build et de démarrage.

render.yaml

```yaml
services:
  - type: web
    plan: free
    name: multilang_site
    runtime: docker
    dockerfilePath: ./Dockerfile
    envVars:
      - key: DATABASE_URL
        fromDatabase:
          name: mysitedb
          property: connectionString
      - key: SECRET_KEY
        generateValue: true
      - key: WEB_CONCURRENCY
        value: 1
    buildCommand: ""
    startCommand: ""
```

Figure 11 – render.yaml

Fichier Dockerfile : Le Dockerfile définit l'environnement de conteneur pour l'application. Il installe les dépendances système, configure Python et installe les dépendances du projet.

```dockerfile
# Copy the current directory contents into the container at /app
COPY . /app

# Upgrade pip, setuptools, and wheel
RUN pip install --upgrade pip

# Install Cython first
RUN pip install "cython<3.0.0"
RUN pip install --no-build-isolation pyyaml==5.4.1

# Install Python dependencies
RUN pip install -r requirements.txt

# Collect static files
RUN python manage.py collectstatic --no-input

# Apply database migrations


RUN python manage.py migrate

# Make port 8000 available to the world outside this container
EXPOSE 8000

# Run the application
CMD ["gunicorn", "multilang_site.wsgi:application", "--bind", "0.0.0.0:8000"]
```

Figure 12 – DockerFile

7 Documentation

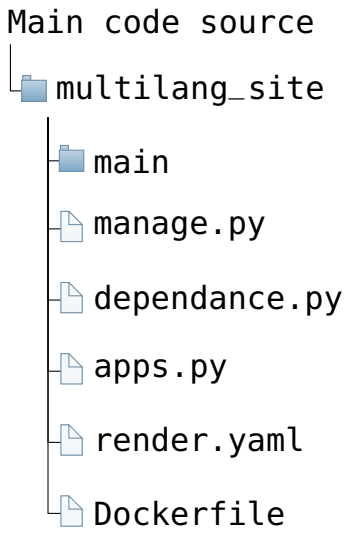
Instructions pour Exécuter le Projet : Un fichier README.md a été fourni avec des instructions détaillées pour installer les dépendances, configurer la base de données, et exécuter le projet.

Lien du depot github : https ://github.com/Amineharoun3/multilingue_rep
Lien du projet hebergé sur render : https ://multilang-site-92pf.onrender.com

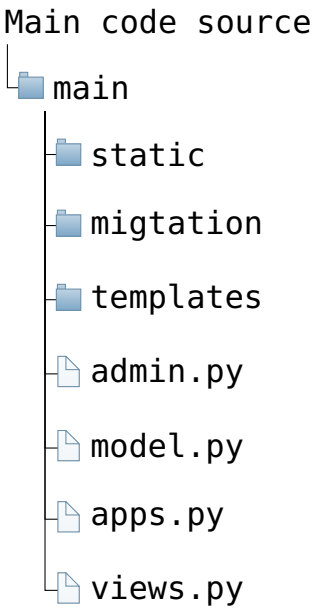
Site consulté pour le deploiement : hhttps ://medium.com/django-unleashed/deploy-your-django-app-with-ease-a-step-by-step-guide-with-render-810ccbf49573

8 Répartition des dossiers

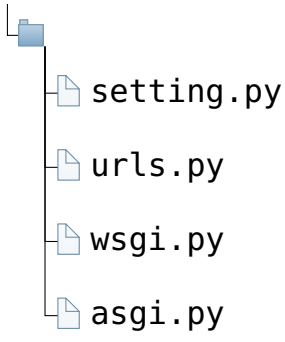
8.1 fichier principale du projet



8.2 fichier de l'application main



8.3 fichier multilang_site



9 Organisation travail

Pendant ce projet, j'ai eu l'occasion d'utiliser chatgpt durant la phase de développement de l'interface pour me familiariser avec Django et pour le deploiement de l'application sur render . Voici un résumé de mon investissement en temps pour les différentes phases :

Phase de développement de l'interface : J'ai consacré environ 6 heures pour me familiariser avec Django et développer l'interface utilisateur. Cela m'a permis de comprendre les principaux concepts et de mettre en place les fonctionnalités de base nécessaires pour le projet.

Développement du chatbot : J’ai également passé environ 6 heures à travailler sur le chatbot, intégrant les fonctionnalités nécessaires et assurant son bon fonctionnement.

Phase de déploiement : J’ai passé environ 8 heures sur la phase de déploiement du projet sur Render, où j’ai résolu divers problèmes de dépendances Python et assuré que le projet était correctement configuré pour être mis en production.