**Documentation Technique de SoWeChat**

**Installation de l'environnement de développement**

* + Cloner le dépôt :   
    git clone https://github.com/votre-repo/SoWeChat\_django.git cd SoWeChat\_django
  + Créer et activer un environnement virtuel :   
    python3 -m venv venv   
    source venv/bin/activate
  + Installer les dépendances :  
    pip install --upgrade pip   
    pip install -r requirements.txt
  + Créer un fichier .env à la racine du projet avec les clés API :  
    PINEKEY=your\_pinecone\_api\_key   
    OPENKEY=your\_openai\_api\_key
  + Appliquer les migrations :  
    python manage.py makemigrations  
    python manage.py migrate

#### Architecture Applicative

L'application est structurée comme suit :

* SoWeChat\_django/ : Répertoire principal de l'application.
* conversations/ : Application Django contenant les modèles, vues, URLs, et templates liés aux conversations.
* static/ : Répertoire des fichiers statiques (CSS, JS).
* templates/ : Répertoire des templates HTML.

Description des dossiers principaux :

* SoWeChat\_django/ : Contient les configurations principales du projet Django, y compris les paramètres, les configurations URL, et les fichiers de gestion.
  + manage.py : Script principal pour interagir avec le projet Django (exécuter le serveur, les migrations, etc.).
  + settings.py : Fichier de configuration principal du projet.
  + urls.py : Fichier de configuration des routes principales du projet.
* conversations/ : Inclut les modèles de données, les vues, les formulaires, les utilitaires, et les templates spécifiques à la gestion des conversations.
  + models.py : Définit les modèles Conversation et Message utilisés pour stocker les données des conversations.
  + views.py : Contient les vues pour gérer la création de conversations, l'affichage des détails des conversations et les interactions de l'utilisateur.
  + urls.py : Définit les routes spécifiques à l'application conversations.
  + forms.py : Définit les formulaires pour saisir les messages des utilisateurs.
  + utils.py : Contient des fonctions utilitaires pour interagir avec OpenAI et Pinecone.
* static/ : Héberge les fichiers statiques tels que les CSS, les JavaScript, et les images.
* templates/ : Contient les fichiers HTML utilisés pour le rendu des vues.

#### Dépendances

Les dépendances listées dans le fichier requirements.txt :

* Django : Utilisé pour développer l'application web. Il fournit un cadre complet pour gérer les bases de données, les routages, les templates, etc.
* openai : Utilisé dans utils.py pour générer des réponses automatiques basées sur les modèles de langage d'OpenAI.
* pinecone-client : Utilisé dans utils.py pour stocker et rechercher des embeddings de texte, facilitant la gestion des conversations et la recherche contextuelle.
* python-dotenv : Utilisé pour charger les clés API et autres configurations sensibles depuis un fichier .env.
* sentry-sdk : Utilisé pour la surveillance des erreurs et des performances, aidant à détecter et résoudre les problèmes en temps réel.
* django-prometheus : Utilisé pour exporter les métriques de performance de l'application Django, permettant une surveillance efficace avec Prometheus.
* numpy : Utilisé dans utils.py pour manipuler des tableaux et calculer des statistiques sur les résultats de recherche.
* pytest-django : Utilisé pour écrire et exécuter des tests unitaires et d'intégration pour l'application Django.
* pytest-cov : Utilisé pour mesurer la couverture des tests, assurant que toutes les parties critiques du code sont testées.

#### Exécution des Tests

Pour exécuter les tests, utilisez la commande suivante : pytest

* test\_urls.py : Contient des tests pour vérifier que les routes de l'application sont correctement configurées.
* test\_views.py : Contient des tests pour vérifier le bon fonctionnement des vues de l'application.