Configuration de la Base de Données PostgreSQL

Ce document explique comment configurer la base de données PostgreSQL pour notre application de gestion immobilière.

Prérequis

Avant de commencer, assurez-vous que PostgreSQL est installé sur votre système. Si ce n'est pas le cas, vous pouvez l'installer en suivant les instructions ci-dessous.

Installation de PostgreSQL

Sur Ubuntu/Debian

```
sudo apt update
sudo apt install postgresql postgresql-contrib
```

Sur macOS (avec Homebrew)

```
brew install postgresql
```

Sur Windows

Téléchargez et installez PostgreSQL depuis le site officiel : https://www.postgresql.org/download/windows/

Étapes de Configuration

1. Création du fichier .env

Créez un fichier `.env` à la racine du projet pour stocker les variables d'environnement sensibles :

```
touch .env
```

Ajoutez les variables suivantes dans le fichier `.env` :

```
 # Configuration de la base de données
DEV_DATABASE_URL=postgresql://username:password@localhost/
gestion_immobilier_dev
TEST_DATABASE_URL=postgresql://username:password@localhost/
gestion_immobilier_test
DATABASE_URL=postgresql://username:password@localhost/gestion_immobilier

# Clé secrète pour la sécurité de l'application
SECRET_KEY=votre-cle-secrete-tres-difficile-a-deviner

# Configuration du serveur de mail (optionnel)
# MAIL_SERVER=smtp.example.com
# MAIL_PORT=587
# MAIL_USE_TLS=True
# MAIL_USERNAME=user@example.com
# MAIL_PASSWORD=password
# MAIL_DEFAULT_SENDER=noreply@example.com
```

Remplacez `username` et `password` par les identifiants que vous utiliserez pour accéder à votre base de données PostgreSQL.

2. Création de l'utilisateur et des bases de données PostgreSQL

Connectez-vous à PostgreSQL en tant qu'utilisateur postgres :

```
sudo -u postgres psql
```

Créez un nouvel utilisateur pour l'application :

```
CREATE USER username WITH PASSWORD 'password';
```

Créez les bases de données pour les différents environnements :

```
 CREATE DATABASE gestion_immobilier_dev;
CREATE DATABASE gestion_immobilier_test;
CREATE DATABASE gestion_immobilier;
```

Accordez tous les privilèges à l'utilisateur sur ces bases de données :

```
GRANT ALL PRIVILEGES ON DATABASE gestion_immobilier_dev TO username;
GRANT ALL PRIVILEGES ON DATABASE gestion_immobilier_test TO username;
GRANT ALL PRIVILEGES ON DATABASE gestion_immobilier TO username;
```

Quittez l'interface PostgreSQL :

```
\q
```

3. Vérification de la connexion à la base de données

Pour vérifier que la connexion à la base de données fonctionne correctement, nous allons créer un script Python simple :

```python
# test_db.py
from app import create_app, db

app = create_app('development')
with app.app_context():
    # Vérification de la connexion
    try:
        db.engine.connect()
        print("Connexion à la base de données réussie !")
    except Exception as e:
        print(f"Erreur de connexion à la base de données : {e}")
```

Exécutez ce script pour vérifier la connexion :

```
python test_db.py
```

Si tout est correctement configuré, vous devriez voir le message "Connexion à la base de données réussie !".

4. Initialisation des migrations de base de données

Une fois la connexion à la base de données vérifiée, nous pouvons initialiser les migrations avec Flask-Migrate :

```
flask db init
```

Cette commande crée un dossier `migrations` avec la configuration nécessaire pour gérer les migrations de base de données.

Résolution des problèmes courants

Erreur "role 'username' does not exist"

Si vous rencontrez cette erreur, cela signifie que l'utilisateur PostgreSQL n'a pas été créé correctement. Assurez-vous d'avoir exécuté la commande `CREATE USER` comme indiqué ci-dessus.

Erreur "password authentication failed for user 'username'"

Vérifiez que le mot de passe dans le fichier `.env` correspond bien à celui que vous avez défini lors de la création de l'utilisateur PostgreSQL.

Erreur "could not connect to server: Connection refused"

Assurez-vous que le service PostgreSQL est bien démarré :

```
 # Sur Ubuntu/Debian
sudo service postgresql status
sudo service postgresql start

# Sur macOS
brew services list
brew services start postgresql
```

Prochaines étapes

Une fois la base de données configurée, nous pourrons passer à l'implémentation des modèles de données avec SQLAlchemy, en suivant le schéma de base de données défini précédemment.