Alembic

Non, **Alembic** n'est pas strictement obligatoire pour le fonctionnement final de votre projet. Cependant, il est fortement recommandé pour gérer les migrations de base de données dans des projets utilisant SQLAlchemy. Si vous ne souhaitez pas utiliser Alembic, vous pouvez gérer les schémas et les modifications de base de données avec des scripts SQL manuels.

Qu'est-ce qu'Alembic ?

Alembic est un outil Python pour gérer les migrations de schéma de base de données lorsque vous utilisez SQLAlchemy comme ORM. Cela vous permet de :

- Ajouter, modifier ou supprimer des colonnes ou des tables sans perdre de données.
- Versionner votre schéma pour suivre les changements.
- Gérer des bases de données dans des environnements de développement, de test et de production.

Comment fonctionne Alembic ?

1. **Initialisation d'Alembic :**
    - Alembic nécessite un fichier de configuration (alembic.ini) et un répertoire pour stocker les versions des migrations.

```bash
alembic init migrations
```

Cela crée :

- Un fichier alembic.ini : configuration d'Alembic.
- Un dossier migrations/ : contient les versions des migrations.

2. **Configuration :**
    - Dans le fichier alembic.ini, configurez la chaîne de connexion à votre base de données :

```ini
sqlalchemy.url = sqlite:///instance/development.db
```

- Configurez également env.py dans le dossier migrations/ pour inclure vos modèles :

```python
from app import db  # Import de SQLAlchemy
from app.models import *  # Importer tous les modèles
target_metadata = db.metadata
```

3. **Générer une migration :**

- Lorsque vous apportez des modifications à vos modèles SQLAlchemy, générez une nouvelle migration.

```bash
alembic revision --autogenerate -m "Description de la modification"
```

Cela crée un fichier de migration dans migrations/versions/.

4. **Appliquer une migration :**
    - Pour appliquer les migrations et mettre à jour la base de données :

```bash
alembic upgrade head
```

5. **Rétrograder une migration :**
    - Pour revenir à une version précédente :

```bash
alembic downgrade -1
```

Avantages d'Alembic :

1. **Gestion des versions :** Vous pouvez revenir à un état antérieur du schéma de base de données si nécessaire.
2. **Automatisation :** Les fichiers de migration sont générés automatiquement à partir des modifications dans vos modèles.
3. **Collaboratif :** Plusieurs développeurs peuvent gérer des migrations en parallèle.
4. **Flexible :** Fonctionne avec différents SGBD (SQLite, MySQL, PostgreSQL).

Alternatives si vous n'utilisez pas Alembic :

1. **Scripts SQL Manuels :**
    - Créez des fichiers .sql pour créer, modifier ou supprimer les tables :

```sql
CREATE TABLE users (
    id CHAR(36) PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    is_admin BOOLEAN DEFAULT FALSE
```

);

```

2. **Exécution directe avec SQLAlchemy :**
    - Utilisez db.create_all() pour créer toutes les tables définies dans vos modèles SQLAlchemy :

```python

from app import db

db.create_all()

```

3. **Gestion manuelle des changements :**
    - Apportez les modifications au schéma directement dans votre base de données sans utiliser d'outil de migration.

Dois-je utiliser Alembic ?

- Si votre projet évolue souvent et que le schéma de base de données change régulièrement, **Alembic** est un excellent choix.
- Pour un projet unique avec un schéma statique, vous pouvez gérer les tables avec des scripts SQL manuels ou db.create_all().

Dans un projet professionnel, Alembic est souvent requis pour une meilleure gestion des versions et des environnements.
```