



Mise en place de la base de données avec SQLModel

Sommaire

- Mise en place de la base de données avec SQLModel
 - Sommaire
 - Connexion à une base de données
 - Utilisation de SQLite (tests locaux)
 - Utilisation d'une base MariaDB (projets réels)
 - Points importants :
 - Création des tables
 - Ce que fait cette ligne :

Connexion à une base de données

SQLModel utilise **SQLAlchemy** en arrière-plan pour établir la connexion avec une base de données relationnelle.

La connexion se fait via un **moteur** que l'on instancie avec `create_engine()` en lui passant une URL de connexion.

Cette URL dépend :

- du **type de base** utilisée (SQLite, PostgreSQL, MariaDB, etc.),
- des **identifiants et paramètres de connexion**.

Utilisation de SQLite (tests locaux)

```
from sqlmodel import create_engine

sqlite_url = "sqlite:///./database.db"
engine = create_engine(sqlite_url, echo=True)
```

- SQLite est une base **légère, fichier unique**.
- Elle ne nécessite **aucun serveur**, ni configuration complexe.
- Parfait pour :
 - les tests manuels en local,
 - les tests unitaires automatisés (`:memory:`),
 - les prototypages rapides.

☑ **Avantage** : portabilité totale, démarrage immédiat. ✗ **Limites** : performances et concurrence limitées, pas adaptée à un environnement de production.

Utilisation d'une base MariaDB (projets réels)

En production ou en développement sérieux, on utilise plutôt une base **serveur** comme PostgreSQL ou MariaDB/MySQL.

Voici un exemple avec **MariaDB**, très utilisée dans les stacks web (équivalent libre de MySQL) :

```
from sqlalchemy import create_engine
import os

# Connexion sécurisée avec des variables d'environnement
user = os.getenv("DB_USER", "user")
password = os.getenv("DB_PASSWORD", "password")
host = os.getenv("DB_HOST", "localhost")
port = os.getenv("DB_PORT", "3306")
db_name = os.getenv("DB_NAME", "mydb")

mariadb_url = f"mysql+pymysql://{user}:{password}@{host}:{port}/{db_name}"

engine = create_engine(mariadb_url, echo=True)
```

Points importants :

- Le préfixe `mysql+pymysql://` est nécessaire pour MariaDB/MySQL avec le driver `pymysql`.
- Les **informations sensibles** (mot de passe, hôte...) sont lues depuis des **variables d'environnement** avec `os.getenv(...)`. Cela évite de **stocker en dur** des identifiants dans le code source.

💡 On peut ensuite charger un fichier `.env` avec `python-dotenv` ou utiliser Docker pour injecter les variables au démarrage.

Création des tables

Une fois l'engine et les modèles définis, SQLAlchemy nous permet de **générer automatiquement les tables** associées dans la base existante :

```
from sqlalchemy import SQLAlchemy
from .models import *

SQLAlchemy.metadata.create_all(engine)
```

⚠ Il faut que les tables soient connus dans le fichier ou est exécuter la commande.

- Sinon, aucune table ne sera créé.

Ce que fait cette ligne :

- Inspecte tous les modèles qui héritent de `SQLModel` avec `table=True`
- Crée les tables correspondantes si elles n'existent pas
- Ne modifie **pas** les tables déjà en place

⚠ Il ne s'agit **pas d'un outil de migration** : si vous modifiez un modèle, il faudra soit :

- supprimer et recréer la base, ou
- utiliser un outil comme **Alembic** (hors de ce cours)