




Cours SQLAlchemy – Modélisation et accès aux données

Objectif

Ce module a pour but de te familiariser avec **SQLModel**, une bibliothèque Python moderne qui simplifie la **modélisation de données** et l'interaction avec une **base SQL**, tout en préparant l'intégration avec une API FastAPI.

SQLModel est construit sur :

- **SQLAlchemy** (ORM pour Python)
- **Pydantic** (validation et sérialisation de données)

 Il permet de **déclarer des modèles typés**, de **créer les tables correspondantes** et d'effectuer des opérations **CRUD** simplement et efficacement.

La partie sérialisation et validation des données sera approfondie dans le module suivant, dédié à **FastAPI**.

Sommaire du module

Fichier	Contenu principal
<code>00-installation-et-manipulation-de-la-bdd-avec-sqlmodel</code>	Installation de SQLAlchemy, des dépendances nécessaires et utilisation des cursors
<code>01-introduction-sqlmodel.md</code>	Présentation de SQLAlchemy, ses objectifs et ses avantages
<code>02-creeer-des-tables.md</code>	Déclaration de modèles SQLAlchemy (simple, one-to-many, M2M)
<code>03-mise-en-place-bdd.md</code>	Connexion à SQLite/MariaDB, gestion sécurisée et création des tables via l'engine
<code>04-requetage-simple.md</code>	Mise en place de la session et Requête simple avec SQLAlchemy
<code>05-requetage-one-to-many.md</code>	Requête One-to-Many avec SQLAlchemy, principe du Relationship / back_populates et gestion des suppressions
<code>06-requetage-many-to-many.md</code>	Requête Many-to-Many avec SQLAlchemy

Pré-requis

- Python ≥ 3.10 (pour `int` | `None` syntaxe)
- Connaissances de base en POO
- Notions de SQL (tables, clés primaires, relations)

Ce que tu vas apprendre

- ✓ Créer un moteur de base de données (SQLite, MariaDB)
- ✓ Définir des modèles typés (tables)
- ✓ Créer et relier des tables automatiquement
- ✓ Manipuler les données via des sessions
- ✓ Implémenter des relations `one-to-many` et `many-to-many`
- ✓ Réaliser les opérations CRUD (Create, Read, Update, Delete)
- ✓ Préparer une base propre à connecter à une future API REST

Et ensuite ?

La suite du cours portera sur **FastAPI**, où l'on utilisera les modèles `SQLModel` pour :

- recevoir des données côté API (`POST /...`)
- valider automatiquement les entrées
- retourner des objets JSON typés

Crédits

Ce module est basé sur la documentation officielle de [SQLModel](#), avec des exemples adaptés pour l'apprentissage pas-à-pas.