Connexion à la base de données avec SQLModel et injection de dépendances

FastAPI s'intègre naturellement avec **SQLModel**, permettant de gérer la persistance des données tout en conservant une architecture propre grâce aux **dépendances injectées** avec `Depends()`.

Sommaire

Créer une session SQLModel avec FastAPI

Pour interagir avec la base de données, on utilise une **session SQLModel** (`Session`).
Plutôt que d'ouvrir manuellement une session dans chaque route, on peut en créer une **fonction de dépendance**.

```
from fastapi import Depends
from sqlmodel import Session
from db import engine  # votre moteur SQLModel (par ex. SQLite, PostgreSQL, etc.)

# Dépendance de session
def get_session():
    with Session(engine) as session:
        yield session
```

On l'utilise ensuite dans les routes pour injecter automatiquement la session :

```
from models import User  # le modèle SQLModel

@app.post("/users/")
def create_user(user: User, session: Session = Depends(get_session)):
    session.add(user)
    session.commit()
    session.refresh(user)
    return user
```

☑ **Ce qu'il se passe ici** :

- La session est injectée automatiquement grâce à `Depends(get_session)`
- Elle est ouverte, utilisée, puis fermée proprement
- Le modèle `User` est ajouté à la base et retourné après commit

Injection de dépendances avec `Depends()`

`Depends()` permet à FastAPI d'injecter des **fonctions auxiliaires**, réutilisables dans plusieurs routes, comme :

- la session DB,
- un utilisateur authentifié,
- des paramètres de pagination,
- une vérification d'accès…

Avantages des dépendances dans FastAPI

- ☑ **Réutilisables** : une seule fonction, utilisée dans plusieurs routes
- ☑ **Testables** : faciles à mocker ou surcharger lors des tests
- ☑ **Découplées** : la logique métier est séparée de la gestion technique