Développement d'une application avec une API**

I. Introduction et concept d'une API

- Définition d'une API (Application Programming Interface)
- Types d'API : REST, GraphQL, SOAP (focus sur REST ici)
- Format de données : JSON, XML (focus sur JSON)
- Méthodes HTTP : GET, POST, PUT, DELETE, PATCH
- Codes de retour HTTP
- Cas d’usage : communication frontend/backend, microservices, etc.

II. Modélisation et manipulation de la base de données avec SQLModel

- Introduction à SQLModel (synthèse de SQLAlchemy + Pydantic)
- Création des modèles de données
- Création des tables
- Connexion à une base MariaDB (ou SQLite en local)
- Opérations CRUD de base
- Session de base et gestion du contexte

III. Création d'une API REST avec FastAPI

- Introduction à FastAPI
- Définir des endpoints (GET, POST, etc.)
- Liaison avec les modèles SQLModel
- Dépendances et injection (par ex. sessions DB)
- Validation et documentation automatique avec Pydantic
- Organisation du projet (routes, services, modèles, etc.)

IV. Tests automatisés avec PyTest

- Pourquoi tester ? Types de tests (unitaires, intégration)
- Introduction à `pytest` et `httpx`
- Fixtures (base de données temporaire, client de test)
- Tester des routes FastAPI
- Exemples de tests avec assertions
- Couverture de code et exécution continue (CI/CD)

V. Environnement de développement et déploiement avec Docker (maybe soon)

- Pourquoi utiliser Docker : portabilité, reproductibilité
- Écrire un `Dockerfile` pour une application Python
- Utilisation de `docker-compose` pour orchestrer plusieurs services (API + base de données)
- Gestion des volumes, réseaux, variables d’environnement
- Bonnes pratiques de développement avec Docker

VI. Exemple concret

- Création d'une API de gestion des commandes (CRUD)
    - FastAPI pour les endpoints
    - SQLModel pour la base de données
    - Pytest pour les tests
    - Docker pour le déploiement
- Exemples de requêtes et réponses
- Documentation de l'API
- Gestion des erreurs et des exceptions
- ...