# Exercice : Création d'une API de gestion d'école en ligne avec FastAPI

\*\*Exercice : Création d'une API de gestion d'école en ligne avec FastAPI\*\*  
  
### Objectif  
  
Cet exercice a pour but de vous aider à comprendre et à illustrer les relations entre différents modèles de données en utilisant FastAPI et SQLAlchemy. Vous allez créer une API de gestion d'une école en ligne qui inclut des relations \*\*1 à 1\*\*, \*\*1 à N\*\*, et \*\*N à N\*\*.  
  
---  
  
### Scénario  
  
Vous devez développer une API pour gérer les professeurs, les étudiants et les cours d'une école en ligne.  
  
L'API doit permettre :  
  
1. \*\*D'ajouter des professeurs et leurs profils détaillés (relation 1 à 1)\*\*.  
2. \*\*D'associer des cours à un professeur (relation 1 à N)\*\*.  
3. \*\*De permettre aux étudiants de s'inscrire à plusieurs cours (relation N à N)\*\*.  
  
---  
  
### Tâches  
  
#### 1. Initialisation du projet  
  
1. Créez un nouveau projet FastAPI.  
2. Configurez SQLAlchemy avec une base de données SQLite ou PostgreSQL.  
3. Créez un fichier `models.py` pour définir les modèles.  
4. Créez un fichier `main.py` pour définir les routes.  
  
#### 2. Relations \*\*1 à 1\*\* : Professeurs et Profils  
  
1. Définissez un modèle \*\*Teacher\*\* avec des informations de base (nom, email).  
2. Définissez un modèle \*\*Profile\*\* qui contient des informations détaillées sur les professeurs (biographie, domaine de spécialité).  
3. Créez une relation 1 à 1 entre les tables `teachers` et `profiles`.  
4. Ajoutez des routes pour :  
 - Créer un professeur avec un profil.  
 - Obtenir les détails d'un professeur et de son profil.  
  
#### 3. Relations \*\*1 à N\*\* : Professeurs et Cours  
  
1. Définissez un modèle \*\*Course\*\* qui inclut les informations de base d'un cours (titre, description).  
2. Créez une relation 1 à N entre les tables `teachers` et `courses` (un professeur peut enseigner plusieurs cours).  
3. Ajoutez des routes pour :  
 - Créer un cours et l'associer à un professeur.  
 - Obtenir tous les cours enseignés par un professeur.  
  
#### 4. Relations \*\*N à N\*\* : Étudiants et Cours  
  
1. Définissez un modèle \*\*Student\*\* qui inclut les informations de base d'un étudiant (nom, email).  
2. Créez une table intermédiaire pour gérer l'association entre les étudiants et les cours.  
3. Ajoutez une relation N à N entre les tables `students` et `courses` (un étudiant peut suivre plusieurs cours, et un cours peut être suivi par plusieurs étudiants).  
4. Ajoutez des routes pour :  
 - Inscrire un étudiant à un ou plusieurs cours.  
 - Obtenir tous les cours auxquels un étudiant est inscrit.  
 - Obtenir tous les étudiants inscrits à un cours.  
  
#### 5. Tests et Documentation  
  
1. Testez votre API avec \*\*Swagger\*\* (intégré à FastAPI).  
2. Créez des exemples de requêtes pour chaque route (POST, GET).  
3. Optionnel : Ajoutez des tests automatisés avec `pytest`.  
  
---  
  
### Consignes supplémentaires  
  
- Utilisez des schémas Pydantic pour valider les données envoyées à l'API.  
- Gérez les erreurs (étudiant ou professeur inexistant, associations invalides, etc.) en utilisant des exceptions HTTP appropriées (e.g., `HTTPException` avec `status\_code=404`).  
- Fournissez une route pour récupérer toutes les données liées à un professeur (son profil, ses cours, et les étudiants inscrits à ses cours).  
  
---  
  
### Exemple de structure de projet  
  
```  
project\_root/  
├── main.py # Points d'entrée de l'application FastAPI  
├── models.py # Modèles SQLAlchemy  
├── schemas.py # Schémas Pydantic  
├── database.py # Configuration de la base de données  
├── routers/ # Routes organisées par fonctionnalité  
│ ├── teachers.py  
│ ├── courses.py  
│ └── students.py  
└── tests/ # Tests unitaires avec pytest  
 └── test\_main.py  
```  
  
---  
  
### Livrables attendus  
  
1. Une API fonctionnelle avec les relations correctement implémentées.  
2. Une documentation Swagger complète.  
3. Un fichier README expliquant comment lancer l'application et tester les routes.  
  
Bonne chance !