**Exercice : Modélisation et Génération des Données dans Laravel 12**

**Objectif :**

Dans cet exercice, vous allez implémenter un ensemble de modèles avec leurs relations, migrations, factories et seeders en utilisant **Laravel 12**. L’objectif est de comprendre comment structurer et gérer une base de données relationnelle dans un projet Laravel.

**Important :** Vous devez réaliser cet exercice **sans utiliser d’intelligence artificielle**. L’objectif est de vous habituer à lire et comprendre la documentation officielle, qui est votre meilleure alliée en tant que développeur. Pour vous aider, vous pouvez consulter la page suivante :  
<https://laravel.com/docs/12.x/eloquent-relationships>

**Contexte :**

Vous devez concevoir un système de gestion de cours en ligne où :

1. **Les utilisateurs (users)** peuvent avoir différents rôles (role) comme **étudiant** ou **enseignant**.
2. **Les utilisateurs ont un profil (profiles)** et un profil est lié a un seul utilisateur contenant des informations personnelles (relation One-to-One).
3. **Les cours (courses)** sont créés par des enseignants (relation avec users via teacher\_id). Chaque cours contient un **titre, une description, un contenu**, ainsi que des informations sur le niveau, la durée et une image associée.
4. **L'inscription aux cours (course\_student)** permet de suivre l’état d’un étudiant dans un cours donné avec un **statut** (en cours, terminé, etc.) et un niveau de **progression** (en pourcentage).
5. **Les commentaires (comments)** permettent aux utilisateurs de commenter soit un **cours**, soit un autre user (utilisation du **polymorphisme** avec commentable\_type et commentable\_id).

**Travail demandé :**

1. **Créer les modèles** et définir les **relations** entre eux :

* Un utilisateur peut avoir un seul profil (One-to-One).
* Un **utilisateur** peut créer plusieurs **cours**.
* Un **utilisateur** peut s’inscrire à plusieurs **cours** et suivre sa **progression**.
* Un **commentaire** peut être attaché soit à un **cours**, soit à une **inscription** via une **relation polymorphique**.

1. **Créer les migrations** pour chaque table en respectant les contraintes suivantes :
   * Clés primaires et étrangères bien définies.
   * Les colonnes nécessaires pour chaque entité avec les bons types de données.
   * Ajout des timestamps (created\_at et updated\_at).
2. **Créer les factories** pour générer des données aléatoires et crédibles pour les entités suivantes :
   * **Users** : avec des rôles aléatoires (student ou teacher).
   * **Profiles :** générer des biographies, avatars et sites web aléatoires pour les utilisateurs.
   * **Courses** : assignés à des enseignants existants.
   * **CourseStudent** : générer des inscriptions aléatoires pour les étudiants.
   * **Comments** : des commentaires aléatoires sur les cours et les inscriptions.
3. **Créer les seeders** pour injecter des données de test en utilisant les factories.
4. **Tester le bon fonctionnement** des relations en exécutant les migrations et les seeders.

**Livrables attendus :**

* Les fichiers de **modèles** avec les relations bien définies.
* Les **migrations** de chaque table respectant les contraintes.
* Les **factories** pour générer des données réalistes.
* Les **seeders** permettant d’initialiser la base avec des données de test.