Générateur de Quiz

Contexte pro du projet

La **Miskatonic University**, prestigieuse université fictive du Massachusetts, souhaite mettre à disposition de ses enseignants un outil numérique pour faciliter la préparation de leurs évaluations.
L'objectif est de développer un **générateur de quiz en ligne** reposant sur deux briques :

- **Un serveur API** permettant de gérer les questions, de créer des quiz et d'assurer l'authentification des utilisateurs.
- **Un client web** (IHM simple) consommant cette API, afin de permettre aux enseignants de visualiser les questions, d'en ajouter et de générer un quiz aléatoire.

Le système devra s'appuyer sur deux bases de données distinctes :

- **MongoDB** pour stocker l'ensemble des questions.
- **SQLite** pour gérer les utilisateurs (comptes, mots de passe hachés, rôles).

Modalités pédagogiques

- **Organisation** : 7 binômes (14 étudiant·e·s).
- **Échéance** : 16 octobre 2025.
- **Travail demandé** :
    - Concevoir et développer les deux applications (**API** et **client web**).
    - Rédiger les **User Stories** (format INVEST, avec critères d'acceptation).
    - Prévoir un **schéma de base de données** clair pour les utilisateurs.
- **Approche** : travail collaboratif, itératif et incrémental. Les binômes devront partir des besoins exprimés et proposer leurs propres choix techniques et organisationnels.

Modalité d'évaluation

Un **oral de 10 minutes par binôme** :

- **5 minutes** de démonstration (présentation de l'IHM et appels API en direct).
- **5 minutes** pour le choix techniques, organisation, limites, pistes d'amélioration.

Livrables

Chaque binôme devra fournir dans un **dépôt GitHub unique** :

1. **Code complet** des deux applications (API et client web).
2. **Documentation OpenAPI** générée et enrichie.
3. **Template de document pour MongoDB**.
4. **MCD de la base utilisateurs SQLite** .
5. **User Stories** rédigées avec critères d'acceptation.
6. **Présentation orale** (diapositives au format PDF ou équivalent).
7. **README** décrivant l'installation, le lancement des applications et les jeux d'essai disponibles.

Critères de performance

- **Architecture claire** : séparation des responsabilités (routes, services, modèles, sécurité).
- **API sécurisée** : gestion correcte de l'authentification, stockage de mots de passe hachés, protection des routes sensibles.
- **Documentation OpenAPI complète**.
- **Bases de données cohérentes** :
    - MongoDB bien structurée pour les questions (documents complets).
    - SQLite correctement conçu pour les utilisateurs.
- **IHM fonctionnelle et lisible** : affichage clair des questions, formulaires de création, génération de quiz.
- **User Stories bien rédigées** : formulées selon le format attendu, avec des critères d'acceptation vérifiables.
