Evaluation - Workshop final - CDA C#

Objectif et notation

- L'objectif de cette évaluation est de pouvoir évaluer les compétences que je n'ai pas pu évaluer durant nos cours / workshop
- Les éléments qui seront évalués seront :
  - Façon de coder (en outre, clean code mais avec moins d'exigences)
  - Développement orienté objet
  - Développement d'une API
  - Développement avec une base de données. Ceci pouvant être fait de deux façons :
    - En SQL directement
    - Avec un ORM (comme Entity)
  - Développement de tests unitaires
- Les éléments abordés ensemble qui **NE seront PAS** notés :
  - DevOps
  - Identity
- A savoir, cette "évaluation" n'est là que pour m'aider à compléter les remarques que j'ai déjà prises sur vous au fur et à mesure de la formation. Pas de panique, ça va bien se passer.
- Si vous ne terminez pas, c'est loin d'être la fin du monde !

Règles de l'évaluation

Contexte :

- Vous êtes chargé de développer une application de gestion d'animaux Cette application doit permettre de gérer une liste d'animaux ainsi que leur race (Une et une seule race par animal). Une création de compte doit être effectuée. Seul un compte connecté peut accéder aux données. Au démarrage de l'application (console ou avec une interface), une personne doit obligatoirement se connecter OU créer un compte (choix simple entre les deux à proposer au user).

UI / UX

- Tout peut être réalisé en projet console ! Avec des choix 1 - ... (Et je ne serai pas très regardant sur la qualité des contrôles de saisies utilisateurs, même si c'est mieux d'en faire à ce niveau, ça pourra apporter un bonus)
- Rien ne vous empêche de faire une petite interface graphique si ça vous dit, mais ça n'apportera pas de point, même si c'est beau ! Il n'y aura pas de point bonus.

Fonctionnalités demandées :

- Créer un compte en BDD :
    - Ajout d'un nouveau compte (Email + Mot de passe, sans confirmation)
        - Bonus : hashage mot de passe efficace en BDD
    - Connexion d'un utilisateur (Email + Mot de passe)
    - **PAS DE GESTION DE RÔLES / DROITS DEMANDEE !**
    - La seule règle est : Un utilisateur a accès à toutes les données de la base, il doit donc être connecté pour accéder aux données.
- Gestion des animaux :
    - Ajouter un nouvel animal (Id, Nom, Description, RaceId)
    - Modifier les informations d'un animal par son ID (Nom, Description, Race)
    - Supprimer un animal par son ID
    - Lister tous les animaux (Pas de pagination demandée)

    - Rechercher un animal par sa race ou son nom (Seulement l'un des deux est à implémenter)

- Gestion des races :

    - Ajouter une nouvelle race (Id, Nom, Description)
    - Modifier une race par son ID (Nom, Description)
    - Lister toutes les races

Contraintes techniques :

- Façon de coder :
    - Respecter les principes de clean code (noms de variables explicites, méthodes courtes, etc.)
- Développement orienté objet :
    - Utiliser des classes et des objets pour modéliser les animaux et leur race.
- Développement d'une API :
    - Développer une API pour gérer les animaux et leur race.
        - Vous pouvez réaliser l'API via fastendpoint ou autre outil de création d'une API, la seule demande est que des endpoints soient accessibles en REST.
- Développement avec une base de données :
    - Les données doivent être stockées dans une base de données SQL (Vous pouvez choisir le SGBD utilisé, MySQL, SQL Server MS, ...)
    - Utiliser soit des requêtes SQL directes, soit un ORM comme Entity Framework pour accéder aux données. s- Développement de tests unitaires :
    - Minimum 5 tests unitaires attendus (Idéalement, 10 même !)
    - Écrire des tests unitaires pour les méthodes principales des classes et pour les contrôleurs de l'API.
    - Utiliser un framework de tests comme NUnit ou xUnit (ou autre si vous préférez)

Livrables attendus :

- Le code source de l'application, soit sous forme de zip avec votre **nom et prénom** dans le nom du zip (sans les dossiers /bin et /obj) soit directement avec un lien git

public (ici aussi sans les /bin et /obj).
- Le / Les scripts pour créer et peupler la base de données (si vous utilisez des requêtes SQL directes).
- Les tests unitaires.

Conseils :

- Pensez à gérer les erreurs et les exceptions de manière appropriée.
- Testez votre API avec des outils comme Postman pour vous assurer qu'elle fonctionne correctement.

Bonus

- Bonus à ceux qui font un Docker qui me permet de lancer le projet sans avoir à exécuter une ligne de SQL à la main