# EONIX

Pour tester vos compétences en programmation, nous souhaitons que vous écriviez deux petits programmes en C#, préparés chez vous. Nous vous conseillons d'utiliser Visual Studio pour ce faire, dont une version Express gratuite est disponible sur internet. Si vous ne disposez pas du matériel nécessaire, nous pouvons vous faire venir une journée dans nos locaux pour que vous les prépariez sur place.

1) Créer un programme console qui va décrire la situation suivante :
    a. Un spectateur croise deux dresseurs de singes. Chaque dresseur a un singe.
    b. Chaque singe connaît une liste de tours, qui ont chacun un nom et sont soit des tours d'acrobatie ou de musique
    c. Les dresseurs peuvent demander à leur singe d'exécuter un tour.
    d. Le spectateur applaudit lors de tours d'acrobatie ou siffle pendant les tours de musique.

    Avec les contraintes suivantes :

    a. Spectateur, singe et dresseur sont des classes.
    b. Le programme doit créer une instance du spectateur, deux instances de dresseurs et deux instances de singe
    c. Chaque dresseur demande à son singe d'exécuter tous ses tours.
    d. Le spectateur réagit aux tours.
    e. Le programme devra afficher des messages du type "spectateur applaudit pendant le tour d'acrobatie 'marcher sur les mains' du singe 2".
    f. Faites très attention au modèle objet et au couplage faible, le singe ne peut pas appeler la méthode Applaudit du spectateur !

2) Dans le cadre d'un projet d'application basée sur des micro-services, vous devez mettre en place un webservice REST/JSON permettant de gérer des personnes dans une base de données. Une personne est définie par les propriétés suivantes :
    a. Id : Guid
    b. Prénom : string
    c. Nom : string

    Pour qu'une personne soit considérée comme valide, son nom et son prénom doivent être non-vides.

    Le webservice doit présenter les endpoints CRUD classiques :

    a. Récupération d'une liste de personnes, avec filtres sur le nom et le prénom. Les filtres doivent être optionnels. Les filtres doivent être insensibles à la casse, et il doit être possible de faire une recherche sur le début ou la fin du nom/prénom (par exemple pour "Sébastien", "Séb" ou "tien" doit matcher).

b. Récupération d'une personne, sur base de son ID.
c. Ajout d'une personne
d. Mise à jour d'une personne
e. Suppression d'une personne

Il est demandé de suivre les standards en terme d'API REST pour la structuration des endpoints.

D'un point de vue technique, votre API doit utiliser ASP.net Core (dernière version en date) et Entity Framework Core pour interagir avec la base de données. Par facilité, nous vous conseillons d'utiliser SQLite comme moteur de DB. Votre code doit utiliser l'approche "code-first".

Vous devrez déposer votre code sur un repository Git, sur le serveur ou le service de votre choix, et accessible par nous. Merci de nous transmettre l'URL du repository par email avant l'entretien technique.

Vous présenterez votre solution en argumentant les choix d'implémentation. Pensez aux tests unitaires.