A23 – POO 4 - Travail pratique 3 (20%)

I - Objectifs de ce TP

Ce travail pratique (TP) vise à :

- Évaluer votre compréhension de toutes les notions vues en cours;
- Vous permettre de vous familiariser avec l'architecture microservices;
- Utiliser efficacement la gestion des erreurs et la journalisation;
- Utiliser la mise en cache;
- Mettre en place l'authentification;
- Sécuriser l'accès à l'application avec l'autorisation.

II - Contexte

Ce travail doit être effectué par groupe de deux étudiants.

Vous devez déposer à partir de Léa un zip contenant le code source de votre application.

III - Date de remise

Votre travail doit être remis au plus tard le vendredi 24 novembre 2023 à 10h00.

IV – Grille d'évaluation

	Excellent	Fonctionnel	Minimal	Insuffisant
Capacité 1 : Adopter des pratiques de programmation reconnues	**Journalisation et gestion des erreurs :** • La journalisation est toujours correctement utilisée; • Les erreurs sont toujours correctement gérées.	**Journalisation et gestion des erreurs :** • La journalisation est presque toujours correctement utilisée; • Les erreurs sont presque toujours correctement gérées.	**Journalisation et gestion des erreurs :** • La journalisation est la plupart du temps correctement utilisée; • Les erreurs sont la plupart du temps correctement gérées.	**Journalisation et gestion des erreurs :** • La journalisation est rarement correctement utilisée; • Les erreurs sont rarement gérées.
Capacité 2 : Programmer en utilisant des fonctions avancées du langage	**Architecture microservices:** • Excellente maitrise des principes de l'architecture microservices • L'architecture proposée est totalement conforme à l'architecture microservices • Les principes de souveraineté de données sont correctement respectés **Sécurité** • L'authentification est toujours correctement utilisée; • La gestion des autorisations est toujours correctement effectuée.	**Architecture microservices :** • Maitrise presque excellente des principes de l'architecture microservices • L'architecture proposée est presque conforme à l'architecture microservices • Les principes de souveraineté de données sont presque respectés **Sécurité** • L'authentification est presque toujours correctement utilisée; • La gestion des autorisations est presque toujours correctement effectuée.	**Architecture microservices:** • Les principes de l'architecture microservices sont la plupart du temps maitrisés • L'architecture proposée est la plupart du temps conforme à l'architecture microservices • Les principes de souveraineté de données sont la plupart du temps respectés **Sécurité** • L'authentification est la plupart du temps correctement utilisée; • La gestion des autorisations est la plupart du temps correctement effectuée.	**Architecture microservices:** • Les principes de l'architecture microservices ne sont pas maitrisés • L'architecture proposée n'est pas conforme à l'architecture microservices • Les principes de souveraineté de données ne sont pas respectés. **Sécurité** • L'authentification est la rarement utilisée; • La gestion des autorisations est rarement effectuée.

IV – Énoncé

Vous avez été contacté par une grande entreprise pour mettre en place une application de recrutement des employés.

La solution baptisée ModernRecrut est constituée des éléments suivants :

- Une interface Web qui sera implémentée en utilisant ASP.NET Core MVC;
- Une application mobile Android;
- Ces applications vont utiliser des services développés avec ASP.NET Core Web API pour répondre aux besoins des utilisateurs.

Les premiers travaux ont permis d'identifier les composantes suivantes :

Microservice	Fonctionnalités
Gestion des offres d'emploi	Création, modification, affichage et suppression des offres d'emploi
Gestion des favoris	Ajout des offres dans les favoris Affichage des offres dans les favoris Suppression des offres dans les favoris
Gestion des utilisateurs	Création, modification et suppression des comptes Création, modification et suppression des rôles Association des comptes aux rôles Connexion Gestion des autorisations
Gestion des postulations	Postulation des candidats Consultations des postulations Mise à jour des postulations
Gestion des documents	Dépôts des documents (CV, lettre de motivation, etc.) Affichage des documents
Gestion des notifications	Envoi des notifications
Application Web	Offre l'interface pour interagir avec les microservices
Application Android	Offre l'interface pour interagir avec les microservices

Pour la phase 2, vous devez implémenter la gestion des utilisateurs et la gestion des documents.

On vous remet ici les notes de l'analyste pour ces parties :

Analyste : un autre détail en ce qui concerne les offres d'emploi ?

Entreprise : oui la consultation des offres d'emploi peut être anonyme. La personne qui consulte les offres doit être en mesure d'ajouter les offres d'emploi qui l'intéresse dans ses favoris. Les favoris ont une durée de 24h. L'utilisateur doit être capable de consulter les offres d'emploi dans ses favoris.

Analyste : ce qui veut dire qu'il y'aura de l'authentification dans l'application ?

Entreprise : Oui, bien sûr. Certaines fonctionnalités de l'application sont accessibles uniquement après authentification. Chaque compte utilisateur dispose d'un identifiant (adresse email), mot de passe, nom et prénom. Les comptes sont associés à des rôles. Un utilisateur peut avoir plusieurs rôles.

Analyste : Ce qui veut dire qu'un candidat pour postuler à une offre d'emploi il doit être authentifié ?

Entreprise : Exactement. Le candidat peut ajouter les offres d'emploi dans ses favoris sans être authentifié. Mais pour postuler à une offre d'emploi il doit être authentifié. Un bouton "postuler" disponible dans la page détaillée de l'offre d'emploi permet au candidat d'accéder au formulaire de postulation. Il doit renseigner ses prétentions salariales et sa journée de disponibilité pour une entrevue.

Analyste : Et qu'est-ce qui se passe après la postulation

Entreprise : après il y'a l'entrevue. A la suite de l'entrevue, les personnes des RH, qui sont également des utilisateurs de l'application, se connectent avec leur comptent et ajoutent des notes à la postulation. Ils peuvent ajouter une ou plusieurs notes. On doit être en mesure d'identifier la personne des RH qui a ajouté des notes.

Analyste : Y'a-t-il d'autres conditions à respecter pour la postulation ?

Entreprise : Oui, le candidat doit avoir déjà déposé au minimum un CV et une lettre de motivation.

Analyste : Ce qui veut dire que le candidat doit avoir un espace de gestion de ses documents ?

Entreprise : Évidemment, il doit avoir à sa disposition une interface permettant de consulter, ajouter et supprimer des documents. Ces documents sont des fichiers et peuvent être des CV, lettre de motivation, diplômes, etc.

Contraintes techniques

1 – Contraintes d'implémentation

Compte tenu de la complexité d'implémentation d'une API de gestion des utilisateurs avec authentification ASP.NET Core Identity, toute cette portion sera directement implémentée dans l'application MVC.

2 – Gestion des données

La gestion des utilisateurs va reposer sur une base de données SQLite.

L'API de gestion des documents doit directement stocker les fichiers dans un répertoire dans le projet correspondant.

3 – Gestion des utilisateurs et authentification

Vous devez mettre en place l'authentification en utilisant SQLite et ASP.NET Core Identity, en tenant compte des éléments suivants :

- Le formulaire de création d'un compte utilisateur doit disposer d'un champ type, pouvant avoir les valeurs Employé et Candidat.
- Vous devez implémenter les formulaires d'affichage, de création et d'attribution de rôles. Vous devez éviter les doublons tant lors de la création des rôles que lors de l'attribution. Les rôles connus de l'application pour l'instant sont : « Employe », « Candidat », « RH », « Admin ».
- Si l'utilisateur est employé, vous devez automatiquement lui associer le rôle « Employe », sinon lui associer le rôle « Candidat » lors de la création.
- Pour simuler l'autorisation, vous allez créer un contrôleur Postulation avec des méthodes d'action et vues vides selon ce qui suit :
    - Une méthode d'action Postuler accessible uniquement aux candidats;
    - Une méthode d'action ListePostulations accessible aux employés;
    - Une méthode d'action Notes accessible aux employés qui sont RH;
    - Toutes les méthodes d'action sont accessibles à l'Admin

4 – Gestion des documents

Implémentez et sécurisez l'espace de gestion des documents du candidat en tenant compte des notes de l'analyste et ce qui suit :

- L'API de gestion des documents doit offrir deux méthodes, l'une permettant d'enregistrer un document et l'autre permettant d'obtenir les documents d'un candidat;
- L'enregistrement d'un document permet de sauvegarder ce dernier dans un répertoire documents du dossier wwwroot. Le nom du fichier est créé en concaténant l'ID de l'utilisateur, le type du document (CV, Lettre de motivation, Diplômes) et un numéro aléatoire. Vous devez utiliser le « _ » comme caractère de séparation;
- L'obtention d'un document prend en paramètre l'identifiant de l'utilisateur et retourne dans une liste de string les noms des fichiers correspondants à ce dernier;

- Le formulaire de création de l'application MVC doit permettre de sélectionner le document à déposer, définir le type de document (CV, Lettre de motivation, Diplômes) et enregistrer. L'utilisateur ne peut déposer que des documents Word et PDF;
- L'affichage des documents doit afficher la liste des documents du candidat avec le type de document et un lien permettant de consulter/sauvegarder le document (pour concevoir le lien, vous pouvez simplement utiliser l'URL de l'api suivi du chemin du fichier. Exemple : https://localhost:5353/documents/cv_565635357353537353537563_1.pdf)

Les tutoriels suivants peuvent vous aider dans l'implémentation de la gestion des documents :

- API Web : https://www.c-sharpcorner.com/article/upload-single-and-multiple-files-using-the-net-core-6-web-api/
- MVC : https://code-maze.com/file-upload-aspnetcore-mvc/

6 – Journalisation

Une journalisation adéquate doit être mise en place et on doit être capable de savoir qui a fait quoi dans l'application.

Travail à faire

1. Créez dans Visual Studio le projet ModernRecrut.Documents.API;
2. À partir des notes ci-dessus, implémentez les différentes applications.