A23 – POO 4 - Travail pratique 2 (20%)

I - Objectif 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.

II - Contexte

Ce travail doit être effectué par groupe de deux ou trois 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 10 novembre 2023 à 9h59.

Aucune remise en retard ne sera acceptée.

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. **Gestion des performances :** • La mise en cache est correctement utilisée.	**Journalisation et gestion des erreurs :** • La journalisation est presque toujours correctement utilisée; • Les erreurs sont presque toujours correctement gérées. **Gestion des performances :** • La mise en cache est presque correctement utilisée.	**Journalisation et gestion des erreurs :** • La journalisation est la plupart du temps correctement utilisée; • Les erreurs sont la plupart du temps correctement gérées. **Gestion des performances :** • La mise en cache est la plupart du temps correctement utilisée.	**Journalisation et gestion des erreurs :** • La journalisation est rarement correctement utilisée; • Les erreurs sont rarement gérées. **Gestion des performances:** • La mise en cache n'est pas utilisée correctement.
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	**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	**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	**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.

IV – Enoncé

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 1, vous devez implémenter l'API de gestion des offres d'emploi, l'API de gestion des favoris et l'interface Web MVC pour ces fonctionnalités.

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

Analyste : Comment commence le processus de recrutement ?

Entreprise : lorsqu'un poste est disponible en entreprise, nous devons publier une offre d'emploi. Pour chaque offre d'emploi, nous devons renseigner l'intitulé du poste, la date de début d'affichage du poste sur notre site, la date de fin d'affichage du poste, la description du poste.

Analyste : Comment sont affichées les offres d'emploi ?

Entreprise : nous voulons qu'une page affiche la liste de toutes les offres d'emploi valides. C'est-à-dire que leur date d'affichage doit être inférieure ou égale à la date du jour et leur date de fin d'affichage doit être supérieure ou égale à la date du jour. Le candidat doit être en mesure de filtrer en fonction du poste. On doit être capable de modifier, supprimer et consulter la fiche détaillée de chaque offre d'emploi.

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.

Contraintes techniques

1 – Gestion des données

L'API Gestion des offres d'emploi doit utiliser une base de données SQLite.

L'API Gestion des favoris doit utiliser une base de données en mémoire via la mise en cache mémoire, selon les critères suivants :

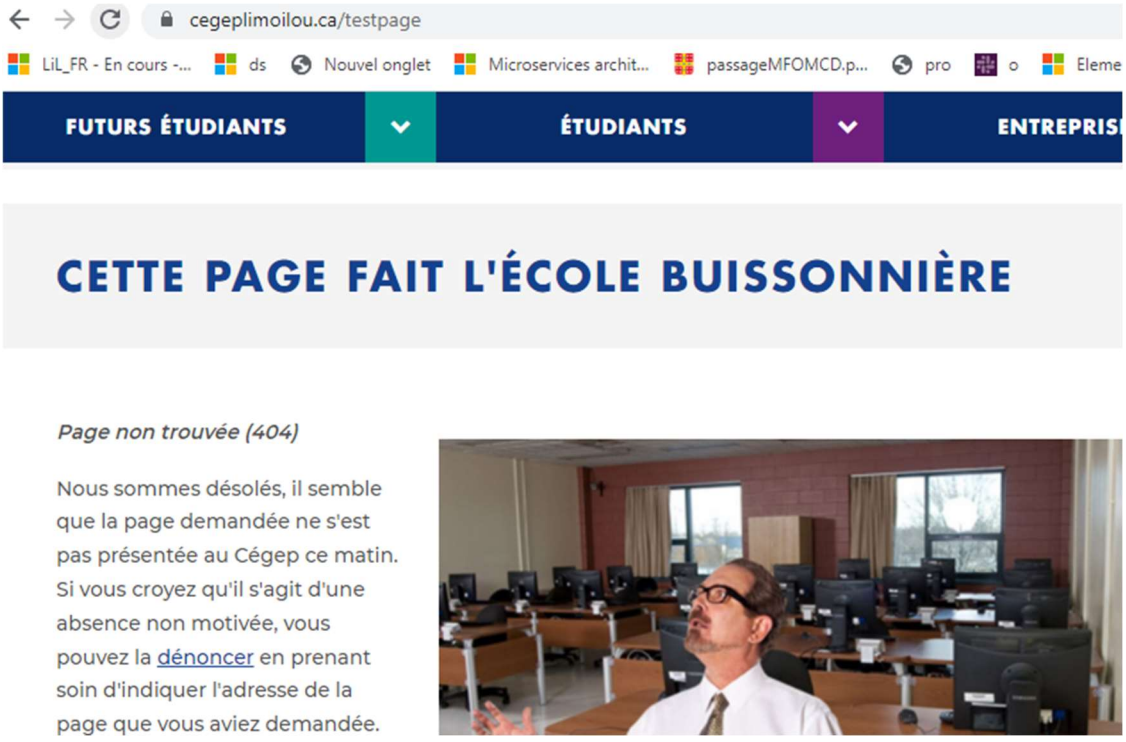
- L'adresse IP de l'utilisateur est utilisée comme clé pour la mise en cache;
- Les favoris ont une durée de 24h;
- Les favoris doivent expirer après 6h en cas de non-accès;
- Le cache doit avoir une taille maximale de 5000000;
- Chaque caractère des données ajoutées au cache à une taille de « 1 ».

2 – Gestion des erreurs

Vous devez mettre en place les éléments suivants pour offrir une gestion des erreurs convenable dans votre application :

- Une page d'erreur personnalisée qui sera affichée pour toute erreur dans votre application;

- Une page personnalisée qui sera affichée à l'écran si l'utilisateur essaye d'accéder à une page qui n'existe pas. Un peu à l'image de ce qui a été fait sur le site du Cegep :



3 - Journalisation

Pour l'application MVC, nous devons journaliser les codes d'erreur HTTP des API Web. La journalisation est catégorisée par services. Les codes 400 – 499 seront journalisés comme des erreurs et les codes 500 – 599 comme critiques.

Les actions utilisateurs seront journalisées avec le niveau Information. Les actions suivantes doivent être journalisées :

- La création, la modification et la suppression d'une offre d'emploi;
- L'ajout et la suppression d'un élément dans les favoris.

Vous devez configurer la journalisation pour journaliser :

- A partir du niveau information dans le fournisseur console;
- A partir du niveau erreur dans le fournisseur Windows;
- Toutes les composantes dans le namespace ModernRecrut avec le niveau Trace.

Travail à faire

1. Créez dans Visual Studio les projets ModernRecrut.MVC, ModernRecrut.Emplois.API et ModernRecrut.Favoris.API;
2. À partir des notes ci-dessus, implémentez les différentes applications.