|  |  |  |
| --- | --- | --- |
| Travail pratique 2 |  | *15 points* |

**Mandat**

Dans le cadre de ce travail, nous voulons développer une application web utilisant un backend de gestion de clients et d’adresses (voir le fichier docker-compose.yaml en lien avec le travail pratique. La documention se trouve sur <http://localhost:2323/swagger>)

Le travail s’effectuera en équipe de deux. Afin de mieux organiser notre développement, nous définirons des fonctionnalités qui seront assignées à chaque membre de l’équipe. Ainsi il vous sera possible de faire votre partie indépendamment de l’avancement de votre collègue. Les fonctionnalités devront être découpées en tâches et chacune d’elles devra être dans un seul commit fonctionnel.

Une grande marge de manœuvre vous est accordée pour ce travail. Assurez-vous de bien définir votre stratégie avant de vous avancer trop loin. Pour la durée du travail, l’enseignant assumera les rôles de client et de chef d’équipe. Dans le doute, référez-vous à lui!

**Fonctionnalités individuelles requises :**

1. **Section « Page clients »**
2. **Affichage des clients**
   * Affichez la liste des clients avec leur nom et prénom;
   * Pour chaque client, ajoutez deux boutons : « Modifier » et « Supprimer »;
   * Lorsque l’utilisateur clique sur « Modifier », redirigez-le vers une page avec l’identifiant du client dans l’URL (voir section « Gestion des clients »);
   * Lorsque l’utilisateur clique sur « Supprimer », redirigez-le une page de confirmation de suppression. Lancez la requête de suppression uniquement si l’utilisateur confirme la suppression.
3. **Filtres dynamiques**
   * Créez un outil de filtre qui se remplit dynamiquement en fonction des données présentes dans votre composant;
   * Les filtres doivent être basés sur les informations d’adresse des clients;
   * Utilisez des cases à cocher pour les catégories de filtres pour les champs municipalité, état et pays;
   * Affichez le nombre de clients correspondant à chaque filtre entre parenthèses;
   * La logique des filtres doit être en logique OU. Par exemple, si on sélectionne les pays « Canada » et « USA », nous voulons avoir les clients qui ont une adresse dans une de ces deux locations (pas les deux en même temps).
4. **Tri des clients**
   * Permettez aux utilisateurs de trier la liste des clients par nom ou par prénom;
   * Si deux noms sont identiques, triez ensuite par prénom;
   * Si deux prénoms sont identiques, triez ensuite par nom.
5. **Page d’accueil**
   * Créez une page d’accueil accessible pour tous les utilisateurs.
6. **Section « Gestion des clients »**
7. **Création de client**
   * Ajoutez un bouton pour créer un nouveau client;
   * Lorsque l’utilisateur clique sur ce lien, redirigez-le vers une page de création de client;
   * L’utilisateur doit pouvoir entrer le nom, le prénom et la date de naissance d’un nouveau client;
   * Lorsque l’utilisateur soumet les informations, le client doit être ajouté à la liste.
8. **Modification d’un client existant**
   * L’utilisateur doit pouvoir modifier les informations d’un client existant (nom, prénom, date de naissance);
   * Lorsque l’utilisateur soumet les modifications, les informations du client doivent être mises à jour.
9. **Gestion des adresses**
   * À partir d’un client, l’utilisateur doit pouvoir ajouter, modifier ou supprimer une ou plusieurs adresses;
   * Un client peut avoir autant d’adresses qu’il souhaite;
   * Chaque adresse doit contenir des informations telles que la rue, la ville, l’état et le pays;
   * Lorsque l’utilisateur soumet les modifications, les informations du client doivent être mises à jour.

**Fonctionnalités globales requises :**

* Lors des fetch, tous les codes d’erreurs du backend doivent tous être gérés. Toutefois, une validation doit être faite également à partir de l’interface afin d’éviter d’avoir des erreurs du backend (champs vides ou invalides).
* Toutes les pages sauf la page d’accueil doivent être accessibles uniquement après s’être authentifié avec Auth0.

**Contraintes techniques :**

* Utilisez uniquement les modules React vus en classe;
* Utilisez uniquement les hooks React vus en classe;
* Ne remettez pas de répertoire build et node\_modules;
* Pas d'utilisation d'intelligence artificielle;
* L’utilisation de useEffect est interdite excepté pour appeler fetch;

**Particularités :**

Démontrez vos compétences en programmation en produisant du code de haute qualité. Une attention particulière sera portée sur la réutilisation de vos composants ainsi que sur le respect des principes DRY et SRP.

Pour l’interface, respectez les bonnes pratiques d’utilisabilité vues en classe;

**Évaluation :**

|  |  |
| --- | --- |
| **Critères individuels** | **Points** |
| Fonctionnalités : | |
| * Implémentation correcte des fonctionnalités individuelles et globales demandées | /60 |
| * Convivialité de l’application | /10 |
| Qualité globale du code (lisibilité, modularité, indentation, commentaires)  Qualité et découpage des commits | /30 |
| Bonus : Implémentation du support multilingue, couverture de tests | /10 |
| **Total** | **/100** |

La proportion des points attribuée à la qualité globale du code ne pourra pas dépasser la proportion des points atteinte pour les fonctionnalités. Par exemple, si une personne cumule 35 points pour les fonctionnalités (50% de 70 points possibles) elle ne pourra pas avoir plus de 15 points pour la qualité du code (50% de 30 points possibles).

Les propositions des étudiants peuvent être évaluées en entrevue après la remise du travail.

Le non respect des contraintes techniques vous donnera immédiatement la note 0.

**Bonus d’équipe (5 points) :**

Sur Git, organisez vos travaux de manière à éviter de faire des commits en parallèle. Structurez votre branche de développement principale de manière à être la plus linéaire possible. Un bonus de 10 % sera accordé à une équipe qui arrive à faire tout son développement sans aucun embranchement (merge).

**À faire avant le 27 mai à minuit :**

* Une personne de votre équipe devra envoyer par Mio le lien de votre dépôt git. Envoyez également la composition de l’équipe avec la fonctionnalité attribuée à chaque membre.

**Remise finale : 9 juin à minuit**