UE Full Stack

TP2 UE Full Stack

TP2: Création d'une Interface Web pour Gérer des Événements et des Artistes

Objectif :

Dans ce TP, vous allez créer la partie frontend d'une application web consommant une API existante (à noter que la documentation est largement perfectible et elle à pour but ici de vous donner le minimum d'information pour que vous puissiez la manipuler; à noter aussi que la mauvaise gestion des erreurs cotés serveur sont volontaires). L'API fournit des endpoints sur des **événements** et des **artistes**, qui sont liés entre eux par une relation many-to-many (un événement peut avoir plusieurs artistes et un artiste peut participer à plusieurs événements).

Vous allez devoir implémenter 4 vues principales, intégrer des validations de formulaire, ajouter de la pagination et gérer les erreurs serveur tout en assurant une attention particulière à l'UX/UI.

Contraintes techniques :

- **Validation** : Implémentez des validations côté client pour les formulaires.
- **Erreurs** : Gérez correctement les erreurs serveur (comme les erreurs 400, 401, 404, 500) et affichez des messages clairs à l'utilisateur.
- **Pagination** : Vous devez implémenter une pagination pour la vue des événements et des artistes.
- **UX/UI** : Assurez-vous que l'interface soit fluide, ergonomique, et responsive. Ajoutez des animations ou des transitions si nécessaire pour améliorer l'expérience utilisateur.

Lancement de l'API :

- Récupérer le JAR fourni avec ce document
- Faite la commande `java -jar event-0.0.1-SNAPSHOT.jar` pour lancer le serveur.
    - Une version de java 17 peut être nécessaire pour faire fonctionner le projet, vous pouvez aussi créer un dockerfile pour exécuter le projet au besoin.
- Une fois le serveur exécuté, vous pouvez accéder à la documentation de l'API via ce lien http://localhost:8080/swagger-ui/index.html

1. Vue 1 : Liste des Événements

Route : `/events`

Fonctionnalités :

- Affichez la liste paginée des événements.
- Chaque événement doit afficher le nom, la date de début, la date de fin, un aperçu des artistes qui participent à l'événement et combien d'artistes il y a en tout dans cet événement.
- Ajoutez un bouton pour accéder aux détails d'un événement.

Validation UX :

- Le système de pagination doit être clair et simple d'utilisation. Les utilisateurs doivent pouvoir naviguer entre les pages sans rafraîchir le navigateur.

---

2. Vue 2 : Détail d'un Événement

Route : `/events/:id`

Fonctionnalités :

- Affichez toutes les informations relatives à l'événement sélectionné (nom, date, liste des artistes associés).
- Permettez la modification des détails de l'événement (nom, date).
- Ajoutez des validations sur les champs du formulaire (par exemple, la date doit être future, le nom doit être d'au moins 3 caractères, etc.).
- Intégrez la possibilité d'ajouter ou de retirer des artistes à l'événement.

Validation UX :

- En cas de soumission du formulaire avec des données invalides, l'utilisateur doit recevoir un retour visuel clair indiquant les erreurs spécifiques (par exemple, "Le nom doit comporter au moins 3 caractères").
- Ajoutez des transitions douces lors de la modification ou du chargement des données.

Gestion des erreurs :

- Gérez les erreurs serveur et affichez des messages utilisateurs si une requête échoue (par exemple, "Impossible de charger les détails de l'événement").

---

3. Vue 3 : Liste des Artistes

Route : `/artists`

Fonctionnalités :

- Affichez la liste paginée des artistes (10 artistes par page).
- Chaque artiste doit afficher : le nom, et un aperçu des événements auxquels il participe.
- Ajoutez un bouton pour accéder aux détails d'un artiste.
- Implémentez une barre de recherche pour filtrer les artistes par nom.

Validation UX :

- La pagination doit rester cohérente avec celle de la vue des événements, avec des transitions fluides lors de la navigation.

---

4. Vue 4 : Détail d'un Artiste

Route : `/artists/:id`

Fonctionnalités :

- Affichez toutes les informations sur l'artiste (nom, liste des événements auxquels il participe).
- Permettez la modification des détails de l'artiste (nom).
- Ajoutez des validations sur les champs du formulaire (le nom doit comporter au moins 3 caractères, etc.).
- Intégrez la possibilité d'ajouter ou de retirer des événements pour cet artiste.

Validation UX :

- Indiquez visuellement les champs obligatoires et affichez des messages clairs en cas d'erreurs de validation.
- Utilisez des notifications ou des modales pour confirmer les ajouts/suppressions d'événements de l'artiste.

Gestion des erreurs :

- Gérez les erreurs serveur, par exemple, si l'API est indisponible ou si l'identifiant de l'artiste n'existe pas (affichez une erreur 404 avec un message utilisateur clair).

---

5. Gestion des Erreurs Serveur

- Pour chaque vue, vous devez gérer les erreurs provenant de l'API. Si une requête échoue, affichez un message d'erreur clair à l'utilisateur.
- Les erreurs doivent être traitées de manière globale via un gestionnaire d'erreurs. Par exemple :
  - **404** : Ressource non trouvée.
  - **500** : Erreur serveur.
- Utilisez des modales ou des notifications pour informer l'utilisateur des erreurs critiques.

---

6. Validation des Formulaires

Les formulaires doivent inclure les éléments de validation suivants :

- **Nom d'un événement ou d'un artiste** : Minimum 3 caractères, obligatoire.
- **Date de début et de fin d'un événement** : La date de début doit être antérieure à la date de fin.

En cas d'erreurs de validation, un message d'erreur doit être affiché sous le champ correspondant.

---

7. Suggestions pour Améliorer l'UX/UI

- **Notifications claires :** lors des actions réussies (comme la modification d'un événement ou d'un artiste).
- **Messages d'erreurs :** bien visibles mais non intrusifs.
- **Responsive design** : Assurez-vous que l'interface fonctionne bien sur toute taille d'écran.

Rendu attendu :

- Une application frontend fonctionnelle avec les 4 vues décrites ci-dessus.
- Utilisation correcte de l'API pour récupérer, ajouter, modifier et supprimer des événements et des artistes.

- Pagination, validation des formulaires, gestion des erreurs serveur et une attention particulière à l'UX/UI.
- Vous pouvez aussi envoyer les outils que vous avez créé autour de ce projet (script de création de données par exemple)