TP - React

Contexte

Vous êtes chargé de concevoir et développer une application web utilisant React permettant la consultation et la gestion de conférences.

L'application doit s'appuyer sur une API REST disponible localement (à l'adresse http://localhost:4555), dont la documentation est accessible à la même adresse.

Un système d'authentification doit être implémenté. Deux types d'utilisateurs coexistent :

- **Utilisateur simple** : accès en lecture seule,
- **Administrateur** : accès aux interfaces d'administration.

Objectifs

Votre application devra permettre :

- La consultation des conférences (liste et détails),
- La gestion des conférences (ajout, modification, suppression) pour les administrateurs,
- La gestion des utilisateurs (consultation et promotion au rôle administrateur) pour les administrateurs.

Consigne Git

Pour le rendu, chaque étudiant devra créer un **dépôt Git**, y pousser son travail, puis m'ajouter en tant que **collaborateur** afin que je puisse récupérer les livrables.
Mon GitHub : richardlemmer-mns

Spécifications fonctionnelles

1. **Pages attendues** (minimum 5) :
    - Page d'accueil : liste de toutes les conférences.
    - Fiche détaillée d'une conférence.
    - Page de connexion.
    - Interface d'administration pour la gestion des conférences.
    - Interface d'administration pour la gestion des utilisateurs.
2. **Système d'authentification** :
    - Permet l'identification des utilisateurs.
    - Gère les droits d'accès en fonction du rôle.
3. **Interface utilisateur** :
    - Chaque conférence possède un code couleur spécifique visible dans la fiche détaillée en guise de thème.
    - Les utilisateurs non-admins ne peuvent pas accéder aux interfaces d'administration.
4. **Gestion des conférences** :
    - Les administrateurs peuvent créer, modifier et supprimer des conférences.
5. **Gestion des utilisateurs** :
    - Les administrateurs peuvent consulter la liste des utilisateurs.
    - Ils peuvent promouvoir un utilisateur simple en administrateur.

Informations techniques

- Dans un répertoire vide, créez un fichier nommé `docker-compose.yml`.
- Copiez-collez-y le contenu de l'annexe ci-dessous.
- Pour lancer l'API, exécutez la commande `docker-compose up` dans le répertoire précédemment créé.
- L'API est disponible à http://localhost:4555.
- L'interface d'administration de la base de données est disponible à http://localhost:9555. Les logs de connexions sont `admin` et `pass`

```yaml
services:
  web:
    image: eltwingo/mns:latest
    working_dir: /app
    pull_policy: always
    ports:
      - "4555:80"
    depends_on:
      - db
  db:
    image: mongo
    environment:
      - MONGO_INITDB_ROOT_USERNAME:root
      - ME_CONFIG_MONGODB_ADMINPASSWORD:root
    volumes:
      - mongo-data:/mongo
  mongo-express:
    image: mongo-express
    environment:
      - ME_CONFIG_MONGODB_SERVER=db
      - ME_CONFIG_MONGODB_PORT=27017
      - ME_CONFIG_MONGODB_ENABLE_ADMIN=false
      - ME_CONFIG_MONGODB_AUTH_DATABASE=cyberconf
    depends_on:
      - db
    ports:
      - "9555:8081"
volumes:
  mongo-data:
```

Modèles de données

Les données sont fournies sous forme de documents MongoDB, avec la structure suivante (un `!` indique un champ obligatoire) :

Conférence

```
{
  !id: String,
  !title: String,
  !date: String,
  !description: String,
  !img: String,
  !content: String,
  duration: String,
  osMap: {
    addressl1: String,
    addressl2: String,
    postalCode: String,
    city: String,
    coordinates: Array
  },
  speakers: [
    {
      !firstname: String,
      !lastname: String
    }
  ],
  stakeholders: [
    {
      !firstname: String,
      !lastname: String,
      job: String,
      img: String
    }
  ],
  design: {
    !mainColor: String,
    !secondColor: String
  }
}
```

Utilisateur

```
{
  !id: String,
  !password: String,
  !type: String // "user" ou "admin"
}
```