AUTEUR : BELHAIMEUR AMINE

Le Bon Coin Clone

Le Bon Coin Clone est une application MERN (MongoDB, Express, React, Node.js) permettant aux utilisateurs de créer, visualiser, modifier et supprimer des annonces. Elle offre également un espace d'administration pour gérer les utilisateurs.

Fonctionnalités principales

1. **Authentification :**
   - Les utilisateurs peuvent s'inscrire et se connecter via une interface sécurisée.
   - L'authentification est gérée avec **JWT** pour sécuriser les pages protégées.
   - **Un seul compte administrateur peut exister à la fois :**
     - Pour être administrateur, il faut s'inscrire avec l'email **`admin@example.com`**.
     - Si un nouvel admin est nécessaire, l'actuel administrateur doit supprimer son compte, ce qui libère la possibilité de se réinscrire avec l'email **`admin@example.com`**.
   - **Compte administrateur actuel :**
     - **Email** : `admin@example.com`
     - **Mot de passe** : `1234`
2. **Gestion des annonces (CRUD) :**
   - Les utilisateurs connectés peuvent :
     - Créer une annonce.
     - Modifier ou supprimer leurs propres annonces.
   - Chaque annonce comprend un titre, une description, un prix et une catégorie.
3. **Espace Administrateur :**
   - Accessible uniquement via le compte admin.
   - Permet de :
     - Visualiser la liste des utilisateurs.
     - Supprimer des utilisateurs.
4. **Gestion de profil :**
   - Les utilisateurs peuvent :
     - Modifier leurs informations (nom, email, mot de passe).
     - Supprimer leur compte.

Fonctionnement du site

1. **Compte Administrateur :**
   - L'administrateur doit obligatoirement utiliser l'email **`admin@example.com`** pour s'inscrire.
   - Si un autre administrateur est nécessaire :
     - L'actuel administrateur doit supprimer son compte.
     - Un nouvel utilisateur peut ensuite s'inscrire avec **`admin@example.com`** pour devenir admin.

    - **Identifiants actuels pour l'admin :**
        - **Email** : `admin@example.com`
        - **Mot de passe** : `1234`
2. **Inscription utilisateur :**
    - Allez sur la page **Inscription** via la barre de navigation.
    - Remplissez le formulaire avec un nom, un email et un mot de passe.
3. **Connexion utilisateur :**
    - Allez sur la page **Connexion**.
    - Saisissez vos identifiants pour vous connecter.
    - Une fois connecté :
        - Vous êtes redirigé vers la page des annonces.
        - Votre session est sécurisée grâce à un token JWT.
4. **Utilisation :**
    - **Page des annonces :**
        - Créez une nouvelle annonce en remplissant le formulaire.
        - Visualisez toutes les annonces disponibles.
        - Modifiez ou supprimez vos annonces si vous êtes connecté.
    - **Espace Administrateur :**
        - Connectez-vous avec le compte admin pour accéder à la gestion des utilisateurs.
    - **Profil :**
        - Modifiez vos informations personnelles.
        - Supprimez votre compte si besoin.

Utilisation des Middlewares dans l'Application

L'application utilise des middlewares pour sécuriser les routes et garantir que seules les actions autorisées sont effectuées par les utilisateurs ou l'administrateur.

1. **`authMiddleware` (Middleware d'authentification)** :
    - Vérifie si un token **JWT** valide est présent dans l'en-tête de la requête.
    - Décode le token pour récupérer l'utilisateur associé et ajoute les informations de l'utilisateur (`req.user`) pour les utiliser dans les routes.
    - Si le token est manquant ou invalide, la requête est bloquée avec un statut **401 (Non autorisé)** ou **403 (Interdit)**.

    Exemple d'utilisation :

    ```
    javascript
    Copier le code
    router.post("/ads", auth, async (req, res) => {
      // Route protégée nécessitant un utilisateur authentifié.
    });
    ```

2. **`adminMiddleware` (Middleware pour l'administrateur)** :
    - Vérifie si l'utilisateur authentifié est un administrateur (`req.user.role === "admin"`).
    - Si l'utilisateur n'est pas un administrateur, la requête est bloquée avec un statut **403 (Interdit)**.

- Utilisé pour protéger les routes d'administration.

Exemple d'utilisation :

```
javascript
Copier le code
router.get("/users", auth, admin, async (req, res) => {
  // Route protégée nécessitant un administrateur.
});
```

Exemple d'intégration dans les routes

- **Authentification des annonces** : Les actions comme la création, la mise à jour ou la suppression des annonces nécessitent une authentification via `authMiddleware`.

```
javascript
Copier le code
router.post("/ads", auth, async (req, res) => {
  // Créer une annonce avec un utilisateur authentifié.
});
```

- **Protection des routes administrateur** : Les actions liées à la gestion des utilisateurs (CRUD) sont réservées à l'administrateur via `adminMiddleware`.

```
javascript
Copier le code
router.get("/users", auth, admin, async (req, res) => {
  // Récupérer tous les utilisateurs, uniquement accessible par
l'admin.
});
```

Gestion des rôles et permissions

- Un utilisateur normal a accès uniquement à ses propres annonces et peut gérer son profil.
- Un administrateur, authentifié avec l'email `admin@example.com`, a accès à des fonctionnalités supplémentaires via des routes protégées par `adminMiddleware`.

En combinant ces deux middlewares, l'application garantit une séparation claire des privilèges entre les utilisateurs et l'administrateur.

Technologies utilisées

- **Backend :**
  - Node.js / Express
  - MongoDB avec Mongoose
  - JWT pour l'authentification
- **Frontend :**
  - React avec React Router

- Bootstrap pour le design
- Axios pour les requêtes API

Lancer le projet

Prérequis

- Node.js installé
- MongoDB installé ou accessible (local ou cloud)

Backend

1. Allez dans le dossier `backend`.
2. Installez les dépendances : `npm install`.
3. Lancez le serveur : `node server.js`.

Frontend

1. Allez dans le dossier `frontend`.
2. Installez les dépendances : `npm install`.
3. Lancez le client : `npm start`.

Notes importantes

1. **Compte Administrateur :**
   - Un seul administrateur peut exister à un instant donné.
   - Pour devenir admin :
     - Utilisez l'email **`admin@example.com`** lors de l'inscription.
     - Si ce compte existe déjà, l'administrateur actuel devra d'abord supprimer son compte pour libérer cet email.
   - **Identifiants actuels pour l'admin :**
     - **Email** : `admin@example.com`
     - **Mot de passe** : `1234`
2. **Configuration MongoDB :**
   - Ajoutez vos informations dans le fichier `.env` (dans `backend`), par exemple :

```
le code
PORT=5000
MONGO_URI=mongodb://127.0.0.1:27017/leboncoinDB
JWT_SECRET=supersecretkey
```