

// src/services/authService.js

export const register = async (userData) => {

const response = await fetch('/api/auth/register', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(userData),

});

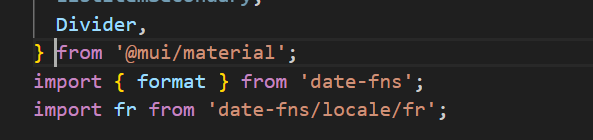
if (!response.ok) {

throw new Error('Erreur lors de l\'inscription');

}

return response.json();

};



import { format } from 'date-fns';

import fr from 'date-fns/locale/fr';

import React, { createContext, useState, useContext, useEffect } from 'react';

import axios from 'axios';

const AuthContext = createContext(null);

export const AuthProvider = ({ children }) => {

  const [user, setUser] = useState(null);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  useEffect(() => {

    checkAuth();

  }, []);

  const checkAuth = async () => {

    try {

      const token = localStorage.getItem('token');

      console.log('Checking auth, token exists:', !!token);  // Log pour débogage

      if (!token) {

        setLoading(false);

        return;

      }

      const response = await axios.get('/api/auth/me', {

        headers: { Authorization: `Bearer ${token}` }

      });

      console.log('Auth check response:', response.data);  // Log pour débogage

      setUser(response.data);

      setError(null);

    } catch (error) {

      console.error('Auth check error:', error.response || error);  // Log détaillé

      localStorage.removeItem('token');

      delete axios.defaults.headers.common['Authorization'];  // Nettoyer aussi les headers

      setUser(null);

      setError('Session expirée');

    } finally {

      setLoading(false);

    }

  };

  const login = async (email, password) => {

    try {

      console.log('Attempting login for email:', email);  // Log pour débogage

      const response = await axios.post('/api/auth/login', { email, password });

      console.log('Login response:', response.data);  // Log pour débogage

      const { token, user: userData } = response.data;

      localStorage.setItem('token', token);

      setUser(userData);

      setError(null);

      // Configure axios pour inclure le token dans toutes les requêtes futures

      axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;

      return userData;

    } catch (error) {

      console.error('Login error:', error.response || error);  // Log détaillé

      const message = error.response?.data?.message || 'Erreur lors de la connexion';

      setError(message);

      throw new Error(message);

    }

  };

  const register = async (userData) => {

    try {

      const response = await axios.post('/api/auth/register', userData);

      const { token, user: newUser } = response.data;

      localStorage.setItem('token', token);

      setUser(newUser);

      setError(null);

      axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;

      return newUser;

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de l\'inscription';

      setError(message);

      throw new Error(message);

    }

  };

  const logout = () => {

    localStorage.removeItem('token');

    delete axios.defaults.headers.common['Authorization'];

    setUser(null);

    setError(null);

  };

  const forgotPassword = async (email) => {

    try {

      await axios.post('/api/auth/forgot-password', { email });

      setError(null);

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de l\'envoi du mail de réinitialisation';

      setError(message);

      throw new Error(message);

    }

  };

  const resetPassword = async (token, newPassword) => {

    try {

      await axios.post('/api/auth/reset-password', { token, password: newPassword });

      setError(null);

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de la réinitialisation du mot de passe';

      setError(message);

      throw new Error(message);

    }

  };

  const updateProfile = async (userData) => {

    try {

      const response = await axios.put('/api/auth/profile', userData, {

        headers: { Authorization: `Bearer ${localStorage.getItem('token')}` }

      });

      setUser(response.data);

      setError(null);

      return response.data;

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de la mise à jour du profil';

      setError(message);

      throw new Error(message);

    }

  };

  const value = {

    user,

    loading,

    error,

    isAuthenticated: !!user,

    login,

    register,

    logout,

    forgotPassword,

    resetPassword,

    updateProfile,

    setError

  };

  if (loading) {

    return <div>Chargement...</div>; // Vous pouvez remplacer ceci par un composant de chargement personnalisé

  }

  return (

    <AuthContext.Provider value={value}>

      {children}

    </AuthContext.Provider>

  );

};

export const useAuth = () => {

  const context = useContext(AuthContext);

  if (!context) {

    throw new Error('useAuth doit être utilisé à l\'intérieur d\'un AuthProvider');

  }

  return context;

};

export default AuthContext;

import { useAuth } from '../contexts/AuthContext';

  const { login } = useAuth();

 const user = await login(formData.email, formData.password);

import React, { createContext, useState, useContext, useEffect } from 'react';

import axios from 'axios';

const AuthContext = createContext(null);

export const AuthProvider = ({ children }) => {

  const [user, setUser] = useState(null);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  useEffect(() => {

    checkAuth();

  }, []);

  const checkAuth = async () => {

    try {

      const token = localStorage.getItem('token');

      if (!token) {

        setLoading(false);

        return;

      }

      const response = await axios.get('/api/auth/me', {

        headers: { Authorization: `Bearer ${token}` }

      });

      setUser(response.data);

      setError(null);

    } catch (error) {

      console.error('Erreur lors de la vérification de l\'authentification:', error);

      localStorage.removeItem('token');

      setUser(null);

      setError('Session expirée');

    } finally {

      setLoading(false);

    }

  };

  const login = async (email, password) => {

    try {

      const response = await axios.post('/api/auth/login', { email, password });

      const { token, user: userData } = response.data;

      localStorage.setItem('token', token);

      setUser(userData);

      setError(null);

      // Configure axios pour inclure le token dans toutes les requêtes futures

      axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;

      return userData;

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de la connexion';

      setError(message);

      throw new Error(message);

    }

  };

  const register = async (userData) => {

    try {

      const response = await axios.post('/api/auth/register', userData);

      const { token, user: newUser } = response.data;

      localStorage.setItem('token', token);

      setUser(newUser);

      setError(null);

      axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;

      return newUser;

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de l\'inscription';

      setError(message);

      throw new Error(message);

    }

  };

  const logout = () => {

    localStorage.removeItem('token');

    delete axios.defaults.headers.common['Authorization'];

    setUser(null);

    setError(null);

  };

  const forgotPassword = async (email) => {

    try {

      await axios.post('/api/auth/forgot-password', { email });

      setError(null);

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de l\'envoi du mail de réinitialisation';

      setError(message);

      throw new Error(message);

    }

  };

  const resetPassword = async (token, newPassword) => {

    try {

      await axios.post('/api/auth/reset-password', { token, password: newPassword });

      setError(null);

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de la réinitialisation du mot de passe';

      setError(message);

      throw new Error(message);

    }

  };

  const updateProfile = async (userData) => {

    try {

      const response = await axios.put('/api/auth/profile', userData, {

        headers: { Authorization: `Bearer ${localStorage.getItem('token')}` }

      });

      setUser(response.data);

      setError(null);

      return response.data;

    } catch (error) {

      const message = error.response?.data?.message || 'Erreur lors de la mise à jour du profil';

      setError(message);

      throw new Error(message);

    }

  };

  const value = {

    user,

    loading,

    error,

    isAuthenticated: !!user,

    login,

    register,

    logout,

    forgotPassword,

    resetPassword,

    updateProfile,

    setError

  };

  if (loading) {

    return <div>Chargement...</div>; // Vous pouvez remplacer ceci par un composant de chargement personnalisé

  }

  return (

    <AuthContext.Provider value={value}>

      {children}

    </AuthContext.Provider>

  );

};

export const useAuth = () => {

  const context = useContext(AuthContext);

  if (!context) {

    throw new Error('useAuth doit être utilisé à l\'intérieur d\'un AuthProvider');

  }

  return context;

};

export default AuthContext;