populare db

import url + headerparams

refa home, pune meniu

login + sign up (CUM retin token dupa register?)

- msj eroare

filtrare

- country

- author

paginare posts (9-10), comments (5)

like postare PUT (btn vizibil la login/register)

sortare ASC/DESC

* nr likeuri
* data timestamp

formular add/edit/del post + comment -> ADAUGA/ACTUALIZEAZA lista automat dupa operatie

- POST/EDIT vezi ID la comment/post ca dupa nu merge edit/delete

- vizibil doar la login(/register)

- put(preluare date existente)

- delete(warning)

- msj eroare

css pt toate componentele

- responsive( vuetify automat)

postare github (sterge todovanilla si altele nerelevante)

async login() {

try {

await this.$refs.form.validate();

if (this.valid) {

this.error = false;

axios.post('http://localhost:3001/login', {

email: this.email,

password: this.password,

})

.then((response) => {

if (response.data.token) {

localStorage.setItem('token', response.data.token);

console.log("Login successful with", this.email, this.password);

console.log("Token stored:", response.data.token);

} else {

console.error("Login failed. No token received.");

this.error = true;

this.errorMessage = response.data.error ? response.data.error.message : "An error occurred during login.";

}

})

.catch((error) => {

console.error("Error during login:", error);

this.error = true;

this.errorMessage = error.response ? error.response.data.error.message : "An error occurred during login.";

});

} else {

this.error = true;

this.errorMessage = "Please input valid data";

console.log("Please input valid data", this.name);

}

} catch (error) {

console.error("Error during login:", error);

this.error = true;

this.errorMessage = error.response ? error.response.data.error.message : "An error occurred during login.";

}

},

//vechi

async login() {

  try {

    await this.$refs.form.validate();

    if (this.valid) {

      this.error = false;

      const response = await axios.post('http://localhost:3001/login', {

        email: this.email,

        password: this.password,

      });

      console.log("HELLO", response);

      if (response.data.token) {

        localStorage.setItem('token', response.data.token);

        console.log("Login successful with", this.email, this.password);

        console.log("Token stored:", response.data.token);

      } else {

        console.log("HELLO2");

        console.error("Login failed. No token received.");

        this.error = true;

        this.errorMessage = response.data.error ? response.data.error.message : "An error occurred during login.";

      }

    } else {

      this.error = true;

      this.errorMessage = "Please input valid data";

      console.log("Please input valid data", this.name);

    }

  } catch (error) {

    console.error("Error during login:", error.message, response.error);

    this.error = true;

    this.errorMessage = error.response ? error.response.data.error.message : "An error occurred during login.";

  }

},

// Salvare token în localStorage cu expirare

localStorage.setItem('token', response.data.token);

// Salvare timestamp în localStorage pentru a ști când tokenul expiră

const expirationTime = new Date().getTime() + 60 \* 60 \* 1000; // 1 oră în milisecunde

localStorage.setItem('tokenExpiration', expirationTime);

// ...

// Verificare dacă tokenul este încă valid înainte de a face o cerere

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

if (tokenExpiration && new Date().getTime() < tokenExpiration) {

// Tokenul este încă valid, puteți face cereri

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

// Faceți cereri cu tokenul

} else {

// Tokenul a expirat, poate fi necesară o nouă autentificare

// sau puteți redirecționa utilizatorul la pagina de autentificare

}

<!-- ... restul codului ... -->

<v-card-actions>

<v-spacer></v-spacer>

<v-btn icon @click="likePost(post.id)">

<v-icon :color="post.liked ? 'red' : ''">{{ post.liked ? 'mdi-heart' : 'mdi-heart-outline' }}</v-icon>

</v-btn>

</v-card-actions>

<!-- ... restul codului ... -->

<v-card-actions>

<v-spacer></v-spacer>

<v-btn icon @click="likePost(post.id)">

<v-icon :color="post.liked ? 'red' : ''">{{ post.liked ? 'mdi-heart' : 'mdi-heart-outline' }}</v-icon>

<span class="ml-1">{{ post.likes }}</span>

</v-btn>

</v-card-actions>

methods: {

// ... alte metode ...

async likePost(postId) {

try {

// Trimite o cerere Axios pentru a aprecia sau dezaprecia postarea

const response = await axios.post(`http://localhost:3001/posts/${postId}/like`);

// Actualizează starea postării în funcție de răspunsul primit

const updatedPost = response.data;

const index = this.posts.findIndex(post => post.id === updatedPost.id);

if (index !== -1) {

this.$set(this.posts, index, updatedPost);

}

} catch (error) {

console.error('Error liking post:', error);

}

},

},

async getAllPosts() {

try {

const response = await axios.get('http://localhost:3001/posts');

// Adaugă informații suplimentare în fiecare postare

this.posts = response.data.map(post => ({

...post,

liked: false, // Poți seta implicit la false, deoarece nu știm starea inițială a like-ului

likes: post.likes || 0, // Adaugă numărul total de like-uri

}));

} catch (error) {

console.error('Error fetching posts:', error);

}

},

<!-- ... restul codului ... -->

<v-row class="mb-4">

<v-col cols="12" md="4">

<v-text-field v-model="authorFilter" label="Author" outlined></v-text-field>

</v-col>

<v-col cols="12" md="4">

<v-text-field v-model="countryFilter" label="Country" outlined></v-text-field>

</v-col>

<v-col cols="12" md="4">

<v-select v-model="sortOption" :items="['date', 'likes']" label="Sort by" outlined></v-select>

</v-col>

</v-row>

<!-- ... restul codului ... -->

async getAllPosts() {

try {

const response = await axios.get('http://localhost:3001/posts');

// Adaugă informații suplimentare în fiecare postare

this.posts = response.data.map(post => ({

...post,

liked: false, // Poți seta implicit la false, deoarece nu știm starea inițială a like-ului

likes: post.likes || 0, // Adaugă numărul total de like-uri

}));

// Sortează postările în funcție de opțiunea aleasă

if (this.sortOption === 'date') {

this.posts.sort((a, b) => b.timestamp - a.timestamp); // Sortează descrescător după dată

} else if (this.sortOption === 'likes') {

this.posts.sort((a, b) => b.likes - a.likes); // Sortează descrescător după numărul de like-uri

}

} catch (error) {

console.error('Error fetching posts:', error);

}

},