**UNIVERSITATEA DIN ORADEA  
FACULTATEA DE INFORMATICĂ ȘI ȘTIINȚE**

**SPECIALIZAREA INFORMATICĂ**

**LUCRARE DE LICENȚĂ**

**Coordonator ştiinţific**Lector Dr. Vasile-Emilian Ion

**Absolvent** RIABOKON KAROLINA

**ORADEA**

**2025**

**UNIVERSITATEA DIN ORADEA  
FACULTATEA DE INFORMATICĂ ȘI ȘTIINȚE   
SPECIALIZAREA INFORMATICĂ**

**BLOG - INSPIRAȚIE**

**Coordonator ştiinţific**Lector Dr. Vasile-Emilian Ion

**Absolvent** RIABOKON KAROLINA

**Cuprins**

[**Capitolul 1. Introducere 4**](#_Toc193960326)

**1.1.** [**Structura lucrării 4**](#_Toc193960327)

**1.2.** [**Motivul lucrării 4**](#_Toc193960328)

**1.3.** [**Obiectivul lucrării 5**](#_Toc193960328)

[**Capitolul 2. Tehnologii folosite 5**](#_Toc193960330)

[**2.1. React 5**](#_Toc193960331)

[**2.2. Firebase 6**](#_Toc193960332)

[**2.3. Structura aplicației**](#_Toc193960333) **9**

[**Capitolul 3. Funcționalități principale ale aplicației**](#_Toc193960334) **14**

[**3.1. Structura generală a aplicației**](#_Toc193960335) **14**

[**3.2. Navigația și rutarea**](#_Toc193960336) **15**

[**3.3. Autentificarea și gestionarea conturilor**](#_Toc193960337) **16**

[**3.4. Integrarea și utilizarea bazei de date Firestore în aplicație**](#_Toc193960337) **25**

[**Capitolul 4. Interfața utilizatorului și funcționalitățile aplicației**](#_Toc193960338) **29**

[**4.1. Interfața utilizatorului și meniul de navigare**](#_Toc193960339) **29**

[**4.2. Admin panel și secțiunea de comentarii**](#_Toc193960340) **33**

[**Concluzie**](#_Toc193960342) **41**

[**Bibliografie**](#_Toc193960343) **41**

**Capitolul 1. Introducere  
  
1.1. Structura lucrării**

Lucrarea este structurată in 4 capitole:  
  
 **Capitolul 1.** “Introducere”, acest capitol conține descrierea privind scopul lucrării,

motivul alegerii subiectului și structura acestei lucrări.  
  
 **Capitolul 2.** “Tehnologii folosite”, în acest capitol se descrie tehnologiile utilizate

pentru dezvoltatea aplicației (React.js, Firebase).  
  
 **Capitolul 3.** “Funcționalități principale ale aplicației”, acest capitol conține funcțiile

rincipale ale aplicație pentru utilizator.  
  
 **Capitolul 4.** “Interfața utilizatorului și funcționalitățile aplicației”, în acest capitol se descrie

despre interfața utilizatorului și al administratorului.

**1.2. Motivul lucrării**  
  
Lucrarea din față descrie aplicația web „Inspirație”. În zilele noastre, popularitatea bunăstării fizice și mentale a crescut foarte mult. Atenția oamenilor este mai concentrată asupra vieților lor, aspirând la un stil de viață sănătos. Subiectul acesta a devenit popular în ultimii zeci de ani. Aplicația această a fost gândită pentru a împărtăși informații utile, și cat mai echilibrate, pentru a arata oamenilor un lifestyle echilibrat din toate punctele de vedere.

M-am hotărât să creez o aplicație, pentru a împărtăși cu oameni pasiunile mele, cu ceea ce trăiesc în prezent și pentru a-i motiva să se găsească la un stil de viață mai sănătos și echilibrat, de a se regăsi pe ei însuși. Prin această aplicație am împărtășit informații precum: lectura, nutriția, sportul și mindfulness. Ideea principală a acestei aplicații este de a inspira oamenii cu pasiunile mele, să se regăsească într-un stil de viață mai sănătos, cat mental, atât si fizic.

Aplicația „Inspirație” poate fi percepută ca un fel de blog, aceasta a fost bine structurată în categorii, iar fiecare informație din categoria respectivă este foarte importantă. Informațiile sunt colectate de mine, le folosesc în fiecare zi pentru a-mi face viața mai frumoasă, al stilului meu de viață sănătos. În proiect sunt incluse tehnologiile folosite, structura aplicației, implementarea funcționalității și altele care au fost folosite pentru o bună funcționalitate a aplicației.

**1.3. Obiectivul proiectului**  
Scopul lucrării este de a dezvolta o aplicație ușor utilizabilă, informabilă, pentru a bucura utilizatorii cu obiceiuri frumoase și cu o experiență plăcută atunci când accesează aplicația aceasta, care oferă funcționalitățile precum:

* **Autentificare și gestionare a conturilor utilizatorilor;**
* **Publicarea și organizarea de articole în categorii distincte: cărți, sport și rețete;**
* **Crearea unei baze de date care să stocheze articolele și informațiile utilizatorilor;**
* **Implementarea unei interfețe atractive și responsive, care să ofere o experiență plăcută pe dispozitive diverse.**

**Capitolul 2. Tehnologii folosite**Proiectul a fost realizat cu tehnologia React pentru interfața web, Firebase Hosting pentru găzduire, și Firebase Database pentru informație. Pentru dezvoltare, efort artistic este necesar și pentru proiectare, așa cum se întâmplă și logica utilizată în JavaScript și React.

**2.1. React**React.js este o bibliotecă open-source pentru limbajul de programare JavaScript, pentru dezvoltarea interfețelor cu utilizatorul. Acesta permite crearea aplicațiilor eficiente prin folosirea unui DOM virtual, care optimizează actualizările interfeței fără a reîncărca întreaga pagină. React are o comunitate puternică, una dintre cele mai populare biblioteci pentru dezvoltarea web.  
  
React este o bibliotecă open-source dezvoltată de Facebook. Aceasta oferă instrumente pentru crearea de interfețe cu utilizatorul folosind arhitectura componentelor. React este o bibliotecă, nu un cadru complet pentru dezvoltarea aplicațiilor.  
  
**Diferența dintre cadru (framework) și o bibliotecă:**

* **Cadru (framework)** – oferă o soluție cuprinzătoare care oferă un set de instrumente, biblioteci și reguli pentru dezvoltarea aplicațiilor. Cadrul definește adesea structura aplicației, iar dezvoltatorului i se oferă mai puțină libertate în alegerea instrumentelor și a deciziilor arhitecturale. Exemple de cadre includ Angular și Ember.
* **Biblioteca** – oferă un set de funcții și componente care ajută la anumite sarcini, dar nu impun arhitectura generală a aplicației. Dezvoltatorul are o mare libertate în alegerea altor instrumente și soluții arhitecturale. React este o bibliotecă care se concentrează pe construirea de interfețe cu utilizatorul.

React a fost introdus pentru prima dată în 2013. A fost creat pentru a rezolva problema actualizării eficiente a interfeței cu utilizatorul în timp real, fără a necesita o reîncărcare a paginii. Proiectat inițial pentru aplicații web, React și-a găsit rapid drumul și în dezvoltarea mobilă, cu React Native. Una dintre ideile cheie ale React este de a descrie declarativ interfețele utilizatorului folosind componente reutilizabile, ceea ce face codul mai ușor de citit și mai ușor de întreținut.

**2.2. Firebase**  
  
Firebase este o platformă **Backend-as-a-Service (BaaS)** oferită de Google. Aceasta pune la dispoziție un set cuprinzător de instrumente și servicii care le permit dezvoltatorilor să creeze, să ruleze și să extindă cu ușurință aplicații mobile și web. Include o bază de date în timp real, autentificare, stocare de date, găzduire și alte capabilități, toate gestionate de pe o singură platformă.

Firebase folosește o bază de date în timp real care sincronizează automat datele pe toate dispozitivele conectate. Aceasta utilizează un model de date NoSQL orientat pe documente, permițând dezvoltatorilor să stocheze date într-un mod flexibil și scalabil. Datele sunt stocate în format JSON. Baza de date acceptă tranzacții atomice și notificări de evenimente în timp real, oferind o experiență fluidă utilizatorului.

Firebase oferă, de asemenea, servicii de autentificare robuste, care facilitează implementarea de către dezvoltatori a autentificării sigure a utilizatorilor în aplicațiile lor. Acceptă mai mulți furnizori de autentificare cum ar fi:

* **E-mail/parolă**
* **Număr de telefon**
* **Furnizori populari terți: Google, Facebook și X(anterior Twitter).**

**Alte servicii oferite de Firebase**

Firebase oferă multe servicii care își completează funcțiile de bază:

* **Firebase Hosting** - Găzduire simplă și rapidă pentru aplicații web și site-uri statice.
* **Firebase Cloud Functions** - Depozit de date în cloud conceput pentru aplicații scalabile care acceptă o structură de date flexibilă.
* **Firebase Storage** - Stocare eficientă pentru fișiere multimedia, oferind distribuție și gestionare sigură a datelor.
* **Firebase Analytics** - Ajută dezvoltatorilor să primeasca rapoarte despre comportamentul utilizatorilor în aplicație.
* **Firebase Cloud Messaging (FCM)** - Permite trimiterea notificărilor către dispozitivele utilizatorilor.
* **Firebase Performance Monitoring** - Acesta permite urmărirea valorilor cheie, cum ar fi CPU, memorie și utilizarea traficului în rețea. Acestea pot fi folosite pentru a înțelege cum funcționează aplicația pe dispozitivele utilizatorilor.

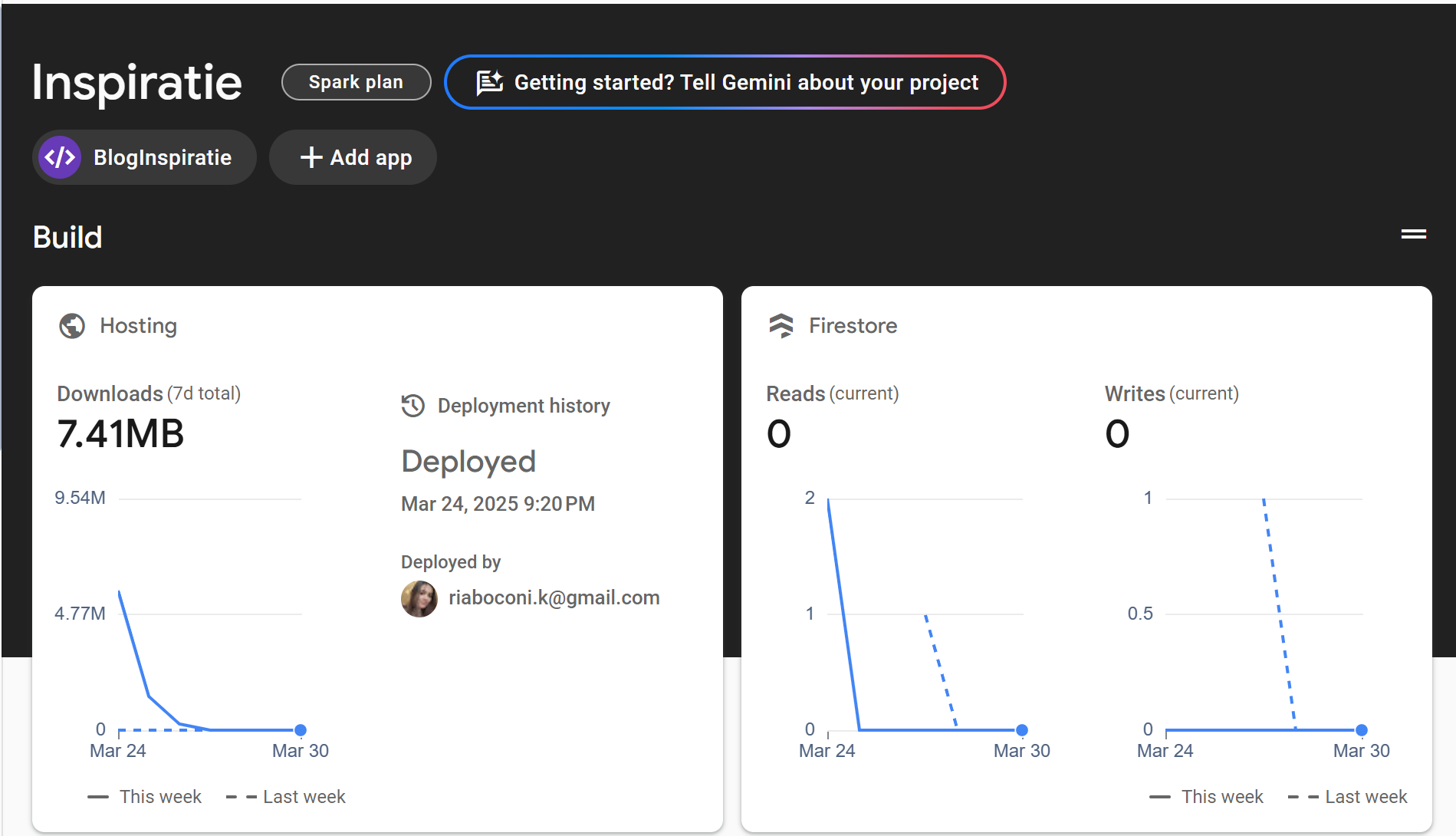
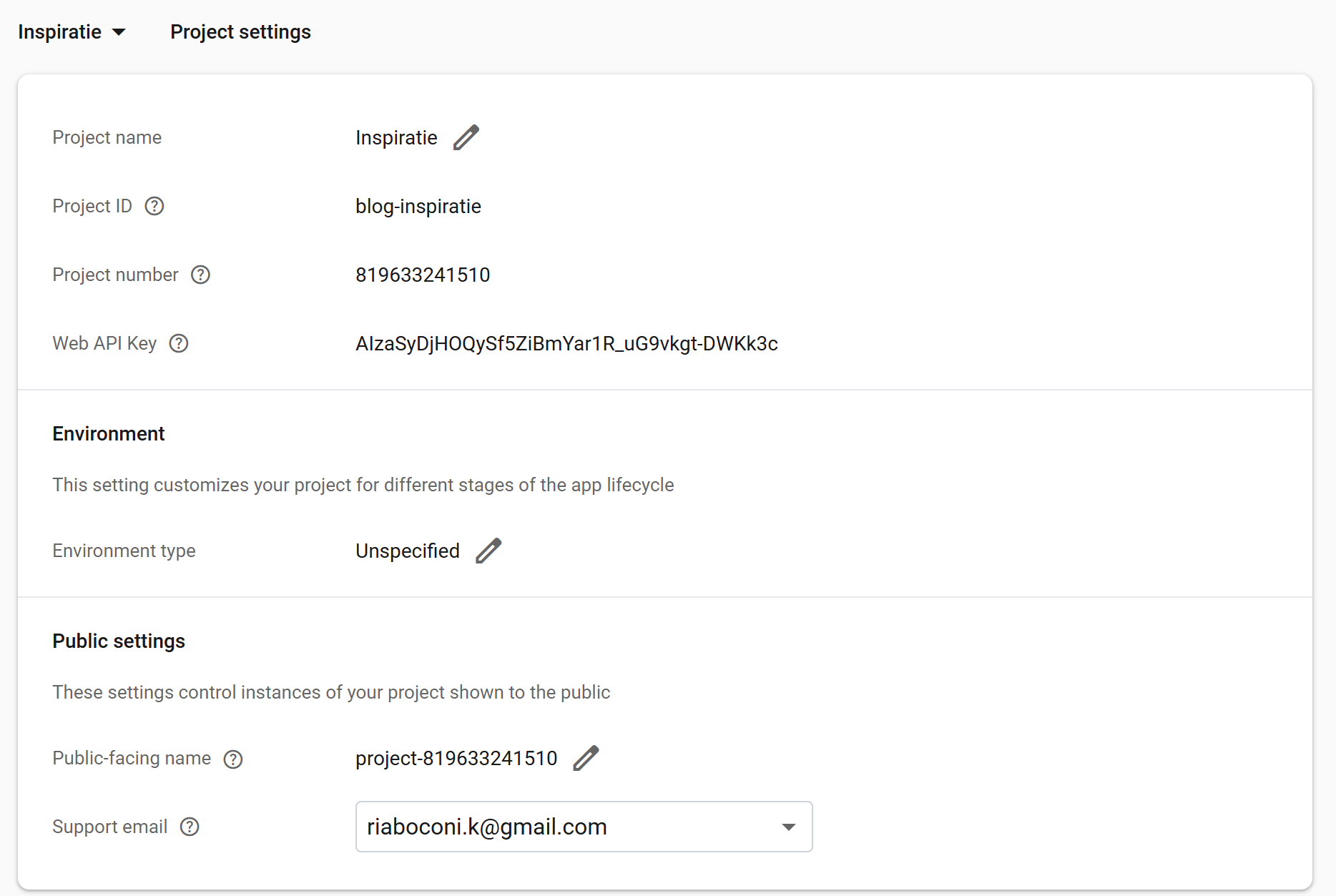


Fig. 2.2.1. Meniu unde se creează o aplicație noua în firebase

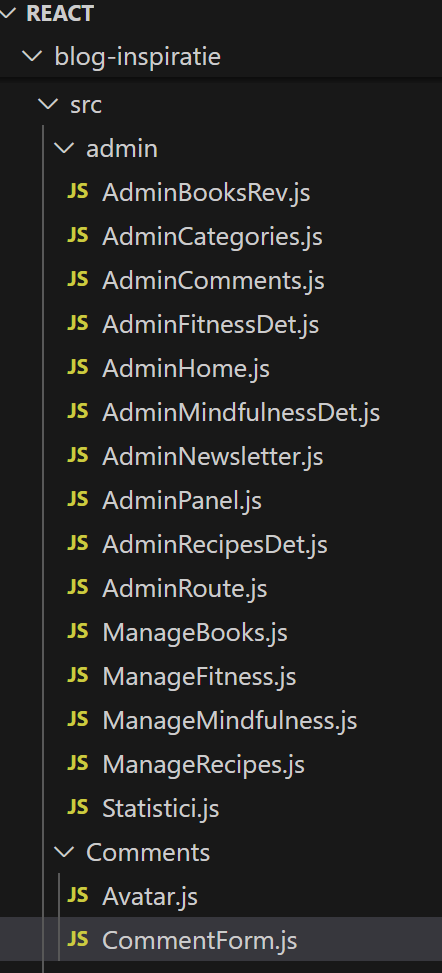
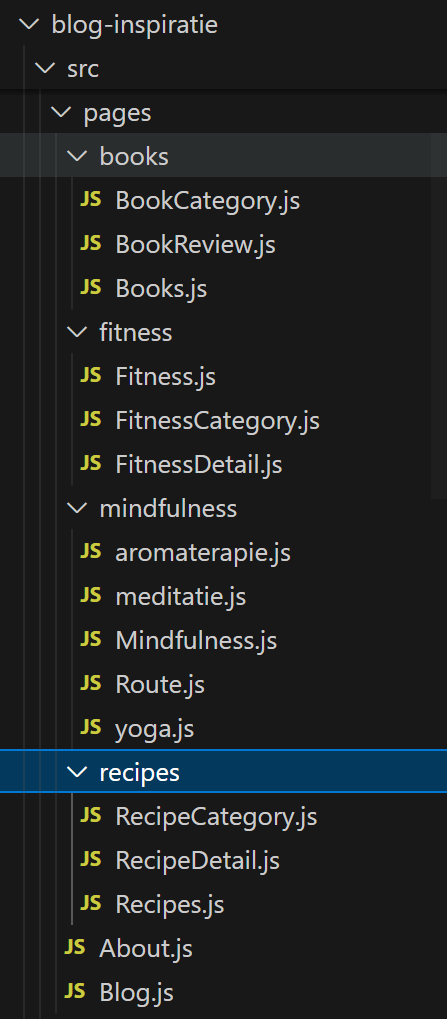
**Avantajele și dezavantajele folosirii Firebase**

* **Economie de timp.** Datorită faptului că platforma are multe instrumente gata făcute, nu va trebui să petreceți mult timp creând un Backend.
* **Versiune gratuită.** Firebase poate fi folosit de freelanceri și studiouri mici.
* **Suport pentru diferite sisteme de operare.** Puteți crea o aplicație pentru sistemele de operare populare, inclusiv iOS și Android.
* **Scalabilitate ușoară.** Când aplicația devine populară și nu există suficiente resurse pentru a o rula, Firebase va crește limitele.
* **Ușurință în utilizare.** Nu aveți nevoie de cunoștințe tehnice complexe de programare pentru a utiliza platforma.
* **Abilitatea de a se concentra pe Frontend.** Deoarece nu este nevoie să programați partea de server, interfața poate fi mai bine dezvoltată.
* **Tot conținutul este stocat în cloud.** Nu va trebui să ocupați spațiu pe dispozitiv. Acest lucru este valabil mai ales pentru aplicațiile cu cantități mari de fișiere video, imagini și audio.

  
Fig. 2.2.2. Meniul ce conține datele aplicației

În acest meniu dezvoltatorul creării aplicației primește date unice. Acesta conține numele proiectului, ID-ul proiectului și cheia de acces API.

**2.3. Structura aplicației**

** **

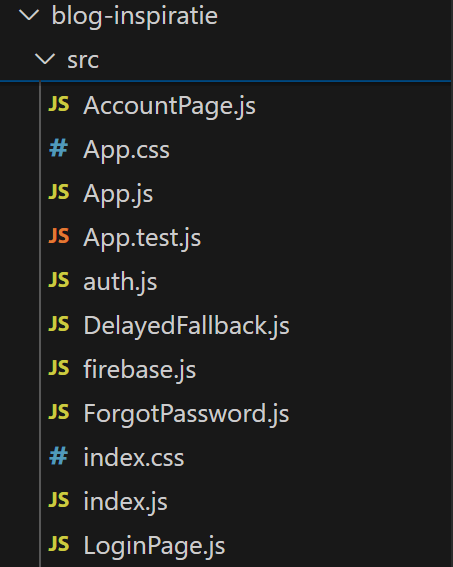
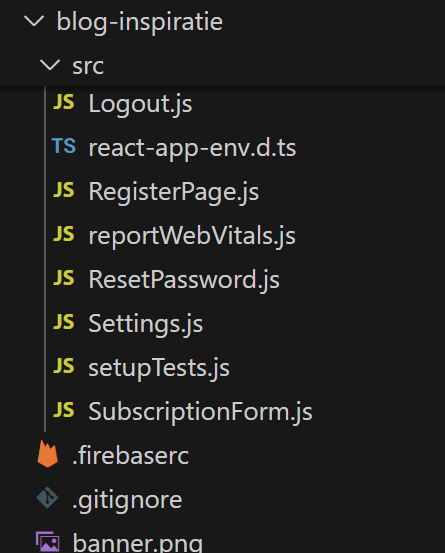
 

Fig. 2.3.1. Conținutul aplicației în foldere și fișiere  
 **Foldere:  
  
blog-inspiratie** reprezintă principalul folder al aplicației React. Acesta conține toate fișierele și folderele aplicației prezentate.  
  
**.firebase** este un folder generat de Firebase, pentru setarea serviciilor Firebase Hosting. Acesta conține setările necesare pentru a publica aplicația online.  
  
**.github** reprezintă un folder convențional folosit pentru a plasa în el lucrări legate de GitHub.  
  
**build** este locul în care este plasată versiunea compilată a activelor atunci când rulezi ***npm build***. Acesta este ceea ce va fi livrat utilizatorului. **dataconnect** folosește tehnologia cloud avansată pentru a oferi aplicației prezentate toate datele într-un format care facilitează crearea rapoartelor accesibile și acționabile de care am nevoie. **public** este folderul care conține toate imaginile de tip (.png, .jpg) ale aplicației, de asemenea conține logo-ul aplicației și fișierul ***index.html*** principal. Aceste fișiere sunt accesibile direct de pe server.  
  
**node\_modules** este un folder care stochează dependențele externe ale proiectului. Ori de câte ori instalez un pachet extern prin „npm”, acesta este stocat în mod implicit în folderul node\_modules situat la rădăcina directorului proiectului.  
**src** conține mai multe fișiere, ce prezintă o funcționalitate bună a aplicației. Conține toate fișierele sursă ale aplicației (.js, .css).  
 **admin** acest foldere conține mai multe fișier cu obiecte importante pentru secțiunea de admin. Cu ajutorul acestui fișier administratorul poate vedea toți utilizatorii logați, posibilitatea de ai șterge, de asemenea există secțiune cu comentarii, unde administratorul poate răspunde din secțiunea aceea la toate comentariile, sau le poate șterge dacă nu respectă regulile aplicației, de asemenea cu ajutorul acestui folder administratorul poate adăuga articole noi pe pagină , de asemenea poate vedea lista utilizatorilor abonați și poate vedea statisticile aplicației.  
 **Comments** cu ajutorul acestui folder se pot adăuga comentarii în aplicație. El se ocupă de sistemul de comentarii în aplicație. Acesta conține fișierele necesare pentru afișarea, adăugarea și stocarea comentariilor utilizatorilor.  
 **pages** este un folder pe care l-am definit ca un container ce conține mai multe fișiere cu paginile aplicației, are de asemenea folderele ***books, recipes, fitness*** și ***mindfulness***. Acesta conține și fișierele ***Blog*** și ***About.***  
  
**books, recipes, fitness, mindfulness** sunt subfoldere din foldetul principal pages, acestea conțin fișierele corespunzătoare fiecărei secțiuni aparte.

**Fișiere:  
  
index.html** este un fișier HTML care servește drept pagină de pornire pentru un site web. Este primul lucru pe care îl văd utilizatorii într-un site. **index.js** fișierul acționează ca punct de intrare inițial pentru aplicația React codificată în JavaScript. De obicei, întâlnești codul responsabil pentru redarea componentei rădăcină a aplicației în DOM.  
  
**package.json** listează dependențele necesare și intervalele de versiuni ale acestora, dar nu versiunile exacte care trebuie instalate. **package-lock.json** este folosit pentru a se asigura că aceleași dependențe sunt instalate în mod constant în medii diferite și pentru a preveni conflictele cauzate de instalarea diferitelor versiuni. **firebase.js** acest fișier structurează Firebase pentru aplicația în React, el oferă acces la autentificare (auth) și baza de date Firestore (db). L-am folosit în această aplicație pentru a face legătura cu Firebase din orice alt fișier al aplicației prezentate. **auth.js** fișierul se ocupă cu autentificarea utilizatorilor și stocarea datelor în Firestore. **App.js** este fișierul principal al aplicației React. Aici sunt definite rutele și componentele de bază care vor fi afișate pe ecran în funcție de navigarea utilizatorilor. Acesta este totodată centrul aplicației. **ResetPassword.js** acest fișier conține o pagină cu ajutorul căreia utilizatorul poate să își reseteze parola în cazul în care a uitat-o. Pentru resetarea parolei se trimite un email folosind serviciile Firebase. **RegisterPage.js** fișierul de față este o pagină care ajută la afișarea formularului de înregistrare pentru utilizatorii noi. Datorită acestui fișier, utilizatorii își pot crea un cont nou, acesta are de completat câteva câmpuri necesare (email, parolă, nume și prenume). **Logout.js** cu ajutorul acestui utilizatorul se deconecteaza din contul său. Atunci când utilizatorul dorește să se deconecteze, este eliminat din aplicație și trimis către pagina de login. **LoginPage.js** acest fișier este destinat descrierii paginii de logare. Aici, utilizatorii introduc datele de conectare (email și parolă). Aplicația verifică aceste date folosind Firebase Auth pentru a vedea dacă sunt corecte. **ForgotPassword.js** fișierul de față conține un formular care permite utilizatorului cu ajutorul adresei de email, să trimită un link pentru resetarea parolei în cazul uitării acesteia. **Settings.js** fișierul de față se ocupă de partea cu setările. Utilizatorul poate să își adauge informații personale, să își schimbe parola dacă a uitat, să își adauge ziua de naștere sau să adauge genul. De asemenea, după salvarea datelor apare un mesaj, dacă datele au fost adăugate sau nu. **SubscriptionForm.js** acest fișier conține un formular care ajută utilizatorii să se aboneze la newsletter. **AdminBooksRev.js** prin acest fișier administratorul are o pagină care oferă posibilitatea de a adăuga recenziilor noi de cărți. **AdminCategories.js** datorită acestui fișier, administratorul poate selecta categoria dorită pentru a adăuga o carte, rețetă, exercițiu sau a adăuga recenzii sau detalii acestora. **AdminComments.js** cuajutorul acestui fișier administratorul poate adăuga comentarii direct din panoul administrativ sau poate șterge comentariile nedorite.  **AdminFitnessDet.js** acest fișier conține o pagină pentru administrator, pentru ca acesta să poată adăuga detalii despre exercițiile fitness sau pentru a modifica detaliile unui exercițiu. **AdminHome.js** acest fișier conține pagina principală a panoului de administrare, acesta permite să vadă lista utilizatorilor și să aibă posibilitatea de a șterge orice utilizator nedorit, de asemenea acesta vede panelul în partea stângă, cu tot ce poate face în panoul administrativ. **AdminMindfulnessDet.js** acest fișier conține o pagină pentru administrator, pentru a gestiona detalii despre categoria mindfulness. **AdminNewsletter.js** acest fișier conține o pagină pentru ca administratorul să vadă ce utilizatori sunt abonați la aplicație. **AdminPanel.js** acest fișier este pentru a combina toate paginile de administrare și a le pune la dispoziția administratorului pentru o utilizare mai rapidă a aplicației. **AdminRecipesDet.js** datorită acestui fișier administratorul poate gestiona detalii despre rețete sănătoase, de asemenea le poate modifica. **AdminRoute.js** acest fișier asigură că pagina de administrator este protejată, și are acces la ea doar administratorului. **ManageBooks.js** acest fișier conține o pagină care oferă administratorului posibilitatea de a adăuga sau șterge cărți în aplicație. **ManageFitness.js** cu ajutorul acestui fișier administratorul are posibilitatea de a adăuga sau șterge exercițiile în aplicație. **ManageMindfulness.js** în acest fișier, administratorul, dacă dorește să adauge sau să șteargă articole legate de mindfulness din aplicație, folosind această pagină creată cu ajutorul acestui fișier. **ManageRecipes.js** acest fișier creează posibilitatea administratorului de a adăuga sau șterge rețetele în aplicație. **Statistici.js** un fișier pentru a arăta administratorului statisticile legate de utilizatori, conținut sau activitatea din aplicație.  
  
**Avatar.js** cu ajutorul acestui fișier, afișează poza utilizatorului cu inițiala prenumelui său în comentarii. **CommentForm.js** acest fișier este creat pentru formularul, prin care utilizatorii pot adăuga comentarii în aplicație. **AccountPage.js** în acest fișier conține pagina pe care utilizatorii își pot vizualiza și edita datele personale. **Blog.js** acest fișier afișează articolele publicate în aplicație. **About.js** cu ajutorul acestui fișier utilizatorii pot să informeze din start ce reprezintă această aplicație, ce poate găsi pe acest site și de cine a fost creată aplicația. **Recipes.js** acest fișier prezintă lista completă al fiecărei rețete pe categorii. **RecipeCategory.js** datorită acestui fișier sunt afișate rețetele în funcție de categoria lor.   
 **RecipeDetail.js** acest fișier creează o pagină, care explică în detaliu o rețetă anume (ingrediente, pași etc.) **aromaterapie.js** fișierul prezentat creează o pagină în care utilizatorii pot găsi informații despre Aromaterapie, inclusiv uleiurile esențiale şi beneficiile sale. **yoga.js** cu ajutorul acestui fișier sunt publicate informații despre yoga și posturi yoga. **meditatie.js** fișierul acesta creează o pagină cu informații și tehnici despre meditație. **Mindfulness.js** acest fișier afișează lista completă a tuturor subcategoriilor din Mindfulness.  
 **Route.js** fișierul acesta este un sistem de navigare, care pune una dintre cele 3 pagini (Meditatie, Yoga, Aromaterapie) în funcție de adresa URL.  **Fitness.js** acest fișier prezintă lista completă al fiecărui exercițiu pe categorii. **FitnessCategory.js** cu ajutorul acestui fișier sunt filtrate exercițiile pe categorii (abdomen, exerciții superioare, etc.). **FitnessDetail.js** este un fișier ce afișează detalii despre exerciții **Books.js** acest fișier afișează lista completă a tuturor cărților pe categorii. **BookCategory.js** cu ajutorul acestui fișier sunt afișate categoriile de cărți și lista cărților din categoria respectivă. **BookReview.js** acest fișier creează o pagină de cărți, unde utilizatorii pot citi recenzia cărții respective. **App.css** este fișierul de stil ce conține reguli CSS ce definesc aspectul întregii aplicații (culori, fonturi, margini etc.). **Capitolul 3. Funcționalități principale ale aplicației  
  
3.1. Structura generală a aplicației**Aplicația este structurată în mai multe module, fiecare având un rol specific în funcționalitatea generală.Am folosit în aplicație React Router pentru a naviga între pagini, acesta oferind utilizatorilor acces în diferite secțiuni ale aplicației.Modulul principal al aplicației este împărțit în felul următor:

* **Autentificare și cont:** Acesta le oferă utilizatorilor posibilitatea de a se autentifica, înregistra și de a folosi contul propriu. Acest modul are legătură directă cu baza de date Firebase pentru a verifica și a salva informațiile utilizatorilor.
* **Setări și abonare:** Acesta oferă utilizatorilor posibilitatea de a schimba datele personale și de a se abona.
* **Admin și comentarii:** Acesta oferă utilizatorilor posibilitatea de a adăuga comentarii sub fiecare articol, iar secțiunea de admin este creată de a avea controlul aplicației, de a urmări dacă nu sunt comentarii nepotrivite, sau de a adăuga articole noi.
* **Blog și About:** Secțiunea Blog conține linkuri către categoriile principale ale aplicației și o pagină About, unde utilizatorii pot afla mai multe despre platformă.
* **Rețete, Cărți, Sport și Mindfulness:** Aceste părți oferă informații specifice fiecărei categorii:  
    
   **- *Rețete***: Această categorie oferă rețete sănătoase împărțite pe categorii precum  
   feluri principale, supe, salate, deserturi și băuturi.  
    
   **- *Cărți***: recenzii de cărți, organizate pe categorii precum dezvoltarea personală,   
   psihologie, spiritualitate și romantism.  
    
   ***- Fitness***: exerciții pentru fiecare parte a corpului, precum exerciții lower, exerciții   
   upper, abdomen, și câteva secrete pentru sala.  
     
   **- *Mindfulness***: conține trei subcategorii:   
   *Meditație*, beneficii, forme de bază și motive de ce să practici   
    
   meditația.   
    
   *Yoga*, câteva tipuri de yoga, poziții de yoga și beneficii.  
    
   *Aromaterapie*, include informații despre ce este aromaterapia, beneficiile sale,  
   uleiuri esențiale populare, modalități de utilizare și sfaturi pentru utilizarea   
   uleiurilor esențiale.

**3.2. Navigația și rutarea**În aplicația aceasta există elemente importante, cum ar fi navigarea între pagini, aceasta se face cu ajutorulu bibliotecei ***react-router-dom***, este utilizată des în aplicații unde este folosit React, acesta permite aplicarea rutelor în funcție de calea URL.  
  
Codul în aplicație cu Router, Routes și Route:

<Router>  
 <Routes>

        <Route path="/login" element={<LoginPage />} />

        <Route path="/register" element={<RegisterPage />} />

        <Route path="/home" element={user ? <AccountPage user={user}   
 onLogout={handleLogout} /> : <NavLink to="/login" />} />

        <Route path="/about" element={<About />} />

        <Route path="/settings" element={<Settings />} />

        <Route path="/subscription" element={<SubscriptionForm />} />

        <Route path="/" element={<Blog />} />

        <Route path="/books" element={<Books />} />

        <Route path="/forgot-password" element={<ForgotPassword />} />

        <Route path="/reset-password" element={<ResetPassword />} />

        <Route path="/admin" element={<AdminRoute><AdminPanel /></AdminRoute>}>

          <Route path="home" element={<AdminRoute><AdminHome /></AdminRoute>} />

          <Route path="categories" element={<AdminRoute><AdminCategories/>  
 </AdminRoute> />

          <Route path="/admin/comments" element={<AdminRoute><AdminComments />   
 </AdminRoute>} />

        </Route>

   </Routes>

</Router>

Fiecare cale din URL este asociată unei componente React, care este afișată atunci când utilizatorul navighează către ruta respectivă. Prin urmare, aplicația asigură o navigare ușoară, fără să se efectueze reîncărcarea completă a paginii, ceea ce îmbunătățește experiența utilizatorului. **3.3. Autentificarea și gestionarea contului**Autentificarea este o etapă importantă a aplicației. Aceasta oferă utilizatorilor posibilitatea de a-și crea cont, de a se autentifica, de a reseta parola sau de a se deloga, de asemenea utilizatorii care au cont, au posibilitatea de a lăsa comentarii sub articolele dorite și să aboneze la aplicație.  
  
**RegisterPage.js**Această pagină permite utilizatorului să creeze un cont nou. Aici se află un formular ce conține adresa de email, nume, prenume și parolă. După înregistrare, utilizatorul este redirecționat pe pagina de logare automat. Dacă adresa de email este deja utilizată, se afișează un mesaj de eroare.

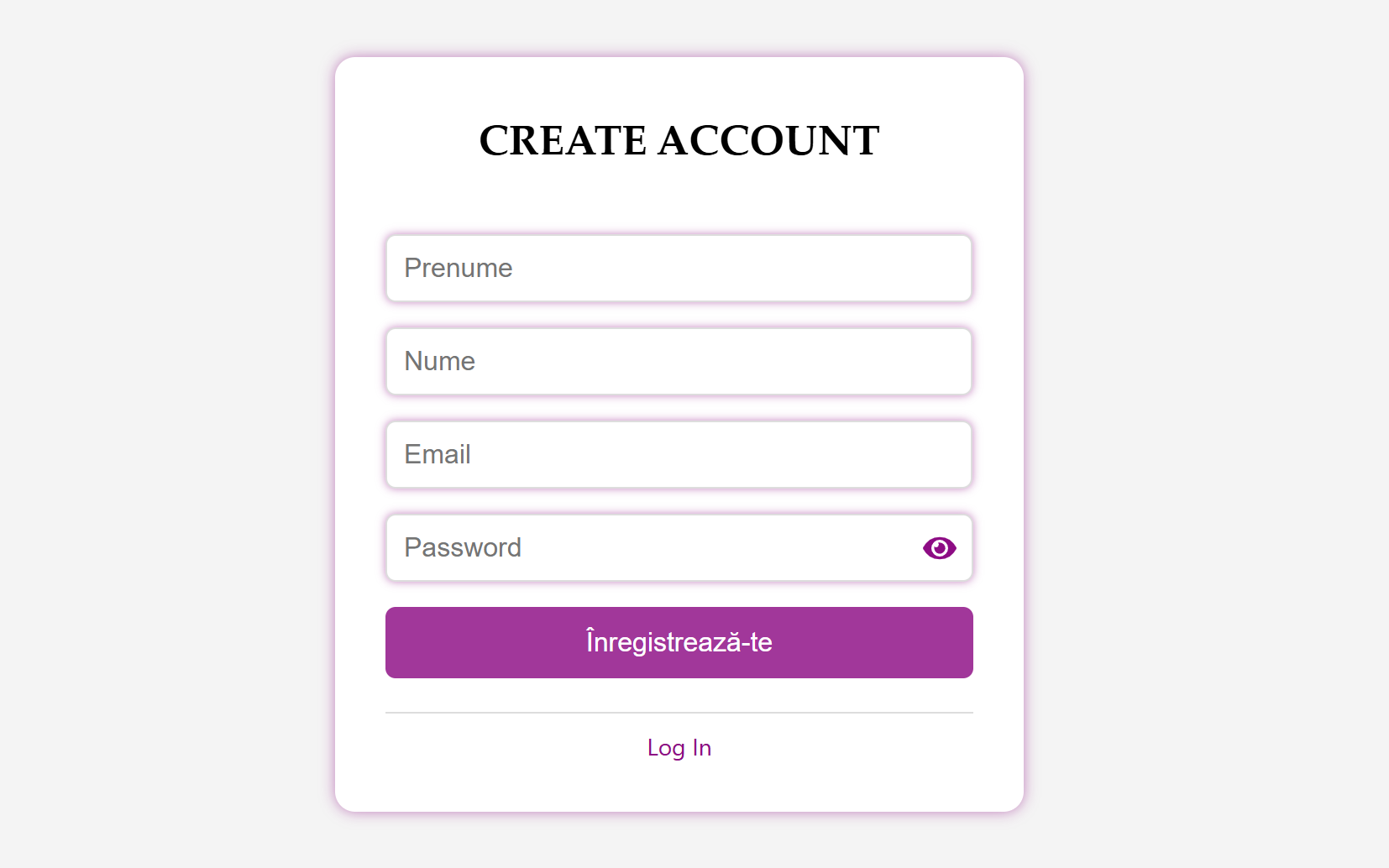


Fig. 3.3.2. Pagina de Register

const handleSubmit = async (e) => {

    e.preventDefault();

    setError('');

    try {

      const userCredential = await createUserWithEmailAndPassword(auth, email, password);

      const user = userCredential.user;

      await setDoc(doc(db, "users", user.uid), {

        firstName: firstName,

        lastName: lastName,

        email: email,

        birthday: "",

        gender: "",

      });

      navigate('/login');

    } catch (error) {

      switch (error.code) {

        case 'auth/email-already-in-use':

          setError('Acest email este deja utilizat, incercati alta.');

          break;

        case 'auth/invalid-email':

          setError('Adresa de email nu este validă.');

          break;

        case 'auth/weak-password':

          setError('Parola este prea slabă. Alege una mai complexă.');

          break;

        default:

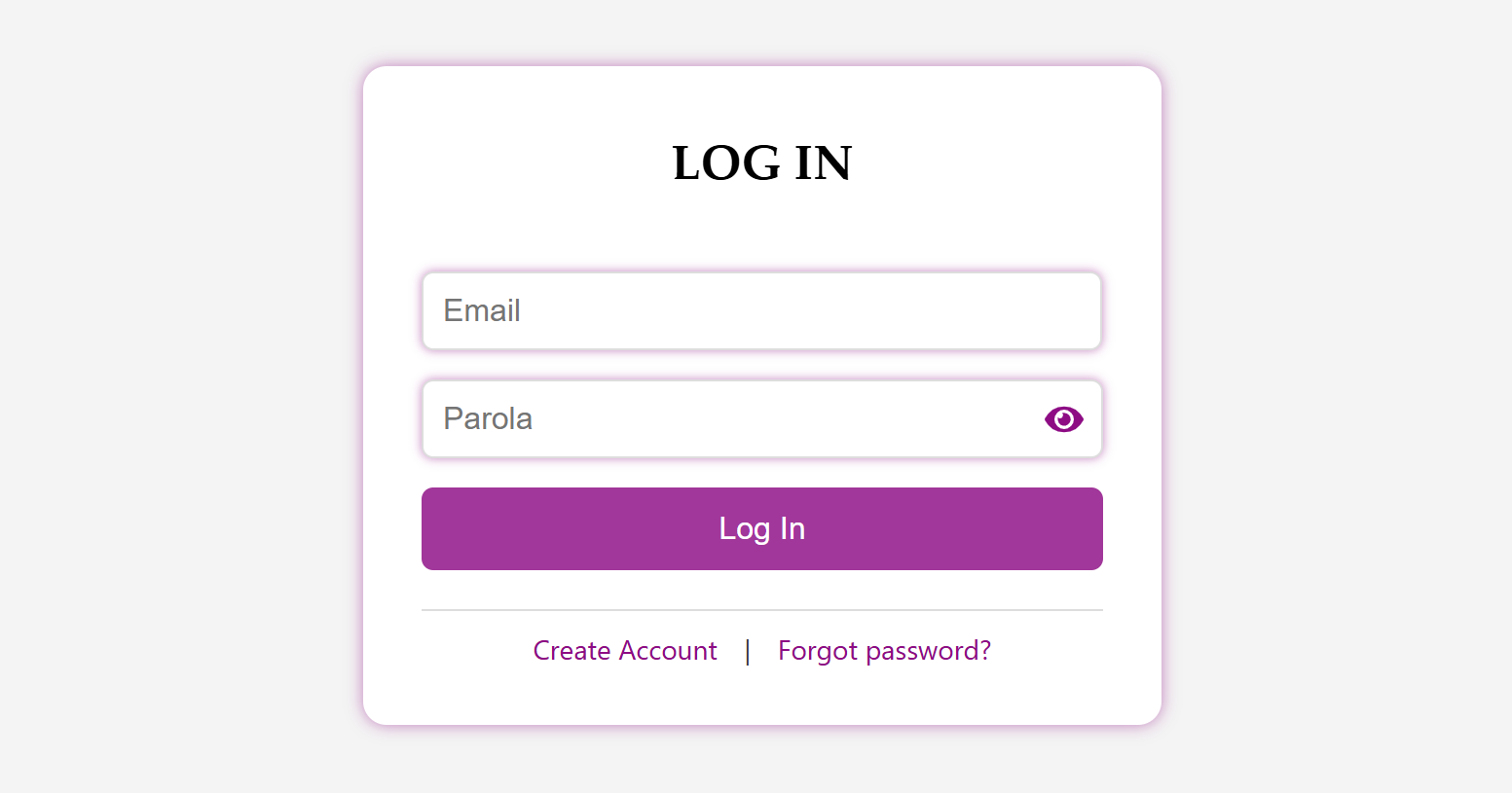
          setError('A apărut o eroare. Te rugăm să încerci din nou.');

      }

    }

  };

**LoginPage.js**Această pagină dată este pentru autentificare. Aici se află un formular care cere utilizatorului să își introducă datele personale – adresa de email și parola. Mai jos, utilizatorul care nu are cont poate accesa opțiunea **Create Account** pentru a-și crea un cont, iar dacă utilizatorul a uitat parola, acesta are posibilitatea de a o reseta apăsând pe **Forgot password?**.

  
 Fig. 3.3.1. Pagina de Login

const handleSubmit = async (e) => {

    e.preventDefault();

    setError("");

    try {

      await login(email, password);

      navigate("/home");

    } catch (error) {

      switch (error.code) {

        case 'auth/invalid-credential':

          setError('Te rog verifică email-ul și parola.');

          break;

        case 'auth/user-not-found':

          setError('Nu am găsit un cont cu acest email.');

          break;

        case 'auth/wrong-password':

          setError('Parola introdusă este greșită. Încearcă din nou.');

          break;

        case 'auth/too-many-requests':

          setError('Ai încercat prea multe logări eșuate. Te rugăm să încerci din nou după câteva minute.');

          break;

        case 'auth/network-request-failed':

          setError('Conexiune eșuată. Verifică-ți conexiunea la internet.');

          break;

        default:

          setError('A apărut o eroare. Te rugăm să încerci din nou.');

      }

    }

  };

**ForgotPassword.js**În cazul în care utilizatorul uită parola, acesta are opțiunea de a introduce adresa de email pentru a primi un link de resetare a parolei, trimis pe adresa folosită la înregistrare. Dacă adresa de email nu există în baza de date, utilizatorul nu va primi niciun email pentru resetare.

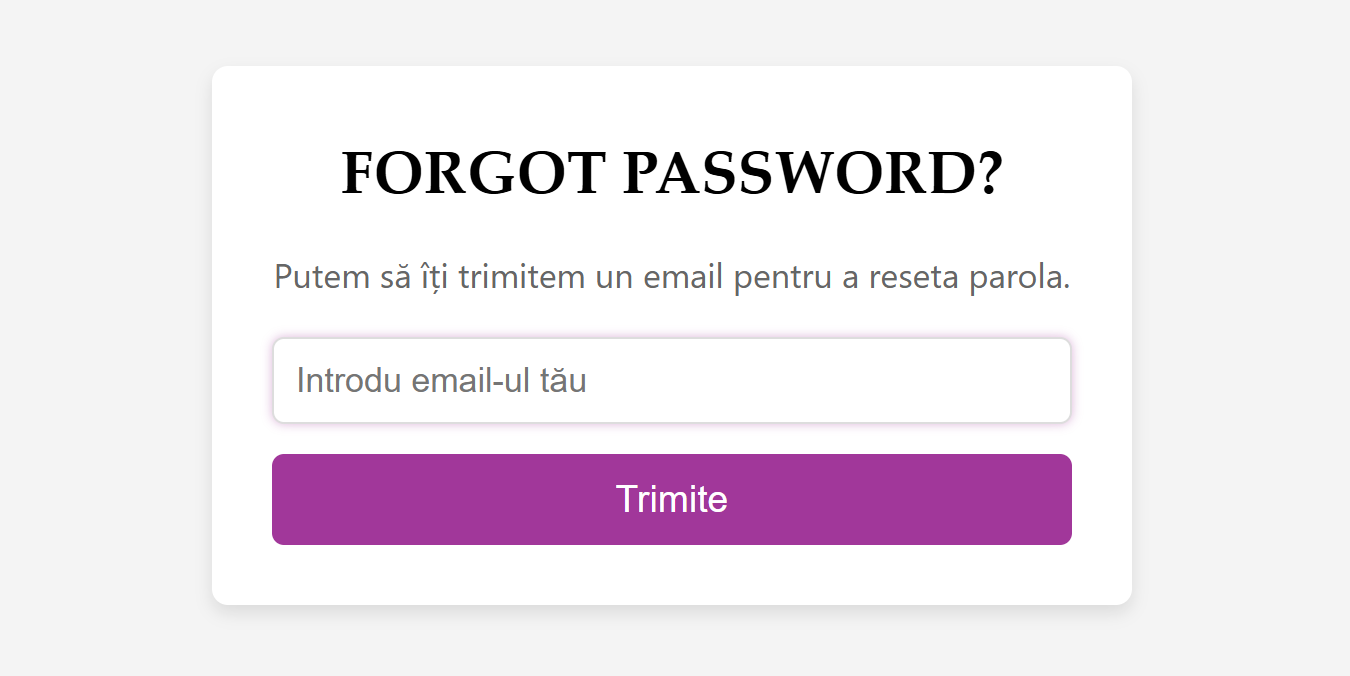


Fig. 3.3.3. Pagina de Forgot password

const handleSubmit = (e) => {

    e.preventDefault();

    if (!emailRegex.test(email)) {

      setError("Emailul introdus nu este valid.");

      return;

    }

    sendPasswordResetEmail(auth, email, actionCodeSettings)

      .then(() => {

        setMessage(

          "Dacă acest email este asociat cu un cont, vei primi instrucțiunile de resetare a parolei."

        );

        setError("");

      })  
 .catch(() => {

        setMessage("");

        setError("A apărut o eroare. Te rugăm să încerci din nou.");

      });

**ResetPassword.js**Această pagină permite utilizatorilor să își reseteze parola dacă au uitat-o. După ce utilizatorul primește un email cu linkul de resetare trimis de Firebase, acesta este redirecționat către această pagină.

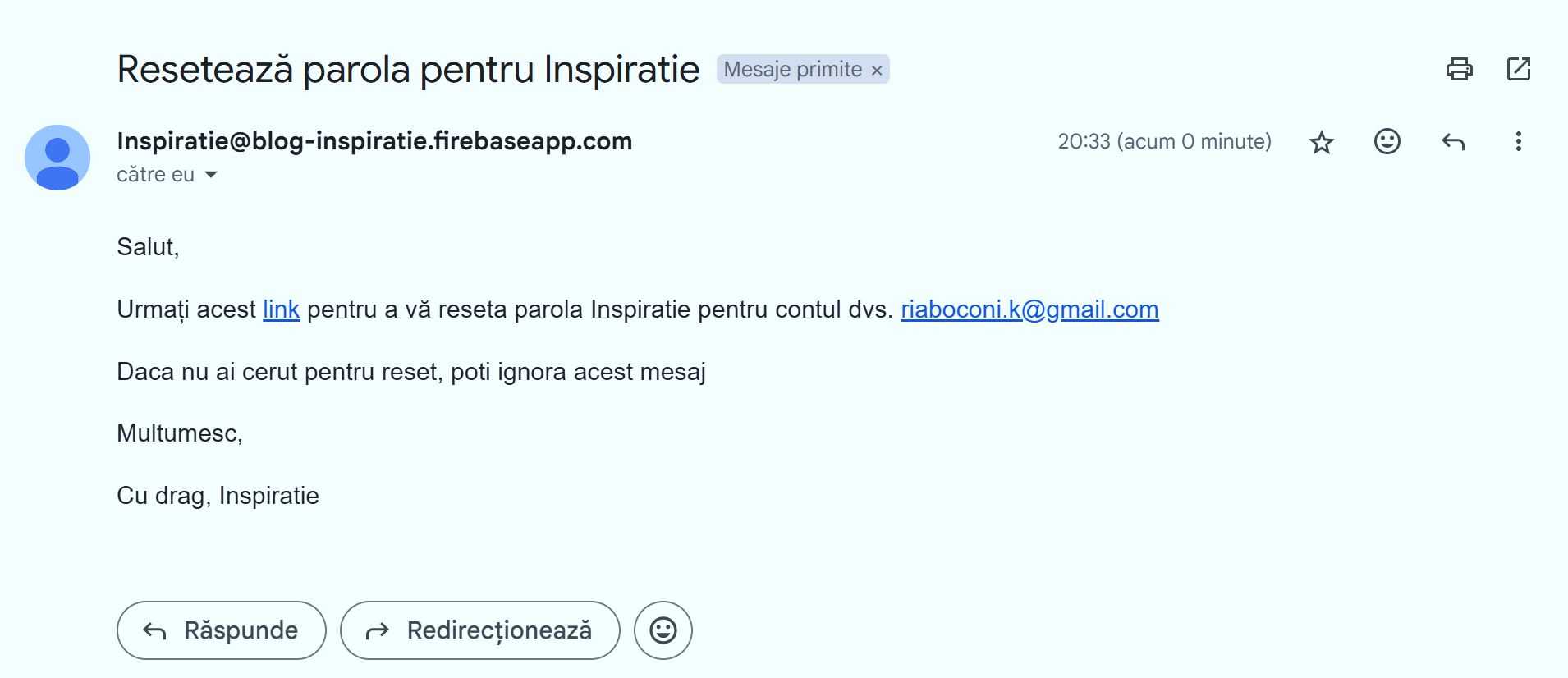
****

Fig. 3.3.4. Email cu link pentru resetarea parolei

Pe această pagină, utilizatorul poate introduce o nouă parolă și o poate confirma. Componenta verifică dacă cele două parole introduse sunt identice. Dacă nu sunt, se afișează o eroare. Dacă sunt identice, parola este actualizată automat în Firebase, folosind metoda ***confirmPasswordReset***. De asemenea, utilizatorii pot vizualiza sau ascunde parola folosind pictogramele cu simbolul „Ochi”.

****

Fig. 3.3.5. Pagina de resetarea parolei

const handleSubmit = (e) => {

    e.preventDefault();

    if (newPassword !== confirmPassword) {

      setIsPasswordMatch(false);

      return;

    }

    const auth = getAuth();

    confirmPasswordReset(auth, oobCode, newPassword)

      .then(() => {

        setMessage("Parola ta a fost resetată cu succes!");

        setError("");

      })

      .catch((error) => {

        setError("A apărut o eroare. Te rugăm să încerci din nou.");

        setMessage("");

      });

  };

**Logout.js**Pentru a se deloga, utilizatorul poate apăsa pe un buton care folosește funcția ***signOut().***

După ce utilizatorul s-a deconectat, utilizatorul va fi trimis automat la pagina de Login, iar dacă apare o eroare, se afișează mesajul „Eroare la deconectare”.

const confirmLogout = () => {

    signOut(auth)

      .then(() => {

        navigate("/login");

      })

      .catch((error) => {

        console.log('Eroare la deconectare:', error.message);

      });

  };

**AccountPage.js**La această pagină au acces doar utilizatorii înregistrați în baza de date.

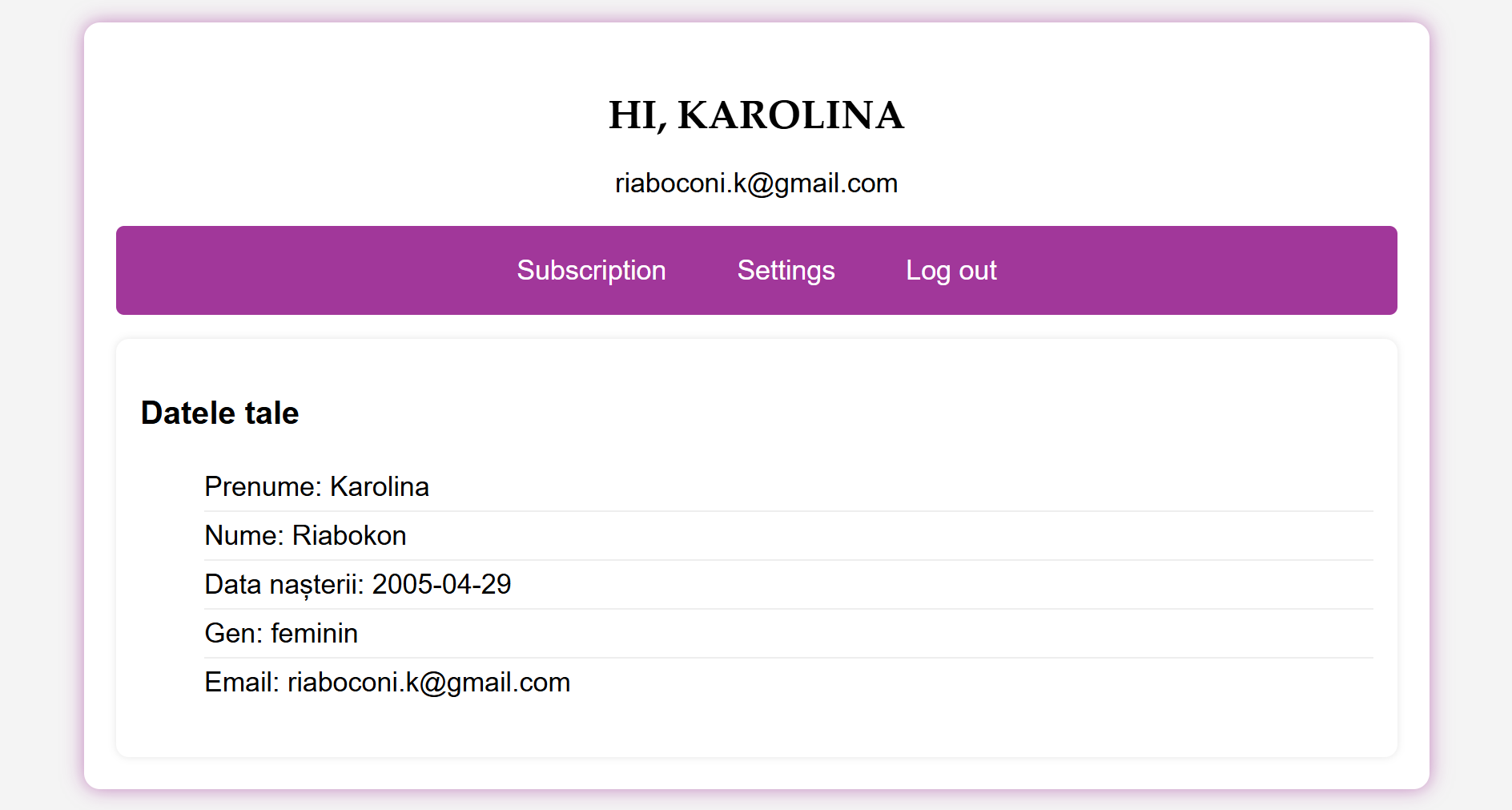
Pe pagina aceasta utilizatorii pot vedea informațiile sale, se pot abona apăsând pe **Subscription**, își pot modifica datele prin butonul **Settings** sau se pot deconecta folosind butonul **Logout**.  
 ****

Fig. 3.3.6. Pagina principală al contului

const unsubscribe = onAuthStateChanged(auth, async (user) => {

      if (user) {

try {

          const userDoc = await getDoc(doc(db, "users", user.uid));

          if (userDoc.exists()) {

            setUserData(userDoc.data());

          } else {

            setUserData(null);

          }

        } catch (error) {

          setUserData(null);

        }

      } else {

        navigate("/login");

      }

      setLoading(false);

    });

    return () => unsubscribe();

  }, [navigate]);

**Settings.js**Acestă secțiune permite utilizatorilor să-și actualizeze datele personale direct din contul lor, într-un mod simplu și rapid. Atunci când se confruntă cu erori, utilizatorilor sunt afișate mesajele potrivite pentru a fi ghidați.

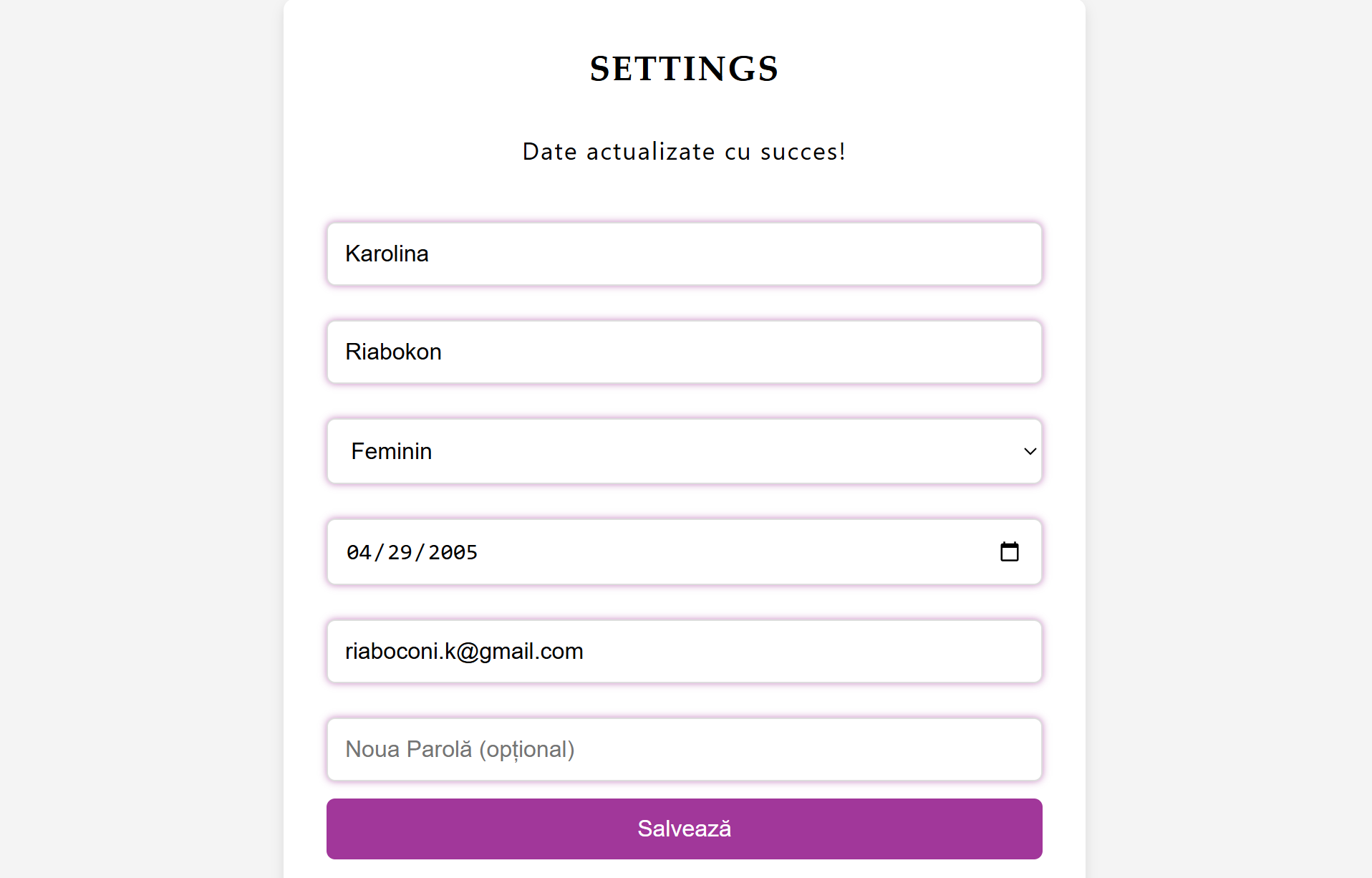
****

Fig. 2.3.7 . Reprezentarea paginii de setări

const handleSave = async (e) => {

    e.preventDefault();

    setError("");

    setSuccessMessage("");

    if (user) {

      try {

        const userRef = doc(db, "users", user.uid);

        await updateDoc(userRef, {

          firstName: userData.firstName,

          lastName: userData.lastName,

          birthday: userData.birthday,

          gender: userData.gender,

        });

        if (userData.email !== user.email) {

          await updateEmail(user, userData.email);

        }

        if (userData.password) {

          await updatePassword(user, userData.password);

        }

        setSuccessMessage("Date actualizate cu succes!");

        setTimeout(() => {

          navigate("/home");

        }, 2000);

      } catch (error) {

        switch (error.code) {

          case 'auth/email-already-in-use':

            setError('Acest email este deja utilizat, încearcă altul.');

            break;

          case 'auth/invalid-email':

            setError('Adresa de email nu este validă.');

            break;

          case 'auth/weak-password':

            setError('Parola este prea slabă. Alege una mai complexă.');

            break;

          default:

            setError('A apărut o eroare. Te rugăm să încerci din nou.');

        }

      }

    }

  };

**3.4. Integrarea și utilizarea bazei de date Firestore în aplicație**  
  
În aplicația acesta este folosit **Firebase Authentication** pentru autentificarea și autorizarea utilizatorilor, pentru abonare și adăugarea comentarilor la articole. **Cloud Firestore** este o bază de date NoSQL care oferă o structură ușor utilizabilă și foarte bună pentru aplicații mai mici, care au nevoie de securitate, dar și simplitate.  
  
Structura bazei de date este simplă, permite un acces rapid la informațiile necesare. Fiecare colecție conține documente cu date specifice pentru a susține funcționarea aplicației:

* **users** – toți utilizatorii sunt stocați într-o colecție numită „users”. Fiecare utilizator are un document identificat printr-un UID (User ID). Documentul include: nume, prenume, adresa de email, data nașterii, genul și rolul (user/admin). Aceste informații pot fi modificate din secțiunea „Settings”.
* **subscribers** – conține adresele de email ale utilizatorilor abonați la newsletter.
* **comments** – păstrează comentariile utilizatorilor. Fiecare document include: adresa de email, utilizatorul care a dat like-ul (likeBy), numărul total de like-uri (likes), mesajul comentariului (message), numele, numele articolului (postID), categoria articolului (postType), eventuale răspunsuri (reply) și ora și data înregistrării (timestamp).
* **recipes** – conține toate rețetele, organizate pe categorii, identificate prin *slug*.
* **recipesDetail** – furnizează informații detaliate pentru fiecare rețetă: ingrediente, pași de preparare etc.
* **fitness** – include exercițiile, organizate pe categorii, detectate prin *slug*.
* **fitnessDetail** – include toate exercițiile. Acesta are informații pentru fiecare exercițiu: beneficiile, mușchii implicați, greșelile frecvente, recomandările și pașii de execuție corectă.
* **books** – are o listă a cărților organizate pe genuri literare. Fiecare carte are detalii precum autor, descriere etc.
* **bookReviews** – colecția dată include toate cărțile, conține review-ul meu despre cartea respectivă.
* **Mindfulness** – această colecție include categorii precum aromaterapie, yoga și meditație. Fiecare document conține o descriere detaliată.
* **mindfulness** – conține toate categoriile de mindfulness, și include o poză și descriere despre fiecare categorie.
* **category-books** – grupează cărțile în funcție de genuri literare.
* **category-recipes** – permite gruparea rețetelor în categorii (ex. supe), facilitând filtrarea acestora.
* **category-fitness** – organizează exercițiile în funcție de tipul lor, pentru o navigare mai eficientă.

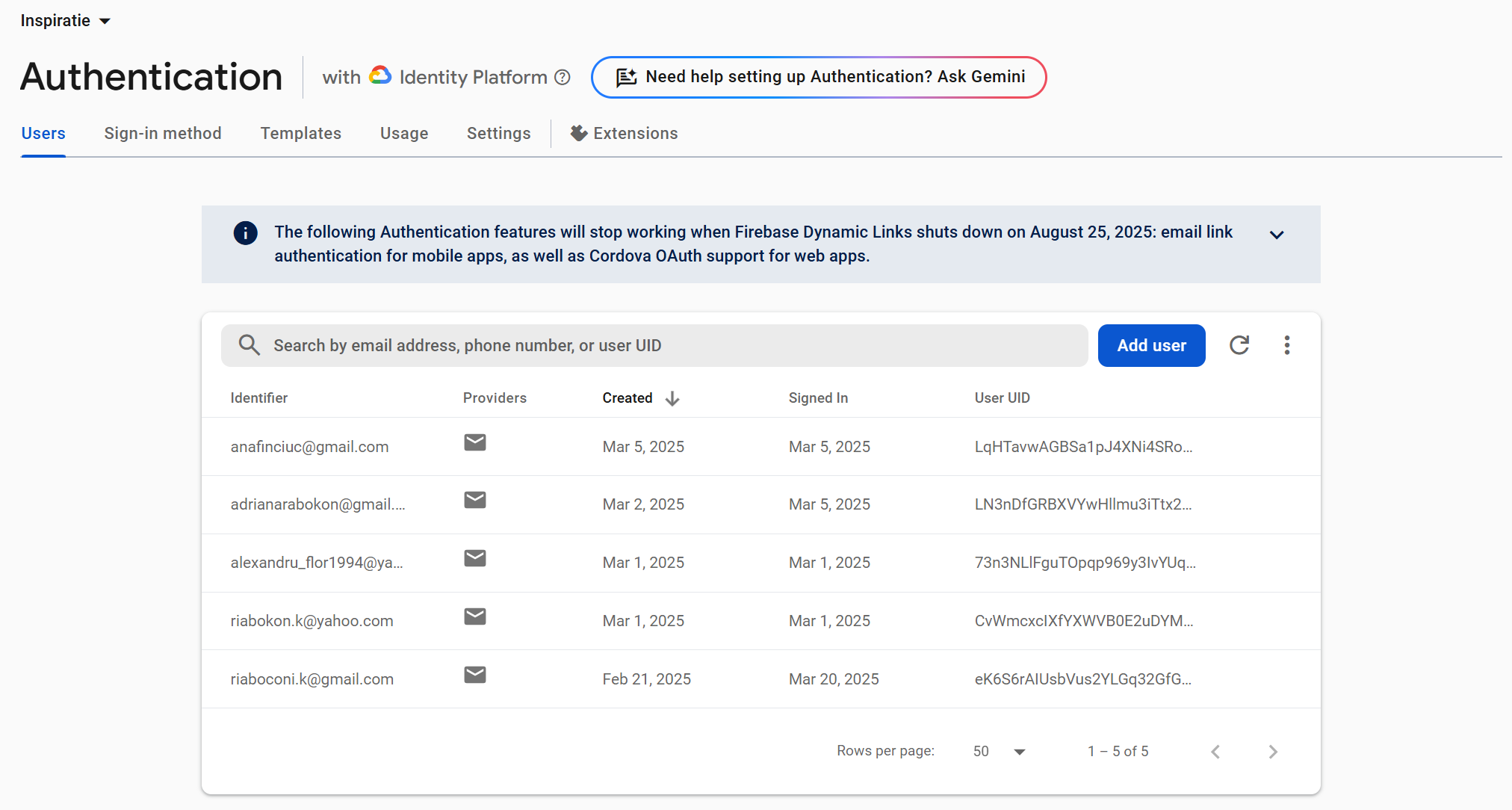


Fig. 3.4.1. Meniul ce conține înregistrarile existențe

Conținut din **firebase.js**

import { initializeApp, getApps } from 'firebase/app';  
import { getAuth } from 'firebase/auth';  
import { getFirestore } from "firebase/firestore";

const firebaseConfig = {

  apiKey: "AIzaSyDjHOQySf5ZiBmYar1R\_uG9vkgt-DWKk3c",

  authDomain: "blog-inspiratie.firebaseapp.com",

  projectId: "blog-inspiratie",

  storageBucket: "blog-inspiratie.firebasestorage.app",  
 messagingSenderId: "819633241510",

  appId: "1:819633241510:web:d828a1d6c815bb1c1ca367"

};

let app;

if (getApps().length === 0) {

  app = initializeApp(firebaseConfig);

} else {

  app = getApps()[0];

}

const auth = getAuth(app);

const db = getFirestore(app);

export { auth, db };

Acest cod inițializează aplicația Firebase și permite accesul la modulele **auth** și **db** pentru autentificare și interacțiunea cu baza de date Firestore.  
  
Conținut din **auth.js**, funcționarea autentificării în aplicația

import { initializeApp } from "firebase/app";  
import { getAuth, createUserWithEmailAndPassword, signInWithEmailAndPassword, signOut } from "firebase/auth";  
import { getFirestore, setDoc, doc } from "firebase/firestore";

const app = initializeApp(firebaseConfig);

const auth = getAuth(app);

const db = getFirestore(app);

const registerUser = async (email, password, firstName, lastName) => {

  try {

    const userCredential = await createUserWithEmailAndPassword(auth, email, password);

    const user = userCredential.user;

    await setDoc(doc(db, "users", user.uid), {

      firstName,

      lastName,

      email: user.email,

      createdAt: new Date(),

    });

    console.log("User registered and added to Firestore successfully!");

  } catch (error) {

    console.error("Error registering user:", error.message);

  }

};

const login = (email, password) => {

  return signInWithEmailAndPassword(auth, email, password);

};

const logout = () => {

  return signOut(auth);

};

export { auth, db, registerUser, login, logout };

Acest cod înregistreaza un utilizator folosind adresa de email, parola, numele și prenumele, iar după stochează informațiile în Firestore prin **registerUser**. Autentifică un utilizator folosind adresa de email și parola, prin login și deconectează utilizatorul folosind **logout**.

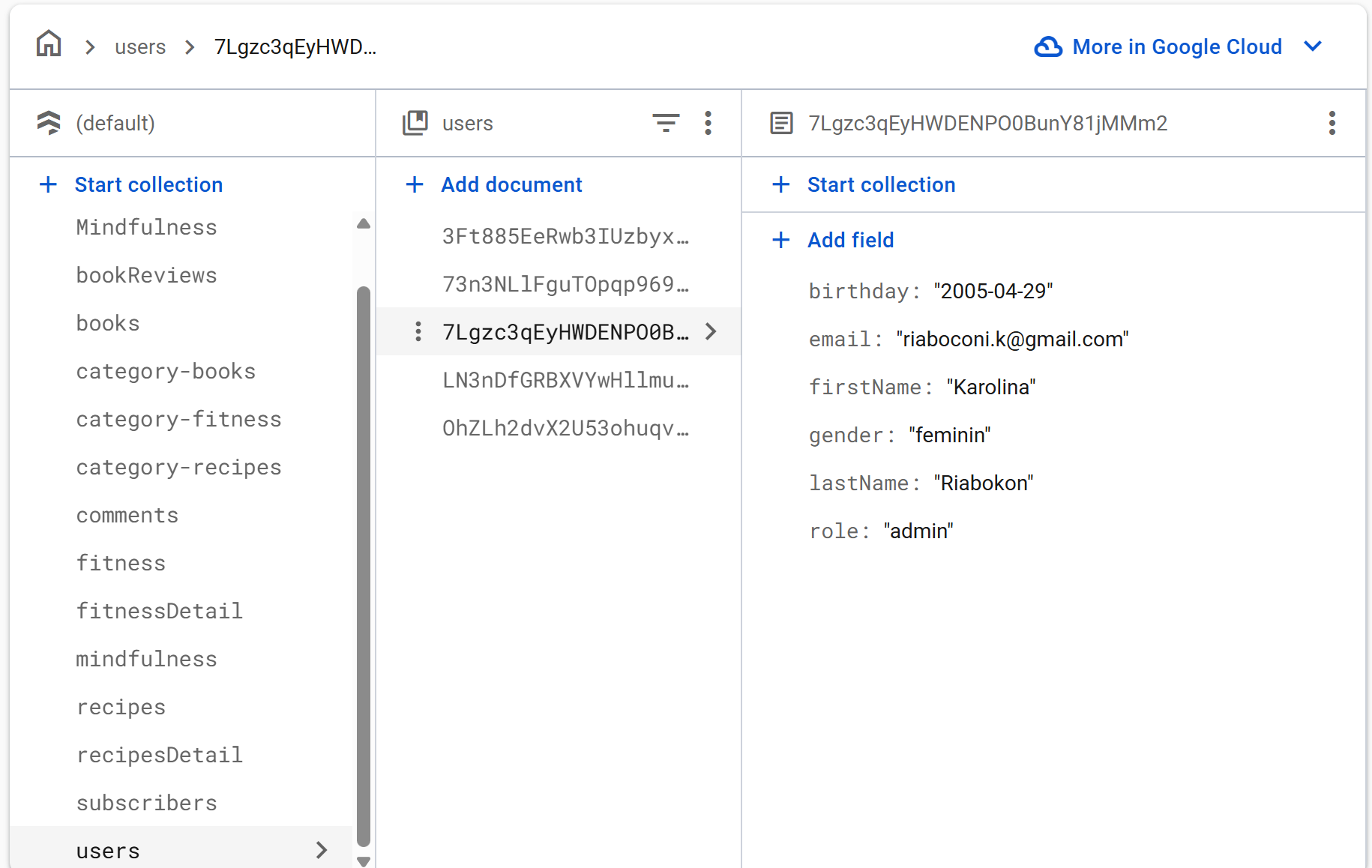


Fig.3.4.2. Structura colecției users si alte colecții în baza de date Firestore

Folosind **Firebase Authentication** și **Firestore**, aplicația obține un sistem sigur, scalabil și ușor de administrat. Structura colecțiilor permite gestionarea eficientă a conținutului și a utilizatorilor, susținând funcționalitatea generală a platformei.   
  
În acest moment, aplicația utilizează o regulă temporară de acces deschis, valabilă până la o anumită dată, pentru a permite dezvoltarea și testarea ușoară.  
  
Am optat pentru Firebase pentru că beneficiază de integrare rapidă cu aplicațiile web moderne, suport pentru autentificare și baze de date în timp real, și totodată, nu necesită un server dedicat. În comparație cu soluții precum MySQL sau MongoDB, Firebase oferă un sistem de securitate integrat și o documentație clară, fiind mai potrivit pentru un proiect de tip blog interactiv.

**Capitolul 4. Interfața utilizatorului și funcționalitățile aplicației  
  
4.1. Interfața utilizatorului și meniul de navigare**Interfața aplicației a fost proiectată într-un mod intuitiv și prietenos, astfel încât utilizatorii să beneficieze de o experiență simplă și eficientă a navigării. Fiecare element este gândit în așa mod, ca să-l dirigeze spre informația pe care o caută.

**Pagina principală**Aplicația este organizată ușor de parcurs pentru utilizator. Pagina principală este afișată imediat ce utilizatorul deschide site-ul.



Fig. 4.1.1. Pagina principală

Aceasta pagină (**fig. 3.5.1**) este pagina principală, include:

* **Logo-ul aplicației**
* **Meniul de căutare** acesta permite de a găsi rapid orice informație utilizatorul doreste
* **Icontița de ulitizator**
* **Meniul de navigare**
* **Categotii vizibile** (Carți, Rețete, Fitness, Mindfulness)
* Fiecare categorie are câte **3 subcategorii** cu imagini și denumiri, utilizatorul de asemenea are posibilitatea de accesa o subcategorie dând clic pe o denumire (imagine)
* Butonul **„See All”** permite accesarea tuturor subcategoriilor

**Codul pentru afișarea subcategoriilor în Blog.js**

const categories = [

  { name: "Books", path: "/books", subcategories: ["Dezvoltare personala", "Romantism", "Spiritualitate"] },

  { name: "Recipes", path: "/recipes", subcategories: ["Feluri principale", "Supe", "Salate"] },

  { name: "Fitness", path: "/fitness", subcategories: ["Exercitii Upper", "Exercitii Lower", "Apdomen"] },

  { name: "Mindfulness", path: "/mindfulness", subcategories: ["Meditatie", "Yoga", "Aromaterapie"] },

];

**Meniul de navigare**  
  
Meniul de navigare reprezintă modaliatea prin care utilizatorul poate naviga ușor prin toate paginile. Această este vizibilă pe toate paginile. De asemenea dropdown-ul apare arunci când utilizatorul trece cu mouse-ul pe o categorie dorită. Meniul include:

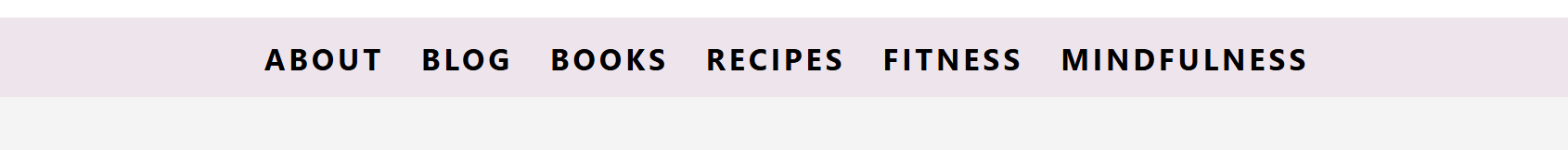


Fig.4.1.2. Meniul de navigare al aplicației

**About** este pagina cu câteva informatii despre aplicație  
  
**Blog** pagina care oferă o scurtă prezentare a aplicației  
  
**Books** un dropdown cu subcategorii: Romantism, Dezvoltarea Personală, Spiritualitate, Psihologie



Fig. 4.1.3. Meniul de navigare al aplicației, al dropdown-ului BOOKS

**Recipes** un dropdown cu subcategorii: Feluri Principale, Salate, Supe, Deserturi, Băuturi

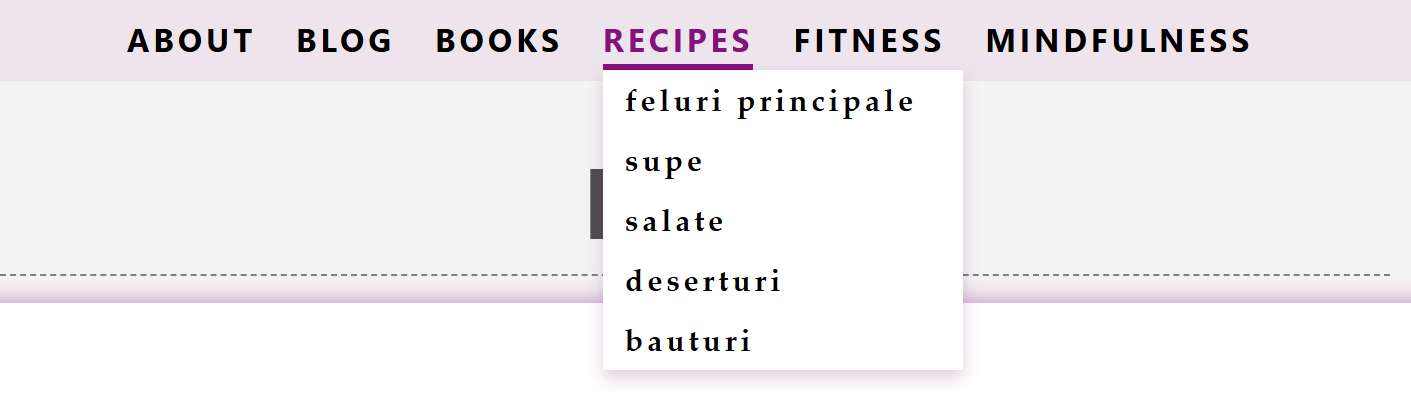


Fig.4.1.4. Meniul de navigare a aplicației, al dropdown-ului RECIPES  
  
**Fitness** un dropdown cu subcategorii: Exerciții Upper, Exerciții Lower, Abdomen, Secrete Gym



Fig.4.1.5. Meniul de navigare al aplicației, al dropdown-ului FITNESS

**Mindfulness** un dropdown cu subcategorii: Meditație, Yoga, Aromaterapie



Fig.4.1.6. Meniul de navigare al aplicației, al dropdown-ului MINDFULNESS

Un fragment din cod pentru meniul de navigare din **App.js**

 <header className="main-header">

        <nav>

          <ul>

            <li><NavLink to="/about">About</NavLink></li>

            <li><NavLink to="/">Blog</NavLink></li>

            <li

              onMouseEnter={() => handleMouseEnter("books")}

              onMouseLeave={handleMouseLeave}

              style={{ position: 'relative' }}>

              <NavLink to="/books">Books</NavLink>

              {dropdown === "books" && (

                <ul className="dropdown-menu"

                  onMouseEnter={() => setDropdown("books")}

                  onMouseLeave={handleMouseLeave}>

                  <li><NavLink to="/books/dezvoltare-personala" className="sub-menu-item">Dezvoltare Personala</NavLink></li>

                  <li><NavLink to="/books/romantism" className="sub-menu-item">Romantism</NavLink></li>

                  <li><NavLink to="/books/spiritualitate" className="sub-menu-item">Spiritualitate</NavLink></li>

                  <li><NavLink to="/books/psihologie" className="sub-menu-item">Psihologie</NavLink></li>

                </ul>

              )}

            </li>

          </ul>

        </nav>

      </header>

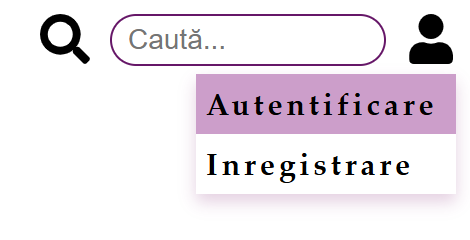
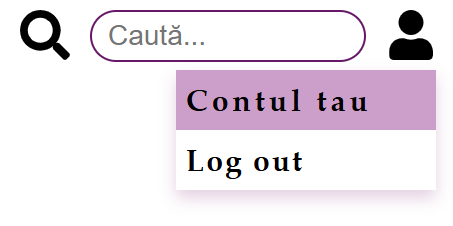
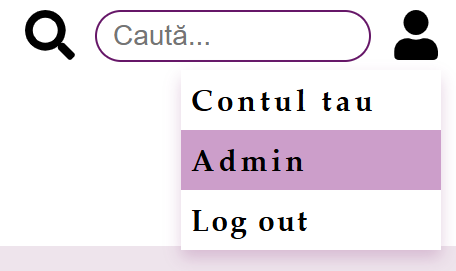
De asemenea, există înca un dropdown. Acesta este în iconița de utilizator. Dacă nu este autentificat, apare *Autentificare* și *Inregistrare*. Iar dacă utilizatorul este autentificat, apare un dropdown cu opțiuni *Contul tau* și *Log out*. De asemena dacă administratorul se autentifică, în dropdown pe langă *Contul tau* și *Log out* apare opțiunea *Admin*.  
  
  

Fig.4.1.7. Dropdown-ul în iconița de utilizator

{userDropdown && (

              <div

                className="dropdown"

                onMouseEnter={handleMouseEnterUser}

                onMouseLeave={handleMouseLeaveUser}

              >

                {!user ? (

                  <>

                    <NavLink to="/login">Autentificare</NavLink>

                    <NavLink to="/register">Inregistrare</NavLink>

                  </>

                ) : (

                  <>

                    <NavLink to="/home">Contul tau</NavLink>

                    {userRole === "admin" && (

                      <NavLink to="/admin/home">Admin</NavLink>

                    )}

                    <Logout className="log-out-dropdown" />

                  </>

                )}

              </div>

            )}

Dacă utilizatorul vrea să iasă din cont, apăsând butonul de *Log out* se afișează mesajul de confirmare, dacă dorește sa iasă utilizatorul, prin **Da** sau **Nu**. Dacă utilizatorul a apăsat Da, acesta revine la pagina de *Login* automat.

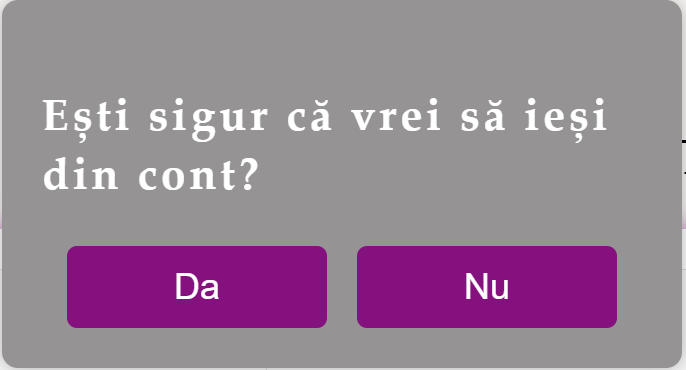


Fig.4.1.8. Mesajul de confirmare pentru Log out

**4.2. Admin panel și secțiunea de comentarii**

**Comentarii**

Toate articolele au o zonă de lăsat comentarii, la capătul paginii. Acest lucru îl poate face doar utilizatorul logat. Dacă utilizatorul nu este logat, acestuia îi apare un mesaj, ce îi zice că ar trebui să se logheze, pentru a lăsa un comentariul.

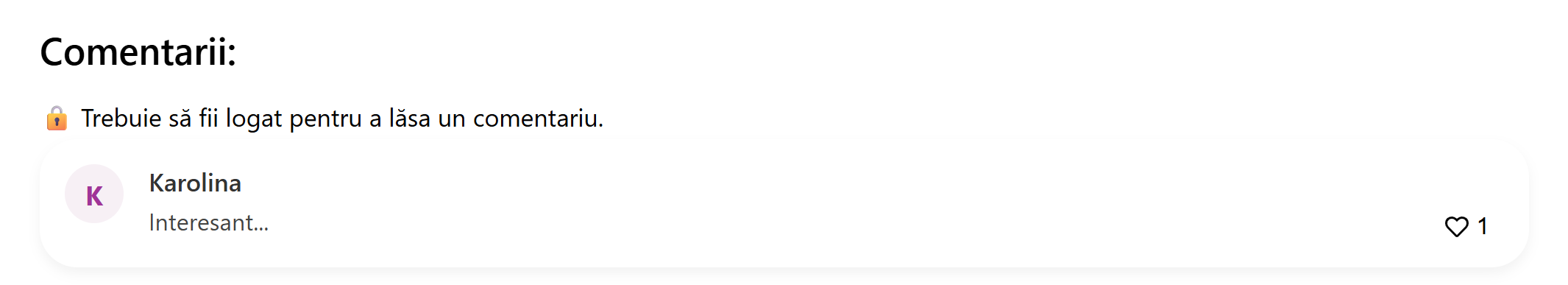


Fig.4.2.1. Secțiunea de comentarii utilizatorului nelogat.

Dacă utilizatorul este logat, acestuia i se oferă posibilitatea de a lăsa comentarii, fără a pune datele lui personale: prenumele si adresa de email. Acestea sunt citite automat din cont și nu pot fi schimbate. După trimiterea comentariului, utilizatorului i se generează automat avatar cu inițiala penumelui și o culoare aleatorie.

Codul pentru a genera inițiala prenumelui din **Avatar.js**

function Avatar({ name }) {

    const initial = name.charAt(0).toUpperCase();

    const bgColor = "#"+((1<<24)\*Math.random()|0).toString(16);

}

Toate comentariile au butonul de apreciere. Dacă un utilizator lasă un like, butonul cu inima va fi plină de culoarea roșie, iar dacă nu a lăsat un like, butonul va fi gol, de culoare gri. Utilizatorul nelogat poate lăsa o apreciere la comentarii.  
  
Utilizatorii nu pot răspunde la comentarii, acest acces are doar administratorul. Raspunsul este atașat sub comentariul inițial, și se poate răspunde doar o dată. De asemenea, administratorul poate șterge comentariile neplăcute din panelul special al administratorului.

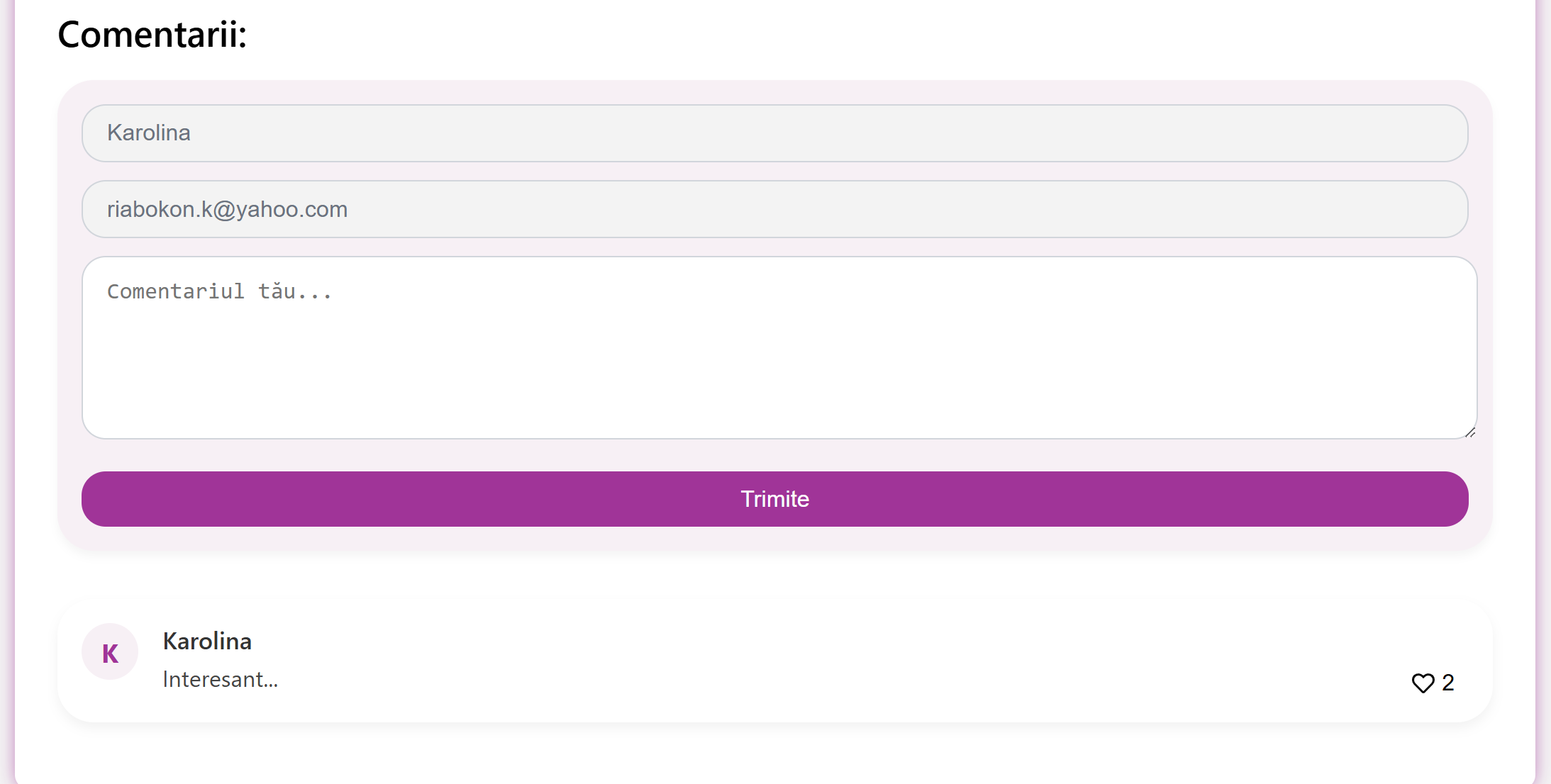
****

Fig. 4.2.2. Secțiunea de comentarii utilizatorului logat.

**Codul pentru afișarea comentariilor**

 <div className="comments-section">

            <h3>Comentarii:</h3>

            <CommentForm postId={bookSlug} postType="Carti" handleAddComment={handleAddComment} />

            {comments.length === 0 ? (

              <p>Nu există comentarii încă.</p>

            ) : (

              comments.map((comment) => (

                <div key={comment.id} className="comment">

                  <div className="avatar">{comment.name[0]?.toUpperCase()}</div>

                  <div className="comment-content">

                    <strong>{comment.name}</strong>

                    <p>{comment.message}</p>

                    {comment.reply && <p className="reply">Răspuns: {comment.reply}</p>}

                    {!comment.replied && userId === ADMIN\_UID && (

                      replyingToId === comment.id ? (

                        <form onSubmit={(e) => handleReplySubmit(e, comment.id)} className="reply-form">

                          <input

                            type="text"

                            value={replyText}

                            onChange={(e) => setReplyText(e.target.value)}

                            placeholder="Scrie un răspuns..."

                          />

                          <div className="form-actions">

                            <button type="submit">Trimite răspuns</button>

                            <button type="button" onClick={() => setReplyingToId(null)} className="cancel-btn">Anulează</button>

                          </div>

                        </form>

                      ) : (

                        <button onClick={() => setReplyingToId(comment.id)} className="reply-btn">Răspunde</button>

                      )

                    )}

                  </div>

                  <div className="like-btn">

                    <button

                      onClick={() => handleLike(comment.id, comment.likes, comment.likedBy)}

                      className={comment.likedBy?.includes(userId) ? "liked" : "not-liked"}

                    >

                      {comment.likedBy?.includes(userId) ? <FaHeart /> : <FaRegHeart />}

                      {comment.likes}

                    </button>

                  </div>

                </div>

              ))

            )}

          </div>

**Admin Panel**

Panoul administrator este o secțiune speciala creată pentru administratori, nimeni nu o poate accesa dacă nu este administratorul, accesul este securizat prin roluri și verificare a adresei de email, am implementat în cod logica de validare a utilizatorului, astfel încât doar administratorul cu adresa de email, să poată intra în cont. De asemenea, Firebase citește rolul utilizatorului din baza de date, dacă este admin sau user. După autentificare, acesta citeste dacă e admin, este trimis direct in **Admin Panel**, iar acesta oferă:

* Vizualizarea utilizatorilor cu datele personale, de asemenea există un buton de **Șterge** utilizatorul, dacă acesta încalcă regulile aplicației, sau alte motive.
* Adăugarea carților, rețetelor, exercițiilor sau categorii de mindfulness, de asemenea detalii sau review-uri.
* Vizualizarea tuturor comentariilor, postate de utilizatori, în acest meniu se poate răspunde sau șterge comentarii neadecvate sau care încalcă regulile platformei.
* Vizualizarea tuturor utilizatorilor abonați.
* Statisticile aplicație.

Mai jos este o porțiune de cod a verificării rolului de admin pentru a intra în cont

 const userRole = email === "riaboconi.k@gmail.com" ? "admin" : "user";

 const handleSubmit = async (e) => {

    e.preventDefault();

    setError("");

    try {

      await login(email, password);

      if (userRole === "admin") {

        navigate('/admin/home');

      } else {

        navigate('/home');

      }

Doar administratorul poate accesa Admin Panel. Dacă un utilizator încearca să acceseze **/admin** acesta va fi redirecționat spre pagina principală **/**. Iar dacă acest lucru înearcă un utilizator nelogat, acesta va fi redirecționat spre **/login**. Acesta este securizat cu ajutorul fișierului **AdminRoute.js**, care detectează dacă este admin sau user.

<Route path="/admin" element={<AdminRoute><AdminPanel /></AdminRoute>}>

          <Route path="home" element={<AdminRoute><AdminHome /></AdminRoute>} />

          <Route path="categories" element={<AdminRoute><AdminCategories /></AdminRoute>} />

</Route>

Crearea articolului nou, este importanta funcție al administratorului, aceasta se face în secțiunea **Categorii**. Administratorul alege în ce categorie dorește să adauge un articol, și este trimis spre pagina sa.



Fig. 4.2.3. Panoul administartiv, secțiunea de adăugarea articolelor

După alegerea categorii, acesta are de adăugat informații despre articol.



Fig. 4.2.4. Panoul administartiv. Secțiunea **Adaugă o rețetă**

Iar după ce s-a adăugat articolul, afișează un mesaj, dacă cartea a fost adăugată sau nu, dacă este adăugată, aceasta este afișată puțin mai jos, în zona ex. **Rețete adăugate** unde se află toate rețetele.



Fig. 4.2.5. Panoul administartiv. Secțiunea **Rețete adăugate**

În secțiunea aceasta administratorul poate șterge articolul dat, sau să fie trimis spre secțiunea de a adăuga date despre această rețetă.

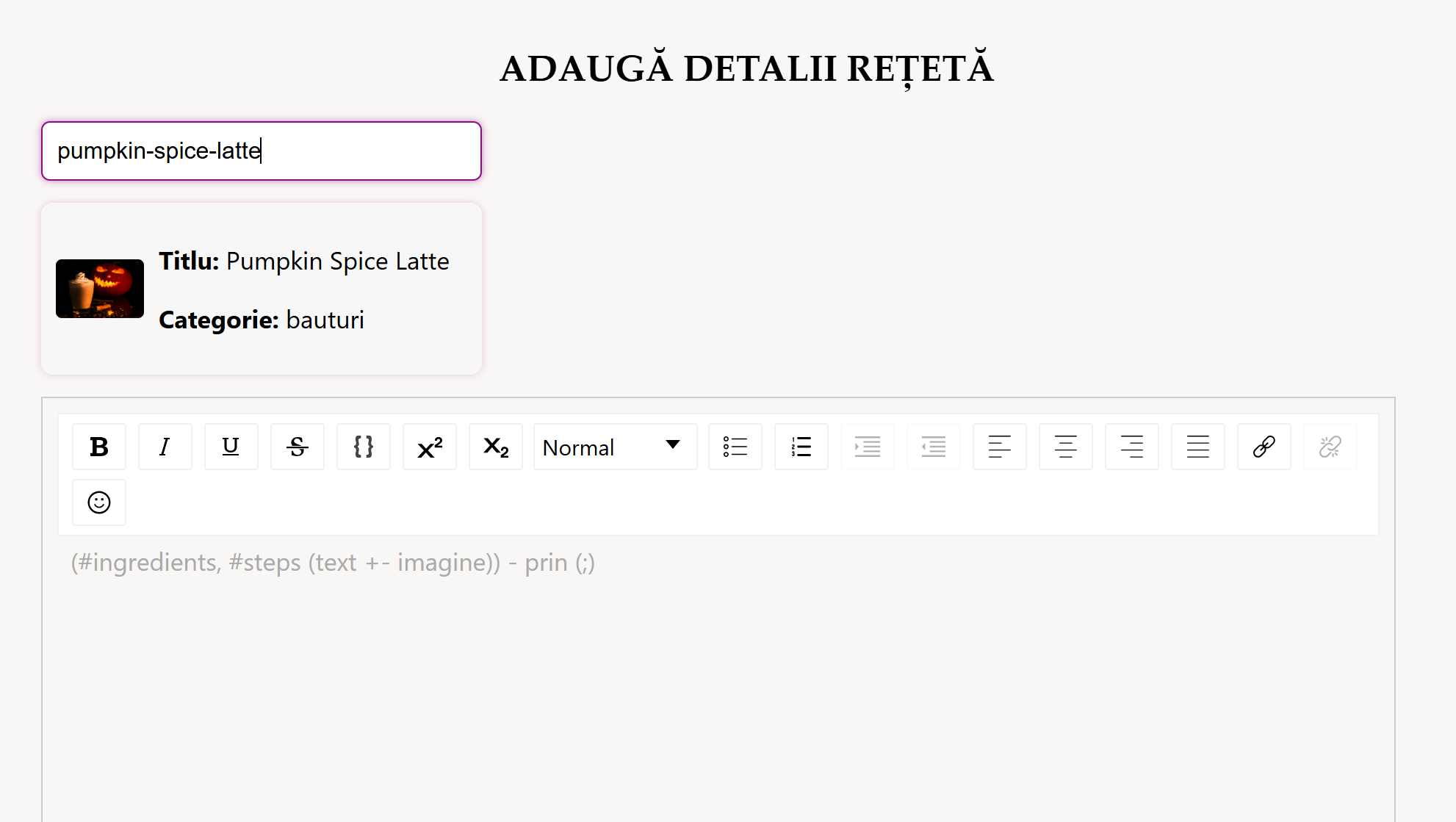


Fig. 4.2.5. Panoul administartiv. Secțiunea **Adaugă detalii rețetă.**

În zona aceasta, administratorul poate adăuga informații detaliate despre reteta. Articolul este găsit cu ajutorul slug-ului rețetei. După ce a fost găsit articolul dorit, administratorul poate începe să scrie detaliile despre acesta, cu ajutorul cuvintelor cheie, pentru a detecta Firebase unde să salveze informațiile acestea. După aceea, mai jos administratorul apasă pe butonul **Salvează detalii**, și aceasta este trimisa în fișierul articolului unde a fost salvat informațiile articolului.

  try {

            await setDoc(doc(db, "recipesDetail", slug), newRecipe);

            setMessage({ text: "Rețeta a fost salvată!", type: "success" });

            setSlug("");

            setCategory("");

            setDescription("");

            setImage("");

            setTitle("");

            setIngredients([]);

            setSteps([]);

            setRecipesDetail(null);

            setEditorState(EditorState.createEmpty());

        } catch (error) {

            console.error("Eroare la salvare:", error);

            setMessage({ text: "Eroare la salvarea rețetei!", type: "error" });

        }

În secțiunea de comentarii, administratorul poate șterge sau răspunde direct din Panoul administrativ la comentarii. Comentariile sunt citite după prenumele, adresa de email, categoria și articolul sub care a fost postat comentariul.

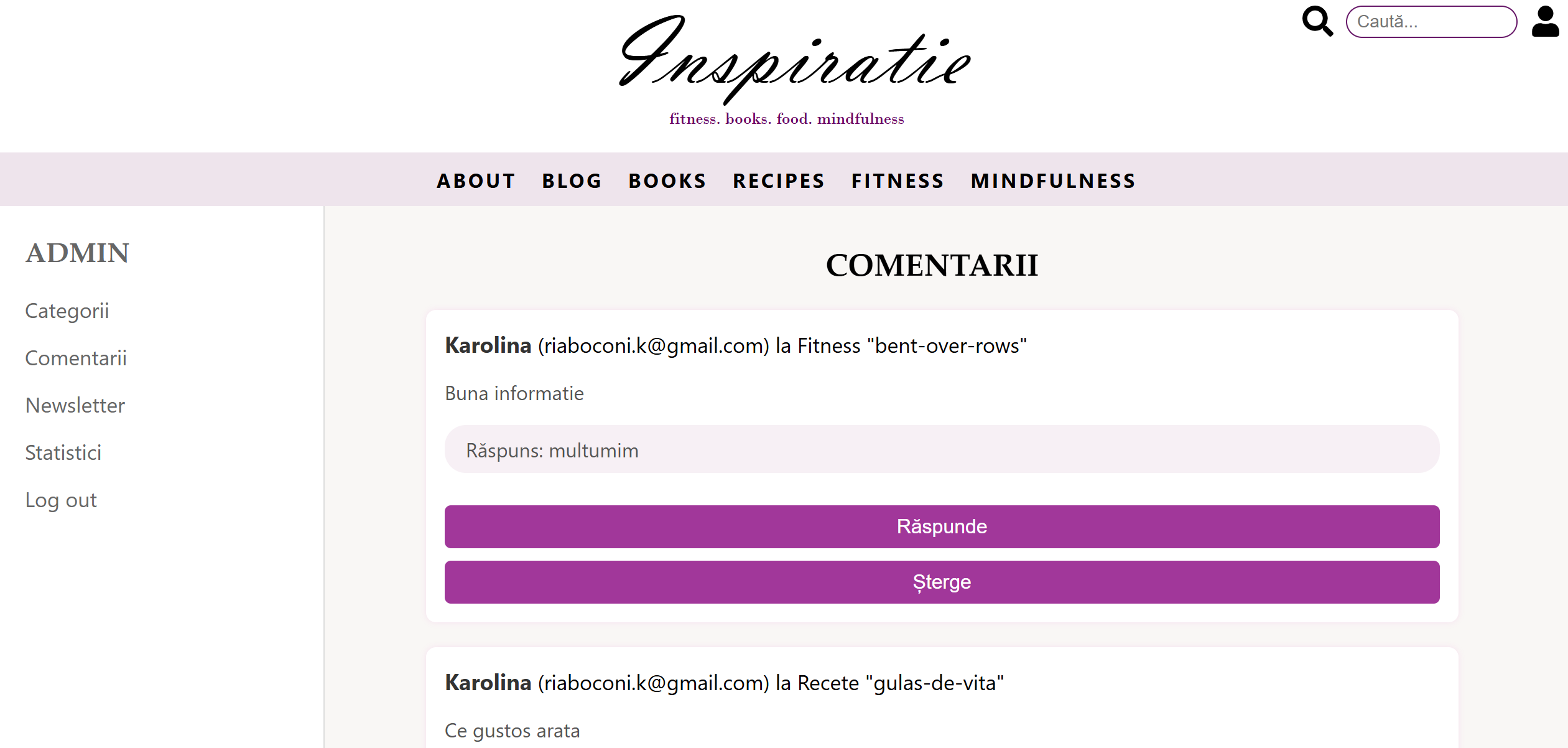


Fig. 4.2.4. Panoul administartiv, lista comentariilor

const handleDelete = async (id) => {

    try {

      await deleteDoc(doc(db, 'comments', id));

      setComments(prevComments => prevComments.filter(comment => comment.id !== id));

    } catch (error) {

      console.error("Eroare la ștergere:", error);

    }

  };

  const handleReply = async (id) => {

    if (!newReply) return;

    const commentRef = doc(db, "comments", id);

    await updateDoc(commentRef, {

      reply: newReply,

    });

    setNewReply("");

    setEditingComment(null);

  };

**Concluzie**

În concluzie, aplicația aceasta este un blog care aduce utilizatorilor informații utile din domeniul lecturii, nutriției, fitnessului și sănătății mentale. Această aplicație este pentru persoanele care își doresc să aibă un stil de viață mai echilibrat, din toate punctele de vedere. Aceasta ușurează accesul la informații mai bine selectate, de asemenea încurajează persoanele care sunt interesate de stilul lor de viață să își ia câteva idei sănătoase în viața de zi cu zi, din acest blog.

Utilizatorii pot citi cărți, pot găsi idei pentru rețete sănătoase și gustoase, aceștia pot găsi inspirații pentru antrenamente mai eficiente, și de a evita greșelile, sau pot descoperi mai multe informații despre lumea spiritului. În plus, aceștia pot lăsa și comentarii sub fiecare articol care li se potrivește. Totul este organizat bine, și structurat în așa fel, ca utilizatorii să aibă acces rapid la orice informație. De asemenea interfața este aranjată bine, într-un stil prielnic. Pentru a avea acces la toate funcțiile, utilizatorul trebuie să își creeze un cont și să fie autentificat. După autentificare aceasta îi oferă posibilitatea de a avea o experiență personalizată.

În timp ce creeam această aplicație, am descoperit multe tehnologii și informații noi, care mi-au fost utile pe parcursul creării, cum ar fi Firebase sau multe elemente noi în React. Această aplicație m-a ajutat să îmi îmbunătățesc cunoștințele și să am posibilitatea de a practica mai bine programarea.

Lucrarea este bine organizată, explicând întreaga aplicație în detalii. Conținutul este organizat bine, începând de la teorii până la aplicația în sine. Este structurată astfel încât sa fie înțeleasă ideea în sine. Conținutul este împărțit în capitole și subcapitole, care au fost alese corect, pentru a explica cât mai clar aplicația, cum ar fi funcționalitățile de bază, interfața utilizatorului, baza de date și importanța administratorului în aplicație. Capitolele sunt părțile importante al aplicației, iar subcapitolele explică în detaliu subiectele acestora.

Pe viitor, aplicația poate fi îmbunătățită cu funcționalități noi cum ar fi dezvoltarea mai complexă a profilului utilizatorului, îmbunătățirea sistemului de comentarii sau îmbunătățirea panoului administrativ, adăugarea articolelor noi despre un stil de viață mai sănătos. De asemenea aspectul vizual al aplicației, să fie mai prielnic, cum ar fi aranjarea aplicației într-un mod mai plăcut și la o mai bună adaptare pentru dispozitive, cum ar fi tableta sau telefonul.

În concluzie, consider că utilizatorii care sunt în căutarea unui stil de viață echilibrat, pot beneficia de această aplicație. În ea se regăsesc informații despre subiecte precum lectura, nutriția, fitnessul și sănătatea mentală. Cred că aplicația are potențial de creștere, oferind o experiență plăcută pentru utilizatori.

**Bibliografie**  
<https://www.w3schools.com/react/default.asp>

<https://firebase.google.com/docs/auth>

<https://reactjs.org/docs/getting-started.html>

<https://legacy.reactjs.org/docs/hooks-state.html>

<https://developer.mozilla.org/en-US/>

<https://total.ong/ghidul-tau-total-despre-yoga-partea-i/>

Jon Duckett. HTML and CSS: Design and Build Websites (2011)

Robin Wieruch. The Road to React: Your journey to master React.js in JavaScript (2023)