Site Web de recenzii pentru cărți

23.07.2024

**Ștefan Amalia Ioana**

**Absolventă**

Cuprins

[1. Introducere (prezentare generală a proiectului) 3](#_Toc380054494)

[2. Prezentarea proiectului (cu evidențierea contribuției personale a candidatului în ansamblul proiectului) 5](#_Toc380054495)

[3. Propuneri de îmbunătățire în cadrul proiectului prezentat 5](#_Toc380054496)

# Introducere (prezentare generală a proiectului)

În era digitală actuală, cărțile continuă să joace un rol esențial în cultivarea cunoștințelor și în dezvoltarea personală. Având în vedere importanța lecturii și dorința de a crea o platformă care să faciliteze accesul și interacțiunea cu literatura, am ales să dezvolt un site web de recenzii pentru cărți. Acest proiect va fi implementat folosind tehnologii moderne și eficiente: Java, Spring Boot, Bootstrap și MySQL Workbench.

Tehnologii Utilizate

* Java - Un limbaj de programare robust și versatil, ideal pentru dezvoltarea aplicațiilor web.
* Spring Boot - Un cadru bazat pe Java care simplifică crearea de aplicații web complexe și scalabile.
* Bootstrap - Un framework de front-end pentru dezvoltarea rapidă și responsive a interfețelor utilizator.
* MySQL Workbench - Un instrument grafic pentru modelarea, dezvoltarea și gestionarea bazelor de date MySQL.
* MySQL Workbench este strâns legat de MySQL, un sistem de gestionare a bazelor de date relaționale (SGBD) open-source, care utilizează SQL (Structured Query Language) pentru gestionarea datelor.

Funcționalitățile Principale ale Proiectului

Proiectul propus va include următoarele funcționalități esențiale:

* Listarea Cărților: utilizatorii vor avea acces la o listă extinsă de cărți, fiecare cu detalii complete, inclusiv titlu, autor, descriere și alte informații relevante.
* Căutare de cărți după numele acesteia sau autor: utilizatorii vor putea căuta cartea dorită după nume sau autor și vor primi ceea ce își doresc, fără a fi nevoiți sa treacă prin toata lista de cărți.
* Lista de Cărți Favorite: platforma va permite utilizatorilor să salveze și să gestioneze o listă personalizată de titluri preferate, facilitând astfel accesul rapid la cărțile de interes.
* Detalii despre Autori: fiecare autor va avea o secțiune dedicată care va include biografii detaliate și bibliografii complete, oferind utilizatorilor o înțelegere mai profundă a contextului și a operei autorului.
* Adăugarea de Cărți Noi: utilizatorii vor putea contribui la extinderea bazei de date prin adăugarea de cărți noi direct din pagina web, asigurând astfel actualizarea și diversificarea continuă a conținutului.
* Secțiune de Comentarii: o funcționalitate de comentarii va permite utilizatorilor să interacționeze între ei, să discute despre cărți și să împărtășească opinii și recomandări.
* Sistem de Log In: platforma va include un sistem de autentificare care va diferenția între utilizatori obișnuiți și administratori, fiecare având acces la diferite funcționalități și nivele de control asupra conținutului și administrării site-ului.
* Sistem de Înregistrare: platforma va include un sistem de înregistrare User care va permite adăugarea unor noi utilizatori care vor primi rolul de user și nu de admin.

# Prezentarea proiectului (cu evidențierea contribuției personale a candidatului în ansamblul proiectului)

**Structura proiectului**

1. Analiza Cerințelor

1.1. Descrierea Utilizatorilor Țintă

1.2. Cerințele Funcționale

2. Design-ul Sistemului

2.1. Arhitectura Generală

2.2. Diagrama UML a Sistemului

2.3. Modelul Bazei de Date

3. Implementarea Proiectului

3.1. Configurarea Mediului de Dezvoltare

3.2. Structura Proiectului în Java și Spring Boot

3.3. Implementarea Interfeței

3.4. Managementul Bazei de Date cu MySQL Workbench

MySQL Workbench va fi utilizat pentru modelarea, dezvoltarea și gestionarea bazei de date, incluzând crearea tabelelor și a relațiilor între ele.

**TheNookBook – Site de recenzii pentru cărți online**

**1. Analiza Cerințelor**

**1.1. Descrierea Utilizatorilor Țintă**

Utilizatorii țintă pentru acest proiect sunt cititorii pasionați și autorii. Cititorii pasionați sunt persoane care citesc frecvent și doresc să împărtășească recenzii și recomandări despre cărțile lor preferate. Autorii, pe de altă parte, sunt scriitori care vor să își promoveze cărțile și să interacționeze direct cu cititorii lor, oferind astfel o platformă pentru feedback și discuții.

**1.2. Cerințele Funcționale**

Listarea cărților (id, titlu, autor, preț) – pagina web, informații afișate din baza de date.

Lista de favorite – utilizatorii pot adăuga carțile lor în lista de preferate/favorite

Detaliile autorilor – o pagină web, unde se găsesc date despre autori

Adăugarea de cărți noi – adăugare carte nou în baza de date din site sub formă de formular

Funcționalitatea de comentarii – conectarea tuturor utilizatorilor pentru a-și putea exprima părerile despre cărțile pe care le-au citit și se gasesc pe site (un fel de club de lectura virtual)

Sistemul de log in (user și admin) – admin: va putea adauga în baza de date noi cărți, va putea modifica datele despre cărți

Sistemul de înregistrare utilizator – user: se poate loga pentru a interacționa cu ceilalți iubitori de cărți

2. Design-ul Sistemului

2.1. Arhitectura Generală

Arhitectura sistemului va fi bazată pe modelul client-server, utilizând un backend în Java cu Spring Boot și un frontend în Bootstrap. MySQL Workbench va fi folosit pentru gestionarea bazei de date.

**2.2. Diagrama UML a Sistemului**

Principalele funcționalități ale aplicației:

FUNCȚII ADMIN:

* Editarea datelor despre carte se face prin apăsarea butonului de lângă butonul Adăugare la favorite
* Ștergerea cărții din baza de date
* Înregistrare carte nouă și afișarea acesteia în pagina Cărți

FUNCȚII USERS:

* Vizualizarea completă a site-ului, a listei de cărți disponibile, a detaliilor despre acestea (unde se afla și secțiunea comentariilor tuturor utilizatorilor pentru fiecare carte în parte și vizualizarea informațiilor despre autori
* Căutarea unei anumite cărți după nume sau după autor

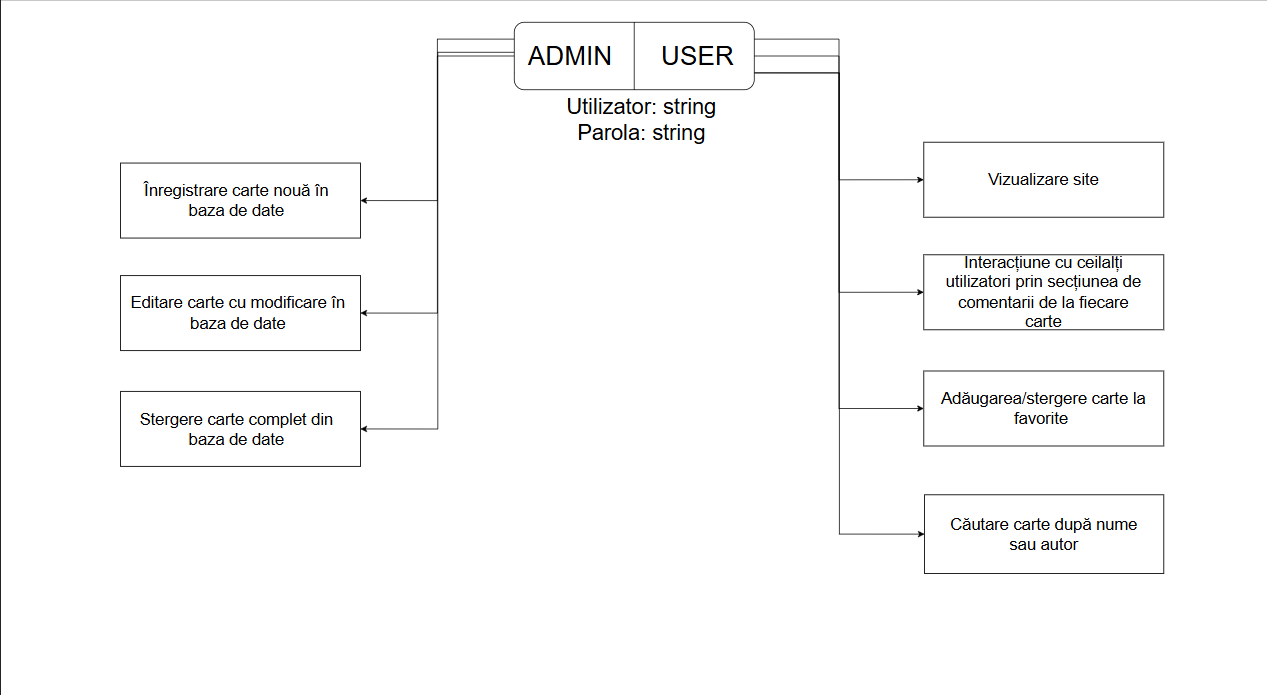


Fig. 1 – Funcțiile aplicației

**2.3. Modelul Bazei de Date**

Baza de date va include următoarele tabele principale:

Users: ID, username, password, email, role

Books: ID, author, name, price, image

Authors: ID, name, bio, nationality, image\_path

My\_Books(Favourites): ID, author, name, price, image

Comments: id, comment\_text, username, book\_id

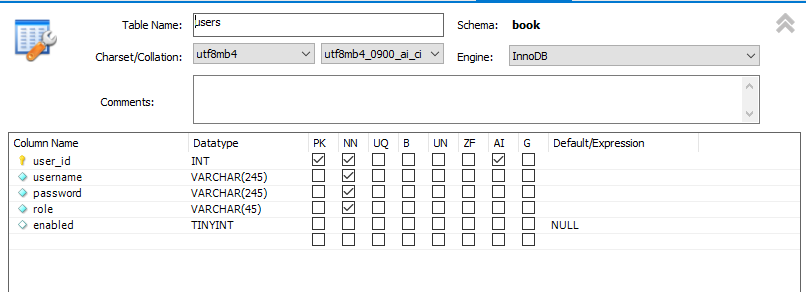


Fig.2 - Tabela users din MySQL WorkBench

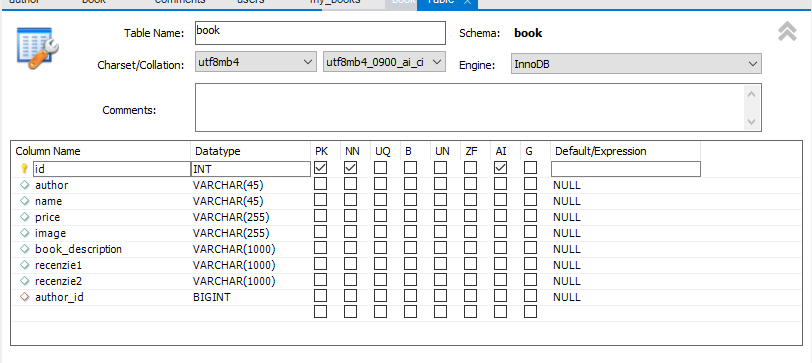


Fig.3 - Tabela book din MySQL WorkBench

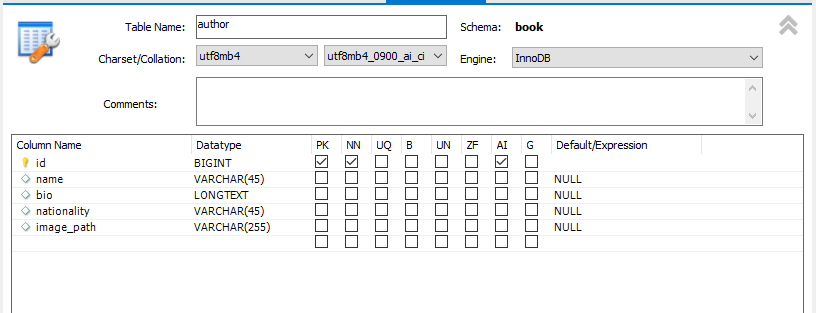


Fig. 4 - Tabela author din MySQL WorkBench

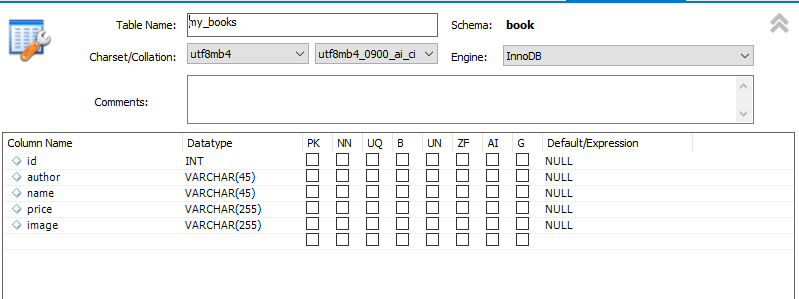


Fig.5 - 5Tabela my\_books (cărți favorite) din MySQL WorkBench

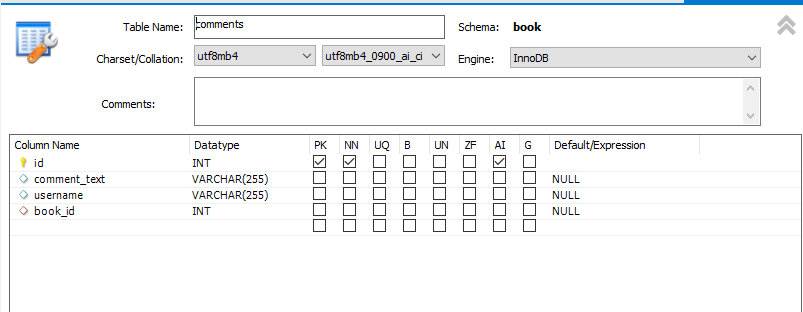


Fig.6 - Tabela comments din MySQL WorkBench

**3. Implementarea Proiectului**

**3.1. Configurarea Mediului de Dezvoltare**

IDE: Eclipse

Biblioteci: Spring Boot, Bootstrap

Baza de date: MySQL Workbench

Crearea proiectului s-a realizat în Spring Initializr, iar ca și dependințe au fost adăugate în pom.xml următoarele:

1. Spring Boot Data JPA - oferă o implementare de tip ORM (Object-Relational Mapping) pentru accesarea și gestionarea datelor din baze de date relaționale, folosind JPA (Java Persistence API) în cadrul aplicațiilor Spring Boot. Persistă datele în stocări SQL folosind Java Persistence API cu Spring Data și Hibernate.
2. Spring Boot Thymeleaf - un motor modern de template-uri Java pentru server-side, utilizabil atât în medii web, cât și în medii standalone. Permite afișarea corectă a HTML-ului în browsere și ca prototipuri statice.
3. Spring Boot Web - construiește aplicații web, inclusiv aplicații RESTful, folosind Spring MVC. Folosește Apache Tomcat ca server integrat implicit.
4. Spring Boot DevTools - oferă un set de instrumente pentru dezvoltare care îmbunătățesc productivitatea, incluzând reîncărcarea automată a aplicațiilor și alte caracteristici care facilitează testarea și dezvoltarea rapidă.
5. MySQL Connecter-J - driver JDBC pentru conectarea aplicațiilor Java la baze de date MySQL, permițând efectuarea de interogări și actualizări ale datelor din baza de date.
6. Spring Boot Security - furnizează funcționalități de securitate pentru aplicațiile Spring Boot, incluzând autentificare, autorizare, protecție CSRF (Cross-Site Request Forgery), configurarea accesului la resurse și gestionarea sesiunilor.
7. Maven Plugin - suplimentează funcționalitățile de build și management ale proiectelor Maven în Spring Boot, facilitând sarcini precum compilarea, testarea, împachetarea și executarea aplicațiilor Spring Boot.

Legare cu baza de date și alte pemisiuni se găsesc în application.properties. Aceste proprietăți configurează o aplicație Spring Boot, detaliind aspecte critice ale funcționării și integrării acesteia cu diverse componente și servicii.

* server.port=8089 definește portul pe care aplicația va asculta cererile HTTP, în acest caz, portul 8089
* spring.datasource.name=book specifică numele sursei de date; spring.datasource.url=jdbc:mysql://localhost:3309/book?ServerTimezone=UTC furnizează URL-ul de conexiune JDBC, incluzând adresa și portul serverului MySQL
* spring.datasource.username=root și spring.datasource.password=root setează acreditările pentru accesul la baza de date
* spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver indică driverul JDBC de utilizat
* spring.jpa.hibernate.ddl-auto=update asigură sincronizarea schemelor bazei de date cu entitățile JPA
* spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect specifică dialectul SQL pentru MySQL 5.7
* Setările spring.main (allow-bean-definition-overriding=true și allow-circular-references=true) permit suprascrierea definițiilor de bean și gestionarea referințelor circulare între beanii Spring

În cadrul unui context Spring, un bean este un obiect care este gestionat de containerul Spring IoC (Inversion of Control).

* Configurațiile pentru Thymeleaf (spring.thymeleaf.prefix=classpath:/templates/, spring.thymeleaf.suffix=.html,spring.thymeleaf.mode=HTML,spring.thymeleaf.cache=false) determină locația și comportamentul șabloanelor HTML
* logging.level.org.springframework.security=DEBUG și logging.level.root=WARN reglează nivelurile de logare pentru componentele de securitate Spring și nivelul de bază al logării, respectiv, facilitând depistarea și diagnosticarea problemelor

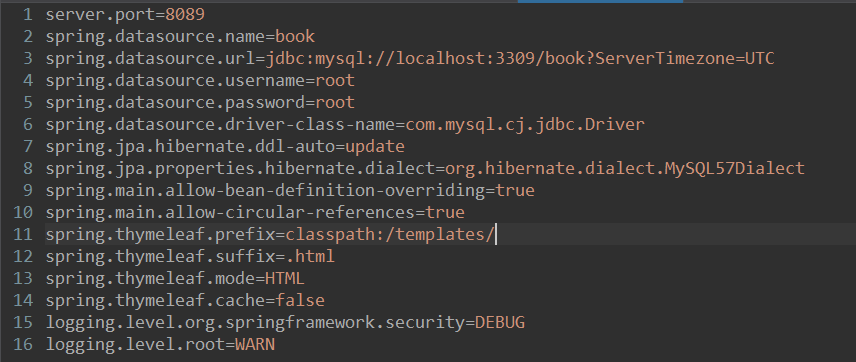


Fig.7 - Application.properties

**3.2. Structura Proiectului în Java și Spring Boot**

**WebSecurityConfig: implementarea securității la nivel de Spring Boot**

@Configuration

@EnableWebSecurity

public class WebSecurityConfig {

@Autowired

private DataSource dataSource;

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Autowired

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.jdbcAuthentication()

.dataSource(dataSource)

.passwordEncoder(passwordEncoder())

.usersByUsernameQuery("select username, password, enabled from users where username=?")

.authoritiesByUsernameQuery("select username, role from users where username=?");

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(authorizeRequests ->

authorizeRequests

.requestMatchers("/images/\*").permitAll()

.requestMatchers("/bookEdit/\*\*", "/book\_register/\*\*", "/deleteBook/\*\*", "/save/\*").hasRole("ADMIN")

.anyRequest().authenticated()

)

.formLogin(formLogin ->

formLogin

.loginPage("/login")

.defaultSuccessUrl("/", true)

.permitAll()

)

.exceptionHandling(exceptionHandling ->

exceptionHandling.accessDeniedPage("/accessDenied")

)

.logout(logout ->

logout

.permitAll()

);

return http.build();

}

}

* Adnotarea *@Bean* este utilizată pentru a indica că metoda passwordEncoder produce un bean care trebuie gestionat de containerul Spring. Acest bean va fi înregistrat în contextul aplicației Spring.
* Metoda passwordEncoder este definită pentru a returna o instanță a clasei BCryptPasswordEncoder. Acest lucru înseamnă că ori de câte ori aplicația are nevoie de un PasswordEncoder, containerul Spring va folosi această metodă pentru a furniza instanța necesară.
* În mod implicit, bean-urile definite astfel sunt singleton-uri, ceea ce înseamnă că Spring va crea o singură instanță a bean-ului și o va reutiliza pe parcursul întregii aplicații. Acest comportament poate fi schimbat prin setarea unei alte valori pentru proprietatea scope a bean-ului, dacă este necesar.
* Adnotarea *@Autowired* facilitează injecția de dependențe, permițând containerului Spring să furnizeze automat instanțele necesare pentru *AuthenticationManagerBuilder* și alte componente necesare pentru configurarea autentificării în aplicație. Acest lucru simplifică gestionarea dependințelor și configurarea aplicației.

**Entity: Clase de entitate (Book, Author, Comments, MyBookList)**

*@Entity*

public class Author {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private Long id;

private String name;

private String bio;

private String nationality;

private String imagePath;

public Author(Long id, String name, String bio, String nationality, String imagePath) {

super();

this.id = id;

this.name = name;

this.bio = bio;

this.nationality = nationality;

this.imagePath = imagePath;

}

//Getters and Setters

}

*@Entity*

public class Comments {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private int id;

private String username;

private String commentText;

*@ManyToOne*

*@JoinColumn*(name = "book\_id")

private Book book;

public Comments() {

}

public Comments(String username, String commentText, Book book) {

this.username = username;

this.commentText = commentText;

this.book = book;

}

//Getters and Setters

}

* *@Entity* - indică faptul că această clasă este o entitate JPA și va fi mapată la o tabelă în baza de date. În acest caz, clasa Author va fi mapată la o tabelă cu același nume (sau un alt nume, dacă este specificat).
* *@Id* - specifică că atributul id este cheia primară a entității. Fiecare entitate trebuie să aibă un identificator unic, iar această adnotare marchează atributul id ca fiind acel identificator.
* *@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***) - indică faptul că valoarea cheii primare va fi generată automat de baza de date folosind o strategie de auto-increment.
* *@ManyToOne* - această adnotare este utilizată pentru a specifica o relație de tip „many-to-one” între două entități, în exemplul de față avem o entitate Review care are relație „many-to-one” cu entitatea Book, fiecare recenzie este asociată cu o carte, iar o carte poate avea mai multe recenzii.
* *@JoinColumn*(name = "book\_id") - această adnotare este folosită pentru a specifica coloana din baza de date care va fi folosită pentru a realiza legătura între cele două entități.

**Repository: Interfețe pentru accesul la date (BookRepository, AuthorRepository, etc.)**

*@Repository*

public interface BookRepository extends JpaRepository<Book,Integer> {

List<Book> findByNameContainingOrAuthorContaining(String name, String author);

}

*@Repository*

public interface CommentRepository extends JpaRepository<Comments, Integer> {

List<Comments> findByBookId(int bookId);

}

* *@Repository -* adnotarea @Repository în Spring este utilizată pentru a indica faptul că o clasă specifică funcționează ca un depozit de date (repository).

- oferă, de asemenea, o capacitate suplimentară de gestionare a excepțiilor. Spring traduce automat excepțiile specifice bazei de date (cum ar fi SQLException) în excepții de tipul DataAccessException, care sunt runtime exceptions. Acest lucru face ca gestionarea excepțiilor să fie mai simplă și mai coerentă în aplicațiile Spring.

**Service: Logică de afaceri (BookService, AuthorService, etc.)**

*@Service*

public class BookService {

*@Autowired*

private BookRepository bRepo;

public void save(Book b) {

bRepo.save(b);

}

public List<Book> getAllBook(){

return bRepo.findAll();

}

public Book getBookById(int id) {

return bRepo.findById(id).get();

}

public void deleteById(int id) {

bRepo.deleteById(id);

}

public List<Book> searchBooks(String query) {

return bRepo.findByNameContainingOrAuthorContaining(query, query);

}

}

*@Service*

public class CommentService {

*@Autowired*

private CommentRepository commentRepository;

public void save(Comments comment) {

commentRepository.save(comment);

}

public List<Comments> getCommentsByBookId(int bookId) {

return commentRepository.findByBookId(bookId); }}

* *@Service* - aceasta este o specializare a adnotării @Component, folosită pentru a indica faptul că o clasă furnizează o anumită funcționalitate (de obicei, logică de business).

**Controller: Gestionarea cererilor HTTP (BookController, AuthorController, etc.)**

*@Controller*

public class BookController {

*@Autowired*

private BookService service;

*@Autowired*

private MyBookListService myBookService;

*@Autowired*

private CommentService commentService;

*@GetMapping*("/")

public String home() {

return "home";

}

*@GetMapping*("/book\_register")

public String bookRegister() {

return "bookRegister";

}

*@GetMapping*("/available\_books")

public ModelAndView getAllBook() {

List<Book> list = service.getAllBook();

return new ModelAndView("bookList", "book", list);

}

*@PostMapping*("/save")

public String addBook(*@ModelAttribute* Book b) {

service.save(b);

return "redirect:/available\_books";

}

*@GetMapping*("/my\_books")

public String getMyBooks(Model model) {

List<MyBookList> list = myBookService.getAllMyBooks();

model.addAttribute("book", list);

return "myBooks";

}

*@RequestMapping*("/mylist/{id}")

public String getMyList(*@PathVariable*("id") int id) {

Book b = service.getBookById(id);

MyBookList mb = new MyBookList(b.getId(), b.getName(), b.getAuthor(), b.getPrice(), b.getImage());

myBookService.saveMyBooks(mb);

return "redirect:/my\_books";

}

*@RequestMapping*("/editBook/{id}")

public String editBook(*@PathVariable*("id") int id, Model model, Authentication authentication) {

if (authentication.getAuthorities().stream().anyMatch(auth -> auth.getAuthority().equals("ROLE\_ADMIN"))) {

Book b = service.getBookById(id);

model.addAttribute("book", b);

return "bookEdit";

} else {

return "accessDenied";

}

}

*@RequestMapping*("/deleteBook/{id}")

public String deleteBook(*@PathVariable*("id") int id, Authentication authentication) {

if (authentication.getAuthorities().stream().anyMatch(auth -> auth.getAuthority().equals("ROLE\_ADMIN"))) {

service.deleteById(id);

return "redirect:/available\_books";

} else {

return "accessDenied";

}

}

*@GetMapping*("/bookDetails")

public String getBookDetails(*@PathVariable*("id") int id, Model model) {

Book book = service.getBookById(id);

List<Comments> comments = commentService.getCommentsByBookId(id);

model.addAttribute("book", book);

model.addAttribute("comments", comments);

return "bookDetails";

}

*@PostMapping*("/addComment")

public String addComment(*@RequestParam*("bookId") int bookId,

*@RequestParam*("username") String username,

*@RequestParam*("comment") String commentText) {

Book book = service.getBookById(bookId);

Comments comment = new Comments(username, commentText, book);

commentService.save(comment);

return "redirect:/bookDetails?id=" + bookId;

}

*@GetMapping*("/search\_books")

public ModelAndView searchBooks(*@RequestParam*(value = "query", required = false) String query) {

List<Book> books;

if (query != null && !query.trim().isEmpty()) {

books = service.searchBooks(query);

} else {

books = service.getAllBook();

}

return new ModelAndView("bookList", "book", books);

}

}

*@Controller*

public class DetailsController {

*@Autowired*

private BookService service;

*@GetMapping* ("/bookDetails/{id}")

public String getBookDetails(*@PathVariable*("id") int id, Model model) {

Book book = service.getBookById(id);

if (book != null) {

model.addAttribute("book", book);

return "bookDetails";

} else {

return "error";

}

}

}

* *@Controller -* indică faptul că această clasă este un controller care gestionează cereri HTTP și returnează modele și vizualizări.
* *@GetMapping()* - definirea rutei și gestionarea cererilor GET la acea rută.
* *@PostMapping()* - definirea rutei și gestionarea cererilor POST la acea rută.
* *@RequestMapping("/mylist/{id}")* - definirea rutei de bază și gestionarea cererilor la acea rută, inclusiv parametrul de cale {id}.

**3.3. Implementarea Interfeței**

Interfața utilizator va fi dezvoltată folosind Bootstrap pentru a asigura un design responsive și modern, dar și a altor elemente de HTML, CSS, JavaScript pentru a aduce elemente de personalizare site-ului.

Prima pagină care apare este cea de logare. Aceasta are proprietatea de a separa utilizatorii în două categorii, prima fiind de admin, iar a doua de user.

Persoana catalogată ca și admin, are dreptul la tot ce înseamnă editare carte, adăugare carte, ștergere din baza de date a cărților.

Această parte de securitate a fost realizată în parte de configurare sistem (backend) folosind SpringSecurity cu ajutorul adnotărilor *@Configuration* și *@EnableWebSecurity*, se definesc mai multe componente esențiale pentru securitate. Așadar, doar persoanele care se găsesc în baza de date au accesul la pagina TheNookBook.

Există în această parte și alte metode care intensifică securitatea site-ului:

* Metoda *passwordEncoder()* definește utilizarea algoritmului BCrypt pentru criptarea parolelor.
* Metoda *configureGlobal()* setează autentificarea utilizatorilor prin interogări SQL pentru a obține informații despre utilizatori și rolurile acestora din baza de date.
* Metoda *securityFilterChain()* configurează reguli pentru accesul la diferite resurse ale aplicației: permite accesul liber la resursele din /images/\*, restricționează anumite URL-uri pentru utilizatorii cu rolul de "ADMIN" și asigură că toate celelalte cereri necesită autentificare.

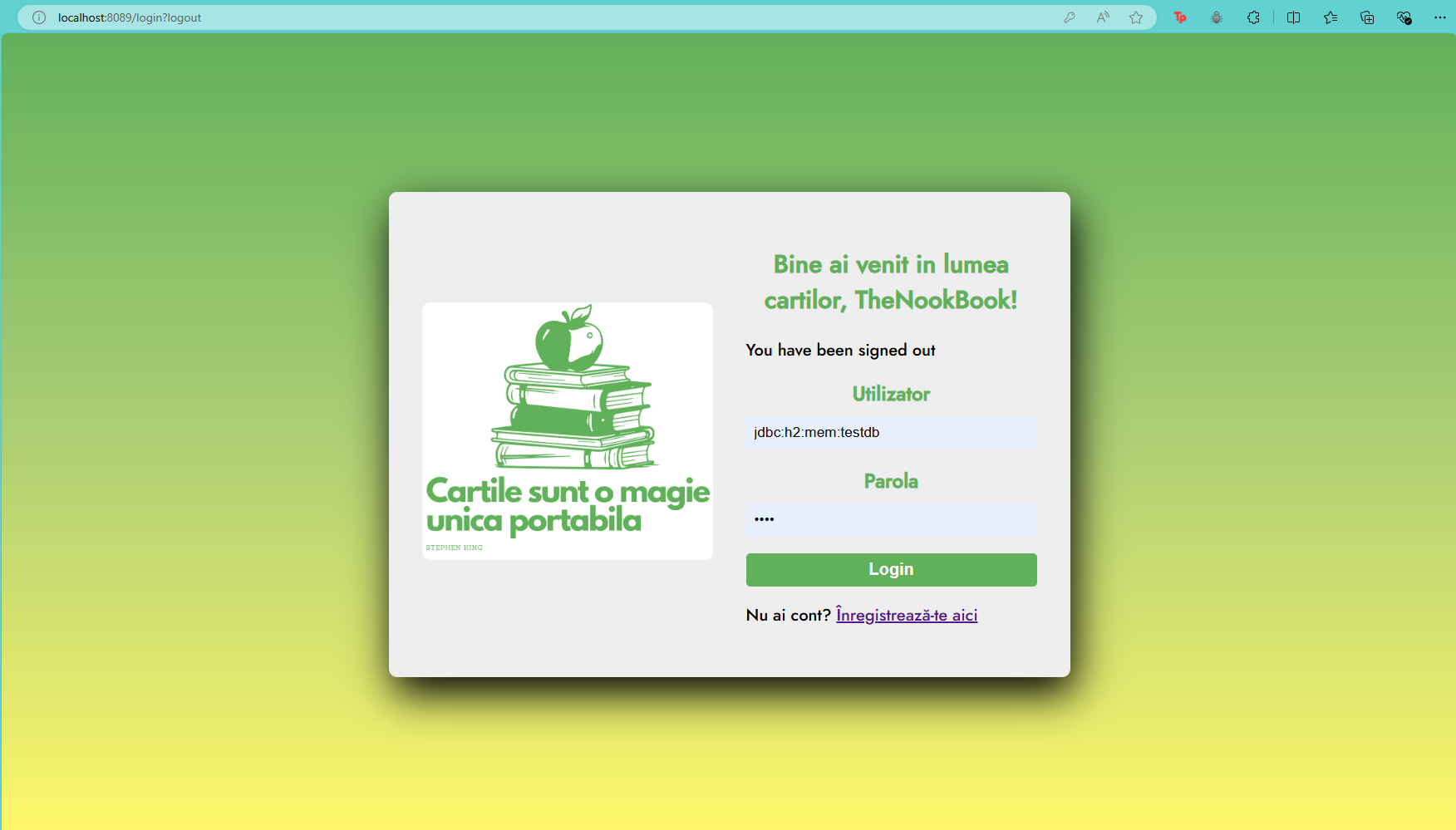


Fig. 8 – LogIn TheNookBook

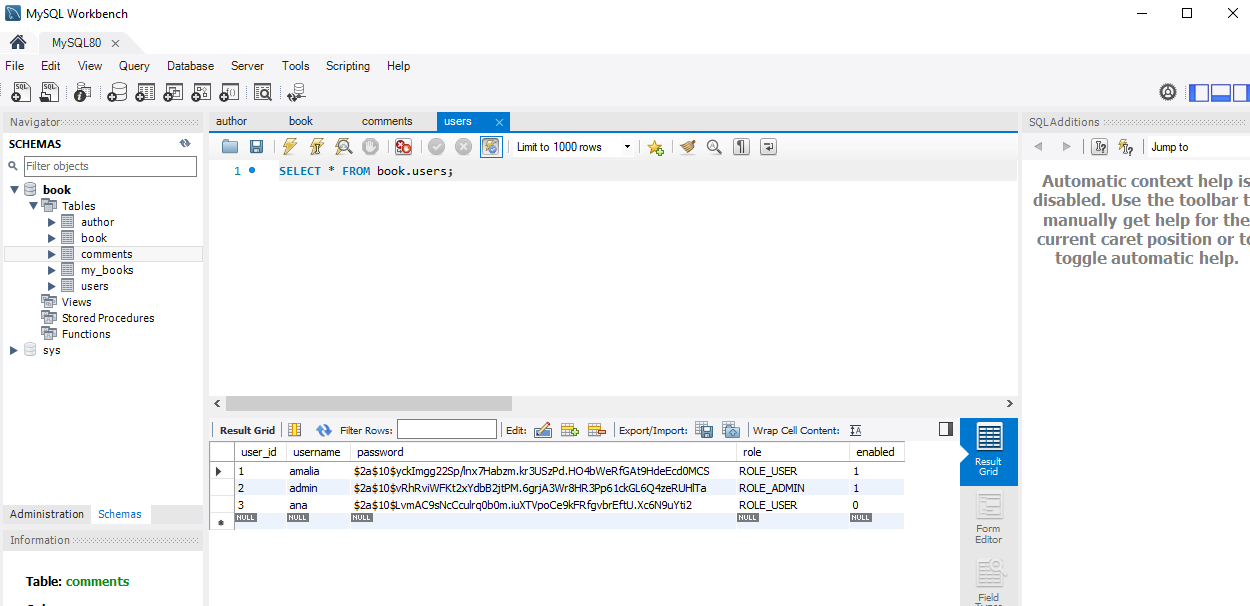


Fig.9 – Baza de date a persoanelor care se pot autentifica

Pentru a adăuga un nou utilizator cu rol de utilizator standard, se accesează pagina de Înregistrare, unde se completează câmpurile pentru nume și parolă. Automat, se atribuie rolul de utilizator standard pentru a menține ordinea și securitatea site-ului, prevenind astfel accesul neautorizat la funcțiile de editare, adăugare sau eliminare a cărților. Rolul utilizatorului standard este de a vizualiza cărțile disponibile și de a interacționa cu ceilalți utilizatori prin intermediul secțiunii de comentarii, care se găsește în partea de Detalii\_carte. Parola aleasă de client, este criptată pentru a corespunde cu cele din baza de date, dar și pentru a oferi un nivel de siguranță pentru fiecare utilizator.

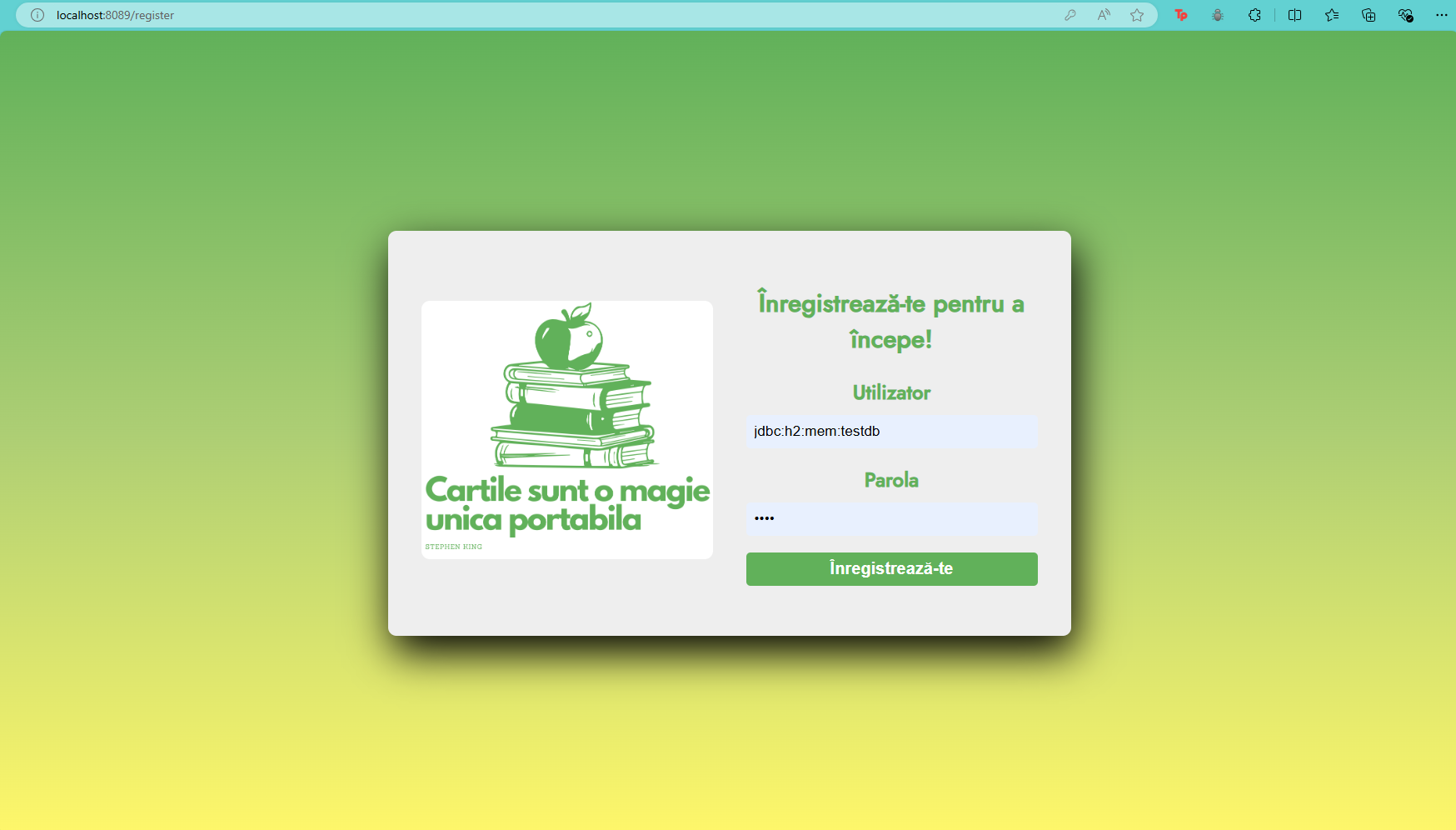
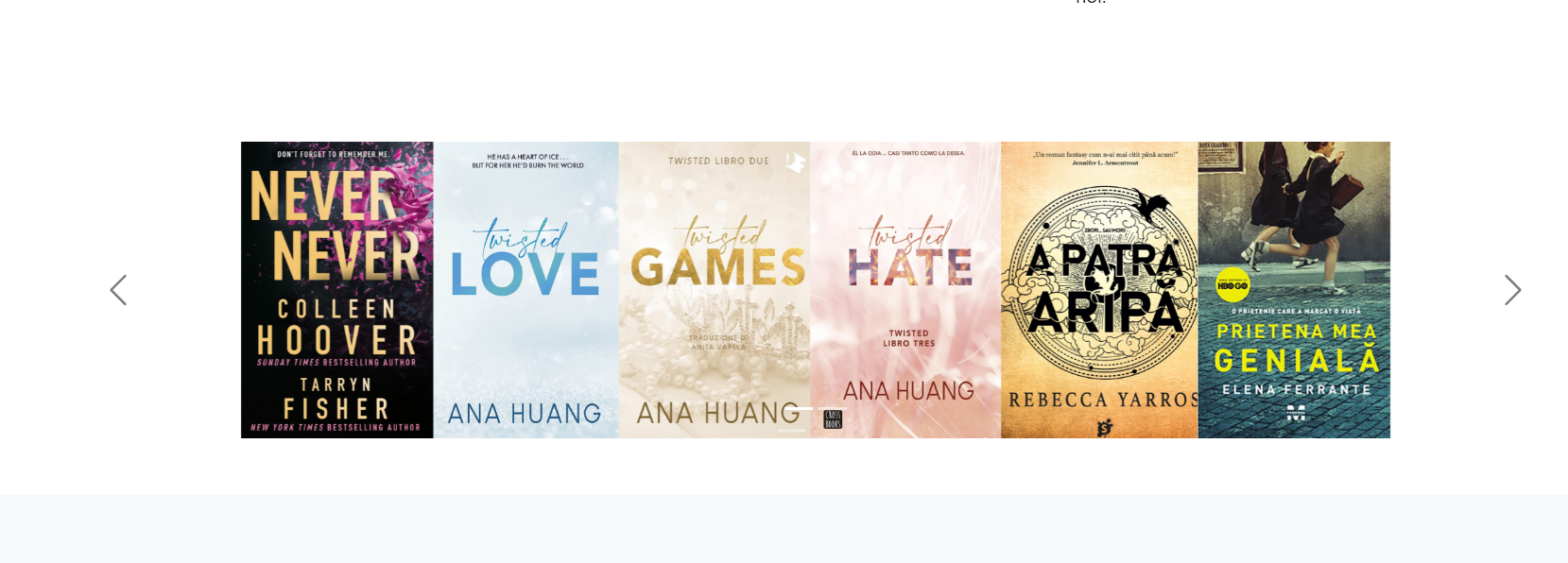
O altă funcție adăugată la acest nivel este cea de utilizator deja existent în baza de date. În cazul în care o altă persoana cu un username identic cu unul deja existent în baza de date, acesta este imediat anunțat de această problemă și astfel se poate gândi la altul. Acest lucru este vizibil prin faptul că după introducerea numelui și parolei, pagina nu se schimbă cu cea de login, astfel rămânând tot cea de înregistrare. 

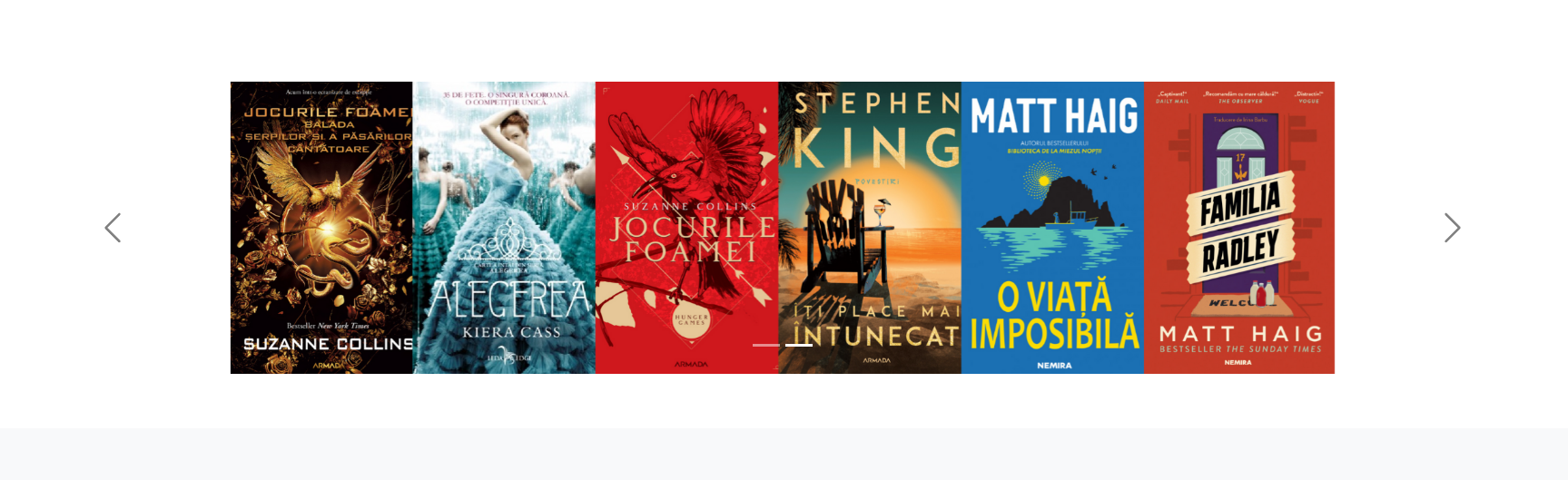
Fig. 10 – Înregistrarea noilor utilizatori în platforma de recenzii TheNookBook

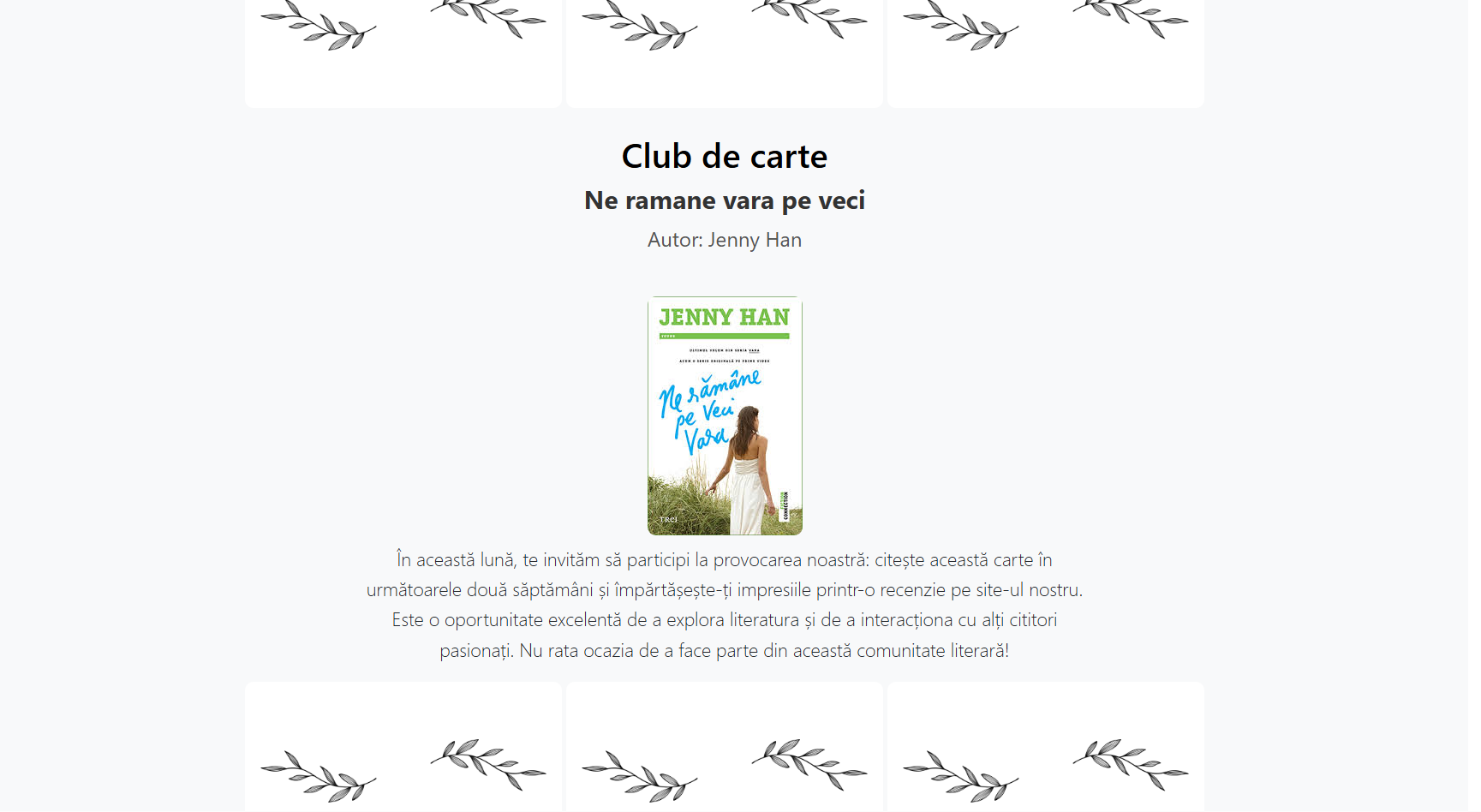
După autentificare, mergem mai departe către pagina de home, unde se găsesc toate informațiile despre tot conceptul TheNookBook. Acesta are la bază, adunarea tuturor pasionaților de cărți la un loc exact ca într-o comunitate. Astfel, pe pagina de start se găsește misiunea site-ului, un carusel cu imagini, secțiunea de club de carte, o campanie tematică de lectură, care se schimbă din două în două săptămâni, campanie care la final așteaptă părerea de la cititor, câteva citate motivaționale despre cărți și alte imagin trimise de utilizatori, care îi surprind pe acesția înconjurați de elementul care le aduce liniște: cartea! În bara de navigare sunt puse toate paginile proiectului: Pagina de start, lista Cărților, lista Favoritelor, Profilul autorilor, Înregistrare carte nouă – valabilă doar pentru persoanele admin, iar în final regăsim butonul de logout.













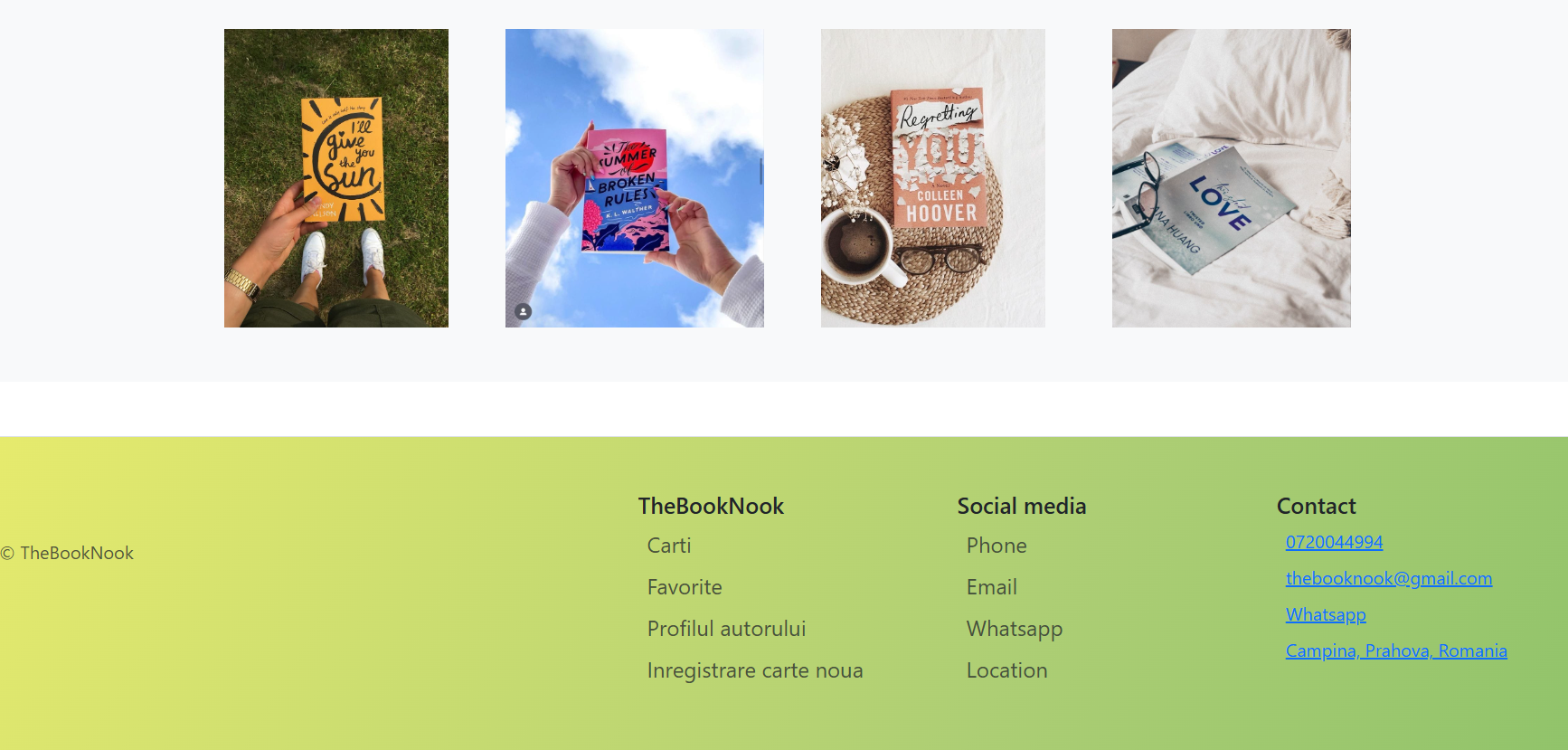


Fig.11 – Pagina de start, cu titlu, bara de navigare și restul elementelor cum ar fi carusel de imagini, logo site, club carte, imagini, citate și alte informații despre site

În pagina de Cărți există funcția de căutarea carte după numele unei cărți, după numele autorului, sau amândouă. Aceasta funcție de căutare este evidențiată înainte de listarea tuturor cărților din baza de date. Se afișează prima carte din baza de date care este găsită și care îndeplinește cerințele.

Apoi se pot vedea cărțile pe care doar cei cu rolul de admin le pot adăuga, sterge, edita.

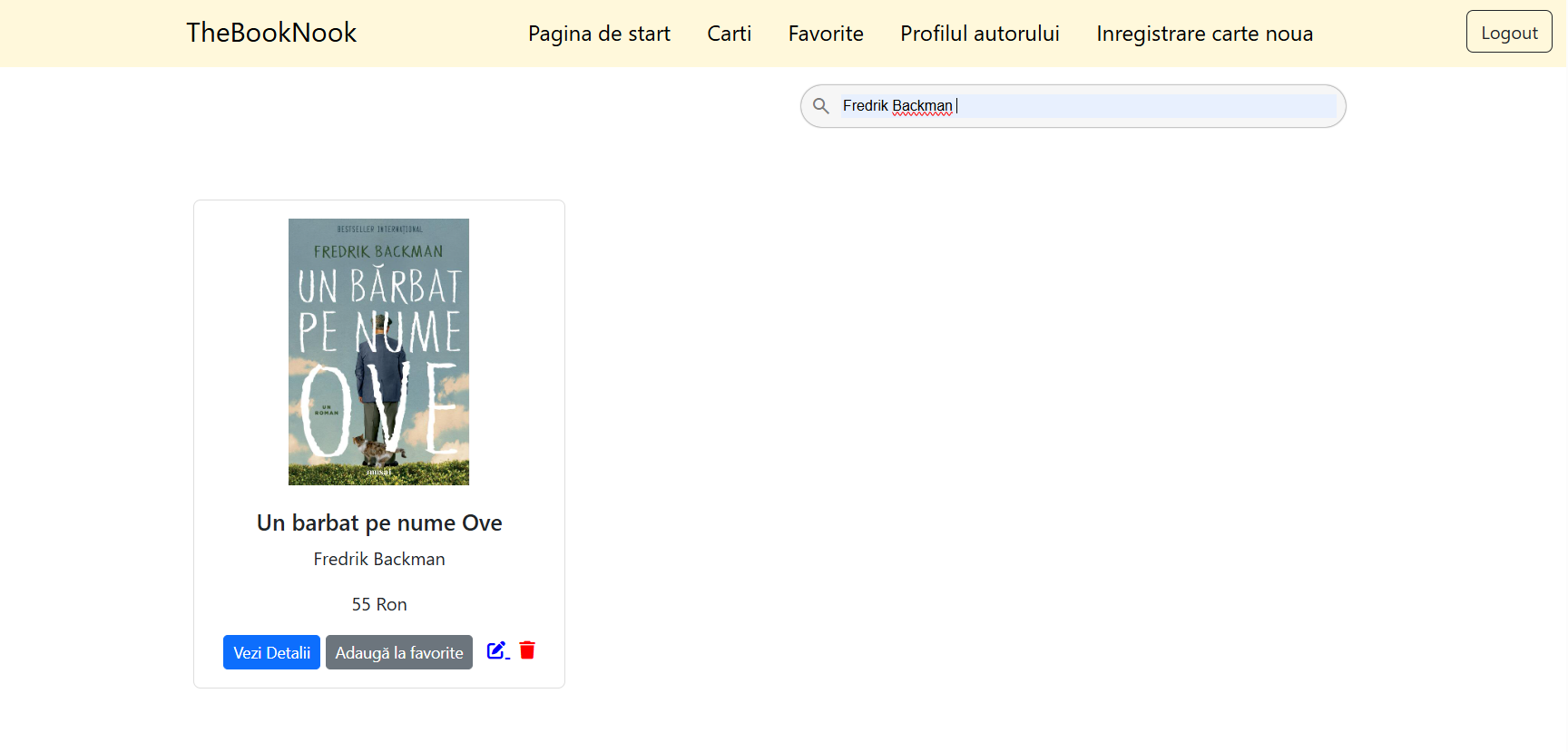


Fig. 12 – Căutarea unei cărți

În bara de navigare avem secțiunea de Cărți, unde sunt listate toate cărțile stocate în baza de date. Fiecare carte este încadrată, prezinta imaginea, titlul și numele autorului, prețul fiecăreia conform pieții (fără promoțiile paginilor care le vând) și funcțiile pe care le pot utiliza persoanele admine ale site-ului TheNookBook, dar și cele de user. Se poate vizualiza detaliile cărții respective, se pot adăuga la lista de favorite pentru persoanele de tip user, iar pentru cei denumiți admin, pe lângă cele enumerate, mai pot edita carte în cazul unei schimbări de preț sau actualizare coperta (în cazul unei relansăre de carte dupa film) prin butonul de edit sau stergerea acesteia din baza de date (în cazul unei greșeli la adăugare). Tot aici, persoana declarată administrator are dreptul la Înregistrare carte nouă (element regăsit în bara de navigare) care permite adăugarea unei noi cărți în baza de date.

Secțiunea de comentarii unde cititorii pot lasa recenzii se găsește în Detalii carte, pentru fiecare carte în parte, pentru a se putea adăuga părerile personale pentru o carte citită. Astfel, când detaliile unei cărți sunt deschise, utilizatorii pot lăsa părerea lor despre carte, în cazul în care au citit-o, sau le pot vizualiza pe cele deja existente, puse de alți utilizatori, ajutându-i pe iubitorii de cărți să își aleagă următoarea carte de citit.

Site-ul are chiar și o secțiune de cartea lunii, unde persoanele denumite admin, schimbă titlul cărții și mențin utilizatorii într-o oarecare provocare, aducând cu gândul la un club de carte, doar că în varianta virtuală. După terminare, sunt îndrumați să lase un comentariu pentru a atrage cât mai mulți oameni în comunitate și pentru a unii iubitorii de cărți din toată țara.

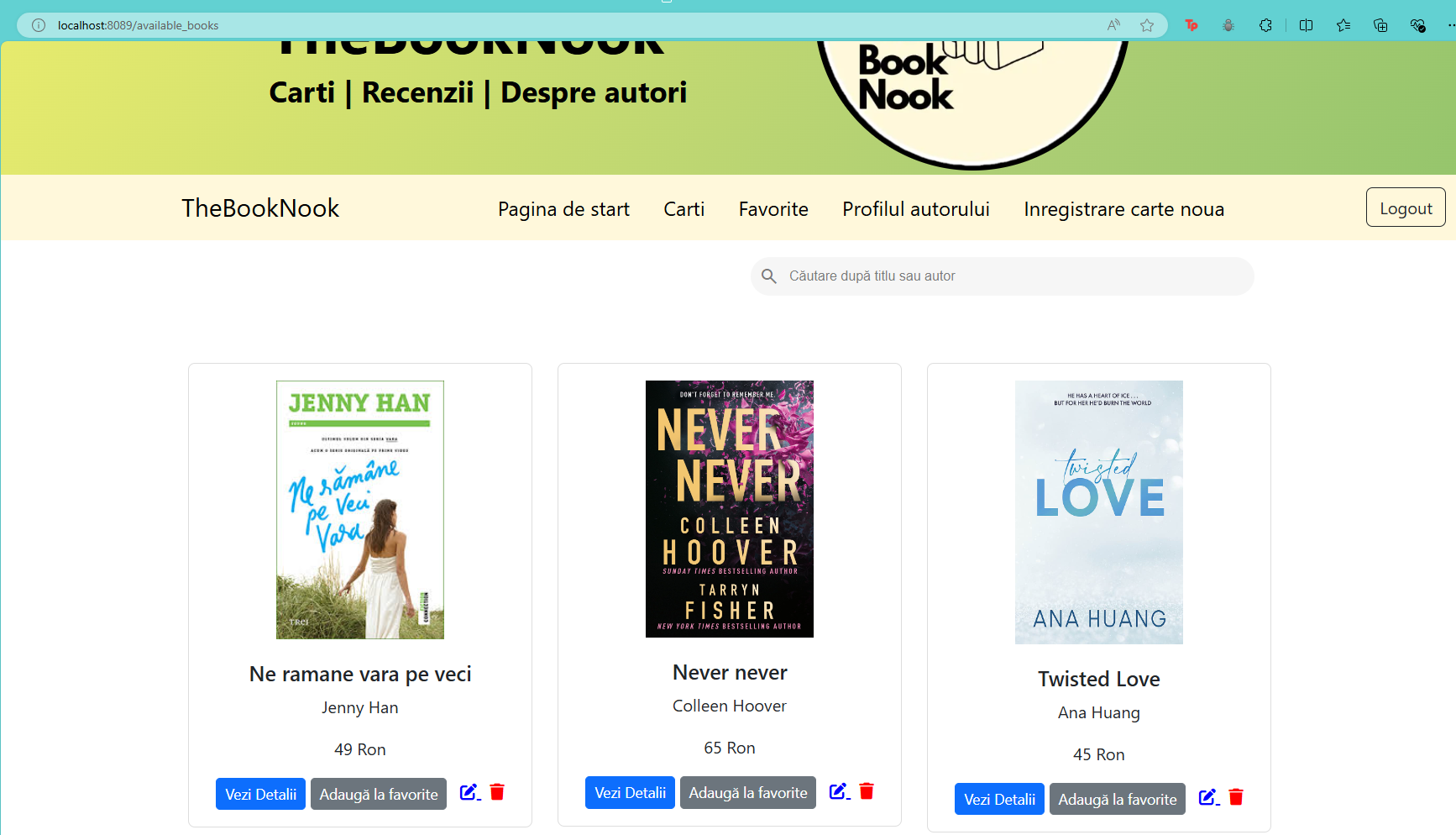


Fig.13 – Pagina de cărți disponibile

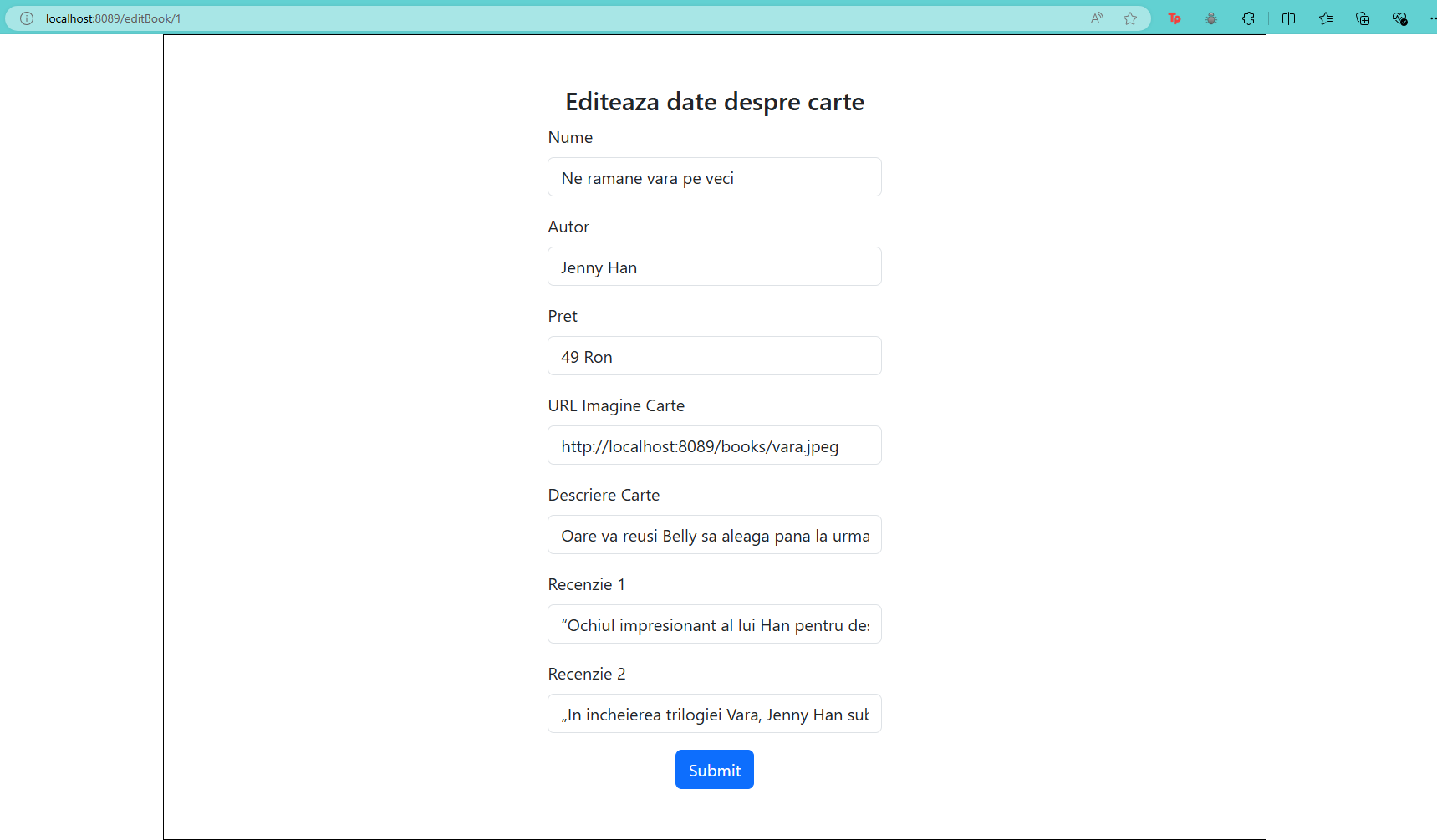


Fig.14 – Editare carte, doar pentru ADMIN

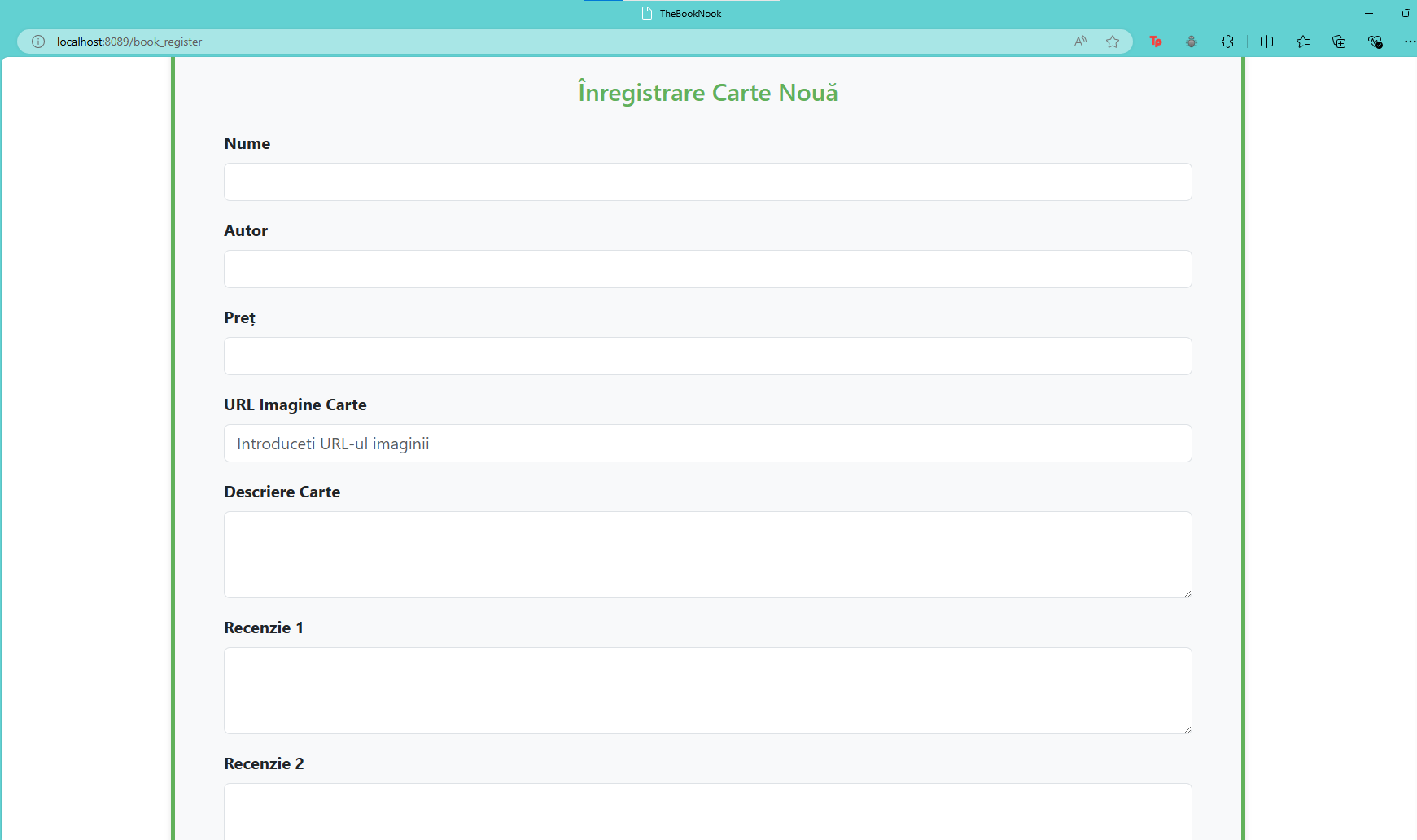


Fig.15 – Înregistrare carte nouă, doar pentru ADMIN

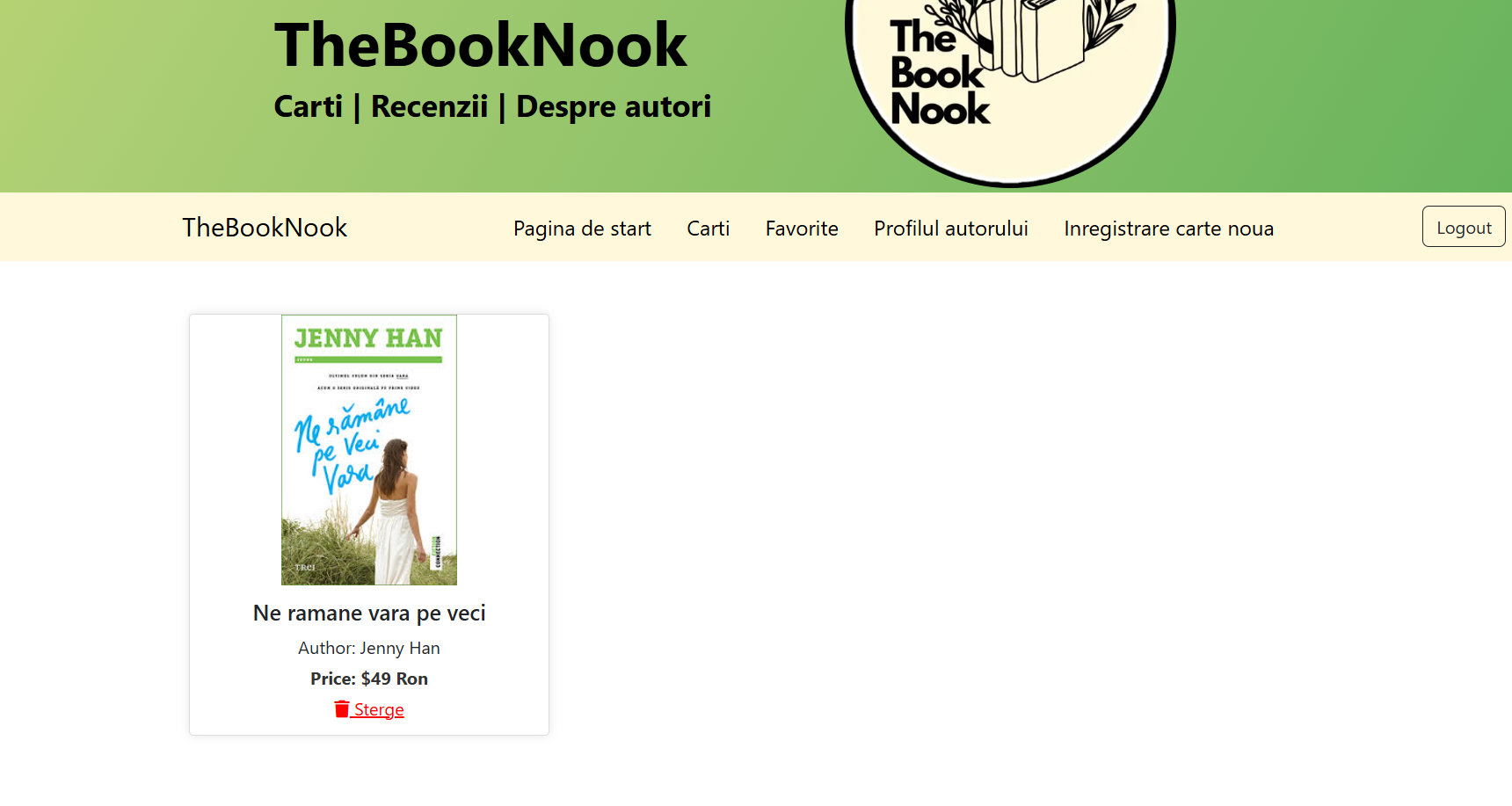
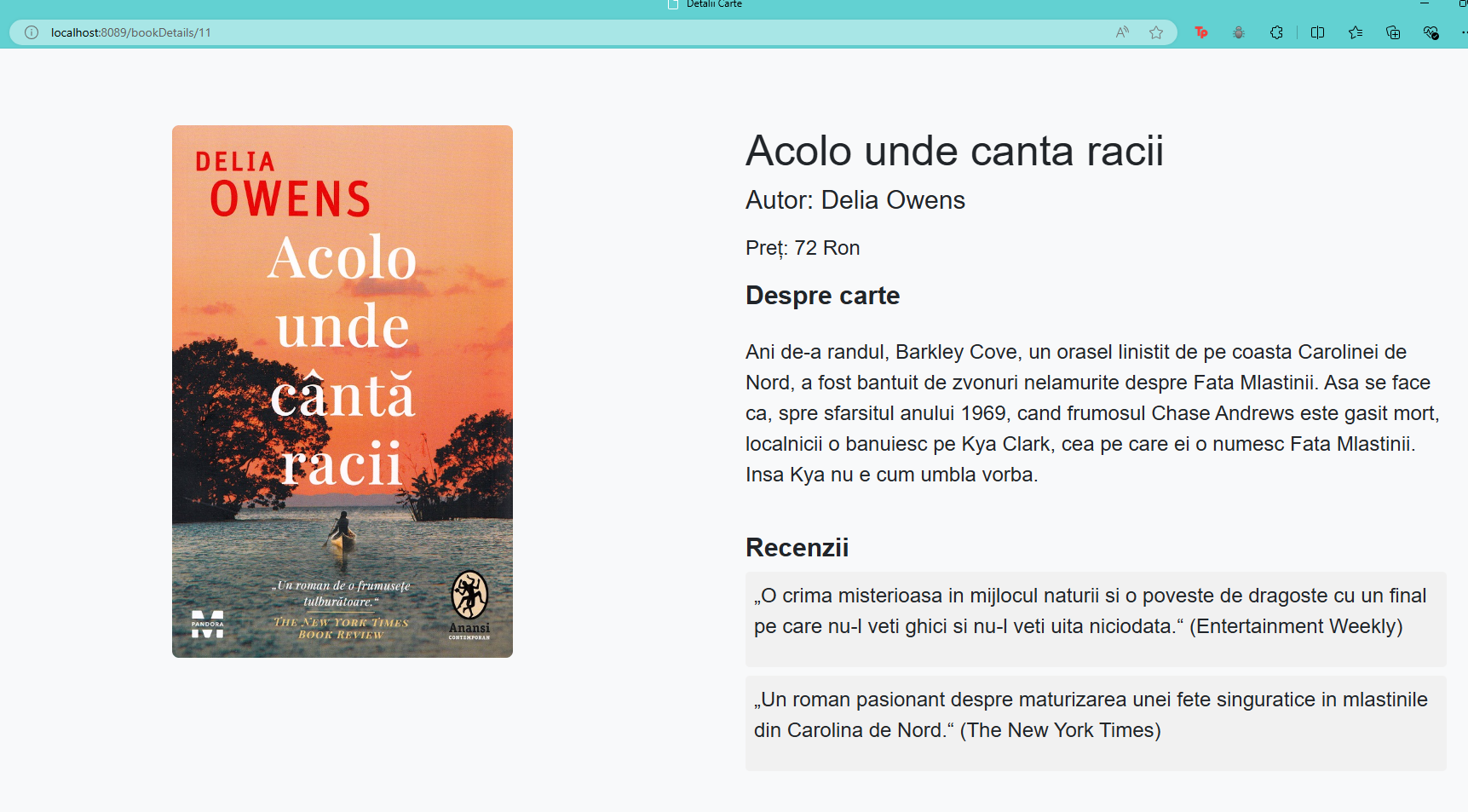


Fig.16 – Lista cărților favorite ale utilizatorilor



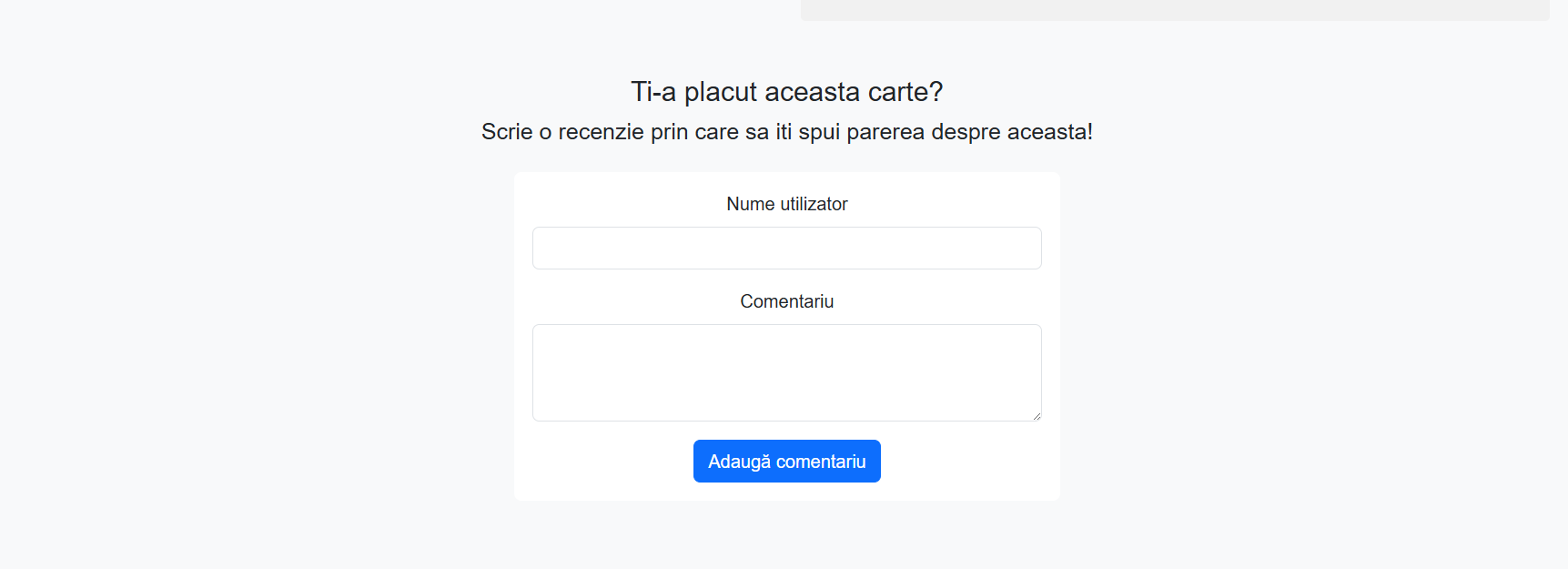


Fig.17 – Pagina detalii carte, secțiunea de comentarii, unde utilizatorii îsi spun părerile despre carte

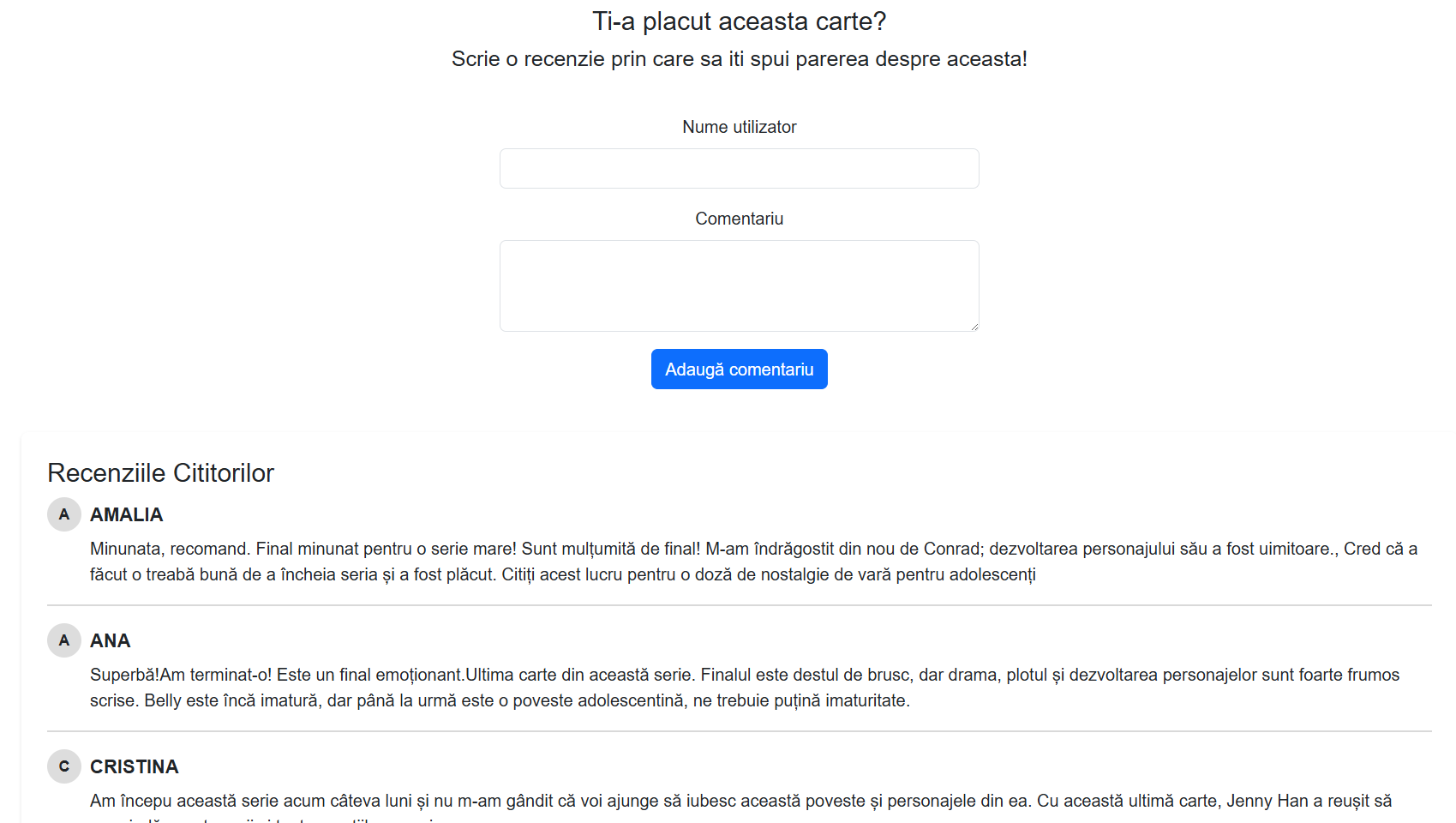


Fig. 18 - Afișarea comentariilor de la utilizatori



Fig. 18 - Pagina autorilor unde sunt listați împreună cu câteva informații despre fiecare

# Propuneri de îmbunătățire în cadrul proiectului prezentat

Aplicația de recenzii cărți online, are la bază un concept simplu, de a aduna toți iubitorii de lectură și de a putea aduce carților cât mai multe recezii de la cititori pentru a îi ajuta pe cei care vor să înceapă o nouă pasiune relaxantă să își aleagă cărțile conform stilului și preferințelor acestora.

Astfel, propunerile de îmbunătățire din cadrul acestui proiect sunt:

* **Interfață mai prietenoasă:**
* Adăugarea de teme personalizabile pentru utilizatori, rearanjarea informațiilor despre autori și legarea acestora de cărțile disponibile din lista de cărți;
* **Funcționalități avansate de căutare și filtrare**:
* Implementarea unei funcții de căutare avansată care să permită filtrarea după mai multe criterii, cum ar fi genul (ănclusiv adăugarea datelor aferente pentru fiecare carte), autorul;
* Adăugarea unor filtre pentru a sorta cărțile după popularitate, rating, sau data adăugării;
* **Moderare și control al conținutului**:
* Implementarea unui sistem de moderare pentru comentarii și recenzii, care să prevină spam-ul și să asigure un mediu de discuție civilizat;
* **Funcționalități de colaborare**:
* Crearea unor grupuri de discuții tematice unde utilizatorii pot colabora și discuta pe subiecte specifice;
* Introducerea unor cluburi de carte virtuale unde utilizatorii pot organiza și participa la întâlniri de lectură;

.

# 