

Examen la Proiectarea aplicatiilor web



Tema: Library Management

Student: Draghici G.M. Andreea-Maria

Grupa: CR 3.1 B

Sectia: Calculatoare Romana

17 Iunie 2022

1. Descrieti rolul si obiectivele aplicatiei. Specificati in detaliu cerintele aplicatiei.

Library Management este o aplicatie folosita pentru gestionarea activitatii unei biblioteci / librarii in format electronic. Aceasta aplicatie este usor de utilizat, sigura si eficienta.

Acest sistem este format dintr-o singura parte, o aplicatie web, care va fi utilizata de bibliotecar, dar si de catre cititori. Aplicatia comunica constant cu baza sa de date.

Astfel spus, aplicatia permite doua tipuri de utilizatori:

- 1. Utilizatorii care au rolul de user si pot vizualiza lista de carti disponibile in librerie de unde user-ul isi poate alege ce carte sa imprumute, pot sa trimita un ticket catre asistenta pentru utilizatori cu privire la orice eroare/ problema intalnita in timpul aplicatiei, pot avea acces la lista de carti care au fost accesate de catre alti utilizatori, astfel incat noii utilizatori sa poata vedea care sunt cele mai recente carti si sa ii ajute in alegerea unei carti potrivite.**
- 2. Utilizatorii care au rolul de administrator si vor avea aceleasi functionalitati ca si utilizatori normali, dar pe langa asta vor mai avea si altele in plus:**
 - Administratorul poate adauga, crea, sterge sau edita carti din lista de carti sau date despre acestea.**
 - Administratorul poate vizualiza lista de carti rezervate.**
 - Administratorul poate modifica informatia starii unei carti anume din baza de date. (Ex: Daca cartea nu se mai afla pe stoc in biblioteca).**
 - Administratorul poate actualiza informatii despre politica de confidentialitatea sau despre cartile rezervate.**

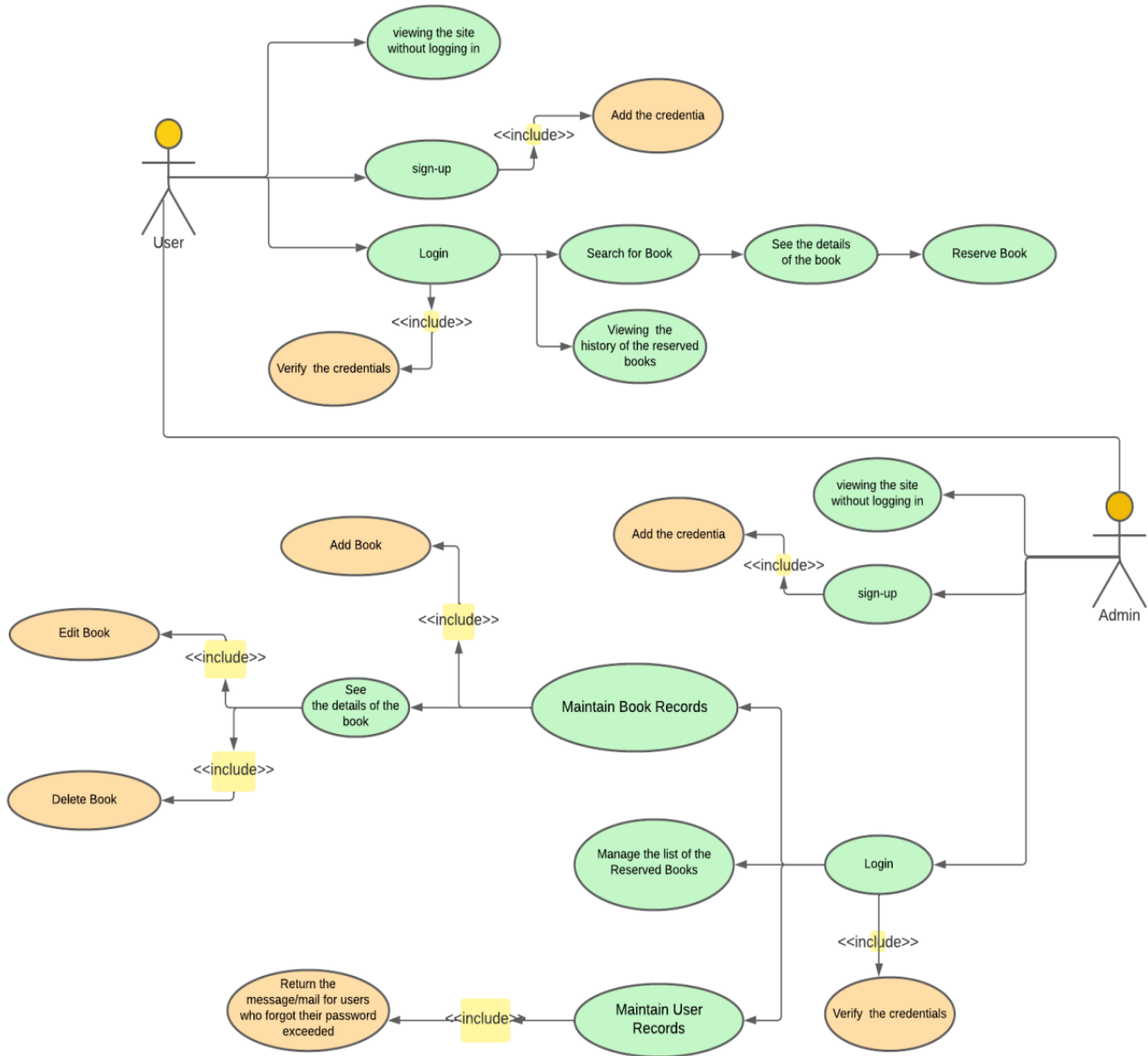
Functionalitatile aplicatiei sunt urmatoarele:

- In primul rand, aplicatia are un formular de inregistrare, unde user-ul isi introduce datele pentru a-si crea contul, iar utilizatorii care isi fac cont vor primi automat rolul de user. Acestia isi pot seta email-ul, parola, etc.**

- Aplicatia dispune si de o pagina de login, unde utilizatori isi vor introduce email-ul si parola, iar daca sunt corecte, ei vor fi redirectionati catre pagina principala de Home.
- Aplicatia are o pagina de contact.
- Aplicatia are o pagina de About care ofera informatii despre site si activitatea acestuia.
- Un utilizator (indiferent de rol) poate accesa functionalitatea aplicatiei, numai dupa ce s-a logat.
- Aplicatia dispune si de o pagina numita Books, unde se gaseste lista cu carti disponibile in librerie de unde user-ul isi poate alege ce carte sa imprumute.
- Aplicatia contine si o pagina de Privacy Policy, care include confidentialitatea si politica aplicatiei raportata reglementarilor si datelor stocate in baza de date.
- Aplicatia mai contine si o pagina numita Library, unde utilizatorul poate vizualiza lista cu numarul de carti rezervate, cat si numarul total de carti din librerie .
- Aplicatia contine si pagina principala de Home, aceasta oferind posibilitatea utilizatorului de a naviga catre lista de carti din librerie, aceasta pagina mai ofera si informatii si fotografii despre cartile care au fost accesate de catre alti utilizatori, astfel incat noii utilizatori sa poata vedea care sunt cele mai recente carti.

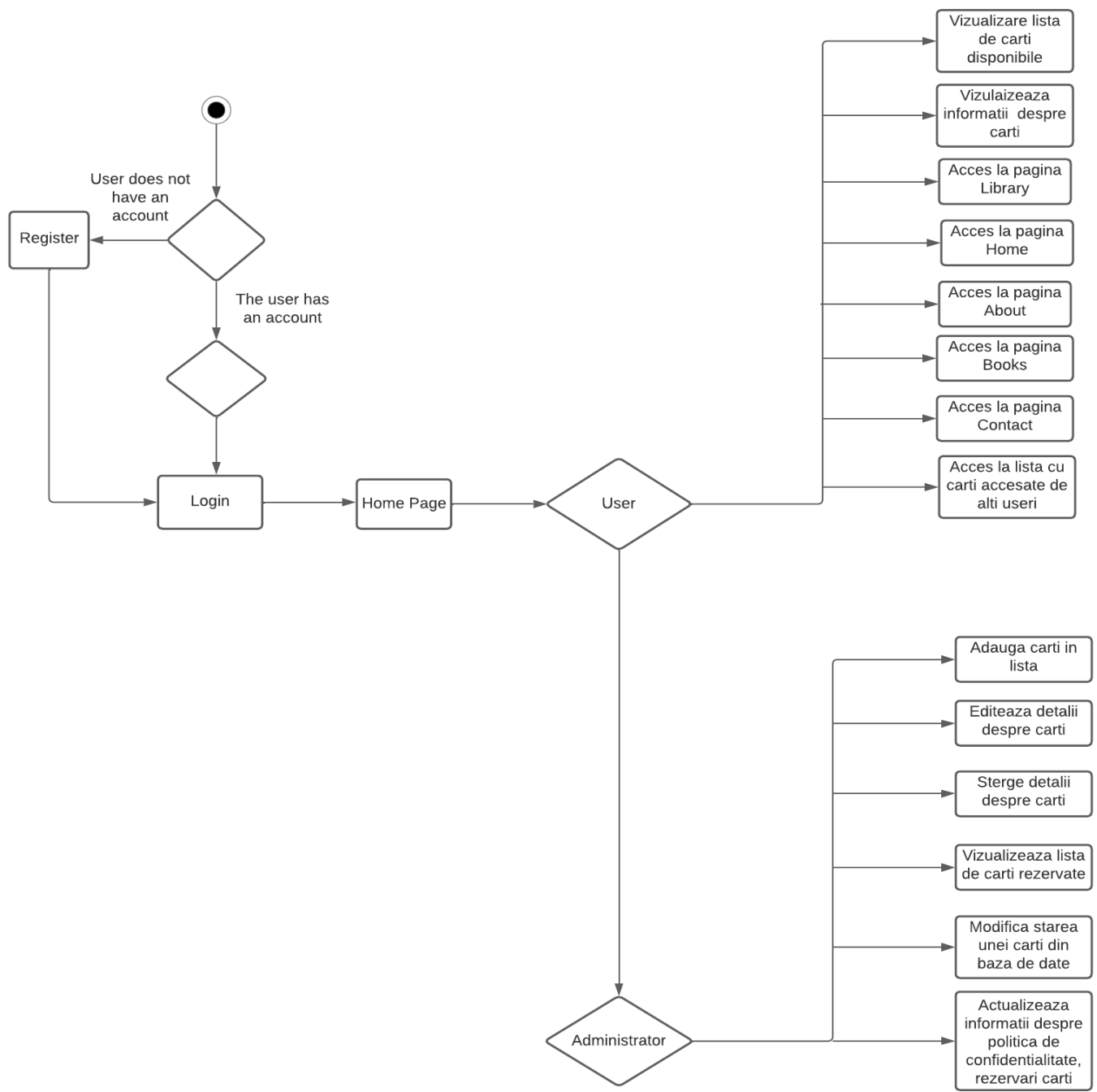
2. Modelati aplicatia folosind diagrame UML (Diagrama de cazuri de utilizare, Diagrama de activitati, Diagrama de clase, Diagrama de stare, Model de structura hipertext, Model de acces, Pagina de prezentare, Diagrama de interactiune, Diagrama de secventa).

❖ Diagrama de cazuri de utilizare:

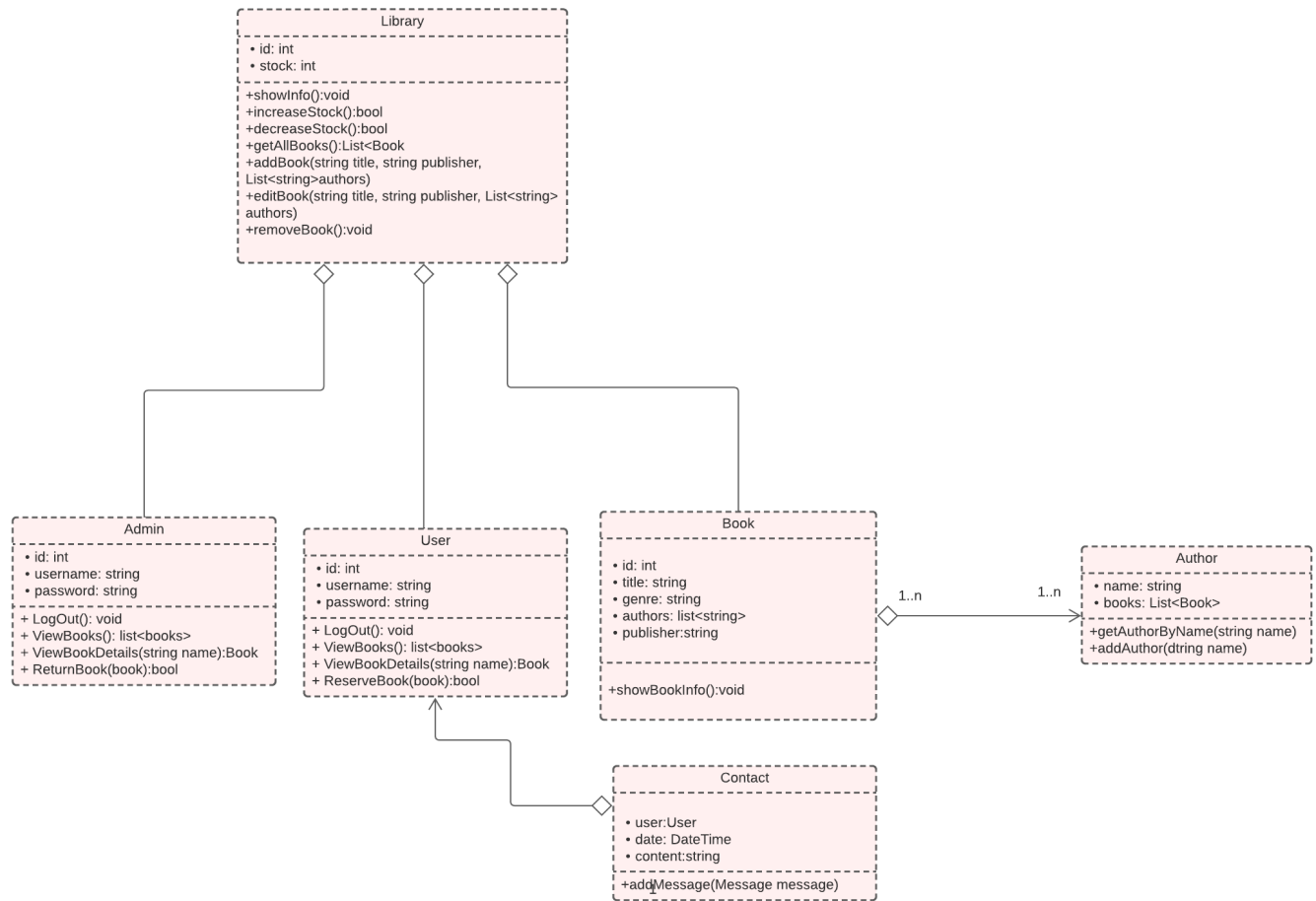


❖ Diagrama de activitati:

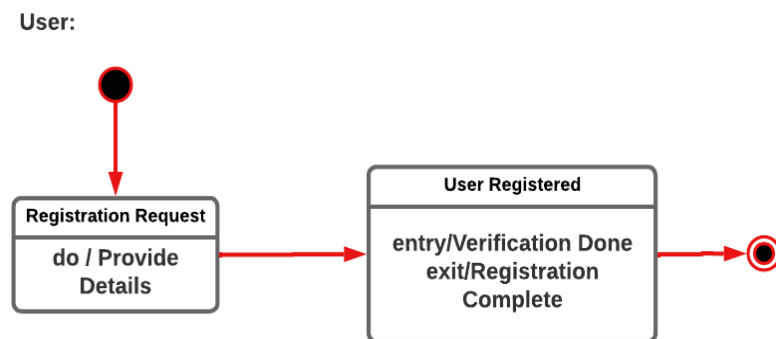
Flow-ul activitatilor este prezentat in urmatoarea diagrama:



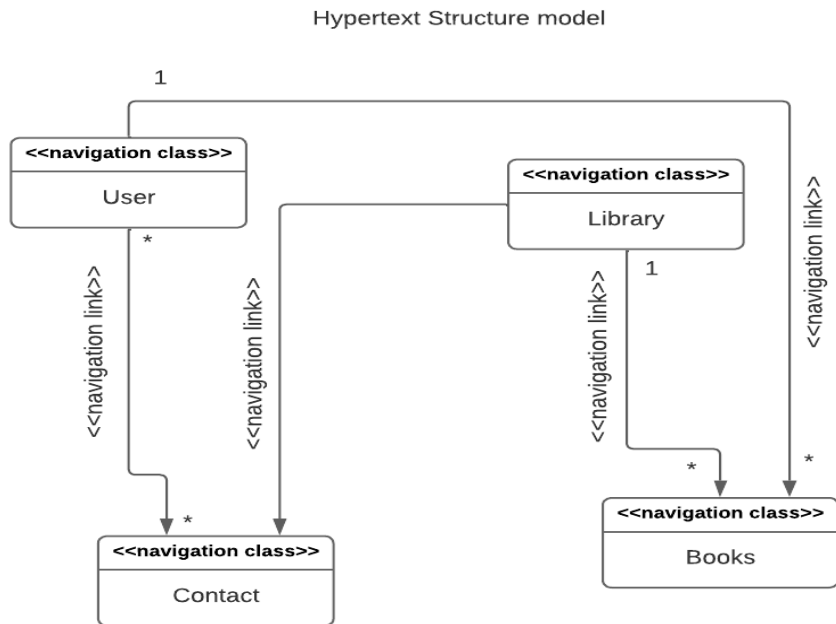
❖ Diagrama de clase:



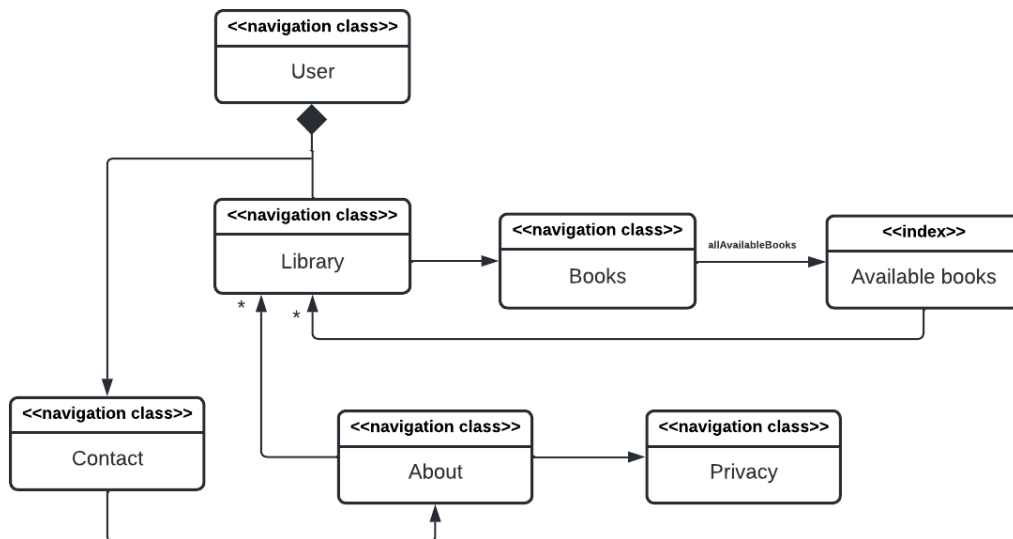
❖ Diagrama de stare:



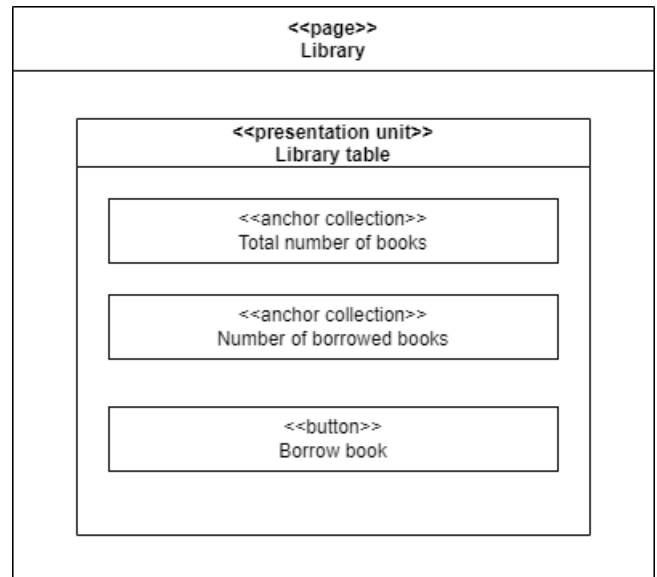
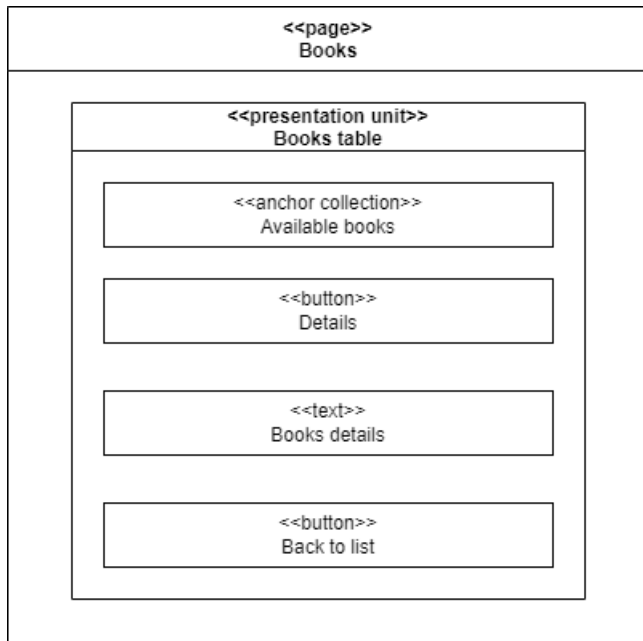
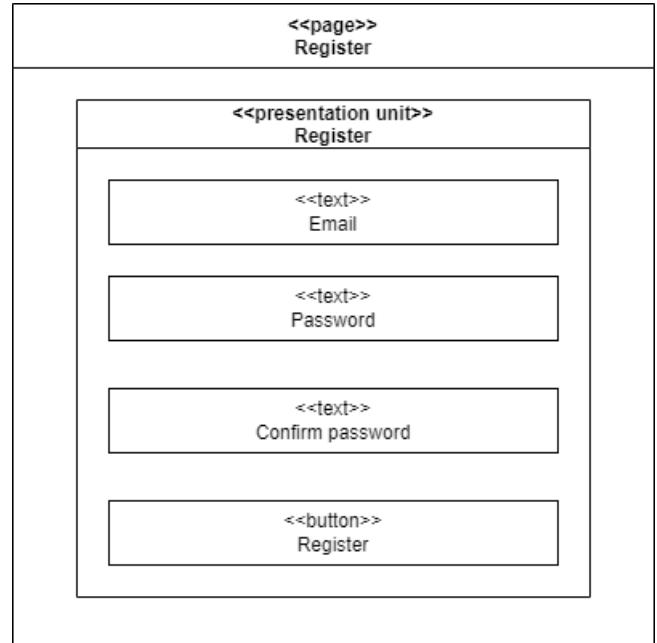
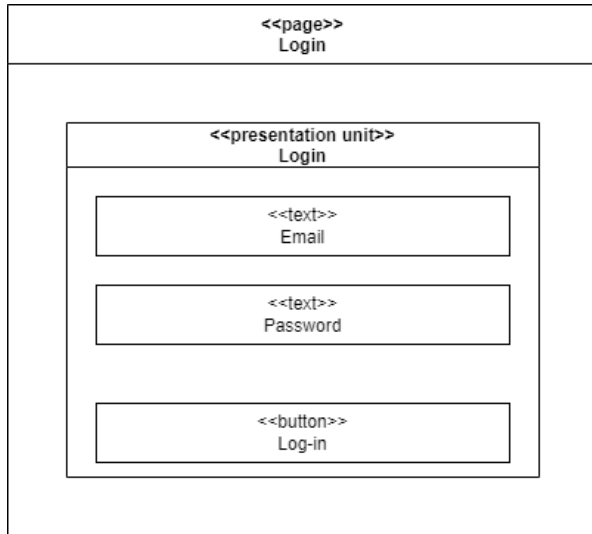
❖ **Model de structura hipertext:**



❖ **Model de acces:**



❖ Pagina de prezentare:



<<page>>
Add a book

<<presentation unit>>
Add a book

<<text>>
Name

<<text>>
Publisher

<<text>>
Category type

<<number>>
Stock id

<<number>>
Message id

<<button>>
Create

<<page>>
Update a book

<<presentation unit>>
Update a book

<<text>>
Name

<<text>>
Publisher

<<text>>
Category type

<<number>>
Stock id

<<number>>
Message id

<<button>>
Edit

<<button>>
Back to list

<<page>>
Delete a book

<<presentation unit>>
Delete a book

<<text>>
Name

<<text>>
Publisher

<<text>>
Category type

<<number>>
Stock id

<<number>>
Message id

<<button>>
Delete

<<button>>
Back to list

<<page>>
Home

<<presentation unit>>
Home

<<card>>
card 1

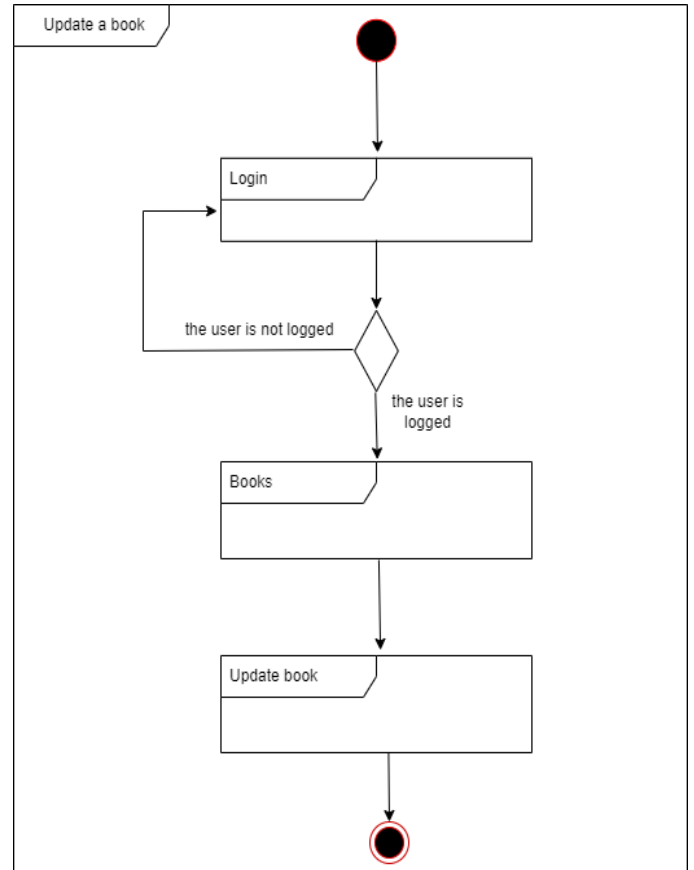
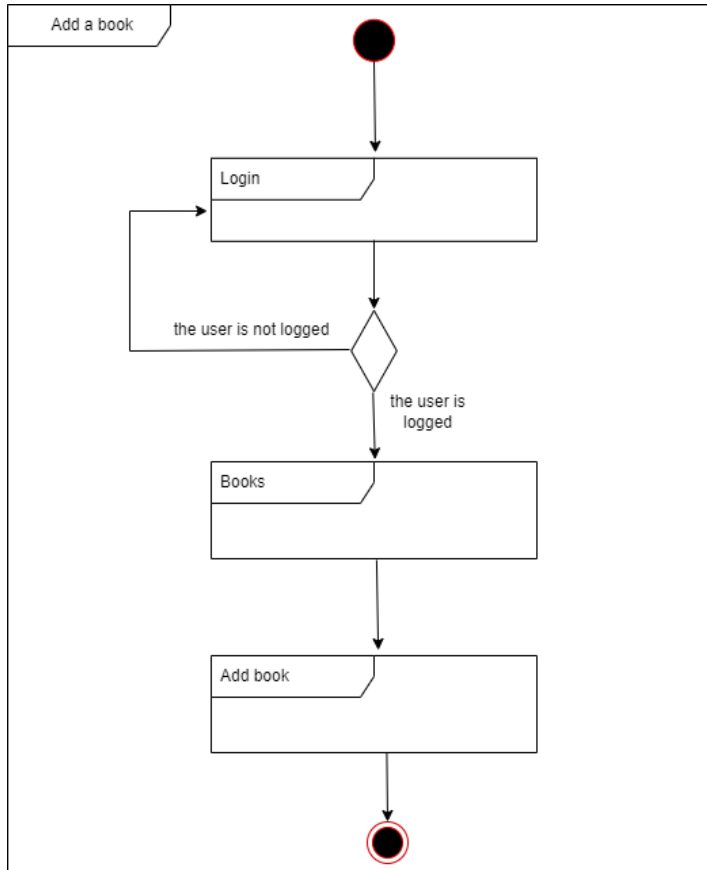
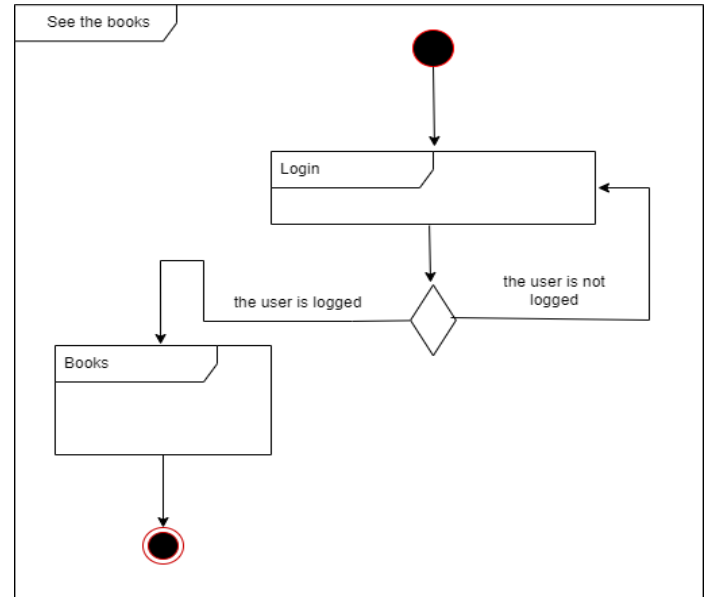
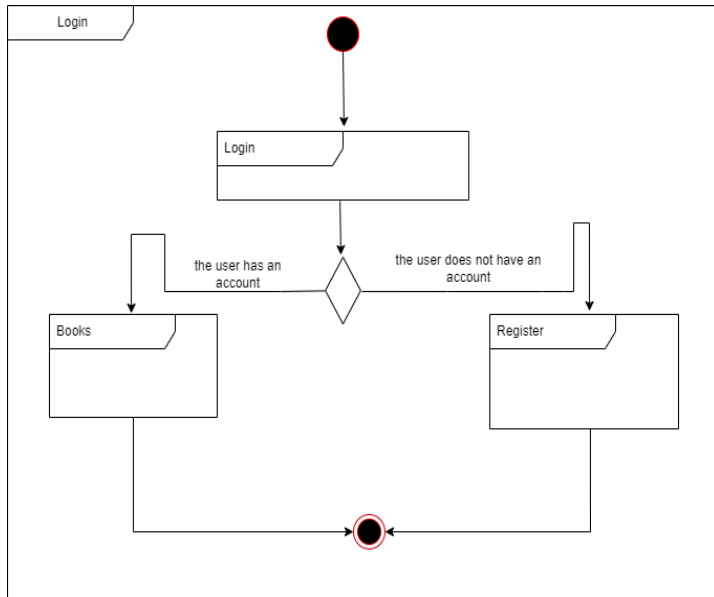
<<card>>
card 2

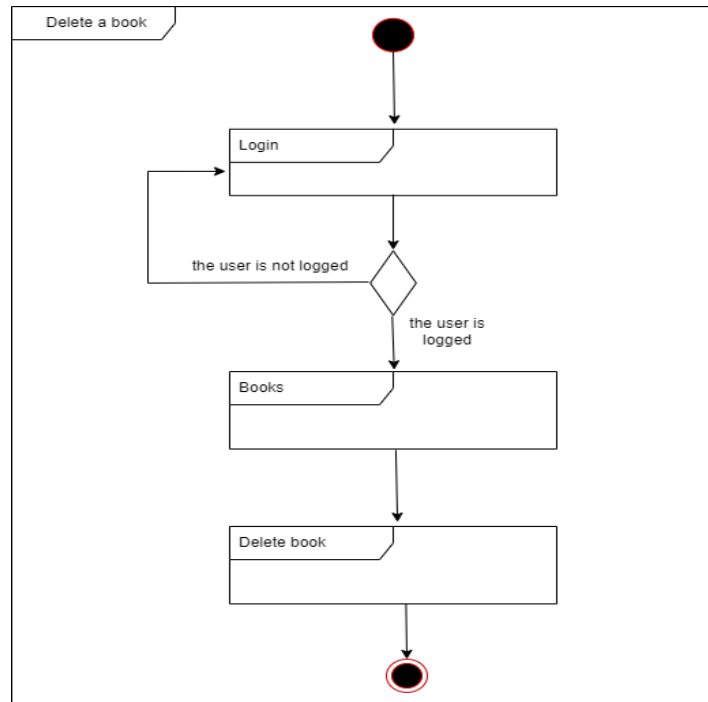
<<card>>
card 3

<<card>>
card 4

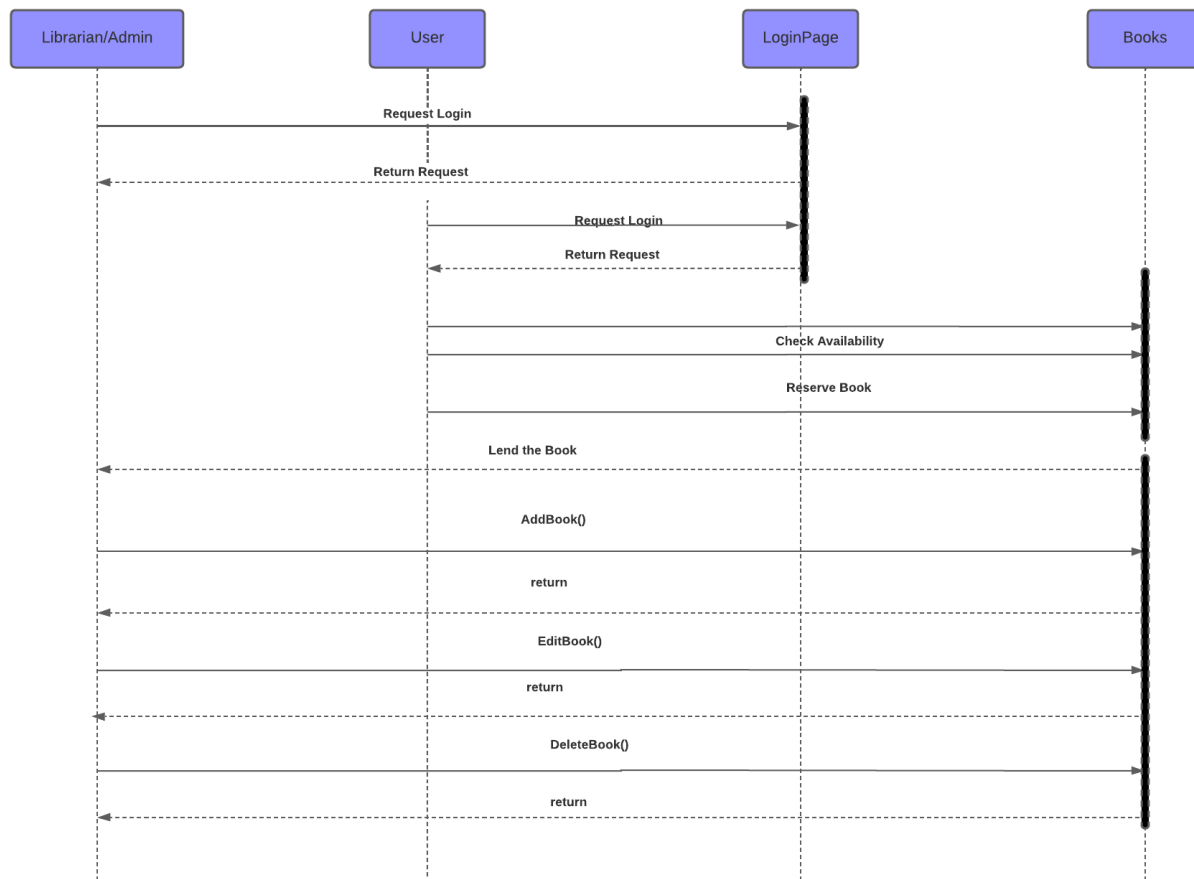
<<button>>
Go to library

❖ **Diagrama de interactiune:**



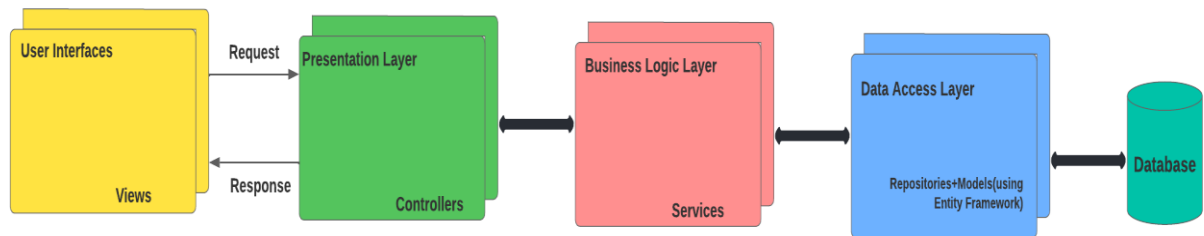


❖ **Diagrama de secventa:**



3. Descrieti arhitectura sistemului si structura bazei de date (inclusiv diagrame).

❖ Arhitectura sistemului:



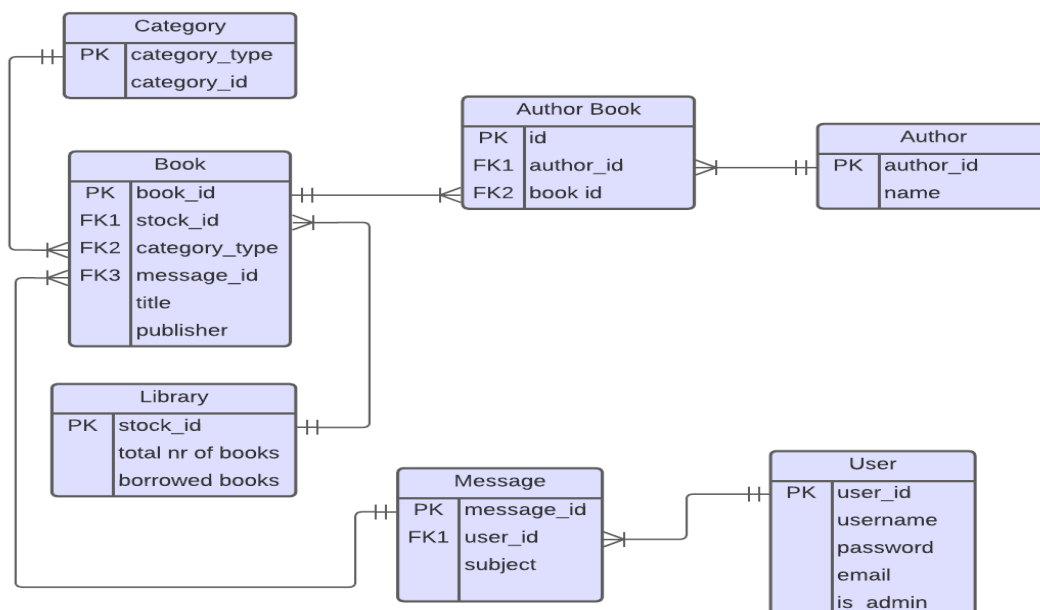
Aplicatia a fost dezvoltata folosind ASP.NET Core 6, Entity Framework Core versiunea 6.0.5 si ASP.NET Identity Core versiunea 6.0.5.

Am utilizat MVC (Model View Controller) si Repository Pattern. Controllerele aplicatiei fac parte din Presentation Layer, in Business Logic Layer sunt incluse serviciile. In Data Access Layer am incadrat Repository, unde fiecare repository va implementa o interfata specifica lui si va mosteni clasa de baza RepositoryBase, care are metode generice ca getAll, getByCondition, Update, Delete, Create, Save, etc si care implementeaza interfata IRepositoryBase.

Astfel s-a respectat principiul Don't repeat yourself. Am definit serviciile in clasa Program.cs si am realizat injectia lor pe constructorul controllerelor in care voiam sa folosesc serviciile respective (Dependency Injection).

Pentru autentificare si autorizare am folosit ASP.NET Core Identity. Am utilizat SQL Server localDB in Visual Studio 2022, unde baza de date este relationala, iar pentru utilizatori am folosit IdentityUser.

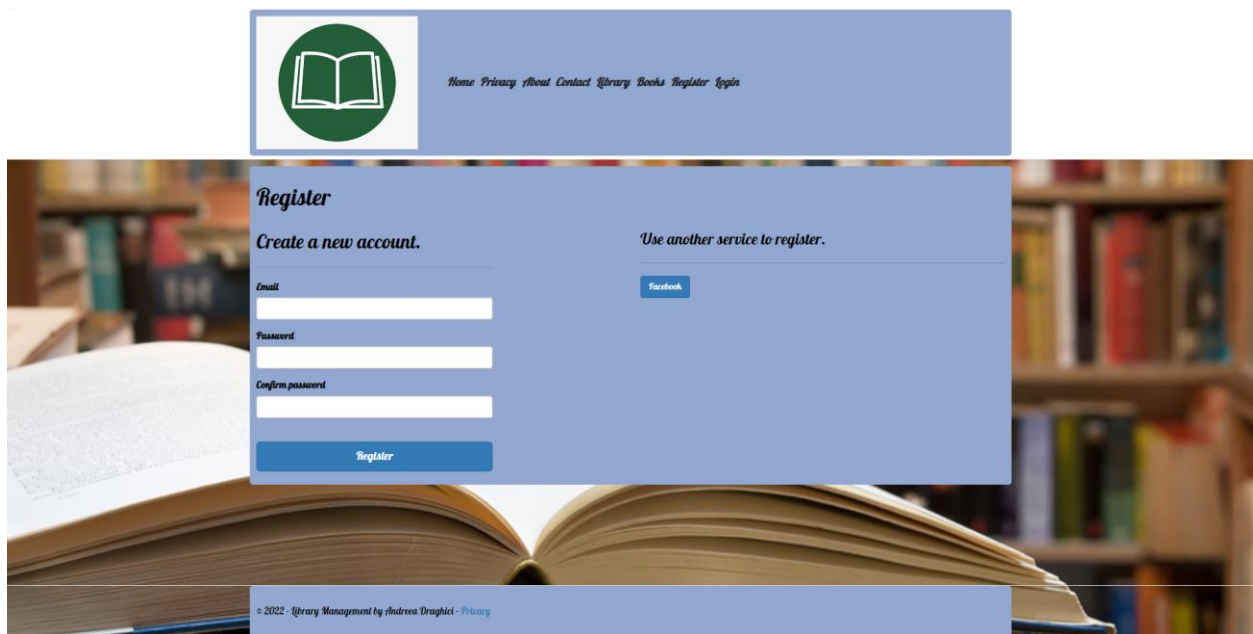
❖ Modelul Entitate-Legatura



4. Descrieti functionalitatile pe care le-ati implementat si ilustrati-le cu capturi de ecran (screenshots).

❖ Register

Pentru partea de autentificare am utilizat functionalitatea Register din ASP.Net Core Identity. Fiecare utilizator care isi creeaza un nou cont de utilizatori, va primi rolul de user si poate avea acces la toate paginile mai putin la pagina de adaugare a cartilor, pagina de stergere a unei carti, pagina de editare a unei carti, cat si pagina de vizualizarea a cartilor imprumutate, aceste functionalitati sunt accesibile doar utilizatorului cu rol de administrator.



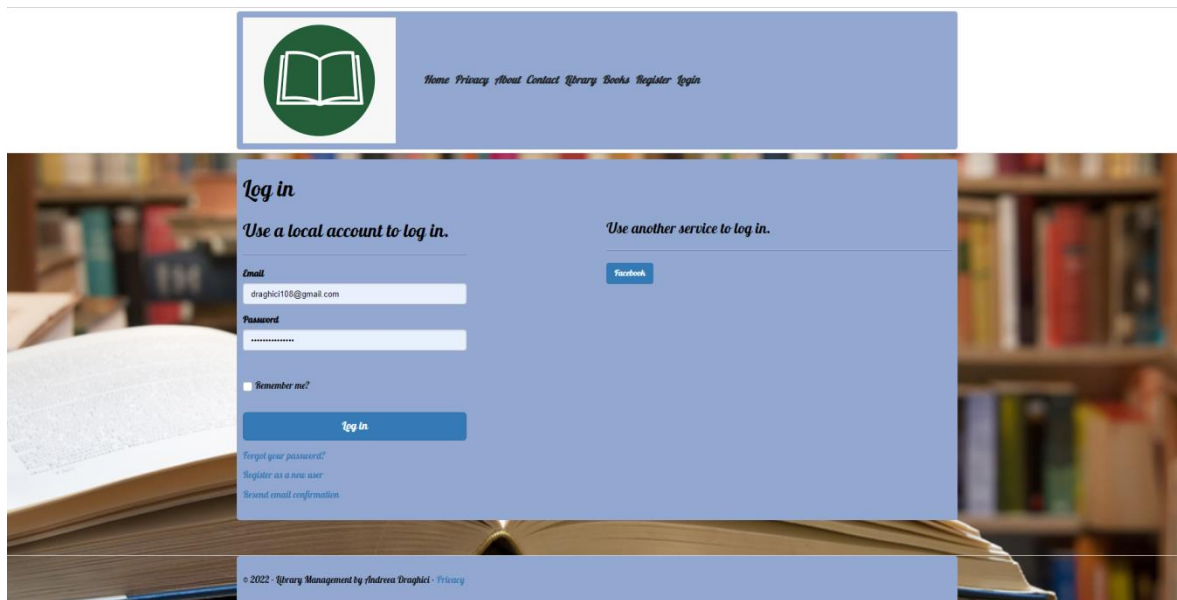
Fiecare utilizator care isi face un cont de utilizator isi poate introduce mail-ul si parola, sau se poate inregistra direct cu un provider de autentificare extern, de exemplu Facebook.

Pentru introducerea datelor am folosit validari. Parola cat si parola care este introdusa in field-ul de confirmare trebuie sa fie identice, de exemplu.

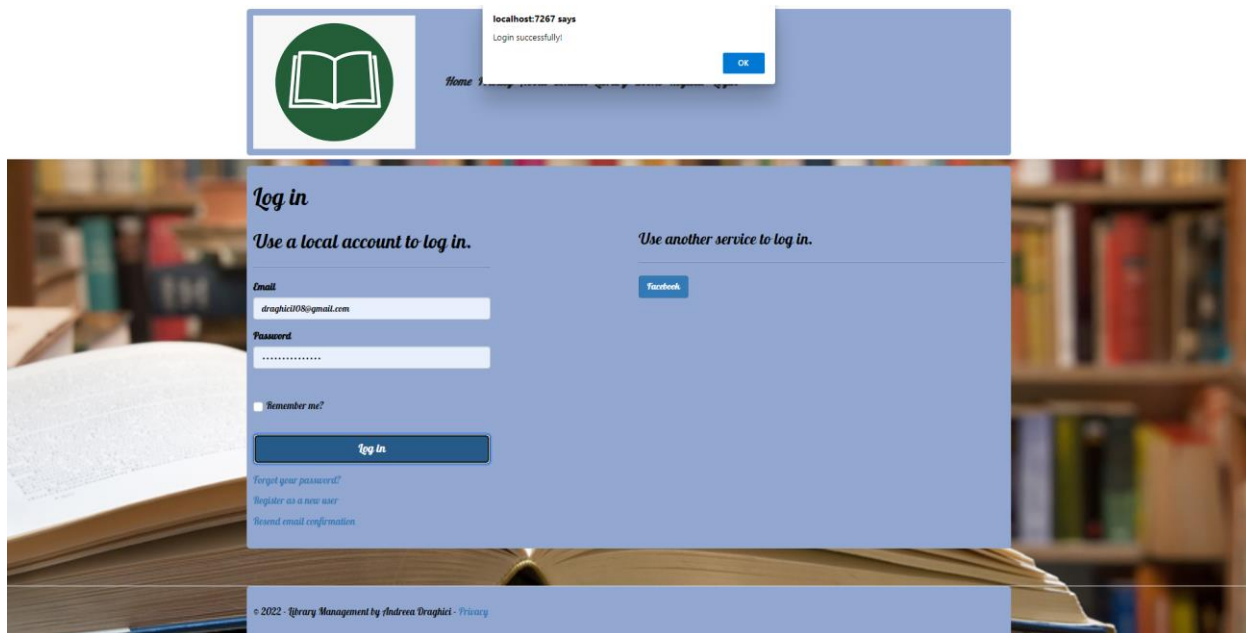
A screenshot of the 'Register' page, focusing on the form fields. The title 'Register' is at the top, followed by the subtitle 'Create a new account.'. Below this are three input fields: 'Email' (containing 'testrldmin@gmail.com'), 'Password', and 'Confirm password'. Both the 'Password' and 'Confirm password' fields are filled with dots. A red error message, 'The password and confirmation password do not match.', is displayed below the 'Confirm password' field. At the bottom of the form is a blue 'Register' button.

❖ Login

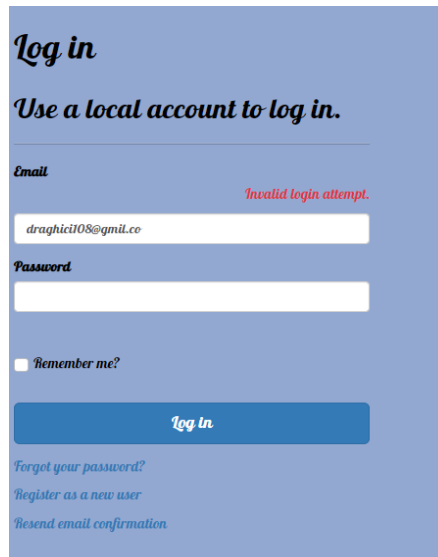
Pentru pagina de login, am utilizat functionalitatea Login din ASP.NET Core Identity. Fiecare utilizator care doreste sa se logheze isi poate introduce mail-ul si parola sau se poate loga direct cu ajutorul unui provider de autentificare extern, de exemplu Facebook. Orice utilizator, indiferent de rol, poate accesa functionalitatea aplicatiei numai dupa ce s-a logat la aplicatie.



Pentru introducerea datelor am folosit validari, in momentul in care datele introduse de utilizatori sunt valide, in partea de sus a ecranului va apare un pop-up alert cu mesajul "Login successfully!", care ii va confirma utilizatorului ca acesta s-a conectat cu succes. Pentru acest lucru, am folosit o functie in JavaScript care va fi apelata atunci cand utilizatorul isi introduce corect datele si apasa pe butonul log-in.



In cazul in care utilizatorul greseste datele cand doreste sa se logheze, va aparea urmatorul mesaj.



Log in

Use a local account to log in.

Email Invalid login attempt.

draghici108@gmail.co

Password

☐ Remember me?

Log in

[Forgot your password?](#)

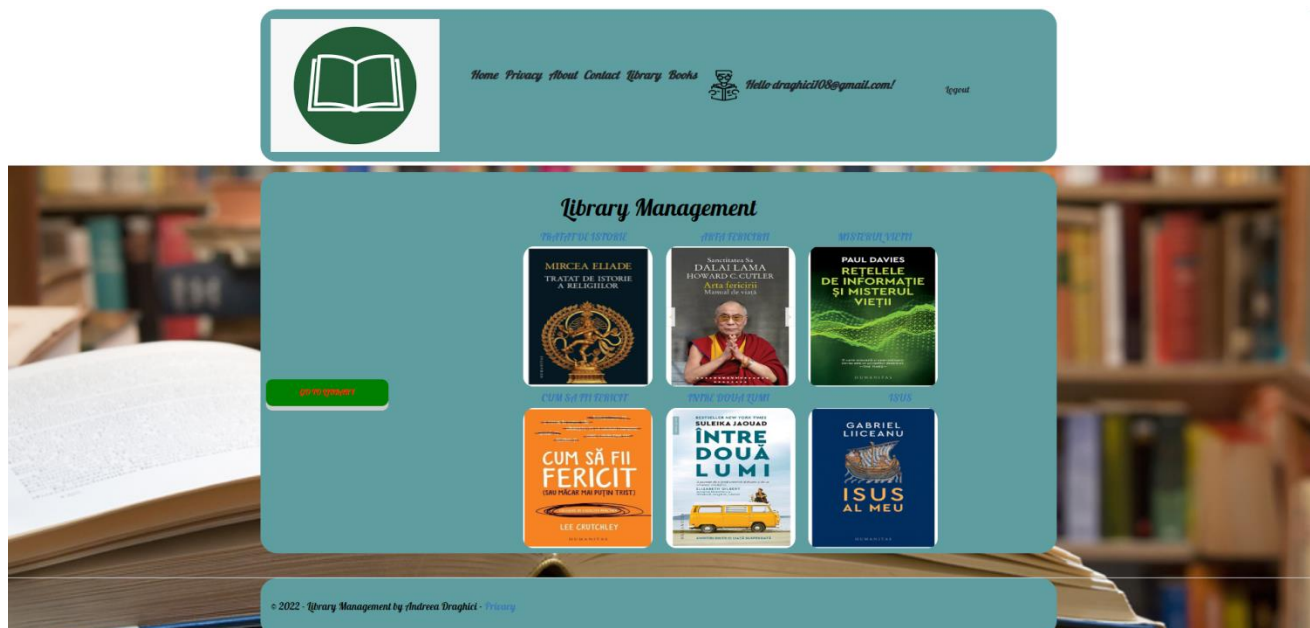
[Register as a new user](#)

[Resend email confirmation](#)

❖ Home

Dupa ce utilizatorul se conecteaza la aplicatie, acesta este redirectionat catre pagina de home, indiferent de rolul pe care acesta il are, iar email-ul acestuia cat si imaginea de profil sunt afisate in bara de navigare. Butonul de logout este si el activ acum.

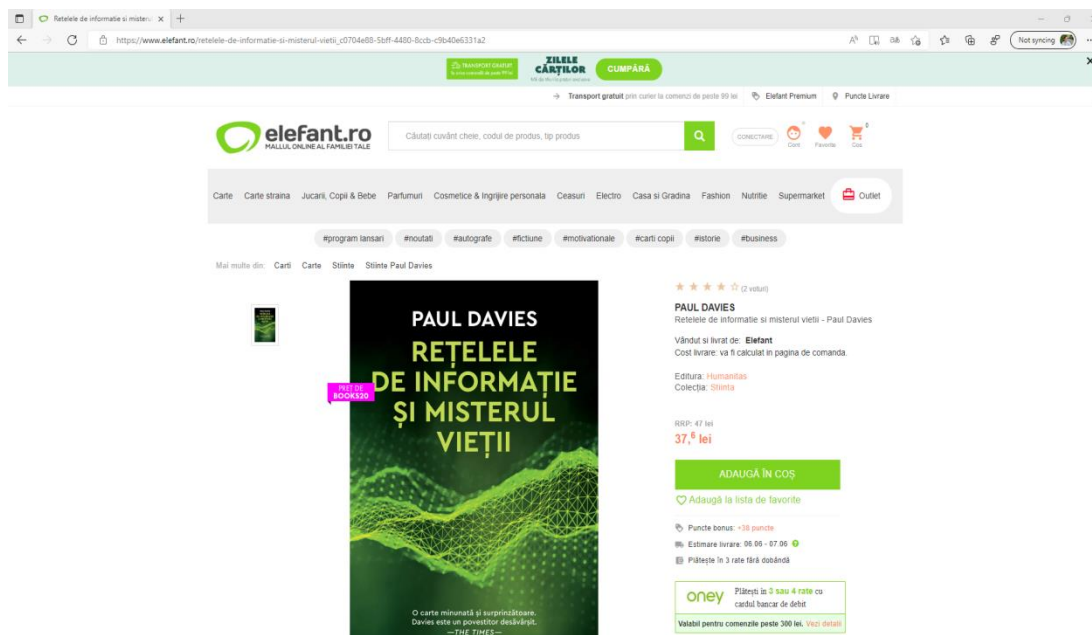
Utilizatorul normal poate naviga catre lista de carti din librerie, acest lucru fiind posibil accesand butonul de 'Go to library', pe langa aceasta functionalitate, utilizatorul mai are acces atat la informatiile, cat si la fotografiile despre cartile care au fost accesate de catre alti utilizatori.



Pentru a vizualiza informatii despre cartile accesate de alti utilizatori, utilizatorul trebuie sa acceseze link-urile care se afla deasupra fiecarui card din pagina de home, acesta va fi redirectionat catre o pagina web externa de pe un site ce contine mai multe informatii despre respectiva carte accesata.



Accesand link-ul “Misterul vietii” de deasupra cardului “Rețelele de informatie si misterul vietii”, utilizatorul a fost redirectionat catre o pagina ce contine mai multe detalii despre cartea respectiva.



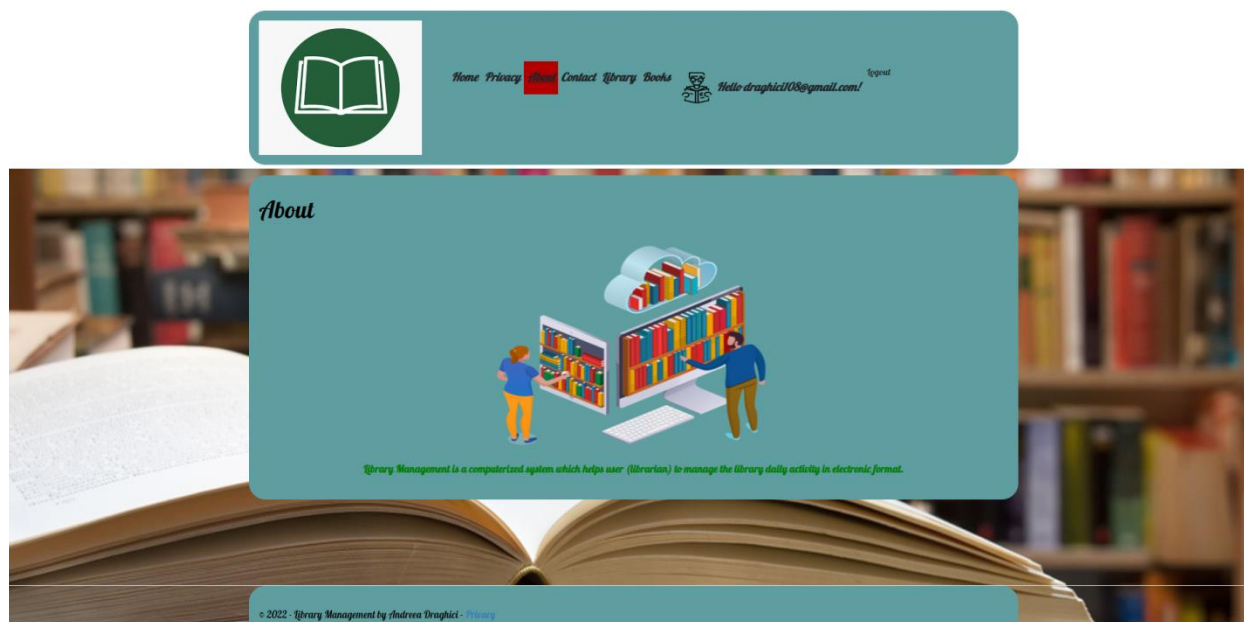
❖ Privacy

Accesand pagina de Privacy Policy, utilizatorul poate vizualiza care este confidentialitatea si politica aplicatiei raportata reglementarilor si datelor stocate in baza de date. Pentru mai multe detalii despre politica de confidentialitatea, utilizatorul poate accesa link-ul Privacy si va fi redirectionat catre o pagina web externa.





❖ About

Accesand pagina About, utilizatorul nostru poate vizualiza informatii despre activitatea aplicatiei noastre.



❖ Books



Utilizatorul are posibilitatea de a vizualiza informatii despre lista cu carti disponibile in librerie, nume, categorie, editura, id-ul carti din stoc. Butoanele Edit si Delete ofera functionalitate doar utilizatorului cu rolul de administrator, in cazul in care acestea sunt accesate de un utilizator cu rolul de user, va aparea un mesaj de forma 'Access denied'.

[Home](#) [Privacy](#) [About](#) [Contact](#) [Library](#) [Books](#)  [Hello draghici08@gmail.com!](#) [Logout](#)

Book Description

name	publisher	category_type	stock_id	message_id	
nothing	nothing	not	0	1	Edit Details Delete
Carte	Cerint	nothing	1	2	Edit Details Delete
Book	Elefant	Aventura	2	3	Edit Details Delete

© 2022 - Library Management by Andreea Draghici - [Privacy](#)

[Home](#) [Privacy](#) [About](#) [Contact](#) [Library](#) [Books](#)  [Hello draghici08@gmail.com!](#) [Logout](#)

Details

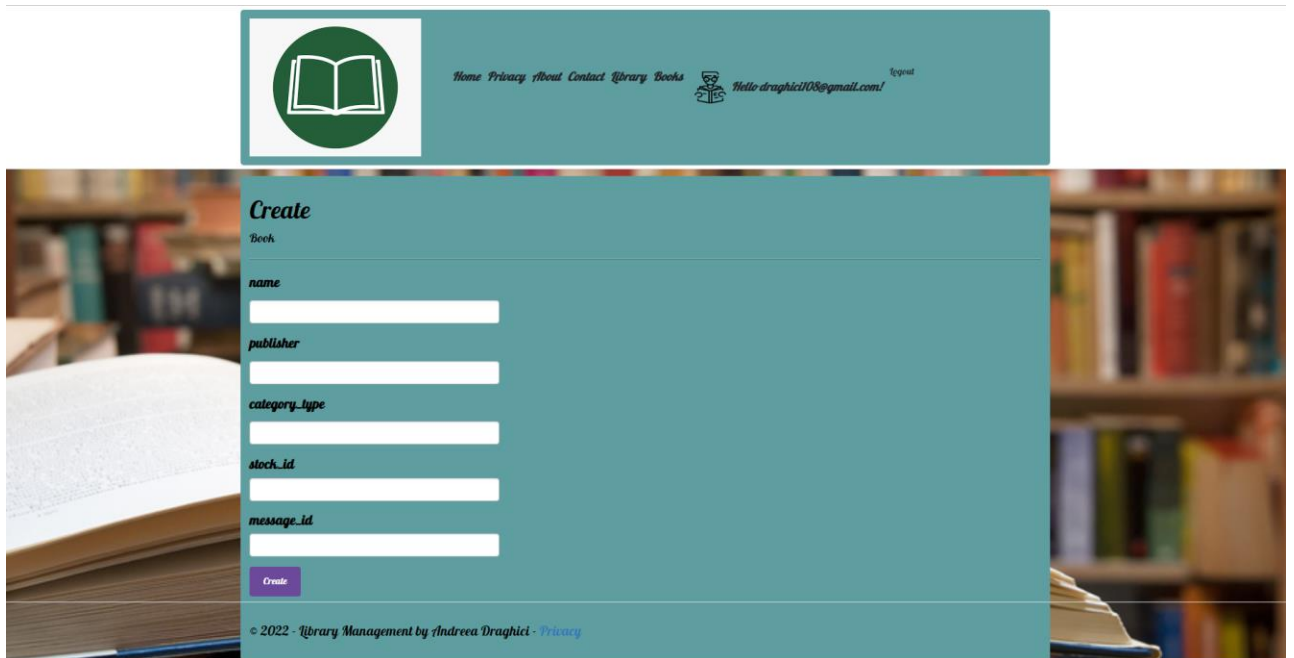
Book

name	nothing
publisher	nothing
category_type	not
stock_id	0
message_id	1

[Edit](#) | [Back to List](#)

© 2022 - Library Management by Andreea Draghici - [Privacy](#)

Utilizatorul cu rolul de administrator poate adauga o noua carte in lista de carti, toate field-urile trebuie completate, daca administratorul nu a introdus date valide sau a uitat sa completeze unul sau mai multe field-uri, atunci operatia va esua si va primi mesaje de validare, acesta va ramane pe aceeasi pagina curenta pentru a incerca iarasi sa adauge date despre carte. Fiecare carte este identificata dupa nume, editura, categorie. Mai mult, se poate specifica si ce id ocupa in stocul cu carti, cat si id-ul pentru mesaj, in cazul in care un utilizatorul cu rolul de user a lasat un mesaj pentru o anumita carte. Pentru a se inregistra cartea in baza de date, administratorul trebuie sa apese pe butonul de create.



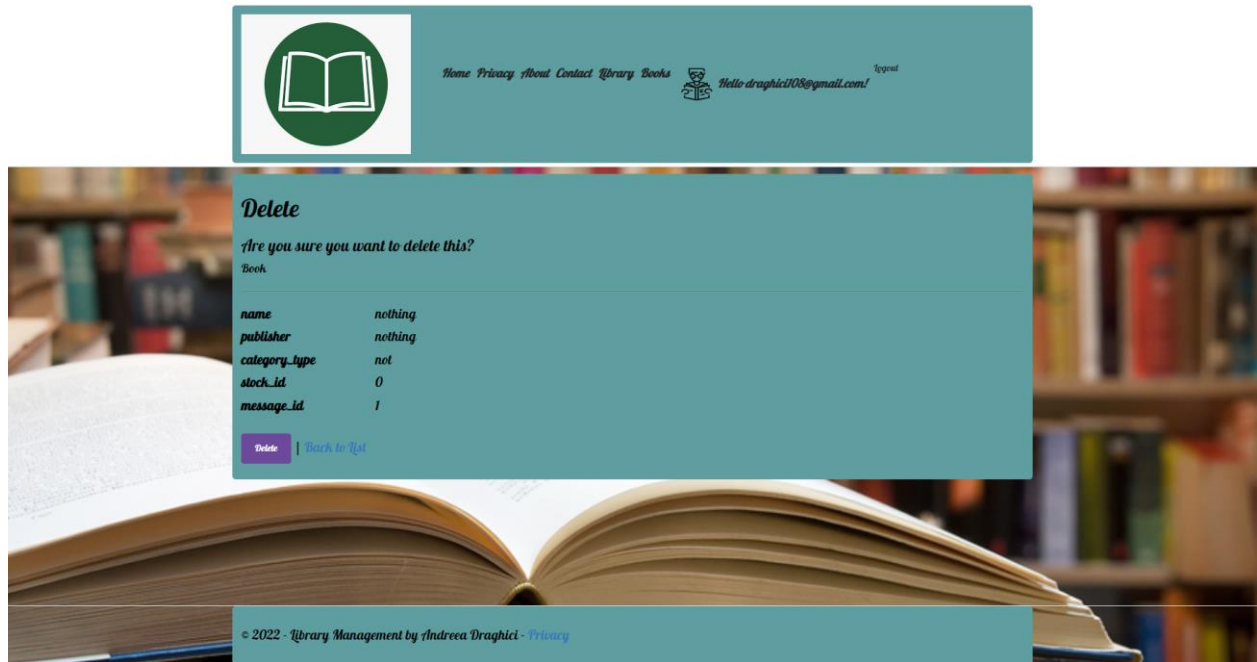
The screenshot shows the 'Create' form for adding a new book. The form is titled 'Create' and has a 'Book' label. It contains five input fields: 'name', 'publisher', 'category_type', 'stock_id', and 'message_id'. A 'Create' button is located at the bottom of the form. The background of the form is a teal color. The top navigation bar is also teal and contains a logo, links for Home, Privacy, About, Contact, Library, and Books, a user profile icon, and a 'Logout' link. The footer of the form area shows the copyright notice '© 2022 - Library Management by Andreea Draghici - Privacy'.

Utilizatorul cu rolul de administrator poate edita anumite informatii despre o carte. La final, pentru a salva continutul despre o anumita carte, acesta trebuie sa apese pe butonul 'save' si baza de date va fi actualizata, in cazul in care doreste sa se intoarca inapoi pe pagina unde se afla lista cu carti, acesta poate accesa butonul 'back to list'.



The screenshot shows the 'Edit' form for updating an existing book. The form is titled 'Edit' and has a 'Book' label. It contains five input fields: 'name', 'publisher', 'category_type', 'stock_id', and 'message_id'. A 'Save' button is located at the bottom of the form. The background of the form is a light blue color. The top navigation bar is also light blue and contains a logo, links for Home, Privacy, About, Contact, Library, and Books, a user profile icon, and a 'Logout' link. The footer of the form area shows the copyright notice '© 2022 - Library Management by Andreea Draghici - Privacy'. A 'Back to List' link is located at the bottom left of the form.

Daca se doreste ca o anumita carte sa fie exclusa din lista de carti, utilizatorul cu rolul de administrator poate realiza acest lucru alegand care carte sa fie stearsa din lista, apoi se apasa butonul 'delete', iar cartea respectiva va fi exclusa din lista de carti si baza de date va fi actualizata, in cazul in care doreste sa se intoarca inapoi pe pagina unde se afla lista cu carti, acesta poate accesa butonul 'back to list'.

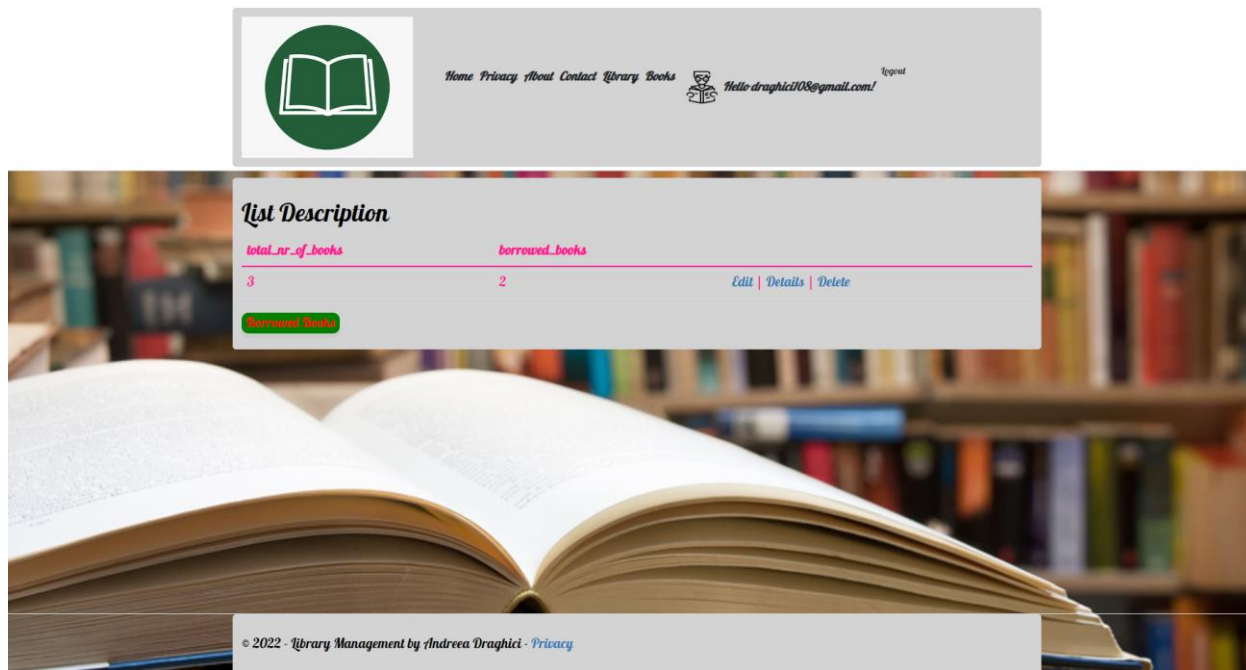


In cazul in care un utilizator cu rolul de user va accesa butonul de 'delete' de pe pagina Books, acesta va fi redirectionat catre o pagina ce contine mesajul 'Access denied'.

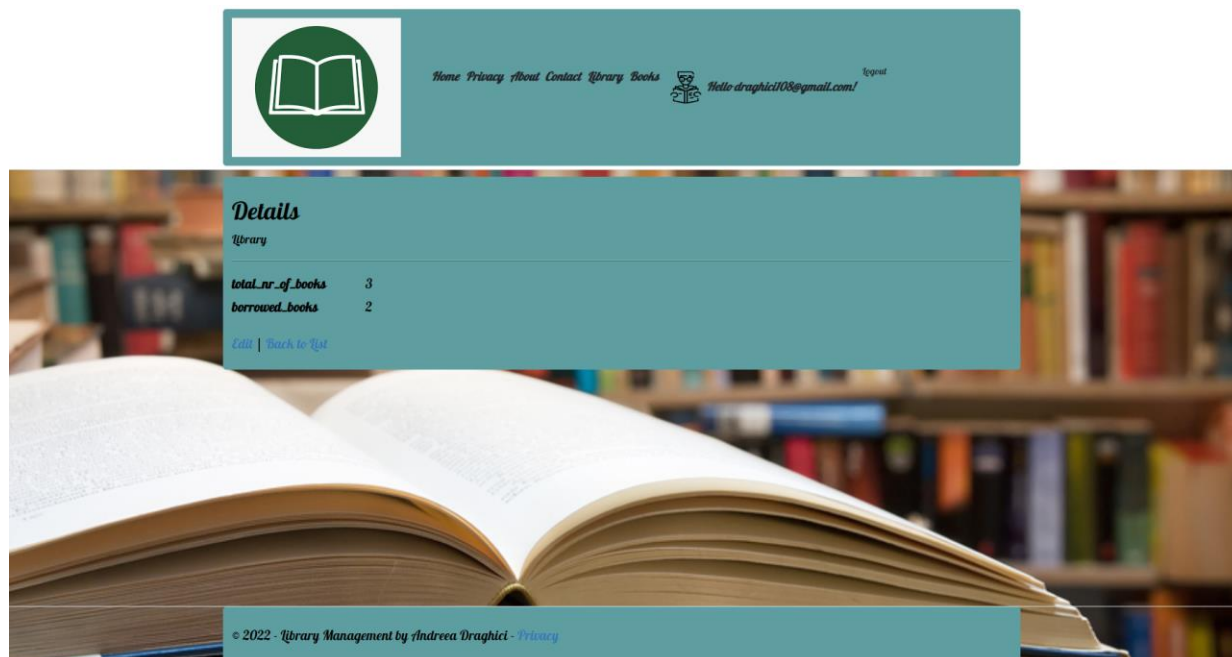


❖ Library

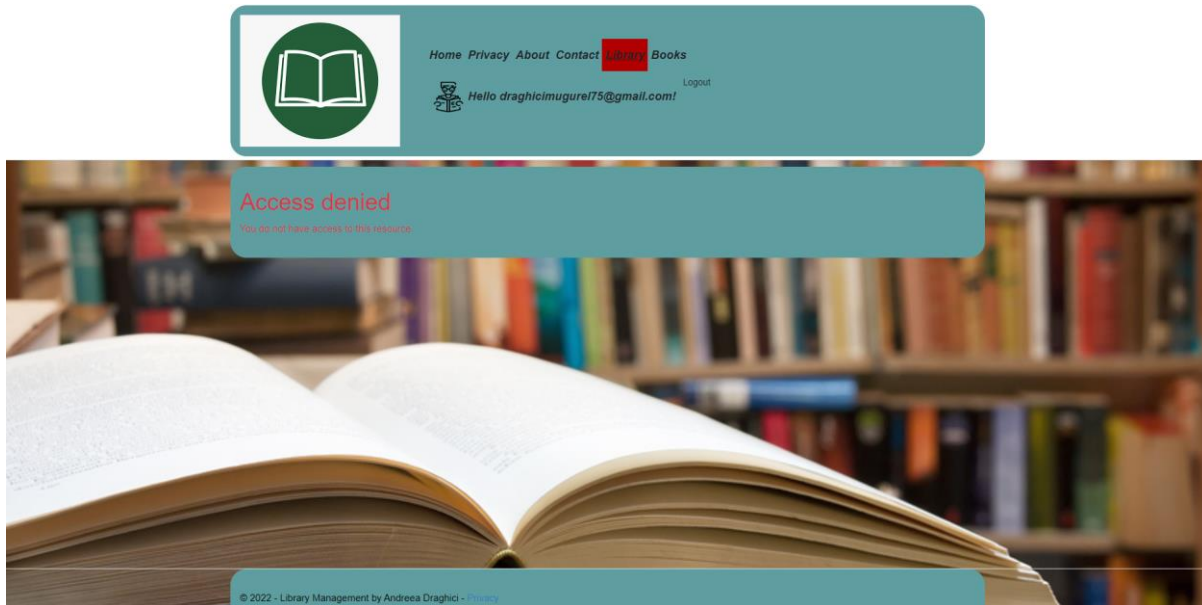
Utilizatorul cu rolul de user poate vizualiza lista cu numarul de carti rezervate, cat si numarul total de carti din librerie. Utilizatorul poate sa imprumute o carte, accesand butonul de 'borrowed book'.



Utilizatorul cu rolul de administrator poate vizualiza lista de carti rezervate, modifica, adauga sau sterge detalii despre aceasta. Un utilizator cu rolul de user nu poate avea acces la aceste functionalitati.

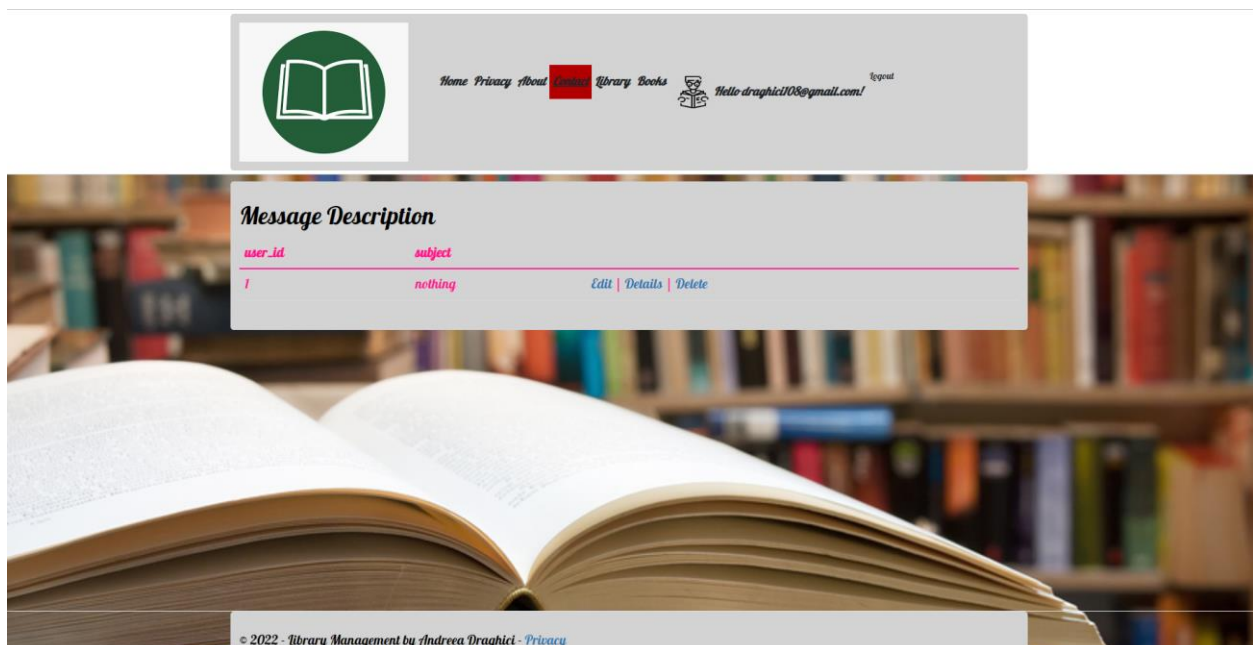


Utilizatorul cu rolul de user nu va avea acces la functionalitatile de create, edit sau delete a listei cu numarul total de carti sau cu numarul de carti rezervate din librerie.



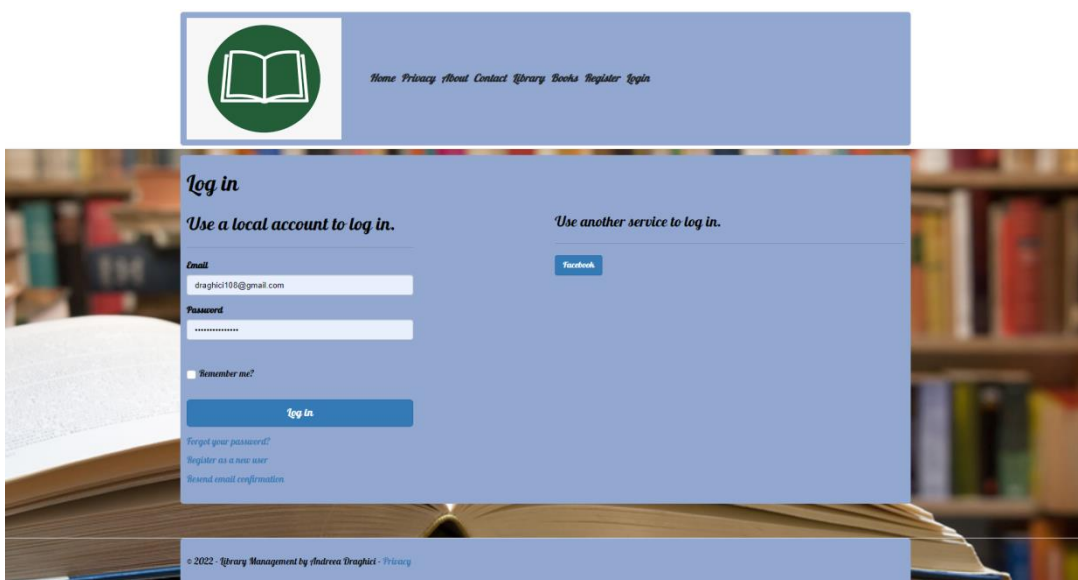
❖ Contact

Utilizatorul poate trimite un ticket catre asistenta pentru utilizatori cu privire la orice problema intalnita in timpul aplicatiei. In baza de date vor fi stocate doar subiectul mesajului si id-ul utilizatorului. Doar utilizatorul cu rolul de administrator poate accesa functionalitatile de 'edit', 'delete', 'create'.



❖ Logout

Utilizatorul se poate deconecta apasand pe butonul de logout, apoi acesta este redirectionat catre pagina de login. Acesta este deconectat si nu va mai putea accesa toate paginile decat atunci cand se va conecta iar. Daca incearca sa acceseze o pagina si nu este conectat, acesta va fi redirectionat catre pagina de login, de specificat faptul ca in momentul cand utilizatorul este deconectat de la aplicatie, acesta devine utilizator anonim si nu va mai putea accesa functionalitatea paginilor decat atunci cand se va conecta din nou.



5. Ce metode de testare ati folosit pentru aplicatia voastra? Descrieti procesul de testare, precum si o problema identificata si solutia aplicata pentru rezolvarea sa.

Pentru a testa aplicatia, am utilizat testarea functionala, astfel am testat fiecare flow dupa implementarea sa.

Cateva exemple din test case-urile efectuate sunt urmatoarele:

❖ Test Case 1:

Title	Butonul de Login
Steps	1. Deschide aplicatia 2. In pagina de Login se introduc: Email-ul si parola 3. Se apasa butonul de login
Expected result	Dupa ce se apasa butonul de login, apare un popup care contine mesajul 'Login successfully' ceea ce inseamna ca butonul respectiv este activ.

❖ **Test Case 2:**

Title	Butonul de Register
Steps	1. Deschide aplicatia 2. In pagina de Register se introduc: Email-ul, parola si iarasi parola pentru a fi confirmata 3. Se apasa butonul de register
Expected result	Dupa ce se apasa butonul de register, utilizatorul va fi inregistrat si conectat automat in aplicatie, iar credentialele acestuia vor fi salvate si stocate in baza de date.

❖ **Test Case 3:**

Title	Pagina de Home
Steps	1. Deschide aplicatia 2. Se introduce datele pentru conectare in aplicatie 3. Validam aceste date prin apasarea butonului de login 4. Apasam pe butonul de OK in popup-ul care ne confirma ca datele sunt valide
Expected result	Utilizatorul este redirectionat catre pagina de Home.

❖ **Test Case 4:**

Title	Introducere date invalide
Steps	1. Deschide aplicatia 2. Se introduce urmatoarele date invalide pentru conectare in aplicatie Email: draghici108@gmail.co Password: 1234 3. Apasam butonul login
Expected result	La introducerea datelor invalide, mesajul de form 'invalid login attempt' va aparea deasupra campului email pe font de culoare rosie.

Aplicatia poate fi testata si utilizand testarea automata. Acest lucru este posibil facand unit teste in C# pentru ASP Net Core. Pentru a realiza aceste teste unitare este nevoie sa includem framework-ul de la Microsoft Unit Testing pentru a gestiona codul.

Un exemplu de metoda pentru testarea realizata este vizibil mai jos:

```
[TestMethod()]
public void GetBooksTest()
{
    //arrange
    var name = "Reservations";
    var book = new List<Book>();

#pragma warning disable CS8602 // Dereference of a possibly null reference.
    Assert.IsNotNull(book);
#pragma warning restore CS8602 // Dereference of a possibly
}
```


Am initializat obiectul si am stabilit valoarea datelor. Sectiunea Assert verifica daca actiunea metodei testate se comporta conform asteptarilor noastre. Se poate observa mai sus ca in urma rularii testului, acesta are o bifa de culoare verde ceea ce inseamna ca este passed.

Pentru automatizarea testelor, consider ca se poate utiliza si Postman, deoarece acesta poate fi utilizat pentru a automatiza o gama larga de teste, de la teste unitare, pana la teste functionale, teste de integrare etc.

6. Elaborati un plan de promovare a aplicatiei voastre.

Pentru a promova aplicatia, as utiliza reclame online, videoclipuri de promovare. Mai exact, as folosi videoclipuri demonstrative care sa arate utilitatea aplicatiei, cat si faptul ca aceasta nu este complicat de utilizat.

As realiza mici campanii pe care sa le distribui mai departe in mediul online, pe retelele de socializare, astfel informatia ar ajunge la cat mai multe persoane si acestea ar putea vedea cum se poate utiliza aplicatia si functionalitatile pe care le ofera aceasta, astfel existand o probabilitate mai mare de a atrage mai multi potentiali utilizatori/clienti. Eventual, as realiza un mic tutorial ca overview al aplicatiei, as incerca sa sa le prezint intr-un mod cat mai clar si concis scopul si atributiile pe care aplicatia le detine.

7. Descrieti aspectele de utilizabilitate si accesibilitate relevante pentru aplicatia voastra; cum ati putea sa le integrati / implementati?

❖ Utilizabilitate:

Aplicatia Library Management este usor de utilizat deoarece toate paginile pot fi accesate din bara de navigatie, ele fiind impartite pe sectiuni: Home, Privacy, About, Contact, Library, Books, Register, Login. Functionalitatea fiecărei pagini iarasi este usor de invatat si utilizat, deoarece pentru paginile ce contin field-uri care trebuie completate, am utilizat mesaje de validare astfel incat utilizatorul sa stie cum sa completeze aceste campuri cu date si informatii corecte.

Contrastul dintre text si background-ul paginilor face ca text-ul sa fie usor de inteles si descifrat, textul fiind vizibil pentru utilizatori. Paginile au un layout destul de friendly, ceea ce inseamna ca utilizatorul se poate concentra pe functionalitati fara sa se simta incarcat de complexitatea paginilor sau a aplicatiei.

❖ Accesibilitate:

Ca si in cazul utilizabilitatii, si in cazul accesibilitatii, contrastul dintre text si background-ul paginilor face ca text-ul sa fie usor de inteles si descifrat, textul fiind vizibil pentru utilizatori, astfel acestia vor putea intelege atat informatia, cat si modul de operare al interfetei cu utilizatorul.

Aplicatia este accesibila indiferent de echipamentul hardware sau denumirea browser-ului pe care utilizatorul le are la indemana, astfel daca tehnologia evolueaza, continutul aplicatiei ramane accesibil.

Dupa ce se logheaza in aplicatie, utilizatorul poate naviga de pe o pagina pe alta utilizand tastele TAB+ENTER sau TAB+SHIFT, mai apoi ENTER. Mai exact, acesta poate sarii la urmatorul sau la anteriorul element activ din pagina. Cand se apasa TAB sau TAB+SHIFT se poate observa ca atunci cand se apasa una din combinatiile de taste specificate, se muta si cursorul virtual si este citita si eticheta link-ului.

8. Descrieti aspectele de securitate relevante pentru aplicatia voastra; cum ati putea sa le integrati / implementati?

Aspectele de securitate ar fi autentificarea si autorizarea pe roluri pentru utilizatorii ce folosesc aplicatia. Atat pentru autentificare, cat si pentru autorizare am folosit Identity, unde clasa User mosteneste IdentityUser.

```
[NotMapped]
36 references
public class User : IdentityUser
{
    [Key]
    13 references
    public int user_id { get; set; }

    12 references
    public string? username { get; set; }
    12 references
    public string? password { get; set; }

    12 references
    public string? email { get; set; }

    12 references
    public bool? is_admin { get; set; }

    0 references
    public Message? Message { get; set; }

    [NotMapped]
    0 references
    public object Ida { get; internal set; }
    [NotMapped]
    0 references
    public object PasswordHasha
    {
        get; internal set;
    }
}
```

Parola este stocata in baza de date folosind un algoritm de hashing inclus in ASP.NET Identity, iar aplicatia este protejata de SQL injection (SQLi) deoarece am folosit Entity Framework Core, acesta avand deja incluse mecanisme pentru prevenirea acestui tip de atac. Pe langa acest lucru, pentru pagina de Register si Login am adaugat si validari de date care vor specifica daca datele introduse respecta formatul precizat.

Un exemplu de format pentru validarea parolei la inregistrarea in aplicatie este faptul ca aceasta trebuie sa cuprinda litere mari, sunt premise cifrele si simbolurile, iar aceasta trebuie sa contina minim 6 caractere.

```
options.Password.RequireDigit = true;  
options.Password.RequireLowercase = false;  
options.Password.RequireNonAlphanumeric = true;  
options.Password.RequireUppercase = true;  
options.Password.RequiredLength = 6;  
options.Password.RequiredUniqueChars = 1;
```

Am adaugat si partea de autorizare, astfel incat sa existe doua roluri pentru utilizatori. Exista doua tipuri de utilizatori, utilizatorii care au rol de administrator si utilizatorii care au rol de user. Diferenta este ca utilizatorii cu rol de user nu vor putea avea acces la paginile destinate unui utilizator cu rol de administrator. Exista si tipul de utilizator anonim, acesta va fi nevoit sa se conecteze la un cont valid pentru a accesa paginile.

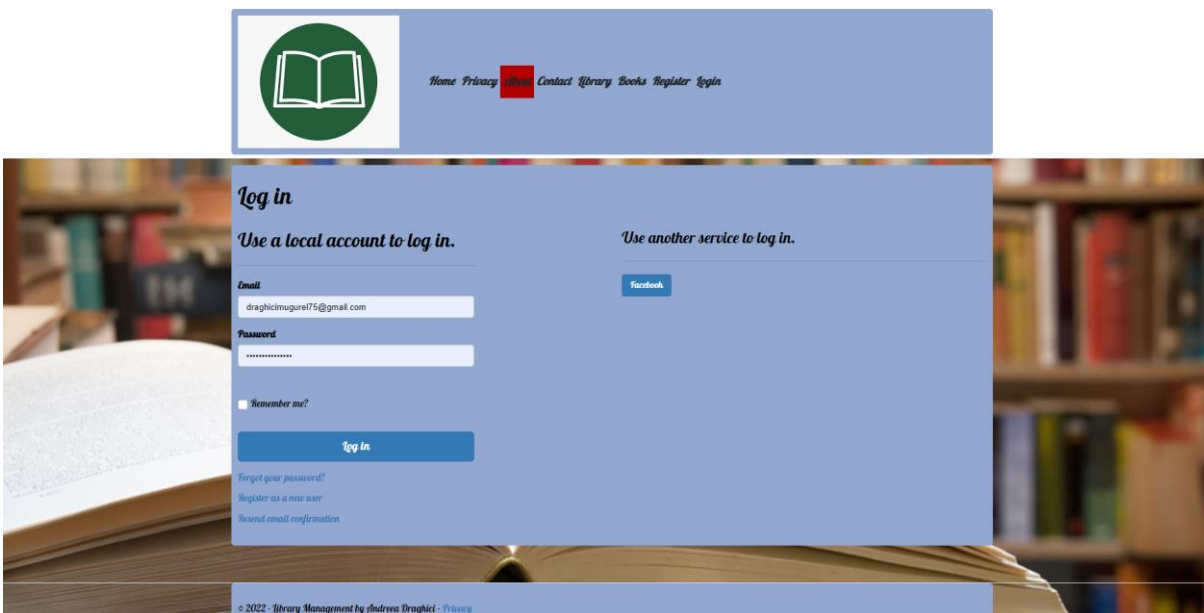
❖ **Autorizare pentru un utilizator cu rol de user:**

In cazul de fata acest tip de utilizator a incercat sa acceseze functionalitatea de update de pe pagina de Library, aceasta functionalitate fiind destinata doar unui utilizator cu rol de administrator.



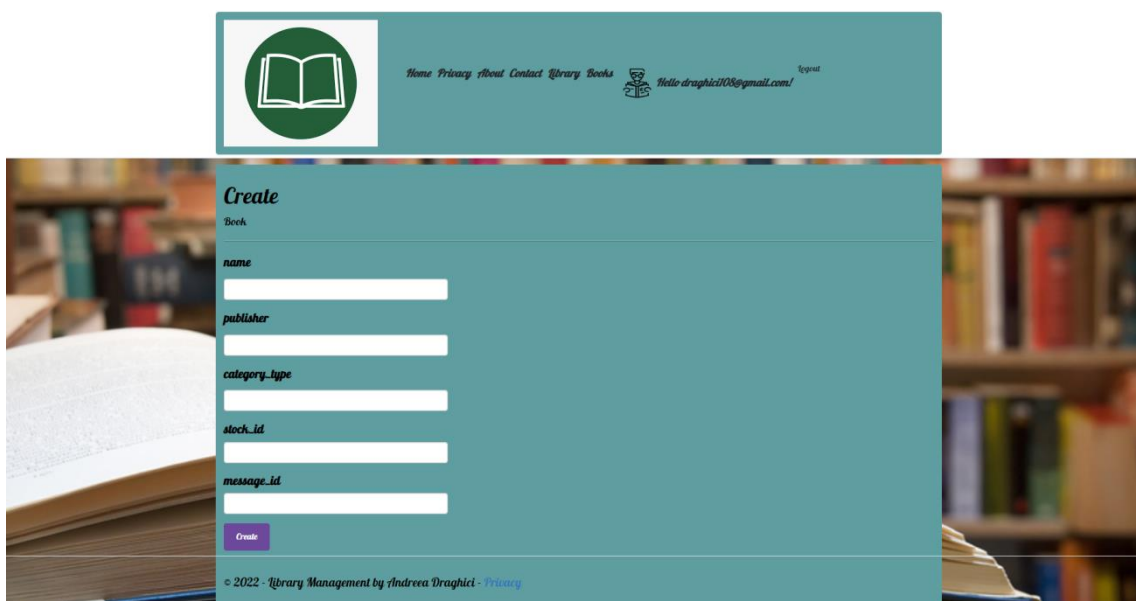
❖ Autorizare pentru un utilizator anonim:

Tipul de utilizator anonim va fi nevoit sa se conecteze la un cont valid pentru a accesa paginile aplicatiei. De exemplu, mai jos se poate observa ca acest utilizator a incercat sa acceseze pagina de About, nefiind conectat la cont, pagina nu este accesibila in acest caz.



❖ Autorizare pentru un utilizator cu rol de administrator:

Acest tip de utilizator poate avea acces si la functionalitatile de update/create/delete. De exemplu, mai jos se poate observa ca utilizatorul cu rol de administrator a accesat functionalitatea de Create de pe pagina Books.



9. Enumerati limitările și punctele slabe ale aplicației voastre, precum și îmbunătățirile / extinderile care s-ar putea realiza ulterior.

Personalizarea paginilor ar putea fi îmbunătățită pe viitor, la fel și partea de funcționalități, ar putea exista mai multe funcționalități, de exemplu:

- Biblioteca să aibă mai multe sucursale, fiecare având nume, număr unic și o adresă.
- Cititorii sau utilizatorii să poată împrumuta mai multe cărți de la diferite sucursale ale bibliotecii.
- Ar putea exista și funcționalitatea de integrare a unui chat unde toți utilizatorii să poată comunica între ei în timp real, unde să schimbe opinii despre cărțile citite, etc.

Din punct de vedere al limitărilor, aplicația este destul de limitată ca și funcționalități, deoarece utilizatorii cu rolul de user pot vizualiza lista de cărți disponibilă în bibliotecă, își pot alege ce carte să împrumute, pot accesa pagina de contact a aplicației, pot avea acces la lista de cărți care au fost accesate de alți utilizatori. Din acest motiv consider că un punct slab este faptul că aplicația nu deține foarte multe funcționalități, al doilea punct este legat de personalizarea paginilor, acestea ar putea avea un layout mai friendly pe viitor.

10. Nota pe care considerați că o meritați la laborator:

Nota pe care consider că o merit la laborator este nota 8, deoarece la fiecare prezentare au existat minusuri la aplicația implementată, pe parcurs de la o prezentare la alta în urma feedback-ului primit de la doamna profesoară, reușind să progrezhez cât de cât în atingerea și implementarea task-urilor.

Nota pe care considerați că o meritați la examen (documentul curent):

Nota pe care consider că o merit la examen este 8, deoarece consider că am atins cât de mult am putut/stiut fiecare subpunct în parte pentru a redacta documentul curent, dar în același timp este loc de mai bine și mai mult în detalierea fiecărui răspuns la întrebări.

Număr de ore alocat pentru elaborarea acestui document:

În cazul de față nu aș putea estima în ore progresul alocat pentru elaborarea documentului, deoarece nu am lucrat constant într-o anumită oră/zi și nu am ținut evidență la un număr de ore alocat.

Resurse folosite pentru elaborarea acestui document:

Resursele folosite sau de unde m-am inspirat pentru a elabora acest document, sunt cursurile de pe classroom, sau anumite resurse din bibliografia cursurilor, link-uri utile.

Preferati acest tip de evaluare sau un examen clasic? Care e mai util / relevant? Care necesita mai mult timp pentru pregatire?

Consider ca acest tip de evaluare este mult mai util, in cadrul acesta notiunile teoretice pot fi puse in evidenta mult mai practic pe aplicatia dezvoltata in cadrul laboratorului, decat daca ar exista un examen clasic. Din punctul meu de vedere, eu prefer acest tip de evaluare, deoarece elaborarea acestui document m-a ajutat sa inteleg si sa imi fixez intr-un mod mai clar si bine notiunile teoretice intr-un mod practic si flexibil.

Impresii generale despre laborator & curs si sugestii de imbunatatire:

Recunosc ca mi-a fost cam greu sa dezvolt o aplicatie intr-un nou limbaj, utilizand un framework nou. C# nu este un limbaj usor, ca sintaxa este destul de diferit (raportandu-ma la limbajele cu care lucrez, Python si Java), dar ca si o impresie obiectiva, reprezinta o experienta acumulata si bine venita in dezvoltarea cunostiintelor mele pe termen lung.

In cadrul laboratorului, cat si al cursului nu am nimic de reprosat, totul a decurs normal si intr-un mod destul de ok, feedback-ul primit in cadrul laboratoarelor a fost unul destul de constructiv si util, discutiile in cadrul cursului sau la final de curs au fost friendly si constructive.

In schimb, ca si o mica sugestie, pe viitor ar fi destul de util pentru cunostiintele studentilor sa fie inclusa si partea de testare automata a aplicatiei in cadrul laboratoarelor, astfel studentii sa poata invata intr-un mod practic despre unit testing/TDD/code refactoring si ar invata practic sa creeze cateva teste care sa asigure ca aplicatia functioneaza conform asteptarilor lor.