Universitatea Transilvania din Brașov
Facultatea de Matematică și Informatică

DOCUMENTAȚIA PROIECTULUI

Library Management

STUDENȚI
Oancea Andrei
Roman Stefania

Cuprins

1. Prezentarea proiectului

Proiectul **Library Management** își propune să gestioneze eficient informațiile despre autori și cărți într-o bibliotecă, inclusiv gestionarea utilizatorilor și a rolurilor acestora. Sistemul permite adăugarea, actualizarea și ștergerea datelor despre autori și cărți, precum și autentificarea și autorizarea utilizatorilor pe baza rolurilor atribuite.

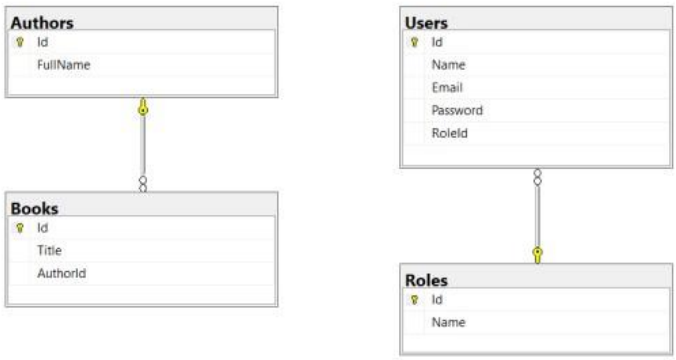
2. Tehnologii Folosite

Proiectul **Library Management** utilizează o serie de tehnologii moderne pentru a asigura funcționalitatea, securitatea și performanța sa.

- ASP.NET Core: Framework-ul principal pentru construirea aplicației web și API-urilor RESTful. ASP.NET Core este utilizat pentru gestionarea controlerelor, rutarea, middleware-urile și integrarea cu alte servicii.
- Entity Framework Core: ORM (Object-Relational Mapper) pentru interacțiunea cu baza de date SQL Server. Facilitează maparea obiectelor C# la tabelele din baza de date.
- SQL Server: Baza de date utilizată pentru stocarea informațiilor despre autori, cărți, utilizatori și roluri.
- Swagger: Pentru documentarea API-ului, facilitând testarea și explorarea endpoint-urilor.
- Dependency Injection: Pentru gestionarea instanțelor de servicii și repo-uri, asigurând decuplarea componentelor și facilitând testarea și menținerea codului.
- Microsoft.AspNetCore.Authorization: Pentru a restricționa accesul la anumite endpoint-uri în funcție de rolurile utilizatorilor.

3. Baza de date

3.1 Diagrama bazei de date



3.2 Relatiile dintre tabele

Tabelul Authors:

- Relație de tip One-to-Many (Unu-la-Mulți) cu **Tabelul Books**.
- Fiecare autor poate avea mai multe cărți.
- Cheia străină **AuthorId** din **Tabelul Books** face referire la **Tabelul Authors**.

Tabelul Users:

- Relație de tip Many-to-One (Mulți-la-Unu) cu **Tabelul Roles**.
- Fiecare utilizator este asociat cu exact un rol.
- Cheia străină **RoleId** din **Tabelul Users** face referire la **Tabelul Roles**.

3.3 Schema bazei de date

Tabelul Authors

- **Id**: Cheia primară pentru autor.
- **FullName**: Numele complet al autorului.

Tabelul Books

- **Id**: Cheia primară pentru carte.
- **Title**: Titlul cărții.
- **AuthorId**: Cheie străină care face referire la tabelul Authors.

Tabelul Users

- **Id**: Cheia primară pentru utilizator.
- **Name**: Numele utilizatorului.
- **Email**: Adresa de email a utilizatorului.
- **Password**: Parola utilizatorului (criptată).
- **RoleId**: Cheie străină care face referire la tabelul Roles.

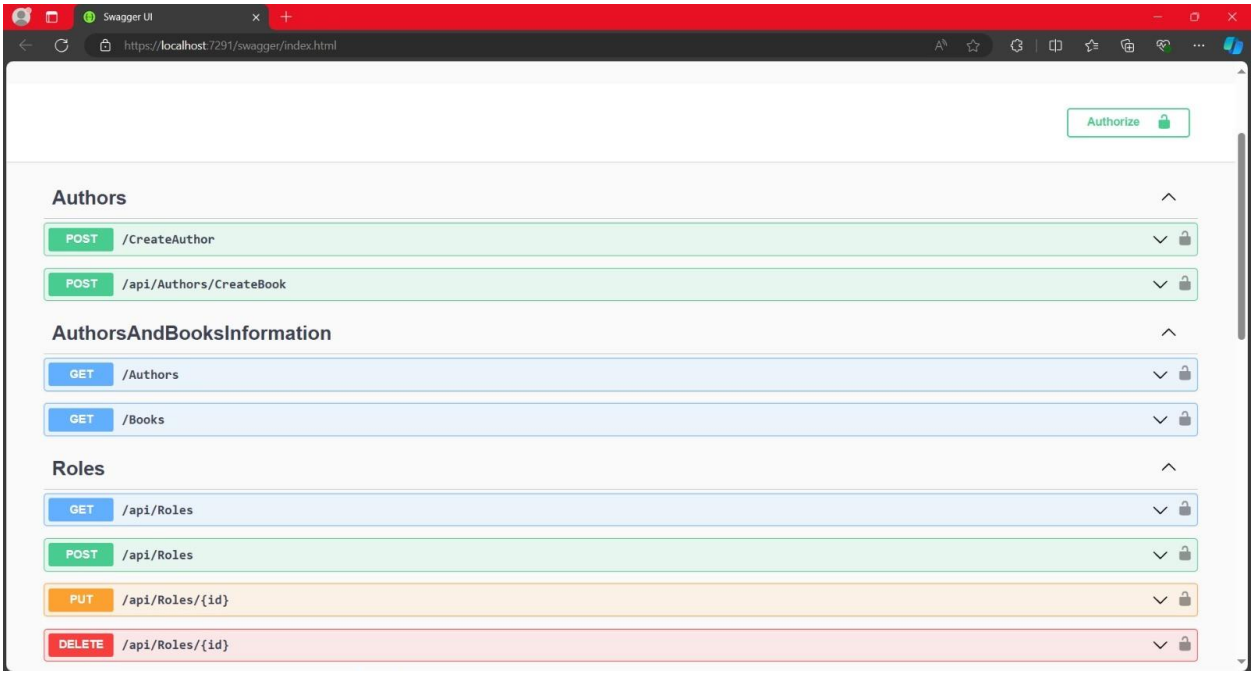
Tabelul Roles

- **Id**: Cheia primară pentru rol.
- **Name**: Numele rolului.

3.4 SWAGGER


Swagger UI
https://localhost:7291/swagger/index.html
Users
POST /register
POST /login
GET /api/Users
PUT /api/Users/{id}
DELETE /api/Users/{id}
Schemas
AuthorRequestDto
AuthorResponseDto
BookRequestDto
BookResponseDto


Swagger UI
https://localhost:7291/swagger/index.html
Schemas
AuthorRequestDto
AuthorResponseDto
BookRequestDto
BookResponseDto
LoginRequestDto
RoleRequestDto
RoleResponseDto
UserRequestDto
UserResponseDto



3.5 Operatii CRUD

`AuthorRepository`

Create (Creare)

- **Metoda**: `public int CreateAuthor(Author author)`
- **Descriere**: Această metodă este utilizată pentru a adăuga un nou autor în baza de date. Ea primește un obiect `Author` și adaugă acest autor în tabelul `Authors` al bazei de date.

Read (Citire)

- **Metoda**: `public List<Author> GetAuthors()`
- **Descriere**: Metoda `GetAuthors` returnează o listă de autori împreună cu detaliile cărților asociate fiecărui autor. Această operație permite obținerea informațiilor despre autori din baza de date.

`BookRepository`

Create (Creare)

- **Metoda**: `public int CreateBook(Book book)`
- **Descriere**: Metoda `CreateBook` este folosită pentru a adăuga o nouă carte în baza de date. Aceasta primește un obiect `Book`, verifică dacă autorul asociat există, și adaugă cartea în tabelul `Books`.

Read (Citire)

- **Metoda**: `public List<Book> GetBooks()`
- **Descriere**: Metoda `GetBooks` returnează o listă de cărți din baza de date, împreună cu detaliile autorilor lor. Această operație permite obținerea informațiilor despre cărți din sistem.

`RoleRepository`

Create (Creare)

- **Metoda**: `public void Create(Role role)`
- **Descriere**: Metoda `Create` este utilizată pentru a adăuga un nou rol în baza de date. Aceasta primește un obiect `Role` și verifică dacă rolul cu numele respectiv nu există deja în tabelul `Roles`.

Read (Citire)

- **Metoda**: `public List<Role> GetAll()`
- **Descriere**: Metoda `GetAll` returnează o listă cu toate rolurile existente în baza de date. Aceasta permite obținerea informațiilor despre toate rolurile disponibile.

`UserRepository`

Create (Creare)

- **Metoda**: `public void Register(User user)`
- **Descriere**: Metoda `Register` este folosită pentru a înregistra un nou utilizator în baza de date. Aceasta primește un obiect `User`, verifică dacă adresa de email nu există deja, și adaugă utilizatorul în tabelul `Users`.

Read (Citire)

- **Metoda**: `public User Get(int id)`
- **Descriere**: Metoda `Get` returnează un utilizator specificat după id-ul său din baza de date, împreună cu detaliile rolului asociat utilizatorului.

Update (Actualizare)

- **Metoda**: `public void Update(int id, User updatedUser)`
- **Descriere**: Metoda `Update` este folosită pentru a actualiza informațiile unui utilizator existent în baza de date. Aceasta primește id-ul utilizatorului și un obiect `User` actualizat, actualizând apoi detaliile utilizatorului în tabelul `Users`.

Delete (Ștergere)

- **Metoda**: `public void Delete(int id)`

- **Descriere**: Metoda `Delete` este utilizată pentru a șterge un utilizator specificat după id-ul său din baza de date. Aceasta elimină utilizatorul din tabelul `Users` și toate referințele asociate.

4 PREZENTAREA APLICATIEI

4.1 Tipuri de Utilizatori

1. **Utilizator cu Rolul "Author"**
   - **Accesibilitate**: Utilizatorii cu acest rol au drepturi limitate, fiind autorizați să creeze autori și cărți.
   - **Endpoint-uri Disponibile**:
     - POST `/CreateAuthor`: Permite crearea unui autor nou.
     - POST `/CreateBook`: Permite crearea unei cărți noi, specificând autorul existent.
2. **Utilizator cu Rolul "Admin"**
   - **Accesibilitate**: Utilizatorii cu acest rol au privilegii extinse și pot gestiona roluri și utilizatori în sistem.
   - **Endpoint-uri Disponibile**:
     - GET `/api/Roles`: Afișează toate rolurile disponibile în sistem.
     - POST `/api/Roles`: Permite crearea unui nou rol.
     - PUT `/api/Roles/{id}`: Permite actualizarea unui rol existent.
     - DELETE `/api/Roles/{id}`: Permite ștergerea unui rol existent.
     - POST `/register`: Permite înregistrarea unui utilizator nou în sistem.
     - POST `/login`: Permite autentificarea utilizatorului și obținerea unui token JWT pentru accesul ulterior la alte endpoint-uri protejate.

4.2 Autentificare și Autorizare

- **Autentificare**: Utilizatorii se autentifică folosind endpoint-ul `/login` pentru a obține un token JWT. Acest token este necesar pentru accesul la endpoint-uri protejate.
-
- **Autorizare**: Endpoint-urile sunt protejate cu ajutorul atributului `[Authorize(Roles = "...")]`, unde se specifică rolurile permise pentru acces. De exemplu, `[Authorize(Roles = "Admin")]` restricționează accesul la endpoint-uri doar pentru utilizatorii cu rolul "Admin"

5 Concluzie

Acest proiect ne-a oferit oportunitatea de a învăța și dezvolta abilități esențiale în dezvoltarea aplicațiilor moderne:

Lucrul cu EF Core și Baze de Date Relaționale:

Am învățat să configurăm și să gestionăm baze de date folosind Entity Framework Core. De asemenea, am dobandit cunostinte despre implementarea relațiilor între entități și gestionarea tranzacțiilor într-un mediu multi-utilizator.

Colaborarea și Managementul Proiectului:

Împărțirea clară a task-urilor și comunicarea eficientă au contribuit la succesul proiectului. Am învățat să gestionăm resursele și prioritățile într-un mod care să maximizeze eficiența echipei.