RAPORT 2 – INTERMEDIAR

Proiect III-Instrumente pentru dezvoltarea programelor

Student: Draghici Andreea-Maria
Grupa: CR4.S1 A
Anul de studiu: IV
Specializarea: Calculatoare Romana

LIBRARY MANAGEMENT

Cuprins

Introducere

Acest raport are scopul de a arata o proiectare minimala a proiectului Library Management. Aplicatia pe care o implementez este o aplicatie web MVC folosind Entity Framework Core si ASP.Net Core, o aplicatie folosita pentru gestionarea activitatii unei biblioteci / librarii in format electronic.

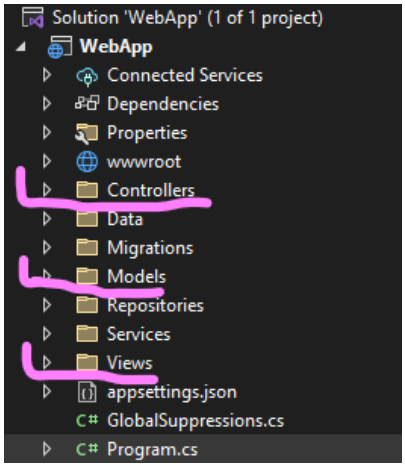
Proiectarea si implementarea minimala

MVC este un sablon architectural care separa aplicatia in 3 componente logice: Model,View si Controller. Fiecare componenta este contruita pentru a trata aspect diferite ale aplicatiei.

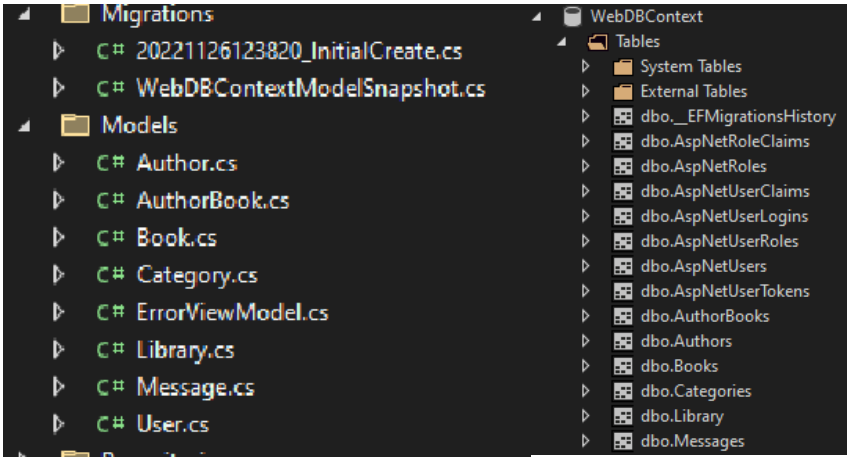
Model – Modelul trateaza cererile care vin de la view-uri si raspunde la instructiunile de la controller de a se actualiza.

View – Reprezinta partea de prezentare a datelor, preluand informatia din model si prezentand-o catre utilizator.

Controller – Reprezinta partea aplicatiei care se ocupa cu interactiunea cu utilizatorul. Aici, se citesc datele, se executa operatiile CRUD, apoi se trimite raspunsul catre view.



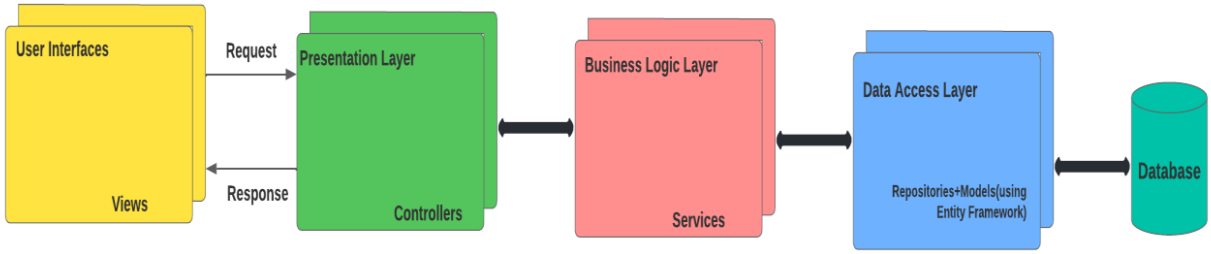
Pentru a crea baza de date si a actualiza tabelele am folosit strategia CodeFirst care presupune crearea modelelor cu proprietati specifice, inregistrarea acestora in clasa context, crearea unei migrari si executarea ei. Apoi baza de date va fi create, iar informatii pot fi introduse in anumite tabele.




```
using Microsoft.EntityFrameworkCore;
using WebApp.Models;

namespace WebApp.Data
{
    16 references
    public class WebDBContext : DbContext
    {
#pragma warning disable CS8618 // Non-nullable field must contain a non-null value when exiting const
        0 references
        public WebDBContext(DbContextOptions<WebDBContext> dbContextOptions) : base(dbContextOptions)
#pragma warning restore CS8618 // Non-nullable field must contain a non-null value when exiting const
        {

        }
        0 references
        public DbSet<Author> Authors { get; set; }
        0 references
        public DbSet<Book> Books { get; set; }
        0 references
        public DbSet<Category> Categories { get; set; }

        0 references
        public DbSet<AuthorBook> AuthorBooks { get; set; }
        0 references
        public DbSet<Library> Library { get; set; }
        0 references
        public DbSet<Message> Messages { get; set; }
        0 references
        public DbSet<User> Users { get; set; }
    }
}
```

Este 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 o sa incadrez 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.

Arhitectura sistemului:



Astfel se va respectat principiul Don't repeat yourself. Serviciile vot fi definite in clasa Program.cs si voi realiza injectia lor pe constructorul controllerelor in care voiam sa folosesc serviciile respective (Dependecy Injection).

Figura de mai jos indica modul cum sunt definite serviciile in clasa Program.cs

```
10    // Add services to the container.
11    builder.Services.AddControllersWithViews();
12    builder.Services.Add(new ServiceDescriptor(typeof(ILog), new ConsoleLogger()));
13
14    builder.Services.AddScoped<IAuthorBookRepository, AuthorBookRepository>();
15    builder.Services.AddScoped<IBookRepository, BookRepository>();
16    builder.Services.AddScoped<IAuthorRepository, AuthorRepository>();
17    builder.Services.AddScoped<ICategoryRepository, CategoryRepository>();
18    builder.Services.AddScoped<ILibraryRepository, LibraryRepository>();
19    builder.Services.AddScoped<IMessageRepository, MessageRepository>();
20    builder.Services.AddScoped<IUserRepository, UserRepository>();
21
22    builder.Services.AddScoped<AuthorBookService>();
23    builder.Services.AddScoped<BookService>();
24    builder.Services.AddScoped<AuthorService>();
25    builder.Services.AddScoped<CategoryService>();
26    builder.Services.AddScoped<LibraryService>();
27    builder.Services.AddScoped<MessageService>();
28    builder.Services.AddScoped<UserService>();
29
30    builder.Services.AddScoped<IRepositoryWrapper, RepositoryWrapper>();
```

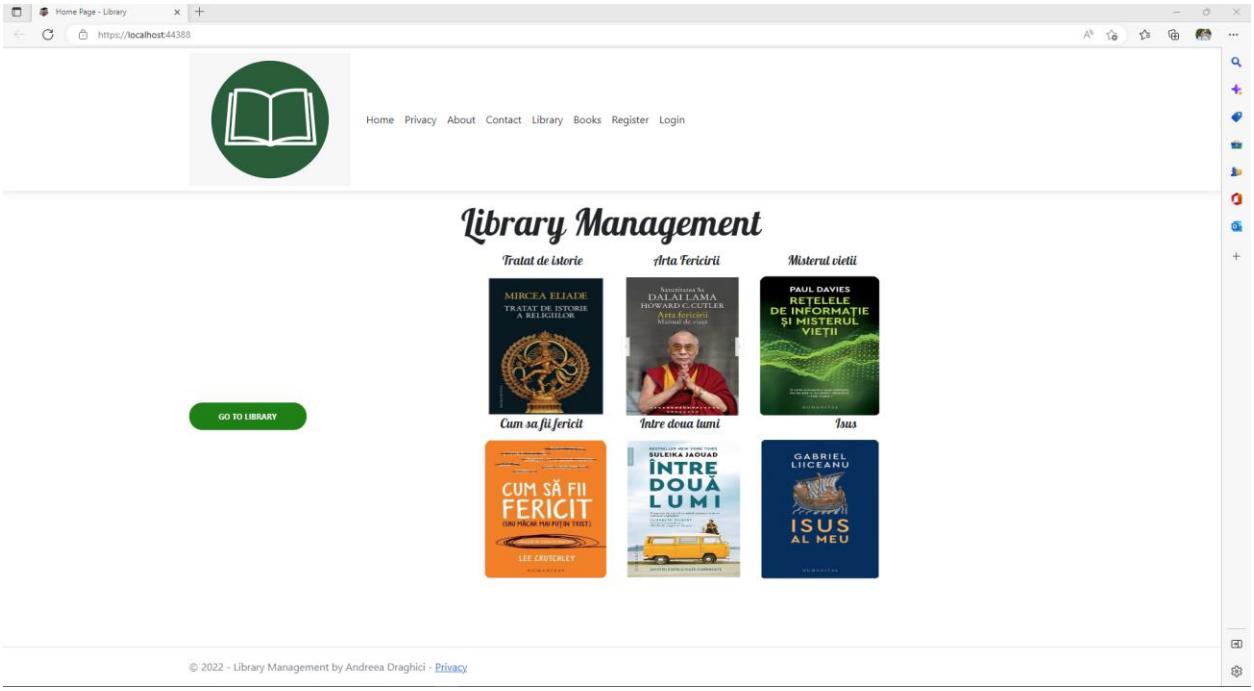
Figura atasata indica modul cum se realizeaza injectarea serviciilor pe constructorul controllerelor

```
private readonly AuthorService _authorService;

0 references
public AuthorsController(AuthorService authorService)
{
    _authorService=authorService;
}
```

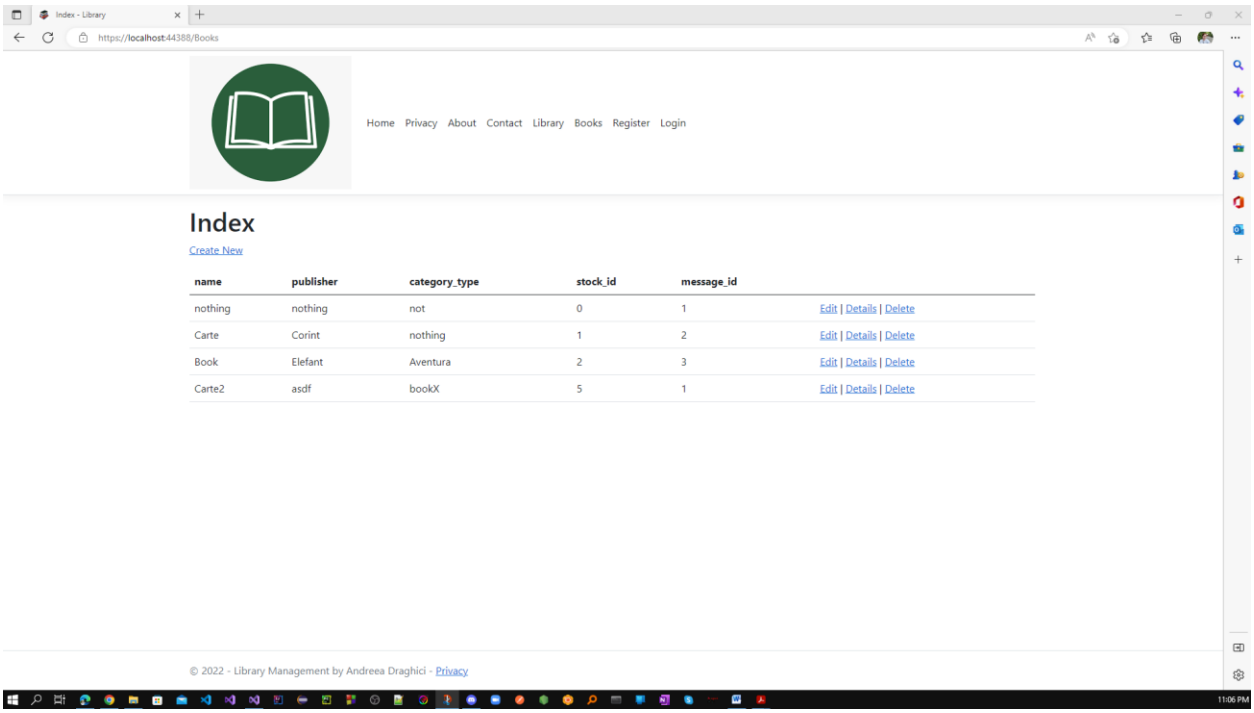
Prezentarea site-ului versiunea curenta:

Pe partea de layout am utilizat CSS , care este un standard pentru formatarea elementelor unui document si HTML care este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afisare intr-un browser.

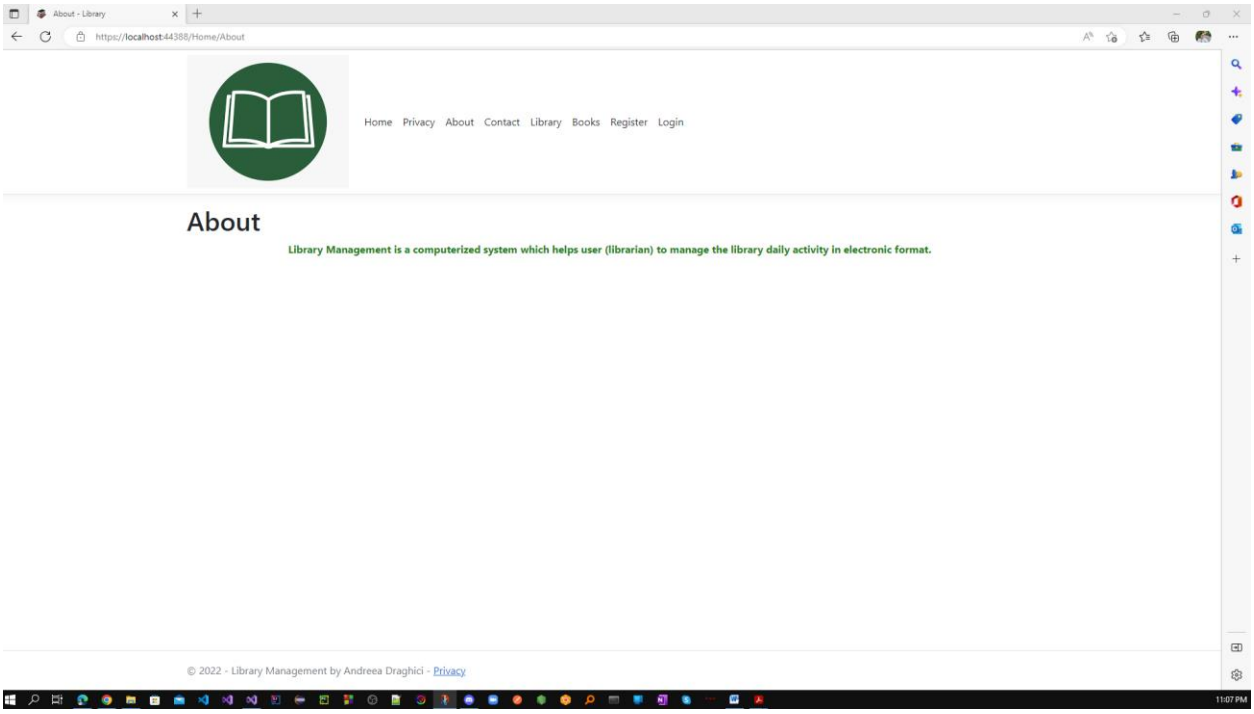
Pagina de home:



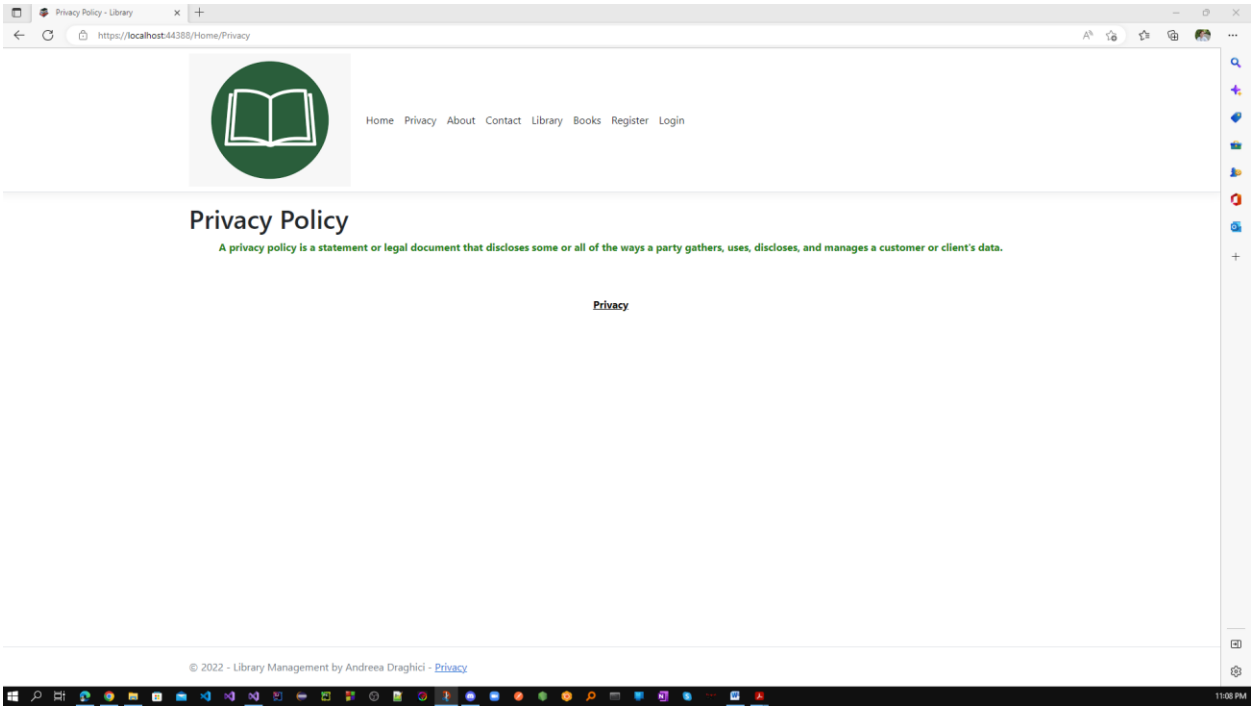
Pagina pentru Books:



Pagina de About:



Pagina de Privacy:



Concluzii

- **S-a incercat respectarea principiilor MVC si Repository Pattern.**
- **Momentan partea de Autentificare nu este implementata, nici partea de stilizare a paginilor nu este completa, si nici partea de functionalitati este finalizata complet.**
- **Am incercat sa adaug cat mai multe explicatii, exemple de cod si imagini cu interfata aplicatiei in stadiul actual pentru a intelege ce am implementat si care ar fi logica din spatele aplicatiei.**