**SVEUČILIŠTE U SPLITU**

**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Preddiplomski stručni studij Informacijska tehnologija

**PETAR MIHALJEVIĆ**

**Z A V R Š N I R A D**

**IZRADA WEB APLIKACIJE ZA PRODAJU KNJIGA**

Split, srpanj 2024.

**SVEUČILIŠTE U SPLITU**

**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Preddiplomski stručni studij Informacijska tehnologija

**Predmet:**Informacijski sustavi

**Z A V R Š N I R A D**

**Kandidat:** Petar Mihaljević

**Naslov rada:**Izrada web aplikacije za prodaju knjiga

**Mentor:**mr. sc. Karmen Klarin, viši predavač

Split, srpanj 2024.

[Sažetak 1](#_Toc170717995)

[Summary 1](#_Toc170717996)

[1. Uvod 2](#_Toc170717997)

[2. Korištene tehnologije i alati 3](#_Toc170717998)

[2.1 ASP.NET Core 3](#_Toc170717999)

[2.2 Entity Framework Core 3](#_Toc170718000)

[2.3 Microsoft SQL Server 4](#_Toc170718001)

[2.4 Identity Framework 4](#_Toc170718002)

[2.5 MVC Pattern 5](#_Toc170718003)

[2.6 Bootstrap 5](#_Toc170718004)

[2.7 Visual Studio 6](#_Toc170718005)

[3. Funkcionalnosti 7](#_Toc170718006)

[3.1 Opće funkcionalnosti 7](#_Toc170718007)

[3.2 Funkcionalnosti za korisnike 7](#_Toc170718008)

[3.3 Funkcionalnosti za administratore 8](#_Toc170718009)

[4. Implementacija 9](#_Toc170718010)

[4.1 Poslužiteljska aplikacija 9](#_Toc170718011)

[4.1.1 Struktura projekta 9](#_Toc170718012)

[4.1.2 Konfiguracija projekta 10](#_Toc170718013)

[4.2 Modeli 13](#_Toc170718014)

[4.3 Autentifikacija i autorizacija 15](#_Toc170718015)

[4.3.1 Identity Framework 16](#_Toc170718016)

[4.3.2 Stranice za Autentifikaciju 18](#_Toc170718017)

[4.4. Kontroleri 19](#_Toc170718018)

[4.4.1 AccountController 19](#_Toc170718019)

[4.4.2 AdminController 22](#_Toc170718020)

[4.4.3 BooksController 23](#_Toc170718021)

[4.4.4 CartController 25](#_Toc170718022)

[4.5 Pogledi 27](#_Toc170718023)

[4.5.1 \_Layout.cshtml 27](#_Toc170718024)

[4.5.2 Pogledi za knjige 27](#_Toc170718025)

[4.5.3 Pogledi za košaricu 30](#_Toc170718026)

[4.5.4 Pogledi za administraciju 33](#_Toc170718027)

[4.5.5 Pogledi za korisnički račun 35](#_Toc170718028)

[5. Zaključak 37](#_Toc170718029)

# Sažetak

Ovaj rad opisuje proces izrade web aplikacije za prodaju knjiga. Tehnologije koje su korištene su ASP.NET Core, Entity Framework Core, Microsoft SQL Server, Identity Framework, MVC Pattern i Bootstrap. Primarni problem na koji se odnosi ovaj projekt je nedostatak korisniku prilagođenih platformi za online kupovinu knjiga. Cilj je izgraditi sustav koji korisnicima omogućuje registraciju i prijavu, pretraživanje i pregled knjiga, njihovo dodavanje u košaricu, kupnju, kao i ocjenjivanje i recenziranje naslova. Administrator ima opciju upravljati katalogom knjiga, nadgledati narudžbe korisnika i pratiti korisničke račune.

**Ključne riječi**: ASP.NET, Microsoft SQL Server, online knjižnica, web aplikacija

# Summary

This paper describes the process of creating web applications for selling books. The technologies used are ASP.NET Core, Entity Framework Core, Microsoft SQL Server, Identity Framework, MVC Pattern and Bootstrap. The primary problem addressed by this project is the lack of user-friendly platforms for buying books online. The goal is to build a system that allows users to register and log in, search and view books, add them to the cart, purchase, as well as rate and review titles. Administrators have the privilege of managing the book catalog, monitor user orders, and monitor user accounts.

**Keywords**: ASP.NET, Microsoft SQL Server, online library, web application.

# 1. Uvod

Ovaj se rad usredotočuje na razvoj web aplikacije za online knjižnicu koja korisnicima omogućuje pretraživanje, pregledavanje, kupnju, ocjenjivanje i recenziranje knjiga.

Odluka o radu na ovom projektu proizlazi iz potrebe za korisnički prilagođenom (engl. *user-friendly*) platformom za upravljanje knjigama i omogućavanje jednostavnog pristupa istima. Razvoj pouzdanog i brzog sustava omogućilo je korištenje modernih tehnologija kao što su ASP.NET Core, Entity Framework Core, Microsoft SQL Server, Identity Framework, MVC Pattern i Bootstrap.

Glavni cilj je stvoriti funkcionalan sustav koji korisnicima omogućuje registraciju/prijavu, pretraživanje/pregledavanje/gledanje/kupnju knjiga kao i njihovo ocjenjivanje i recenziranje. Administrator ima mogućnost upravljanjem kataloga knjiga, narudžba i korisničkih računa.

U drugom poglavlju su opisani alati i tehnologije korištene u projektu. Treće poglavlje sadrži opis glavnih funkcionalnosti sustava, uključujući korisničke i administratorske funkcionalnosti. Četvrto poglavlje opisuje implementaciju glavnih direktorija web aplikacije.

# 2. Korištene tehnologije i alati

U ovom poglavlju su opisane tehnologije i alati korišteni u izradi web aplikacije za prodaju knjiga.

## 2.1 ASP.NET Core

ASP.NET Core je višeplatformski okvir otvorenog koda (engl. *open-source cross-platform framework*) za izgradnju modernih web aplikacija visoki performansi. ASP.NET Core omogućuje izradu aplikacija koje mogu raditi na Windows, macOS i Linux operativnim sustavima. Ovaj radni okvir nudi bogat skup alata i biblioteka za razvoj web aplikacija, uključujući podršku za MVC (engl. *Model-View-Controller*) obrazac. Osim toga uključuje i podršku za *dependency injection*, konfiguraciju i logiranje.

**Ključne karakteristike ASP.NET Corea su:**

* **Višeplatformska podrška - m**ogućnost rada na različitim operativnim sustavima.
* **Visoke performance - o**ptimiziran za vrhunske performanse.
* **Modularna arhitektura -** fleksibilan i proširiv putem NuGet paketa.
* **Ujedinjeni programski model - i**ntegrira MVC, Web API i Razor Pages u jedan programski model.

## 2.2 Entity Framework Core

Entity Framework Core (EF Core) je objektno-relacijsko mapiranje (engl. *Object Relational Mapping*, ORM) za .NET, koji omogućuje programerima rad s bazama podataka koristeći .NET objekte. EF Core podržava različite baze podataka, uključujući Microsoft SQL Server, SQLite, PostgreSQL i druge. Omogućuje rad s podacima koristeći visoko apstraktne objekte umjesto pisanjem SQL upita, čime se ubrzava razvoj i održavanje aplikacija.

**Ključne karakteristike EF Corea su:**

* **Višeplatformska podrška - m**ogućnost rada na različitim operativnim sustavima.
* **Modeliranje - o**mogućuje definiranje modela podataka koristeći C# klase.
* **Praćenje promjena - a**utomatsko praćenje promjena nad entitetima.
* **Migracije - u**pravljanje promjenama u bazi podataka kroz migracije.
* **Upiti - p**isanje upita koristeći LINQ (Language Integrated Query).

## 2.3 Microsoft SQL Server

Microsoft SQL Server je relacijski sustav za upravljanje bazama podataka (engl. *Relation Database Management System*, RDBMS) razvijen od strane Microsofta. SQL Server je poznat po svojoj pouzdanosti, sigurnosti i skalabilnosti, što ga čini popularnim izborom za poslovne aplikacije. SQL Server nudi podršku za Transact-SQL, skup programskih ekstenzija za SQL koji omogućuje složene upite i manipulaciju podacima.

**Ključne karakteristike Microsoft SQL Servera su:**

* **Visoke performance - o**ptimiziran za brze upite i rukovanje velikim količinama podataka.
* **Sigurnost - n**apredne sigurnosne osobine, uključujući enkripciju i autentifikaciju.
* **Skalabilnost - m**ogućnost skaliranja baze podataka kako bi se podržale rastuće potrebe aplikacije.
* **Servisi za integraciju - a**lati za integraciju podataka iz različitih izvora.
* **Servisi za izradu izvješća - a**lati za izradu izvješća i analiza podataka.

## 2.4 Identity Framework

Identity Framework je dio ASP.NET Core koji pruža osnovne funkcionalnosti za autentifikaciju i autorizaciju korisnika. Omogućuje jednostavnu implementaciju registracije, prijave i upravljanja korisničkim računima. Identity Framework podržava različite načine autentifikacije, uključujući korisničko ime/lozinku, OAuth, OpenID Connect. Također, Identity Framework podržava i vanjske pružatelje identiteta kao što su Google, Facebook i Microsoft.

**Ključne karakteristike Identity Frameworka su:**

* **Upravljanje korisnicima - u**pravljanje korisničkim računima, uključujući registraciju, prijavu i resetiranje lozinke.
* **Uloge -** podrška za uloge za autorizaciju korisnika.
* **Dvofaktorska autentifikacija - d**vofaktorska autentifikacija za dodatnu sigurnost.
* **Vanjski pružatelji identiteta – podrška za** vanjske pružatelje identiteta.

## 2.5 MVC Pattern

MVC je obrazac koji razdvaja aplikaciju u tri glavne komponente: Model*,* Pogled (engl. View)iKontroler (engl. Controller). Ovaj obrazac omogućuje jasno razdvajanje odgovornosti, što poboljšava održavanje i testiranje aplikacije. ASP.NET Core podržava MVC obrazac, što olakšava izradu skalabilnih i održivih web aplikacija.

**Ključne karakteristike MVC Patterna su:**

* **Odvajanje odgovornosti - j**asna podjela između modela, pogleda i kontrolera.
* **Testabilnost - p**oboljšana mogućnost testiranja pojedinih komponenti aplikacije.
* **Skalabilnost - l**aka skalabilnost aplikacije zahvaljujući modularnom dizajnu.
* **Održavanje - o**lakšano održavanje i proširenje aplikacije.

## 2.6 Bootstrap

Bootstrap je najpopularniji radni okvir za izradu korisničkog sučelja za izradu responzivnih i mobilno-prilagođenih web stranica. Razvijen od strane Twittera, Bootstrap nudi bogat skup unaprijed definiranih CSS i JavaScript komponenti koje omogućuju brzi razvoj modernih i dosljednih korisničkih sučelja. Korištenje Bootstrapa smanjuje vrijeme potrebno za razvoj prilagodljivih i estetski ugodnih web stranica.

Ključne karakteristike Bootstrapa su:

* Responzivni dizajn - automatsko prilagođavanje izgleda stranice različitim veličinama ekrana, uključujući mobilne uređaje, tablete i desktop računala.
* Unaprijed definirane komponente - veliki broj unaprijed definiranih CSS i JavaScript komponenti poput navigacijskih traka, botuna i obrazaca.
* Kompatibilnost s preglednicima - osigurana kompatibilnost s najnovijim verzijama svih popularnih preglednika.
* Podrška zajednice - aktivna zajednica koja pruža podršku i resurse za razvoj.

## 2.7 Visual Studio

Visual Studio je integrirano razvojno okruženje (engl. *Integrated development environment,* IDE) koje je razvila tvrtka Microsoft. Ovaj moćan alat podržava razvoj različitih vrsta aplikacija, uključujući web, mobilne i desktop aplikacije. Visual Studio je najčešće korišten alat za razvoj aplikacija na .NET platformi, uključujući ASP.NET Core aplikacije.

**Ključne karakteristike programa Visual Studio su:**

* **Razvoj aplikacija - o**mogućuje razvoj aplikacija na različitim jezicima kao što su C#, VB.NET, F#, JavaScript, TypeScript, Python i mnogi drugi.
* **IntelliSense - n**apredna specifičnost automatskog dovršavanja kôda koja olakšava pisanje kôda, smanjuje broj pogrešaka i ubrzava razvoj.
* **Otklanjanje pogrešaka (engl. *debugging*) - s**nažni alati koji omogućuju otkrivanje i ispravljanje pogrešaka u kôdu.
* **Integrirani Git - i**ntegrirana podrška za verzioniranje kôda putem Gita, omogućujući jednostavno praćenje promjena i suradnju s drugim programerima.
* **Alati za testiranje - p**odrška za pisanje i pokretanje testova, uključujući jedinično testiranje, integracijsko testiranje i automatizirano testiranje.
* **Alati za dizajn - g**rafički alati za izradu korisničkog sučelja, baza podataka i arhitekture aplikacije.
* **Proširenja - v**elik broj dodataka i proširenja dostupnih putem Visual Studio Marketplacea koji omogućuju prilagodbu i proširenje funkcionalnosti IDE-a.

# 3. Funkcionalnosti

U ovom poglavlju su nabrojane i opisane sve glavne funkcionalnosti web aplikacije kroz sve tri korisničke uloge (gost, registrirani korisnik i administrator.)

## 3.1 Opće funkcionalnosti

Web aplikacija za online knjižnicu pruža nekoliko osnovnih funkcionalnosti koje su dostupne svim korisnicima. Ove funkcionalnosti omogućavaju korisnicima jednostavan pristup informacijama o knjigama i lako korištenje sustava:

* Registracija i prijava korisnika - aplikacija omogućava korisnicima da se registriraju i prijave u sustav. Registracija uključuje unos osnovnih informacija kao što su korisničko ime, email adresa i lozinka. Nakon registracije, korisnici mogu koristiti svoje podatke za prijavu. Prijava autentificira korisnike i omogućava im pristup dodatnim funkcionalnostima aplikacije.
* Pretraga knjiga - korisnici mogu pretraživati knjige koristeći razne kriterije kao što su naziv knjige, autora ili kategorije. Pretraga vraća rezultate koji zadovoljavaju navedene kriterije, olakšavajući korisnicima pronalazak željene knjige.
* Pregled detalja knjige - svaka knjiga u sustavu ima stranicu s detaljnim informacijama. Korisnici mogu pregledavati informacije kao što su naslov, autor, opis, cijena, ocjene i recenzije drugih korisnika, i tako dalje. Ova funkcionalnost omogućava korisnicima da donesu odluke o kupovini knjiga.

## 3.2 Funkcionalnosti za korisnike

Korisnici aplikacije imaju pristup nekoliko specifičnih funkcionalnosti koje im omogućavaju interakciju s knjigama na različite načine:

* Dodavanje knjiga u košaricu - korisnici mogu dodavati knjige u svoju košaricu kako bi ih kasnije kupili. Svaka knjiga na stranici s detaljima ima opciju "Dodaj u košaricu". Korisnici mogu pregledavati sadržaj svoje košarice prije dovršetka kupovine, te ukloniti knjige iz košarice.
* Kupovina knjiga - nakon dodavanja knjiga u košaricu, korisnici mogu započeti postupak kupovine. Proces kupovine uključuje unos podataka za dostavu i plaćanje. Nakon što su svi potrebni podaci uneseni, korisnici mogu potvrditi narudžbu. Sustav generira potvrdu o narudžbi i korisnici dobivaju informacije o statusu narudžbe.
* Ocjenjivanje i recenziranje knjiga - korisnici mogu ocjenjivati i recenzirati knjige koje su kupili. Svaka knjiga ima opciju za ostavljanje ocjene i pisanih recenzija. Ova funkcionalnost omogućava korisnicima da podijele svoje mišljenje o knjigama i pomažu drugim korisnicima pri odabiru knjiga.

## 3.3 Funkcionalnosti za administratore

Administratori aplikacije imaju pristup posebnim funkcionalnostima koje im omogućavaju upravljanje aplikacijom i sadržajem:

* Upravljanje katalogom knjiga - administratori mogu dodavati nove knjige u katalog, uređivati postojeće knjige i brisati knjige koje više nisu dostupne. Za svaku knjigu mogu se unijeti detalji kao što su naslov, autor, opis, cijena, dostupnost, kategorija i drugi. Ova funkcionalnost omogućava administratorima da održavaju ažuriran i točan katalog knjiga.
* Pregled i upravljanje narudžbama - administratori mogu pregledavati sve narudžbe koje su korisnici napravili. Mogu vidjeti detalje svake narudžbe, uključujući informacije o korisniku, naručenim knjigama, statusu narudžbe i podacima za dostavu. Administratori mogu ažurirati status narudžbi.
* Administratori mogu pregledavati i upravljati korisničkim računima. To uključuje pregled i izmjenu osnovnih informacija o korisnicima.

# 4. Implementacija

U ovom poglavlju je opisana implementacija poslužiteljske aplikacije, modela, kontrolera, pogleda, te autentifikacije i autorizacije za web aplikaciju.

## 4.1 Poslužiteljska aplikacija

U ovom potpoglavlju je opisana struktura projekta, te implementacija poslužiteljske aplikacije, uključujući konfiguraciju projekta kroz program.cs i appsettings.json.

### 4.1.1 Struktura projekta

Projekt je organiziran u nekoliko ključnih direktorija koji zajedno čine ASP.NET Core aplikaciju. Svaki od ovih direktorija ima svoju specifičnu ulogu u aplikaciji.

Direktorij Controllers sadrži kontrolere koji upravljaju tijekovima aplikacije. Kontroleri obrađuju dolazne HTTP zahtjeve, pozivaju odgovarajuće metode modela i vraćaju odgovarajuće poglede korisniku. Kontroleri su podijeljeni na:

* AccountController.cs - kontroler za upravljanje korisničkim računima.
* AdminController.cs - kontroler za administracijske funkcionalnosti kao što su upravljanje knjigama, narudžbama i korisnicima.
* BooksController.cs - kontroler za pretragu i prikaz knjiga.
* CartController.cs - kontroler za upravljanje košaricom i kupovinom.

Direktorij Models sadrži sve modele koji predstavljaju podatke aplikacije. Svaki model odgovara tablici u bazi podataka i koristi se za definiranje strukture podataka. Modeli su podijeljeni na:

* Book.cs – model koji predstavlja knjigu s atributima kao što su naslov, autor, cijena, itd.
* Author.cs - model koji predstavlja autora.
* Category.cs - model koji predstavlja kategoriju knjiga.
* Order.cs - model koji predstavlja narudžbu.
* OrderDetail.cs - model koji predstavlja detalje narudžbe.
* Review.cs - model koji predstavlja recenziju knjige.
* ApplicationDbContext.cs - DbContext klasa koja upravlja vezom s bazom podataka i omogućava interakciju s modelima.

Direktorij Views sadrži sve poglede aplikacije. Pogledi su odgovorni za prikaz korisničkog sučelja. Pogledi su podijeljeni na:

* Account - pogledi za registraciju, prijavu, odjavu i upravljanje korisničkim računom.
* Admin - pogledi za administracijske funkcionalnosti.
* Books - pogledi za prikaz i pretragu knjiga.
* Cart - pogledi za upravljanje košaricom i procesom kupovine.
* Shared - zajednički pogledi kao što je \_Layout.cshtml

Direktorij Migrations sadrži migracije baze podataka. Migracije omogućavaju praćenje promjena u modelima podataka i njihovo primjenjivanje na bazu podataka. Svaka migracija predstavlja skup promjena koje se primjenjuju na bazu podataka kako bi se uskladila s trenutnim modelima.

Direktorij wwwroot sadrži statičke datoteke kao što su na primjer slike knjiga koje se poslužuju klijentima.

Direktorij Areas koristi se za organiziranje funkcionalnosti aplikacije u zasebne module. U ovom projektu, Areas direktorij se koristi za organizaciju funkcionalnosti povezane s identitetom (autentifikacija i autorizacija).

### 4.1.2 Konfiguracija projekta

Konfiguracija ASP.NET Core aplikacije ključna je za pravilno funkcioniranje svih komponenti sustava. Ova konfiguracija se provodi u dvije glavne datoteke: Program.cs i appsettings.json.

Datoteka Program.cs je početna točka ASP.NET Core aplikacije. U ovoj datoteci definiraju se servisi koji se koriste u aplikaciji, konfiguriraju se middleware komponente i postavlja se osnovna konfiguracija aplikacije (Ispis 1).

var builder = WebApplication.CreateBuilder(args);

// Session for Cart

builder.Services.AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromMinutes(60);

options.Cookie.HttpOnly = true;

options.Cookie.IsEssential = true;

});

// Add services to the container.

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Register Identity services and add roles

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)

.AddRoles<IdentityRole>() // Add support for roles

.AddEntityFrameworkStores<ApplicationDbContext>();

// Add other services like controllers, Razor Pages, etc.

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Create roles on application startup

CreateRoles(app.Services).Wait();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You might want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseSession();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapRazorPages();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

async Task CreateRoles(IServiceProvider serviceProvider)

{

using (var scope = serviceProvider.CreateScope())

{

var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>();

string[] roleNames = { "Admin", "User" };

foreach (var roleName in roleNames)

{

var roleExist = await roleManager.RoleExistsAsync(roleName);

if (!roleExist)

{

// Create the roles and seed them to the database

await roleManager.CreateAsync(new IdentityRole(roleName));

}

}

}

}

**Ispis 1**: Direktorij Program.cs

Objašnjenje ispisa 1:

* Konfiguracija sesije - definira se timeout i postavke kolačića za sesiju.
* Dodavanje servisa za bazu podataka - konfigurira se ApplicationDbContext da koristi SQL Server.
* Identity - registriraju se Identity servisi za upravljanje korisničkim računima i ulogama.
* Kontroleri - dodaju se servisi za kontrolere.
* Middleware - konfigurira se middleware za rukovanje pogreškama, HTTPS preusmjeravanje, posluživanje statičkih datoteka, sesije, autentifikaciju i autorizaciju.
* Kreiranje uloga - prilikom pokretanja aplikacije, kreiraju se osnovne uloge ako već ne postoje.

Datoteka appsettings.json sadrži konfiguracijske postavke za aplikaciju, uključujući postavke za bazu podataka, logiranje i druge opcije (Ispis 2).

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=MyBookshopDb;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

}

**Ispis 2**: Direktorij appsettings.json

Objašnjenje ispisa 2:

* Logging - postavke za logiranje koje definiraju razine logiranja za različite dijelove aplikacije.
* AllowedHosts - definira koji hostovi su dozvoljeni za pristup aplikaciji.
* ConnectionStrings -sadrži vezu na bazu podataka. U ovom slučaju, koristi se lokalna SQL Server baza podataka nazvana MyBookshopDb.

## 4.2 Modeli

Model Book predstavlja knjigu i sadrži atribute i relacije prikazane u ispisu 3:

public class Book

{

public int BookId { get; set; }

public string Title { get; set; }

public decimal Price { get; set; }

public string CoverImageUrl { get; set; }

public int AuthorId { get; set; }

public Author Author { get; set; }

public int CategoryId { get; set; }

public Category Category { get; set; }

public int Stock { get; set; }

public string Description { get; set; }

public string ISBN { get; set; }

public int PublicationYear { get; set; }

public string Language { get; set; }

public float Weight { get; set; }

public ICollection<Review> Reviews { get; set; }}

**Ispis 3**: Direktorij *Book.cs*

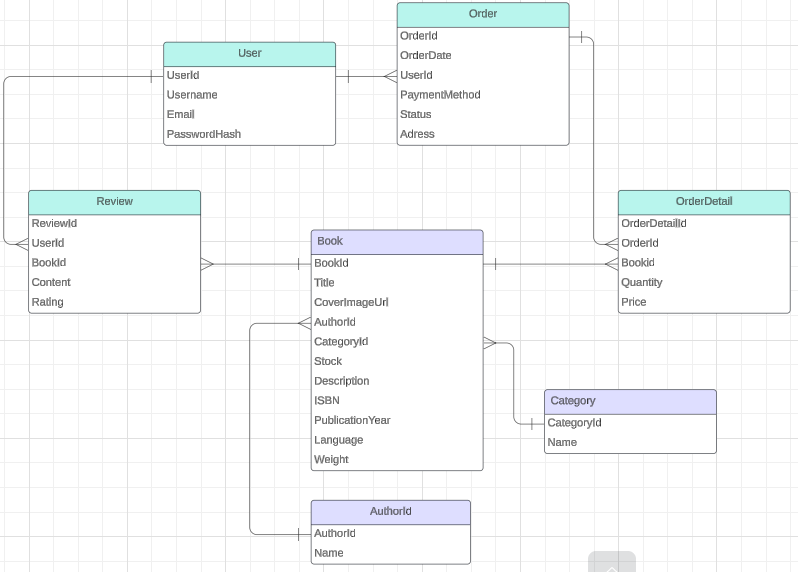
Opis atributa i relacija modela Book prikazanih u ispisu 3:

* BookId**:** Primarni ključ.
* Title: Naziv knjige.
* Price: Cijena knjige.
* CoverImageUrl: URL slike naslovnice.
* AuthorId: Strani ključ koji referencira autora.
* CategoryId: Strani ključ koji referencira kategoriju.
* Stock: Količina knjiga na zalihi.
* Description: Opis knjige.
* ISBN: Međunarodni standardni broj knjige.
* PublicationYear: Godina izdavanja.
* Language: Jezik knjige.
* Weight: Težina knjige.
* Reviews: Kolekcija recenzija vezanih za knjigu.

Uz model Book, aplikacija koristi i sljedeće modele:

* Author.cs - model koji predstavlja autora.
* Category.cs - model koji predstavlja kategoriju.
* Order.cs - model koji predstavlja narudžbu.
* OrderDetail.cs - model koji predstavlja detalje narudžbe.
* Review.cs - model koji predstavlja recenziju.

Svaki od ovih modela ima svoje atribute i relacije koje definiraju kako su podaci povezani i kako se njima upravlja unutar aplikacije (slika 1).



**Slika 1**: ER Dijagram relacija modela

Datoteka ApplicationDbContext konfigurira kontekst baze podataka i definira DbSetove za svaki model (ispis 4).

public class ApplicationDbContext : IdentityDbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Book> Books { get; set; }

public DbSet<Author> Authors { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Order> Orders { get; set; }

public DbSet<Review> Reviews { get; set; }

public DbSet<OrderDetail> OrderDetails { get; set; }

}

**Ispis 4**: Direktorij ApplicationDbContext.cs

## 4.3 Autentifikacija i autorizacija

U ovom potpoglavlju je opisana implementacija autentifikacije i autorizacije unutar web aplikacije za prodaju knjiga.

### 4.3.1 Identity Framework

ASP.NET Core Identity je sustav koji omogućava jednostavno upravljanje korisničkim računima, autentifikacijom i autorizacijom unutar aplikacije. U ovom projektu korišten je Identity Framework za upravljanje korisnicima i ulogama.

IdentityUser klasa predstavlja korisnika u Identity Frameworku. Korištenje ove klase omogućava upravljanje osnovnim korisničkim informacijama kao što su korisničko ime, email, lozinka i uloga.

U datoteci Program.cs, Identity Framework je konfiguriran na način prikazan na ispisu 5.

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)

.AddRoles<IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>();

**Ispis 5**: Konfiguracija Identity Frameworka

Konfiguracija Identity Frameworka prikazana na ispisu 5 uključuje:

* AddDefaultIdentity - dodaje osnovne servise za autentifikaciju i autorizaciju koristeći IdentityUser klasu.
* AddRoles - dodaje podršku za uloge (IdentityRole).
* AddEntityFrameworkStores - konfigurira Identity da koristi ApplicationDbContext za pohranu podataka.

Registracija korisnika implementirana je u RegisterModel klasi unutar Areas/Identity/Pages/Account/Register.cshtml.cs datoteke. Kada korisnik unese svoje podatke i pošalje obrazac za registraciju, podaci se validiraju i novi korisnik se stvara s pomoću UserManager servisa (ispis 6).

public async Task<IActionResult> OnPostAsync(string returnUrl = null)

{

returnUrl ??= Url.Content("~/");

ExternalLogins = (await \_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();

if (ModelState.IsValid)

{

var user = CreateUser();

var result = await \_userManager.CreateAsync(user, Input.Password);

if (result.Succeeded)

{

\_logger.LogInformation("User created a new account with password.");

var userId = await \_userManager.GetUserIdAsync(user);

var code = await \_userManager.GenerateEmailConfirmationTokenAsync(user);

code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));

var callbackUrl = Url.Page(

"/Account/ConfirmEmail",

pageHandler: null,

values: new { area = "Identity", userId = userId, code = code, returnUrl = returnUrl },

protocol: Request.Scheme);

await \_emailSender.SendEmailAsync(Input.Email, "Confirm your email",

$"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

if (\_userManager.Options.SignIn.RequireConfirmedAccount)

{

return RedirectToPage("RegisterConfirmation", new { email = Input.Email, returnUrl = returnUrl });

}

else

{

await \_signInManager.SignInAsync(user, isPersistent: false);

return LocalRedirect(returnUrl);

}

}

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

// If we got this far, something failed, redisplay form

return Page();

}

**Ispis 6**: Implementacija registracije koristnika unutar direktorija RegisterModel

Uloge u aplikaciji omogućuju kontrolu pristupa različitim dijelovima aplikacije na temelju korisničke uloge. U ovom projektu definirane su dvije osnovne uloge: Admin i User. Uloge se definiraju i dodaju u bazu podataka prilikom pokretanja aplikacije u Program.cs datoteci (Ispis 1).

Autorizacija na temelju uloga koristi se za zaštitu određenih dijelova aplikacije. Na primjer, samo korisnici s Admin ulogom mogu pristupiti administracijskim kontrolerima. Ovo se postiže korištenjem Authorize atributa (ispis 7).

[Authorize(Roles = "Admin")]

public class AdminController : Controller

{

// Metode unutar za AdminControllera

}

**Ispis 7**: Primjer korištenja Authorize atributa

Ovaj pristup osigurava da samo autorizirani korisnici mogu pristupiti određenim dijelovima aplikacije i izvršavati specifične radnje.

### 4.3.2 Stranice za Autentifikaciju

ASP.NET Core Identity pruža unaprijed definirane stranice za autentifikaciju i autorizaciju, ali se mogu prilagoditi prema specifičnim potrebama aplikacije. U ovom projektu implementirane su stranice za registraciju, prijavu, odjavu i uređivanje profila korisnika.

* Register.cshtml -stranica za registraciju korisnika omogućava novim korisnicima kreiranje računa. Korisnik unosi svoje korisničko ime, email adresu, lozinku i potvrdu lozinke.
* Login.cshtml - stranica za prijavu korisnika omogućava postojećim korisnicima pristup sustavu unosom svog emaila i lozinke.
* Logout.cshtml -stranica za odjavu korisnika omogućava korisnicima sigurno napuštanje aplikacije.
* EditProfile.cshtml - stranica za uređivanje profila korisnika omogućava korisnicima promjenu njihovih osnovnih podataka tako što sadrži tri poveznice: poveznicu za promjenu korisničkog imena, poveznicu za promjenu lozinke i poveznicu za promjenu email adrese (ispis 8).

@{

ViewData["Title"] = "Edit Profile";

}

<div class="container mt-4">

<h1 class="text-center">@ViewData["Title"]</h1>

<div class="row justify-content-center">

<div class="col-md-6">

<div class="list-group my-4">

<**a** **asp-action**="ChangeUsername" class="list-group-item list-group-item-action d-flex justify-content-between align-items-center">

Change Username

<span class="badge bg-primary rounded-pill">&raquo;</span>

</**a**>

<**a** **asp-action**="ChangeEmail" class="list-group-item list-group-item-action d-flex justify-content-between align-items-center">

Change Email

<span class="badge bg-primary rounded-pill">&raquo;</span>

</**a**>

<**a** **asp-action**="ChangePassword" class="list-group-item list-group-item-action d-flex justify-content-between align-items-center">

Change Password

<span class="badge bg-primary rounded-pill">&raquo;</span>

</**a**>

</div>

</div>

</div>

</div>

<style>

.list-group-item-action:hover {

background-color: #f8f9fa;

}

</style>

**Ispis 8**: Direktorij EditProfile.cshtml

## 4.4. Kontroleri

U ovom potpoglavlju je opisana implementacija kontrolera web aplikacije za prodaju knjiga.

### 4.4.1 AccountController

AccountController je odgovoran za upravljanje korisničkim računom i obuhvaća nekoliko ključnih metoda koje omogućuju korisnicima pregled povijesti narudžbi, detalja narudžbi, uređivanje profila, promjenu korisničkog imena, emaila i lozinke.

Metoda OrderHistory omogućava korisnicima pregled njihove povijesti narudžbi. Preuzima narudžbe iz baze podataka za trenutnog korisnika i prosljeđuje ih pogledu. Korisnici također imaju mogućnost sortiranja narudžbi prema datumu ili statusu. Implementacija metode OrderHistory unutar AccountControllera prikazana je na ispisu 9.

public async Task<IActionResult> OrderHistory(string sortOrder)

{

var user = await \_userManager.GetUserAsync(User);

if (user == null)

{

return View("Error");

}

IQueryable<Order> ordersQuery = \_context.Orders

.Where(o => o.UserId == user.Id)

.Include(o => o.OrderDetails)

.ThenInclude(od => od.Book);

ViewData["DateSortParam"] = string.IsNullOrEmpty(sortOrder) ? "date\_desc" : "";

ViewData["StatusSortParam"] = sortOrder == "status" ? "status\_desc" : "status";

switch (sortOrder)

{

case "date\_desc":

ordersQuery = ordersQuery.OrderByDescending(o => o.OrderDate);

break;

case "status":

ordersQuery = ordersQuery.OrderBy(o => o.Status);

break;

case "status\_desc":

ordersQuery = ordersQuery.OrderByDescending(o => o.Status);

break;

default:

ordersQuery = ordersQuery.OrderBy(o => o.OrderDate);

break;

}

var orders = await ordersQuery.ToListAsync();

return View(orders);

}

**Ispis 9:** Metoda OrderHistory

Metoda OrderDetails omogućava korisnicima pregled detalja određene narudžbe. Preuzima narudžbu i njezine detalje iz baze podataka na temelju ID-a narudžbe (ispis 10).

public async Task<IActionResult> OrderDetails(int id)

{

var order = await \_context.Orders

.Include(o => o.OrderDetails)

.ThenInclude(od => od.Book)

.FirstOrDefaultAsync(o => o.OrderId == id && o.UserId == \_userManager.GetUserId(User));

if (order == null)

{

return NotFound();

}

return View(order);

}

**Ispis 10:** Metoda OrderDetails

Metoda ChangeUsername omogućava korisnicima promjenu njihovog korisničkog imena. Prikazuje obrazac za unos novog korisničkog imena i ažurira korisnički račun u bazi podataka (ispis 11).

public async Task<IActionResult> ChangeUsername(ChangeUsernameViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

var user = await \_userManager.GetUserAsync(User);

if (user == null)

{

return View("Error");

}

var result = await \_userManager.SetUserNameAsync(user, model.NewUsername);

if (result.Succeeded)

{

await \_signInManager.RefreshSignInAsync(user);

return RedirectToAction(nameof(EditProfile));

}

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

return View(model);

}

**Ispis 11**: Metoda ChangeUsername

Ostale metode unutar direktorija AccountController uključuju:

* EditProfile: prikazuje stranicu za uređivanje profila korisnika.
* ChangeEmail: omogućava korisnicima promjenu email adrese.
* ChangePassword: omogućava korisnicima promjenu lozinke.

### 4.4.2 AdminController

AdminController je odgovoran za administracijske funkcionalnosti aplikacije, uključujući upravljanje knjigama, narudžbama i korisnicima.

Metoda AddBook omogućava administratorima dodavanje novih knjiga u katalog. Prikazuje obrazac za unos podataka o knjizi, a nakon potvrde dodaje knjigu u bazu podataka. Implementacija metode AddBook unutar direktorija AdminController je prikazana na ispisu 12.

public IActionResult AddBook()

{

ViewBag.Authors = new SelectList(\_context.Authors, "AuthorId", "Name");

ViewBag.Categories = new SelectList(\_context.Categories, "CategoryId", "Name");

return View();

}

[HttpPost]

public async Task<IActionResult> AddBook(Book book)

{

book.Stock = book.Stock > 0 ? book.Stock : 0;

\_context.Add(book);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(ManageBooks));

}

**Ispis 12:** Metoda AddBook

**Metoda UpdateOrderStatus** omogućava administratorima ažuriranje statusa narudžbi (ispis 13).

public async Task<IActionResult> UpdateOrderStatus(int orderId, OrderStatus status)

{

var order = await \_context.Orders.FindAsync(orderId);

if (order == null)

{

return NotFound();

}

order.Status = status;

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(ListOrders));

}

**Ispis 13**: Metoda UpdateOrderStatus

Ostale metode direktorija AdminController uključuju:

* ManageBooks - omogućava administratorima pregled i upravljanje katalogom knjiga. Preuzima sve knjige iz baze podataka i prosljeđuje ih pogledu.
* EditBook - omogućava administratorima uređivanje postojećih knjiga u katalogu. Prikazuje obrazac za uređivanje podataka o knjizi i ažurira podatke u bazi podataka.
* ListOrders - prikazuje sve narudžbe i omogućava administratorima pregled i upravljanje narudžbama.
* OrderDetails - prikazuje detalje određene narudžbe.
* ListUsers - prikazuje sve korisnike i omogućava administratorima pregled i upravljanje korisničkim računima.
* EditUser - omogućava administratorima uređivanje podataka o korisnicima.

### 4.4.3 BooksController

BooksController je odgovoran za pretragu i pregled knjiga unutar aplikacije. Ovaj kontroler omogućava korisnicima pretraživanje dostupnih knjiga i pregled detaljnih informacija o svakoj knjizi.

Metoda Index omogućava korisnicima pretragu i pregled liste knjiga. Preuzima sve knjige iz baze podataka i prosljeđuje ih pogledu. Korisnici mogu pretraživati knjige prema naslovu, autoru ili kategoriji. Implementacija metode Index unutar BooksControllera je prikazana na ispisu 14.

public async Task<IActionResult> Index(string searchString, int? categoryId, int? authorId, decimal? minPrice, decimal? maxPrice)

{

ViewData["Categories"] = new SelectList(\_context.Categories, "CategoryId", "Name");

ViewData["Authors"] = new SelectList(\_context.Authors, "AuthorId", "Name");

var booksQuery = from b in \_context.Books

.Include(b => b.Author)

.Include(b => b.Category)

.Include(b => b.Reviews)

where b.Stock > 0

select b;

if (!string.IsNullOrEmpty(searchString))

{

booksQuery = booksQuery.Where(s => s.Title.Contains(searchString));}

if (categoryId.HasValue && categoryId > 0)

{

booksQuery = booksQuery.Where(x => x.CategoryId == categoryId);

}

if (authorId.HasValue && authorId > 0)

{

booksQuery = booksQuery.Where(x => x.AuthorId == authorId);

}

if (minPrice.HasValue)

{

booksQuery = booksQuery.Where(x => x.Price >= minPrice);

}

if (maxPrice.HasValue)

{

booksQuery = booksQuery.Where(x => x.Price <= maxPrice);

}

var books = await booksQuery.ToListAsync();

var booksWithAverageRatings = books.Select(b => new

{

Book = b,

AverageRating = b.Reviews.Any() ? b.Reviews.Average(r => r.Rating) : 0

});

ViewBag.BooksWithAverageRatings = booksWithAverageRatings;

return View(books);

}

**Ispis 14:** Metoda Index

Metoda Details omogućava korisnicima pregled detalja određene knjige. Preuzima knjigu iz baze podataka na temelju ID-a knjige i uz recenzije i ocjene ju prosljeđuje pogledu (ispis 15).

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var book = await \_context.Books

.Include(b => b.Author)

.Include(b => b.Category)

.Include(b => b.Reviews)

.AsNoTracking()

.FirstOrDefaultAsync(m => m.BookId == id);

if (book == null)

{

return NotFound();

}

double? averageRating = book.Reviews?.Count > 0 ? (double?)book.Reviews.Average(r => r.Rating) : null;

ViewBag.AverageRating = averageRating;

return View(book);

}

**Ispis 15:** Metoda Details

Metoda AddReview omogućava korisnicima da dodaju recenziju za određenu knjigu. Ova metoda prima bookId, content, i rating kao parametre, kreira novu instancu modela Review, povezuje je s korisnikom i knjigom, te sprema u bazu podataka (ispis 16).

public async Task<IActionResult> AddReview(int bookId, string content, int rating){

if (ModelState.IsValid){

Review newReview = new Review

{

BookId = bookId,

Content = content,

Rating = rating,

UserId = User.FindFirstValue(ClaimTypes.NameIdentifier)

};

\_context.Add(newReview);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Details), new { id = bookId });

}

return View();

}

**Ispis 16:** Metoda AddReview

### 4.4.4 CartController

CartController je odgovoran za upravljanje košaricom i procesom kupovine unutar aplikacije. Ovaj kontroler omogućava korisnicima pregled košarice, dodavanje i uklanjanje knjiga iz košarice, te dovršavanje kupovine.

Metoda AddToCart omogućava korisnicima dodavanje knjiga u košaricu. Preuzima ID knjige, te ju dodaje u košaricu (ispis 17).

public IActionResult AddToCart(int bookId, int quantity)

{

var book = \_context.Books.SingleOrDefault(b => b.BookId == bookId);

if (book != null && book.Stock > 0)

{

var cart = GetCart();

var cartItem = cart.Find(item => item.BookId == bookId);

if (cartItem != null)

{

var newQuantity = cartItem.Quantity + quantity;

cartItem.Quantity = Math.Min(newQuantity, book.Stock);

}

else

{

cart.Add(new CartItem { BookId = bookId, Title = book.Title, Price = book.Price, Quantity = Math.Min(quantity, book.Stock) });

}

SaveCart(cart);

}

return RedirectToAction("Details", "Books", new { id = bookId });

}

**Ispis 17:** Metoda AddToCart

Metoda RemoveFromCart omogućava korisnicima uklanjanje knjiga iz košarice. Preuzima ID stavke iz košarice, uklanja je i ažurira sadržaj košarice (ispis 18).

public IActionResult RemoveFromCart(int bookId)

{

var cart = GetCart();

var cartItem = cart.Find(item => item.BookId == bookId);

if (cartItem != null)

{

cart.Remove(cartItem);

SaveCart(cart);

}

return RedirectToAction("Index");

}

**Ispis 18:** Metoda RemoveFromCart

Ostale metode unutar direktorija CartController uključuju:

* Index - prikazuje sadržaj korisnikove košarice. Preuzima sve stavke iz košarice za trenutnog korisnika i prosljeđuje ih pogledu.
* Checkout - prikazuje obrazac za unos podataka za dostavu i plaćanje.
* ProcessCheckout - obrađuje podatke unesene u obrascu za kupovinu, kreira novu narudžbu i sprema je u bazu podataka.
* PaymentSuccessful **-** prikazuje korisniku potvrdu o uspješno obavljenoj kupovini.

## 4.5 Pogledi

U ovom potpoglavlju je opisana implementacija pogleda web aplikacije za prodaju knjiga.

### 4.5.1 \_Layout.cshtml

\_Layout.cshtml je glavni layout stranice koji definira osnovnu strukturu svih stranica aplikacije. Layout obično sadrži zajedničke elemente kao što je navigacija prikazana na slici 2.

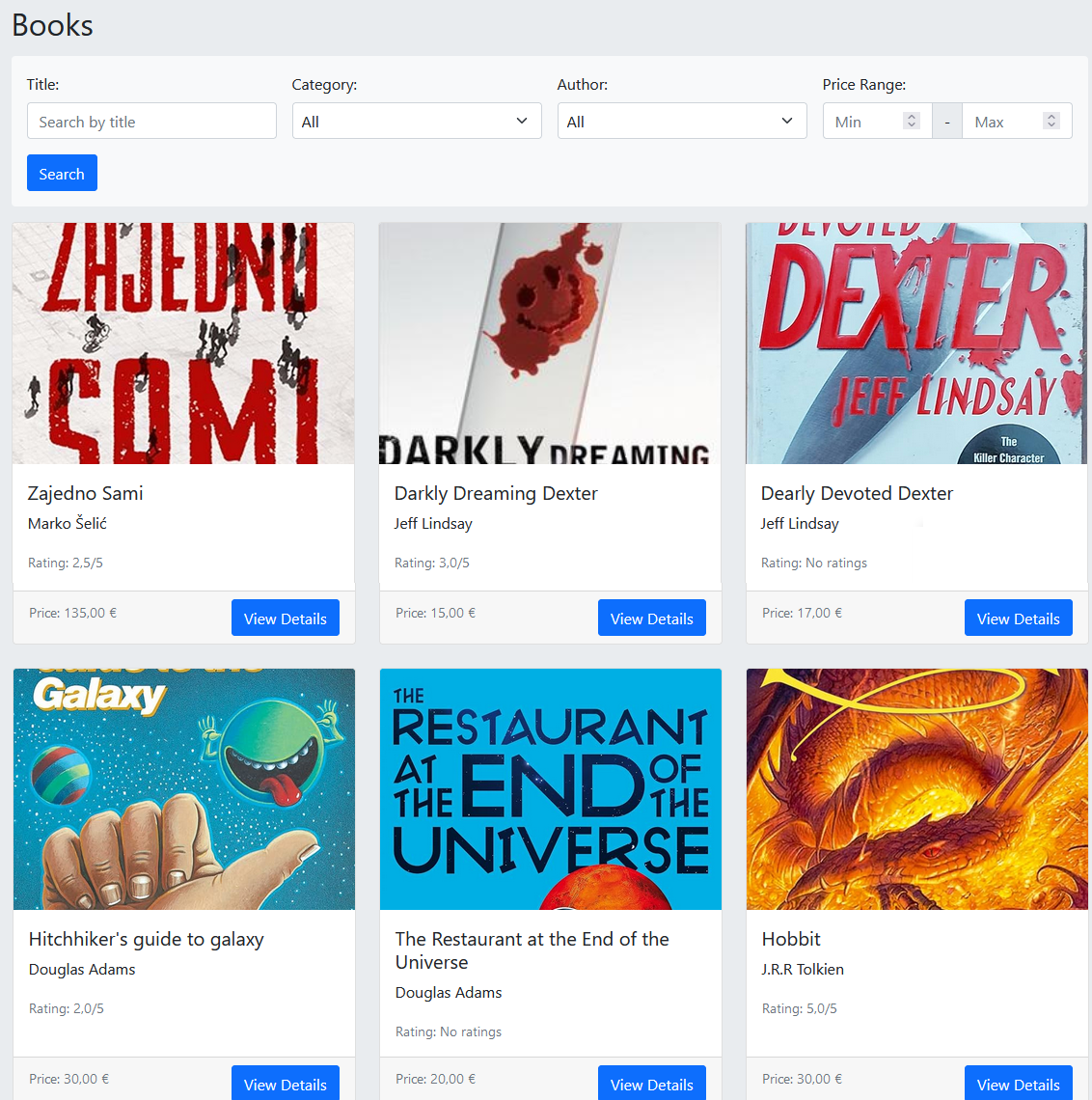


**Slika 2:** Navigacija

### 4.5.2 Pogledi za knjige

Pogledi za knjige omogućavaju korisnicima pregled dostupnih knjiga i detalja o pojedinoj knjizi.

Index.cshtml je pogled za prikaz liste knjiga (slika 3). Ovaj pogled prikazuje sve dostupne knjige s osnovnim informacijama kao što su naslov, autor i cijena. Također omogućava korisnicima pretraživanje knjiga. Implementacija pogleda Index.cshtml je prikazana na ispisu 19.



**Slika 3:** Korisničko sučelje implementirano pogledom Books/Index.cshtml

<h2 class="mb-3">Books</h2>

<div class="search-area">

<**form** **asp-controller**="Books" **asp-action**="Index" method="get">

<div class="row g-3 align-items-end">

<div class="col-md-3">

<label class="form-label">Title:</label>

<input type="text" class="form-control" name="searchString" value="@ViewData["CurrentFilter"]" placeholder="Search by title">

</div>

<div class="col-md-3">

<label class="form-label">Category:</label>

<**select** class="form-select" **name**="categoryId" **asp-items**="ViewBag.Categories as SelectList">

<**option** **value**="">All</**option**>

</**select**>

</div>

<div class="col-md-3">

<label class="form-label">Author:</label>

<**select** class="form-select" **name**="authorId" **asp-items**="ViewBag.Authors as SelectList">

<**option** **value**="">All</**option**>

</**select**>

</div>

<div class="col-md-3">

<label class="form-label">Price Range:</label>

<div class="input-group">

<input type="number" class="form-control" name="minPrice" placeholder="Min">

<span class="input-group-text">-</span>

<input type="number" class="form-control" name="maxPrice" placeholder="Max">

</div>

</div>

<div class="col-md-2">

<button type="submit" class="btn btn-primary">Search</button>

</div>

</div>

</**form**>

</div>

<div class="row row-cols-1 row-cols-md-3 g-4">

@foreach (var book in Model)

{

double? averageRating = book.Reviews?.Count > 0 ? (double?)book.Reviews.Average(r => r.Rating) : null;

<div class="col">

<div class="card h-100">

<img src="@book.CoverImageUrl" class="card-img-top" alt="Book Cover" style="height: 250px; object-fit: cover;">

<div class="card-body">

<h5 class="card-title">@book.Title</h5>

<p class="card-text">@book.Author.Name</p>

<small class="text-muted">Rating: @(averageRating.HasValue ? $"{averageRating:F1}/5" : "No ratings")</small>

</div>

<div class="card-footer">

<small class="text-muted">Price: @book.Price.ToString("C", CultureInfo.GetCultureInfo("de-DE"))</small>

<**a** **asp-action**="Details" **asp-route-id**="@book.BookId" class="btn btn-primary float-end">View Details</**a**>

</div>

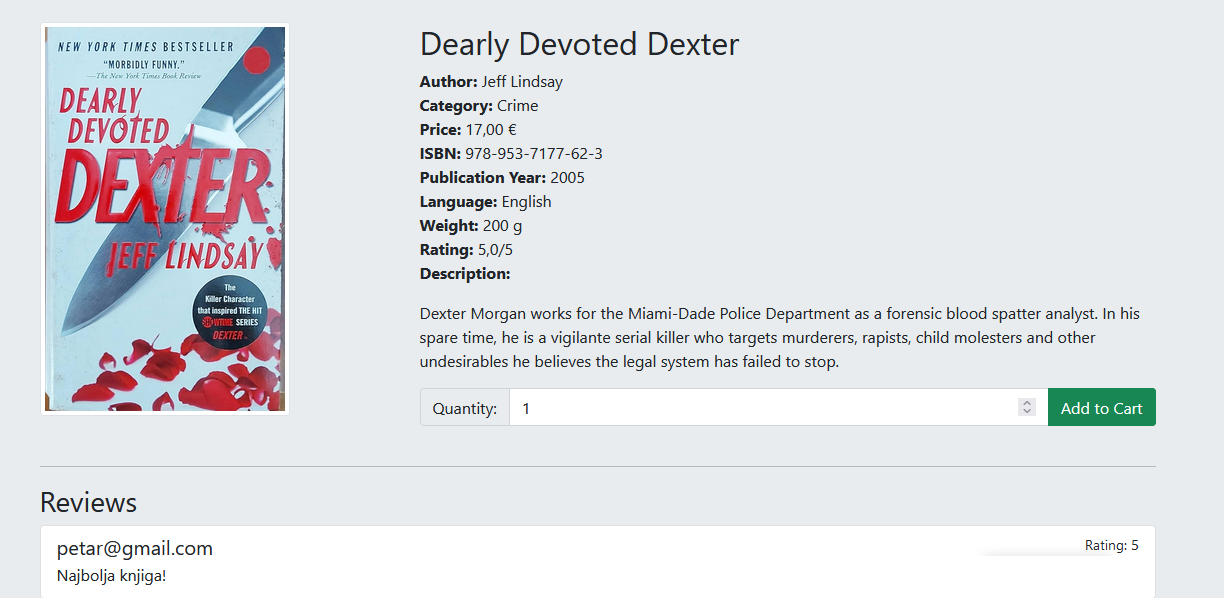
</div>

</div>}

</div>

**Ispis 19**: Direktorij Books/Index.cshtml

Details.cshtml je pogled za prikaz detalja o pojedinoj knjizi (slika 4). Ovaj pogled prikazuje sve informacije o knjizi, uključujući naslov, autora, opis, cijenu, kategoriju, ISBN, godinu izdavanja, jezik, težinu, ocjenu i korisničke recenzije.



**Slika 4:** Korisničko sučelje implementirano pogledom Books/Details.cshtml

### 4.5.3 Pogledi za košaricu

Pogledi za košaricu omogućuju korisnicima pregled sadržaja košarice, procesiranje kupovine i potvrdu uspješne kupovine.

Index.cshtml je pogled za prikaz sadržaja košarice (slika 5). Ovaj pogled prikazuje sve knjige koje je korisnik dodao u košaricu, s mogućnošću uklanjanja knjiga iz košarice. Implementacija pogleda Cart/Index.cshtml prikazana je na ispisu 20.

<div class="container mt-4">

<h2>Your Shopping Cart</h2>

@if (Model.Count == 0)

{

<div class="alert alert-info" role="alert">Your cart is empty.</div>

}

else

{

<table class="table table-hover">

<thead class="table-dark">

<tr>

<th>Title</th>

<th>Price</th>

<th>Quantity</th>

<th>Total</th>

<th>Action</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@item.Title</td>

<td>@item.Price.ToString("C", CultureInfo.GetCultureInfo("de-DE"))</td>

<td>@item.Quantity</td>

<td>@((item.Price \* item.Quantity).ToString("C", CultureInfo.GetCultureInfo("de-DE")))</td>

<td>

<**form** **asp-controller**="Cart" **asp-action**="RemoveFromCart" method="post">

<input type="hidden" name="bookId" value="@item.BookId" />

<button type="submit" class="btn btn-danger">Remove</button>

</**form**>

</td>

</tr>

}

</tbody>

</table>

<div class="text-end">

<h3>Total: @(Model.Sum(item => item.Price \* item.Quantity).ToString("C", CultureInfo.GetCultureInfo("de-DE")))</h3>

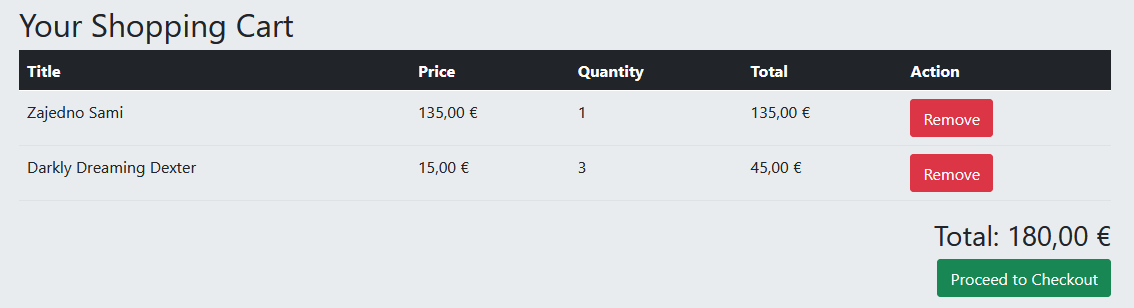
<**a** **asp-controller**="Cart" **asp-action**="Checkout" class="btn btn-success">Proceed to Checkout</**a**>

</div>

}

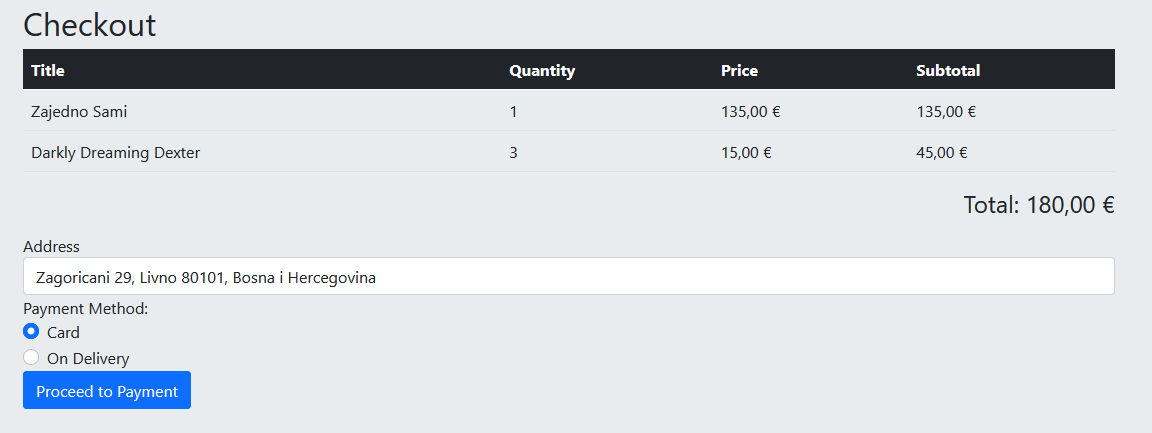
</div>

**Ispis 20:** Direktorij Cart/Index.cshtml



**Slika 5:** Korisničko sučelje implementirano pogledom Cart/Index.cshtml

Checkout.cshtml je pogled za proces kupovine (slika 6). Ovaj pogled omogućava korisnicima unos podataka potrebnih za procesiranje narudžbe, kao što su adresa za dostavu i način plaćanja.



**Slika 6:** Korisničko sučelje implementirano pogledom Checkout.cshtml

PaymentSuccessful.cshtml je pogled za potvrdu uspješne kupovine. Ovaj pogled prikazuje korisniku poruku koja potvrđuje da je kupovina uspješno završena. Implementacija pogleda PaymentSuccessful.cshtml je prikazana na ispisu 21.

<h2>Payment Successful!</h2>

<div class="alert alert-success" role="alert">

Your payment was successful, and your order has been placed.

</div>

<div>

<h4>Order ID: @Model.OrderId</h4>

<h4>Date: @Model.OrderDate.ToString("f")</h4>

<h4>Address: @Model.Address</h4>

<h4>Status: @Model.Status</h4>

<h5>Order Details:</h5>

<table class="table table-bordered">

<thead>

<tr>

<th>Title</th>

<th>Price</th>

<th>Quantity</th>

<th>Total</th>

</tr>

</thead>

<tbody>

@foreach (var detail in Model.OrderDetails)

{

<tr>

<td>@detail.Book.Title</td>

<td>@detail.Price.ToString("C", System.Globalization.CultureInfo.GetCultureInfo("de-DE"))</td>

<td>@detail.Quantity</td>

<td>@((detail.Price \* detail.Quantity).ToString("C", System.Globalization.CultureInfo.GetCultureInfo("de-DE")))</td>

</tr>

}

</tbody>

</table>

</div>

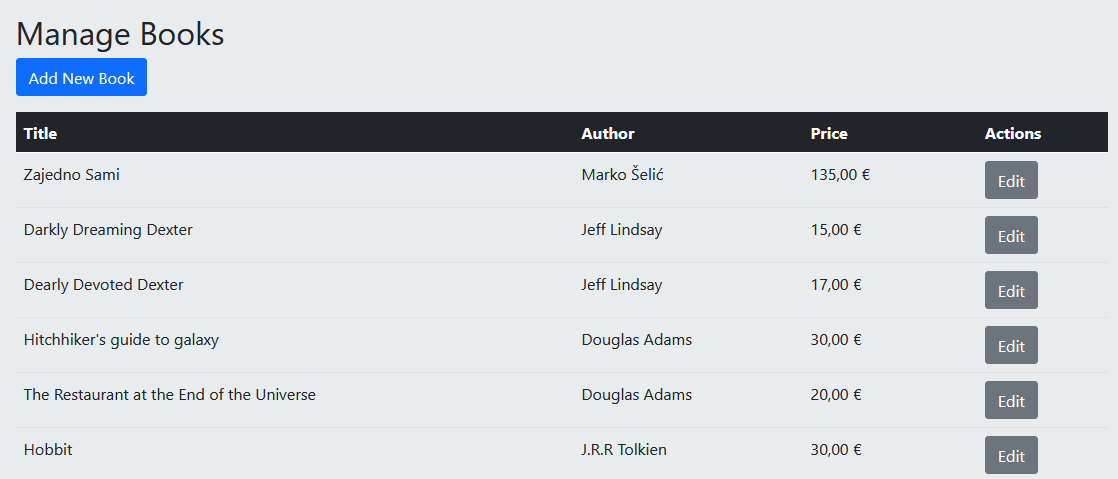
<**a** **asp-controller**="Books" **asp-action**="Index" class="btn btn-primary">Continue Shopping</**a**>

**Ispis 21**: Direktorij PaymentSuccessful.cshtml

### 4.5.4 Pogledi za administraciju

Pogledi za administraciju omogućuju administratorima upravljanje knjigama, narudžbama i korisnicima.

ManageBooks.cshtml je pogled za upravljanje knjigama (slika 7). Ovaj pogled prikazuje listu svih knjiga i omogućuje administratorima dodavanje, uređivanje ili brisanje knjiga. Implementacija pogleda ManageBooks.cshtml prikazana je na ispisu 22.



**Slika 7:** Korisničko sučelje implementirano pogledom ManageBooks.cshtml

<div class="container mt-4">

<h2>@ViewData["Title"]</h2>

<div class="mb-3">

<**a** **asp-action**="AddBook" class="btn btn-primary">Add New Book</**a**>

</div>

<table class="table table-hover">

<thead class="table-dark">

<tr>

<th>Title</th>

<th>Author</th>

<th>Price</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

@foreach (var book in Model)

{

<tr>

<td>@book.Title</td>

<td>@book.Author.Name</td>

<td>@book.Price.ToString("C", CultureInfo.GetCultureInfo("de-DE"))</td>

<td>

<**a** **asp-action**="EditBook" **asp-route-id**="@book.BookId" class="btn btn-secondary">Edit</**a**>

</td>

</tr>

}

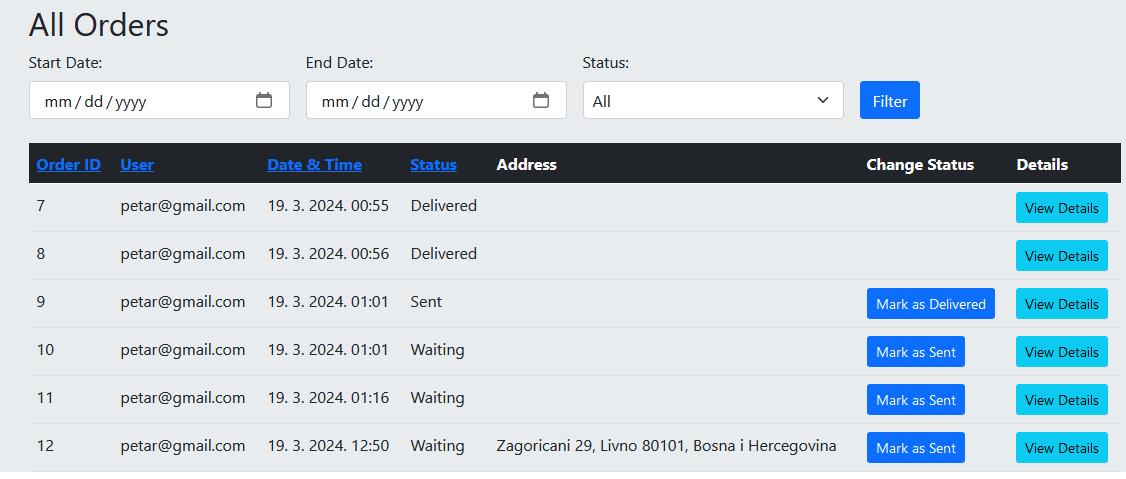
</tbody>

</table>

</div>

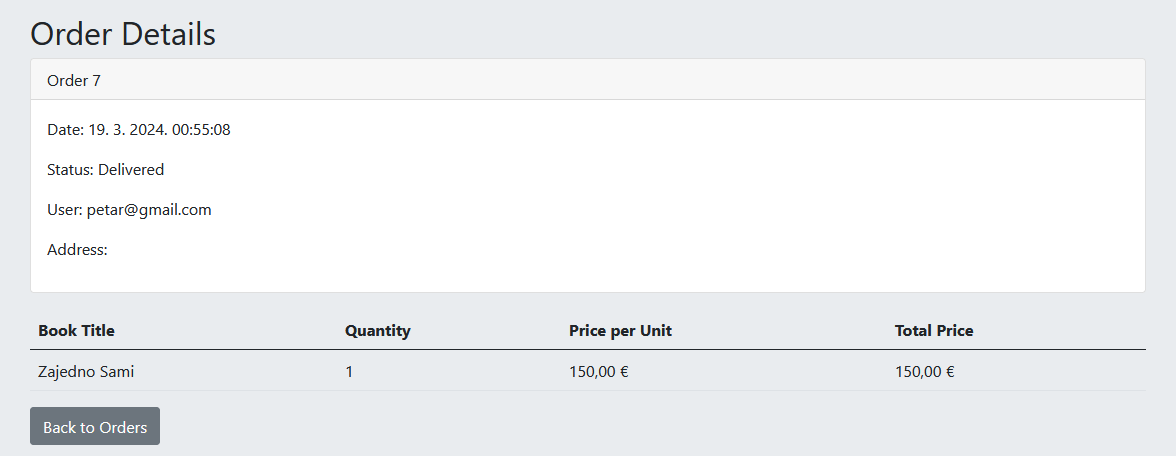
**Ispis 22**: Direktorij ManageBooks.cshtml

ListOrders.cshtml je pogled za prikaz liste narudžbi (slika 8). Ovaj pogled omogućava administratorima pregled svih narudžbi s osnovnim informacijama o narudžbama.



**Slika 8**: Korisničko sučelje implementirano pogledom ListOrders.cshtml

OrderDetails.cshtml je pogled za prikaz detalja narudžbe (slika 9). Ovaj pogled prikazuje sve informacije o narudžbi, uključujući korisnika, naručene knjige, cijene i status narudžbe.



**Slika 9**: Korisničko sučelje implementirano pogledom OrderDetails.cshtml

Ostali pogledi za administraciju uključuju:

* AddBook.cshtml - pogled za dodavanje nove knjige, omogućava unos podataka o novoj knjizi.
* EditBook.cshtml - pogled za uređivanje postojeće knjige, omogućava izmjenu podataka o knjizi.
* ListUsers.cshtml - pogled za prikaz liste korisnika, omogućava pregled svih korisnika s osnovnim informacijama.
* EditUser.cshtml - pogled za uređivanje korisnika, omogućava izmjenu podataka o korisniku.

### 4.5.5 Pogledi za korisnički račun

Pogledi za korisnički račun omogućuju korisnicima upravljanje svojim profilom, narudžbama i prijavom/odjavom. Registracija, prijava i odjava su *scaffoldani* s pomoću Microsoft Identityja.

ChangeUsername je pogled koji omogućuje korisniku promjenu korisničkog imena (ispis 24).

<div class="container">

<h2 class="mt-3">@ViewData["Title"]</h2>

<**form** **asp-action**="ChangeUsername" method="post">

<div class="row">

<div class="col-md-6">

<div class="form-group">

<**label** **asp-for**="NewUsername"></**label**>

<**input** **asp-for**="NewUsername" class="form-control" />

<**span** **asp-validation-for**="NewUsername" class="text-danger"></**span**>

</div>

<button type="submit" class="btn btn-primary">Update Username</button>

</div>

</div>

</**form**>

</div>

**Ispis 24**: Direktorij ChangeUsername.cshtml

Ostali pogledi za korisnički račun uključuju:

* EditProfile.cshtml - pogled sa poveznicama za izmjenu korisničkog imena, email adrese i lozinke.
* ChangeEmail.cshtml - pogled za promjenu email adrese.
* ChangePassword.cshtml - pogled za promjenu lozinke.
* OrderHistory.cshtml - pogled za prikaz povijesti narudžbi korisnika, uključujući mogućnost sortiranja.
* OrderDetails.cshtml - pogled za prikaz detalja narudžbe.

# 5. Zaključak

U ovom radu opisana je izrada web aplikacije za online knjižaru koristeći moderne tehnologije kao što su ASP.NET Core, Entity Framework Core, Microsoft SQL Server, Identity Framework, MVC Pattern i Bootstrap. Kroz implementaciju različitih funkcionalnosti za korisnike i administratore, ostvareni su glavni ciljevi projekta: omogućavanje registracije i prijave korisnika, pretrage i pregleda knjiga, dodavanja knjiga u košaricu, kupovine knjiga te ocjenjivanja i recenziranja knjiga.

Administratori su dobili alate za učinkovito upravljanje katalogom knjiga, pregledom i upravljanjem narudžbama te korisničkim računima. Struktura projekta, konfiguracija, modeli, kontroleri i pogledi su detaljno objašnjeni kako bi se pružio jasan pregled implementacije.

Kroz ovaj projekt, stečena su vrijedna znanja i vještine u radu s ASP.NET Core frameworkom i pratećim tehnologijama. Također, demonstrirana je važnost dobro dizajniranog korisničkog sučelja i iskustva za uspjeh web aplikacije.

Ovaj rad može poslužiti kao osnova za daljnji razvoj i proširenje funkcionalnosti, uključujući integraciju s vanjskim sustavima i poboljšanje performansi i sigurnosti aplikacije. Budući razvoj može uključivati proširenje korisničkih mogućnosti kao što su praćenje čitanja i preporuke knjiga na osnovu ranije kupljenih knjiga. Sigurnost se može povećati implementacijom dvofaktorske autentifikacije i enkripcije osjetljivih podataka. Dodatno, mogu se uvesti izvješća o prodanim knjigama i korisničkim aktivnostima koja će pomoći u boljem razumijevanju potreba korisnika.

**Literatura**

1. What is Visual Studio?, <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022> (posjećeno 4.1.2024.)
2. ASP.NET Core Documentation, <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0> (posjećeno 3.3.2024.)
3. Entity Framework Core Documentation, <https://learn.microsoft.com/en-us/ef/> (posjećeno 5.3.2024.)
4. Identity Framework Documentation, <https://learn.microsoft.com/en-us/aspnet/identity/> (posjećeno 5.3.2024.)
5. Overview of ASP.NET Core MVC, <https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website> (posjećeno 5.3.2024.)
6. What is SQL Server?, <https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16> (posjećeno 10.3.2024.)
7. Bootstrap Introduction, <https://getbootstrap.com/docs/5.0/getting-started/introduction/> (posjećeno 3.4.2024.)
8. Paypal sandbox testing guide, <https://developer.paypal.com/tools/sandbox/> (posjećeno 4.4.2024.)
9. Udemy, <https://www.udemy.com> (posjećeno 4.4.2024.)