Predstavljanje .NET 6 i MVC tehnologija kroz primjer online knjižare

Latinčić, Ivan

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Applied Sciences in Information Technology / Veleučilište suvremenih informacijskih tehnologija**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:289:888036

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-02-22**

Repository / Repozitorij:

VSITE Repository - Repozitorij završnih i diplomskih radova VSITE-a

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

VELEUČILIŠTE SUVREMENIH INFORMACIJSKIH TEHNOLOGIJA

STRUČNI PRIJEDIPLOMSKI STUDIJ INFORMACIJSKIH TEHNOLOGIJA

Ivan Latinčić

ZAVRŠNI RAD

PREDSTAVLJANJE .NET 6 I MVC TEHNOLOGIJA KROZ PRIMJER ONLINE KNJIŽARE

Zagreb, listopada 2024.

Veleučilište suvremenih informacijskih tehnologija
10000 Zagreb, Ulica Vjekoslava Klaića 7

Studij:	Stručni prijediplomski studij informacijskih tehnologija smjer programiranje
Student:	**Ivan Latinčić**
Matični broj:	2019028

Zadatak završnog rada

Predmet:	Programiranje na Internetu
Naslov:	**Predstavljanje .NET 6 i MVC tehnologija kroz primjer online knjižare**
Zadatak:	Razviti i pripremiti za produkcijski rad online knjižaru. Koristiti C#, ASP.NET i MS SQL tehnologije za razvoj konačnog rješenja. Opisati postupak izrade. Opisati .NET 6 tehnologiju.
Mentor:	Mariza Maini, pred.
Zadatak uručen kandidatu:	30.10.2023.
Rok za predaju rada:	24.10.2024.
Rad predan:	_____

Povjerenstvo:

Jurica Đurić, v. pred.	član predsjednik	_____
Mariza Maini, pred.	mentor	_____
mr. sc. Julijan Šribar, v. pred.	član	_____

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS KODOVA

1. UVOD

U ovom radu je opisana izrada mrežne platforme korištenjem .NET 6.0. MVC tehnologije s pomoću Visual Studija 2022, integriranog razvojnog okruženja tvrtke Microsoft. Ova mrežna platforma osmišljena je kao prostor za sve pojedince, „knjigoljupce“, bilo one čija je kolekcija knjiga dosegla zabrinjavajuću brojku pa se žele riješiti pokoje ili pak one koji svoju knjišku kolekciju žele povećati. Ovo poslovanje se razlikuje od onog klasičnog, prvenstveno zbog toga što ne postoji jedan univerzalan prodavač, koji je najčešće sama trgovina, nego svoje proizvode u principu može prodavati bilo koja privatna osoba kreiranjem vlastitog korisničkog računa na ovoj aplikaciji. Dakle, ova platforma je koncipirana na principu „više prodavača-više kupaca“. Postoje 3 vrste korisnika ove platforme: administrator aplikacije, kupac te prodavač.

Za kreiranje korisničkog sučelja korišteni su prezentacijski jezici: HTML, CSS te Bootstrap, a za izradu dinamičkih mrežnih stranica, programski jezik C# i .NET 6 tehnologija, kao i MVC obrazac te Entity Framework.

U prvome dijelu rada ukratko je opisan princip rada te sve značajke potrebne za realizaciju, kao i potencijalni problemi tijekom realizacije takozvane *multi-vendor* platforme. Prije izrade same platforme, potrebno je najprije definirati tko sve može obnašati ulogu prodavača, jesu li to isključivo privatne osobe, kompanije ili oboje. Zatim je potrebno definirati kako će se naplaćivati provizija, odnosno naknada za izlaganje oglasa unutar aplikacije. Bitno je razriješiti i pitanje dostave, tako da se najprije definiraju svi mogući oblici dostave te tko je zadužen za isporuku pošiljki: prodavači ili vlasnik platforme. Od ključne važnosti je stvoriti i funkcionalan i učinkovit sustav isplate prodavačima.

Drugi dio razrađuje sve tehnologije i alate koji su korišteni za izradu aplikacije. Teme obrađene u ovome dijelu uključuju: predstavljanje .NET i ASP .NET tehnologija, povijest i analizu MVC (engl. *Model-View-Controller*) obrasca, prikaz mogućnosti objektno-relacijskog okvira Entity Framework Core te razlike između pristupa razvoja „kôd prvo“ i pristupa razvoja „baza prvo“, analizu obrazaca *Repository* i *Unit of Work*, opis sustava članstva ASP.NET Identity koji omogućuje autentifikaciju (engl. *authentication*) i autorizaciju (engl. *authorization*) korisnika unutar .NET aplikacije te proces bilježenja različitih događaja u stvarnome vremenu, zvan *Logging*.

U trećem dijelu, opisan je proces izrade same aplikacije. Prvo je definirana struktura aplikacije te je naveden popis svih direktorija i glavnih datoteka projekta, uz njihov kratki opis. Zatim je obrađen proces kreiranja baze podataka za ovaj projekt te su pojašnjene sve tablice unutar nje. Treće potpoglavlje ovoga dijela sadrži detaljne opise svih funkcionalnosti koje ova platforma pruža svojim korisnicima. Svaka korisnička uloga ima svoj jedinstveni skup funkcionalnosti na

raspolaganju. Neke od ključnih funkcionalnosti koje su omogućene korisnicima s ulogom „Korisnik“ uključuju izradu korisničkog računa, prijavu putem internih ili vanjskih računa (Google i Facebook), promjenu zaporke, zadavanje nove zaporke u slučaju zaboravljanja stare, dodavanje stvari u košaricu, kao i samu kupnju. Korisnici čija je korisnička uloga „Prodavač“ imaju mogućnost postavljanja, ažuriranja i brisanja vlastitih oglasa.

Završni dio ovoga rada sadrži detaljni osvrt na proces izrade ove platforme, rekapitulaciju svih problema prisutnih pri procesu razvoja, buduća potencijalna proširenja i moguća poboljšanja korisničkog iskustva.

2. PLATFORMA ZA PRODAJU RABLJENIH KNJIGA

U ovome poglavlju je ukratko opisan princip rada takozvanih *multi-vendor* platformi te svi osnovni pojmovi i procesi koji se za njih vežu.

2.1. Opis rada

Multi-vendor platforme su komercijalne internetske trgovine koje za razliku od „klasičnih“, imaju više od jednoga prodavača. Platforma razvijena u okviru ovog rada je namijenjena prodaji rabljenih knjiga u fizičkom formatu te svaki prodavač, nakon registracije na platformu, ima mogućnost izrade vlastitog oglasa. Ukaže li se potreba, platforma se jednostavno može proširiti i na prodaju ostalih vrsta proizvoda.

Postoje tri vrste korisnika: administrator, prodavač te kupac. Administrator je osoba koja je zadužena za kontroliranje svih procesa koji se odvijaju unutar platforme. Administrator ima mogućnost ažuriranja i brisanja postojećih te dodavanja novih žanrova knjiga te njihovih podžanrova, unutar kojih će svi proizvodi koje prodavači oglašavaju biti svrstani. Ova vrsta korisnika aplikacije ima i mogućnost pregleda svih oglasa, kao i njihovo brisanje. Administratoru aplikacije dostupno je i sučelje koje prikazuje popis svih narudžbi sa svim relevantnim informacijama, kao što su: identifikacijska oznaka, datum i vrijeme njihovog nastanka, kupac koji je izvršio narudžbu, stavke narudžbe, ukupan iznos te stanje narudžbe, bilo onih trenutno u procesu izvršavanja ili onih već obrađenih.

Unutar sučelja administratoru se nudi:

1. mogućnost evidentiranja stanja plaćanja svake narudžbe
2. storniranje narudžbi.

Administratoru je na raspolaganju i interaktivno sučelje koje omogućava:

1. ispis svih trenutno dostupnih korisničkih uloga (engl. *user roles*)
2. ažuriranje korisničkih uloga (promjena njihova naziva)
3. prikaz svih korisnika aplikacije koji obnašaju određenu ulogu
4. brisanje ili dodavanje korisnika za pojedine uloge
5. dodavanje novih korisničkih uloga po želji.

Kao zasebna funkcionalnost, administratoru se nudi mogućnost prikaza svih registriranih korisnika, s njihovim korisničkim imenom i adresom elektroničke pošte. Administrator također ima i mogućnost blokiranja, tj. brisanja korisničkih računa prodavača iz baze podataka u slučaju korisnikova nepoštivanja pravila poslovanja, kao što je primjerice neisporučivanje plaćenih pošiljki.

Administrator platforme jedina je korisnička uloga čiji odabir nije dostupan prilikom registracije novoga korisnika, nego je ona dodijeljena posebnome korisniku, s adresom elektroničke pošte i lozinkom, tijekom procesa izrade same platforme. Administrator, po potrebi, može određenim korisnicima dodijeliti ovu ulogu pomoću prethodno spomenutog sučelja.

Druga korisnička uloga prisutna unutar ove platforme je uloga korisnika. Funkcionalnosti koje uloga korisnika (kupca) ima na raspolaganju su univerzalne za svaku internetsku trgovinu, a one uključuju:

1. pregled svih proizvoda
2. pregled proizvoda po kategorijama i pripadajućim potkategorijama
3. dodavanje proizvoda u korisničku košaricu
4. plaćanje
5. kreiranje narudžbe.

Korisniku se također pruža zasebni padajući izbornik s nekoliko stavki:

1. korisnikov profil
2. sučelje za promjenu lozinke
3. sučelje za dodavanje interne lozinke (u slučaju kada se korisnik prijavljuje u sustav putem eksternih poslužitelja, kao što su primjerice Facebook i Google)
4. prikaz svih trenutno aktivnih ili dovršenih narudžbi.

Zadnji tip korisnika platforme je prodavač, čije će specifikacije detaljno biti opisane u nastavku.

2.2. Uloga prodavača

Prilikom početne faze izrade ove platforme, jedno od glavnih pitanja, čije je rješenje bilo od ključne važnosti za logiku implementiranja daljnjih funkcionalnosti, bilo je „Tko sve može biti prodavač: privatne osobe, tvrtke ili oboje?“ Tijekom početne faze rada, funkciju prodavača mogu obnašati samo privatne osobe. Ovo je u skladu s trenutnom vizijom poslovanja ove platforme. U budućnosti, ovisno o poslovnim ciljevima i vizijama platforme, prodavači bi mogli biti i ovlaštene kompanije, kao što je slučaj unutar mnogih svjetski poznatih *multi-vendor* platformi, primjerice Amazon, AliExpress te eBay.

2.3. Provizija

Kako bi poslovanje i održavanje platforme bilo isplativo, ona mora ostvarivati određenu zaradu. Zarada se ostvaruje provizijom, odnosno naknadom za korištenje usluga koje platforma pruža. Plaćanje određene naknade za mogućnost izlaganja svojih proizvoda može se slikovito predočiti kao digitalni ekvivalent klasične tržnice. Provizija se može implementirati na nekoliko

načina: kao fiksna cijena bez obzira na cijenu samoga proizvoda, kao određeni postotak cijene samoga proizvoda, kao naknada koju su prodavači obvezni plaćati platformi u određenom razdoblju (npr. mjesečna ili godišnja provizija) ili pak kao naknada koju su prodavači obvezni plaćati platformi na svaki određeni broj oglasa koje postave. Na platformi, provizija iznosi 5 % ukupne cijene proizvoda koju je zadao prodavač. Cijena proizvoda bez provizije, kao i ona nakon provizije, vidljiva je svakom korisniku registriranom kao prodavač, unutar pogleda pod nazivom „Popis mojih proizvoda“.

2.4. Dostava

Za funkciju dostave proizvoda, bilo je važno ustanoviti hoće li ju obnašati vlasnik aplikacije ili će za dostavu biti zaduženi sami prodavači. Kao rješenje ovoga problema bila je odabrana druga opcija, prvenstveno zbog njene jednostavnosti i logičnosti. Cijena dostave uključena je u cijenu proizvoda.

2.5. Isporuka isplate prodavačima

Budući da se prodaja na *multi-vendor* platformama odvija preko platforme, a ne izravno preko samoga prodavača, tj. osobe ili tvrtke koja je postavila određeni prodajni oglas, nužno je imati realiziran mehanizam kojim će zarada od pojedinog proizvoda biti proslijeđena njegovom prodavaču. U trenutnoj fazi razvoja ove aplikacije ne postoji implementirana funkcionalnost unutar same aplikacije, kojom bi administrator platforme mogao proslijediti zaradu na bankovni račun prodavača, nego se prijenos obavlja „ručno“.

3. KORIŠTENE TEHNOLOGIJE I ALATI

U ovom dijelu rada detaljno su opisane ključne tehnologije i alati za razvoj mrežne platforme.

3.1. .NET

Preteča .NET-a je programska platforma .NET Framework, koju je objavio Microsoft 2002. godine. Ova platforma sadrži mnoštvo alata, programskih jezika te biblioteka za izradu različitih vrsta aplikacija, bilo mrežnih stranica, konzolnih aplikacija, mrežnih obrazaca (engl. *Web Forms*) ili slično, isključivo za operativni sustav Windows. Aplikacije koje se razvijaju pomoću ovog razvojnog okruženja mogu biti bazirane i pokrenute ili na nekom poslužitelju (engl. *server based*) ili na programerovom računalu.(Microsoft) (Microsoft) Postoji niz programskih jezika dostupnih na .NET platformi, a C#, F# i Visual Basic su najčešće korišteni. (Microsoft)

Kôd se kompajlira u takozvani objektno-orijentirani asemblerski jezik, zvan CIL (engl. *Common Intermediate Language*), koji se pohranjuje u datoteke s nastavcima .dll ili .exe. CIL kôd se zatim prevodi u strojni ili izvorni kôd, s pomoću *JIT* kompilacije (engl. *Just-InTime Compilation*) koju pruža CLR (engl. *Common Language Runtime*). CLR je integralna komponentna .NET Frameworka čija je temeljna funkcija osiguravanje da se aplikacije izvršavaju na optimalan, učinkovit i siguran način.(Agrawal, 2023) (Agrawal, 2023)

Glavne komponente CLR-a uključuju (Agrawal, 2023) (Agrawal, 2023):

1. *JIT*
2. *CLS* (engl. *Common Language Specification*) predstavlja skup različitih sintaktičkih pravila koja osiguravaju da različiti programski jezici unutar .NET platforme mogu surađivati i biti razumljivi CLR-u.
3. *CTS* (engl. *Common Type System*) definira pravila za tipove podataka koji su zajednički svim programskim jezicima unutar .NET platforme, osiguravajući da različiti jezici mogu dijeliti i koristiti iste tipove podataka na način razumljiv CLR-u.
4. Sustav za automatsko upravljanje memorijom (engl. *Garbage Collector*) čiji je zadatak automatsko upravljanje memorijom.

Iz potrebe razvoja aplikacija koje su kompatibilne i s drugim operacijskim sustavima, kao što su Linux i macOS, Microsoft u lipnju 2016. godine objavljuje višeplatformskog (engl. *cross-platform*) nasljednika .NET Frameworka, .NET Core. Za razliku od svog prethodnika, ovaj razvojni okvir je otvorenog koda (engl. *open source*), što označava da njegov izvorni kôd svi korisnici mogu pregledati, koristiti te izmijeniti na GitHubu, uz poštivanje Apache 2.0 i MIT licenci.

.NET Core podržava razvoj i izvršavanje mnoštva aplikacija različitih vrsta. Neke od njih su: mrećne, mobilne, *cloud*, stolne te IoT (engl. *Internet of Things*) aplikacije, video igre te strojno učenje. Korištenjem NuGet paketa, koji omogućuju različite značajke, postiže se modularna arhitektura .NET Core aplikacija.(TutorialsTeacher) (TutorialsTeacher)

Nakon izlaska .NET Core 3.1 na tržište, Microsoft odlučuje ujediniti sve postojeće .NET platforme: .NET Framework, .NET Core te alat za razvoj mobilnih aplikacija Xamarin, u jedinstvenu platformu pod nazivom .NET. Prva verzija .NET-a je bila verzija .NET 5. Unutar ovog rada, fokus će biti na .NET 6.0. tehnologiji. .NET 6.0. tehnologija je izdana 9. studenoga 2021. godine, s novom verzijom programskog jezika C#, C# 10 te s novom verzijom integriranog razvojnog okruženja, Visual Studio 2022. Time se pojednostavio postupak pisanja koda, odnosno količina koda koju programer mora sam ručno napisati znatno se smanjila. (Chand, 2022)(Chand, 2022)

3.1.1. ASP. NET

ASP.NET je tehnologija koje služi za izradu dinamičkih mrežnih stranica i aplikacija koristeći .NET razvojni okvir. Ovo razvojno okruženje je nasljednik Active Server Pages tehnologije. Microsoft ga je objavio 5. siječnja 2002. godine, u isto vrijeme kada i prvobitnu verziju .NET Frameworka, u nadi da će biti vodeći alat za izradu dinamičkih mrežnih aplikacija i servisa. U sklopu ovog razvojnog okvira bile su dostupne dvije biblioteke: Win Forms, za razvoj stolnih aplikacija te ASP.NET Forms, za izradu mrežnih aplikacija. Potonji alat dizajniran je na temelju uzorka upravljača stranice (engl. *Page Controller Pattern*). Dakle, svaka stranica aplikacije strukturirana je kao zasebni mrežni obrazac. Svaki obrazac sastoji se od dva dijela: .Aspx datoteke, koja sadrži kôd zaslužan za vizualnu prezentaciju koda korisniku te klase koja je implementirala *Page* upravitelj i pružila potrebnu logiku za izvršavanje funkcionalnosti stranice. Godine 2009. na tržište izlazi alternativa Web Forms razvojnom okviru bazirana na MVC obrascu, ASP.NET MVC. Krajem 2013. godine, svi zasebni okviri (Web Forms, MVC, WEB API i SignalR) objedinjeni su u jedinstvenu platformu pod nazivom ASP.NET. ASP.NET funkcionira na temelju HTTP (engl. *Hypertext Transfer Protocol*) protokola te koristi HTTP naredbe i pravila za uspostavljanje dvosmjerne komunikacije između preglednika i poslužitelja. (Garcia, 2019)

ASP .NET, kao specijalizirani dio .NET-a, pruža sljedeće funkcionalnosti usmjerene na razvoj mrežnih aplikacija(Microsoft) (Microsoft):

1. osnovni okvir (engl. *base framework*) za obradu mrežnih zahtjeva

2. sintaksu za generiranje predložaka mrežne stranice (engl. *web-page templating syntax*), odnosno Razor sintaksu
3. biblioteke za razvoj tipičnih internetskih obrazaca, npr. MVC obrasca
4. sustav za autorizaciju i autentifikaciju
5. proširenje sustava za uređivanje koda.

ASP .NET Core je nasljednik ASP.NET-a. Otvorenog je koda te je razvijen s ciljem da bude podložan za rad, uz Windows, i na macOS i Linux operativnim sustavima.

3.2. MVC obrazac

MVC (engl. *Model-View-Controller*) je arhitektonski obrazac koji se koristi u razvoju aplikacija za poboljšanje organizacije i razdvajanje koda. Ova tehnika podjele koda na tri komponente: model (podatak), pogled (pregled podataka) i upravitelj (logika obrade podataka), omogućuje programerima da jednostavno razdvoje funkcionalnosti aplikacije, što rezultira lakšim održavanjem koda. (Anderson, 2016)(Anderson, 2016)

Koraci za generiranje sadržaja korisniku koji pristupa MVC aplikaciji su(Odyssey, n.d.) (Dot Net Odyssey, n.d.) (Dot Net Odyssey):

1. Korisnik upisuje željenu adresu (engl. *URL – Uniform Resource Locator*) u preglednik i pristupa aplikaciji.
2. Zahtjev dolazi do poslužitelja te se onda prosljeđuje sustavu za preusmjeravanje (engl. *routing*) unutar aplikacije.
3. Sustav za preusmjeravanje na osnovu *URL* uzorka određuje odgovarajući upravitelj za taj zahtjev.
4. Ovisno o zahtjevu, upravitelj komunicira s modelom kako bi dohvatio potrebne podatke iz spremišta podataka.
5. Nakon što upravitelj zaprimi potrebne podatke, on dohvaća odgovarajući pogled te ga popunjava podacima.
6. Upravitelj vraća pogled s podacima, najčešće u obliku HTML sadržaja ili u obliku JSON-a ili XML datoteke.
7. Poslužitelj odgovara početnom zahtjevu generiranjem traženog sadržaja te pruža njegov prikaz klijentu.

3.2.1. Povijest MVC obrasca

Tvorcem MVC obrasca smatra se norveški informatičar Trygve Reenskaug. Kreirao ga je početkom 70-ih godina prošlog stoljeća. Reenskaugova inicijalna vizija MVC obrasca je izgledala ovako (Reenskaug, 1979):

1. Model: Modeli predstavljaju znanje, tj. sve one podatke potrebne za uspješno rješavanje određenog problema te mogu biti prikazani kao jedan objekt ili kao struktura povezanih objekata. Iz perspektive krajnjega korisnika trebala bi postojati veza jedan-na-jedan između modela (njihovih dijelova) i njihove reprezentacije u stvarnome svijetu. Slijedeći ovu logiku, svaki dio modela bi trebao predstavljati određeni dio problema, kojeg taj model nastoji predstaviti. Dakle, moglo bi se reći da je veza između modela i vanjskog svijeta, svojevrsni upravitelj koji je odgovoran za komunikaciju između ta dva dijela sustava.
2. Pogled: Pogledi reprezentiraju vizualni prikaz modela kojeg predstavljaju. Prema Reenskaugovom tumačenju, pogledi bi trebali naglašavati određene atribute svojega modela, ali bi isto tako određene atribute trebali potiskivati i zanemarivati.
3. Upravitelj: Upravitelj figurativno predstavlja vezu između sustava i njegovog korisnika. On pruža korisniku odgovarajuće podatke te njihov prikaz temeljen na zatraženim informacijama. Dakle, upravitelj bi nakon zaprimljenih korisnikovih informacija, trebao te informacije prevoditi u odgovarajuće poruke, koje se dalje prosljeđuju konkretnim pogledima.

Bitno je napomenuti kako je MVC obrazac nastao daleko prije nego što su internetski preglednici postali aktualni te je prvobitno bio namijenjen kao obrazac za izradu grafičkih korisničkih sučelja (engl. *GUI*, tj. *Graphical user interface*). Njegova praktična primjena, takozvani MVC model 1, izgledao je ovako(Walther, 2008) (Walther, 2008):

1. Model predstavlja podatak kojeg aplikacija prikazuje.
2. Pogled služi kao jedna od nekoliko mogućnosti vizualne prezentacije podatka iz modela. Jedan podatak može imati nekoliko pogleda vezanih uz sebe.
3. Upravitelj bilježi akcije koje je korisnik poduzeo te na temelju njih modificira model. Upravitelj tako primjerice može bilježiti akcije koje korisnik napravi mišem te onda izmjenjuje model.

Dakle, prema originalnoj verziji MVC obrasca, pogledi su neizravno ažurirani iz modela, što znači da prilikom izmjene modela, model pokreće događaj (engl. *event*) na temelju kojeg se korisniku prikazuje odgovarajući pogled. Također, upravitelji nemaju izravni kontakt s pogledima, nego upravitelj prvotno ažurira model te se onda, budući da „promatraju“ modele, ažuriraju i pogledi. (Walther, 2008)

Značenje MVC obrasca promijenilo se dolaskom JSP-a (engl. *JavaServer Pages*) 1999. godine, tehnologije za izradu dinamičkih internetskih stranica. U specifikaciji JSP tehnologije,

navedena su dva modela za isporuku internetskih stranica, od kojih drugi utječe i na razvoj ažurirane verzije MVC obrasca. Dakle, više ne postoji nikakva veza između pogleda i modela te se sada sva komunikacija između njih odvija posredovanjem upravitelja. (Walther, 2008)
Model 2 predstavlja drastičan pomak u odnosu na model 1 te je njegova praktična primjena, izgledala ovako (Walther, 2008)(Walther, 2008):

1. Model: omogućava upravljanje podacima tako da ih se može kreirati, čitati, ažurirati te brisati iz jednog ili nekoliko izvora, kao što su primjerice baze podataka (engl. *CRUD*: *Create*, *Read*, *Update*, *Delete*).
2. Pogled: korisničko sučelje koje korisniku prikazuje informacije o modelu.
3. Upravitelj: mehanizam kontrole toka kojim korisnik upravlja aplikacijom.

3.2.2. Model

Modeli nisu ništa drugo nego C# klase sa svojim metodama i svojstvima. Koriste se za definiranje podataka koji se pohranjuju unutar baze podataka, kao i poslovne logike koja se implementira unutar aplikacije.

3.2.3. Pogled

Pogled (engl. *view*) služi za izradu korisničkog sučelja, tj. za prezentaciju podataka korisniku. Poslovna logika ne bi smjela biti sadržana u pogledima. Za izradu pogleda koriste se HTML, CSS, JavaScript i Razor kôd. U .NET projektima, pogledi se spremaju unutar mape *Views*.
Unutar svake mape *Views* nalaze se i datoteke (Troelsen & Japikse, 2022)(Troelsen & Japikse, 2022):

1. _ViewStart.cshtml: izvršava svoj kôd prije nego što je bilo koji pogled učitan. Ovo se ne odnosi na parcijalne poglede (engl. *partial views*), kao ni na predloške (engl. *layouts*). Može se koristiti i za postavljanje podrazumijevanog predloška za poglede unutar kojih on nije specificiran.
2. _ViewImports.cshtml: služi za navođenje direktiva koje će biti dostupne svim pogledima unutar istog direktorija ili poddirektorija.

Pogledi mogu biti strogo ili dinamički tipizirani. Strogo tipizirani pogledi rade samo s jednom instancom nekog modela ili s kolekcijom instanci modela istog tipa. Bitno je napomenuti da unutar pogleda može stajati samo jedna direktiva @model. Pogled je dinamički tipiziran ako ne sadrži direktivu @model. Ovaj tip se koristi za one poglede koji ili ne prikazuju ni jedan model ili rade s više od jednog modela. (Mezei, 2023)(Mezei, 2023)
Neke od prednosti korištenja pogleda unutar mrežne aplikacije uključuju (Smith & Brock, 2022):

1. Poboljšana organizacija aplikacije: pogledi su unutar projekta grupirani po funkcionalnostima koje aplikacija pruža svojim korisnicima što omogućuje jednostavnije snalaženje.
2. Neovisnost: dijelovi aplikacije su labavo povezani (engl. *loosely coupled*), što znači da se pogledi mogu izraditi i promijeniti, odvojeno od komponente poslovne logike i pristupa podacima.

3.2.4. Upravitelj

Upravitelj (engl. *controller*) služi kao posrednik između modela i pogleda. Upravitelju se preusmjeravaju korisnički zahtjevi te ih on najprije obrađuje, a zatim prosljeđuje potrebne podatke modelu za obradu. Nakon zaprimanja podataka iz odgovarajućeg modela, upravitelj određuje putem kojeg pogleda ih se treba prikazati korisniku. Upravitelji se koriste za definiranje i grupiranje različitih metoda, takozvanih akcija (engl. *Action method*). Ova agregacija akcija omogućuje zajedničku primjenu skupova pravila, vezanih uz usmjeravanje (engl. *Routing*), predmemoriju (engl. *Cache*) te autorizaciju (engl. *Authorization*). Zahtjevi se na radnje preslikavaju usmjeravanjem. Nakon obrade korisničkog zahtjeva, upravitelj generira povratni podatak koji se šalje korisniku u obliku odgovarajuće reakcije aplikacije. (Smith & Addie, 2023)(Smith & Addie, 2023)

Vrste povratnih podataka su (Mezei, 2023)(Mezei, 2023):

1. ViewResult, koji se koristi za prikazivanje pogleda
2. RedirectResult, koji se koristi za preusmjeravanje korisnika na neki drugi URL
3. RedirectToActionResult, koji se koristi za preusmjeravanje korisnika na drugu akciju, tj. metodu koja se nalazi unutar istog ili nekog drugog upravitelja
4. ContentResult, koji se koristi za slanje tekstualnih poruka ili XML datoteka
5. JsonResult, koji se koristi za slanje JSON (engl. *JavaScript Object Notation*) objekta kao rezultata metode
6. StatusCodeResult, koji se koristi za slanje HTTP statusnog koda kao rezultata metode.

Metode unutar upravitelja mogu primati korisničke zahtjeve kao (W3Schools):

1. GET: podrazumijevana metoda, kojom se dohvaćaju određeni podaci s poslužitelja. Svi traženi podaci se korisniku šalju *URL*-om te su vidljivi unutar adresne trake preglednika.
2. POST: metoda kojom se podaci šalju poslužitelju kako bi bili obrađeni ili spremljeni.
3. DELETE: metoda za brisanje postojećih resursa.
4. PUT: metoda kojom se postojeći resursi ažuriraju.

3.3. Entity Framework Core

Entity Framework Core je objektno-relacijski okvir, dostupan unutar .NET tehnologije, za razvoj aplikacija koje su pretežito orijentirane podacima. Radi se o nasljedniku Entity Frameworka, koji je pušten na tržište 11. kolovoza 2008. godine. Neke od novih značajki koje se vežu za ovaj okvir, u odnosu na prošle verzije, su to što je otvorenog koda (engl. *open source*) te što je višeplatformski (engl. *cross-platform*). Jedna od glavnih funkcionalnosti koje ovaj okvir pruža je objektno-relacijsko preslikavanje (engl. *ORM; Object-Relational Mapping*), tehnika kojom se podaci iz relacijske baze podataka transformiraju u modele u određenom programskom jeziku. Objektno-relacijsko preslikavanje programerima omogućava korištenje metoda i svojstava nad određenim objektima, koji predstavljaju podatke iz baze podataka, što znači da više ne postoji potreba za pisanjem klasičnih SQL upita nad bazom podataka. (ZZZ Projects, 2023)

Neke od prednosti korištenja Entity Framework Corea, uključuju (ZZZ Projects, 2023):

1. Složenost koda je znatno smanjena zahvaljujući automatizaciji zadataka poput upravljanja konekcijama i generiranja SQL upita. Korištenjem Entity Framework Corea, programerova produktivnost se povećava jer se mogu jednostavnije napisati određeni dijelovi koda, što skraćuje vrijeme potrebno za razvoj.
2. Entity Framework Core pruža podršku za jedinstveni jezik za upite (engl. *LINQ*) koji omogućava pojednostavljeno i brže pisanje upita nad podacima, bez obzira iz kojeg izvora potječu.

Entity Framework Core podržava dvije vrste pristupa razvoju:

1. kôd prvo (engl. *Code-First*)
2. baza prvo (engl. *Database-First*).

Za pristup različitim vrstama baza podataka, Entity Framework Core koristi poslužiteljske modele (engl. *provider model*), kao npr. SQL Server, koji su dostupni putem NuGet paketa, koje programeri sami moraju instalirati unutar aplikacije koju razvijaju.

3.3.1. Pristup kôd prvo

Pristup kôd prvo se koristi kada ne postoji već kreirana baza podataka, koja će se upotrebljavati za rad aplikacije. Na temelju *POCO* (engl. *Plain Old CRL Objects*) klasa, koje su u najviše slučaja ujedno i modeli koji predstavljaju podatke ključne za razvoj aplikacije te kontekstne klase (engl. *DBContext*), migracijom se kreira baza podataka. Svaka *POCO* klasa predstavlja jednu tablicu u bazi podataka, a svaki atribut, tj. svojstvo klase, se mapira u zasebni stupac. Uz atribute, *POCO* klase sadrže i odgovarajuće konstruktore, te *get* i *set* metode za postavljanje i

dobavljanje atributa. Kontekstna klasa (engl. *DBContext*) sadrži popis svih entiteta, koji zapravo predstavljaju tablice unutar baze podataka. Za pretvorbu obične *POCO* klase u tablicu, koristi se klasa DbSet<TEntity>.

Entity Framework Core koristi mehanizam migracija, koji omogućava da se shema baze podataka automatski ažurira u slučaju izmjene modela, a da pritom ne dođe do gubitka postojećih podataka. (Rout, 2022)

Za kreiranje migracija, ako one nisu automatske, kao i za provođenje određenih akcija nad istima, potrebno je izvršiti sljedeće naredbe unutar *Package Manager Console* (ZZZ Projects, 2023):

1. naredba *add-migration* <naziv migracije> omogućava kreiranje migracije
2. naredbom *update-database* se ažurira baza podataka
3. naredbom *update-database* <naziv migracije> baza podataka se vraća u ono stanje u kojem je bila nakon provedbe specificirane migracije
4. naredbom *get-migrations* se ispisuje lista svih migracija unutar projekta
5. naredbom *remove-migration*, moguće je poništiti zadnje izvršenu migraciju, ako se nakon njenog izvršavanja nije ažurirala baza podataka
6. naredbom *drop-database* se briše cijela baza podataka.

3.3.2. Pristup baza prvo

Pristup baza prvo se koristi u slučaju kada je baza podataka kreirana prije kontekstne klase (engl. *DBContext*) te klasa domene. Kod ovoga pristupa, klase se kreiraju na temelju baze podataka, koja je bila razvijena unutar nekog od poslužitelja baza podataka.

3.4. Repository i Unit of Work

Unutar projekta, za osiguravanje konzistentnosti i integriteta podataka, koriste se *Repository* obrasci te klasa zvana *Unit of Work*. Obrascem *Repository*, stvara se sloj apstrakcije između modula niže i modula više razine. Implementacijom ovih obrazaca, podaci, koji su u ovom slučaju pohranjeni u bazi podataka, se odvajaju od poslovne logike. Sve metode, tj. logika pristupanja podacima, smješta se na jedno mjesto te se time smanjuje potreba pisanja istog koda na svakom pojedinom dijelu aplikacije koji to iziskuje. Svaki *Repository* obrazac se sastoji od 3 različita glavna dijela(Saraç, 2023) (Saraç, 2023):

1. sučelje *repository*ja unutar kojeg su definirane sve metode koje se izvode nad podacima, najčešće *CRUD* metode
2. implementacija *repository*ja
3. model.

3.4.1. Repository obrazac

Za implementaciju obrasca *repository* može se koristiti generički pristup, negenerički pristup ili kombinacija generičkog i negeneričkog pristupa. Unutar sučelja *IGenericRepository*, deklarirane su *CRUD* metode, koje se koriste za kreiranje, čitanje, ažuriranje i brisanje podataka. Unutar ove aplikacije, prisutno je nekoliko obrazaca *Repository*, koji su kombinacija generičkog i negeneričkog pristupa. Dakle, uz sve standardne metode koje su dio *Generic Repository*ja, definiraju još i svoje dodatne metode.

3.4.2. Unit of Work

Unit of Work je vrsta obrasca koja osigurava da više repozitorija dijeli isti kontekst. Ovaj obrazac spaja sve transakcije koje se odvijaju u svakom zasebnom repozitoriju, u jednu veliku transakciju. Sve promjene će biti izvršene odjednom. Ako barem jedna od transakcija rezultira pogreškom, drugim riječima, ako se ne uspije održati integritet svih podataka, onda se ni jedna promjena, koja bi se obrascem *Unit of Work* trebala izvršiti, neće izvršiti. (Koirala, 2013)

Unit of Work se izvodi kroz dva koraka (Koirala, 2013):

1. Prvi korak je održavanje listi objekata pohranjenih u memoriji, nad kojima je bila izvršena određena promjena tijekom određene transakcije.
2. Nakon izvršenja same transakcije, sve izmjene izvršene nad zasebnim objektima se šalju bazi podataka kao jedna transakcija kako bi se unutar nje na kraju i pohranile.

3.5. ASP.NET Identity

ASP.NET Identity je sustav koji funkcionira na temelju članstva te omogućuje autentifikaciju i autorizaciju korisnika .NET aplikacije. Autentifikacija je proces pomoću kojeg korisnik potvrđuje svoj identitet prijavom u sustav, najčešće adresom elektroničke pošte i zaporkom, dok je autorizacija proces koji određuje razinu korisnikova pristupa aplikaciji. Sustav sam kreira sve potrebne tablice unutar baze podataka vezane uz korisnike i uloge (engl. *roles*) koje obnašaju. ASP.NET Identity se služi okvirom Entity Framework i pristupom *Code-First* prilikom stvaranja potrebnih entiteta koji će se nalaziti unutar baze podataka.(Galloway, 2023) (Galloway, 2023)

Također, sustav može sam generirati sve upravitelje, modele i poglede vezane uz registraciju i prijavu korisnika. No, u ovome radu su te funkcionalnosti implementirane ručno radi lakše prilagodbe i primjenjivosti unutar aplikacije.

3.6. Logging

Logging je proces bilježenja različitih događaja unutar aplikacije u stvarnome vremenu. Podaci su vezani uz izvršavanje aplikacije, čime znatno pomažu programerima u otklanjanju pogrešaka,

rješavanju problema i testiranju aplikacija. Najčešće se bilježe informativne poruke te podaci o nastaloj pogrešci unutar aplikacije. Informativne poruke bilježe standardne događaje u trenutku njihova izvršavanja, kao što su npr. poziv metode, provjera autentičnosti korisnika, naplata proizvoda i slično. Poruke o pogreškama pružaju sve potrebne podatke za otklanjanje pogrešaka u programu. *ConsoleLogProvider* bilježi informacije izravno na *debug output* konzolu unutar razvojnog okruženja Visual Studio. (SolarWinds Loggly, 2022)

Unutar ove aplikacije, za kreiranje te pohranjivanje zapisa koristi se vanjski poslužitelj NLog. Svi zapisi se pohranjuju unutar XML datoteke koja je pohranjena na poslužitelju na kojem se aplikacija razvija.

4. PRAKTIČNI RAD – PREDSTAVLJANJE .NET 6 I MVC TEHNOLOGIJE KROZ PRIMJER ONLINE KNJIŽARE

U ovom dijelu rada prikazana je projektna struktura, baza podataka, izrada korisničkog sučelja i važniji dijelovi koda koji omogućavaju glavne funkcionalnosti aplikacije.

4.1. Struktura projekta

Struktura projekta se sastoji od nekoliko mapa, od kojih svaka predstavlja smislenu cjelinu te je složena u skladu s pravilima i propisima korištenja MVC obrasca (Tablica 1). Upute nalažu da se unutar direktorija *Models* moraju nalaziti svi modeli, tj. C# klase potrebne za realizaciju projekta, unutar direktorija *Controllers*, svi upravitelji, a svi pogledi smješteni su u poddirektorije unutar direktorija *Views*, čiji naziv odgovara nazivu upravitelja s kojim su povezani.

Tablica 1. Popis svih direktorija i glavnih datoteka projekta i njihov kratki opis

Direktorij	Opis
Connected Services	Reference na sve servise prisutne unutar projekta, kao npr. SQL Server baza podataka.
Dependencies	Reference na *NuGet* pakete unutar projekta.
Properties	Sadrži launchSettings.json datoteku, unutar koje su smještene sve informacije potrebne za pokretanje aplikacije.
Wwwroot	*CSS* datoteke, js. biblioteke te eksterne biblioteke *bootstrap* i *jquery.*
Controllers	Upravitelji
Data	Klase, repozitoriji, *ViewModels, ViewComponents* i enumeratori svrstani po odgovarajućim direktorijima.
Migrations	Migracijske datoteke kreirane pristupom *Code-Firs*t.
Models	Modeli
Views	Pogledi raspodijeljeni po poddirektorijima.
appsettings.json	Datoteka s konfiguracijskim podacima.
nlog.config	Konfiguriranje za *logging* platformu pod nazivom Nlog.
Program.cs	Klasa koja je „ulazna“ točka aplikacije.

4.2. Baza podataka

Baza podataka koja se koristi unutar ove aplikacije kreirana je pristupom *Code-First* s pomoću okvira Entity Framework Core. Tablice unutar baze podataka su kreirane iz takozvanih entiteta,

tj. klasa koje se nalaze unutar mape *Models*. Svi entiteti su definirani unutar kontekstne klase s pomoću klase DbSet<TEntity> (Kôd 1).

```
namespace BookStore.Data
{
  public class BookStoreDbContext : IdentityDbContext<ApplicationUser>
  {
    public BookStoreDbContext(DbContextOptions<BookStoreDbContext> options) : base(options)
    {

    }

    public DbSet<Category> Categories { get; set; }
    public DbSet<SubCategory> SubCategories { get; set; }
    public DbSet<Product> Products { get; set; }
    public DbSet<ShoppingCartItem> ShoppingCartItems { get; set; }
    public DbSet<Order> Orders { get; set; }
    public DbSet<OrderItem> OrderItems { get; set; }
      public DbSet<PaymentMethod> PaymentMethods { get; set; }
      public DbSet<ShippingInfo> ShippingInfos { get; set; }
  }
}
```

Kôd 1. Kontekstna klasa iz koje sve tablice nastaju pristupom *Code-First*

Za realizaciju ovog procesa potrebno je najprije instalirati Microsoft.EntityFrameworkCore.SqlServer kao i Microsoft.EntityFrameworkCore.Tools NuGet pakete. Baza podataka će biti fizički smještena unutar SQL Server poslužitelja. Kodom za spajanje (engl. *connection string*) (Kôd 2), koji se nalazi unutar datoteke appsettings.json, definira se mjesto pohranjivanja prethodno spomenute baze podataka.

```
"ConnectionStrings": {
"DefaultConnection": "Data Source=.;Initial Catalog=BookStDatabase;Integrated Security=True;Pooling=False;Trusted_Connection=True;MultipleActiveResultSets=true"
},
```

Kôd 2. Kôd za spajanje

Kako bi korištenje kreiranog *Db* konteksta unutar aplikacije bilo moguće, potrebno ga je ponajprije dodati unutar klase *Program* (Kôd 3).

```
builder.Services.AddDbContext<BookStoreDbContext>(options => { options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")); });
```

Kôd 3. Kôd za dodavanje Db konteksta unutar aplikacije

Za kreiranje baze podataka i njeno ažuriranje, potrebno je izvršiti određene naredbe unutar NuGet Package Managera, kao što su *add-migration* te *update-database*. Sve migracije, točnije promjene nad bazom podataka, spremljene su unutar foldera *Migrations*. Ovaj proces je detaljnije opisan unutar poglavlja 3.3.

Unutar baze podataka, uz tablice koje su definirane klasama DbSet<TEntity>, nalaze se i one koje su automatski generirane sustavom ASP.NET Identity te omogućavaju autorizaciju i autentifikaciju korisnika (Tablica 2).

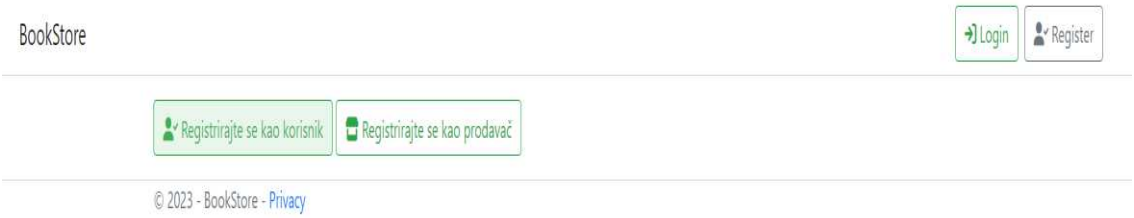
Tablica 2. Popis svih tablica u korištenoj bazi podataka

Naziv tablice	**Opis**
AspNetUsers	Korisnički podaci: korisničko ime, e-mail, zaporka.
AspNetRoles	Popis svih korisničkih uloga unutar aplikacije.
AspNetRoleClaims	Podaci vezani za korisničke zahtjeve na osnovi uloge koju obnašaju.
AspNetUserRoles	Povezuje korisničke uloge s korisnicima.
AspNetUserLogins	Podaci o prijavi korisnika.
AspNetUserClaims	Podaci vezani za korisničke zahtjeve.
AspNetUserTokens	Spremnik tokena za vanjsku autentifikaciju korisnika.
Categories	Podaci o kategorijama, tj. književnim žanrovima: korisnička oznaka, naziv te opis kategorije.
OrderItems	Podaci o stavkama narudžbe: identifikacijska oznaka, količina, cijena, identifikacijska oznaka proizvoda, identifikacijska oznaka narudžbe.
Orders	Podaci o narudžbama: identifikacijska oznaka, identifikacijska oznaka korisnika, status narudžbe, status plaćanja narudžbe, datum i vrijeme nastanka narudžbe, popis stavki narudžbe.
PaymentMethods	Podaci vezani uz način plaćanja: identifikacijska oznaka i naziv.
Products	Podaci o proizvodima: identifikacijska oznaka, naziv, autor, jezik, stanje (fizičko stanje proizvoda, npr. korišteno), opis cijena, slika, dostupnost proizvoda, kategorija, potkategorija, prodavač.
ShippingInfos	Podaci vezani uz dostavu pošiljke: korisnik koje je izvršio narudžbu, ime, prezime, adresa, grad/mjesto, poštanski broj, županija, broj mobitela (kontakt).

ShoppingCartItems	Podaci vezani uz stavke koje se nalaze unutar košarice: identifikacijska oznaka, identifikacijska oznaka proizvoda, naziv proizvoda, količina, identifikacijska oznaka košarice.
SubCategories	Podaci o potkategorijama: identifikacijska oznaka, naziv, identifikacijska oznaka kategorije, naziv kategorije.

4.3. Autentifikacija korisnika

Aplikacija nudi mogućnost registracije dva tipa korisnika: kupca i prodavača. Svaki od ta dva tipa ima pristup samo određenim modulima i funkcionalnostima. Neautentificiranom korisniku omogućen je samo pregled proizvoda. Prilikom klika na tipku za registraciju, korisnik može odabrati ulogu kojom se želi registrirati (Slika 1).



Slika 1. Prikaz odabira korisničke uloge prilikom registracije

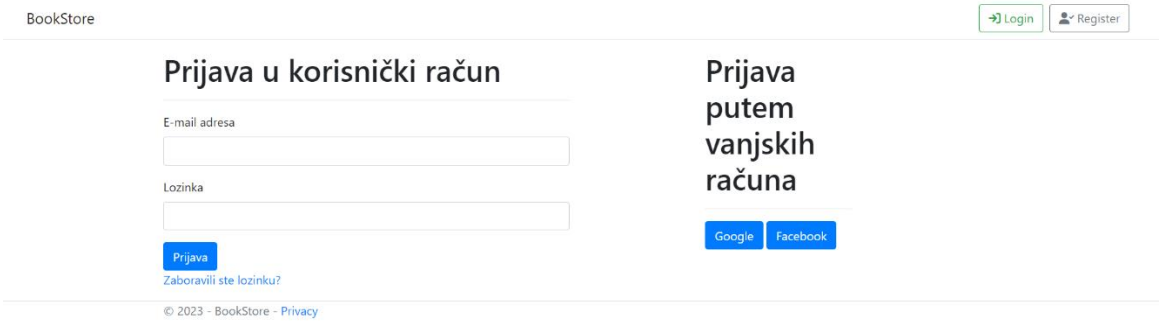
Podaci koji su neophodni za uspješnu registraciju jednaki su za oba tipa, a oni uključuju unos: adrese elektroničke pošte, lozinke te potvrde lozinke (Slika 2).

Registracija nije moguća u slučaju kada u bazi podataka već postoji korisnik s unesenom adresom elektroničke pošte. Za uspješnu registraciju, korisnička lozinka mora zadovoljavati određene uvjete: mora se sastojati od minimalno šest znakova, mora sadržavati barem po jedno malo slovo, veliko slovo, brojku te barem jedan znak koji nije alfanumerički. Lozinka i potvrda lozinke se moraju podudarati.

Slika 2. Prikaz sučelja za registraciju

Ako su svi uvjeti zadovoljeni, korisnik dobiva potvrdu o uspješnoj registraciji te ga se preusmjerava na stranicu za prijavu. U budućnosti je moguće proširiti ovu funkcionalnost tako

da, prije nego što je prijava uopće omogućena, korisnik mora verificirati svoj korisnički račun adresom elektroničke pošte. U trenutnoj fazi rada aplikacije, ova funkcionalnost se simulira mehanizmom *Logging*. Naime, nakon što korisnik unese sve nužne podatke za registraciju, generira se jedinstveni *URL* koji omogućuje potvrdu identiteta preko adrese elektroničke pošte. Za uspješnu prijavu u sustav korisnik mora biti registriran. Podaci koji se traže prilikom prijave su adresa elektroničke pošte i lozinka. Korisnicima se omogućuje i prijava putem korisničkih računa s vanjskih poslužitelja, npr. Google i Facebook, te im se u tom slučaju automatski dodjeljuje uloga kupca. Izgled sučelja za prijavu u sustav prikazan je na Slika 3.



Slika 3. Sučelje za prijavu u sustav

Korisnici koji su se prijavili preko vanjskog poslužitelja imaju mogućnost kreiranja i lokalnog računa preko sučelja koje omogućuje postavljanje interne lozinke. E-mail adresa za internu prijavu jednaka je onoj za vanjsku.

Ako korisnik zaboravi svoju e-mail adresu ili zaporku, platforma omogućava resetiranje, tj. ponovnu uspostavu korisničkih podataka. U trenutnoj fazi razvoja platforme, prethodno spomenuta funkcionalnost je implementirana s pomoću *Logging* sustava koji je detaljnije pojašnjen u poglavlju 3.6. Kako se aplikacija bude razvijala, ova funkcija se može proširiti tako da se link za stranicu na kojoj se izvršava resetiranje korisničkih podataka šalje na korisnikovu adresu elektroničke pošte. Svi registrirani korisnici ove aplikacije imaju mogućnost promjene svoje zaporke po osobnoj želji.

4.4. Autorizacija korisnika

Prilikom inicijalnog pokretanja aplikacije, ako je tablica unutar koje se pohranjuju korisničke uloge prazna, automatski se kreiraju tri početne uloge: administrator platforme, kupac te prodavač, kao i korisnici koji obnašaju te uloge.

```
public static async Task SeedUsersAndRolesAsync(IApplicationBuilder
builder)
{
  using (var applicationservices =
builder.ApplicationServices.CreateScope())
  {
```

```
    var                          roleManager                          =
applicationservices.ServiceProvider.GetRequiredService<RoleManager<I
dentityRole>>();
    if (!await roleManager.RoleExistsAsync(UserRoles.Admin))
    {
      await                                roleManager.CreateAsync(new
IdentityRole(UserRoles.Admin));
    }
    var                          userManager                          =
applicationservices.ServiceProvider.GetRequiredService<UserManager<A
pplicationUser>>();
    if (await userManager.FindByEmailAsync("admin@admin.com") == null)
    {
      var newAdminUser = new ApplicationUser()
      {
        Email = "admin@admin.com",
        EmailConfirmed = true,
        UserName = "Admin"
      };
      await userManager.CreateAsync(newAdminUser, "@Dmin123");
      await userManager.AddToRoleAsync(newAdminUser, UserRoles.Admin);
    }
  }
}
```

Kôd 4. Metoda za kreiranje korisničke uloge administratora te pripadajućeg korisnika

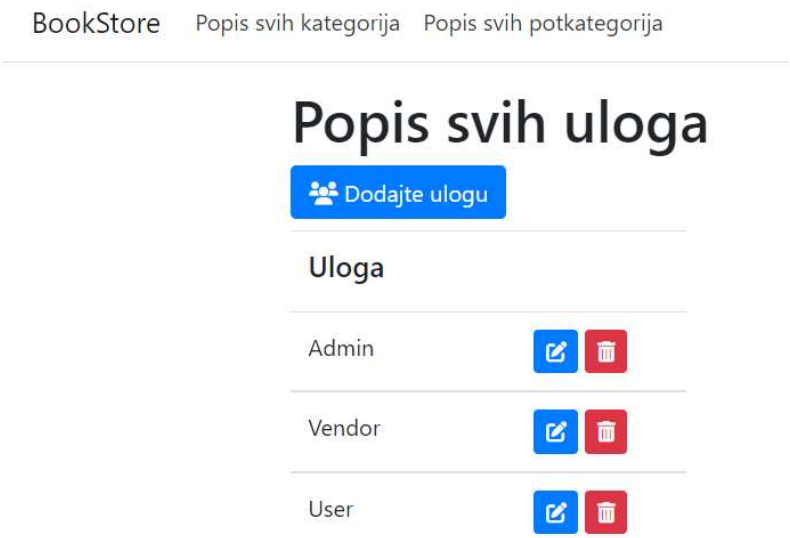
Asinkronom metodom *SeedUsersAndRolesAsync* se kreiraju korisničke uloge i inicijalni korisnici. Kôd 4 prikazuje primjer kreiranja admistratorske uloge i pripadajućeg korisnika. Ova metoda je dio *AppDbInitializer* klase. Kôd koristi *Dependency Injection* oblikovni obrazac koji upravlja zavisnostima između objekata tako da ih „ubrizgava“ ili prosljeđuje klasi, umjesto da klasa sama kreira te objekte.

Autorizacija, točnije, ograničavanje pristupa unutar MVC obrasca se postiže atributom [Authorize], koji se deklarira unutar upravitelja čije se akcije, pa tako i njima pripadajuće poglede, želi zaštititi od neovlaštenog pristupa. Atribut [Authorize] može stajati sam za sebe, bez dodatnih parametara, ali može i sadržavati parametre vezane uz ulogu ili korisničko ime. U slučaju kada parametri nisu postavljeni, pristup upravitelju i svim njegovim svojstvima i akcijama je dopušten svim prijavljenim korisnicima.

4.4.1. Korisnička uloga administratora

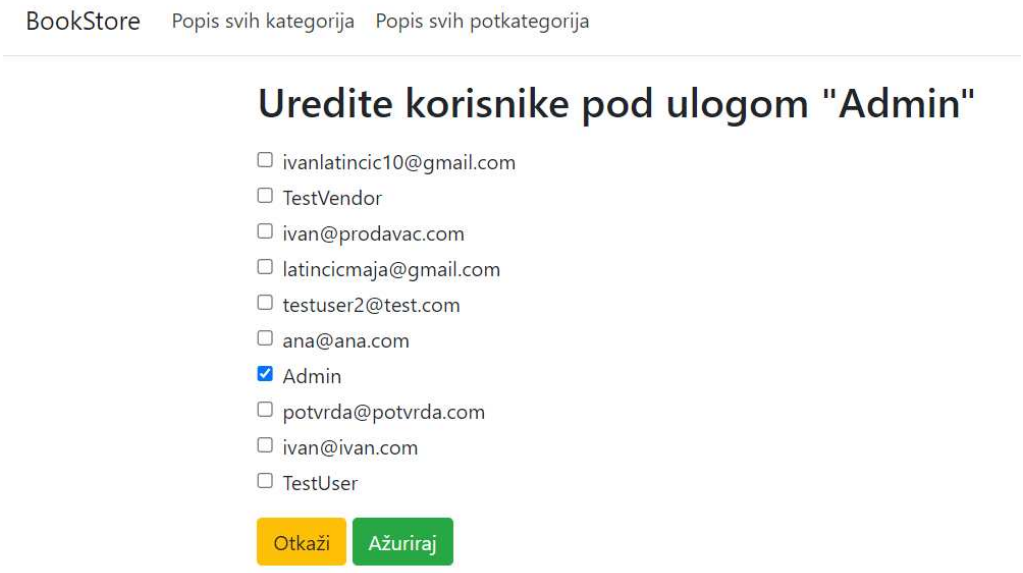
Za razliku od ostalih uloga, administratora aplikacije nije moguće kreirati registracijom, već samo dodjelom posebnih ovlasti unutar sučelja *Roles*, čiji je pristup dostupan samo administratoru aplikacije.

Jedna od funkcionalnosti koje administrator aplikacije ima na raspolaganju je dodavanje novih uloga te uređivanje postojećih (Slika 4).



Slika 4. Sučelje za popis postojećih te dodavanje novih uloga

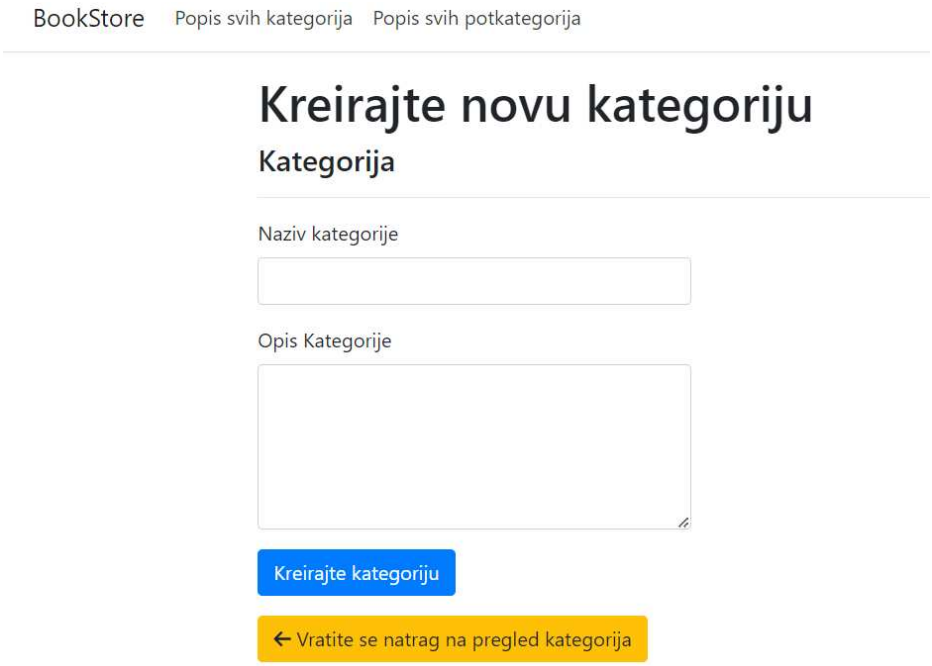
Dodatna funkcionalnost je i dodavanje te brisanje korisnika iz određenih uloga korištenjem okvira za izbor (engl. *checkbox*). Slika 5 prikazuje uređivanje korisnika pod korisničkom ulogom Admin.



Slika 5. Dodavanje i brisanje korisnika iz pojedine uloge

Sljedeća funkcija koju administrator obnaša je kreiranje kategorija i potkategorija, koje predstavljaju književne žanrove i podžanrove unutar kojih će sve knjige, koje će biti na prodaju, biti svrstane. Za svaku kategoriju i potkategoriju, potrebno je unijeti nekoliko obaveznih podataka, kao što su: naziv i kratki opis. Uz to, za potkategoriju je potrebno pomoću padajućeg

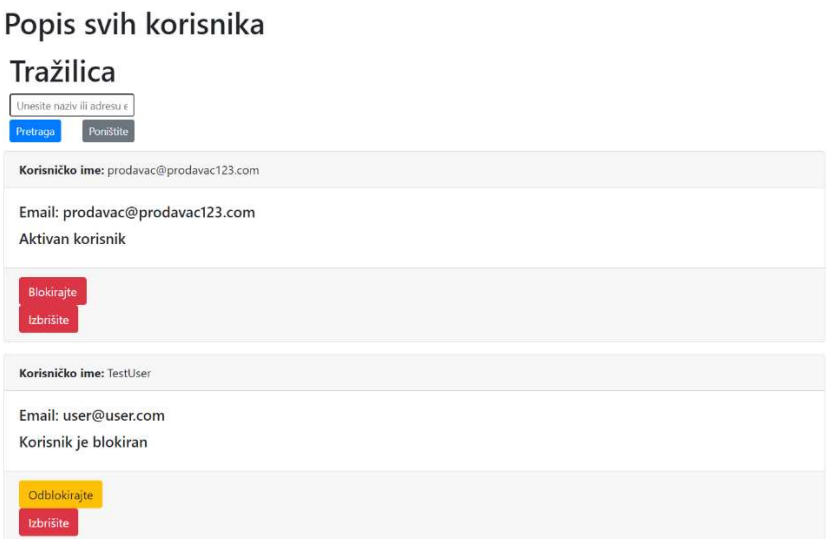
izbornika odabrati kojoj kategoriji ona pripada. Sučelje za kreiranje nove kategorije je prikazano na Slika 6. Podaci koji su obavezni za unos u kodu su označeni atributom *Required*. Administrator ima mogućnost izvedbe *CRUD* operacija nad kreiranim kategorijama i potkategorijama.



Slika 6. Sučelje za kreiranje nove kategorije

Popis svih trenutno registriranih korisnika je poseban pogled unutar aplikacije i dostupan je samo administratoru (Slika 7).

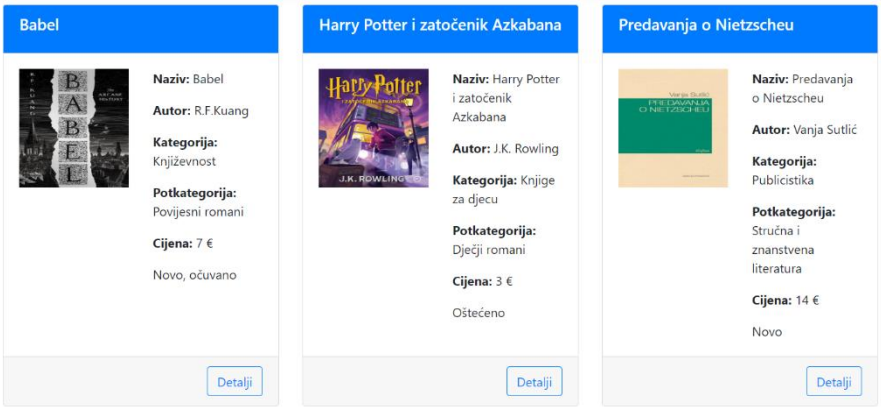
Na samome vrhu pogleda dostupna je tražilica koja administratoru omogućava pretraživanje željenih korisnika prema njihovom korisničkom imenu ili adresi elektroničke pošte. Kao rezultat pretraživanja, administratoru se prikazuju korisničko ime i adresa elektroničke pošte svakog korisnika. Uz pregled podataka, administrator ima mogućnost privremenog blokiranja pojedinog korisnika, čime mu onemogućava prijavu u sustav. Također, postoji opcija brisanja, tj. deaktivacije korisničkog računa pojedinog korisnika.



Slika 7. Popis svih korisnika

4.4.2. Korisnička uloga prodavača

Svaki prodavač ima mogućnost postavljanja oglasa za svoje proizvode. Unutar korisničkog izbornika, jedna od dostupnih stavki je „Popis mojih proizvoda“, čijim odabirom se izlistavaju svi proizvodi korisnika koji je trenutno prijavljen u sustav te posebna tipka pod nazivom „Dodaj novi proizvod“, kojom prodavači postavljaju nove oglase. Prilikom kreiranja oglasa, prodavači moraju ispuniti formular sa svim potrebnim podacima za izvršavanje ove operacije, kao što su: naziv, autor, jezik, fizičko stanje proizvoda (npr. „korišteno“), cijena, slika, opis te kategorija i potkategorija unutar koje je određeni proizvod svrstan. Slika 8 prikazuje pregled popisa proizvoda prodavača koji je trenutno prijavljen u aplikaciju.



Slika 8. Popis prodavačevih proizvoda

4.4.3. Korisnička uloga kupca

Popis svih proizvoda, kao i popis proizvoda po kategorijama, tj. potkategorijama, dostupan je svim korisnicima na uvid, bez obzira na njihovu korisničku ulogu. No, dodavanje proizvoda u košaricu i njihova kupnja nisu mogući bez registracije korisnika kao kupca.

Unutar podnožja svakog oglasa nalazi se tipka koja omogućava korisnicima dodavanje proizvoda u svoju košaricu. Kada se određeni proizvod nađe u košarici nekog kupca, tipka za dodavanje proizvoda više nije dostupna te biva zamijenjena oznakom „Proizvod trenutno nije dostupan“. Korisnička košarica za svaki dodani proizvod sadrži: naziv, količinu, koja je podrazumijevano jedan, te cijenu. Pored svakog proizvoda nalazi se znak za umanjenje (-) kojim se proizvod briše iz košarice. U podnožju košarice se nalazi ukupan zbroj cijena svih stavki unutar košarice.

Na samom dnu stranice, smještene su dvije tipke:

1. „Nastavi kupnju“: korisnika se preusmjerava na stranicu s popisom svih trenutno postavljenih oglasa
2. „Završi kupnju“: korisnika se preusmjerava na stranicu s unosom podataka za dostavu.

Prikaz košarice nalazi se na Slika 9.

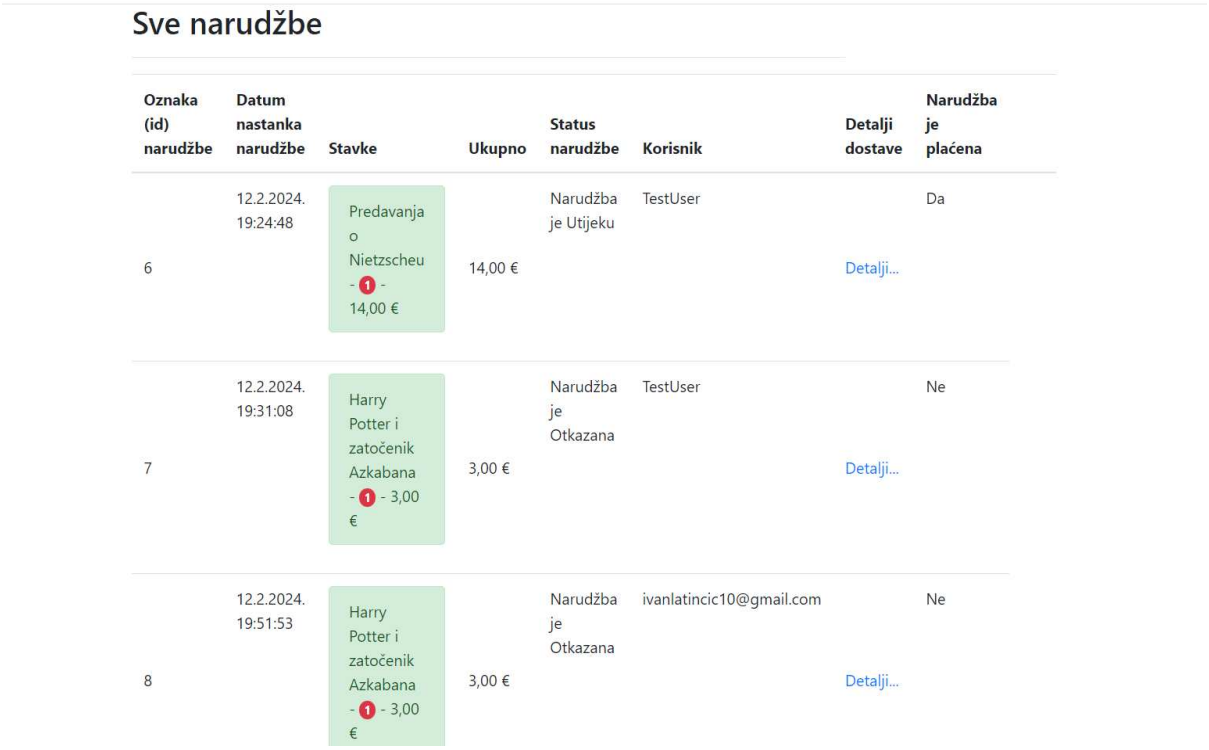
Slika 9. Prikaz košarice

Za dostavu je potrebno unijeti: ime, prezime, adresu, grad, poštanski broj, županiju te broj telefona. Korisnik ima dvije opcije kod odabira plaćanja svoje pošiljke: pouzeće i uplata na račun. Svi podaci nužni za uplatu na bankovni račun platforme, kao što su primatelj (naziv platforme), bankovni račun primatelja, model i poziv na broj primatelja, nalaze se unutar zasebnog pogleda pod nazivom „O nama“. U budućnosti postoji mogućnost realizacije sučelja za kupnju kreditnim karticama, što znači da će korisnici moći unijeti sve relevantne podatke o svojim kreditnim karticama za uspješno izvođenje transakcije. Za dostavu pošiljke zadužen je prodavač.

4.5. Sučelje za prikaz narudžbi

Svaka narudžba se sastoji od: identifikacijske oznake, datuma i vremena nastanka narudžbe, statusa narudžbe, stavki narudžbe, korisnika koji je napravio tu narudžbu, statusa plaćanja te podataka za dostavu (Slika 10). Mogući statusi narudžbe su: u tijeku, izvršena, otkazana. Prilikom kreiranja narudžbe, njeno stanje se automatski bilježi kao „u tijeku“. Sučelje za pregled narudžbi dostupno je i korisnicima s ulogom kupac, kao i administratoru, no razlika je

u podacima koje korisnici imaju na uvid te različitim operacijama koje se mogu obavljati nad narudžbama. Prva razlika je da administrator platforme ima uvid u sve narudžbe, bez obzira na njihovog naručitelja, dok se kupcima prikazuju samo one narudžbe koje su oni napravili. Kupci također imaju mogućnost otkazivanja narudžbe klikom na tipku „Cancel“, pri čemu se pod poljem za stanje odabrane narudžbe bilježi da je narudžba otkazana. Administrator ima na raspolaganju dvije dodatne tipke: tipka za oznaku da je korisnikova uplata sjela na bankovni račun platforme te tipka za storniranje narudžbe, za slučaj da administrator nije zaprimio uplatu u razdoblju od 5 radnih dana.



Slika 10. Pregled sučelja za narudžbe

5. ZAKLJUČAK

Na hrvatskom tržištu još uvijek ne postoji komercijalna C2C (engl. *customer to customer*) mrežna platforma s *multi-vendor* modelom poslovanja i uskom specijalizacijom na rabljene knjige. Ova neistražena strana tržišta, s poslovne strane, ima mnoštvo potencijala za daljnji razvoj te se, potaknuta ovom činjenicom, razvila ideja za kreiranje ove platforme. Tržišna potreba za rabljenom robom i funkcionalnim, interaktivnim aplikacijama koncipiranim na ovaj način, raste iz dana u dan.

Ova mrežna platforma temeljena je na principu postojanja tri vrste korisnika, od kojih svatko obnaša različite korisničke funkcije te sukladno tome ima pristup različitim mrežnim stranicama. Razvijena aplikacija omogućuje svim privatnim osobama postavljanje vlastitih oglasa za prodaju rabljenih knjiga, kao i njihovu kupnju, uz niz mnogih drugih funkcionalnosti dostupnih putem interaktivnog i intuitivnog korisničkog sučelja. Administratori mogu kreirati kategorije, upravljati korisnicima, dodjeljivati uloge te pregledavati i kontrolirati narudžbe, uz dodatne funkcionalnosti. Glavni zadatak prilikom razvoja platforme bio je osiguravanje konzistentnosti podataka unutar baze podataka te omogućavanje adekvatnog rada prilikom istodobnog pristupa velikog broja korisnika aplikaciji.

Uspješnu izradu ove aplikacije omogućile su različite tehnologije i alati, poput prezentacijskih jezika HTML, CSS i Bootstrap, koji su zaslužni za izradu intuitivnog korisničkog sučelja, zatim okvir za objektno-relacijsko mapiranje otvorenog koda Entity Framework te sustav članstva pod nazivom *Identity*. Poseban naglasak stavljen je na razvojni okvir .NET 6 te na MVC obrazac oblikovanja.

Razvijena aplikacija pruža bazično rješenje trenutnog problema, no kroz daljnja proširenja i nadogradnje moguće je značajno poboljšati učinkovitost samoga sustava, na primjer, implementacijom sustava automatskog prebacivanja uplate na račun prodavača. U trenutnom stadiju poslovanja platforme, proces plaćanja prodavaču se odvija izvan same aplikacije. Nadalje, uplate su ograničene na gotovinsko plaćanje prilikom preuzimanja ili bankovnim transferom. S ciljem jačanja konkurentske pozicije platforme i povećanja učinkovitosti poslovanja, preporučuje se uvođenje dodatnih opcija plaćanja, poput plaćanja servisom elektroničnog plaćanja (npr. PayPal) te sustava za plaćanje kreditnim karticama (engl. *Credit Card Payment Getaway*). Pored trenutnih funkcionalnosti, potrebno je dodatno poraditi i na vizualnom doživljaju same aplikacije, koji je trenutno nedovoljno atraktivan i dosljedan. Implementacija alata poput JavaScripta, AJAX poziva ili AngularJS-a može značajno unaprijediti korisničko iskustvo te pristupačnost same aplikacije. Integracijom navedenih tehnologija može se povećati razina dinamičnosti pa stoga i brzina učitavanja što će pridonijeti

poboljšanom ukupnom korisničkom dojmu. Također, sva poboljšanja omogućit će bolje interaktivne mogućnosti, čime se potiče angažman korisnika i olakšava navigacija kroz funkcionalnosti aplikacije.

LITERATURA

1. Agrawal, A., (2023), Common Language Runtime (CLR) in C#, https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/ (pristupljeno 10.7.2024.)
2. Anderson, R., (2016), Adding a controller, https://jakeydocs.readthedocs.io/en/latest/tutorials/first-mvc-app/adding-controller.html (pristupljeno 20.10.2023.)
3. Chand, M.,(2022), What Is New In .NET 6.0., https://www.c-sharpcorner.com/article/what-is-new-in-net-6-0/ (pristupljeno 9.10.2023.)
4. Galloway, J., (2023), Introduction to ASP.NET Identity, https://learn.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity (pristupljeno 29.10.2023)
5. Garcia, D. J., (2019), The History of ASP.NET – Part II (Covers ASP.NET MVC), https://www.dotnetcurry.com/aspnet/1493/aspnet-history-part-2-mvc (pristupljeno 10.7.2024.)
6. Koirala, S.,(2013), Unit of Work Design Pattern, https://www.codeproject.com/articles/581487/unit-of-work-design-pattern (pristupljeno 25 10 2023.)
7. Mezei, R. A., (2023), *Introduction to the Development of Web Applications Using ASP .Net (Core) MVC*, Springer Cham, p. 144;163
8. Microsoft, What is .NET Framework, https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework (pristupljeno 15.10.2023.)
9. Microsoft, What is ASP.NET, https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet (pristupljeno 10.7.2024.)
10. Odyssey, D. N., How Web Works & ASP.Net MVC fits into Web Application Development, https://www.dotnetodyssey.com/asp-net-mvc-5-free-course/how-web-works-asp-net-mvc-fits-web-application-development/ (pristupljeno 24.10.2023.)
11. Reenskaug, T., (1979), MODELS - VIEWS – CONTROLLERS, https://folk.universitetetioslo.no/trygver/1979/mvc-2/1979-12-MVC.pdf (pristupljeno 13.10.2023.)
12. Rout, P., (2022), Entity Framework Code First Approach - Dot Net Tutorials, https://dotnettutorials.net/lesson/introduction-to-entity-framework-code-first-approach/ (pristupljeno 22.10.2023.)
13. Saraç, A., (2023), What is Repository Pattern?, https://www.linkedin.com/pulse/what-repository-pattern-alper-sara%C3%A7/ (pristupljeno 25.10.2023.)
14. SolarWinds Loggly, (2022), .NET Logging Basics, https://www.loggly.com/ultimate-guide/net-logging-basics/ (pristupljeno 30.10.2023.)
15. Smith, S. & Addie, S., (2023), Handle requests with controllers in ASP.NET Core MVC, https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/actions?view=aspnetcore-7.0 (pristupljeno 20.10.2023.)
16. Smith, S. & Brock, D., (2022), Views in ASP.NET Core MVC, https://learn.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-7.0&WT.mc_id=dotnet-35129-website (pristupljeno 19.10.2023.)
17. Troelsen, A., Japikse, P., (2022), *Pro C# 10 with .NET 6*, Apress, p. 3224
18. TutorialsTeacher, .NET Core Overview, https://www.tutorialsteacher.com/core/dotnet-core (pristupljeno 15.10.2023.)

19. W3Schools, HTTP Request Methods, https://www.w3schools.com/tags/ref_httpmethods.asp (pristupljeno 20.10.2023.)
20. Walther, S., (2008), The Evolution of MVC, https://stephenwalther.com/archive/2008/08/24/the-evolution-of-mvc (pristupljeno 13.10.2023.)
21. ZZZ Projects, (2023), EF Core Migrations, https://www.learnentityframeworkcore.com/migrations (pristupljeno 23.10.2023.)
22. ZZZ Projects, (2023), What is Entity Framework Core, https://www.learnentityframeworkcore.com/ (pristupljeno 22.10..2023.)

SAŽETAK

Unutar ovog rada opisan je razvoj i priprema za produkcijski rad *multi-vendor* internet trgovine rabljenih knjiga korištenjem oblikovnog obrasca model-pogled-upravitelj (engl. *Model-View-Controller*; MVC) unutar .NET 6 tehnologije. Primarni cilj je detaljno opisati i prikazati kroz praktičnu primjenu svih mogućnosti koje tehnologija ASP.NET MVC pruža, uz potporu niza drugih korištenih alata i programskih okvira. Primjenom sustava ASP.NET Identity, omogućeno je razgraničenje pristupa različitim dijelovima razvijene aplikacija. Dakle, implementacijom korisničkih uloga (engl. *roles*): administrator, korisnik (kupac) i prodavač, svaki korisnik, ovisno o njegovoj ulozi, ima drugačiji doživljaj same aplikacije te može izvršavati određene jedinstvene funkcije. Prodavačima je omogućeno postavljanje vlastitih prodajnih oglasa, s njihovim ažuriranjem i brisanjem, dok kupci imaju mogućnost kupnje i narudžbe željenih proizvoda. Neke od funkcija koje korisnička uloga administratora obnaša, uključuju: izvršavanje *CRUD* operacija nad kategorijama i potkategorijama, dodavanje novih i brisanje iz baze podataka postojećih korisničkih funkcija, dodjeljivanje novih te oduzimanje postojećih korisničkih uloga određenim korisnicima, blokiranje korisnika te pregled narudžbi, kao i njihovo potencijalno storniranje.

Ključne riječi: *multi-vendor*, web shop, MVC, .NET 6, Entity Framework Core, Microsoft Identity, MS SQL, C#

SUMMARY

This paper describes the development and preparation for production work of a multi-vendor online second-hand book store using the Model-View-Controller (MVC) form within the .NET 6 technology. The primary goal is to describe in detail and show through practical application all the possibilities that .NET MVC technology provides, with the support of a number of other used tools and programming frameworks, when creating a web application. By applying the ASP.NET Identity system, it is possible to limit the access to different parts of the developed application to specific users. Therefore, by implementing user roles, such as administrator, user (buyer) and seller, each user, depending on their role, has a different experience of the application itself and many perform certain unique functions. Sellers are enabled to set up their own sales ads, as well as update and delete them, while customers have the option of buying and ordering the desired products. Some of the functions performed by the administrator user role include: performing CRUD operations (Create, Read, Update, Delete) on categories and subcategories, adding new, as well as deleting existing users from the database, assigning new and withdrawing existing user roles from specific users, blocking of desired users and review of orders, as well as their potential cancellation.

Keywords: multi-vendor, web shop, MVC, .NET 6, Entity Framework Core, Identity, MS SQL, C#