Izrada internet trgovine upotrebom ASP.NET MVC 5 i Entity Frameworka

Huljak, Marko

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:046184

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-23**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij računarstva

IZRADA INTERNET TRGOVINE UPOTREBOM ASP.NET MVC 5 I ENTITY FRAMEWORKA

Diplomski rad

Marko Huljak

Osijek, 2020.

SADRŽAJ

1. UVOD 1
2. POSTOJEĆA PROGRAMSKA RJEŠENJA ZA INTERNET TRGOVINE 3
2.1. Karakteristike 3
2.2. Klasifikacija 4
2.3. Pregled i analiza 5
3. MODEL PROGRAMSKOG RJEŠENJA ZA PRODAJU RAČUNALA I RAČUNALNE OPREME 10
3.1. Koncept programskog rješenja izrađenog u okviru ovog rada 10
3.2. Zahtjevi na programsko rješenje 11
3.3. Arhitektura programskog rješenja 12
3.3.1. MVC 13
3.4. Korištene tehnologije i razvojni alati 14
3.4.1. Tehnologije za izradu korisničkog sučelja 15
3.4.2. ASP.NET MVC 15
3.4.3. Entity Framework 16
4. IMPLEMENTACIJA PROGRAMSKOG RJEŠENJA ZA PRODAJU RAČUNALA I RAČUNALNE OPREME 17
4.1. Baza podataka programskog rješenja 17
4.2. Korisnička strana programskog rješenja 18
4.2.1. Registracija i prijava 19
4.2.2. Korisnički profil 20
4.2.3. Kategorije 21
4.2.4. Tražilica proizvoda 22
4.2.5. Detalji proizvoda 23
4.2.6. Košarica 23
4.2.7. Završetak kupovine 24
4.2.8. Povijest narudžbi 25
4.3. Administratorska strana programskog rješenja 26
4.3.1. Pregled registriranih korisnika 27
4.3.2. Upravljanje proizvodima 28
4.3.3. Upravljanje kategorijama 29

4.3.4. Upravljanje narudžbama 29
4.4. Demonstracija virtualnog plaćanja 30
4.5. Implementacija sigurnosnog mehanizma 32
5. ZAKLJUČAK 33
LITERATURA 35
SAŽETAK 37
ABSTRACT 38
ŽIVOTOPIS 39
PRILOZI 40

1. UVOD

U današnje vrijeme, gotovo svaka veća trgovina, pa čak i neke manje, imaju svoju web stranicu. Osim toga što može pružiti novi zamah razvoju poslovanja u trgovinama, internet trgovina je pogodna i za kupce. Na njoj se nalaze razni sadržaji i katalozi pomoću kojih trgovine olakšavaju korisnicima pristup informacijama o proizvodima, uslugama i cijenama što rezultira širenjem poslovanja, a samim time i profitom. Osim toga, internet trgovine koriste mrežne platne sustave koji se odnose na virtualne novčane transakcije, što je revolucioniralo obradu plaćanja smanjenjem papirologije, transakcijskih troškova i troškova rada. Može se reći da internet trgovina djeluje kao simbioza između trgovine i kupaca i time se brzo učvršćuje kao prihvaćena i uobičajena poslovna paradigma.

U počecima internet trgovine, prema [1], mnoge trgovine, primjerice Toys "R" Us, koje su odbijale implementirati u svoje poslovanje programska rješenja internet trgovine, imale su značajne zaostatke nad konkurencijom, što je rezultiralo smanjenjem profita ili čak zatvaranjem poduzeća. Dok s druge strane, trgovine koje su u vrlo ranoj fazi implementirale ovakva rješenja, jako su se brzo razvijale i povećavale svoj profit. Čak i trgovine koje nisu fizički postojale, odnosno nisu imale fizički poslovni objekt (eng. *Brick and Mortar*), nego su prodavale svoje proizvode i/ili usluge samo putem interneta, su ubrzo postale popularne i cijeli svijet ih je počeo koristiti.

Današnja programska rješenja internet trgovina dijele se na platforme koje omogućuju razvijanje i upravljanje internet trgovinama te na web stranice. Glavni predstavnik takvog rješenja kao platforme je Magento dok je općepoznati predstavnik rješenja kao web stranice Amazon, koji bilježi najviše online prodaja u svijetu. Preko 63% svih kupovina počinju online, neovisno o tome razgledavaju li kupci proizvode online pa nakon toga fizički posjete trgovinu ili naruče direktno putem interneta. Prometu ostvarenom putem internet trgovanja značajno doprinose internet trgovine računala i računalne opreme. Globalni primjer takve trgovine je Newegg, koji osim standardnih filtera nudi i brojne filtere tehničkih specifikacija koji se pojavljuju ovisno o kategoriji proizvoda prema kojom se pretražuje.

Zadatak ovog diplomskog rada je izraditi web aplikaciju internet trgovine proizvoljnog sadržaja i prilagodljivog dizajna korisničkog sučelja koristeći ASP.NET MVC i Entity Framework tehnologije. Programsko rješenje se treba sastojati od administratorske i korisničke strane kojima se

pristupa putem registracije i prijave. Korisnici moraju imati mogućnost uređivanja vlastitog profila, imati uvid u povijest naručenih i kupljenih proizvoda kao i stavljanje u košaricu te kupovinu istih. Osim detalja proizvoda, korisnicima mora biti omogućena i galerija slika tog proizvoda. Proizvodi trebaju biti grupirani u kategorije te im se treba omogućiti pristup putem tražilice. Košarica mora imati mogućnost upravljanja količine proizvoda unutar nje. Prilikom završetka kupovine, korisnik treba biti preusmjeren na testno okruženje PayPala. Administrator mora imati mogućnost upravljanja proizvodima i kategorijama pod kojima se proizvodi prikazuju te uvid u osnovne informacije registriranih korisnika.

Programsko rješenje internet trgovine izrađeno prema navedenim zahtjevima, većinu funkcionalnosti, osim registriranim korisnicima, omogućuje i gostima. Svim korisnicima je pružen izbor plaćanja – prilikom pouzeća ili putem platnog servisa. Administratoru je pružena i mogućnost upravljanja narudžbama.

Nastavak rada organiziran je na sljedeći način: pregled postojećih rješenja dan je u drugom poglavlju, dok je u trećem predstavljen model aplikacije izrađene u okviru ovog diplomskog rada zajedno sa zahtjevima na istu, arhitekturom i tehnologijama pomoću kojih je realizirana. Zatim, u četvrtom poglavlju je objašnjena implementacija aplikacije prema principu zahtjev-rješenje, dok je u petom poglavlju ovog rada dan zaključak i ukratko predstavljen koncept navedene aplikacije.

2. POSTOJEĆA PROGRAMSKA RJEŠENJA ZA INTERNET TRGOVINE

Danas se pojam internet trgovine kao programskog rješenja više ne odnosi samo na internet stranicu pojedine trgovine, već i na sva ponuđena rješenja koja dolaze uz nju, kao što su platforme, odnosno programski okviri (eng. *frameworks*). Takvi primjeri su WooCommerce i Shopify platforme, usmjerene na prilagodljivu (eng. *responsive*) i skalabilnu izradnju internet trgovina, kao i upravljanje istima. S obzirom da danas ima mnogo klijenata koji zahtijevaju programsko rješenje sa dosta specifičnim dizajnom i značajkama koje nisu prisutne u navedenim platformama, ovaj rad je usredotočen na izradu aplikacije koja uključuje isključivo web stranicu. Budući da je takvo programsko rješenje jeftino i relativno brzo za izradu, navedena aplikacija se nameće kao optimalno rješenje za potrebe lokalnih trgovina.

2.1. Karakteristike

Programska rješenja internet trgovine pružaju široki raspon značajki (eng. *features*) i funkcija koje pojednostavljuju i podržavaju mnoge aspekte trgovanja putem interneta. Neke od bitnijih karakteristika takvih programskih rješenja, prema [2], su:

- **Elektroničko plaćanje** – omogućava jednostavno izvršenje virtualnih transakcija udovoljavajući pritom najvišim zahtjevima sigurnosti,
- **Automatizacija obračuna** – uključuje izračunavanje cijena, poreza, cijene otpreme i ostalih troškova kako bi kupci imali predodžbu o tome koliko ukupno trebaju platiti za odabrane proizvode,
- **Dostupnost** – sve internet trgovine dostupne su kupcima 24 sata na dan i to svakim danom u godini,
- **Sveprisutnost** – kupovina putem internet trgovine je omogućena sa bilo koje lokacije uz prisutnost interneta,
- **Marketing** – prodavač bez prevelikih ulaganja korištenjem internet trgovine može lako širiti svoje tržište na globalnoj razini,
- **Služba za korisnike** – kupcu je omogućena podrška internet trgovine tijekom cijelog procesa kupnje putem raznih servisa,
- **Automatizacija skladišta** – prilikom prodaje proizvoda, računi se automatski izrađuju te se ažurira broj dostupnih proizvoda. Na taj način je olakšana evidencija skladišta.

2.2. Klasifikacija

Postoji mnogo čimbenika prema kojima se internet trgovine mogu klasificirati. Kao što je rečeno u [3], najvažnije podjele su prema proizvodima ili uslugama koje nude i prodaju, zatim poslovnim modelima, tj. subjektima s kojima obavljaju transakcije ili čak platformama na kojima su zasnovane.

Uz najtipičniji tip robe koju internet trgovine prodaju, odnosno fizički proizvod, brojne nude digitalne proizvode koji se odnose na predmete u digitalnom formatu uključujući e-knjige, softvere, online tečajeve, itd. Primjer trgovina za prodaju online tečajeva su gotovo sve platforme za internetsko učenje kao što je Coursera. S druge strane, svaki put kada korisnik unajmi nastavnike, savjetnike ili općenito osobu za izvršenje neke usluge putem mrežnih platformi poput Upworka, tada posluje sa internet trgovinama utemeljenim na uslugama.

Trenutno postoje četiri glavne vrste internet trgovine klasificirane na temelju poslovnih modela, odnosno uloga u trgovini:

- **Poduzeće-poduzeće** (eng. *Business-to-Business – B2B*) – model u kojemu poduzeće prodaje svoje proizvode i/ili usluge ostalim poduzećima. Primjer tvrtke koja koristi takav model, prema [4], je Samsung, jedan od najvećih Appleovih dobavljača OLED zaslona u proizvodnji iPhonea,
- **Poduzeće-potrošač** (eng. *Business-to-Consumer – B2C*) – najpoznatiji poslovni model koji podrazumijeva poslovanje organizacije s krajnjim korisnicima, npr. kao što to čini internet trgovina Links,
- **Potrošač-potrošač** (eng. *Consumer-to-Consumer – C2C*) – odnosi se na transakciju između samih potrošača. Primjer ovog modela susreće se na eBayu koji korisnicima dodatno pruža online aukciju,
- **Potrošač-poduzeće** (eng. *Consumer-to-Business – C2B*) – poslovni model usko povezan sa *Crowdfunding* platformom poput Kickstartera u kojoj potrošač financira, odnosno donira novac poslovnom objektu, tj. poduzeću.

Ostale manje zastupljene vrste modela uključuju uloge vlade, menadžera i zaposlenika kako bi definirale više specijaliziranih poslovnih modela za internet trgovine. Međutim, bilo koja od tih vrsta se može smatrati podvrstama četiri glavna modela. Budući da je programsko rješenje izrađeno u okviru ovog diplomskog rada zamišljeno kao manja internet trgovina čija je svrha isključivo

prodaja proizvoda krajnjim korisnicima, isto će se svrstati pod poduzeće-potrošač (B2C) poslovni model.

Platforma internet trgovine se odnosi na sami softver koji pokreće internet stranicu poduzeća koji se bavi trgovanjem. Funkcionalnost košarice je najočitiji primjer značajke koja se nalazi na platformi internet trgovine. Ostale značajke mogu uključivati korisne alate za upravljanje različitim aspektima poslovanja. Platforme se s obzirom na infrastrukturu i postupke postavljanja, mogu podijeliti u sljedeće skupine:

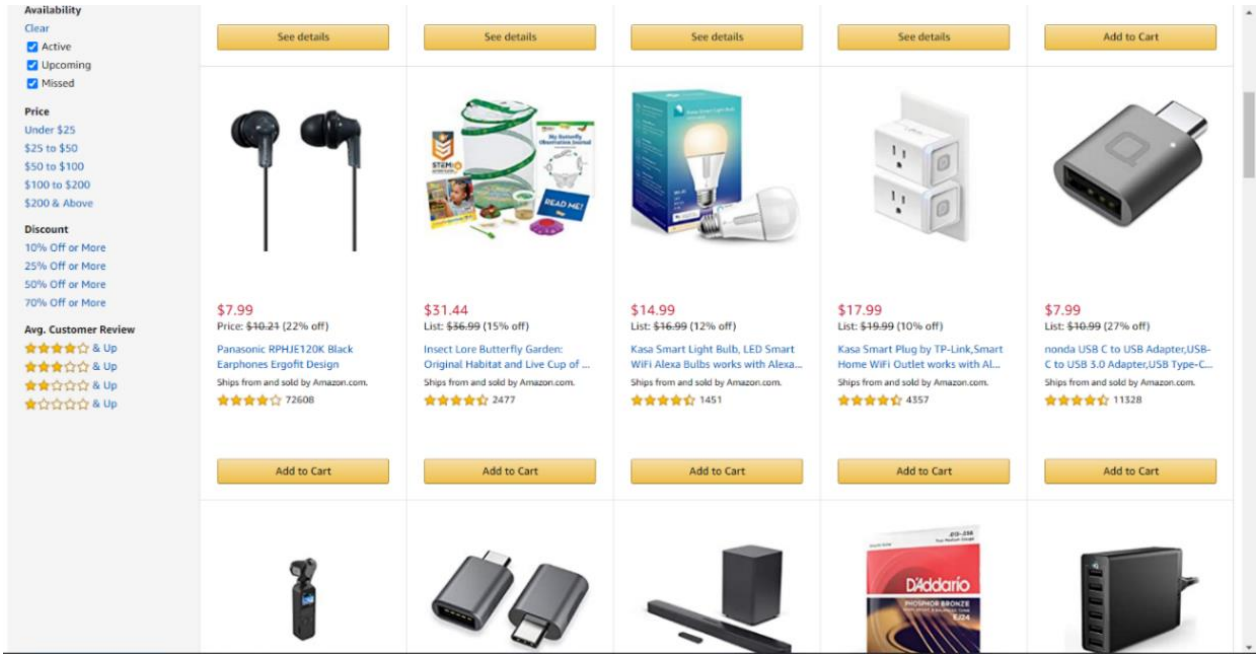
- **Platforma otvorenog kôda** (eng. *Open source*) – kao i bilo koji softver otvorenog kôda, ova se vrsta platforme može koristiti i mijenjati bez plaćanja mjesečne naknade ili naknade za licencu kao što to omogućuje PrestaShop,
- **Platforma softvera kao usluge** (eng. *Software as a service*) – platforma utemeljena u „oblaku“, odnosno softver koji je krajnjim korisnicima dostupan putem web preglednika. Davatelj usluge naplaćuje korisnika ovisno o vremenu i količini korištenih značajki. Ranije spomenute platforme WooCommerce i Shopify pripadaju ovoj skupini,
- **Platforma licenciranog softvera** (eng. *Commercial software*) – platforma s obzirom na plaćanje softvera suprotna platformi otvorenog kôda. Najpoznatiji primjer ove platforme je CS-Cart.

2.3. Pregled i analiza

Postojeće internet trgovine koriste razne trikove kako bi privukli što više kupaca i prodali što više proizvoda. Također su napravljena na način da značajke poput registracije i kupnje budu što lakše i pristupačnije. Počevši od same registracije korisnika, uz standardni način upisivanja korisničkog imena, lozinke i ispunjavanja svih potrebnih polja za izradu računa, mnoge web trgovine imaju funkcionalnost registracije putem društvenih mreža kao što su Facebook, Twitter i sl. Samim time korisnicima se olakšava registracija na način da samo jednim klikom vrlo brzo ispune sva tražeća polja i time povežu svoj postojeći račun sa web trgovinom. To je također pogodno napraviti i iz sigurnosnih razloga kao što je slučaj kada se zaboravi lozinka. Mnoge poznatije internet trgovine služe se svim informacijama koje mogu prikupiti od prijavljenog korisnika (čak i gosta) kao što je npr. povijest pretraživanja te na osnovu toga korisniku ponuditi slične proizvode. Kao što je rečeno u [5], gotovo sva općepoznata globalna programska rješenja poput eBaya, koriste

umjetnu inteligenciju (eng. *Artificial Intelligence - AI*) za stvaranje personaliziranog korisničkog iskustva kupovine. Osim toga, korisnicima su omogućene brojne druge automatizirane značajke poput virtualnog pomagača (eng. *chatbot*) koji asistira u bilo kakvim pitanjima, pritužbama i u otklanjanju problema s kojima se korisnici prilikom kupovine mogu susresti. Primjenjujući ovu inovaciju, tvrtke mogu u kratkom vremenskom roku zadovoljiti potrebe mnogih kupaca.

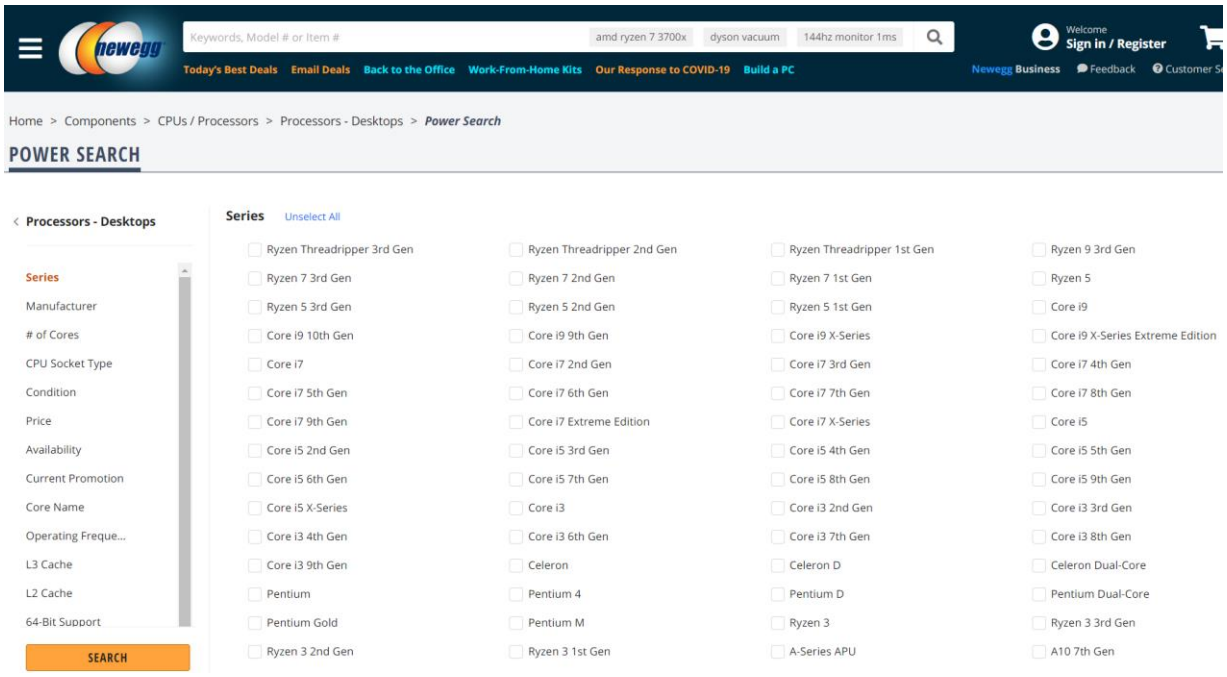
Svjetski poznato programsko rješenje je prikazano na slici 2.1. Riječ je o internet trgovini Amazon, dostupnoj na [6], koja se uz poduzeće-potrošač poslovni model temelji i na potrošač-potrošač modelu. Drugim riječima, korisnicima nudi opciju prodaje vlastitih artikala na njihovoj stranici, za što naravno uzima određenu proviziju.



Sl. 2.1. *Primjer sučelja internet trgovine Amazon*

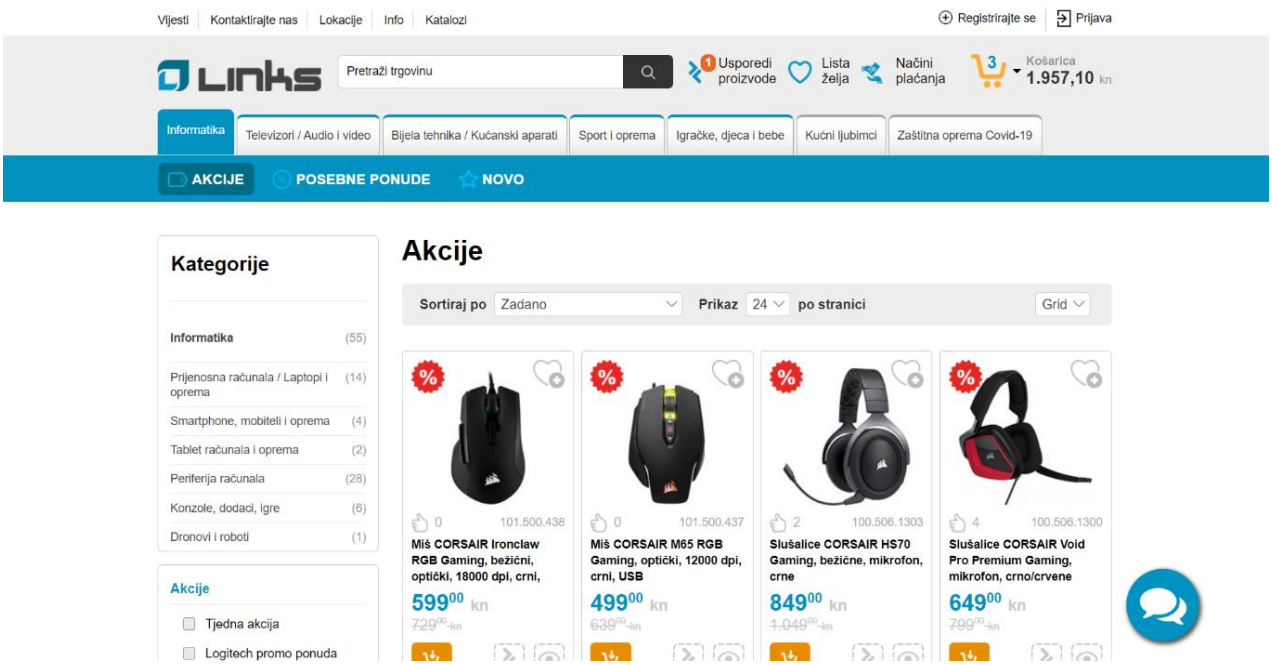
Aplikacija ima mogućnost ocjenjivanja proizvoda kao i dodavanja komentara koji su vidljivi svim ostalim korisnicima koji pregledavaju taj proizvod. Premda nudi brojne značajke i proizvode čime je opravdano najpoznatija internet trgovina u svijetu, mana joj je što nema fiksnu traku zaglavlja na kojoj se nalazi tražilica (kao što je vidljivo na slici). Tako se korisnik prilikom pretrage, ukoliko se nalazi pri dnu stranice, mora uvijek vratiti na početak stranice što bi nekim korisnicima s vremenom postalo iritantno. Također ne sadrži gumb za pomicanje prikaza na sami vrh stranice. Takvi nedostaci nisu prisutni u programskom rješenju izrađenom u okviru ovog diplomskog rada.

Na slici 2.2. je prikazano još jedno programsko rješenje internacionalne razine koje je dostupno na [7]. Za razliku od Amazona, Newegg od proizvoda nudi samo proizvode iz područja tehnologije. Prilikom traženja željenih artikala, uz sve tradicionalne filtere poput okvirne cijene, ocjene, dostupnosti na temelju lokacije i sl., pojavljuju se dodatni filteri tehničkih specifikacija ovisno o kategoriji proizvoda koji se pretražuje. Primjerice ako se odabere kategorija stolnih računala, automatski se pojavljuju filteri koji omogućavaju odabir željenog proizvođača, tehnologije hlađenja, unutarnjih komponenti itd.



Sl. 2.2. *Primjer sučelja internet trgovine Newegg*

Slika 2.3. prikazuje sučelje internet trgovine Links (dostupna na [8]), koje osim gore navedenih značajki ima značajke poput liste želja, usporedbe proizvoda i tražilice. Budući da nudi jednostavan dizajn prikaza proizvoda kao i filtere pretrage, dizajn programskog rješenja izrađenog u okviru ovog diplomskog rada je napravljen po uzoru na ovaj. Links nudi i značajku prikaza svih Links prodavaonica na karti, kako bi korisniku olakšalo odabrati njemu bližu lokaciju i pritom osobno preuzeti kupljeni proizvod.



Sl. 2.3. *Primjer sučelja internet trgovine Links*

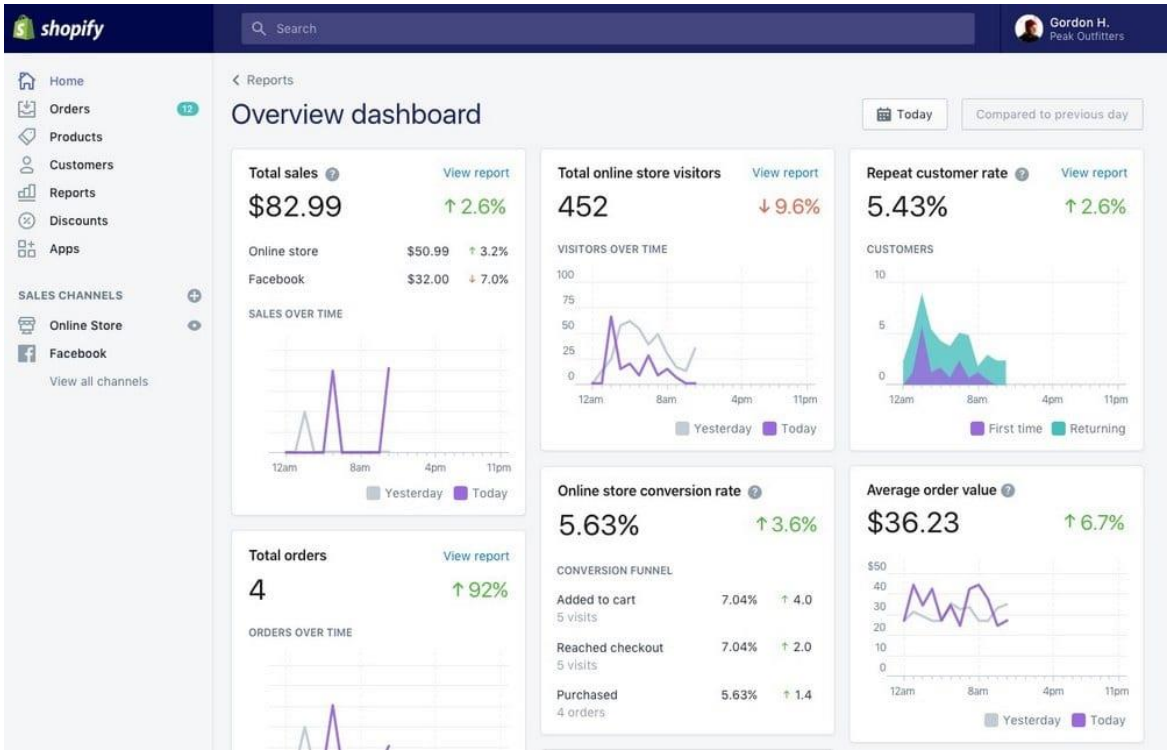
Tablica 2.1. prikazuje usporedbu ranije spomenutih postojećih programskih rješenja internet trgovina. Links za razliku od Amazona i Newegga ima jednostavan i minimalističan dizajn koji se svodi na bolju preglednost ponuđenih proizvoda kao i lakše snalaženje na stranici, što je kupcu od iznimne važnosti. Osim toga, sadrži fiksnu navigacijsku traku kao i gumb za navigiranje prema vrhu stranice što bi trebalo biti prisutno na gotovo svakoj internet trgovini što u konačnici daje bolje korisničko iskustvo (eng. *user experience*) u odnosu na globalna programska rješenja.

Tab. 2.1. Usporedba navedenih internet trgovina

	Amazon	Newegg	Links
Prodaja	Globalna	Globalna	Lokalna
Proizvodi	Razno	Elektronika i oprema	Elektronika i oprema
Poslovnica	✘	✘	✓
Poslovni model	B2C, C2C	B2C	B2C
Korisničko iskustvo	Vrlo dobro	Vrlo dobro	Odlično

Budući da programsko rješenje izrađeno u okviru ovog rada ima i administratorsku stranu aplikacije, radi primjera je bitno navesti i jedno od postojećih programskih rješenja platformi za izradu i

upravljanje internet trgovine kao što je Shopify, dostupno na [9]. Ova platforma (slika 2.4) nudi brojne značajke, a neke od njih su upravljanje sadržajem kao i izrada korisničkog sučelja pomoću ponuđenih predložaka na temelju željenih preferencija. Programsko rješenje izrađeno u okviru ovog diplomskog rada po uzoru Shopify platforme sadrži značajke poput upravljanja kategorijama, proizvodima, narudžbama i uvid u registrirane korisnike.



Sl. 2.4. *Sučelje kontrolne ploče platforme Shopify*

3. MODEL PROGRAMSKOG RJEŠENJA ZA PRODAJU RAČUNALA I RAČUNALNE OPREME

U ovom poglavlju opisani su, prije svega, koncept i ideja internet trgovine izrađene u okviru ovog diplomskog rada. Zatim su predstavljeni zahtjevi koje je nužno zadovoljiti kako bi se takva internet trgovina i ostvarila, te će nakon toga biti opisana i sama arhitektura aplikacije. Na kraju poglavlja će se spomenuti tehnologije i razvojni programski alati pomoću kojih će ovo programsko rješenje biti ostvareno. Zadatak ovog diplomskog rada je izraditi web aplikaciju internet trgovine proizvoljnog sadržaja i responzivnog dizajna uz prisutnost svih tradicionalnih značajki takvog programskog rješenja. Premda je teško po pitanju brojnosti funkcionalnosti kao i njihove optimizacije nadmašiti gigante postojećih programskih rješenja poput Amazona, AliExpressa ili Newegga, rad će biti usmjeren na izradu jeftinog programskog rješenja koje bi zadovoljilo potrebe za neke lokalne trgovine. U ovom slučaju, riječ je o internet trgovini za prodaju računala i računalne opreme jer se takva vrsta proizvoda često naručuje preko interneta zbog dobro opisanih specifikacija. Primjerice, nije potrebno isprobati hoće li komponente odgovarati računalu jer se unaprijed znaju specifikacije kompatibilnosti i performansi.

3.1. Koncept programskog rješenja izrađenog u okviru ovog rada

Glavni cilj ovog programskog rješenja je pružiti potpunu web podršku kojom bi se kupcu omogućilo da virtualno kupuje željene proizvode. Uz to, bitno je prodavaču pružiti sučelje za upravljanje web trgovinom, točnije, proizvodima, kategorijama, narudžbama kao i registriranim korisnicima. Prema tome, programsko rješenje bi bilo podijeljeno na administratorsku i korisničku stranu kojima se pristupa putem registracije i prijave.

Također, pristup aplikaciji bi imali i neregistrirani kupci, odnosno gosti koji za razliku od registriranih nemaju opciju uvida i uređivanja vlastitog profila kao ni uvida u povijest naručenih i kupljenih proizvoda. Još jedna prednost registriranih korisnika je da prilikom kupovine ne moraju popunjavati traženu formu već će ona automatski biti ispunjena podacima koje je korisnik unio prilikom registracije. Prilikom plaćanja, korisnicima će na izbor biti pružene dvije opcije – plaćanje pouzećem ili plaćanje putem platnog servisa, u ovom slučaju PayPala. Ukoliko bi se korisnik odlučio za plaćanje putem virtualne transakcije, za demonstraciju toga će biti preusmjeren na *sandbox* testno okruženje PayPala na kojem je moguće završiti kupovinu.

3.2. Zahtjevi na programsko rješenje

Zahtjevi se dijele na funkcionalne i nefunkcionalne. Funkcionalni zahtjevi su vezani uz funkcionalnost same web trgovine i moraju biti ostvareni. Takvi zahtjevi na korisničku stranu su:

- Kupac mora imati uvid u opis, odnosno specifikacije proizvoda, uključujući galeriju proizvoda,
- Kupcu mora biti omogućeno stavljanje proizvoda u košaricu i kupovina istog,
- Košarica mora omogućiti prikaz svih proizvoda unutar nje, povećanje/smanjenje količine proizvoda te dodavanje/uklanjanje istog,
- Proizvodi moraju biti grupirani po kategorijama, te im se mora omogućiti i pristup pomoću tražilice,
- Prilikom završetka kupovine, korisniku je nužno pružiti različite opcije plaćanja od kojih barem jedna mora biti povezana sa nekim testnim platnim servisom kao što je PayPal Sandbox,
- Registriranim kupcima mora se pružiti sučelje za upravljanje vlastitim profilom kao i sučelje povijesti naručenih i kupljenih proizvoda.

Administratorska strana mora pružiti osnovne značajke za upravljanje web trgovinom. Takve značajke moraju proizaći iz sljedećih funkcionalnih zahtjeva:

- Prodavač mora imati uvid u korisnike i sve njihove informacije potrebne za pošiljku naručenih proizvoda,
- Prodavaču se mora omogućiti uvid i upravljanje narudžbama čiji bi se status automatski morao ažurirati na korisničkoj strani, točnije sučelju povijesti narudžbi,
- Prodavač mora imati mogućnost dodavanja/uklanjanja kategorija i proizvoda kao i mogućnost mijenjanja istih.

Nefunkcionalni zahtjevi su zahtjevi koji ne moraju biti ispunjeni da bi sustav pružao specificiranu funkcionalnost, ali svakako pridonose kvaliteti web sjedišta. Takvi zahtjevi na korisničku i administratorsku stranu su:

- Aplikacija mora zadovoljiti osnovne standarde korisničkog iskustva kao što su preglednost, jednostavnost i intuitivnost,

- Dizajn sučelja mora biti prema suvremenim standardima, a elementi kao i njihov sadržaj, moraju zadovoljavati prilagodljivost s obzirom na rezoluciju uređaja sa kojeg se pristupa aplikaciji,
- Aplikacija mora biti pristupačna putem svih standardnih web preglednika,
- Performanse aplikacije ne smiju narušavati korisničko iskustvo,
- Aplikacija mora biti lako proširiva budućim značajkama,
- Aplikacija mora jamčiti sigurno spremanje korisničkih podataka.

Budući da je zajednički cilj bilo koje internet trgovine privući kupce, te na koncu prodati proizvod, korisničko iskustvo je od velike važnosti koju kao zahtjev svaka web trgovina mora zadovoljiti. Što se tiče sigurnosnih zahtjeva, aplikacija mora biti sigurna jer sadrži privatne podatke korisnika. Takvu razinu sigurnosti je potrebno ostvariti pomoću pouzdanog kriptiranja podataka. Nadalje, sustav mora slijediti općeprihvaćena sigurnosna pravila:

- **Povjerljivost** – samo administrator ima dopuštenje uvida u korisničke podatke (sve osim lozinke) i njihove narudžbe,
- **Integritet** – isključivo korisnici mogu mijenjati svoje osobne podatke,
- **Autentičnost** – nitko ne može pristupiti, mijenjati ili brisati podatke drugih računa.

3.3. Arhitektura programskog rješenja

Postoje razni uzorci arhitekture softvera (eng. *software architectural patterns*), a samo se neki koriste za izradnju internet trgovina. Najosnovniji uzorak današnjice je uzorak slojevite arhitekture (eng. *layered architecture*) u kojemu su komponente logički organizirane u horizontalne slojeve među kojima vlada razdioba brige (eng. *separation of concerns*), odnosno svaki sloj ima svoju ulogu u aplikaciji. Ovaj uzorak nigdje eksplicitno ne određuje koliko slojeva mora sadržavati, a podrazumijeva minimum od tri sloja. Većina lokalnih internet trgovina koriste troslojnu arhitekturu dok one na globalnoj razini koriste višeslojnu jer zahtijevaju puno veću pouzdanost i veću skalabilnost.

Troslojnu (eng. *3-layered*) arhitekturu čini prezentacijski sloj, sloj poslovne logike i sloj baze podataka. Primjerice, prezentacijski sloj internet trgovine služi kako bi reagirao na korisničke zahtjeve koje bi zatim proslijedio sloju poslovne logike gdje bi se npr. izračunao iznos košarice na

temelju cijene proizvoda definirane u sloju baze podataka. Prema tome se može zaključiti da kod ovog uzorka svaki korisnički zahtjev mora proći kroz jedan po jedan sloj – od najvišeg do najnižeg. Mana toga je što za neke slojeve postoje razlozi zašto bi bilo u redu da se taj sloj može preskočiti. Primjer bi bio kada se šalje zahtjev koji nema nikakve koristi od sloja poslovne logike već samo od sloja baze podataka kako bi korisniku prikazao određene podatke.

Brojne internet trgovine koriste interaktivna sučelja koja direktno komuniciraju sa bazom podataka što znači da prilikom neke izmjene podataka gotovo instantno reflektiraju tu promjenu. Takav pristup zahtjeva fleksibilnost slojeva, odnosno drugačiji raspored slojeva arhitekture. Ovdje dolazi do izražaja MVC arhitektura koja se temelji na trokutastom rasporedu koji će se detaljnije pojasniti u sljedećem potpoglavlju. Uostalom, programsko rješenje izrađeno u okviru ovog rada je razvijano pomoću ASP.NET MVC programskog okvira koji je temeljen na MVC arhitekturi. Prema tome, uzorak arhitekture ove aplikacije implicitno postaje MVC.

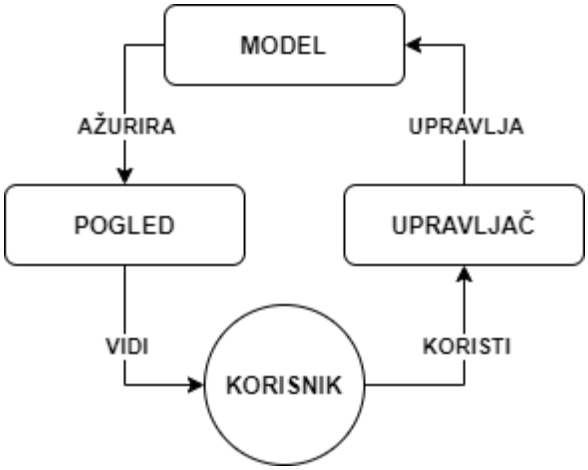
3.3.1. MVC

Model-pogled-kontroler (eng. *Model-View-Controller - MVC*) je uzorak softverske arhitekture koji se najčešće koristi u web aplikacijama. Aplikacija koja koristi ovu arhitekturu je podijeljena u tri glavne logičke komponente: model, pogled i kontroler. Glavna svrha ovakve podijele je da se odvoje reprezentacije informacija od načina na koji se iste prezentiraju i prihvaćaju od strane korisnika, čime se znatno smanjuje složenost izrade, testiranja i održavanja aplikacije. S obzirom da je ovakav uzorak tradicionalno korišten u *desktop* aplikacijama, vrlo brzo je postao popularan s pojavom aplikacija zasnovanim na web-u. Prema tome, danas gotovo svi programski jezici podržavaju ovu arhitekturu. Više o MVC-u u [10].

Pomoću ovog uzorka, korisnički se zahtjevi preusmjeravaju na kontroler koji je odgovoran za rad s modelima ili prikazima. Kontroler, odnosno upravljač pretvara te zahtjeve u naredbe kojima definira ponašanje aplikacije te na taj način djeluje kao „sučelje“ između komponente modela i komponente pogleda. Model je središnja komponenta ovog uzorka koja reagira na svu logiku baziranu na podacima s kojima korisnik vrši neku radnju. Model dakle, izravno upravlja logikom i pravilima aplikacije te se također koristi za komuniciranje sa bazom podataka. Pogled može biti bilo koji prikaz prethodno definiranih informacija poput grafikona ili dijagrama.

Kao što je vidljivo na slici 3.1., MVC arhitektura osim što dijeli aplikaciju na tri komponente, utvrđuje njihovu međusobnu interakciju:

- **Model** indicira sebi pridruženim pogledima kada je došlo do neke promjene. Ovakve indikacije omogućuju pogledu da prikaže ažurirano stanje modela.
- **Pogled** daje prikaz informacija korisniku koje dobiva od modela.
- **Upravljač** reagira na korisničke zahtjeve tako da ih interpretira kao naredbe kojima manipulira odgovarajućim modelima.



Sl. 3.1. *Prikaz interakcije komponenata MVC arhitekture*

3.4. Korištene tehnologije i razvojni alati

U ovom dijelu rada ukratko su opisane tehnologije i alati pomoću kojih je realizirano programsko rješenje za internet trgovinu, a njihova implementacija će biti prikazana u praktičnom dijelu rada. Integrirano razvojno okruženje (eng. *Integrated Development Environment*) korišteno za izradu ovog rada je Visual Studio koji se, osim za izradu web aplikacija, koristi i za izradu desktop i mobilnih aplikacija baziranih na platformama poput Windows formi, WPF (eng. *Windows Presentation Foundation*) i .NET platforme. Neke od glavnih komponenti koje Visual Studio sadrži su uređivač programskog kôda (eng. c*ode editor*) koji podržava IntelliSense, odnosno komponenta koja predlaže ostatak kôda i refaktoriranje istog, te komponenta za ispravljanje grešaka (eng. *debugger*).

3.4.1. Tehnologije za izradu korisničkog sučelja

Sučelja programskog rješenja izrađenog u okviru ovog diplomskog rada su izrađena pomoću standardnih tehnologija za izradu korisničkog sučelja (eng. *frontend*). Za oblikovanje sadržaja stranice korišten je opisni jezik HTML (eng. *HyperText Markup Language*), zatim stilski opisni jezik CSS (eng. *Cascading Style Sheets*) za uređivanje izgleda i rasporeda elemenata stranice te skriptni programski jezik JavaScript koji omogućuje interaktivnost web stranice. Dizajn programskog rješenja izrađenog u okviru ovog rada je baziran na postojećem predlošku otvorenog programskog kôda Bootstrap, slobodno dostupnim na [11], odnosno službenoj stranici besplatnih predložaka. Isti je modificiran prema vlastitim zahtjevima na aplikaciju.

Prema [12], Bootstrap je najčešće korišteni *frontend* programski okvir koji koristi ranije spomenute tehnologije kako bi znatno olakšao dizajniranje prilagodljivih stranica. Sastoji se od gotovih predložaka za forme, gumbove, navigaciju i ostalih komponenti sučelja. Prilagodljivi, odnosno responzivni dizajn podrazumijeva da se raspored komponenti dinamički prilagođava s obzirom na rezoluciju ekrana uređaja na kojem se pregledava (računalo, pametni telefon ili tablet). Pored toga, Bootstrap je kompatibilan sa svim verzijama poznatijih internet preglednika kao što su Google Chrome, Mozilla Firefox, Opera itd.

Osim toga, za upotpunjavanje sučelja korišten je FontAwesome skup alata (eng. *toolkit*) za font i ikone čija se dokumentacija nalazi u [13]. Zasnovan je na CSS i LESS stilskim jezicima, a glavna mu je svrha da korisnicima vizualno olakša snalaženje na sučelju, odnosno sučelje čini više razumljivim (eng. *user-friendly*).

3.4.2. ASP.NET MVC

ASP.NET MVC je programski okvir otvorenog kôda za izradu robusnih, dinamičkih web aplikacija baziranih na MVC arhitekturi. Budući da su komponente kao i njihove međusobne interakcije ove arhitekture već ranije opisane, ovdje će biti opisane u kontekstu ASP.NET MVC tehnologije. Prema tome, aplikacija se dijeli na tri glavne komponente:

- Model može biti klasa ili skup klasa napisanih u C# programskom jeziku koje definiraju i opisuju podatke poslovne logike. Model je zadužen za sve podatke koji se trebaju prikazati iako on nema nikakvu informaciju o tome gdje i kako će se ti podaci prikazivati,

- Pogled je HTML dokument (*.cshtml* datotečnog nastavka) za prikaz korisničkog sučelja aplikacije koji uz pomoć Razor sintakse omogućuje dinamičan prikaz sadržaja. Razor je sintaksa koja omogućuje kombiniranje opisnog kôda sa kôdom poslužiteljske strane (eng. *server-side*) temeljenog na C# programskom jeziku. Detaljniji opis Razor sintakse slijedi u nastavku ovog rada,
- Kontroler je specijalna C# klasa koja omogućuje komunikaciju sa korisnikom i fleksibilnost cijele logike aplikacije. On utvrđuje međusobnu interakciju modela i pogleda. Najprije komunicira sa modelom i onda odlučuje koji prikaz će se prikazati krajnjem korisniku.

Prioritet izvršavanja Razor sintakse ima naravno poslužiteljski kôd nakon čijeg se izvršavanja šalje pogled pregledniku. Prisutnost poslužiteljskog kôda na klijentskoj strani omogućuje izvršavanje puno kompleksnijih zadataka nego što je to moguće samo sa tehnologijama klijentske strane. Prema tome, omogućuje pristup bazi podataka kao i dinamično generiranje sadržaja, a da se pritom iz perspektive web preglednika ne razlikuje od klasičnog načina generiranja, kao što je opisano u [14].

3.4.3. Entity Framework

Prema [15], Entity Framework je objektno-relacijski (eng. *Object-relational mapping - ORM*) programski okvir otvorenog kôda koji se koristi za razvoj aplikacija koje su pretežito orijentirane podacima. Glavna svrha ovog programskog okvira je povećanje produktivnosti razvojnih programera eliminacijom potrebe pisanja tradicionalnih SQL upita nad bazom podataka. To ostvaruje na način da se podacima pristupa uz pomoć ugrađenih metoda nad objektima. Ti objekti su zapravo tablice, pogledi i procedure relacijske baze podataka koji su preslikani, tj. mapirani pomoću ORM tehnike. Entity Framework nudi tri pristupa razvoja:

- **Prvo-baza** (eng. *Database-first*)
- **Prvo-model** (eng. *Model-first*)
- **Prvo-kôd** (eng. *Code-first*)

Programsko rješenje izrađeno u okviru ovog rada je realizirano pomoću prvo-kôd pristupa koji je, za razliku od klasične izrade baze podataka, pretežito namijenjen za bazu koja još uvijek ne postoji, već ju on sam izrađuje. Također može poslužiti i za izradu odgovarajućih tablica unutar postojeće baze na temelju prethodno izrađenih klasa, tj. modela. Prilikom svake izmjene modela, ovaj pristup omogućuje automatsko ažuriranje tablica na temelju tih promjena.

4. IMPLEMENTACIJA PROGRAMSKOG RJEŠENJA ZA PRODAJU RAČUNALA I RAČUNALNE OPREME

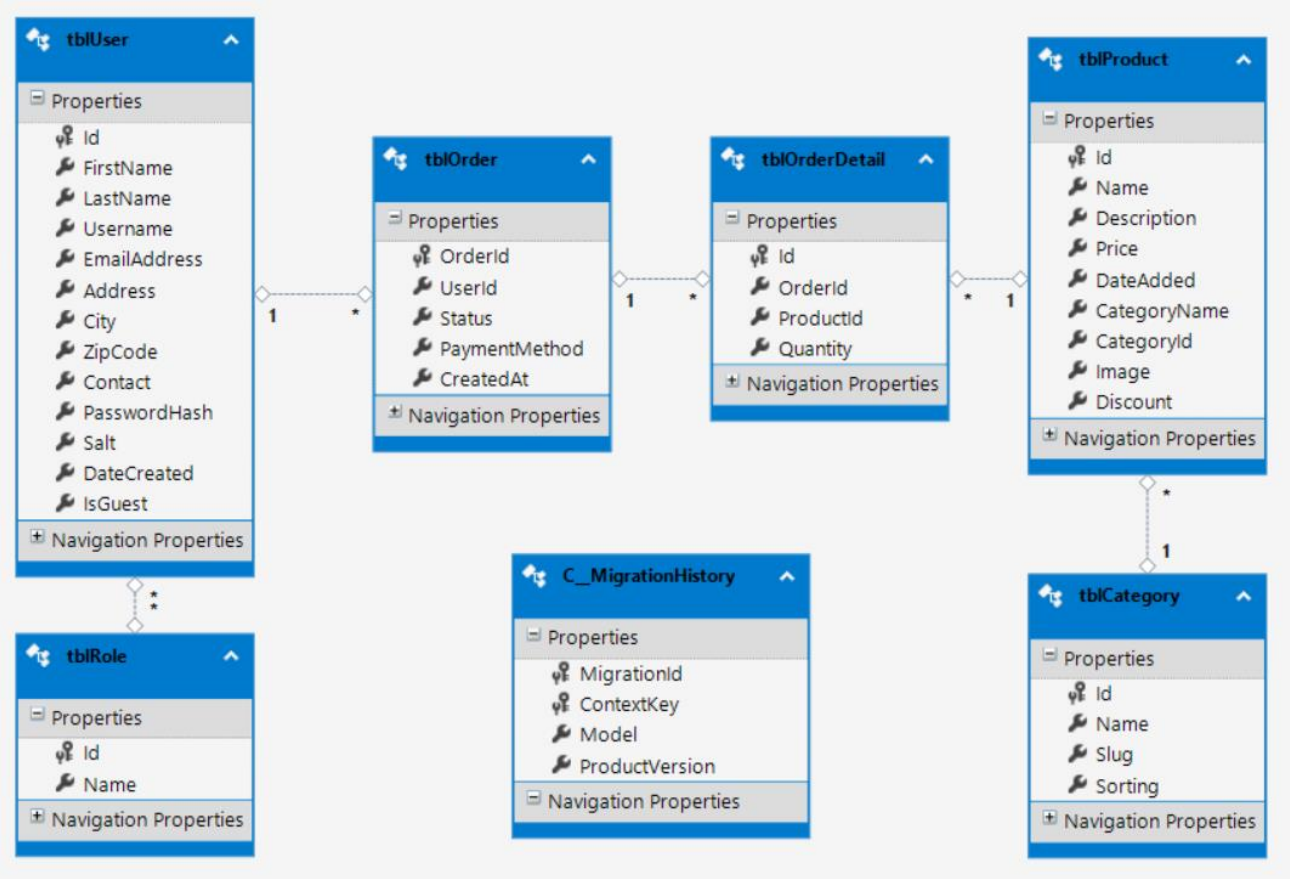
U ovom dijelu rada će se opisati kako je izrađena baza podataka koja je prilagođena zahtjevima na programsko rješenje izrađeno u okviru ovog rada. Nakon toga će biti prikazan tok izrade aplikacije, počevši prvo od korisničke strane. Zatim slijedi opis izvedbe administratorske strane te će na kraju ovog poglavlja biti prikazana implementacija sigurnosnog mehanizma aplikacije.

4.1. Baza podataka programskog rješenja

U ovom radu korišten je SQL Server Database sustav za upravljanje bazama podataka koji je integrirani dio Visual Studio okruženja. Kao što je ranije spomenuto, za izradu odgovarajuće lokalne relacijske baze podataka koristiti će se prvo-kôd pristup razvoja. Iako samo korisnik računala na kojem je lokalna baza ima pristup istoj, SQL Server Database uveliko pomaže developeru uvidjeti greške u kôdu kako bi ih mogao ispraviti prije nego se baza podataka napravi na pravom serveru.

Svaki model aplikacije koji se upotrebljava za spremanje podataka prema prvo-kôd pristupu, uz pomoć migracija izrađuje odgovarajuću tablicu unutar baze podataka. Migracija je obična *.cs* datoteka generirana od strane Entity Frameworka u kojoj su definirana sva pravila modela za izradu pojedinih tablica. Tako se tim pravilima mogu odrediti kojeg će tipa biti pojedini atributi tablice kao i definirati primarni ili strani ključ iste. Još jedna karakteristika korištenja migracija je da omogućuje popunjavanje određenih tablica baze podataka inicijalnim podacima, za što je zaslužna *Seed* metoda unutar koje se definiraju ti podaci. To je korisno primjerice kada se prilikom izmjene modela svi podaci te tablice obrišu, tada se ne mora ručno popunjavati kako bi se demonstrirao rad aplikacije. *Seed* metoda se može pozvati u jednom od tri različita slučaja: prilikom promjene modela, prilikom nepostojanja baze podataka ili prilikom svakog pokretanja aplikacije.

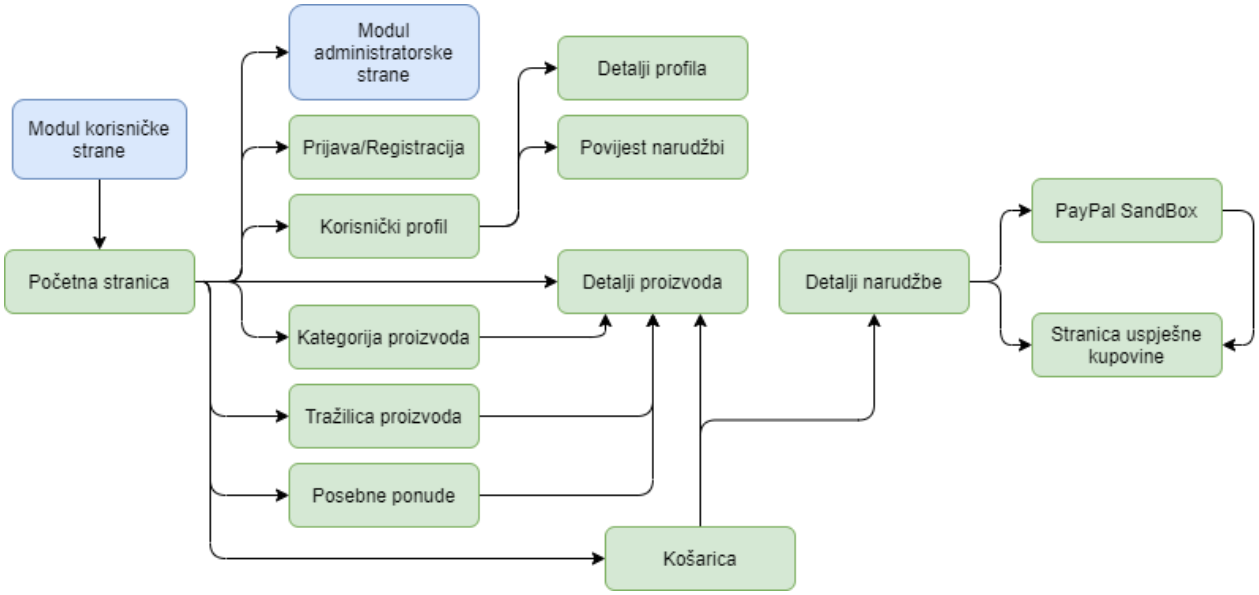
Kao rezultat migriranja, na slici 4.1. se može vidjeti prikaz dijagrama izrađene baze podataka koji predstavlja sve tablice kao i njihove međusobne veze. Tablica *MigrationHistory* je kako joj samo ime kaže, tablica povijesti migracija. Nju izrađuje sam Entity Framework automatski tijekom migriranja, a glavna joj je svrha pružanje uvida u promjene modela baze podataka, odnosno njezinih tablica.



Sl. 4.1. *Dijagram baze podataka za programsko rješenje izrađeno u okviru ovog rada*

4.2. Korisnička strana programskog rješenja

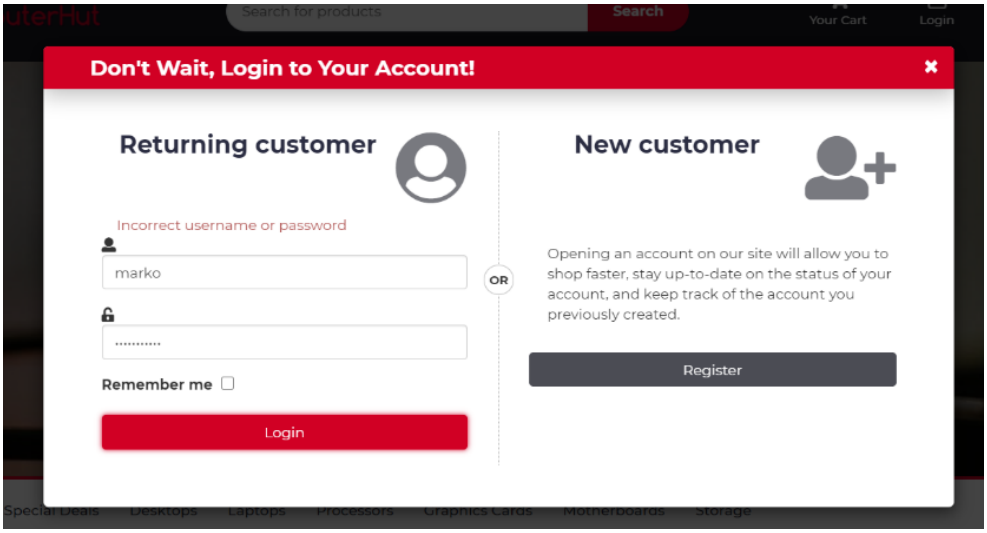
Korisnička strana je strana aplikacije koja je namijenjena krajnjim korisnicima, odnosno kupcima. U njih se ubrajaju registrirani ali i neregistrirani korisnici, tj. gosti koji također imaju privilegiju pretraživanja i kupovanja proizvoda. U nastavku će biti opisana implementacija registracije i prijave korisnika kao i svih bitnih značajki internet trgovine prateći zahtjeve na programsko rješenje izrađeno u okviru ovog rada. Na slici 4.2. je prikazana struktura korisničke strane čija je polazišna točka početna stranica internet trgovine kojoj je moguće pristupiti bez ikakve prijave ili registracije. Pogled početne stranice kao i svi ostali pogledi koriste zajednički pogled (eng. *layout*) koji se sastoji od trake zaglavlja, podnožja i navigacijske trake za snalaženje po kategorijama proizvodima. Tako se npr. početnoj stranici ili tražilici proizvoda može pristupiti iz bilo kojeg pogleda.



Sl. 4.2. *Struktura korisničke strane programskog rješenja izrađenog u okviru ovog rada*

4.2.1. Registracija i prijava

Za implementaciju logike registracije i prijave korisnika napravljen je *AccountController* upravljač koji sadrži sve potrebne metode poput metoda za provjeru podudaranosti te šifriranje lozinke. Korišteni mehanizam zaštite podataka biti će opisan u poglavlju 4.4. Kako bi se registrirani korisnici razlikovali od administratora i gosta, prilikom izrade novog računa im se dodjeljuje odgovarajuća uloga, odnosno ID uloge koji se čita iz tablice *Roles*. Uloge su prethodno definirane unutar *Seed* metode čija je svrha ranije opisana.



Sl. 4.3. *Iskočni prozor obrasca za prijavu postojećih korisnika*

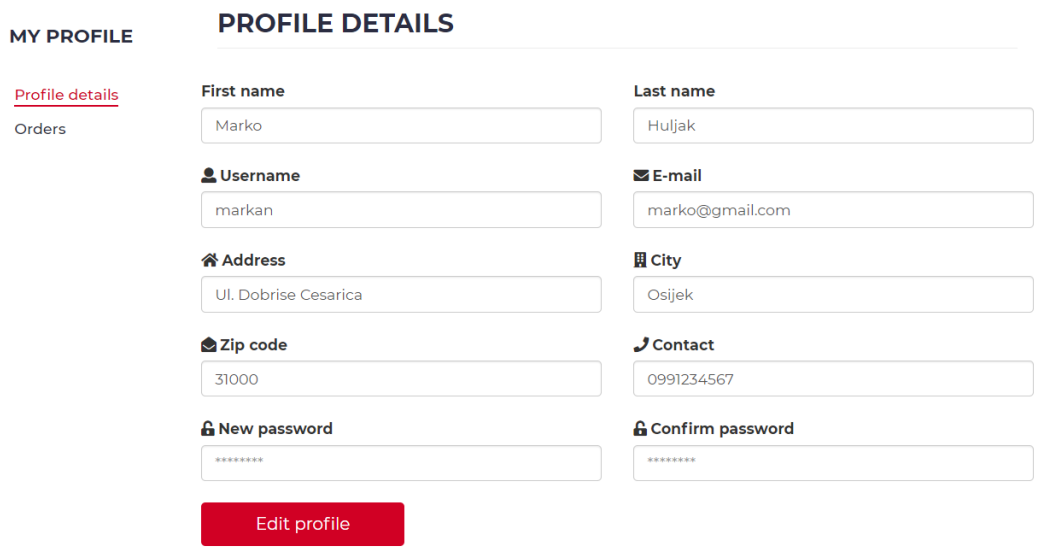
Budući da je obrazac za prijavu korisnika implementiran kao iskočni (eng. *popup*) prozor (slika 4.3.) koji je ujedno zajednički element svih pogleda (može mu se pristupiti iz bilo kojeg pogleda), provjera unesenih podataka je napravljena bez osvježavanja stranice uz pomoć asinkronog upita.

Validacija lozinke unutar *UserVM* modela koji se koristi prilikom izrade korisničkog računa se vrši pomoću ugrađenih pravila atributa modela koje pruža ASP.NET programski okvir. Osim toga, omogućuje i jednostavno umetanje željenih poruka upozorenja.

4.2.2. Korisnički profil

Sljedeća važna značajka internet trgovine koju je nužno omogućiti registriranim korisnicima je uvid kao i ažuriranje vlastitih informacija koje su unijeli tijekom registracije. Ovdje je poslovna logika jako slična logici za registraciju, a razlikuje se po tome što se najprije moraju dohvatiti podaci trenutno prijavljenog korisnika kako bi se isti zatim proslijedili *UserProfile* pogledu i pridružili odgovarajućim poljima obrasca. To je lako ostvarivo uz pomoć ugrađenog ASP.NET mehanizma koji podatke prijavljenog korisnika dohvaća na temelju spremljenog autentikacijskog kolačića (eng. *cookie*). U ovom radu je objekt prijavljenog korisnika dohvaćen na temelju imena korisnika, a moguće je i preko e-mail adrese pošto su oba podatka jedinstveni unutar baze podataka.

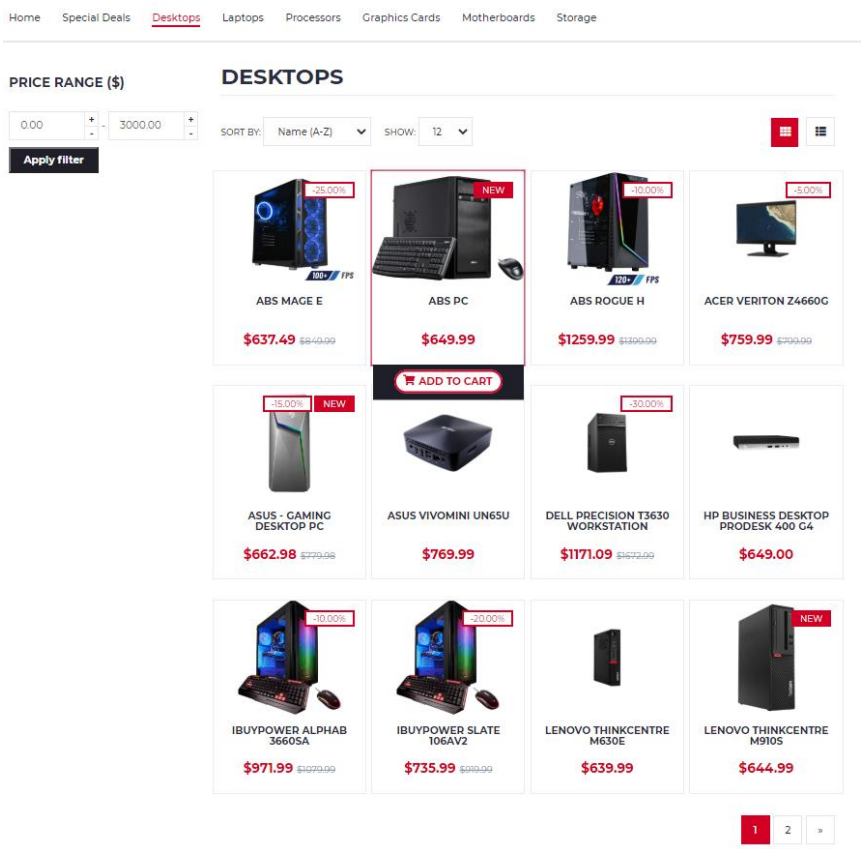
Kao što je vidljivo na slici 4.4., osim detalja profila, korisniku se nudi i kartica (eng. *tab*) uvida povijesti naručenih i kupljenih proizvoda, čija će se implementacija pojasniti na kraju ovog potpoglavlja.



Sl. 4.4. *Sučelje korisničkog profila*

4.2.3. Kategorije

Pozadinska strana sučelja kategorije (slika 4.5.) se temelji na *Grid* funkciji koja je univerzalna za prikazivanje proizvoda po kategorijama, traženom nazivu proizvoda ili proizvoda sa sniženim cijenama. Sučelje sadrži osnovne filtere kao što su odabir stranice prikaza, broja prikazanih proizvoda, sortiranje po imenu ili cijeni i filter raspona cijene. Također nudi i dva različita načina prikaza proizvoda. Isti mogu biti raspoređeni kao rešetka (eng. *grid*) elemenata (kao što se vidi na spomenutoj slici) ili mogu biti poredani kao lista proizvoda. Svakim pozivom, ranije spomenuta funkcija zahtijeva parametre svih filtera koji su poslani putem URL-a. Prvim pozivom postavljaju se zadani parametri gore navedenih filtera kako bi dobili početni prikaz stranice. Kako bi se znalo koji točno pogled treba vratiti, osim parametara filtera, funkciji su dodana još dva ulazna parametra. Parametar *cat* se odnosi na ime kategorije koja se prikazuje, a drugi parametar je vezan za tražilicu proizvoda koja će se ukratko objasniti u nastavku ovog poglavlja. Tako se, dakle na temelju ta dva parametra utvrđuje hoće li se vratiti pogled za traženu kategoriju, pogled rezultata tražilice ili pogled za posebne ponude (eng. *special deals*), odnosno proizvode sa sniženom cijenom.



Sl. 4.5. *Sučelje kategorije proizvoda*

Kao što je vidljivo u programskom kôdu na slici 4.6. se provjerava parametar *cat* kako bi se utvrdilo radi li se o kategoriji. Nakon toga se postavlja ime pogleda kojem će se na kraju ove funkcije proslijediti lista modela dohvaćenih proizvoda za traženu kategoriju. Osim liste modela, pogledu je važno proslijediti i privremene varijable poput imena kategorije koje se sprema u *ViewBag* dinamičku varijablu ASP.NET MVC programskog okvira. Na taj se način također prosljeđuju i parametri filtera kako bi pri ponovnom pozivu ove funkcije isti bili zapamćeni.

```
else if (cat != "") // Category
{
    // Set page type
    viewName = "Category";
    // Get category id
    CategoryModel categoryDTO = db.Categories.Where(x => x.Slug == cat).FirstOrDefault();
    int catId = categoryDTO.Id;
    // Store category name
    ViewBag.CategoryName = categoryDTO.Name;
    // Init the list
    productVMList = db.Products.ToArray().Where(x => x.CategoryId == catId).Select(x => new
ProductVM(x));
    // Check if there are no products
    if (productVMList == null || productVMList.ToList().Count == 0)
    {
        ViewBag.Message = "There are no products in " + categoryDTO.Name + " category.";
        viewType.Add(viewName, null);
        return View(viewType.FirstOrDefault().Key);
    }
}
```

Slika 4.6. *Postavljanje pogleda za prikaz proizvoda na temelju odabrane kategorije*

4.2.4. Tražilica proizvoda

Budući da je *Grid* funkcija implementirana kao zajednička funkcija svih prikaza proizvoda, implementacija tražilice je ništa više nego dodavanje još jednog pogleda, uvjeta i parametra funkcije i naravno forme preko koje se tražilica poziva. Kao što se već ranije spomenulo, drugi parametar za utvrđivanje pogleda koji se treba prikazati je vezan za tražilicu, točnije, to je parametar filtera tražilice. Prema tome, programski kôd prikazan na slici 4.7. je sličan kôdu za dohvaćanje proizvoda za kategoriju. Ovdje se naravno dodaje ime pogleda koji je namijenjen za prikaz rezultata pretrage. Također se prosljeđuje privremena *ViewBag* varijabla, ali u ovom slučaju se koristi *searchString*, tj. traženo polje znakova kojim se filtrira lista svih postojećih proizvoda u bazi podataka.

```
else if (searchString != "") // Search
{
    // Set page type
    viewName = "Search";
    // Save search result
    ViewBag.SearchString = searchString;
    // Init the list
    productVMList = db.Products.ToArray().Where(x => x.Name.IndexOf(searchString,
StringComparison.OrdinalIgnoreCase) >= 0).Select(x => new ProductVM(x));
    // Check if there are no products
    if (productVMList == null || productVMList.ToList().Count == 0)
    {
        ViewBag.Message = "Your search for '" + searchString + "' did not match any
products.";
        viewType.Add(viewName, null);
        return View(viewType.FirstOrDefault().Key);
    }
}
```

Slika 4.7. *Postavljanje pogleda za prikaz proizvoda na temelju traženog naziva*

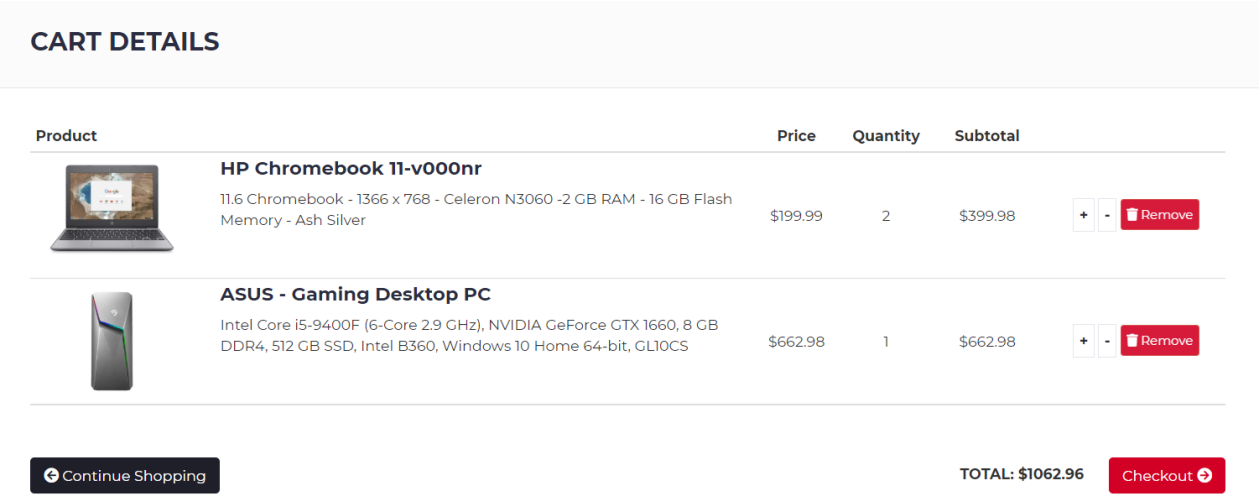
4.2.5. Detalji proizvoda

Prema zahtjevima na aplikaciju, bilo je potrebno omogućiti prikaz detalja proizvoda. Za to je implementiran poseban pogled na kojem se osim opisa, cijene i naziva proizvoda prikazuju i slike istog. Galerija slika je implementirana pomoću postojećeg JavaScript alata imena FancyBox čija se dokumentacija nalazi u [16]. Korištenje ovog alata je vrlo jednostavno. Za osnovni prikaz galerije, sve što je bilo potrebno je svakom elementu slike dodati klasu *fancybox* te u JavaScript dijelu kôda pozvati istoimenu funkciju koja se odnosi na sve elemente sa tom klasom. Osim toga, bilo je potrebno omogućiti dodavanje proizvoda u košaricu kao i odabir količine željenog proizvoda.

4.2.6. Košarica

Značajka dodavanja proizvoda u košaricu je implementirana pomoću AJAX poziva metode kontrolera košarice kojoj se šalje ID proizvoda. *AddToCartPartial* metoda zatim pretražuje postoji li već taj proizvod u košarici kako bi znala treba li ga dodati sa svim njegovim detaljima (ukoliko ne postoji) ili samo inkrementirati količinu istog. Premda se sadržaj košarice sprema u varijablu sesije koja se pamti po cijeloj stranici internet trgovine, korisnicima je uz klasičan prikaz košarice sa svim detaljima također omogućen i uvid u osnovne detalje košarice. Košarica kao iskočni prozor implementirana je kao parcijalni pogled, odnosno element zajedničkog pogleda. Kao takva, vidljiva je, ne samo na onim pogledima na kojima se nalaze proizvodi, nego i na svim ostalim pogledima korisničke strane internet trgovine. Što se tiče funkcionalnosti ovog iskočnog prozora, korisnik može direktno uklanjati proizvode iz košarice što se također vrši asinkronim pozivom odgovarajuće metode istoga kontrolera. Osim toga može i pristupiti potpunom prikazu košarice ili ako ga ne zanimaju svi detalji, može direktno pristupiti pogledu na kojem se završava kupovina.

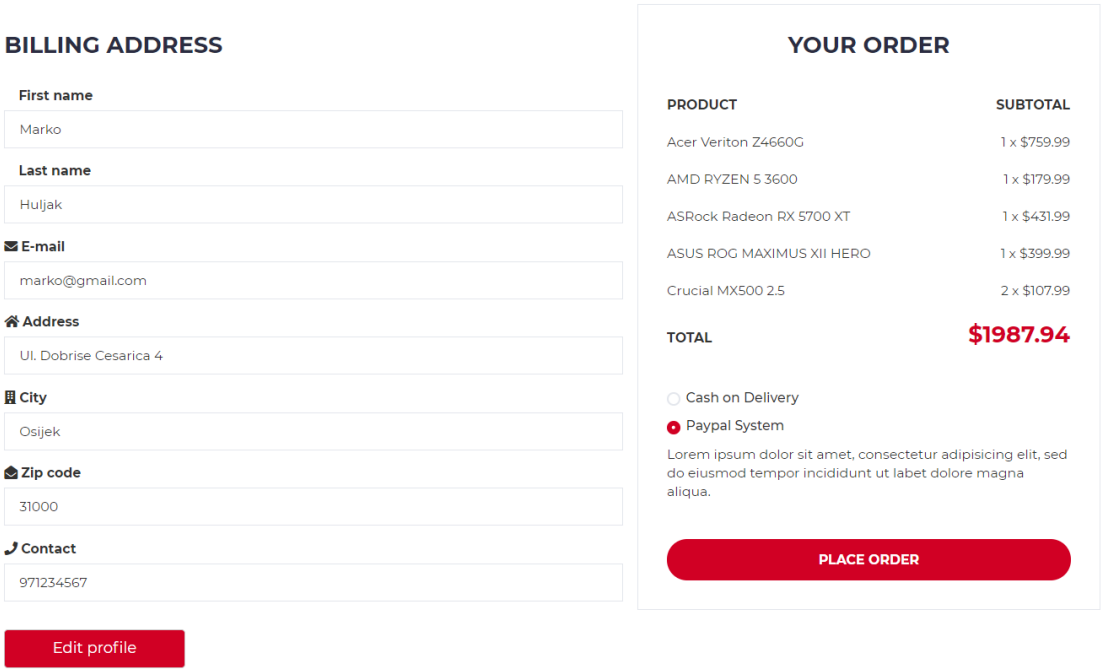
Na slici 4.8. je prikazan cjelokupni prikaz košarice u kojem korisnici imaju potpunu kontrolu nad košaricom, odnosno mogu mijenjati količinu pojedinih proizvoda ili ih potpuno ukloniti iz košarice. Implementacija ovih funkcionalnosti je također izvršena slanjem ID-a proizvoda prema metodi koja kao rezultat vraća JSON objekt kojim se bez osvježavanja stranice ažuriraju HTML elementi količine proizvoda kao i ukupnog iznosa košarice. Ukoliko se korisnici odluče završiti kupovinu, klikom na *Checkout* gumb će biti preusmjereni na krajnju točku korisničke strane koja će se detaljnije opisati u nastavku.



Sl. 4.8. *Sučelje košarice*

4.2.7. Završetak kupovine

Kao što je bilo rečeno, kupovinu mogu obavljati registrirani i neregistrirani korisnici. U primjeru dolje (slika 4.9.) kupovinu obavlja registrirani kupac, stoga je obrazac unaprijed popunjen njegovim podacima. Međutim, u slučaju da gost obavlja kupovinu, obrazac je inicijalno prazan te ga tada popunjava shodno validacijama polja. Kao i u poslovnoj logici registracije, u ovom slučaju se određenoj metodi kontrolera također prosljeđuje objekt korisnika. *Checkout* metoda zatim provjerava da li je trenutni korisnik registriran ili ne. Ukoliko se radi o gostu, istog je također potrebno spremiti u *Users* tablicu podataka kako bi administrator imao uvid u njegove podatke potrebne za pošiljku proizvoda. Kako bi administratoru, odnosno voditelju ove internet trgovine bilo lakše upravljati narudžbama neregistriranih korisnika, gostu se kao korisničko ime dodaje „*Guest_*“ s nastavkom nasumično generiranog jedinstvenog broja (eng. *Globally Unique Identifier – GUID*).



Sl. 4.9. *Pregled narudžbe*

Objektu narudžbe se uz identifikacijski broj kupca dodjeljuju način plaćanja i početni status narudžbe koji ovisi o načinu plaćanja. Ukoliko se radi o plaćanju pouzećem, status postaje *Pending* što označava da je narudžba u tijeku, dok u slučaju plaćanja testnim servisom PayPala status postaje *Paid* što označava da je narudžba plaćena. Svaki objekt narudžbe ima sebi pridružen objekt detalja narudžbe koji sadrži ID proizvoda i njegovu količinu. Tako će prema primjeru na spomenutoj slici objekt narudžbe imati pridruženo šest objekata detalja narudžbe. Na kraju ovog procesa je nužno isprazniti košaricu što se vrši postavljanjem sesije košarice na *null* vrijednost.

4.2.8. Povijest narudžbi

Kako bi se upotpunilo osnovno korisničko iskustvo prema internet trgovini, korisniku je bitno omogućiti prikaz povijesti svih narudžbi koje je obavio. Za to je bilo potrebno u *Account* kontroler dodati metodu koja dohvaća narudžbe na temelju identifikacijskog broja trenutno prijavljenog korisnika te ih prosljeđuje prema *Orders* pogledu (slika 4.10.). Ovdje korisnik osim uvida u detalje narudžbe može pratiti status istih koji se ažurira prema promjenama koje može vršiti samo administrator ove internet trgovine. Budući da se tom pogledu može pristupiti samo preko korisničkog profila, može se zaključiti da je povijest narudžbi dostupna samo registriranim korisnicima.

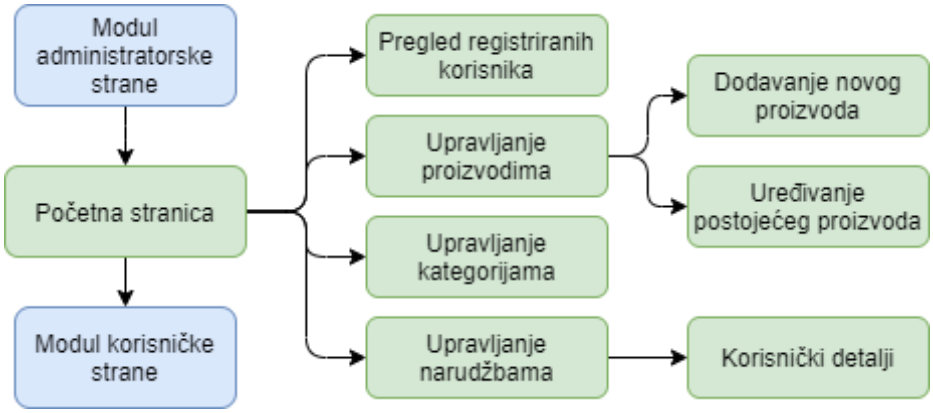
ORDERS

Order details	Date created	Total	Status	Payment Method
AMD RYZEN 5 3600 x 1 ASRock Phantom Gaming D Radeon RX 570 x 1 ASUS ROG MAXIMUS XII HERO x 1 Crucial MX500 2.5 x 2	6/13/2020 12:06:56 AM	$915.95	Paid	Paypal
GIGABYTE - AERO 15 OLED SA-7US5020SH x 1	6/13/2020 12:09:08 AM	$1014.30	Completed	Cash
WD Red 4TB x 1	6/13/2020 12:09:30 AM	$109.99	Pending	Cash

Sl. 4.10. *Povijest narudžbi*

4.3. Administratorska strana programskog rješenja

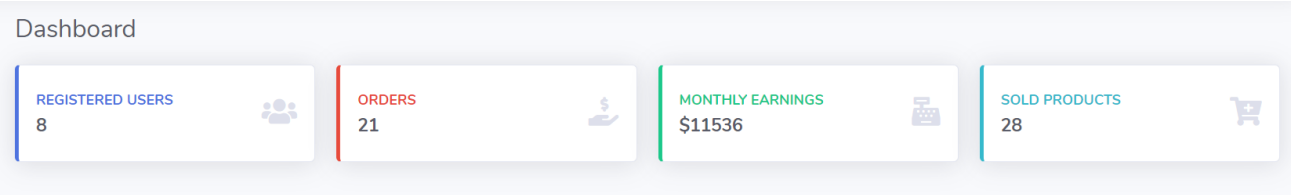
Kako bi struktura programskog kôda bila dobro organizirana i fleksibilna, administratorska strana je implementirana kao zasebni dio aplikacije, odnosno modul koji sadrži zasebne kontrolere, modele i poglede. Da bi administrator pristupio ovoj strani aplikacije najprije se mora prijaviti pomoću zadane lozinke i korisničkog imena definiranih unutar *Seed* metode. Prilikom uspješne prijave, na navigacijskoj traci korisničke strane se pojavljuje gumb *Admin area* koji vodi prema administratorskoj strani. Budući da je struktura ove strane, prikazana na slici 4.11., zapravo nastavak strukture korisničke strane, ista uključuje modul korisničke strane, ali samo kao povratnu vezu prema korisničkoj strani.



Sl. 4.11. *Struktura administratorske strane programskog rješenja izrađenog u okviru ovog rada*

Ovdje kao i na korisničkoj strani, svi pogledi koriste zajednički pogled koji uključuje navigacijske elemente. Na početnoj stranici se prikazuje kontrolna ploča (eng. *dashboard*) koja

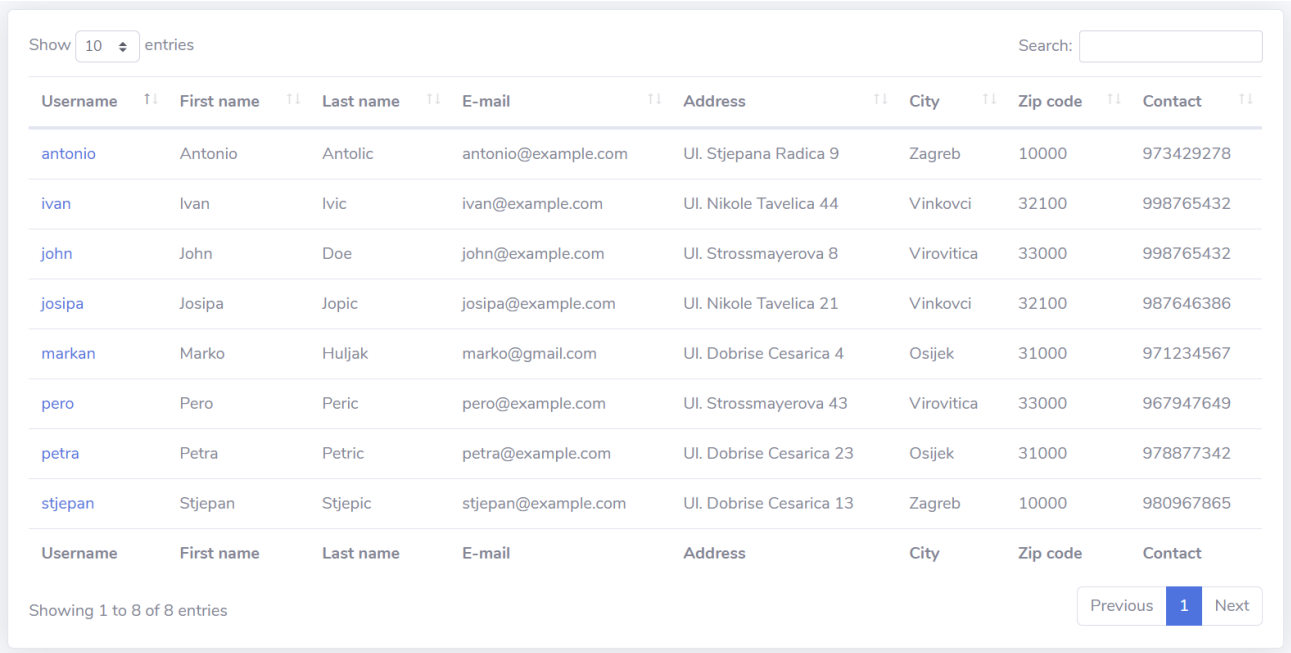
prikazuje kratke informacije o internet trgovini poput broja registriranih korisnika, broja narudžbi, zatim ukupno prodanih proizvoda te mjesečne zarade (slika 4.12.).



Sl. 4.12. *Kontrolna ploča administratorske strane*

4.3.1. Pregled registriranih korisnika

Prateći zahtjeve na programsko rješenje izrađeno u okviru ovog rada, voditelju ove internet trgovine je omogućen uvid u registrirane korisnike. Tablica korisnika prikazana na slici 4.13. je implementirana pomoću postojeće JavaScript biblioteke čija se dokumentacija korištenja nalazi u [17]. Bitno je napomenuti kako ova tablica prilikom interakcije s njezinim „*on the fly*“ filterima, odnosno dinamički prikazuje rezultat, što znači da najprije dohvati cijelu listu registriranih korisnika od koje prikazuje samo određene korisnike na temelju promjena tih filtera.

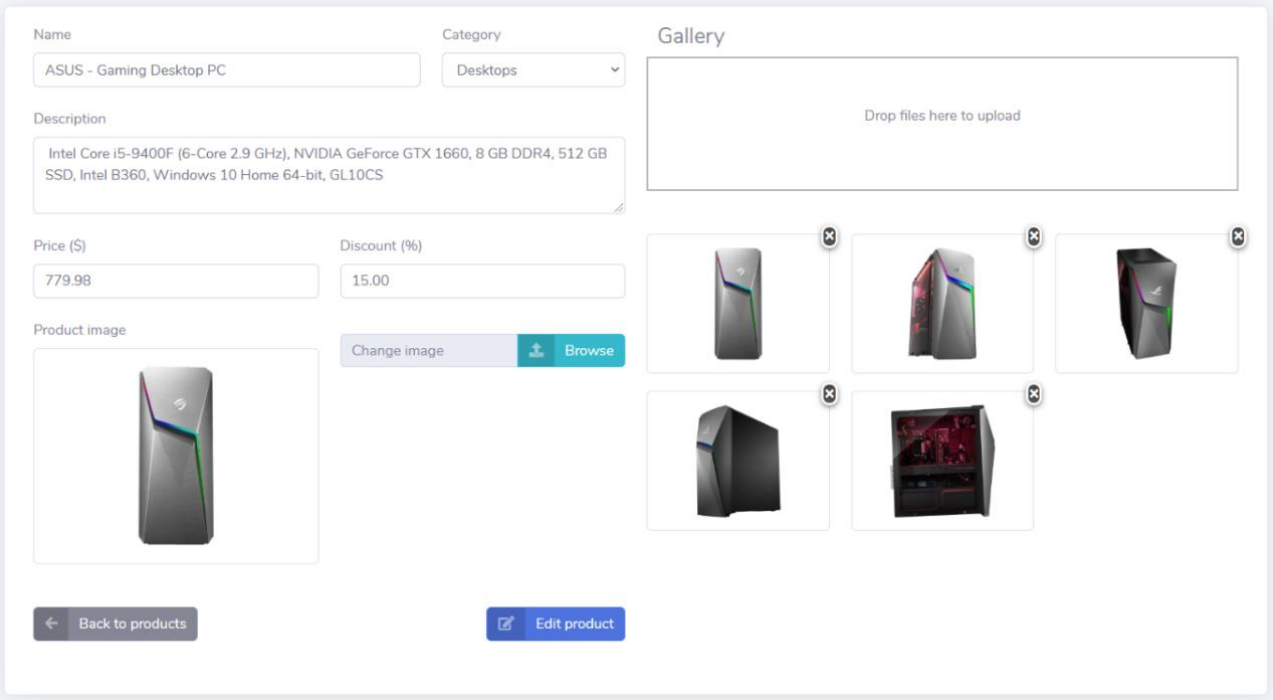


Sl. 4.13. *Pregled registriranih korisnika*

4.3.2. Upravljanje proizvodima

Jedna od najbitnijih funkcionalnosti administratorske strane je mogućnost upravljanja proizvodima. Tablica pregleda proizvoda je također implementirana pomoću ranije spomenute JavaScript biblioteke. Uz njezine ugrađene filtere, ovdje je dodan filter prikaza proizvoda po željenoj kategoriji. Osim toga, u tablicu je dodan stupac akcija u kojem se za svaki pojedini proizvod nalaze mogućnosti brisanja i uređivanja istog. Sučelje uređivanja je prikazano na slici 4.14. na kojoj je uz uređivanje detalja proizvoda vidljiva i mogućnost učitavanja slika.

Budući da slike zahtijevaju mnogo više memorije za pohranu nego uobičajeni podaci poput naziva i cijene proizvoda, spremanje slika u bazu podataka se pokazalo kao loša opcija. U tom slučaju bi se ista brzo prepunila ako bi se radilo o velikoj količini proizvoda, a samim time i slika što bi narušavalo performanse aplikacije. Prema tome, slike će se u programskom rješenju izrađenom u okviru ovog rada spremati lokalno, odnosno u *images* direktorij slika koji se nalazi u korijenskom direktoriju projekta. Na spomenutoj slici se vidi „*drag and drop*“ zona koja je namijenjena za višestruko učitavanje slika za galeriju. Zona je implementirana pomoću JavaScript biblioteke koja je dostupna u [18].

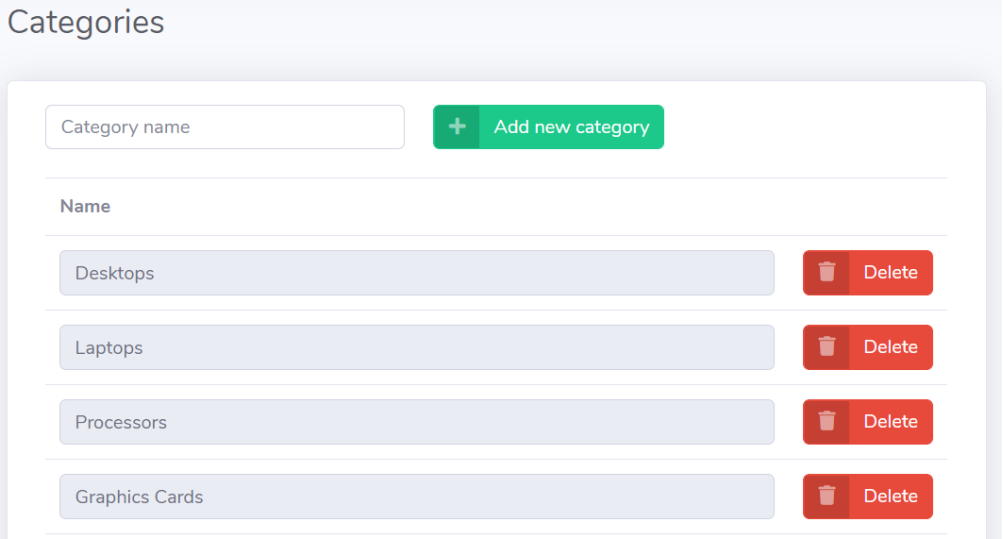


Sl. 4.14. *Sučelje za uređivanje proizvoda*

Analogno implementaciji uređivanja proizvoda je omogućena funkcionalnost za dodavanje novog proizvoda.

4.3.3. Upravljanje kategorijama

Kategorije pod kojima se proizvodi prikazuju na korisničkoj strani se dodaju pomoću sučelja prikazanog na slici 4.15. Komunikacija sa poslovnom logikom određenih metoda *Shop* kontrolera se vrši putem asinkronih poziva pomoću kojih administrator može mijenjati naziv i uklanjati kategorije. Također mu je omogućeno i mijenjanje redoslijeda kategorija prema kojim će iste biti prikazane na navigacijskoj traci korisničke strane. Upravljanje redoslijedom se vrši putem *„drag and drop“* pristupa koji je u ovom slučaju implementiran pomoću SortableJS biblioteke dostupne u [19].



Sl. 4.15. *Pregled kategorija*

4.3.4. Upravljanje narudžbama

Najvažnija značajka vođenja internet trgovine je upravljanje narudžbama čije je sučelje prikazano na slici 4.16. S obzirom na to da je za pregled narudžbi potrebno prikazati podatke poput korisničkog imena i detalja narudžbe koji ne pripadaju istom modelu, ASP.NET MVC omogućuje model pogleda (eng. *view model*) koji povezuje te modele, a služi isključivo za prikaz podataka. Kako bi administrator imao uvid u podatke gosta, prilikom klika na njegovo korisničko ime mu se prikazuje pogled korisničkih detalja. To također vrijedi i za registrirane korisnike.

Sl. 4.16. *Pregled narudžbi*

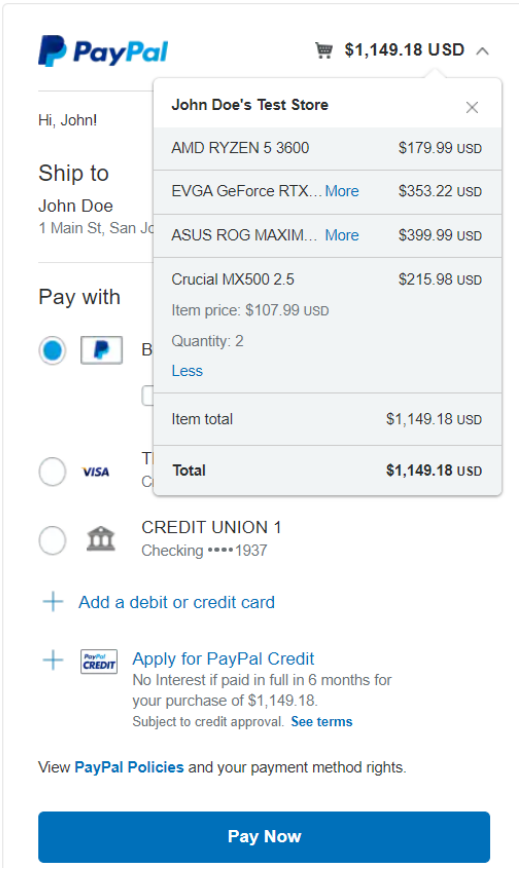
Upravljanje narudžbama se svodi na odabiranje statusa narudžbe iz padajućeg izbornika (eng. *dropdown*) čijom se promjenom asinkrono šalje ID i status narudžbe koji je u zapisu cijelih brojeva. Na temelju tih podataka se u *ChangeOrderStatus* metodi najprije dohvaća model narudžbe kojem se zatim ažurira status koji je definiran kao specijalni vrijednosni tip podataka, odnosno *enum*.

4.4. Demonstracija virtualnog plaćanja

Prije same implementacije PayPal Sandbox testnog okruženja za plaćanje bilo je potrebno stvoriti PayPal poslovni račun nakon čega se potrebno prijaviti na PayPal stranicu za developere. Ista pruža opcije postavljanja *sandbox* okruženja poput određivanja URL adrese kojoj se pristupa prilikom uspješno obavljene transakcije. Osim toga bilo je potrebno postaviti testne račune koji postoje samo u *sandbox* okruženju, a predstavljaju sudionike obavljanja testne transakcije. Tako se razlikuju osobni račun koji predstavlja kupca ili pošiljatelja u transakciji, i poslovni račun koji se odnosi na trgovca ili primatelja u transakciji.

Nadalje, u programskom rješenju izrađenom u okviru ovog rada je bilo potrebno dodati programski kôd forme kojom se svi podaci košarice prosljeđuju prema testnom okruženju. Dokumentacija korištenja forme se nalazi u [20], tj. službenoj stranici PayPala. Pojedini proizvod unutar košarice je predstavljen pomoću tri elementa unosa koji se odnose na ime, cijenu i količinu proizvoda. Važno je

napomenuti kako ova forma korisniku nije vidljiva jer služi isključivo za komuniciranje sa testnim okruženjem. Forma je implementirana kao parcijalni pogled koji je dio pogleda za pregled narudžbe, a okida se pomoću logike klijentske strane. Prilikom preusmjeravanja na testno okruženje, korisniku se prikazuje sučelje koje sadrži uvid svih proizvoda košarice kao i ukupan iznos potreban za obavljanje transakcije (slika 4.17.). Ovdje korisnik završava virtualnu transakciju nakon koje bude preusmjeren na pogled programskog rješenja izrađenog u okviru ovog rada kojim se korisnika informira o uspješno obavljenoj kupovini.

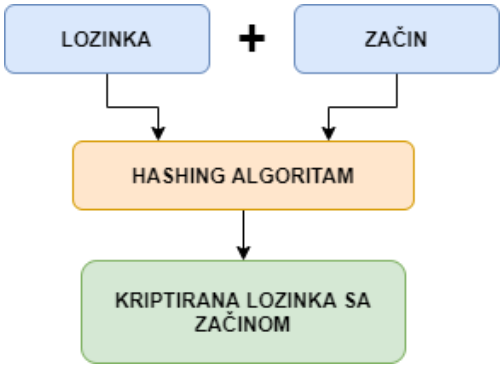


Sl. 4.17. *Plaćanje putem PayPal Sandbox testnog okruženja*

Kako bi se integriralo aktivno (eng. *live*) PayPal okruženje za plaćanje, najprije je potrebno unutar Visual Studia instalirati PayPal paket pomoću NuGet upravitelja paketima (eng. *package manager*). Zatim je potrebno podesiti *web.config*, odnosno konfiguracijski dokument projekta kako bi se mogla uspostaviti veza između aplikacije i PayPal servisa. Što se tiče logike za cjelokupni proces *online* plaćanja, u proizvoljnom kontroleru je potrebno dodati metode u kojima se vrše pozivi API (eng. *Application Programming Interface*) metoda PayPal servisa.

4.5. Implementacija sigurnosnog mehanizma

Internet trgovina za prodaju računala i računalne opreme mora jamčiti sigurnost korisničkih podataka. To se odnosi na sigurno spremanje lozinke u bazu podataka. Kako administrator baze podataka ne bi imao uvid u lozinke korisnika, potrebno ih je pohraniti u obliku beznačajnih znakova. Postoji niz različitih načina za kriptiranje lozinke, a u ovom programskom rješenju se koristi kriptiranje pomoću *hashinga*. *Hashing* je postupak koji omogućuje generiranje vrijednosti iz niza teksta pomoću određenog matematičkog algoritma. Rezultat takvog generiranja je kriptirana vrijednost koja podatke čini sigurnima. Mana ovog postupka je što za jedan niz znakova uvijek daje isti rezultat. Tako bi hakeri lako pomoću SQL ubrizgavanja (eng. *injection*) uspjeli dekriptirati lozinku. Ovaj problem se eliminira dodavanjem začina (eng. *salt*), odnosno nasumično generiranog niza znakova koji se dodaje na lozinku koja se *hashira* (slika 4.18.).



Sl. 4.18. *Kriptiranje lozinke sa začinom*

Od brojnih algoritama za *hashiranje*, prema [21], među najsigurnijima je BCrypt. Međutim, dizajniran je da bude spor kako bi hakerima otežao dekripciju. Cilj sigurnosnog mehanizma internet trgovine je postići ravnotežu između sigurnosti i vremena koje je potrebno za provjeru lozinke kako se ne bi narušilo korisničko iskustvo. SHA-512 (eng. *Secure Hash Algorithm*) algoritam se kao značajno brzi *hashing* algoritam pokazao dovoljno sigurnim, a pogotovo za svrhu programskog rješenja izrađenog u okviru ovog rada. Stoga se isti koristi pomoću istoimene ugrađene ASP.NET metode. Ovaj sigurnosni mehanizam je implementiran pomoću tri metode *Account* kontrolera. Najprije se poziva *CreateSalt* metoda koja generira začin koji bi se zajedno sa lozinkom predao u *CreateHash* metodu čiji rezultat, dobiven SHA-512 metodom, se sprema u bazu podataka kao hash lozinke. Osim toga, također je potrebno spremiti i začin, odnosno ključ koji će služiti za dekriptiranje lozinke prilikom autentikacije korisnika.

5. ZAKLJUČAK

Internet trgovina je promijenila način na koji ljudi kupuju i prodaju proizvode. Sve više trgovaca uočavaju prednosti web mjesta trgovine kako bi povećali prodaju proizvoda i prisvojili kupce i iz drugih zemalja. Prema tome, programska rješenja internet trgovine su danas znatno tražena, bilo da se radi o globalnoj ili lokalnoj razini. Navedeno je i glavni razlog zašto se kao zadatak ovog diplomskog rada postavlja izrada takvog programskog rješenja.

Programska rješenja internet trgovine danas se, osim kao web stranice, nameću i kao platforme za izradu i održavanje tih web stranica. S obzirom na postojanje različitih vrsta internet trgovina, u ovom radu su predstavljene najvažnije podjele istih. Internet trgovine se, osim prema tipu robe koje prodaju, razlikuju prema poslovnim modelima i platformama, kao i tipu softvera na kojem su zasnovane. Osim toga, za svaki od tipova internet trgovina navedena su neka od postojećih rješenja i predstavile njihove glavne značajke koje su služile kao uzor za postavljanje zahtjeva na ovaj rad.

Glavni zahtjevi na programsko rješenje izrađeno u okviru ovog diplomskog rada obuhvaćaju pružanje osnovnih značajki internet trgovine kao web stranice i samo neke od značajki platforme za izradu i održavanje takve web stranice. Takvo programsko rješenje bi trebalo razlikovati dva tipa korisnika – prodavača i registriranog kupca. Kupca bi se prilikom završetka kupovine trebalo preusmjeriti na testno okruženje platnog servisa poput PayPala. Prodavaču, odnosno administratoru se mora omogućiti upravljanje proizvodima i kategorijama prema kojima su isti grupirani.

Kao odgovor na postavljene zahtjeve, u sklopu ovog rada je izrađena internet trgovina koja može zadovoljiti potrebe neke lokalne trgovine koja se bavi prodajom računala i računalne opreme. Osim registriranog kupca i prodavača, ovo programsko rješenje razlikuje i gosta, odnosno neregistriranog kupca koji obavlja kupovinu bez registracije. Kupcima se daje na izbor plaćanje prilikom pouzeća ili putem platnog servisa.

Programska izvedba programskog rješenja izrađenog u okviru ovog rada je većinom ostvarena uz pomoć ASP.NET MVC i Entity Framework tehnologija, a podijeljena je u dva modula – korisnički i administratorski modul. Korisničkom modulu se može pristupiti bez ikakve registracije i prijave dok se administratorskom pristupa isključivo putem prijave pomoću prethodno definirane lozinke i imena. Korisničke lozinke se kriptiraju pomoću *hashing* algoritma i kao takve spremaju u bazu

podataka kako administrator ne bi imao uvid u iste. Baza podataka je izrađena pomoću *code-first* pristupa te je ista popunjena inicijalnim podacima uz pomoć *Seed* metode Entity Frameworka.

Registriranim korisnicima je omogućen korisnički profil koji nudi upravljanje osobnim podacima kao i uvid u povijest kupljenih proizvoda. Svim korisnicima se prikazuju proizvodi koje mogu filtrirati po rasponu cijene, imenu i kategoriji. Košarica je implementirana sa značajkama upravljanja količinama proizvoda koji su unutar nje. U svrhu demonstracije procesiranja plaćanja putem platnog servisa, integrirano je PayPal Sandbox testno okruženje. Administratoru je osim značajki upravljanja proizvodima i kategorijama omogućena značajka upravljanja narudžbama.

Trenutno programsko rješenje izrađeno u okviru ovog rada ima sve funkcionalnosti koje pružaju potpuni doživljaj kupovine putem interneta. Moguće ga je koristiti kao optimalno rješenje za manje lokalne trgovine, uz to da se testni servis *online* plaćanja zamijeni sa aktivnim. Buduća nadogradnja navedenog programskog rješenja bi bila implementacija liste želja (eng. *wish list*) kao i sustava komentiranja i ocjenjivanja proizvoda.

LITERATURA

[1] N. Patel, 5 Most Important Factors for Success in E-commerce [online], Neil Patel, dostupno na: https://neilpatel.com/blog/factors-for-ecommerce-success/ [7.7.2020.]

[2] 3dcart tim, What is Ecommerce and how can it benefit your business?, Benefits of Ecommerce for Business [online], dostupno na: https://www.3dcart.com/ecommerce-saas.html [18.5.2020.]

[3] G. Gil, Types of eCommerce Business Models, Websites, and Platforms [online], 3dcart, dostupno na: https://blog.3dcart.com/types-of-ecommerce-business-models-websites-platforms [20.5.2020.]

[4] K. Young-won, 22.10.2019., Samsung increases OLED supply for Apple's latest iPhones [online], The Investor, dostupno na: http://www.theinvestor.co.kr/view.php?ud=20191022000805 [7.7.2020.]

[5] S. Bagchi, 30.11.2018., 5 Game Changing Technologies In E-Commerce [online], CXOtoday, dostupno na: https://www.cxotoday.com/news-analysis/5-game-changing-technologies-in-e-commerce/ [28.6.2020.]

[6] Amazon [online], dostupno na: https://www.amazon.com/ref=nav_logo [24.5.2020.]

[7] Newegg [online], dostupno na: https://www.newegg.com/d/Product/PowerSearch?SubCategory=343&N=100007671&IsNodeId=1 [24.5.2020.]

[8] Links [online], dostupno na: https://www.links.hr/hr/discounted-products/informatika-IT [25.5.2020.]

[9] Shopify [online], dostupno na: https://www.shopify.com/ [25.5.2020.]

[10] A. Vyas, 3.10.2018., MVC Pattern [online], Medium, dostupno na: https://medium.com/@anshul.vyas380/mvc-pattern-3b5366e60ce4 [29.5.2020.]

[11] Colorlib, 21.3.2018., Electro [online], dostupno na: https://colorlib.com/wp/template/electro/ [29.5.2020.]

[12] Bootstrap [online], dostupno na: https://getbootstrap.com/ [29.5.2020.]

[13] FontAwesome [online], dostupno na: https://fontawesome.com/ [29.5.2020.]

[14] TutorialsTeacher tim, ASP.NET MVC Architecture [online], dostupno na: https://www.tutorialsteacher.com/mvc/mvc-architecture [31.5.2020.]

[15] E. Musso, 7.3.2020., Entity Framework in C# [online], C#Corner, dostupno na: https://www.c-sharpcorner.com/article/entity-framework-introduction-using-c-sharp-part-one/ [31.5.2020.]

[16] FancyBox [online], dostupno na: http://fancybox.net/ [7.6.2020.]

[17] PayPal tim, How do I add PayPal checkout to my custom shopping cart? [online], dostupno na: https://www.paypal.com/ms/smarthelp/article/how-do-i-add-paypal-checkout-to-my-custom-shopping-cart-ts1200 [9.6.2020.]

[18] DataTables [online], dostupno na: https://datatables.net/ [13.6.2020.]

[19] DropZone [online], dostupno na: https://www.dropzonejs.com/ [13.6.2020.]

[20] SortableJS [online], dostupno na: https://sortablejs.github.io/Sortable/ [14.6.2020.]

[21] Kim Jacobson, 11.5.2020., Hashing: What You Need to Know About Storing Passwords [online], Security Boulevard, dostupno na: https://securityboulevard.com/2020/05/hashing-what-you-need-to-know-about-storing-passwords/ [25.7.2020.]

SAŽETAK

Internet trgovina u današnje vrijeme je sve više zastupljenija u odnosu na klasičnu trgovinu. Prema tome, programska rješenja internet trgovine iznimno su popularna. Takva rješenja se dijele na rješenje kao samog web mjesta trgovine i rješenje kao platforme, odnosno programske okvire za izradu i upravljanje takvog web mjesta.

U okviru ovog rada je, prema postavljenim zahtjevima na programsko rješenje, izrađena internet trgovina koja je namijenjena prvenstveno manjim lokalnim tvrtkama za prodaju računala i računalne opreme. Internet trgovina se sastoji od dva modula – korisničkog i administratorskog. Korisnički modul omogućuje pregled, pretraživanje, kupovinu proizvoda kao i uvid u povijest naručenih proizvoda. Administratorski modul sadrži značajke za upravljanje kategorijama, proizvodima i narudžbama. Prilikom završetka kupovine, korisnik je preusmjeren na PayPal Sandbox testno okruženje za plaćanje. Kupovinu mogu obavljati i gosti, odnosno neregistrirani korisnici.

Zamjenom testnog okruženja za plaćanje sa aktivnim, navedeno se programsko rješenje jednostavno može pretvoriti u komercijalno. Buduća nadogradnja ovog programskog rješenja bi bila implementacija liste želja kao i sustava komentiranja i ocjenjivanja proizvoda.

Ključne riječi: ASP.NET MVC, Entity Framework, internet trgovina, online plaćanje

ABSTRACT

E-commerce nowadays is increasingly present compared to classic commerce. Therefore, e-commerce software solutions are quite popular. Such solutions are either a web shop itself or a platform, i.e. a framework for development and management of such a web shop.

As a part of this thesis, and according to the requirements for the software solution, an online store was created, intended primarily for smaller local companies that sell computers and computer equipment. The online store consists of two modules: a user module and administrator module. The user module allows customers to view, search and purchase products as well as to view the history of the products they have ordered. The administrator module has features for managing categories, products and orders. Upon completion of the purchase, the user is redirected to the PayPal Sandbox test environment for payment. Purchases can also be made by guests, i.e. unregistered users.

By replacing the sandboxed payment environment with an active one, the online store can be used commercially. Possible future upgrades of this online store would be the implementation of a wish list as well as a product commenting and rating system.

Key words: ASP.NET MVC, Entity Framework, online payment, web shop

ŽIVOTOPIS

Marko Huljak rođen je u Virovitici, Republika Hrvatska, 10. studenog 1995. godine. Završio je Osnovnu školu Vladimira Nazora u Virovitici, nakon koje 2010. godine upisuje Tehničku školu u Virovitici, smjer elektrotehnika, koju završava 2014. godine. Iste godine upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Preddiplomski studij završava 2017. godine, zatim na istom fakultetu upisuje diplomski studij računarstva, blok informacijskih i podatkovnih znanosti. Tijekom diplomskog studija u Osijeku počinje raditi na radnom mjestu razvojnog programera u tvrtki GDi.

Vlastoručni potpis

_____

Marko Huljak

PRILOZI

[P1] DVD s programskim kôdom

[P2] Programski kôd [online], dostupno na: https://github.com/MAD3RO/WebShop