Izrada web trgovine s direktnim pristupom bazi podataka koristeći LINQ

Galović, Ivan

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:630840

Rights / Prava: Attribution-NonCommercial-NoDerivs 3.0 Unported/Imenovanje-Nekomercijalno-Bez prerada 3.0

Download date / Datum preuzimanja: **2024-04-18**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Galović

IZRADA WEB TRGOVINE S DIREKTNIM PRISTUPOM BAZI PODATAKA KORISTEĆI LINQ

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ivan Galović

Matični broj: 45860/17–R

Studij: Informacijski sustavi

IZRADA WEB TRGOVINE S DIREKTNIM PRISTUPOM BAZI PODATAKA KORISTEĆI LINQ

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Danijel Radošević

Varaždin, kolovoz 2020.

Ivan Galović

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog rada je izrada web trgovine u programskom jeziku C# i povezane baze podataka nad kojom se izvršavaju upiti pomoću LINQ-a koji je direktno implementirani okvir(engl. framework) u C#-u. Na početku rada objašnjeni su metode i tehnike rada koje su korištene tijekom implementacije web trgovine. Zatim je opisan logički model aplikacije ERA modelom, ulogama svakog korisnika pomoću navigacijskih dijagrama i postavljanjem Entity Framework-a kako bi se mogao koristiti LINQ. Slijedi implementacija u kojoj su objašnjene glavne stranice(engl. Master pages) te kako bi izgledao header, body i footer svake stranice i spojio sa sadržajem svake od podređenih stranica koristeći bootstrap okvir . Nakon toga detaljnije se objašnjava korištenje različitih LINQ upita na stranicama kako bi se popunjavale tablice, repeater-i i padajuće liste , ali i koristile implementirane procedure iz baze podataka. Na kraju rada dan je zaključak u kojemu se opisuju prednosti korištenja ovih tehnologija i komentari na obrađenu temu.

Ključne riječi: web trgovina; C#;LINQ;ERA model; Entity Framework; Master page; Bootstrap;

Sadržaj

1. Uvod 1
2. Metode i tehnike rada 2
2.1. Visual Studio IDE 2
2.2. Programski jezik C# 3
2.3. ASP.NET okvir 3
2.4. SQL Server 4
2.5. Entity Framework 4
2.6. LINQ 4
2.7. Bootstrap 5
3. Logički model aplikacije 5
3.1. ERA model 5
3.2. Navigacijski modeli 7
3.3. Postavljanje potrebnih tehnologija za rad 9
4. Fizički model 11
4.1. Uloga administratora 11
4.1.1. Administrator master page 11
4.1.1.1. Navigacija 12
4.1.2. Prikaz stranice administratora 13
4.2. Uloge neregistriranog i registriranog korisnika 14
4.2.1. Prikaz zajedničke stranice registriranog i neregistriranog korisnika 14
4.3. LINQ upiti korišteni na stranicama 15
4.3.1. Popunjavanje padajuće liste koristeći LINQ 16
4.3.2. Join i objekti anonimnog tipa 17
4.3.3. Queryable<T> FirstOrDeafult() metoda i lambda izrazi 19
4.3.4. Korištenje generiranih procedura u usporedbi s LINQ-om 19
5. Zaključak 22
6. Popis literature 23
7. Popis slika 24

1. Uvod

Internetska stranica se sastoji od niza dokumenata kojima se pristupa pomoću web preglednika. Ona omogućava prikaz teksta i poveznica(linkova) kojima se pristupa preko URL (Uniform Resource Locator) adrese. Osim poveznica i tekstova, na internetskoj stranici se mogu prikazati i multimedijski dokumenti kao što su slike, zvukovi i drugi. Jednostavnije internetske stranice odgovaraju pojmu web stranica koja je napisana u HTML-u, dok složenije odgovaraju pojmu web aplikacija.

Web aplikacija koristi programski sustav koji generira web stranice i dokumente napisane u programskom jeziku koje se izvršavaju na poslužitelju. Također, koristi se baza podataka u koju se spremaju slike, videi i rezultati korisničkog pretraživanja, a na kraju se ti podaci prikazuju korisnicima.

Kako su se razvojem tehnologije razvili tableti i pametni telefoni, danas se više ne pristupa web stranicama samo putem osobnih računala. Prema tome, dizajner web stranica treba misliti i na prilagodbu veličini zaslona raznih uređaja (responzivnost) koji podržavaju pristup internetu.

Razvijene su i mnoge pogodnosti korištenja interneta kao što su npr. internet bankarstvo, čitanje vijesti na Internetu, kupnja proizvoda putem interneta itd. Upravo zbog sve češće korištenosti interneta i kupovine na njemu sam i ja pokušao napraviti web aplikaciju koja će zadovoljiti potrebe online kupovine (engl. webshop).

Jedan od takvih rješenja jest LINQ, language integrated query, koji svojom funkcionalnošću i samom definicijom podsjeća na klasični ORM iako on to nije. Naime, radi se o rješenju koje je integrirano u programski jezik C# te eliminira (ali ne isključuje) potrebu za middleware programima te uvelike pojednostavljuje proces programerima i ubrzava ga korisnicima. LINQ kao takav stoga je vrlo korišteno rješenje budući da, osim što je efikasan, nudi i jednostavnost sintakse koja je vrlo slična SQL-u stoga ju je vrlo lako za naučiti.

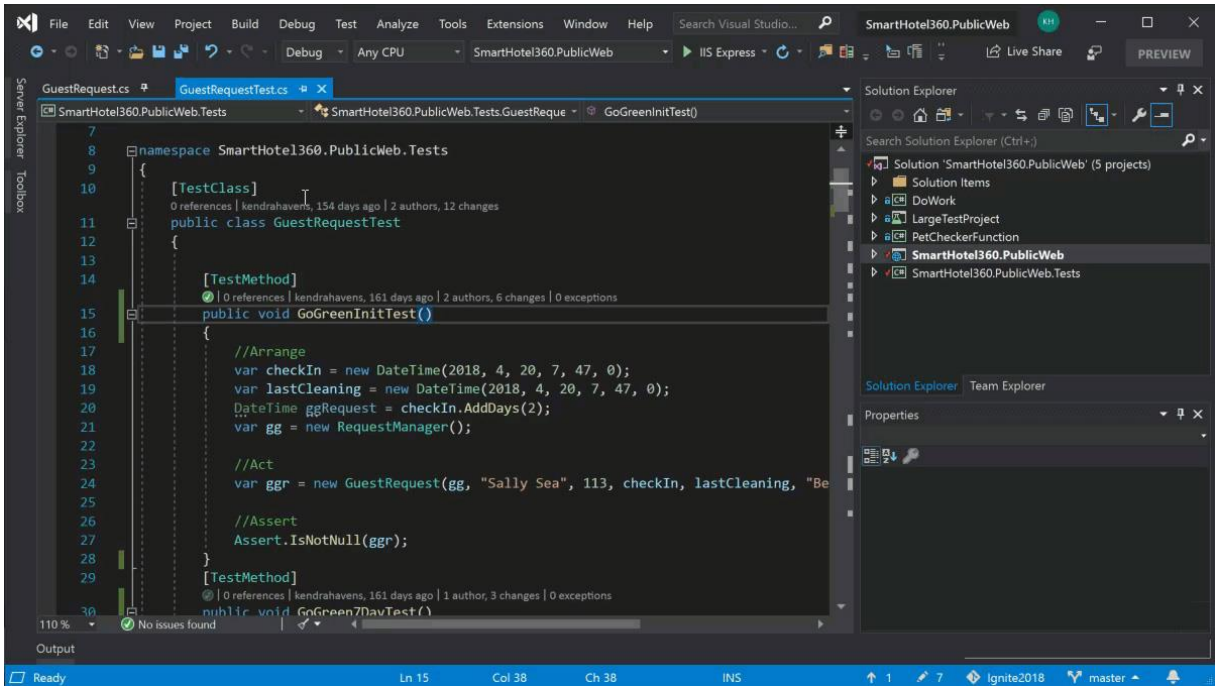
U ovome radu tako ću reći nešto o alatima koji su se koristili, a zatim objasniti postavljanje samog logičkog modela aplikacije. Kao svojevrsnu sintezu rada iskoristit ću zadnji naslov u kojem ću sve to potkrijepiti fizičkim modelom web aplikacije na kojemu ću objasniti izgled stranice i korištenje LINQ funkcionalnosti na istoj.

2. Metode i tehnike rada

Prilikom izrade web aplikacije korišteni su sljedeći alati: Visual Studio IDE, C#, ASP.NET, SQL Server , LINQ i Bootstrap razvojni okvir. Visual studio je razvojno okruženje koje se koristi za servise, računalne , mobilne i web aplikacije. C# je jedan od skupa programskih jezika u okviru Visual Studi-a. Jedan od servisa koje sadrži Visual Studio je i ASP.NET, a on se koristi za razvoj web aplikacija. Baza podataka napravljena je pomoću SQL Server Managment Studi-a, a sama implementacija unutar Visual Studi-a omogućena je pomoću Entity Frameworka. Nadalje će ukratko biti opisani alati korišteni prilikom realizacije ovog projekta.

2.1. Visual Studio IDE

Microsoft Visual Studio je integrirano razvojno okruženje (IDE) Microsofta. Koristi se za razvoj računalnih programa, kao i web stranica, web aplikacija, web usluga i mobilnih aplikacija[1]. Visual studio podržava razvoj aplikacija za sve Microsoftove platforme i jezike(C#, F#, VisualBasic, C++) te uz njih omogućuje korištenje jezika za web(HTML5,CSS,Javascript). Tijekom izrade web aplikacije korišten je Visual Studio 2019 , a licenca za njega dobivena je od strane Fakulteta organizacije i informatike pomoću MSDNAA repozitorija.



Slika 1: Visual Studio IDE [12]

2.2. Programski jezik C#

C# je danas programski jezik opće namjene. Uključuje mogućnost jakih i dinamičnih tipova, imperativno, deklarativno, funkcionalno, generičko, objektno i komponentno orijentirano programiranje. Jezik također sadrži generiku, parcijalne tipove, anonimne metode, iteratore, anonimne tipove, upitne izraze, lambda izraze, parcijalne metode, dinamičko povezivanje, ugrađene tipove za interoperabilnost, asinkrone metode, podršku za paralelizam itd.[2]. Nekoliko C# značajki potpomažu konstrukciji robusnih i izdržljivih aplikacija: Recikliranje automatski čisti memoriju koju zauzimaju nekorišteni objekti. Rukovanje iznimkama pruža strukturalan i ekstenzivan pristup otkrivanju grešaka i oporavku. Sigurnost jezika čini čitanje ne inicijaliziranih varijabli nemogućim, kao i indeksiranje polja izvan njihovih granica. [3] . Koristi se iz širokog raspona razloga, ali njegova popularnost leži u njegovoj upotrebi za sljedeće zadatke:

1. **Backend usluge**
2. **Windows aplikacije**
3. **Izrada web stranica**
4. **Razvoj igara**

2.3. ASP.NET okvir

.NET okvir je Microsoftov okvir, odnosno skup biblioteka koje pojednostavljuje razvoj aplikacije namijenjene za Microsoft tehnologije; Windows, Windows Store, Windows Phone, Windows Server i Windows Azure[2]. Dvije osnovne komponente .NET okvira su:

1.Common Language Runtime(CLR)- sloj apstrakcije između koda i operacijskog sustava

2.(Base) Class Library(BCL) – skup biblioteka s predefiniranim klasama i API-ima

Jedan od najčešće korištenih okvira je i ASP.NET okvir koji se koristi za razvoj web aplikacija na strani poslužitelja[4]. Može se koristiti za stvaranje dinamičnih web stranica, web aplikacija i web usluga. ASP.NET koristi HTTP naredbe i pravila za postavljanje bilateralne komunikacije i suradnje između pretraživača i servera[5]. Nudi velik broj kontrola, poput tekstualnih okvira i gumba te konfiguriranje i manipuliranje koda radi stvaranja HTML stranica.

2.4. SQL Server

SQL Server je jedan od najčešće korištenih sustava za upravljanjem bazom podataka koji koristi SQL upitni jezik za dodavanje novih i manipulaciju postojećih podataka. U SQL Server Management Studi-u(SSMS) napravljena je čitava baza podataka koju ću nešto kasnije objasniti.

2.5. Entity Framework

Jedan od ORM (eng. Object Relational Mapping) alata je i Entity Framework(ADO.NET). ADO.NET je skup klasa za interakciju s izvorima podataka kao što su baze podataka i XML datoteke[6]. Temelji se i dalje na objektima tipa Connection, Command i DataReader, ali omogućava rad s podacima na višoj razini apstrakcije. Glavna korist ovog okvira (frameworka) je ta da programer ne mora misliti o strukturi baze podataka jer može raditi sa konceptualnim modelom podataka koji reflektira poslovne objekte iz baze[7]. Prema tome, nije više potrebno pisati SQL upite za dohvaćanje podataka iz baze i unos podataka u bazu.

2.6. LINQ

Tehnologija kojoj Entity Framework omogućava pristup, modeliranje i izvršavanje upita nad podacima generiranim od strane aplikacije je LINQ(Language Integrated Query). Dio je .NET Frameworka te je C# njegov „matični“ programski jezik. Kao takav on služi kao temelj za logiku objektno – orijentiranoga programiranja kojim se LINQ služi.[8]

Jedna od prednosti korištenja LINQ-a nad drugim mogućnostima jest njegova jednostavna i efikasna priroda koja ne traži nikakve posrednike između aplikacije i njezinih podataka. Osim što je efikasan, LINQ se može pohvaliti i iznimno jednostavnom sintaksom koja se uvelike bazira na sintaksi SQL-a, ali koristeći objektno – orijentirani način pisanja koda na kojemu je baziran C#. Ovo je vidljivo u načinu na koji LINQ obrađuje upite; on ih naime, za razliku od klasičnog jezika za modeliranje podacima, na upite gleda kao na klase ili metode što je vrlo blisko objektno – orijentiranim programerima stoga i lako shvatljivo[9].

Može se, dakle, reći kako LINQ uzima najbolje od oba svijeta: moderni objektno – orijentirani pogled na upite, ali koristeći iznimno jednostavnu sintaksu baziranu na već mnogima poznatom SQL-u.

2.7. Bootstrap

Bootstrap je divovska zbirka praktičnog koda koja se može upotrebljavati u HTML-u, CSS-u i Javascript-u. To je ujedno i razvojni okvir koji programerima i dizajnerima omogućuje brzu izradu potpuno odgovarajućih responzivnih web stranica i web aplikacija[10].

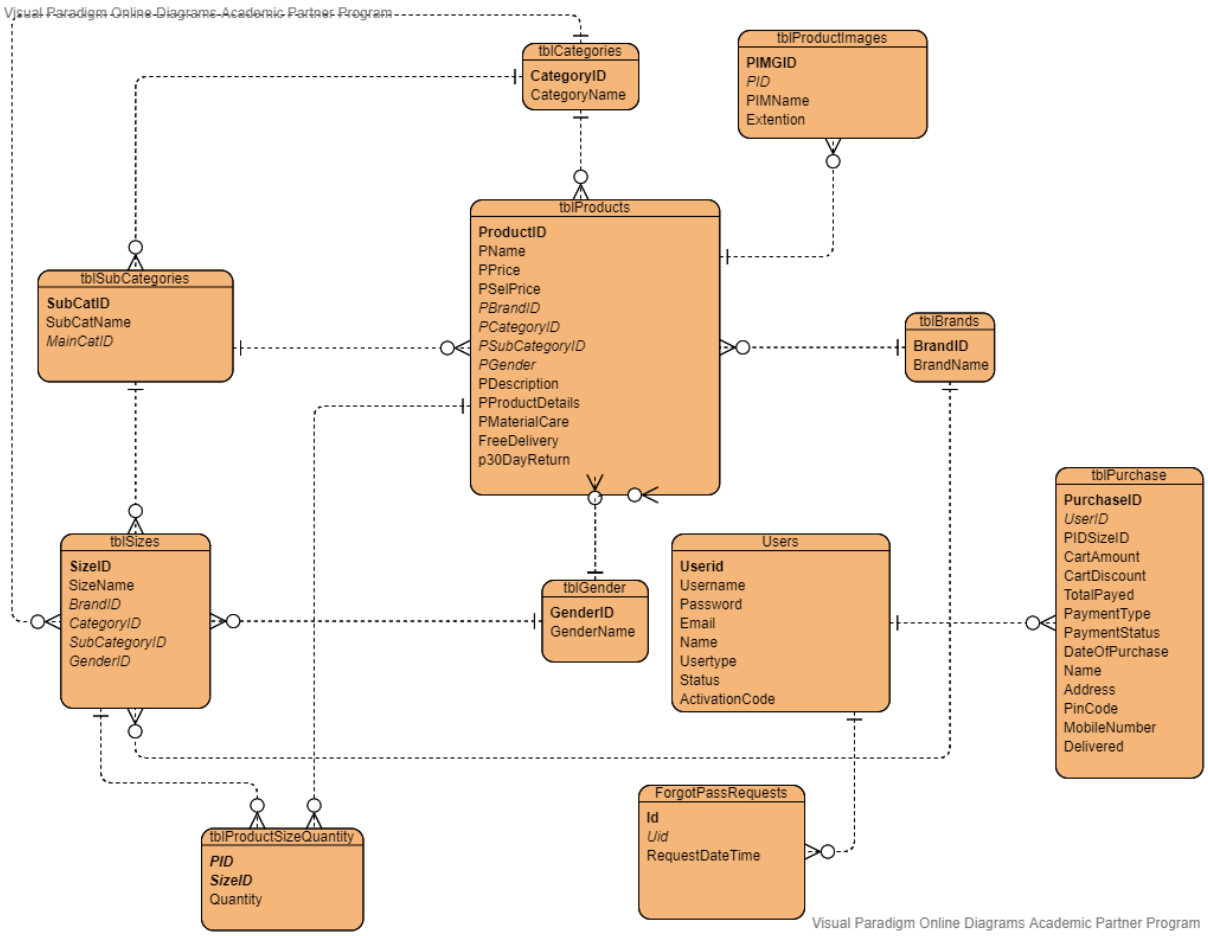
3. Logički model aplikacije

U ovom dijelu rada opisat ću aplikacijsku domenu ovog rada , prikazati i objasniti ERA model i mogućnosti uloga korisnika pomoću navigacijskih dijagrama.

U praktičnom dijelu ovog rada izradit ću web aplikaciju koja će se baviti naručivanjem odjeće i njenim upravljanjem putem internet trgovine(engl. webshopa). U aplikaciji će postojati tri tipa korisnika: neregistrirani korisnik, registrirani korisnik i administrator. Neregistrirani korisnik će moći pogledati sve proizvode koji se nalaze u ponudi internet trgovine. Kako bi uspio staviti proizvode u košaricu i naručivati proizvode morat će se prvo registrirati kako bi postao registrirani korisnik. Na kraju, administrator će imati opcije dodavanja ,uređivanja i brisanja proizvoda, kategorija i potkategorija, njihovih marki, veličina i pregledavanja narudžbi od strane registriranog korisnika.

3.1. ERA model

Na slici 2 prikazan je ERA model baze podataka u koju se spremaju podaci aplikacije. Mogućnosti unutar web aplikacije objašnjene su u prethodnom poglavlju. Kako bi korisnik mogao naručivati proizvode s web trgovine, mora se prethodno na nju registrirati. Tome služi relacija „Users“. Prema toj relaciji, svaki korisnik mora imati korisničko ime(Username), lozinku(Password), email, ime i prezime(Name) , tip korisnika(Usertype), status(Status) i aktivacijski kod (ActivationCode). Tip korisnika može biti U kao User ili A kao Administrator.

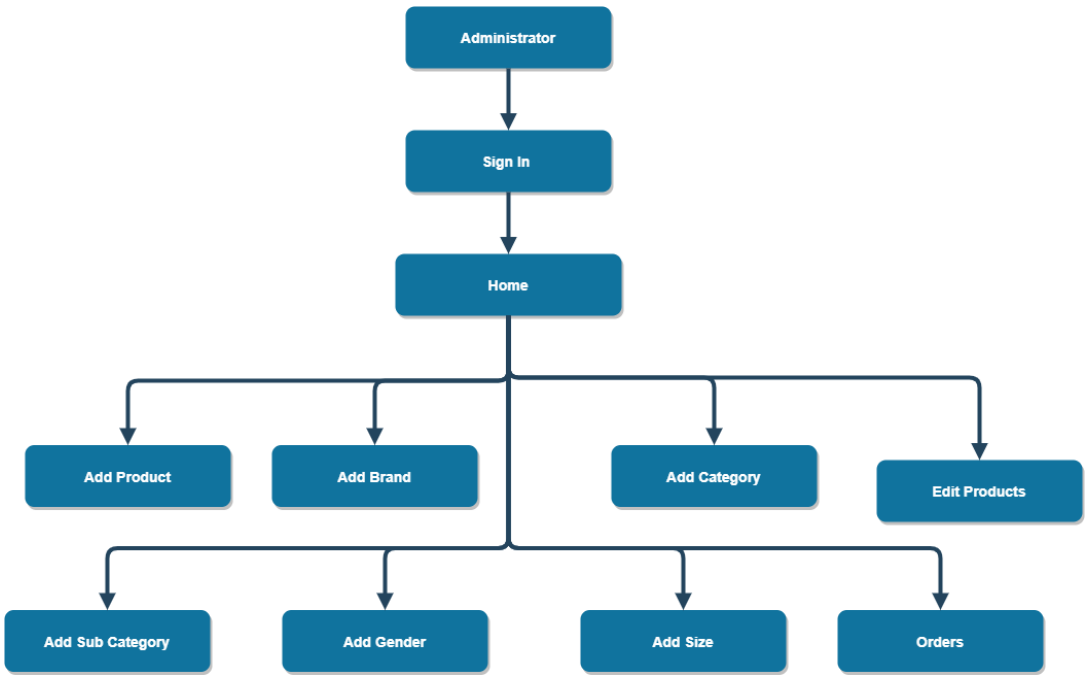


Slika 2: ERA model (slika autora)

U slučaju da je korisnik administrator može dodavati, izmjenjivati ili brisati proizvode (relacija„tblProducts“), marke (relacija„tblBrands“), spol (relacija„tblGender“), veličinu odjeće (relacija„tblSizes“),kategorije(relacija„tblCategories“),potkategorije(relacija„tblSubCategories“) i narudžbe(relacija „tblPurchase“). Relacije „tblBrands“, „„tblGender“, „„tblCategories“ imaju vrlo jednostavnu strukutru;sastoje se od primarnog ključa i jednog dodatnog atributa. Ostale tablice imaju složeniju strukturu. Jedna kategorija može imati više potkategorija pa relacija „tblSubCategories“ ima vanjski ključ „MainCatID“ koji se referencira na primarni ključ „CategoryID“ relacije „tblCategories“. Veličina se može odnositi na više marki, spolova, kategorija i potkategorija pa relacija „tblSizes“ ima vanjske ključeve na svaku od spomenutih relacija. Najvažnija relacija ERA modela je „tblProducts“. Kod dodavanja ili izmjene proizvoda, kao i kod relacije „tblSizes“, proizvod se može odnositi na više marki, spolova, kategorija i potkategorija pa također imamo vanjske ključeve na sve te relacije. Prilikom dodavanja proizvoda moramo mu napisati ime(atribut „PName“), cijenu(atribut „PPrice“), prodajnu cijenu(ako je proizvod na popustu, atribut „PSelPrice“), odabrati odgovarajuću marku, kategoriju, spol za koji je proizvod namijenjen, opis proizvoda, materijal od kojeg je napravljena

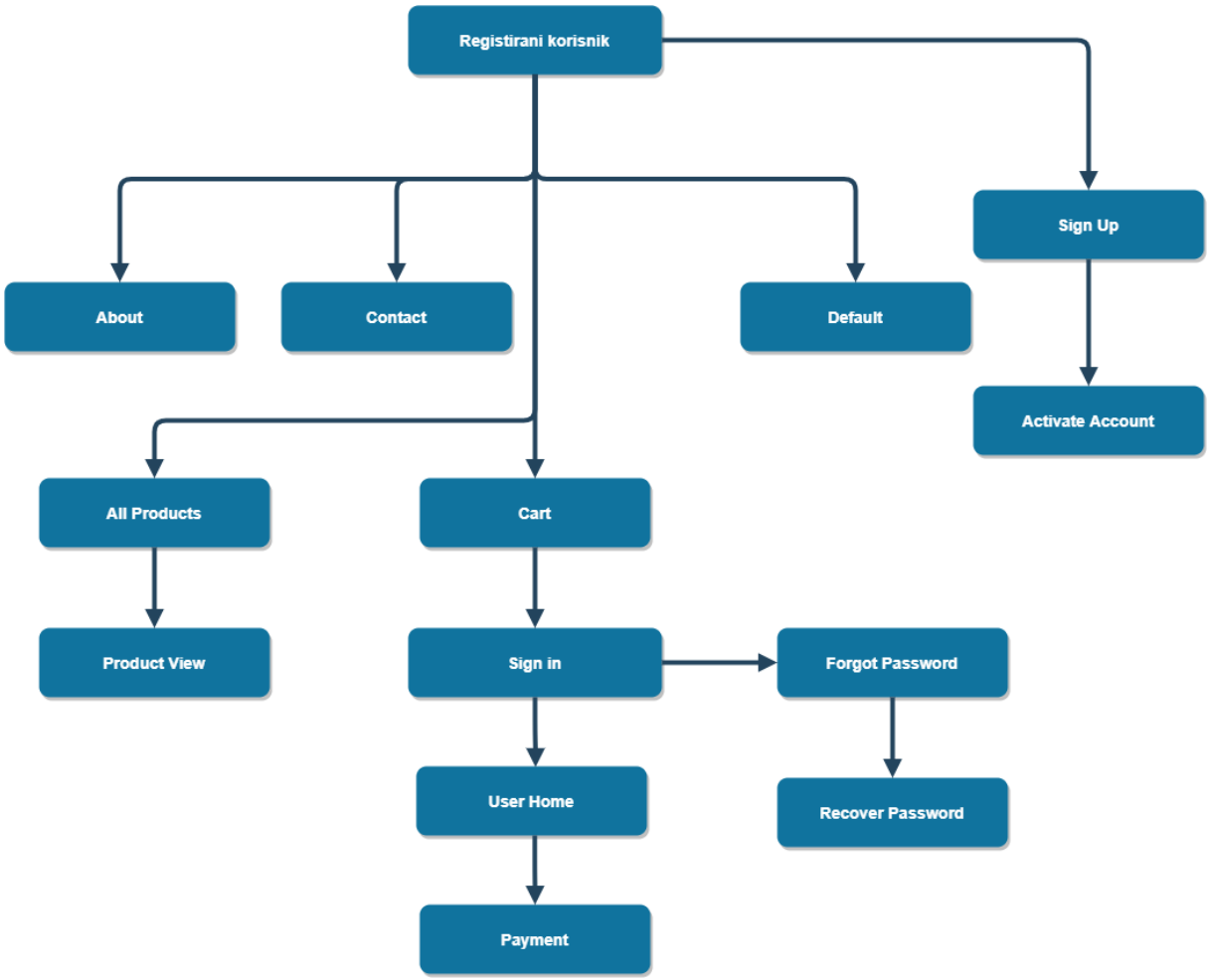
odjeća(atribut “PMaterialCare“), je li besplatna dostava i je li moguće vratiti proizvod u roku od 30 dana . Prilikom dodavanja proizvoda popunjavaju se i relacije „tblProductSizeQuantity“ i „tblProductImages“. U relaciju „tblProductSizeQuantity“ se unosi količina koja je dostupna za određeni proizvod i njegovu veličinu. Ova relacija je ujedno slabi entitet jer sadrži vezu više naprema više. Svaki proizvod može imati više veličina , a i veličina se može odnositi na više proizvoda, tako da relacija sadrži primarni ključ koji se sastoji od više atributa koji su ujedno i vanjski ključevi na relacije „tblProducts“ i „tblSizes“. Relacija „tblProductImages “ popunjava se slikom za određeni proizvod(vanjski ključ „PID“) i ekstenzijom(atribut „Extention“). Jedina relacija koja je još ostala za objasniti je „tblPurcahase“. U nju se podaci upisuju prilikom narudžbe kupca koji ima ulogu „user“. Svaka narudžba ima svoj jedinstveni broj narudžbe(atribut „PurchaseID“), korisnika koji je naručio narudžbu(atribut „UserID“), proizvod i veličinu proizvoda(atribut „PIDSizeID“), iznos košarice(atribut „CartAmount“), popust(ukoliko ga je bilo- atribut „CartDiscotunt“), ukupnu cijenu narudžbe(atribut „TotalPayed“), način plaćanja (karticom ili prilikom isporuke- atribut „Payment Type“), status narudžbe (može biti plaćena ili ne plaćena, ovisno o vrsti plaćanja), datum narudžbe (atribut „DateOfPurchase“) i je li narudžba dostavljena ili nije (atribut „Delivered“). Uz sve to sadrži i osobne podatke naručitelja kao što su ime i prezime(atribut „Name“), adresa (atribut „Address“) i telefonski broj (atribut „MobileNumber“).

3.2. Navigacijski modeli



Slika 3: Navigacijski model- administrator

U web aplikaciji imamo tri navigacijska modela : administrator, registrirani korisnik i neregistrirani korisnik. Administrator, kako bi mogao nešto raditi, mora se prvo prijaviti u aplikaciju . Prijavom dolazi na početnu stranicu za administratora s koje onda može ići na bilo koju od stranica. Njegove mogućnosti na svakoj stranici objašnjene su u prethodnom poglavlju.



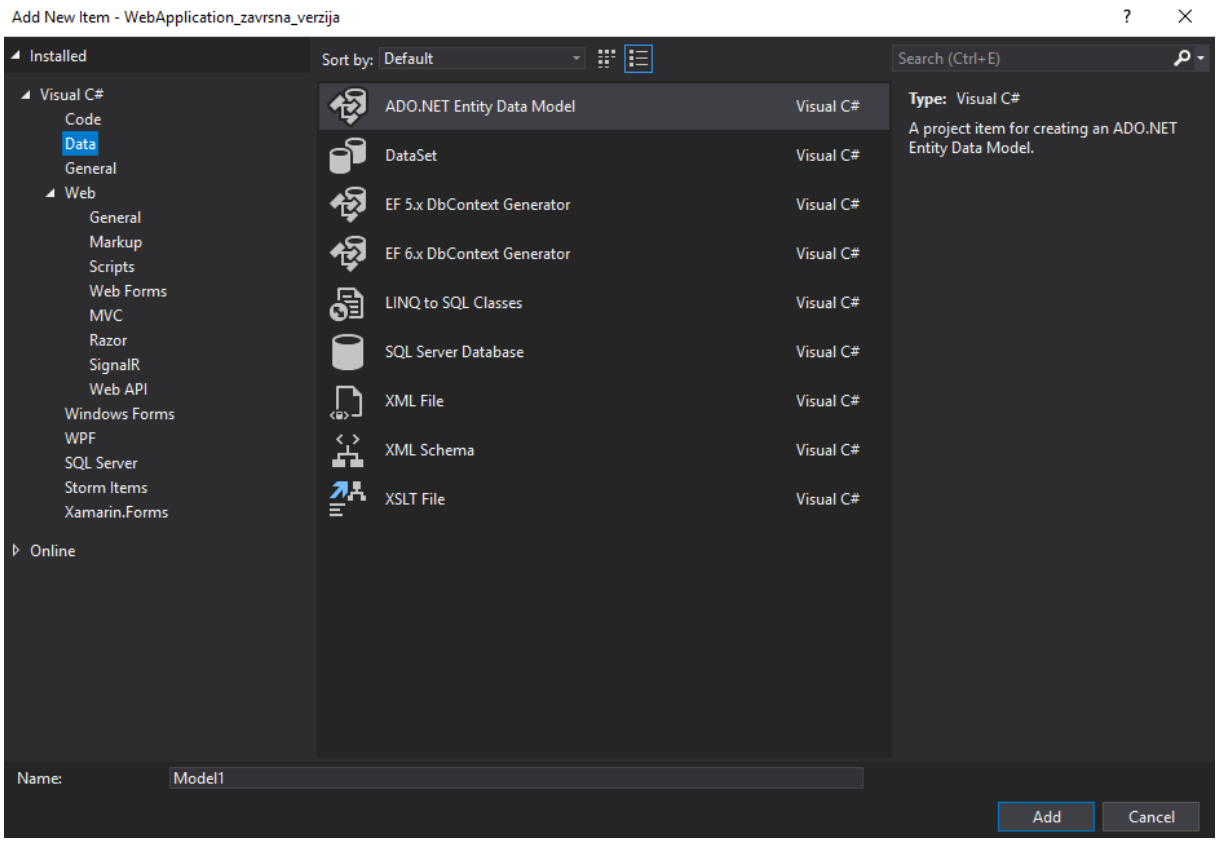
Slika 4: Navigacijski model- registrirani korisnik

Stranice kojima korisnik može pristupiti bez prijave su: „About“ , „Contact“, „Default“(Naslovna), „All Products“ , „Cart“ ,„Sign up“, „Sign in“, „Forgot Password“ i „Recover Password“. Klikom na određeni proizvod sa stranice „All Products“ može otići na „Product View“ i dodati taj proizvod u košaricu. Korisnik može vidjeti sve svoje proizvode u košarici(„Cart“) , ali kako bi ih mogao naručiti , mora se prijaviti u aplikaciju . Prilikom prijave u aplikaciju, korisnik može otići na zaboravljenu lozinku („Forgot Password“) te nakon nje zatražiti slanje lozinke na mail. Prijavom u aplikaciju korisnik može otići na stranicu plaćanja(„Payment“) kako bi naručio proizvode. Jedina razlika između registriranog korisnika i

neregistriranog korisnika je ta što se neregistrirani korisnik ne može prijaviti u aplikaciju, a samim time ne može naručiti proizvode s web trgovine.

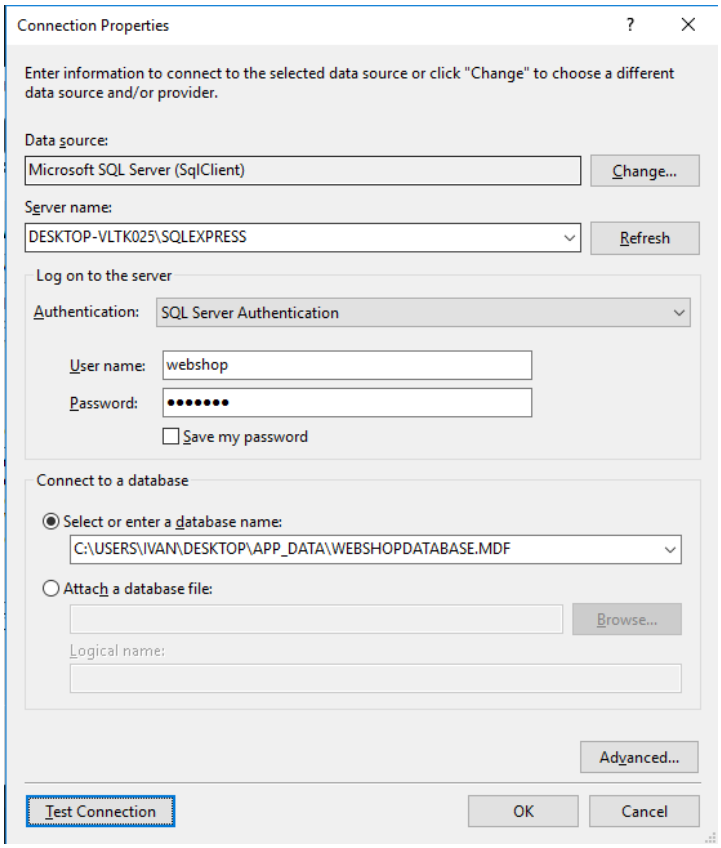
3.3. Postavljanje potrebnih tehnologija za rad

Nakon odabira web aplikacije za naš projekt u Visual Studi-u ,slijedi dodavanje veze na bazu podataka kako bismo koristili LINQ. Prije nego što krenemo na sam postupak potrebno je instalirati okvir(framework) za Entity Framework. Nakon instalacije desnim klikom na naš projekt i odabirom Add->New Item otvara nam se ovakav prozor:



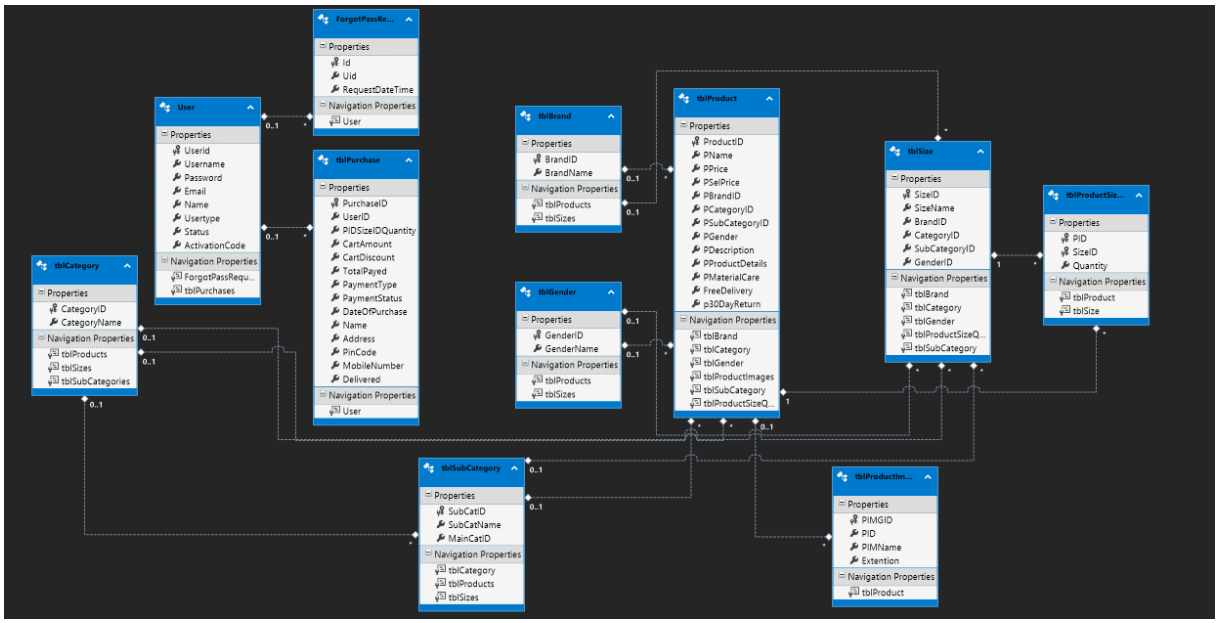
Slika 5: Odabir ADO.NET Entity Data Modela

Pod Visual C# odjeljkom kliknemo na Data i nakon toga odaberemo ADO.NET Entity Data Model. Upišemo odgovarajuće ime i kliknemo na Add. U sljedećem prozoru koji nam se otvori odaberemo EF Designer from Database jer želimo generirati Entity Data Model iz postojeće baze podataka. Nadalje, odaberemo New Connection i postavimo vezu na bazu podataka iz koje želimo generirati model i kliknemo OK.



Slika 6: Postavljanje veze na bazu

Označimo Yes, include the sensitive data in the connection string i Save connection settings in Web.config as: i odaberemo Next. Nakon toga odaberemo koje relacije, poglede , procedure i funkcije želimo staviti u naš model i kliknemo Finish. Dobivamo sljedeći generirani model.



Slika 7: ADO.NET Entity Data Model

Za svaku relaciju i njezine atribute Visual studio je izgenerirao klasu s istim imenom svake relacije i svojstva u klasi za sve atribute u relaciji. Također nam je izgenerirao i metode za svaku funkciju i proceduru. Ovime smo završili logički model i možemo krenuti s fizičkim kreiranjem aplikacije.

4. Fizički model

U ovom poglavlju rada napisat ću upute o jednoj web stranici za ulogu administratora i za ulogu registriranog korisnika i čemu svaka od njih služi te na kraju to potkrijepiti izgledom tih stranica i objasniti LINQ upite koje sam koristio za dohvaćanje podataka iz baze . Glavna tematika ove web aplikacije je prodaja odjeće, ime obrta je DEMA. Kao i svaka web trgovina i moja će imati web stranice tipične za nju : naslovnica za običnog korisnika i administratora, stranica za registraciju korisnika, prijava korisnika, stranica koja sadrži sve proizvode, stranica o nama(About), stranica za kontakt(Contact), stranica za dodavanje u košaricu itd. Za svaku ulogu napravljena je glavna stranica(Master page) . Svrha glavne stranice je da dizajn bude konzistentan za svaku stranicu u web aplikaciji. Ona definira izgled i ponašanje za sve stranice u aplikaciji.

4.1. Uloga administratora

Počnimo sa ulogom administratora. Administrator može dodavati, uređivati ili obrisati proizvode, spolove, marke, kategorije, potkategorije, veličine i narudžbe dobivene na stranici.

4.1.1. Administrator master page

Master page „AdminMaster.master“ bit će glavna stranica za sve stranice kojima će administrator moći pristupiti. Sadržaj svake stranice koju korisnik uloge administrator zatraži spojit će se sa sadržajem zatražene stranice. Kako sam naveo u drugom poglavlju koristit ću bootstrap razvojni okvir , a njegovo pozivanje nalazit će se u head dijelu svake stranice.

```
<head runat="server">
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Welcome</title>
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet" />
    <link href="css/Custom-Cs.css" rel="stylesheet" />
</head>
```

4.1.1.1. Navigacija

Dio navigacije prikazan je sljedećim kodom:

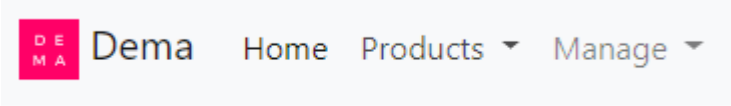
```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
<a class="navbar-brand" href="AdminHome.aspx">
<img alt="Logo" src="Images/DemaLogo.png" height="30"/>Dema </a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">
                    <ul class="navbar-nav mr-auto">
                        <li class="nav-item active">
                            <a class="nav-link" href="AdminHome.aspx">Home
<span class="sr-only">(current)</span></a>
                        </li>
                            <div class="dropdown-menu" aria-
labelledby="navbarDropdown">
 <a class="dropdown-item" href="AddProduct.aspx">Add Product</a>
                            </div>
              </div>
      </nav>
```

Čitava navigacija napravljena je pomoću bootstrap razvojnog okvira[11]. Najbitnije css klase koje se koriste u ovom primjeru su:

1. Navbar- stil navigacije
2. Navbar-expand- koriste se za odgovarajuće kolapse i klase sheme boja(bg-light)
3. Navbar-brand - za nazive tvrke, proizvoda ili projekta
4. Navbar-toggler- za upotrebu dodataka za kolaps i druge navigacijske promjene
5. Navbar-nav – podrška za padajuću listu
6. Collapse navbar-collapse – grupiranje i skrivanje navbra sadržaja prijelomnom točkom
7. Dropdown-menu – padajući izbornik
8. Dropdown-item – element u padajućoj listi

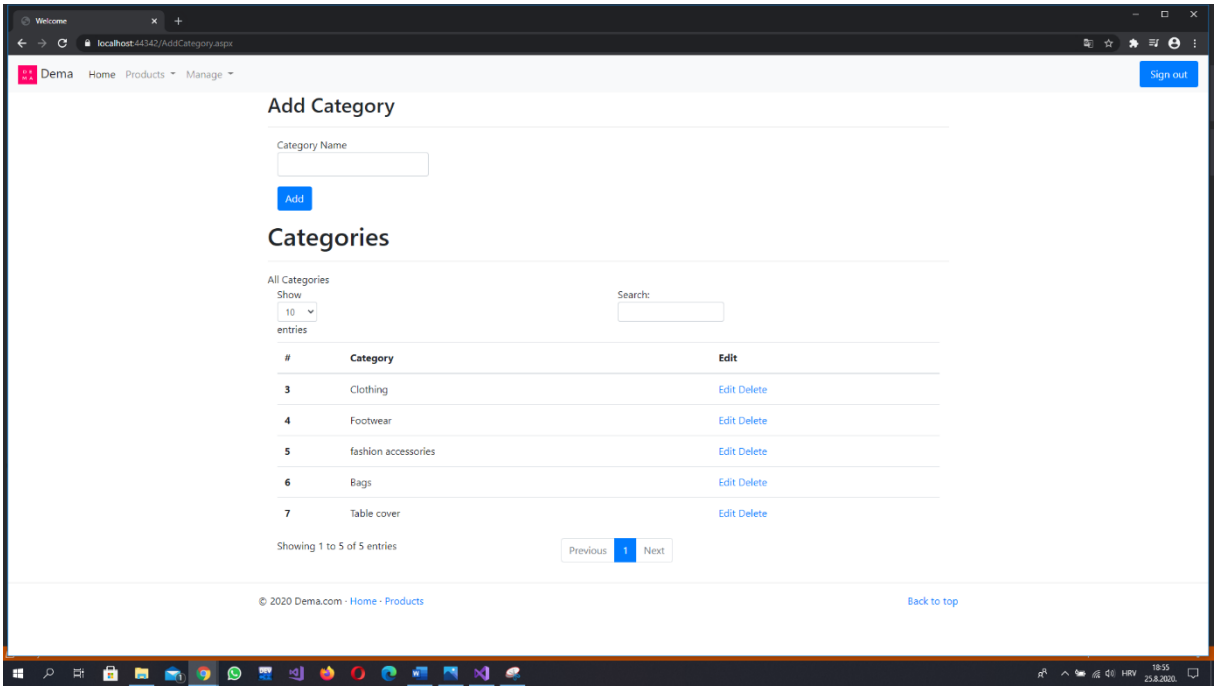
Rezultat koda je navigacija na svakoj stranici koju administrator koristi.



Slika 8: Navigacija kod uloge administrator

4.1.2. Prikaz stranice administratora

Na ovoj stranici moguće je dodati kategoriju na web trgovinu, vidjeti popis kategorija koje nalaze u bazi podataka web stranice , uređivati i obrisati kategoriju.



Slika 9: Izgled stranice AddCategory.aspx

Najbitniji dio ove stranice je tablica u kojoj se prikazuju sve kategorije. Dio koda u htmlu za ovu tablicu :

```
<table id="Categories" class="table">
                        <thead>
                            <tr>
                                <th>#</th>
                                <th>Category</th>
                                <th>Edit</th>
                            </tr>
                        </thead>
                        <tbody>
```

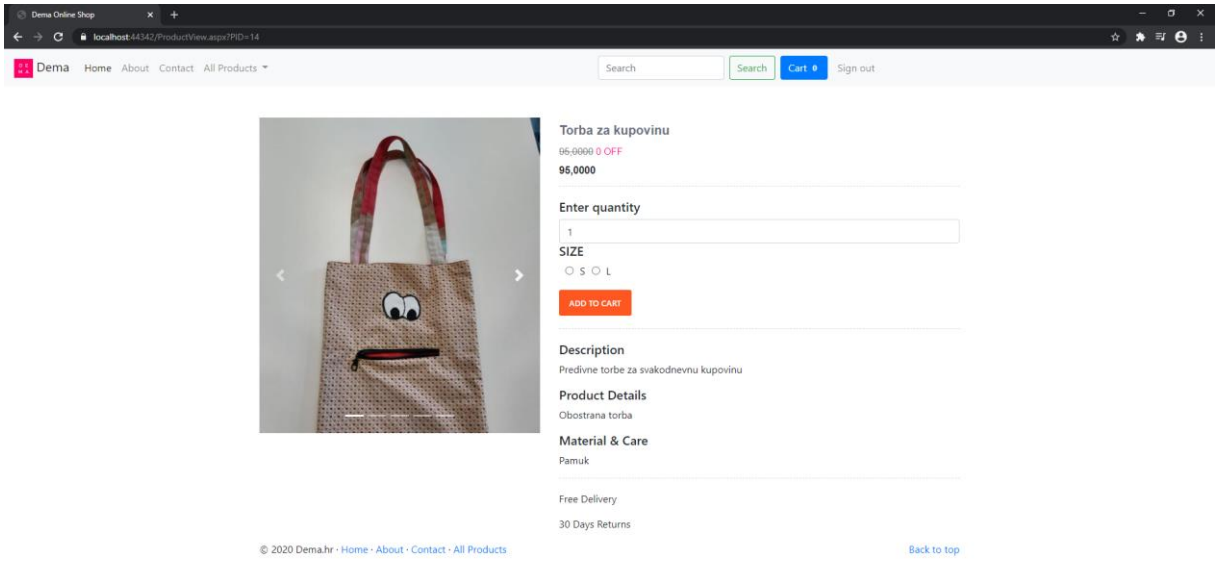
```
                    <tr>
                        <th><asp:Label ID="Label2" runat="server" Text='<%#
Eval("CategoryID") %>'></asp:Label></th>
                        <td><asp:Label ID="Label3" runat="server" Text='<%#
Eval("CategoryName") %>'></asp:Label>
</tr>
</tbody>
</table>
```

4.2. Uloge neregistriranog i registriranog korisnika

Navigacijskim modelima objašnjeno je kojim sve stranicama mogu pristupati registrirani i neregistrirani korisnik. Ovim poglavljem prikazat ću izgled jedne stranice koju koriste registrirani korisnik i neregistrirani korisnik. Za ove stranice neću govoriti o glavnoj stranici(master page) jer je napravljena na sličan način kao i AdminMaster.master.

4.2.1. Prikaz zajedničke stranice registriranog i neregistriranog korisnika

Stranica koju sam odabrao za objasniti je ProductView.aspx



Slika 10: Prikaz stranice ProductView.aspx

Klikom na bilo koji proizvod vidimo slike proizvoda koje možemo listati pomoću slidera, naziv proizvoda, cijenu i cijenu s popustom. Korisnik je u mogućnosti odabrati količinu(ako je

takva dostupna sa skladišta poduzeća) i veličinu proizvoda kojeg želi. Također mu je dan i opis proizvoda , od kojeg je materijala napravljen proizvod, je li besplatna dostava za ovaj proizvod i je li moguć povrat proizvoda u roku od 30 dana. Pritiskom na gumb „Add to Cart“ košarica se automatski popunjava i ažurira se broj prozivoda u navigaciji pod gumbom Cart.

4.3. LINQ upiti korišteni na stranicama

Tablica iz poglavlja 4.1.2. sastoji se od 3 stupca: Id, Category i Edit i popunjava se na temelju LINQ upita:

```
using (var context = new WebshopEntities())
{
 var upit = from c in context.tblCategories select c;
 rptrCategory.DataSource = upit.ToList();
 rptrCategory.DataBind();
}
```

Kako bismo koristili LINQ bitna je prva linija ovog koda. Pomoću nje pristupamo podacima u bazi podataka „Webshop Entites“.Postupak kako bismo mogli koristiti ovu liniju koda objašnjen je u poglavlju 3.3. Vidimo da sintaksa LINQ upita podsjeća na onu SQL-a, stoga ju je vrlo lako za naučiti jer iz njega crpi inspiraciju za jednostavan oblik upita. Upit vraća sve podatke iz relacije „tblCategories“. Na temelju njega popunjava se repeater unutar kojeg se nalazi i ova tablica. Pretvaramo taj upit u listu kako bismo mogli prikazivati podatke u tablici. Na kraju metodom .DataBind() „vežemo“ podatke. Funkcijom Eval prikazujemo podatke koje želimo dobivene od strane upita u dizajnd dijelu stranice(.aspx dijelu koda). Jedino što nam je ostalo je „pregaziti“ .ToString() metodu kako bi se prikazivao atribut „CategoryName“ i to je prikazano kodom:

```
public override string ToString()
{
  return CategoryName;
}
```

Sljedeći kod prikazuje dodavanje nove kategorije na stranicu

```
 using (var context = new WebshopEntities())
 {
  string name = txtCatName.Text;
  tblCategory category = new tblCategory
  {
    CategoryName = name
```

```
};
 context.tblCategories.Add(category);
 context.SaveChanges();
 }
txtCatName.Text = string.Empty;
BindBrandsRptr();
```

Ovaj dio koda ne koristi LINQ, ali pokazuje jednostavnost dodavanja podataka u tablicu pomoću Entity Framework-a. Prvo se, kao i prethodnom primjeru koda, moramo povezati na bazu. Koristimo klasu koju nam je generirao Entity Framework i kako je atribut „CategoryID“ samoinkrementirajući, jedino što nam treba je atribut „CategoryName“. Ostalo je još jedino spremiti promjene u tablici i pozvati metodu BindBrandsRptr() kako bi se ažurirala tablica i prikazala nova kategorija u njoj. Slični upiti nalaze se na sljedećim stranicama: AddBrand.aspx i AddGender.aspx pa za te stranice neću prikazivati LINQ upite.

4.3.1. Popunjavanje padajuće liste koristeći LINQ

Na stranici AddProduct.aspx koristimo puno padajućih listi kako bi se znalo za koju marku se dodaje proizvod, spol , kategoriju , potkategoriju i koju veličinu . Padajuća lista za marku popunjava se na sljedeći način korištenjem LINQ upita:

```
using (var context = new WebshopEntities())
{
 var upit = from b in context.tblBrands select b;
 ddlBrands.DataSource = upit.ToList();
 ddlBrands.DataTextField = "BrandName";
 ddlBrands.DataValueField = "BrandID";
 ddlBrands.DataBind();
 ddlBrands.Items.Insert(0, new ListItem("-Select-", "0"));
}
```

LINQ upit je sličan kao i upit za dohvaćanje kategorija u prethodnom poglavlju. Jedina razlika je ta što se ovaj puta popunjava padajuća lista s rezultatima upita. Također su bitne i metode „.DataTextField“ i „.DataValueField“. Ovo nam je bitno kako bismo u padajućoj listi „ddlBrands“ odabirom string vrijednosti unosili integer vrijednost u tablicu. To će se koristiti i za uspoređivanje vrijednosti u LINQ upitima koje ću objasniti nešto kasnije u ovom poglavlju. Na potpuno jednak način se popunjavaju padajuće liste za kategoriju i spol.

Za sljedeći upit trebao sam dodati padajućoj listi „ddlCategory“ događaj OnSelectedIndexChanged kako bih mogao uspoređivati vrijednosti iz prethodnih padajućih listi i na temelju toga prikazao potrebne podatke administratoru prilikom dodavanja novog proizvoda.

```
protected void ddlCategory_SelectedIndexChanged(object sender, EventArgs e)
 {
    string selectedCategory = ddlCategory.SelectedItem.Value;
    int CategoryID = Convert.ToInt32(selectedCategory);
    using (var context = new WebshopEntities())
    {
       var upit = from sc in context.tblSubCategories
                  where sc.MainCatID == CategoryID select sc;
       ddlSubCategory.DataSource = upit.ToList();
       ddlSubCategory.DataTextField = "SubCatName";
       ddlSubCategory.DataValueField = "SubCatID";
       ddlSubCategory.DataBind();
       ddlSubCategory.Items.Insert(0, new ListItem("-Select-", "0"));
       ddlSubCategory.Enabled = true;
     }
 }
```

LINQ , baš kao i SQL koristi izraze kao što su from, where i select. U ovom upitu, na temelju odabrane kategorije , uspoređuje se vanjski ključ relacije „tblSubCategories“ s odabranom kategorijom iz padajuće liste „ddlCategory“. Samim time , omogućeno je i biranje potkategorije iz padajuće liste „ddlSubCategory“.

4.3.2. Join i objekti anonimnog tipa

Na AddSize.aspx stranici dodaju se , mogu se izmjenjivati i brisati veličine za proizvode. Pomoći nje ću pokazati primjer LINQ upita koji koristi join-ove i kako izabrati rezultat kao novu klasu. Unutar projekta dodao sam novu klasu „BindBrands“ koja sadržava atribute koje želim projicirati u tablicu(ne trebaju mi svi atributi ovih relacija). Upit je sljedeći:

```
var upit = from A in context.tblSizes
join B in context.tblCategories on A.CategoryID equals B.CategoryID
join C in context.tblBrands on A.BrandID equals C.BrandID
join D in context.tblSubCategories on A.SubCategoryID equals D.SubCatID
join E in context.tblGenders on A.GenderID equals E.GenderID
           select new BindBrands
           {
            SizeID = A.SizeID,
```

```
    SizeName = A.SizeName,
    CategoryID = A.CategoryID,
    SubCategoryID = A.SubCategoryID,
    GenderID = A.GenderID,
    CategoryName = B.CategoryName,
    BrandName = C.BrandName,
    SubCatID = D.SubCatID,
    SubCatName = D.SubCatName,
    MainCatID = D.MainCatID,
    GenderName = E.GenderName
};
```

Join u LINQ-u se koristi na sličan način kao inner join u SQL-u. Dvije su razlike. Prva razlika je što se kod LINQ-a nakon ključne riječi „on“ koristiti atribut iz prve relacije i druga razlika je ta što se umjesto znaka „=“ koristi ključna riječ „equals“. Često se, osim nove klase , koriste i anonimni tipovi podataka za projekciju rezultata. U oba slučaja podatke možemo projicirati u tablicu , ali kada selektiramo objekt anonimnog tipa ne možemo direktno pristupiti njegovim svojstvima. Teoretski bi mogli, ali bi morali upotrijebiti refleksiju. Najjednostavnije objašnjenje je da ako objekt trebamo koristiti u daljnjem kontekstu stranice stvorimo novi objekt s pripadajućim atributima, a ako ne upotrijebiti ćemo objekt anonimnog tipa. Prema tome, projekciju iz prethodnog primjera mogli smo napisati na sljedeći način:

```
select new
{
 A.SizeID,
 A.SizeName,
 A.CategoryID,
 A.SubCategoryID,
 A.GenderID,
 B.CategoryName,
 C.BrandName,
 D.SubCatID,
 D.SubCatName,
 D.MainCatID,
 E.GenderName
};
```

4.3.3. Queryable<T> FirstOrDeafult() metoda i lambda izrazi

Popis sučelja u LINQ-u svodi se na sastavnicu Queryable. Queryable označava sve one funkcionalnosti koje provode upite nad izvorom podataka gdje su podaci poznati te ujedno omogućava i polimorfne upite. Važno je također reći kako klase koje koriste Queryable upite implementiraju funkcionalnosti sučelja Queryable<T>. Pozivanjem jedne od metoda unutar tog sučelja izvršava se upit npr. FirstOrDefault(). Ona vraća prvi element niza ili zadanu vrijednost ako niz ne sadrži elemente. Primjer LINQ upita koji vraća klasu s podacima tipa User koji vraća prvog korisnika s emailom povučenim iz url-a koij sam korisniku poslao na mail prilikom aktivacije korisničkog računa prikazan je sljedećim kodom .

```
string email = Request.QueryString["emailaddress"];
User user = (from u in context.Users where u.Email == email select
u).FirstOrDefault();
```

U kodu nadalje možemo koristiti varijablu user kako bismo pristupili svojstvima objekta User.

Ovaj upit mogli bismo napisati i pomoću lambda izraza. Lambda izrazi predstavljaju način na koji možemo definirati metodu izravno na mjestu na kojem je želimo i pozvati. Tu se obično radi o metodama koje su vrlo jednostavne ili ih ne planiramo pozivati nigdje osima na tom mjestu. Struktura lambda izraza je sljedeća : **parametriZaObradu =>kodKojiVršiObradu** [7]. U ovom slučaju lambda izraz koristit ćemo u svrhu pretraživanja liste.

```
User user = context.Users.FirstOrDefault(x => x.Email == email);
```

Rezultat je potpuno isti kao iz prethodnog upita.
Složeniji primjer LINQ upita koji koristi lambda izraze :

```
List<String> CookiePIDList = CookiePID.Split(',').Select(i =>
i.Trim()).Where(i => i != string.Empty).ToList();
```

Upit vraća listu stringova. Prvo se kolačić sa stranice podijeli na više elemenata nakon svakog zareza iz kojeg se onda uklanjaju svi vodeći i slijedeći znakovi s razmaka iz trenutnog niza objekata , provjerava se da string nije prazan i na kraju se sve to pretvara u listu.

4.3.4. Korištenje generiranih procedura u usporedbi s LINQ-om

U poglavlju 3.3. objasnio sam kako se postavlja Entity Framework i samim time generirao tablice i procedure. Sada ću objasniti kako se te procedure koriste u Entity Frameworku-u i kako se procedura može napisati pomoću LINQ-a. Kod procedure napisan u SQL-u:

```
CREATE PROCEDURE [dbo].[procBindAllProducts]
```

```
AS
SELECT A.*,B.*,C.BrandName,A.PPrice-A.PSelPrice as DiscAmount,B.PIMName as
ImageName from tblProducts A
inner join tblBrands C on C.BrandID=A.PBrandID
outer apply(
select top 1 * from tblProductImages B where  B.PID = A.ProductID order by
B.PID desc
      ) B
      order by A.ProductID desc
RETURN 0
```

Ova procedura dohvaća sve podatke iz relacija „tblProducts“, „tblProductImages“ i atribut „BrandName“ iz relacije „tblBrands“. Unutar nje koristi se inner join i outer apply. Outer apply služi slično kao i left join uz brže izvođenje upita. Unutar outer apply-a imamo select top 1 kako bismo dohvatili samo jednu sliku , odnosno bez da nam upit izbaci isti proizvod više puta s različitim slikama(naravno ako postoji više slika). Procedura se koristi kao metoda nad zadanim konteksom:

```
using (var context = new WebshopEntities())
{
   var upit1 = context.procBindAllProducts();
   rptrProducts.DataSource = upit1.ToList();
   rptrProducts.DataBind();
}
```

Druga linija ovog koda može koristiti lambda izraze nakon pozivanja procedure ako želimo npr. dobiti proizvode s određenom kategorijom:

```
var upit2 = context.procBindAllProducts().Where(x => x.PCategoryID ==
categoryID);
```

Bilo koja procedura može se napisati koristeći LINQ:

```
using (var context = new WebshopEntities())
 {
    var upit = from a in context.tblProducts
    from s in context.tblSizes
    from b in context.tblProductImages.Where(x => x.PID ==
a.ProductID).Take(1)
    where a.ProductID == PID1 && s.SizeID == SizeID1
    select new ItemsInCart
    {
    ProductID = a.ProductID,
```

```
    PName = a.PName,
    PSelPrice = a.PSelPrice,
    PPrice = a.PPrice,
    PIMName = b.PIMName,
    Extention = b.Extention,
    SIZEIDD = s.SizeID,
    SizeNamee = s.SizeName
    };
```

Cijeli outer apply dio koda napisan je sljedećom linijom koda:

```
from b in context.tblProductImages.Where(x => x.PID == a.ProductID).Take(1)
```

Koristimo također lambda izraz uz metodu Where i Take. Take(1) ima istu funkciju kao select top 1.

5. Zaključak

Cilj ovog rada bio je prikazati web aplikaciju koristeći drugačiji pristup bazi podataka- LINQ. Na početku su objašnjeni alati koji su se koristili u ovom radu i postavljanje logičkog modela kako bi se mogao koristiti LINQ.

Iako sam se već bavio web programiranjem(engl. Web developmentom) i LINQ-om na kolegijima ovog fakulteta, nikada prije nisam pokušao spojiti te dvije stvari. U web programiranju sam za programiranje na strani poslužitelja koristio PHP , a u C#-u sam do sada radio na windows formama. Pokušao sam spojiti te dvije stvari tako da nisam koristio spojeni način rada s bazom, već objektno relacijsko mapiranje (engl. Object-Relational Mapping) koristeći LINQ tehnologiju i C# kao jezik za programiranje na strani poslužitelja.

C# koristi okvir(engl. framework) ASP.NET za kreiranje web aplikacija na strani poslužitelja. U njemu je moguće razvijati web stranice , web aplikacije i web usluge. Nudi velik broj kontrola, poput tekstualnih okvira i gumba te konfiguriranje i manipuliranje koda radi stvaranja HTML stranica i različite predloške za korištenje kao što su MVC(engl. Model-View-Controller) , Web Forms, Web API i Single Page Application. Samim time omogućava lakši razvoj web aplikacije programeru.

LINQ je u mnogočemu revolucionarna tehnologija koja ima zaista nevjerojatan broj primjena i mogućnosti. Svojim funkcionalnostima on gotovo u potpunosti eliminira potrebu za aplikacijama koje služe za objektno-relacijsko mapiranje i tako uvelike ubrzava pristup podacima koje stvaraju aplikacije. Iako se govori o revolucionarnoj tehnologiji, LINQ je po svojoj prirodi i potrebama jednostavan za korištenje i u potpunosti se prilagođava potrebama i navikama programera koji ga koriste. Ovo je posebno vidljivo u samoj sintaksi LINQ-a koji se može pisati na dva načina – jednostavniji način, inspiriran SQL-om i onaj kompliciraniji, inspiriran sintaksom C#-a koji djeluje kao svojevrsni „host“ LINQ-u.

Spojivši ove dvije tehnologije uspio sam napraviti funkcionalnu web trgovinu. Uložio sam puno vremena i truda kako bih sve uspio implementirati, ali na kraju sam proširio svoje znanje web programiranja (engl. Web developmenta) i C#-a implementirajući upite na drugačiji način nego u PHP-u - koristeći LINQ.

6. Popis literature

[1] O. J. du Preez, Visual Studio 2019 In Depth: Discover and make use of the powerful features of the Visual Studio 2019 IDE to develop better and faster mobile, web , and desktop applications, 1. izd. New Delhi: BPB Publications, 2019.

[2] M. Mijač, I. Švogor, B. Tomaš, „Odabrana poglavlja programskog inženjerstva“, 2014. [Na internetu]. Dostupno na: https://www.scribd.com/doc/220657903/Programsko-inzenjerstvo#scribd . [Pristupljeno: 11-kol-2020].

[3] A. Hejlsberg, M. Torge, S. Wiltamuth, P. Golde, The C# Programming Language. Washington: Addison-Wesley Professional, 2008.

[4] S. Walther, ASP.NET Unleashed, 2. izd. Indianapolis: Sams Publishing, 2004.

[5] Tutorialspoint, „ASP.NET-Introduction“, 2020.[Na internetu]. Dostupno na: https://www.tutorialspoint.com/asp.net/asp.net_introduction.htm . [Pristupljeno: 12-kol-2020].

[6] Rakesh, „Basics of ADO.NET“, 2019. [Na internetu]. Dostupno na: https://www.c-sharpcorner.com/UploadFile/18fc30/understanding-the-basics-of-ado-net/ . [Pristupljeno 12-kol-2020].

[7] Kolegij Programsko inženjerstvo, „Osnove LINQ-a“ , https://elf.foi.hr/pluginfile.php/63684/mod_resource/content/3/Osnove%20rada%20sa%20Entity%20Framework-om.pdf . [Pristupljeno 12-kol-2020] .

[8] M. Torgsen, „Querying in C#: how language integrated query (LINQ) works“, 2007.[Na internetu]. Dostupno na: https://dl.acm.org/doi/abs/10.1145/1297846.1297922 . [Pristupljeno 11-kol-2020].

[9] TutorialsTeacher , „What is LINQ?“ (bez dat.) [Na internetu]. Dostupno na: https://www.tutorialsteacher.com/linq/what-is-linq . [Pristupljeno 12-kol-2020].

[10] Alexandre Ouellette, „What is Bootstrap: A Beginners Guide“ , 2017.[Na internetu]. Dostupno na: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/ . [Pristupljeno 11-kol-2020] .

[11] Bootstrap, „Navbar“ , 2020. [Na internetu]. Dostupno na: https://getbootstrap.com/docs/4.0/components/navbar/ . [Pristupljeno 10-srp-2020].

[12] Microsoft, „Visual Studio 2019“, 2020. [Na internetu]. Dostupno na: https://visualstudio.microsoft.com/vs/ . [Pristupljeno 9-kol-2020].

7. Popis slika

Slika 1: Visual Studio IDE [12] 2

Slika 2: ERA model (slika autora) 6

Slika 3: Navigacijski model- administrator 7

Slika 4: Navigacijski model- registrirani korisnik 8

Slika 5: Odabir ADO.NET Entity Data Modela 9

Slika 6: Postavljanje veze na bazu 10

Slika 7: ADO.NET Entity Data Model.....10

Slika 8: Navigacija kod uloge administrator.....13

Slika 9: Izgled stranice AddCategory.aspx.....13

Slika 10: Prikaz stranice ProductView.aspx 14