UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

ZAKLJUČNA NALOGA

RAZVOJ POSLOVNIH APLIKACIJ V OKOLJU MICROSOFT PRISM 4

MATJAŽ ŠUBER

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Razvoj poslovnih aplikacij v okolju Microsoft PRISM 4

(Development of business applications with Microsoft PRISM 4)

Ime in priimek: Matjaž Šuber
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Peter Rogelj
Somentor: doc. dr. Iztok Savnik

Koper, avgust 2013

Ključna dokumentacijska informacija

Ime in PRIIMEK: Matjaž ŠUBER
Naslov zaključne naloge: Razvoj poslovnih aplikacij v okolju Microsoft PRISM 4

Kraj: Koper
Leto: 2013
Število listov: 69 Število slik: 26 Število tabel: 1
Število prilog: 1 Št. strani prilog: 1
Število referenc: 9
Mentor: doc. dr. Peter Rogelj
Somentor: doc. dr. Iztok Savnik
UDK:
Ključne besede: aplikacija, modul, spletni servis
Izvleček:
Zaključna projektna naloga opisuje način/pristop razvoja poslovnih aplikacij v okolju Microsoft PRISM. Microsoft PRISM določa navodila, ki razvijalcem pomagajo razviti bogate, fleksibilne in enostavno spremenljive »WPF – Windows Presentation Foundation« namizne aplikacije ter »RIA – Rich Internet Applications« aplikacije, ki so razvite v aplikacijskem ogrodju Microsoft Silverlight.
Kot primer poslovne aplikacije zaključna naloga opisuje razvoj trgovine. Slednja se imenuje STrgovina, ter vključuje osnovne funkcionalnosti, ki jih trgovci in poslovodja pri svojem delu potrebujejo. Razvoj aplikacije STrgovina je opisan v prvih šestih fazah linearnega programskega procesa. Med slednje sodijo faza definicije problema, faza študije izvedljivosti, faza analize in definiranja zahtev, faza načrtovanja sistema, faza načrtovanja programa ter faza izvedbe. Linearni programski proces poleg zgoraj omenjenih faz vključuje še fazo testiranja, fazo predaje in fazo vzdrževanja programskega produkta. Ker gre za akademski projekt so slednje faze izpuščene. Znotraj posameznih faz so opisani potrebni koraki in navodila, ki omogočajo razvoj poslovnih aplikacij, kot to določa okolje Microsoft PRISM. Uporabniški vmesnik poslovne aplikacije STrgovina je razvit v aplikacijskem ogrodju Microsoft Silverlight. Za shranjevanje podatkov je uporabljena podatkovna baza MS SQL »Microsoft SQL«. Vsa potrebna razvojna orodja so za namene zaključne projektne naloge brezplačna ter pridobljena preko študentskega računa na spletni strani Microsoft DreamSpark for Academic Institutions [5].

Key words documentation

Name and SURNAME: Matjaž ŠUBER
Title of final project paper: Development of business applications with Microsoft PRISM 4

Place: Koper
Year: 2013
Number of pages: 69 Number of figures: 26 Number of tables: 1
Number of appendix: 1 Number of appendix pages: 1
Number of references: 9
Mentor: doc. dr. Peter Rogelj
Co-Mentor: doc. dr. Iztok Savnik
UDC:
Keywords: application, module, web service
Abstract:

Final project paper describes a method/approach of business application development with Microsoft PRISM. Microsoft PRISM provides guidance to help developers design and build rich, flexible and easy-to-mantain "WPF - Windows Presentation Foundation" desktop applications and "RIA - Rich Internet Applications" applications that are developed in the Microsoft Application Framework known as Silverlight.

As an example of business application this paper describes development of a shop. The latter is called STrgovina and includes basic functionality that merchants and manager needs. The development of STrgovina is described in first six stages of the linear software process. Among the latter are problem definition, feasibility study, analysis and requirements definition, system design, program design and implementation. The linear software process also include testing phase, transfer phase and maintenance phase of the software product. Since this is an academic project those three phases were skipped. Within each phase necessary steps and instructions that enable the development of business applications in Microsoft PRISM environment are described. The application user interface was developed in the Microsoft Application Framework known as Silverlight. For the data storage was used MS SQL »Microsoft SQL« database. All development tools used in the project were freely obtained through the student account on the website Microsoft DreamSpark for Academic Institutions [5].

Zahvala

Zahvaljujem se mentorju doc. dr. Petru Roglju in somentorju doc. dr. Iztoku Savniku za strokovno pomoč in spodbudo pri izdelavi zaključne projektne naloge.

Zahvalil bi se staršema, mami Bogdani in očetu Andreju, ker sta mi omogočila študij, me podpirala na moji študijski poti in mi zmeraj stala ob strani.

Kazalo vsebine

1 UVOD ... 1
2 DEFINICIJA PROBLEMA ... 2
3 ŠTUDIJA IZVEDLJIVOSTI ... 3
4 ANALIZA IN DEFINIRANJE ZAHTEV ... 5
4.1 Namen sistema ... 5
4.2 Funkcionalne zahteve ... 5
4.3 Naravne zahteve po podatkih ... 6
4.4 Primeri uporabe ... 7
4.4.1 Primer uporabe 1 ... 7
4.4.2 Primer uporabe 2 ... 8
4.4.3 Primer uporabe 3 ... 9
4.4.4 Primer uporabe 4 ... 10
4.4.5 Primer uporabe 5 ... 11
4.5 Nefunkcijske zahteve ... 12
4.5.1 Varnost ... 12
4.5.3 Razširljivost ... 12
5 NAČRTOVANJE SISTEMA ... 15
5.1 Komponente in vmesniki ... 15
5.1.1 Spletni uporabniški vmesnik – Prism.STrgovina ... 17
5.1.1.1 Moduli – Prism.STrgovina.Modules ... 17
5.1.2 Spletni strežnik – Prism.STrgovina.Web ... 18
5.1.3 Spletni strežnik – Prism.STrgovina.WS ... 19
6 NAČRTOVANJE PROGRAMA ... 21
6.1 Spletni uporabniški vmesnik – Prism.Strgovina ... 21
6.1.1 Razred Bootstrapper ... 22
6.1.2 »Model – View – ViewModel« načrtovalski vzorec ... 23
6.2 Moduli – Prism.STrgovina.Moduli ... 24
6.2.1 Microsoft PRISM komponente, ki bodo vsebovane v aplikaciji STrgovina ... 25
6.2.1.1 Region Manager ... 25
6.2.1.2 Module Manager ... 25
6.2.1.3 Event Aggregator ... 25
6.2.2 Modul »Prijava« - Prism.STrgovina.Moduli.Prijava ... 26
6.2.2.1 Opis glavnih aktivnosti modula »Prijava« ... 28
6.2.2.1.1 Aktivnost prijave uporabnika ... 28

6.2.2.1.2 Aktivnost pridobivanja osnovnih informacij o trenutnem uporabniku. 29
6.2.3 Modul »Trgovec« - Prism.STrgovina.Moduli.Trgovec 30
6.2.3.1 Opis glavnih aktivnosti modula »Trgovec« 32
6.2.3.1.1 Aktivnost dodajanja novega artikla 33
6.2.3.1.2 Aktivnost pregledovanja in iskanja strank 33
6.2.3.1.3 Aktivnost dodajanja novega nakupa in tiskanja računa 34
6.2.4 Modul »Poslovodja« - Prism.STrgovina.Moduli.Poslovodja 35
6.2.4.1 Opis glavnih aktivnosti modula »Poslovodja« 37
6.2.4.1.1 Aktivnost brisanja obstoječega trgovca 37
6.2.4.1.2 Aktivnost pregledovanja statističnih podatkov o nakupih za stranke ... 38
6.3 Spletni strežnik IIS – Prism.STrgovina.Web 39
6.4 WCF Spletni servis – WCFService 41
6.4.1 WCF spletne storitve 41
6.4.2 Uporaba WCF spletnih storitev na primeru storitve VrniArtikelID 42
6.4.3 Poslovni razredi, imenovani Upravitelji 43
6.5 MS SQL podatkovna baza – Prism.STrgovina.DB 44
7 IZVEDBA 45
7.1 Spletni uporabniški vmesnik – Prism.STrgovina 45
7.1.1 Razred Bootstrapper 46
7.1.2 »Model – View – ViewModel« načrtovalski vzorec 47
7.2 Moduli 48
7.2.1 Inicializacija modulov 48
7.2.2 Proces nalaganja modulov s pomočjo komponente ModuleManager 49
7.2.3 Registracija in nalaganje uporabniških pogledov v regije 49
7.2.4 Uporaba komponente »EventAggregator« 50
7.2.5 Uporaba komponente »ServiceReference« 51
7.3 WCF spletni servis »ServiceLink« 52
7.4 WCF spletni servis »WCFService« 52
7.4.1 Poslovni objekti imenovani »Upravitelji« 53
8 ZAKLJUČEK 55
Literatura 56
Viri 57

Kazalo preglednic

Tabela 1: WCF spletne storitve vsebovane v razredu WCFService 42

Kazalo slik in grafikonov

Slika 1: UML diagram primera uporabe za dodajanje, urejanje in brisanje artiklov 7
Slika 2: UML diagram primera uporabe za dodajanje, urejanje in brisanje strank 8
Slika 3: UML diagram primera uporabe za prodajo 9
Slika 4: UML diagram primera uporabe za dodajanje, urejanje in brisanje trgovcev 10
Slika 5: UML diagram primera uporabe za poslovanje 11
Slika 6: UML postavitveni diagram 15
Slika 7: Moduli poslovne aplikacije STrgovina 18
Slika 8: WCF spletni service - WCFService 19
Slika 9: Spletni uporabniški vmesnik – lupina 22
Slika 10: Bootstrapper 22
Slika 11: MVVM razredi in medsebojna odvisnost 23
Slika 12: Prikaz mehanizma, ki ga implementira storitev EventAggregator 25
Slika 13: UML Sekvenčni diagram inicializacije modula »Prijava« 26
Slika 14: UML diagram aktivnosti za proces prijave uporabnikov v modulu »Prijava« 28
Slika 15: UML diagram aktivnosti pridobivanja osnovnih informacij o uporabniku 29
Slika 16: Sekvenčni diagram inicializacije modula »Trgovec« 30
Slika 17: UML diagram aktivnosti za proces dodajana novega artikla 33
Slika 18: UML diagram aktivnosti za proces pregledovanja in iskanja strank 34
Slika 19: UML diagram aktivnosti za proces dodajanja novega nakupa in tiskanja računa 34
Slika 20: Sekvenčni diagram inicializacije modula »Poslovodja« 35
Slika 21: UML diagram aktivnosti za proces brisanja obstoječega trgovca 37
Slika 22: UML diagram aktivnosti pregledovanja statističnih podatkov o nakupih 38
Slika 23: UML komponentni diagram poslovne aplikacije STrgovina 40
Slika 24: Koraki pri poganjanju Silverlight spletnih aplikacij v fazi razvoja 41
Slika 25: UML razredni diagram poslovnih objektom imenovanih Upravitelji 43
Slika 26: UML razredni diagram podatkovne baze Prism.STrgovina.DB 44

Seznam kratic

Seznam kratic definira pomen kratic, ki so uporabljene v zaključni nalogi.

- ***PRISM:*** je kodno ime za razvojna navodila poznana pod izrazom »Composite Application Guidance for WPF and Silverlight«.
- ***WPF:*** »Windows Presentation Foundation« je programska oprema za upodabljanje grafičnih uporabniških vmesnikov v Windows usmerjenih aplikacijah.
- ***WCF:*** »Windows Communication Foundation« je aplikacijsko ogrodje za razvoj storitveno usmerjenih aplikacij.
- ***RIA:*** »Rich Internet application« predstavlja spletno aplikacijo, ki premore vse lastnosti namiznih aplikacij.
- ***MVVM:*** »Model – View - ViewModel« je načrtovalski vzorec, ki loči poslovno in predstavitveno logiko aplikacije od grafičnega uporabniškega vmesnika.
- ***IIS:*** »Internet Information Services«je spletni strežnik.
- ***ORM:*** »Object-Relational mapping« je programerska tehnika, ki je namenjena pretvorbi podatkov med nezdružljivimi podatkovnimi tipi objektno orientiranega sveta.
- ***MS SQL:*** »Microsoft SQL Server« je sistem za upravljanje z relacijskimi podatkovnimi bazami.
- ***XAML:*** »Extensible Application Markup Language« je deklarativni XML jezik, ki omogoča inicializacijo strukturiranih vrednosti in objektov.
- ***STrgovina:*** predstavlja naziv/ime poslovne aplikacije, ki je opisana v pričujoči zaključni nalogi.
- ***Prism.STrgovina:*** je imensko področje, ki določa osnovno lupino aplikacije STrgovina.
- ***Prism.STrgovina.Modules.Prijava:*** je imensko področje, ki določa modul »Prijava«.
- ***Prism.STrgovina.Modules.Trgovec:*** je imensko področje, ki določa modul »Trgovec«.
- ***Prism.STrgovina.Modules.Poslovodja:*** je imensko področje, ki določa modul »Poslovodja«.
- ***Prism.STrgovina.Web:*** je imensko področje, ki določa spletni strežnik, ki je namenjen gostovanju poslovne aplikacije STrgovina.
- ***Prism.STrgovina.WS:*** je imensko področje, ki določa spletni strežnik, ki vsebuje WCF spletni servis in MS SQL relacijsko podatkovno bazo.
- ***Prism.STrgovina.DB:*** predstavlja naziv MS SQL relacijske podatkovne baze.

Slovarček besed

Slovarček besed opisuje glavne pojme, ki so uporabljeni v zaključni nalogi.

- ***Aplikacija:*** predstavlja računalniški program, ki uporabnikom omogoča opravljanje določenih nalog.
- ***Sestavljena aplikacija:*** je aplikacija, sestavljena iz določenega števila neodvisnih komponent imenovanih moduli.
- ***Bootstraper:*** je razred, ki je namenjen inicializaciji aplikacije razvite v okolju Microsoft PRISM.
- ***Modularnost:*** pomeni zmožnost načrtovanja in izvedbe zahtevnih aplikacij s pomočjo neodvisnih funkcionalnih enot imenovanih moduli.
- ***Modul:*** je zaokrožen paket funkcionalnosti, ki je lahko neodvisno načrtovan, implementiran in testiran.
- ***Katalog modulov:*** predstavlja seznam, ki določa lokacijo modulov aplikacije, ter njihove inicializacijske lastnosti.
- ***Razred ModuleManager:*** je osnovni razred, ki omogoča nalaganje modulov v aplikacijo.
- ***Regije ali področja:*** predstavljajo logični prostor, v katerega se lahko nalagajo Silverlight uporabniški vmesniki ali pogledi
- ***Razred RegionManager:*** je osnovni razred, ki omogoča registracijo in nalaganje uporabniških pogledov posameznega modula v regije, ki so definirane na lupini aplikacije.
- ***Lupina aplikacije »Shell«:*** predstavlja osnovni uporabniški vmesnik aplikacije. Določa postavitev regij, v katere se nalagajo uporabniški pogledi posameznih modulov.
- ***Razred EventAggregator:*** je osnovni razred, ki omogoča komunikacijo med moduli aplikacije.
- ***Spletni servis:*** je programska funkcija, ki je dostopna na določenem spletnem naslovu ter ponuja spletne storitve.
- ***Spletna storitev:*** predstavlja metodo, ki omogoča opravljanje določene funkcionalnosti.

1 UVOD

Razvoj poslovnih aplikacij je v zadnjih letih zelo pomembna dejavnost/aktivnost, kajti v današnjem času podjetja brez uporabe informacijske in komunikacijske tehnologije zelo težko konkurirajo ostalim podjetjem v svetu. Poslovne aplikacije omogočajo uporabnikom delo s poslovno inteligentnimi podatki, ter nudijo podporo pri poslovanju podjetja. Vsako podjetje deluje na določenem poslovnem področju, zato se mora poslovna aplikacija, ki jo podjetje uporablja implementirati tako, da bo zadovoljila vse poslovne potrebe uporabnikov. Pogosto so poslovne aplikacije prilagojene posameznemu podjetju.

Pri razvoju poslovnih aplikacij se razvijalci držijo ustaljenih tirnic, ki določajo programski proces, torej proces gradnje programskega produkta od zaznane potrebe po programskem produktu, do konca vzdrževanja. Programski proces sestoji iz aktivnosti/korakov, ki jih je potrebno izvesti v ustreznem vrstnem redu.

V nalogi, je opisan način/pristop razvoja poslovne aplikacije STrgovina v okolju Microsoft PRISM. Microsoft PRISM ponuja navodila, ki razvijalcem pomagajo razvijati bogate, fleksibilne in enostavno spremenljive »WPF - Windows Presentation Foundation« namizne aplikacije, »Rich Internet Applications – RIAs« aplikacije, ki so razvite skupaj z »Microsoft Silverlight« spletnim klientom ter aplikacije, ki tečejo na mobilni platformi »Windows Phone 7« [2]. PRISM je kodno ime za navodila, ki jih formalno poznamo pod imenom »Application Guidance for WPF and Silverlight«. Zaradi enostavnosti in konsistentnosti so se ta navodila kasneje preimenovala v PRISM. Namenjen je razvijalcem programske opreme, ki razvijajo WPF ali Silverlight aplikacije, ki tipično vključujejo več pogledov, bogato uporabniško izkušnjo in podatkovno vizualizacijo. Te aplikacije tipično komunicirajo z raznimi spletnimi servisi, prek večplastne arhitekture. Pri razvoju poslovnih aplikacij v okolju Microsoft PRISM velikokrat pričakujemo da se bo aplikacija konstantno večala zaradi novih poslovnih zahtev. PRISM vključuje programsko platformo Microsoft .NET verzija 4, aplikacijsko platformo Silverlight 4 ter arhitekturni vzorec MVVM – Model, View, View-Model (Model, Pogled, Pogled-Model).

Cilj te naloge je predstaviti okolje Microsoft PRISM 4, ter pripadajoče tehnologije, ki pripomorejo k razvoju bogatih, fleksibilnih, enostavno spremenljivih poslovnih aplikacij.

Naloga je sestavljena iz sedmih delov. Vsak izmed njih predstavi pripadajočo fazo programskega procesa ter navodila/priporočila okolja Microsoft PRISM.

2 DEFINICIJA PROBLEMA

Problem razvoja poslovnih aplikacij, je v njihovi specifičnosti. Le-te so razvite za točno določene uporabnike, ter premorejo funkcionalnosti, ki omogočajo hitrejše ter enostavnejše poslovanje. Prav zaradi tega je pri njihovem razvoju potrebno vložiti veliko truda v začetne faze programskega procesa, predvsem v fazo analize zahtev, kjer se definirajo funkcionalnosti, ki jih bo končni produkt imel.

Poslovna aplikacija STrgovina je namenjena trgovcem, ki želijo imeti popolno kontrolo nad svojim poslovanjem. Vsebovati mora veliko koristnih funkcionalnosti med katere spadajo upravljanje z artikli, upravljanje s strankami in trgovci ter statistični pregled nad poslovanjem podjetja. Kot vsaka trgovina mora vsebovati blagajno, ki je namenjena izdelavi in storniranju nakupov ter tiskanju računov. Uporabniki aplikacije STrgovina bodo trgovci in poslovodja. Trgovci želijo imeti popolni nadzor nad poslovno pomembnimi podatki, torej želijo dodajati, urejati in brisati artikle ter stranke. Poleg tega trgovci želijo imeti pravico do upravljanja z blagajno, ki bo omogočala ustvarjanje novih nakupov, storniranje obstoječih ter tiskanje računov. Druga vrsta uporabnikov pa je poslovodja. Poslovodja želi imeti popolni nadzor nad trgovci, torej želi dodajati, urejati in brisati trgovce. Poleg tega potrebuje statistični pregled nad poslovanjem podjetja. Sistem mora biti zasnovan tako, da bo poslovodji omogočal pregled vsote nakupov za posamezno stranko v izbranem časovnem obdobju.

Cilj projekta je razviti poslovno aplikacijo, ki bo izpolnjevala zgoraj navedene funkcionalnosti. Razvita bo kot spletna aplikacija, saj bo tako dostopna preko svetovnega spleta z uporabno ustreznih spletnih brskalnikov. Zaradi različnih uporabniških pravic bo omogočala dva pogleda in sicer pogled trgovca in pogled poslovodje.

Tema zaključne naloge je razvoj poslovnih aplikacij v okolju Microsoft PRISM. Ker gre za licenčno programsko opremo, je potrebno omeniti, da bi bili stroški projekta sorazmerni s ceno licenc pridobljenih za uporabo razvojnih orodij, ki omogočajo razvoj v okolju Microsoft PRISM. V našem primeru bodo stroški projekta ničelni kajti gre za akademski projekt.

3 ŠTUDIJA IZVEDLJIVOSTI

Pri študiji izvedljivosti projekta STrgovina imamo dve možni oviri. Prva ovira se nanaša na stroške projekta. Ker je tema zaključne naloge »razvoj poslovnih aplikacij v okolju Microsoft PRISM« je potrebno omeniti, da bom pri razvoju uporabljal licenčno programsko opremo. Licenčna programska oprema ni odprta koda. Za uporabo tehnologij, ki so zaščitene z lastniško licenco je potrebno le-to najprej kupiti. V večini primerov lastniška licenca dovoljuje namestitev programske opreme le na določeno število računalnikov in ne dovoli vpogleda v izvorno kodo. Ker gre za akademski projekt sem vso potrebno programsko opremo pridobil brezplačno preko študentskega računa na spletni strani Microsoft DreamSpark for Academic Institutions [5].

Druga ovira je zaključni rok izdelave projekta. Projekt STrgovina mora biti zaključen do konca avgusta 2013, zato smo časovno omejeni na devet mescev oziroma petinštirideset tednov. Pri razvoju aplikacije je potrebno upoštevati da gre za večji projekt, zato bom aplikacijo predvidoma razdelil na štiri komponente. Komponente aplikacije STrgovina bodo podatkovna baza, spletni servis, spletni strežnik in klient. Podatkovna baza je namenjena shranjevanju poslovno pomembnih podatkov, spletnih servis omogoča dostop in upravljanje s podatkovno bazo, spletni strežnik gosti spletnega klienta, spletni klient pa je namenjen končnim uporabnikom, da lahko grafično upravljajo z aplikacijo. Poleg zgoraj omenjenih komponent je potrebno upoštevati še čas izdelave dokumentacije oziroma zaključne naloge. Slednjo bomo pisali skozi celoten proces razvoja, zato je predviden čas za izdelavo posamezne komponente deset tednov. Preostalih pet tednov bo namenjenih recenzijam zaključne naloge, ter morebitnim popravkom poslovne aplikacije STrgovina.

Pri razvoju poslovnih aplikacij običajno sodeluje skupina razvojnih programerjev, sistemskih analitikov ter končnih uporabnikov. Ker gre za akademski projekt, bom sam prevzel vse zgoraj naštete vloge. Prav zaradi tega bodo stroški osebja ničelni.

Za izvedbo aplikacije STrgovina načrtujemo uporabo naslednjega programja. Podatkovna baza bo MS SQL »Microsoft SQL Server database«. MS SQL je relacijska podatkovna baza, ki je namenjena shranjevanju podatkov spletnih aplikacij. Spletni servis bo za upravljanje z bazo uporabljal programski produkt imenovan »ADO.NET Entity Framework«. »ADO.NET Entity Framework« je del ORM »Object-relational mapping« družine. ORM v računalniškem svetu opisuje programersko tehniko, ki je namenjena pretvorbi podatkov med nezdružljivimi podatkovnimi tipi objektno orientiranega sveta. Ta v osnovi kreira virtualne objekte nad podatkovno bazo, ki jih lahko uporabimo v programski kodi. Na ta način se odpravi potreba po SQL poizvedbah. Spletni strežnik bo

IIS »Internet Information Services«, ker omogoča gostovanje Microsoft Silverlight spletnih aplikacij. Spletni klient bo razvit v okolju Microsoft PRISM. Le ta v povezavi s tehnologijo Microsoft Silverlight in arhitekturnim vzorcem MVVM »Model - View - ViewModel« omogoča implementacijo bogatih, fleksibilnih, enostavno spremenljivih uporabniški vmesnikov.

4 ANALIZA IN DEFINIRANJE ZAHTEV

Analiza in definiranje zahtev ali sistemska analiza predstavlja tretjo fazo linearnega programskega procesa. Izvaja jo sistemski analitik, ki na osnovi informacij o realnem problemu izlušči jasne in eksplicitne zahteve. Pri tem mu pomagajo naročnik in končni uporabniki. V našem primeru gre za akademski projekt, zato bom sam prevzel vlogo sistemskega analitika, končnega uporabnika in naročnika. Izziv analize zahtev je definirati tak sistem, ki ustreza uporabnikovim potrebam in pričakovanjem. Rezultat sistemske analize je dokument, imenovan specifikacija zahtev, ki vsebuje popoln opis obnašanja oziroma delovanja sistema. S pomočjo specifikacije zahtev želimo uskladiti razumevanje problema in rešitve med sistemskim analitikom, ter končnimi uporabniki, se seznaniti s potrebami in z omejitvami uporabnikov sistema ter določiti izhodišča za fazi načrtovanja in validacije programskega produkta.

4.1 Namen sistema

Poslovna aplikacija STrgovina je namenjena trgovcem, ki želijo imeti popolno kontrolo nad svojim poslovanjem. Vsebuje veliko koristnih funkcionalnosti med katere spada upravljanje z artikli, upravljanje s strankami in trgovci ter statistični pregled nad poslovanjem podjetja. Kot vsaka trgovina vsebuje blagajno, ki je namenjena izdelavi in storniranju nakupov ter tiskanju računov. Ker gre za akademski projekt, dana poslovna aplikacija ne vsebuje plačilnega sistema.

4.2 Funkcionalne zahteve

Končni uporabniki poslovne aplikacije STrgovina so trgovci in poslovodja. Funkcionalne zahteve so zato ločene na dva sklopa, in sicer sklop trgovec in sklop poslovodja.

Trgovec:

- dodajanje, urejanje, brisanje artiklov;
- dodajanje, urejanje, brisanje strank,;
- prodaja (dodajanje artiklov na seznam, možnost vpisa identifikacijske številke stranke, možnost tiskanja računov, možnost storniranja nakupov).

Poslovodja:

- dodajanje, urejanje, brisanje trgovcev;
- pregled nakupov za posamezno stranko v izbranem časovnem obdobju;
- pregled vsote nakupov za posamezno stranko v izbranem časovnem obdobju.

4.3 Naravne zahteve po podatkih

Artikel:
Poslovna aplikacija STrgovina bo trgovcem omogočala upravljanje z artikli. Artikel bo entiteta, ki bo vsebovala naziv artikla, opis artikla, zalogo artikla, davčno stopnjo, davek, ceno in ceno brez davka.

Stranka:
Poslovna aplikacija STrgovina bo trgovcem omogočala upravljanje s strankami. Stranka bo entiteta, ki bo vsebovala ime in priimek stranke, njen naslov, telefon in e-pošto.

Trgovec:
Poslovna aplikacija STrgovina bo poslovodji omogočala upravljanje s trgovci. Trgovec bo entiteta, ki bo vsebovala ime in priimek trgovca, njegov naslov, telefon, e-pošto, geslo in status. Status je atribut, ki bo določal vlogo uporabnika aplikacije STrgovina. Med vloge sodita vloga trgovca in vloga poslovodje.

Nakup:
Poslovna aplikacija STrgovina bo trgovcem omogočala delo z blagajno. Blagajna bo namenjena ustvarjanju in storniranju nakupov, zato potrebujemo entiteto, ki bo hranila podatke o nakupih. Entiteta Nakup bo vsebovala datum in status nakupa. Status nakupa bo podatek, ki bo nosil informacijo o tem, ali je dani nakup storniran ali ne. Poleg tega bo morala entiteta Nakup vsebovati še podatek o trgovcu in stranki, ki sta pri nakupu sodelovala.

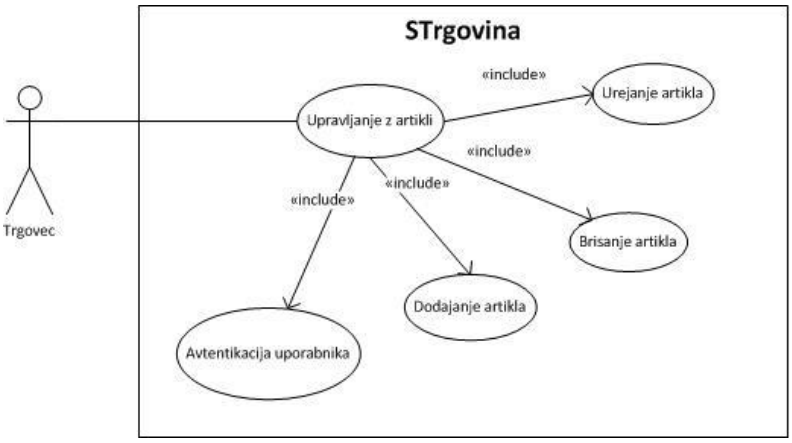
Nakup_Artikel:
Pri povezovanju nakupa z artikli, bomo potrebovali dodatno entiteto, ki bo predstavljala relacijo med entiteto Nakup in entiteto Artikel. Slednjo bomo poimenovali Nakup_Artikel. Vsebovati bo morala enolični identifikator artikla in nakupa ter nekatere podatke o artiklih, ki so vsebovani v dotičnem nakupu. Podatke o artiklih bomo v entiteto Nakup_Artikel prepisovali, kajti le tako bomo omogočili konsistentno in verodostojno pregledovanje zgodovine nakupov. Med podatke o artiklih bomo vključili podatek o nazivu artikla, podatek o količini, podatek o davčni stopnji, podatek o davku ter ceni artikla.

4.4 Primeri uporabe

4.4.1 Primer uporabe 1

Ime: Dodajanje, urejanje in brisanje artiklov

Udeleženci: Trgovec



Slika 1: UML diagram primera uporabe za dodajanje, urejanje in brisanje artiklov

Opis:

Trgovec se mora pred začetkom uporabe poslovne aplikacije STrgovina avtenticirati. V polje »Uporabniško ime:« vpiše dodeljeno uporabniško ime, v polje »Geslo:« pa svoje geslo. Po uspešno opravljeni prijavi se trgovcu prikaže začetni uporabniški vmesnik, kjer trgovec izbere možnost »Artikli«. V osnovni delovni površini se prikaže obrazec za delo z artikli. Tukaj lahko trgovec dodaja, ureja ali briše artikle.

Dodajanje artikla:

Trgovec na obrazcu za delo z artikli klikne na gumb »Dodaj artikel«. Odpre se modalno okno, ki od trgovca zahteva vpis naslednjih podatkov o artiklu: ime/naziv, opis, zalogo, ceno brez davka ter podatek o davčni stopnji. Po uspešno vnesenih podatkih trgovec klikne na gumb »Dodaj«. Sistem doda artikel v podatkovno bazo.

Urejanje artikla:

Trgovec na obrazcu za delo z artikli v iskalno polje vpiše iskani naziv artikla. Če sistem najde iskani artikel, le tega prikaže v tabeli. Trgovec nato označi želeni zadetek ter klikne na gumb »Odpri«. Odpre se pojavno okno, kjer trgovec vpiše nove podatke, ter klikne na gumb »Uredi«. Sistem izbrani artikel posodobi v podatkovni bazi.

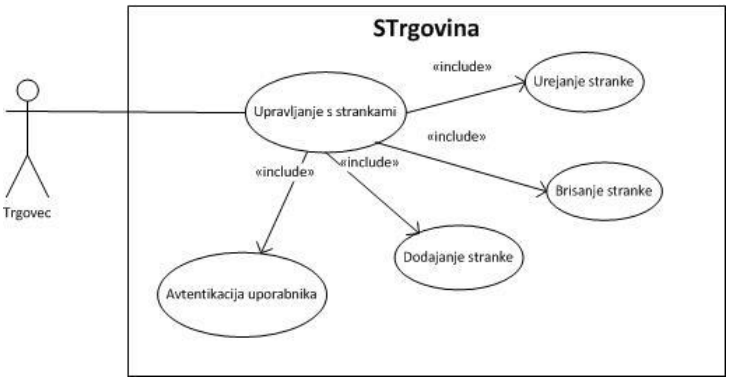
Brisanje artikla:

Trgovec na obrazcu za delo z artikli v iskalno polje vpiše iskani naziv artikla. Če sistem najde iskani artikel, le tega prikaže v tabeli. Trgovec nato označi želeni zadetek ter klikne na gumb »Odpri«. Odpre se pojavno okno, kjer trgovec klikne na gumb »Izbriši«. Sistem izbriše izbrani artikel iz podatkovne baze.

4.4.2 Primer uporabe 2

Ime: Dodajanje, urejanje in brisanje strank

Udeleženci: Trgovec



Slika 2: UML diagram primera uporabe za dodajanje, urejanje in brisanje strank

Opis:

Trgovec se mora pred začetkom uporabe poslovne aplikacije STrgovina avtenticirati. V polje »Uporabniško ime:« vpiše dodeljeno uporabniško ime, v polje »Geslo:« pa svoje geslo. Po uspešno opravljeni prijavi se trgovcu prikaže začetni uporabniški vmesnik, kjer trgovec izbere možnost »Stranke«. V osnovni delovni površini se prikaže obrazec za delo s strankami. Tukaj lahko trgovec dodaja, ureja ali briše stranke.

Dodajanje stranke:

Trgovec na obrazcu za delo s strankami klikne na gumb »Dodaj stranko«. Odpre se modalno okno, ki od trgovca zahteva vpis naslednjih podatkov o stranki: ime, priimek, naslov, telefon in e-poštni naslov. Po uspešno vnesenih podatkih trgovec klikne na gumb »Dodaj«. Sistem doda stranko v podatkovno bazo.

Urejanje stranke:

Trgovec na obrazcu za delo s strankami v iskalno polje vpiše ime stranke. Če sistem najde iskano stranko, le-to prikaže v tabeli. Trgovec nato označi želeni zadetek ter klikne na gumb »Odpri«. Odpre se pojavno okno, kjer trgovec vpiše nove podatke, ter klikne na gumb »Uredi«. Sistem izbrano stranko posodobi v podatkovni bazi.

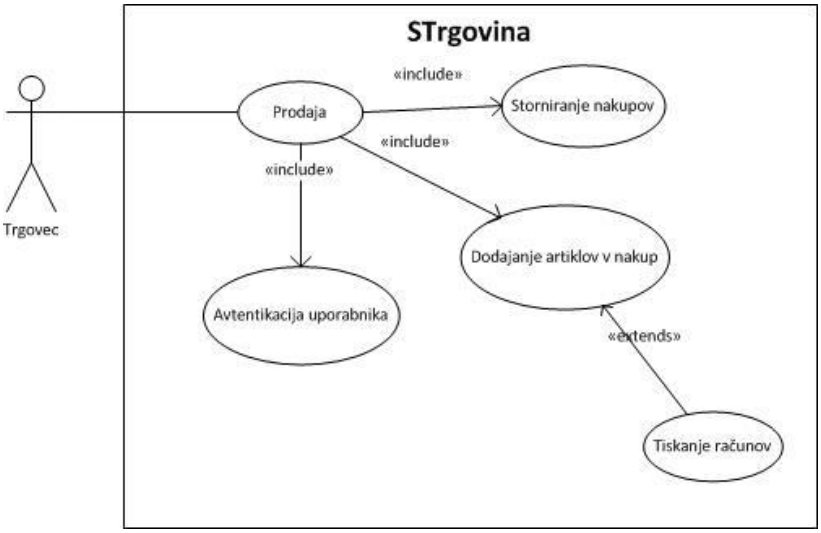
Brisanje stranke:

Trgovec na obrazcu za delo s strankami v iskalno polje vpiše ime stranke. Če sistem najde iskano stranko, le-to prikaže v tabeli. Trgovec nato označi želeni zadetek ter klikne na gumb »Odpri«. Odpre se pojavno okno, kjer trgovec klikne na gumb »Izbriši«. Sistem izbriše izbrano stranko iz podatkovne baze.

4.4.3 Primer uporabe 3

Ime: Prodaja

Udeleženci: Trgovec, stranka



Slika 3: UML diagram primera uporabe za prodajo

Opis:

Trgovec se mora pred začetkom uporabe poslovne aplikacije STrgovina avtenticirati. V polje »Uporabniško ime:« vpiše dodeljeno uporabniško ime, v polje »Geslo:« pa svoje geslo. Po uspešno opravljeni prijavi se trgovcu prikaže začetni uporabniški vmesnik, kjer trgovec izbere možnost »Blagajna«. V osnovni delovni površini se prikaže blagajna.

Dodajanje artiklov v nakup:

Če želi trgovec ustvariti nov nakup, klikne na gumb »Ustvari nakup«. Sedaj se trgovcu omogoči dodajanje artiklov v nakup. Trgovec izbere identifikator artikla, vpiše količino ter klikne na gumb »Dodaj v nakup«. Ko trgovec konča z dodajanjem artiklov v nakup, v polje »pika kartica« vpiše identifikacijsko številko stranke, v kolikor jo ta ima. Po uspešno vnesenih podatkih trgovec klikne na gumb »Zaključi nakup«. Prikaže se mu račun v elektronski obliki, katerega lahko trgovec natisne.

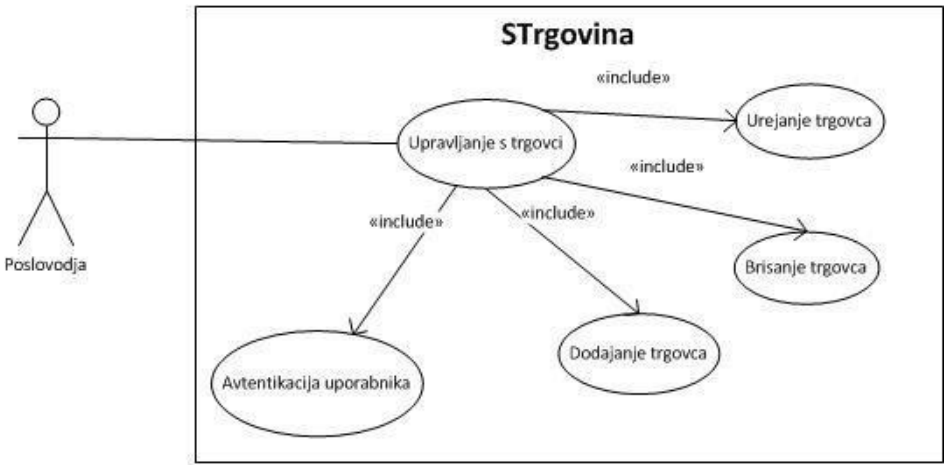
Storniranje nakupa:

Če želi trgovec stornirati nakup klikne na gumb »Storniraj nakup«. Odpre se modalno okno, kjer trgovec v iskalno polje vpiše identifikator nakupa ter klikne na gumb »Išči«. Če sistem najde iskani nakup, le-tega prikaže v tabeli. Trgovec nato označi želeni zadetek ter klikne na gumb »Storniraj«. Sistem izvede storniranje nakupa ter o poteku transakcije obvesti trgovca.

4.4.4 Primer uporabe 4

Ime: Dodajanje, urejanje in brisanje trgovcev

Udeleženci: Poslovodja



Slika 4: UML diagram primera uporabe za dodajanje, urejanje in brisanje trgovcev

Opis:

Poslovodja se mora pred začetkom uporabe poslovne aplikacije STrgovina avtenticirati. V polje »Uporabniško ime:« vpiše dodeljeno uporabniško ime, v polje »Geslo:« pa svoje geslo. Po uspešno opravljeni prijavi se poslovodji prikaže začetni uporabniški vmesnik, kjer poslovodja izbere možnost »Trgovec«. V osnovni delovni površini se prikaže obrazec za delo s trgovci. Tukaj lahko poslovodja dodaja, ureja ali briše trgovce.

Dodajanje trgovca:

Poslovodja na obrazcu za delo s trgovci klikne na gumb »Dodaj trgovca«. Odpre se modalno okno, ki od poslovodje zahteva vpis naslednjih podatkov o trgovcu: ime, priimek, naslov, telefon, e-pošto, geslo ter status. Po uspešno vnesenih podatkih poslovodja klikne na gumb »Dodaj«. Sistem doda trgovca v podatkovno bazo.

Urejanje trgovca:

Poslovodja na obrazcu za delo s trgovci v iskalno polje vpiše ime trgovca. Če sistem najde iskanega trgovca, le-tega prikaže v tabeli. Poslovodja nato označi želeni zadetek ter klikne na gumb »Odpri«. Odpre se pojavno okno, kjer poslovodja lahko spreminja zgolj status trgovca. Po izbranem statusu poslovodja klikne na gumb »Uredi«. Sistem izbranega trgovca posodobi v podatkovni bazi.

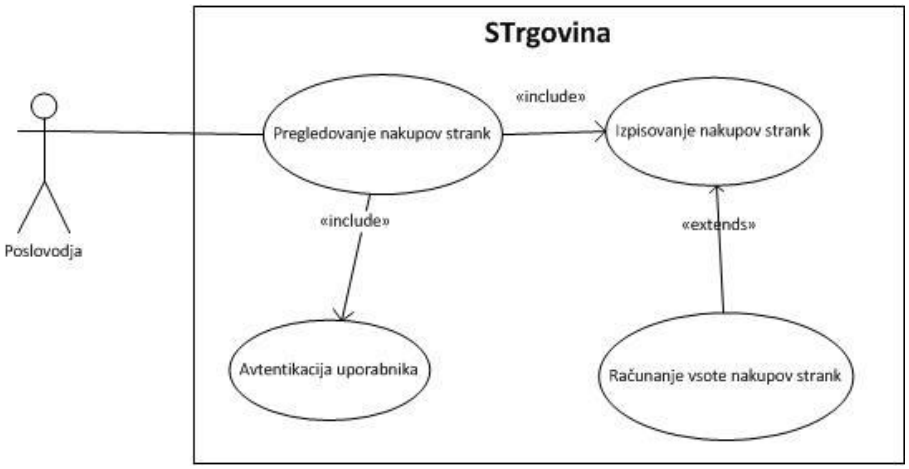
Brisanje trgovca:

Poslovodja na obrazcu za delo s trgovci v iskalno polje vpiše ime trgovca. Če sistem najde iskanega trgovca, le-tega prikaže v tabeli. Poslovodja nato označi želeni zadetek ter klikne na gumb »Odpri«. Odpre se pojavno okno, kjer poslovodja klikne na gumb »Izbriši«. Sistem izbriše izbranega trgovca iz podatkovne baze.

4.4.5 Primer uporabe 5

Ime: Poslovanje

Udeleženci: Poslovodja



Slika 5: UML diagram primera uporabe za poslovanje

Opis:

Poslovodja se mora pred začetkom uporabe poslovne aplikacije STrgovina avtenticirati. V polje »Uporabniško ime:« vpiše dodeljeno uporabniško ime, v polje »Geslo:« pa svoje geslo. Po uspešno opravljeni prijavi se poslovodji prikaže začetni uporabniški vmesnik, kjer poslovodja izbere možnost »Poslovanje«. V osnovni delovni površini se prikaže obrazec, ki omogoča pregled nad poslovanjem.

Izpisovanje nakupov strank:

Poslovodja na obrazcu za poslovanje v iskalno polje vpiše ime stranke ter klikne na gumb »Išči«. Če sistem najde iskano stranko, le-to prikaže v tabeli. Poslovodja nato označi želeni zadetek ter klikne na gumb »Nakupi«. Odpre se pojavno okno, ki vsebuje tabelo nakupov izbrane stranke.

Računanje vsote nakupov strank:

Poslovodja na obrazcu za poslovanje v iskalno polje vpiše ime stranke ter klikne na gumb »Išči«. Če sistem najde iskano stranko, le to prikaže v tabeli. Poslovodja nato označi želeni zadetek ter klikne na gumb »Nakupi«. Odpre se pojavno okno, ki vsebuje tabelo nakupov izbrane stranke ter informacijo o vsoti nakupov. Poleg tega modalno okno poslovodji omogoča izbiro časovnega obdobja, za katerega želi seznam nakupov ter pripadajočo vsoto nakupov izbrane stranke.

4.5 Nefunkcijske zahteve

Specifikacija zahtev poleg funkcijskih vsebuje tudi nefunkcijske zahteve. Le-te določajo kriterije, s katerimi lahko ocenjujemo delovanje sistema. Omenjene zahteve delimo na obratovalne in evolucijske. Med obratovalne sodijo varnost sistema, uporabljivost sistema, … Evolucijske zahteve pa določajo možnosti testiranja, vzdrževanja, razširljivosti in nadgradljivosti. V nadaljevanju bom opisal varnost sistema ter možnost razširljivosti sistema.

4.5.1 Varnost

Aplikacija STrgovina je poslovna aplikacija, ki omogoča upravljanje z artikli, trgovci in s strankami, poleg tega pa omogoča uporabo blagajne in statistični pregled nad poslovanjem podjetja. Tukaj se uporabniki srečujejo s poslovno pomembnimi in zaupnimi podatki, zato mora biti dostop v aplikacijo omejen in nadzorovan. Vstop v sistem je zaščiten z uporabniškim imenom in geslom, ki ga poznajo le aktivni uporabniki. Ti morajo uporabniško ime in geslo vpisati ob prvem dostopu do aplikacije. Prijava uporabnika je veljavna dokler je veljavna seja, torej dokler uporabnik ne zapre brskalnika oziroma sproži akcijo odjave, ki jo sistem omogoča.
Direkten dostop do baze s strani uporabnika bo povsem onemogočen. Vnos podatkov v podatkovno bazo bo kontroliran s strani sistema. Uporabniško vneseni podatki bodo najprej prešli skozi formalno kontrolo sistema, ki jih lahko zavrne ali odobri. V primeru zavrnitve se bo uporabniku na zaslonu prikazal opis napake oziroma razlog zavrnitve. V primeru odobritve pa bodo podatki shranjeni v podatkovno bazo.

4.5.3 Razširljivost

Zahteve poslovnih aplikacij se skozi čas spreminjajo. V času razvoja aplikacije in predvsem v fazi vzdrževanja lahko pride do novih poslovnih priložnosti in izzivov ter do prihoda novih tehnologij. Prav zaradi tega je pomembno razvijati aplikacije, ki so fleksibilne in enostavno razširljive. Načrtovanje take aplikacije lahko predstavlja velik izziv. Lastnost fleksibilnosti zahteva takšno arhitekturo, ki omogoča razvoj in testiranje neodvisnih komponent. Le-te se lahko kasneje nadgrajuje in ločeno testira, brez da bi pri tem vplivali na delovanje ostalih komponent sistema.
Eden izmed učinkovitih pristopov reševanja zgoraj omenjenih problemov je, da aplikacijo razdelimo na določeno število slabo povezanih, medsebojno neodvisnih komponent, katere lahko kasneje integriramo v lupino aplikacije tako, da tvorimo skladno rešitev. Aplikacije, ki so načrtovane in razvite po zgoraj opisanem pristopu, so poznane pod nazivom »Sestavljene aplikacije« in so vključene v okolje Microsoft PRISM.

Prednosti, ki jih zagotavlja pristop »Sestavljenih aplikacij«:

- **Ponovna uporaba**. »Prism« podpira ponovno uporabo, tako da omogoča enostaven razvoj, testiranje in integriranje komponent ter storitev v eno ali več poslovnih aplikacij [2].

- **Razširljivost**. »Prism« podpira razvoj aplikacij, ki so enostavno razširljive. Omogoča razvoj komponent, ki jih lahko na enostaven način integriramo ali zamenjamo z alternativnimi rešitvami v realnem času. Poleg tega pa razvijalcem ponuja možnost, da poslovno aplikacijo razdelijo na posamezne module, ki so lahko neodvisno posodobljeni in uvedeni v aplikacijo [2].

- **Fleksibilnost**. »Prism« omogoča razvoj fleksibilnih aplikacij. Prednost takih aplikacij je v tem, da jih lahko posodobimo takoj, ko so nove funkcionalnosti in zmožnosti implementirane [2].

- **Skupinski razvoj**. »Prism« spodbuja skupinski razvoj poslovnih aplikacij tako, da članom ekipe omogoča neodvisen razvoj in uvedbo novih komponent. Pri tem pripomore k zmanjševanju navzkrižnega dela ter omogoča, da se posamezni člani ekipe osredotočijo zgolj na določeno funkcionalno področje poslovne aplikacije [2].

- **Kakovost**. »Prism« pripomore k povečevanju kakovosti razvoja poslovnih aplikacij tako, da omogoča celovito testiranje posameznih komponent in modulov aplikacije, preden se integrirajo v skupen sistem [2].

Za poslovno aplikacijo STrgovina predlagam uporabo treh plasti. Prva plast je predstavitvena plast, ki omogoča komunikacijo med končnimi uporabniki in sistemom ter vključuje »Silverlight« spletni uporabniški vmesnik. Druga plast je poslovna plast, ki vsebuje vsa poslovna pravila, ki so nujno potrebna za poslovanje trgovine. Tretja plast pa je plast pristopa k podatkom, ki skrbi za dostop in rokovanje s podatkovno bazo.

Predstavitvena in poslovna plast bosta združeni v spletnem klientu, ki bo implementiran po pristopu, kot ga določa okolje Microsoft PRISM. Sestavljen bo iz treh komponent ali modulov. Prva komponenta, imenovana »Prijava«, bo vsebovala spletne uporabniške vmesnike ter vso potrebno poslovno logiko, ki omogoča administracijo uporabnikov. Druga komponenta, imenovana »Trgovec«, bo vključevala spletne uporabniške vmesnike in poslovno logiko, ki omogoča delo trgovcem. Tretja komponenta, imenovana

»Poslovodja«, pa bo vsebovala uporabniške vmesnike in poslovno logiko, ki bo omogočala delo poslovodji. Komponente bodo med seboj slabo povezane, kot to določa pristop »Sestavljenih aplikacij«.

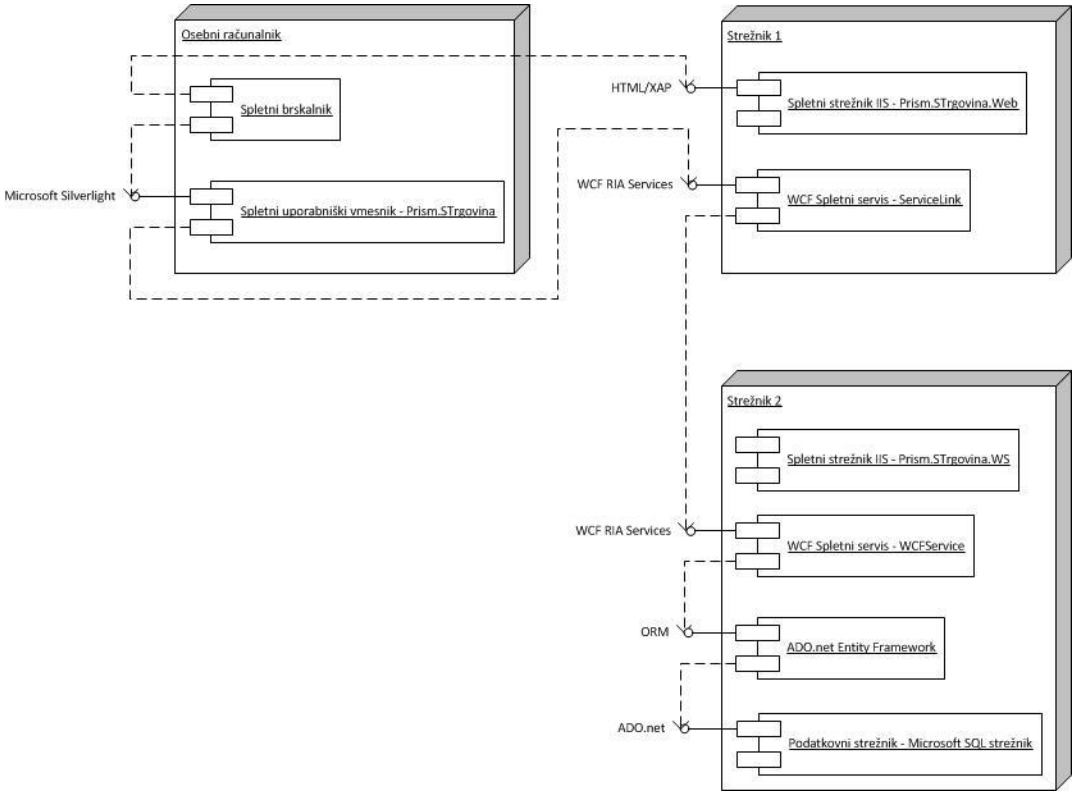
Plast pristopa k podatkom skrbi za dostop in rokovanje s podatkovno bazo. Združena bo v WCF spletnem servisu, ki bo za rokovanje z bazo uporabljal tehnologijo »ADO.NET Entity Framework«.

Aplikacija STrgovina bo tako razdeljena na tri plasti. Predstavitvena in poslovna plast bosta združeni v slabo povezanih in medsebojno neodvisnih komponentah, imenovanih moduli. Plast pristopa k podatkom pa bo definirana znotraj WCF spletnega servisa. Prav zaradi takšne delitve bo aplikacija enostavno razširljiva, fleksibilna ter bo omogočala enostaven skupinski razvoj.

5 NAČRTOVANJE SISTEMA

Načrtovanje sistema predstavlja četrto fazo razvoja programskih produktov. Predstavlja aktivnost, kjer preslikamo model specifikacije zahtev v načrtovalski model sistema. Le-ta prikazuje sistem kot skupino podsistemov, njihove vmesnike in medsebojno odvisnost. Dober načrtovalski model je povod k stabilni arhitekturi, ki zagotavlja zanesljivost sistema, enostavno vzdrževanje in dodajanje novih funkcionalnosti.

5.1 Komponente in vmesniki



Slika 6: UML postavitveni diagram

Arhitektura poslovne aplikacije STrgovina bo določena s sedmimi komponentami. Prva komponenta, imenovana »Prism.STrgovina«, bo vsebovala Silverlight spletni uporabniški vmesnik/klient, modul »Prijava«, modul »Trgovec« in modul »Poslovodja«. Silverlight spletni vmesnik bo implementiran po principu, ki ga določa okolje Microsoft PRISM. Vseboval bo skupno lupino, v katero bodo posamezni moduli vstavljali svoje poglede.

Druga komponenta, imenovana »Prism.STrgovina.Web«, bo Microsoft IIS spletni strežnik, ki bo namenjen gostovanju spletnega klienta »Prism.STrgovina« ter WCF spletnega servisa »ServiceLink«.

Tretjo komponento določa WCF spletni servis, imenovan »ServiceLink«, ki bo omogočal komunikacijo med Silverlight spletnim uporabniškim klientom in WCF spletnim servisom, imenovanim »WCFService«.

Četrta komponenta, imenovana »Prism.STrgovina.WS«, bo Microsoft IIS spletni strežnik, ki bo namenjen gostovanju WCF spletnega servisa, imenovanega »WCFService«.

Peto komponento določa WCF spletni servis, imenovan »WCFService«. Slednji bo vključeval posebne razrede, imenovane »Upravitelji«, ki bodo vsebovali logiko uporabe ORM tehnologije »ADO.net Entity Framework«. Torej, logika, ki bo skrbela za dostop in upravljanje s tabelo »Artikli«, bo vsebovana v razredu »UpraviteljArtikel«; logika, ki bo skrbela za dostop in upravljanje s tabelo »Stranka«, bo v razredu »UpraviteljStranka«; logika, ki bo skrbela za dostop in upravljanje s tabelo »Trgovec«, bo v razredu »UpraviteljTrgovec«, in logika, ki bo skrbela za dostop in upravljanje s tabelama »Nakup« in »Nakup_Artikel«, pa v razredu »UpraviteljPoslovanja«. Uporaba in dostop do zgoraj omenjenih poslovnih razredov, imenovanih »Upravitelji«, bosta omogočena preko storitvenih vmesnikov, ki bodo implementirani v skladu s pravili, ki jih določa tehnologija WCF RIA Services. Slednja omogoča hitro implementacijo večplastnih Silverlight spletnih aplikacij, brez da bi se pri tem osredotočali na povezavo storitev med klienti in strežniki za upravljanje s podatki [6].

Šesto komponento določa Microsoftova tehnologija »ADO.net Entity Framework«. Slednja bo skrbela za dostop in upravljanje s podatkovno bazo. »ADO.net Entity Framework« sodi med ORM »Object-relational mapping« programsko opremo, ki je namenjena pretvorbi podatkov med nezdružljivimi tipi sistemov v objektno-usmerjenih programskih jezikih. Nad tabelami podatkovne baze kreira virtualne objekte, ki jih lahko uporabimo znotraj programskega jezika [7].

Sedmo komponento določa Microsoftov podatkovni strežnik MS SQL. Ta bo vključeval Microsoft SQL podatkovno bazo.

V nadaljevanju zaključne naloge so podrobneje opisane zgoraj omenjene komponente. Opisane so v treh sklopih. Prvi sklop, imenovan »Spletni uporabniški vmesnik – Prism.STrgovina«, opisuje Silverlight spletni uporabniški vmesnik »Prism.STrgovina« ter module »Prism.STrgovina.Modules«, ki določajo njegovo funkcionalnost. Drugi sklop, imenovan »Spletni strežnik – Prism.STrgovina.Web«, opisuje spletni strežnik »Prism.STrgovina.Web«, ki gostuje spletnega uporabniškega klienta ter WCF spletni servis »ServiceLink«. Tretji sklop, imenovan »Spletni strežnik – Prism.STrgovina.WS«, pa vsebuje opis WCF spletnega servisa »WCFService«, »ADO.net Entity Framework«

programske opreme in opis MS SQL podatkovnega strežnika.

5.1.1 Spletni uporabniški vmesnik – Prism.STrgovina

Spletni uporabniški vmesnik »Prism.STrgovina« bo implementiran po principu, ki ga določa okolje Microsoft PRISM. Slednje definira načrtovalske vzorce, ki omogočajo razvoj fleksibilnih in enostavno razširljivih poslovnih aplikacij. Razvoj aplikacij v okolju Microsoft PRISM temelji na tehnologiji Silverlight. Microsoft Silverlight je aplikacijsko ogrodje, ki omogoča pisanje in poganjanje bogatih spletnih aplikacij. Uporabniški vmesniki so deklarirani v formatu XAML (Extensible Application Markup Language) in programirani s pomočjo podskupine .NET programskega ogrodja [3].

Uporabniški vmesnik aplikacije STrgovina bo implementiran kot lupina, ki bo predstavljala osnovni izgled aplikacije. Definirana bo s tremi področji ali regijami, v katere bodo moduli nalagali svoje poglede. Regija ali področje predstavlja logični prostor, v katerega se lahko nalagajo Silverlight uporabniški vmesniki ali pogledi [2]. Prednost uporabe regij je, da omogočajo spremembe na izgledu osnovnega uporabniškega vmesnika, brez da bi pri tem posegali v aplikacijsko logiko. Lupina aplikacije STrgovina bo vsebovala regijo »MenuRegion«, regijo »MainRegion« in regijo »LoginRegion«. V regijo »MenuRegion« bodo moduli nalagali poglede, ki bodo vsebovali menijsko vrstico, v regijo »MainRegion« poglede, ki bodo vsebovali poslovne procese (npr. pogled, ki je namenjen upravljanju z artikli …), v regijo »LoginRegion« pa poglede, ki so namenjeni administraciji uporabnikov.

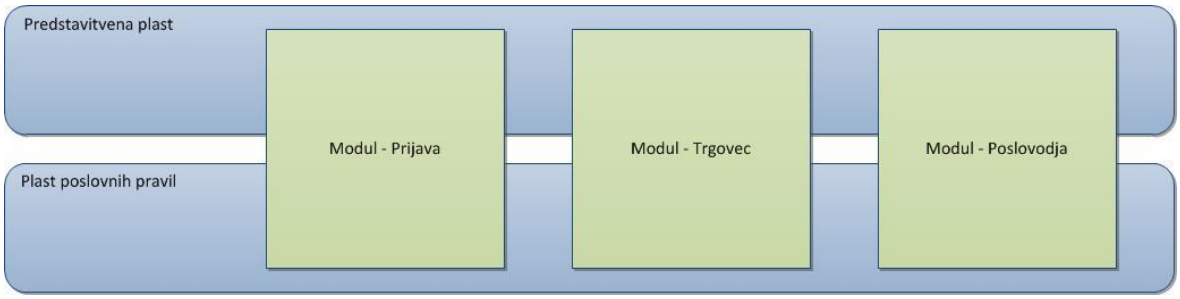
5.1.1.1 Moduli – Prism.STrgovina.Modules

Spletni uporabniški vmesnik »Prism.STrgovina« bo vseboval modul »Trgovec«, modul »Prijava« in modul »Poslovodja«. Modul je zaokrožen paket funkcionalnosti, ki je lahko neodvisno načrtovan, implementiran in testiran. V večini primerov se moduli razvijajo in vzdržujejo v ločenih razvojnih skupinah. Module lahko uporabimo za predstavitev specifične poslovne funkcionalnosti, saj združujejo vso potrebno poslovno logiko, uporabniške vmesnike in podatkovne modele [2]. V aplikaciji STrgovina bo za administracijo uporabnikov skrbel modul Prijava, za upravljanje z artikli, s strankami in z blagajno modul Trgovec, za upravljanje s trgovci in statističnim pregledom poslovanja trgovine pa modul Poslovodja.

Kot že omenjeno, lahko poslovno aplikacijo STrgovina razdelimo na predstavitveno plast, poslovno plast in plast pristopa k podatkom. Moduli poslovne aplikacije STrgovina definirajo predstavitveno in poslovno plast. Predstavitvena plast združuje grafične uporabniške vmesnike, ki omogočajo komunikacijo med končnimi uporabniki in

sistemom. Moduli implementirajo dano plast, saj vsebujejo vse potrebne uporabniške poglede. Plast poslovnih pravil določa poslovna pravila uporabe aplikacije ter vsebuje formalno kontrolo uporabniško vnesenih podatkov. Moduli vsebujejo posebne razrede, ki definirajo pravila uporabe poslovne aplikacije STrgovina, zato določajo plast poslovnih pravil.

Tretja plast je plast pristopa k podatkom, ki je vsebovana v WCF spletnem servisu »WCFService«. Slednji bo opisan v nadaljevanju zaključne naloge.



Slika 7: Moduli poslovne aplikacije STrgovina

Aplikacija STrgovina bo za odkrivanje in nalaganje modulov v lupino uporabljala posebno zbirko imenovano katalog modulov. Katalog modulov je sestavni del okolja Microsoft PRISM, saj vsebuje vse potrebne informacije o modulih, ki so nujno potrebni za delovanje poslovne aplikacije [2].

5.1.2 Spletni strežnik – Prism.STrgovina.Web

Poslovna aplikacija STrgovina bo za gostovanje spletnega uporabniškega vmesnika »Prism.STrgovina« uporabljala Microsoftov spletni strežnik IIS (Internet Information Services), ker omogoča gostovanje Microsoft Silverlight spletnih aplikacij. Pojem spletni strežnik se lahko navezuje na strojno opremo (računalnik) ali programsko opremo (računalniški program). Osnovna funkcija spletnega strežnika je gostovanje spletnih strani ali spletnih aplikacij [1].
V primeru aplikacije STrgovina bo spletni uporabniški vmesnik implementiran s pomočjo tehnologije Silverlight. Prav zaradi tega pa se dostop in uporaba spletne aplikacije omejita na brskalnike, ki podpirajo dano tehnologijo.
Spletni strežnik bo vseboval tudi WCF spletni servis, imenovan »ServiceLink«. Le-ta bo omogočal komunikacijo med Silverlight spletnim uporabniškim vmesnikom in WCF spletnim servisom imenovanim »WCFService«. Več informacij o WCF spletnem servisu bo opisanih v nadaljevanju.

5.1.3 Spletni strežnik – Prism.STrgovina.WS

Poslovna aplikacija STrgovina bo za gostovanje WCF spletnega servisa uporabljala Microsoftov spletni strežnik IIS (Opis spletnega strežnika: 5.1.2 Spletni strežnik – Prism.STrgovina.Web). Spletni servis je programska funkcija, ki je dostopna na določenem spletnem naslovu. Predstavlja metodo komunikacije med dvema elektronskima napravama preko svetovnega spleta [8].

Plast pristopa k podatkom omogoča dostop in rokovanje s podatkovno bazo ter je implementirana v WCF spletnem servisu imenovanem »WCFService«.

Slika 8: WCF spletni service - WCFService

»Windows Communication Foundation« ali WCF je skupina API-jev »application programming interface« v .NET programskem ogrodju, ki služi za implementacijo povezanih in storitveno usmerjenih aplikacij »SOA«. »SOA« je niz načel in metodologij za načrtovanje in razvoj programske opreme v obliki storitev [9]. Storitev ima dobro definirano poslovno funkcionalnost, katero lahko uporabimo za različne namene. Klienti npr. spletne aplikacije lahko dostopajo in uporabljajo dane storitve preko storitvenih vmesnikov, poznanih pod kratico WSDL (Web Service Description Language).

WCF klienti se na WCF storitev povežejo preko končnih točk. Končna točka je določena z URL naslovom, kjer se storitev nahaja ter iz zavezujočih informacij. Zavezujoče informacije vsebujejo komunikacijski protokol, preko katerega bo storitev dostopna ter razne varnostne mehanizme. WCF vsebuje že definirane zavezujoče informacije. Med slednje sodijo SOAP preko http-ja, SOAP preko tcp-ja, ter SOAP preko sporočilnih vrst. Interakcija med WCF storitvami in WCF klienti je določena s SOAP lupino. SOAP (Simple Object Access Protocol) je protokol za izmenjavo strukturiranih informacij med spletnimi servisi preko računalniškega omrežja. Za format sporočil uporablja XML (Extensible Markup Language) notacijo, za prenos sporočil pa običajno http (Hypertext Transfer Protocol) ali SMTP (Simple Mail Transfer Protocol) protokol.
Torej, kadar želi klient dostopati in uporabljati določeno WCF spletno storitev, mora

izpolnjevati vse zahteve končne točke, preko katere je dana storitev na voljo.

WCF spletni servis poslovne aplikacije STrgovina bo ponujal pet skupin spletnih storitev. Med slednje sodijo storitve, ki omogočajo delo z artikli, storitve, ki omogočajo delo s strankami, storitve, ki omogočajo delo s trgovci, storitve, ki so namenjene poslovanju, in storitve, ki omogočajo delo poslovodji.

Za dostop in rokovanje s podatkovno bazo bo spletni servis aplikacije STrgovina uporabljal Microsoftovo tehnologijo, imenovano ADO.net Entity Framework. Slednja sodi med ORM (Object-relational mapping) programsko opremo, ki je namenjena pretvorbi podatkov med nezdružljivimi sistemi v objektno usmerjenih programskih jezikih [7]. V splošnem ORM tehnologija kreira virtualno podatkovno bazo, ki jo lahko uporabimo znotraj programskega jezika.

Podatkovni strežnik poslovne aplikacije STrgovina bo Microsoft SQL strežnik. Le-ta bo vključeval Microsoft SQL podatkovno bazo. Dostop in upravljanje s podatkovno bazo bo zagotovljeno prek ADO.net gonilnika, ki ga bo uporabljala ORM tehnologija ADO.net Entity Framework.

6 NAČRTOVANJE PROGRAMA

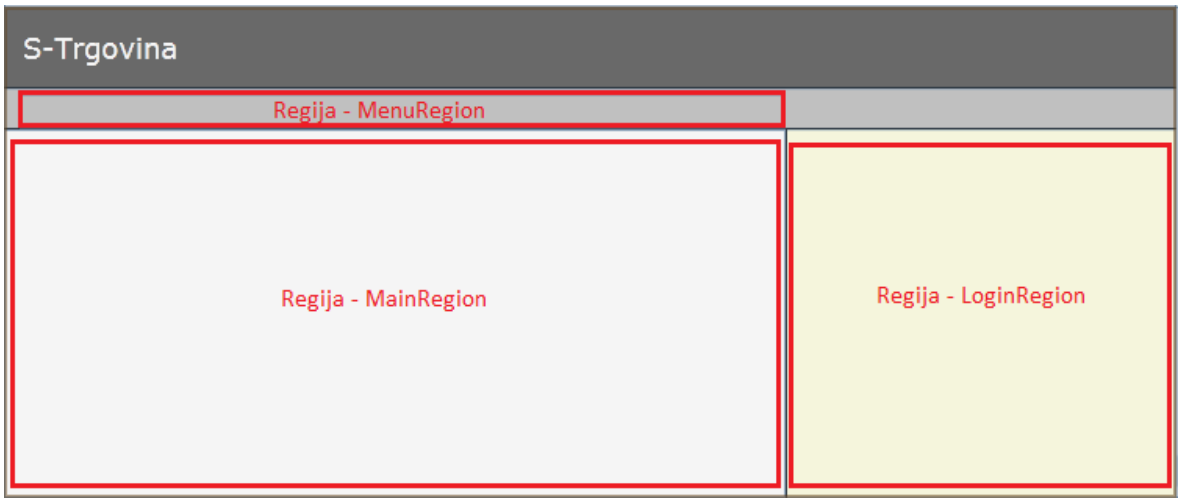
Načrtovanje programa predstavlja peto fazo razvoja programskih produktov. V dani fazi se načrtovalci usmerjajo predvsem v načrtovanje komponent sistema. Glavne aktivnosti so: identifikacija dodatnih objektov, dopolnitev obstoječih načrtov sistema, natančno definiranje vmesnikov med posameznimi podsistemi ter optimizacija načrta objektov.

6.1 Spletni uporabniški vmesnik – Prism.Strgovina

Spletni uporabniški vmesnik aplikacije STrgovina predstavlja vmesnik med uporabniki in sistemom. Namenjen je grafičnemu upravljanju s poslovno pomembnimi podatki, kot so podatki o artiklih, strankah, trgovcih in nakupih. Spletni uporabniški vmesnik bo uporabnikom omogočal dodajanje, urejanje, pregledovanje in brisanje artiklov, strank in trgovcev, ter upravljanje z blagajno.

Implementiran bo po principu, ki ga določa okolje Microsoft PRISM. Le-to določa smernice pri razvoju bogatih spletnih aplikacij. Uporabniški vmesnik bo zato implementiran v aplikacijskem ogrodju Microsoft Silverlight. Microsoft Silverlight je aplikacijsko ogrodje, ki omogoča pisanje in poganjanje bogatih spletnih aplikacij. Uporabniški vmesniki so deklarirani v formatu XAML (Extensible Application Markup Language) in programirani s pomočjo podskupine .NET programskega ogrodja.

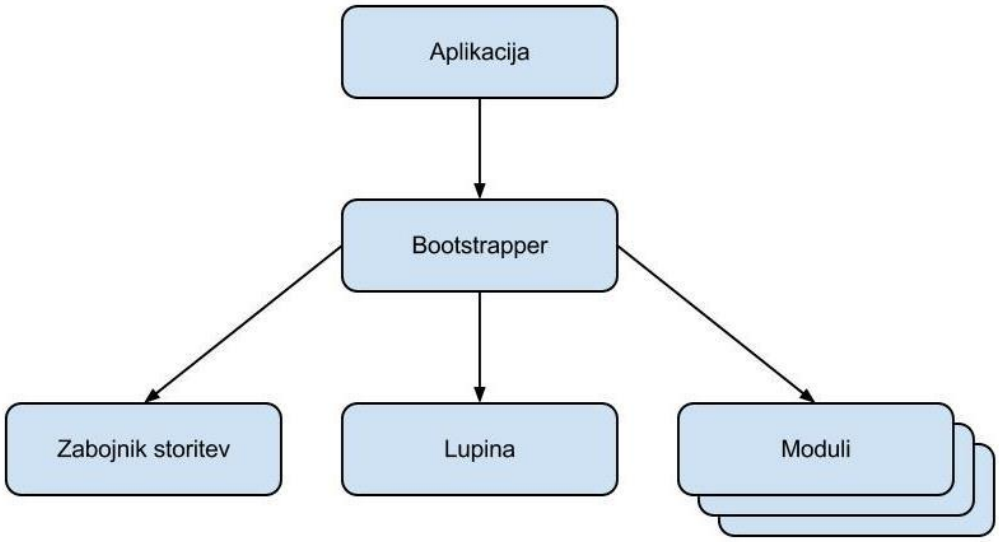
Osnovni uporabniški vmesnik bo implementiran kot lupina aplikacije. Lupina je gostitelj v katerega se nalagajo moduli. Predstavlja osnovno postavitev oziroma izgled aplikacije. Izgled je določen z regijami ali s področji. Regija ali področje predstavlja logični prostor, v katerega se lahko nalagajo Silverlight uporabniški vmesniki ali pogledi. Prednost uporabe regij je, da omogočajo spremembe na izgledu osnovnega uporabniškega vmesnika, brez da bi pri tem posegali v aplikacijsko logiko. Lupina aplikacije STrgovina bo vsebovala regijo »MenuRegion«, regijo »MainRegion« in regijo »LoginRegion«. V regijo »MenuRegion« bodo moduli nalagali poglede, ki bodo vsebovali menijsko vrstico, v regijo »MainRegion« poglede, ki bodo vsebovali poslovne procese (npr. pogled, ki je namenjen upravljanju z artikli, …), v regijo »LoginRegion« pa poglede, ki so namenjeni administraciji uporabnikov.



Slika 9: Spletni uporabniški vmesnik – lupina

6.1.1 Razred Bootstrapper

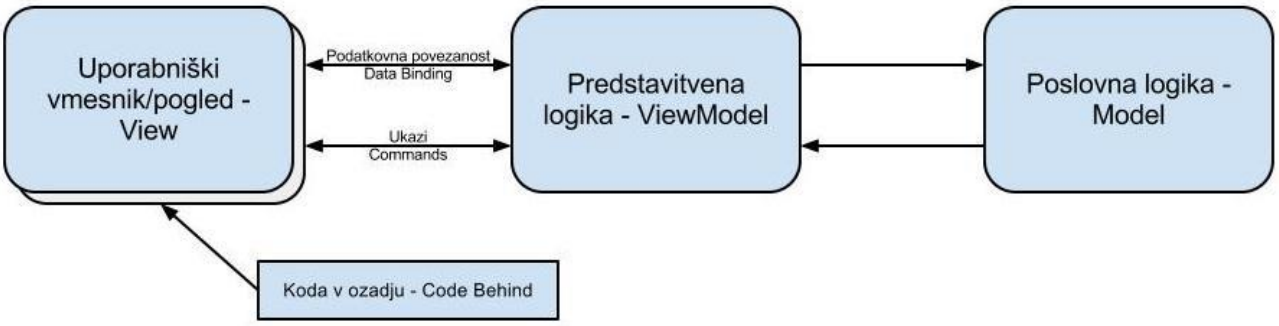
Bootstrapper je razred, ki povezuje aplikacijo z Microsoft PRISM programskimi knjižnicami. Med te sodita MEF ali Unity programski knjižnici, ki omogočata razvoj razširljivih aplikacij. Vsaka aplikacija, ki želi delovati po pristopu, ki ga določa okolje Microsoft PRISM, potrebuje svojo implementacijo razreda Bootstrapper. Razred Bootstrapper deduje iz razreda MefBootstrapper ali UnityBootstrapper. Namenjen je inicializaciji potrebnih Microsoft PRISM komponent, med katere sodi inicializacija lupine in modulov aplikacije, ter vseh potrebnih storitev, ki jih moduli potrebujejo za svoje delovanje.



Slika 10: Bootstrapper

6.1.2 »Model – View – ViewModel« načrtovalski vzorec

Za prikaz in implementacijo aplikacijske in poslovne logike v modulih aplikacije STrgovina bomo uporabili načrtovalski vzorec »Model-View-ViewModel« MVVM. MVVM je načrtovalski vzorec, ki loči poslovno in predstavitveno logiko aplikacije od grafičnega uporabniškega vmesnika [2]. Prednost takega pristopa je, da omogoča ločeno testiranje grafičnega uporabniškega vmesnika, predstavitvene in poslovne logike ter povečuje zmožnost vzdrževanja in širjenja funkcionalnosti.



Slika 11: MVVM razredi in medsebojna odvisnost

Uporabniški vmesnik/pogled – View:

Glavna naloga uporabniškega vmesnika je definiranje strukture in videza aplikacije. Sestavljen je iz dveh delov. Prvi del predstavlja uporabniku vidno komponento – grafični uporabniški vmesnik, ki je definiran v deklarativnem jeziku, imenovanem XAML. XAML ali »Extensible Application Markup Language« predstavlja osnovo za oblikovanje grafičnega uporabniškega vmesnika. Drugi del je definiran kot programski razred, ki lahko vsebuje predstavitveno in poslovno logiko uporabniškega vmesnika ter ga poznamo pod izrazom koda v ozadju. V primeru uporabe MVVM načrtovalskega vzorca razred koda v ozadju vsebuje zgolj inicializacijo razreda »ViewModel«, ki vsebuje predstavitveno logiko uporabniškega vmesnika.

Predstavitvena logika – ViewModel:

Razred »ViewModel« v načrtovalskem vzorcu MVVM vsebuje predstavitveno logiko in podatke, ki jih potrebuje uporabniški vmesnik oziroma pogled. »ViewModel« v splošnem implementira vse lastnosti in ukaze na katere se grafični uporabniški vmesnik poveže, ter jih nato prikaže oziroma izvede [2]. Razred »ViewModel« predstavlja povezavo med grafičnim uporabniškim vmesnikom in poslovno logiko ter podatki, ki so vsebovani v razredu »Model«.

Poslovna logika – Model:

Razred »Model« v načrtovalskem vzorcu MVVM predstavlja poslovno logiko in podatke. Poslovna logika skrbi za pridobivanje in upravljanje s poslovno pomembnimi podatki ter za preverjanje in določanje poslovnih pravil.

V poslovni aplikaciji STrgovina razred »ViewModel« določa tako predstavitveno kot tudi poslovno logiko. Do takega sklepa smo prišli zaradi uporabe WCF spletnega servisa in tehnologije WCF RIA Services, ki nam omogoča da klient prepozna vse podatkovne objekte in metode, ki jih implementira WCF spletni servis. Posledično se uporaba razreda »Model« ne obrestuje, saj vse potrebne podatkovne objekte pridobimo od WCF spletnih storitev.

6.2 Moduli – Prism.STrgovina.Moduli

Modul je zaokrožen paket funkcionalnosti, ki je lahko neodvisno načrtovan, implementiran in testiran. V večini primerov se moduli razvijajo in vzdržujejo v ločenih razvojnih skupinah. Module lahko uporabimo za predstavitev specifične poslovne funkcionalnosti, saj združujejo vso potrebno poslovno logiko, uporabniške vmesnike in podatkovne modele[2]. Modul je označen s posebnim razredom, ki implementira IModule vmesnik. Med inicializacijo moduli registrirajo svoje storitve in uporabniške poglede. Le-te v odvisnosti od zahtev nato naložijo v določeno regijo. Moduli prepoznajo regije na lupini aplikacije s pomočjo razreda RegionManager. Razred RegionManager je odgovoren za vzdrževanje informacij o regijah na lupini aplikacije, ter za proces nalaganja uporabniških pogledov v regije.

Aplikacija STrgovina bo vsebovala tri module. Modul »Prijava« bo skrbel za administracijo uporabnikov, modul »Trgovec« za upravljanje z artikli, s strankami in z blagajno, modul »Poslovodja« pa za upravljanje s trgovci in statističnimi podatki poslovanja trgovine. Aplikacija STrgovina bo za odkrivanje in nalaganje modulov v lupino uporabljala posebno zbirko imenovano katalog modulov. Katalog modulov je sestavni del okolja Microsoft PRISM, saj vsebuje vse potrebne informacije o modulih. Za vsak modul vsebuje lokacijo inicializacijskega razreda ter informacijo o načinu inicializacije. Modul je lahko inicializiran ob zagonu aplikacije ali pa na zahtevo. V primeru, da bo modul inicializiran na zahtevo, razred Bootstrapper poskrbi, da aplikacija prepozna dani modul, ter ga na uporabniško zahtevo inicializira.

6.2.1 Microsoft PRISM komponente, ki bodo vsebovane v aplikaciji STrgovina

Za implementacijo poslovne aplikacije STrgovina kot sestavljene aplikacije v okolju Microsoft PRISM bomo uporabili tri osnovne komponente, in sicer: komponenta Region Manager, komponenta Module Manager in komponenta Event Aggregator.

6.2.1.1 Region Manager

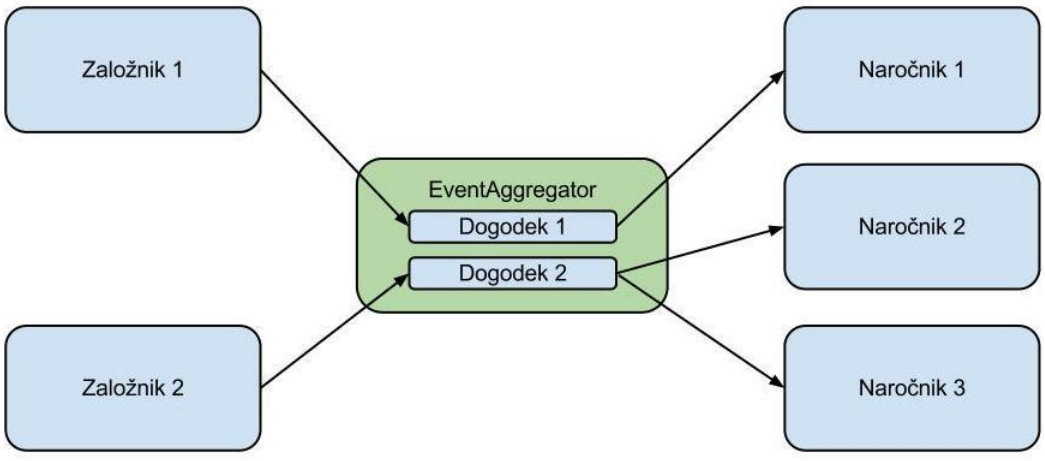
Region Manager je storitev ali komponenta okolja Microsoft PRISM, ki je namenjena procesu registracije uporabniških pogledov posameznega modula, ter procesu nalaganja danih uporabniških pogledov v regije, ki so definirane na lupini aplikacije [2].

6.2.1.2 Module Manager

Module Manager je storitev ali komponenta okolja Microsoft PRISM, ki je namenjena procesu nalaganja modulov v aplikacijo. Vsebuje informacijo o modulih, ki jo pridobi iz kataloga modulov. Pri nalaganju modulov s komponento Module Manager je pomembno, da se modul inicializira na zahtevo [2].

6.2.1.3 Event Aggregator

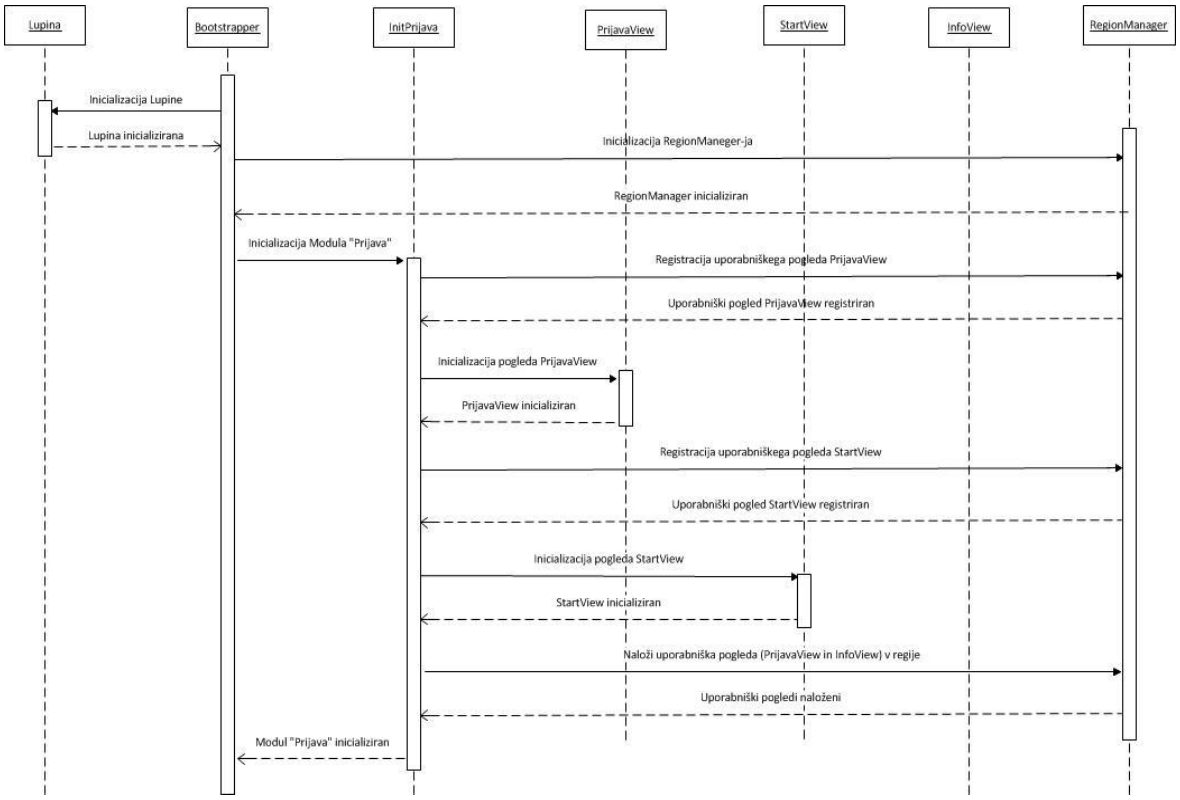
Komponente sestavljenih aplikacij velikokrat potrebujejo medsebojno komunikacijo. Microsoft PRISM omogoča takšno komunikacijo preko storitve, imenovane Event Aggregator [2]. Slednja implementira mehanizem objavljanja in naročanja na dogodke. Komponente sestavljenih aplikacij se tako razdelijo na založnike. Tiste, ki dogodke objavljajo, ter na naročnike, ki se na dogodek naročajo [4]. Mehanizem omogoča komunikacijo med komponentami, brez da bi se le-te medsebojno povezovale.



Slika 12: Prikaz mehanizma, ki ga implementira storitev EventAggregator

6.2.2 Modul »Prijava« - Prism.STrgovina.Moduli.Prijava

Modul »Prijava« bo reševal poslovni problem administriranja uporabnikov. Administracija uporabnikov predstavlja prvi korak pri uporabi poslovne aplikacije STrgovina, zato se bo modul »Prijava« kot prvi inicializiral ter svoje poglede naložil v ustrezne regije. Vseboval bo tri uporabniške poglede. Prvi pogled bo »PrijavaView«, ki bo vseboval prijavno formo, drugi pogled bo »StartView«, ki bo vseboval splošne informacije o poslovni aplikaciji STrgovina, in tretji pogled bo »InfoView«, ki bo vseboval podrobnejše informacije o trenutno prijavljenem uporabniku. Ob inicializaciji bo modul »Prijava« registriral in naložil pogled »PrijavaView« v regijo »LoginRegion« ter registriral in naložil pogled »StartView« v regijo »MainRegion«. Uporabniški pogled »InfoView« bo modul registriral in naložil v regijo »LoginRegion« po uspešno izvedeni prijavi uporabnika.



Slika 13: UML Sekvenčni diagram inicializacije modula »Prijava«

Uporabniški vmesnik/pogled »PrijavaView«:

Uporabniški pogled »PrijavaView« je namenjen administriranju uporabnikov. Sestavljen je po zgoraj opisanem načrtovalskem vzorcu MVVM. »View« oziroma grafični uporabniški vmesnik vsebuje prijavno formo aplikacije STrgovina ter razred »PrijavaView.cs«, ki določa kodo v ozadju. Namenjen je inicializaciji poslovnega razreda »PrijavaViewModel«, ki določa predstavitveno in poslovno logiko danega uporabniškega vmesnika.

Razred »PriavaViewModel« omogoča izvedbo akcije prijava uporabnika. Slednja zahteva poizvedbo v podatkovno bazo, ter logiko nalaganja novih pogledov v lupino aplikacije STrgovina. Prav zaradi tega razred vsebuje tri pomembne komponente. Prva komponenta je del tehnologije WCF RIA Services in se imenuje ServiceReference. Komponenta ServiceReference omogoča klientu dostop do WCF spletnega servisa ter do WCF storitev. Druga komponenta se imenuje ModuleManager, ki je del Microsoft PRISM knjižnice. Omogoča nalaganje modulov v aplikacijo, pri čemer morajo imeti moduli podatek o inicializaciji določen na zahtevo. Tretja komponenta je prav tako del Microsoft PRISM knjižnice in se imenuje EventAggregator. Omogoča izmenjevanje raznih dogodkov in akcij med moduli aplikacije.

Uporabniški vmesnik/pogled »StartView«:

Uporabniški pogled »StartView« je namenjen prikazovanju splošnih informacij o poslovni aplikaciji STrgovina. Sestavljen je zgolj iz grafičnega uporabniškega vmesnika, ki vsebuje opis glavnih funkcionalnosti poslovne aplikacije STrgovina ter razred »StartView.cs«, ki predstavlja kodo v ozadju. Ker dani uporabniški vmesnik ne omogoča nobenih funkcionalnosti, ne premore predstavitvene ali poslovne logike.

Uporabniški vmesnik/pogled »InfoView«:

Uporabniški pogled »InfoView« je namenjen prikazovanju splošnih informacij o trenutno prijavljenem uporabniku. Sestavljen je po zgoraj opisanem načrtovalskem vzorcu MVVM. »View« oziroma grafični uporabniški vmesnik vsebuje pregled osnovnih informacij prijavljenega uporabnika ter razred »InfoView.cs«, ki določa kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »InfoViewModel«, ki določa predstavitveno in poslovno logiko danega uporabniškega vmesnika.

Razred »InfoViewModel« omogoča pridobitev informacij o trenutno prijavljenem uporabniku. Za pridobitev danih informacij potrebuje dostop do WCF spletnega servisa in WCF spletnih storitev ter identifikator uporabnika. Za dostop do WCF spletnega servisa in storitev razred »InfoViewModel« potrebuje komponento, ki je del tehnologije WCF RIA Services ter se imenuje ServiceReference. Za pridobitev identifikatorja trenutno prijavljenega uporabnika pa potrebuje komponento, ki je del Microsoft PRISM knjižnice in se imenuje EventAggregator.

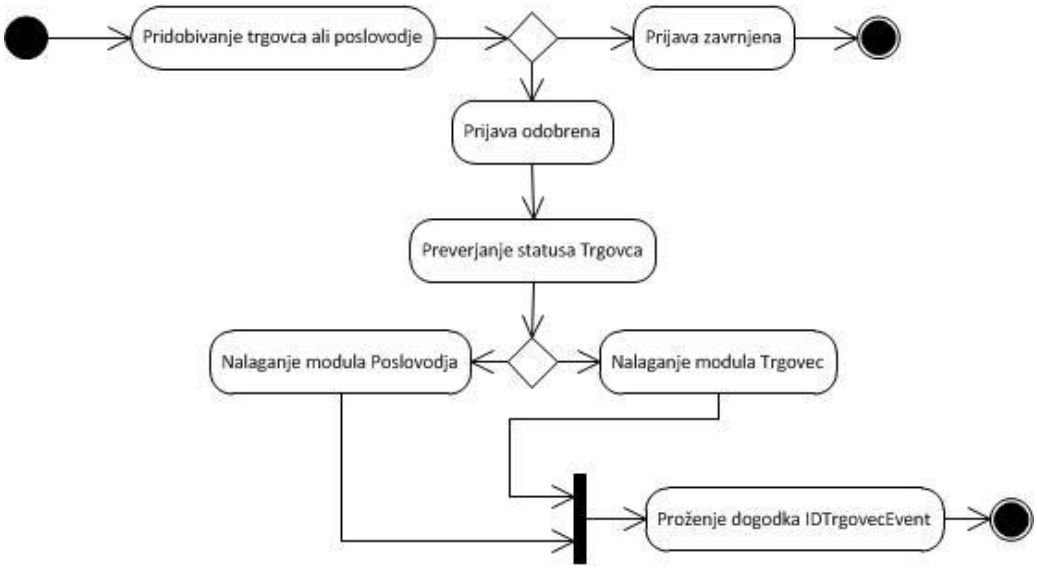
6.2.2.1 Opis glavnih aktivnosti modula »Prijava«

Glavni aktivnosti, ki jih modul »Prijava« določa, sta aktivnost prijave uporabnika in aktivnost pridobivanja osnovnih informacij o trenutno prijavljenem uporabniku. Obe aktivnosti sta podrobneje opisani v nadaljevanju zaključne naloge.

6.2.2.1.1 Aktivnost prijave uporabnika

Udeleženi razredi ali storitve:

- uporabniški pogled »PrijavaView«;
- poslovni razred »PrijavaViewModel«;
- WCF spletna storitev »TrgovecPrijava«.



Slika 14: UML diagram aktivnosti za proces prijave uporabnikov v modulu »Prijava«

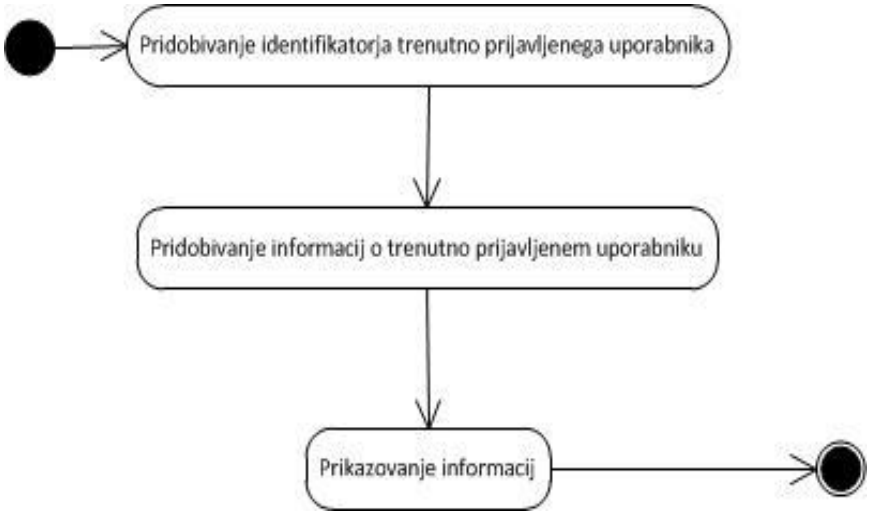
Pri procesu prijave uporabnika bo razred »PrijavaViewModel« dostopal do WCF storitve, imenovane »TrgovecPrijava«. Ta storitev omogoča iskanje trgovcev v podatkovni bazi s točno določenim uporabniškim imenom in geslom. Storitev vrne iskanega trgovca v primeru, da se navedeno uporabniško ime in geslo nahajata v bazi, sicer vrne prazen objekt. Klient na drugi strani pridobi odgovor v obliki objekta Trgovec ter nadaljuje z akcijo prijave trgovca. V primeru, da objekt, ki ga klient pridobi, ni prazen, razred »PrijavaViewModel« preveri, ali je dani trgovec tudi poslovodja. Če je dani trgovec poslovodja, razred »PrijavaViewModel« s pomočjo komponente ModuleManager naloži modul »Poslovodja« v aplikacijo. Ko se modul »Poslovodja« naloži v aplikacijo, razred »PrijavaViewModel« s pomočjo komponente EventAggregator objavi dogodek, imenovan »IDTrgovecEvent«, kateri nosi informacijo o identifikatorju prijavljenega trgovca oziroma

poslovodje. Če dani uporabnik predstavlja zgolj trgovca, razred »PrijavaViewModel« s pomočjo komponente ModuleManager naloži modul »Trgovec«. Ko se modul »Trgovec« naloži v aplikacijo, razred »PrijavaViewModel« s pomočjo komponente EventAggregator objavi dogodek, imenovan »IDTrgovecEvent«, ki nosi informacijo o identifikatorju prijavljenega trgovca oziroma poslovodje. V primeru, da je objekt Trgovec prazen, pa razred »PrijavaViewModel« uporabniku sporoči, da dani trgovec ne obstaja.

6.2.2.1.2 Aktivnost pridobivanja osnovnih informacij o trenutnem uporabniku

Udeleženi razredi ali storitve:

- uporabniški pogled »InfoView«;
- poslovni razred »InfoViewModel«;
- WCF spletna storitev »VrniTrgovecID«.

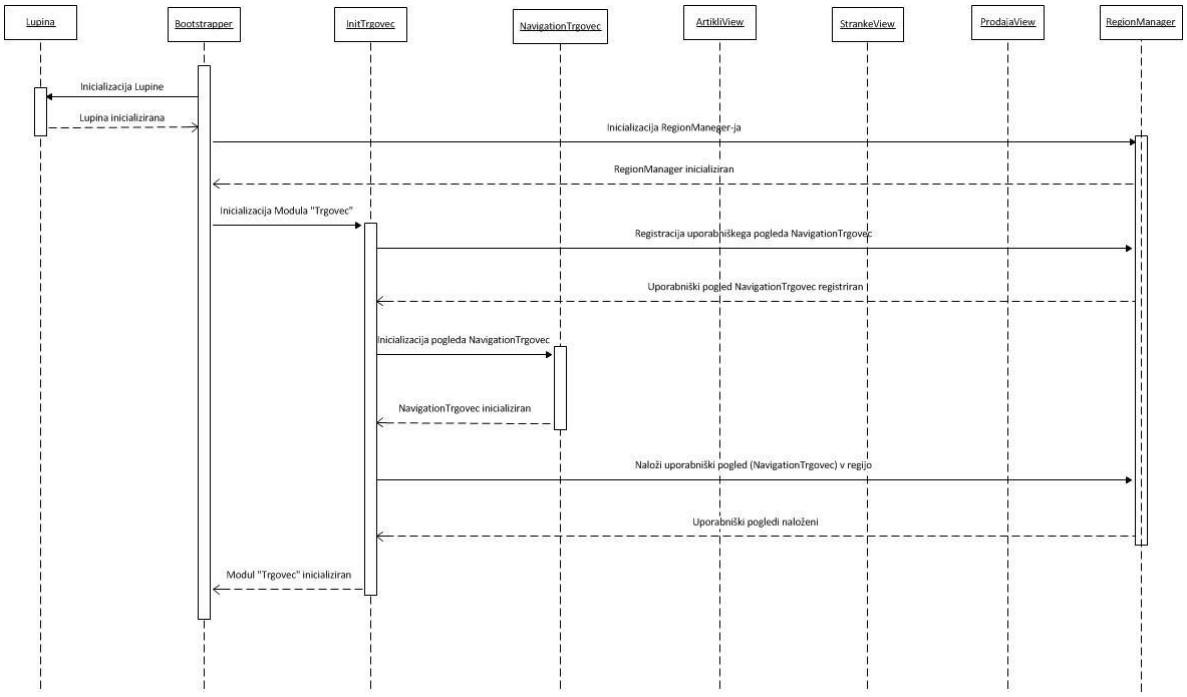


Slika 15: UML diagram aktivnosti pridobivanja osnovnih informacij o uporabniku

Pri procesu pridobivanja osnovnih informacij o trenutno prijavljenem uporabniku bo razred »InfoViewModel« dostopal do WCF spletne storitve, imenovane »VrniTrgovecID«. Ta storitev omogoča iskanje trgovcev v podatkovni bazi s točno določenim identifikatorjem. Da bi razred »InfoViewModel« lahko uporabil dano storitev, potrebuje identifikator trenutno prijavljenega trgovca. Tega pridobi s pomočjo komponente EventAggregator, in sicer tako, da se naroči na dogodek, imenovan »IDTrgovecEvent«. Po uspešno pridobljenem identifikatorju, razred »InfoViewModel« pridobi podatke o trenutno prijavljenem uporabniku preko WCF spletne storitve »VrniTrgovecID«. Rezultat storitve se shrani v objekt Trgovec, ki ga grafični uporabniški vmesnik »InfoView« prikaže uporabniku.

6.2.3 Modul »Trgovec« - Prism.STrgovina.Moduli.Trgovec

Modul »Trgovec« bo uporabnikom omogočal dodajanje, urejanje, brisanje in iskanje artiklov oziroma strank ter uporabo blagajne, ki bo omogočala ustvarjanje novih nakupov, pregledovanje in storniranje že obstoječih ter tiskanje računov. Vseboval bo dva osnovna pregleda, uporabniški pogled za poslovanje, pogled za navigacijo in tri modalna okna. Med preglede sodita uporabniški pogled »ArtikliView«, ki bo omogočal pregledovanje in iskanje artiklov, ter uporabniški pogled »StrankaView«, ki bo omogočal pregledovanje in iskanje strank. Uporabniški pogled za poslovanje bo »ProdajaView«, ki bo vseboval blagajno. Uporabniški pogled za navigacijo, imenovan »NavigationTrgovec«, bo vseboval menijsko vrstico. Med modalna okna pa sodijo: modalno okno za dodajanje, brisanje ali urejanje artiklov, imenovano »DodajArtikelChildView«; modalno okno za dodajanje, brisanje ali urejanje strank, imenovano »DodajStrankaChildView«; ter modalno okno za pregledovanje in storniranje nakupov, imenovano »StornirajNakupChildView«.
Modul »Trgovec« se bo inicializiral na zahtevo. Zahteva za inicializacijo bo nastala med akcijo prijave uporabnika v modulu »Prijava«. Ob inicializaciji bo modul »Trgovec« registriral in naložil zgolj uporabniški pogled »NavigationTrgovec« v regijo »MenuRegion«. Ostali pogledi se bodo nalagali v regijo »MainRegion« ob uporabniški zahtevi. Uporabniška zahteva za nalaganje pogledov bo določena z navigacijo. Navigacija bo implementirana z menijsko vrstico, kjer bo klik na določen menijski element sprožil nalaganje izbranega pogleda v regijo »MainRegion«.



Slika 16: Sekvenčni diagram inicializacije modula »Trgovec«

Uporabniški vmesnik/pogled »NavigationTrgovec«:

Uporabniški pogled »NavigationTrgovec« skrbi za navigacijo med delovnimi pogledi. Sestavljen je po principu, ki ga določa načrtovalski vzorec MVVM. »View« oziroma grafični uporabniški vmesnik vsebuje menijsko vrstico ter razred »NavigationTrgovec.cs«, ki določa kodo v ozadju. Namenjen je inicializaciji poslovnega razreda »NavigationTrgovecViewModel«, ki določa predstavitveno logiko danega uporabniškega vmesnika.

Razred »NavigationTrgovecViewModel« omogoča navigacijo med delovnimi uporabniškimi pogledi, med katere sodijo »ArtikliView«, »StrankeView« in »ProdajaView«. Proces navigacije določa način, po katerem se na uporabniško zahtevo v regijo »MainRegion« naloži ustrezni delovni pogled. Prav zaradi tega razred »NavigationTrgovecViewModel« vsebuje Microsoft PRISM komponento, imenovano RegionManager s katero lahko naloži zahtevan uporabniški pogled v regijo »MainRegion«.

Uporabniški vmesnik/pogled »ArtikliView«:

Uporabniški vmesnik »ArtikliView« uporabnikom omogoča pregledovanje in iskanje artiklov.
Poleg tega vsebuje povezavo do modalnega okna, ki omogoča dodajanje, brisanje in urejanje artiklov. Sestavljen je po principu, ki ga določa načrtovalski vzorec MVVM. »View« oziroma grafični uporabniški vmesnik prikazuje delovno površino, ki uporabnikom omogoča pregledovanje in iskanje artiklov. Združuje razred »ArtikliView.cs«, ki predstavlja kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »ArtikliViewModel«.

Razred »ArtikliViewModel« določa predstavitveno in poslovno logiko upravljanja z artikli. Proces upravljanja z artikli potrebuje dostop do WCF spletnega servisa in WCF storitev. Prav zaradi tega dani razred potrebuje komponento, ki je del tehnologije WCF RIA Services ter se imenuje ServiceReference. Komponenta ServiceReference omogoča klientu dostop do WCF spletnega servisa ter do WCF storitev.

Uporabniški vmesnik/pogled »StrankeView«:

Uporabniški vmesnik »StrankeView« uporabnikom omogoča pregledovanje in iskanje strank. Poleg tega vsebuje povezavo do modalnega okna, ki omogoča dodajanje, brisanje in urejanje strank. Sestavljen je po principu, ki ga določa načrtovalski vzorec MVVM. »View« oziroma grafični uporabniški vmesnik prikazuje delovno površino, ki

uporabnikom omogoča pregledovanje in iskanje strank. Združuje razred »StrankeView.cs«, ki predstavlja kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »StrankeViewModel«.

Razred »StrankeViewModel« določa predstavitveno in poslovno logiko upravljanja z artikli. Proces upravljanja s strankami potrebuje dostop do WCF spletnega servisa in WCF storitev. Prav zaradi tega dani razred potrebuje komponento, ki je del tehnologije WCF RIA Services ter se imenuje ServiceReference. Komponenta ServiceReference omogoča klientu dostop do WCF spletnega servisa ter do WCF storitev.

Uporabniški vmesnik/pogled »ProdajaView«:

Uporabniški vmesnik »ProdajaView« vsebuje blagajno poslovne aplikacije STrgovina ter povezavo do modalnega okna, ki omogoča pregledovanje in storniranje nakupov. Blagajna v splošnem uporabniku omogoča ustvarjanje novih nakupov ter tiskanje računov. Uporabniški vmesnik »ProdajaView« je sestavljen po principu, ki ga določa načrtovalski vzorec MVVM. »View« oziroma grafični uporabniški vmesnik definira izgled blagajne ter vsebuje razred »ProdajaView.cs«, ki določa kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »ProdajaViewModel«.

Razred »ProdajaViewModel« definira predstavitveno in poslovno logiko blagajne. Blagajna je osrednja funkcionalnost poslovne aplikacije STrgovina. Za delovanje potrebuje dostop do WCF spletnega servisa in WCF storitev. Prav zaradi tega dani razred potrebuje komponento, ki je del tehnologije WCF RIA Services ter se imenuje ServiceReference. Komponenta ServiceReference omogoča klientu dostop do WCF spletnega servisa ter do WCF storitev.

6.2.3.1 Opis glavnih aktivnosti modula »Trgovec«

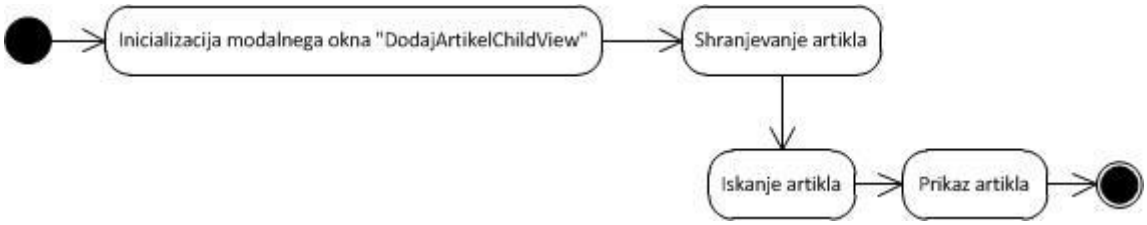
Modul »Trgovec« vsebuje dva osnovna pregleda, delovni pogled, pogled za navigacijo in tri modalna okna. Glavne aktivnosti, ki jih modul »Trgovec« določa, so: aktivnost pregledovanja ali iskanja artiklov, aktivnost dodajanja, urejanja ali brisanja obstoječega artikla, aktivnost pregledovanja in iskanja strank, aktivnost dodajanja, urejanja ali brisanja obstoječe stranke, aktivnost dodajanja novega nakupa ter tiskanja računa in aktivnost pregledovanja ali storniranja že obstoječega nakupa.

V zaključni nalogi so opisane zgolj tri glavne aktivnosti, ki pa določajo osnovo vseh aktivnosti v modulu »Trgovec«.

6.2.3.1.1 Aktivnost dodajanja novega artikla

Udeleženi razredi ali storitve:

- uporabniški pogled »ArtikliView«;
- poslovni razred »ArtikliViewModel«;
- modalno okno »DodajArtikelChildView«;
- poslovni razred »DodajArtikelChildViewModel«;
- WCF spletna storitev »SaveArtikel«;
- WCF spletna storitev »VrniArtikel«.



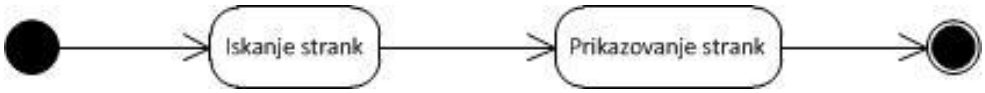
Slika 17: UML diagram aktivnosti za proces dodajana novega artikla

Pri procesu dodajanja novega artikla bo razred »ArtikliViewModel« inicializiral modalno okno »DodajArtikelChildView«. Slednje je implementirano po principu načrtovalskega vzorca MVVM. »View« oziroma grafični uporabniški vmesnik vsebuje razred »DodajArtikelChildView.cs«, ki predstavlja kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »DodajArtikelChildViewModel«. Slednji bo za dodajanje novega artikla dostopal do WCF spletne storitve, imenovane »SaveArtikel«. Ta storitev omogoča vnos novega artikla v tabelo Artikel ter identifikator vnesenega artikla vrne klientu. Klient pridobi identifikator vnesenega artikla in nato dostopa do spletne storitve, imenovane »VrniArtikel«. Slednja omogoča iskanje artiklov v podatkovni bazi s točno določenim identifikatorjem. V primeru, da artikel z določenim identifikatorjem obstaja, dana storitev podatke o artiklu shrani v objekt Artikel in le-tega vrne klientu. Klient pridobi objekt Artikel ter ga posreduje grafičnemu uporabniškemu vmesniku »DodajArtikelChildView«, ki podatke prikaže uporabniku.

6.2.3.1.2 Aktivnost pregledovanja in iskanja strank

Udeleženi razredi ali storitve:

- uporabniški pogled »StrankeView«;
- poslovni razred »StrankeViewModel«;
- WCF spletna storitev »VrniStranka«.



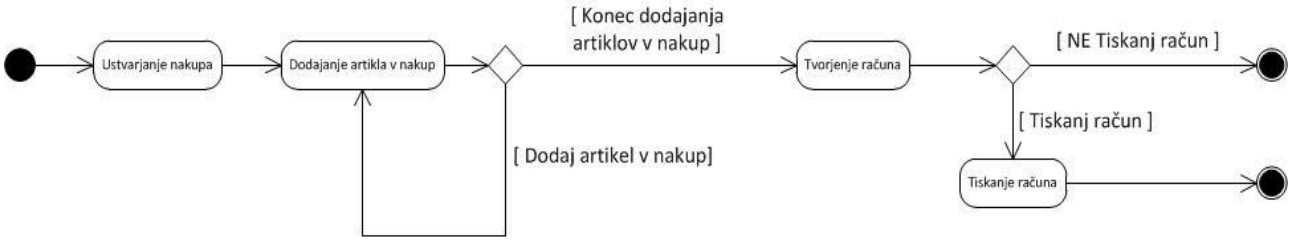
Slika 18: UML diagram aktivnosti za proces pregledovanja in iskanja strank

Pri procesu pregledovanja in iskanja določene stranke bo razred »StrankeViewModel« dostopal do WCF storitve, imenovane »VrniStranka«. Slednja omogoča iskanje strank v podatkovni bazi po imenu, ki mora vsebovati niz, ki ga uporabnik vpiše v iskalno polje. Če v podatkovni bazi obstajajo iskane stranke, dana storitev podatke o strankah shrani v seznam strank in le-tega vrne klientu. Klient pridobi seznam strank in ga posreduje grafičnemu uporabniškemu vmesniku »StrankeView«, ki podatke prikaže uporabniku.

6.2.3.1.3 Aktivnost dodajanja novega nakupa in tiskanja računa

Udeleženi razredi ali storitve:

- uporabniški pogled »ProdajaView«;
- poslovni razred »ProdajaViewModel«;
- WCF spletna storitev »UstvariNakup«;
- WCF spletna storitev »DodajArtikelVNakup«.

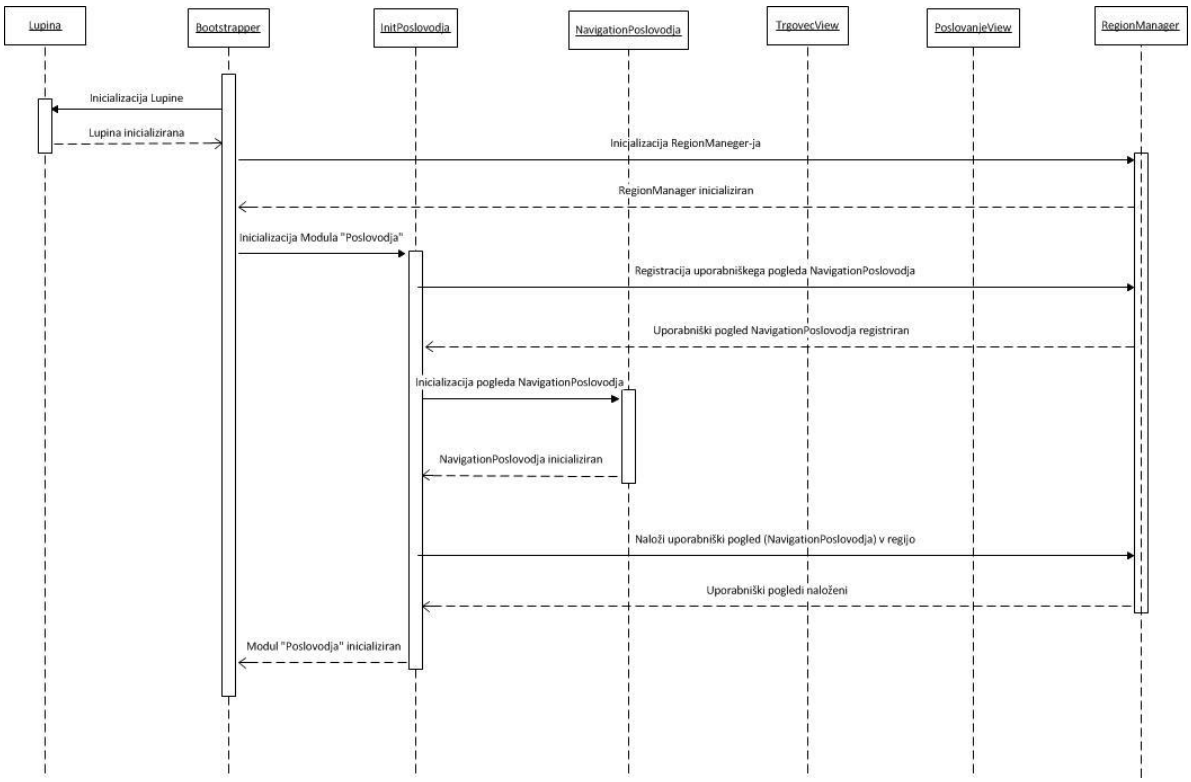


Slika 19: UML diagram aktivnosti za proces dodajanja novega nakupa in tiskanja računa

Pri procesu ustvarjanja novega nakupa bo razred »ProdajaViewModel« dostopal do WCF storitve imenovane »UstvariNakup«. Ta storitev v podatkovni bazi, natančneje v tabeli Nakup ustvari novo vrstico ter identifikator dane vrstice vrne klientu. Klient na drugi strani pridobi odgovor ter ga posreduje grafičnemu uporabniškemu vmesniku, ki ga prikaže. Po uspešno ustvarjenem nakupu razred »ProdajaViewModel« dodaja artikle v nakup, tako da dostopa do metode, imenovane »DodajArtikelVNakup«. Slednja omogoča dodajanje artiklov v nakup ter kot parametre sprejme identifikator nakupa in identifikator ter količino izbranega artikla. Po končanem vnosu artiklov v nakup razred »ProdajaViewModel« pokliče privatno metodo »ZakljuciNakup«, ki tvori račun in omogoči njegovo tiskanje.

6.2.4 Modul »Poslovodja« - Prism.STrgovina.Moduli.Poslovodja

Modul »Poslovodja« bo uporabnikom omogočal dodajanje, brisanje in iskanje trgovcev ter statistični pregled nad poslovanjem trgovine. Vseboval bo dva osnovna pregleda, pogled za navigacijo in dve modalni okni. Med preglede sodita uporabniški pogled »TrgovecView«, ki bo omogočal pregledovanje in iskanje trgovcev, ter uporabniški pogled »PoslovanjeView«, ki bo omogočal pregledovanje in iskanje strank. Uporabniški pogled za navigacijo »NavigationPoslovodja« bo vseboval menijsko vrstico. Med modalni okni pa sodita modalno okno za dodajanje in brisanje trgovcev, imenovano »DodajTrgovecChildView« ter modalno okno, imenovano »StrankaNakupiChildView«, ki omogoča statistični pregled nakupov za posamezno stranko.
Modul »Poslovodja« se bo inicializiral na zahtevo. Zahteva za inicializacijo bo nastala med akcijo prijave uporabnika v modulu »Prijava«. Ob inicializaciji bo modul »Poslovodja« registriral in naložil zgolj uporabniški pogled »NavigationPoslovodja« v regijo »MenuRegion«. Ostali pogledi se bodo nalagali v regijo »MainRegion« ob uporabniški zahtevi. Uporabniška zahteva za nalaganje pogledov bo določena z navigacijo. Navigacija bo implementirana z menijsko vrstico, kjer bo klik na določen menijski element sprožil nalaganje izbranega pogleda v regijo »MainRegion«.



Slika 20: Sekvenčni diagram inicializacije modula »Poslovodja«

Uporabniški vmesnik/pogled »NavigationPoslovodja«:

Uporabniški pogled »NavigationPoslovodja« skrbi za navigacijo med delovnimi pogledi. Sestavljen je po principu, ki ga določa načrtovalski vzorec MVVM. »View« oziroma grafični uporabniški vmesnik vsebuje menijsko vrstico ter razred »NavigationPoslovodja.cs«, ki določa kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »NavigationPoslovodjaViewModel«, ki določa predstavitveno logiko danega uporabniškega vmesnika.

Razred »NavigationPoslovodjaViewModel« omogoča navigacijo med delovnimi uporabniškimi pogledi, med katere sodita »TrgovecView« in »PoslovanjeView«. Proces navigacije določa način, po katerem se na uporabniško zahtevo v regijo »MainRegion« naloži ustrezni delovni pogled. Prav zaradi tega razred »NavigationPoslovodjaViewModel« vsebuje Microsoft PRISM komponento, imenovano RegionManager, s katero lahko naloži zahtevan uporabniški pogled v regijo »MainRegion«.

Uporabniški vmesnik/pogled »TrgovecView«:

Uporabniški vmesnik »TrgovecView« uporabnikom omogoča pregledovanje in iskanje trgovcev.
Poleg tega vsebuje povezavo do modalnega okna, ki omogoča dodajanje in brisanje trgovcev. Sestavljen je po principu, ki ga določa načrtovalski vzorec MVVM. »View« oziroma grafični uporabniški vmesnik prikazuje delovno površino, ki uporabnikom omogoča pregledovanje in iskanje trgovcev. Združuje razred »TrgovecView.cs«, ki predstavlja kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »TrgovecViewModel«.

Razred »TrgovecViewModel« določa predstavitveno in poslovno logiko upravljanja s trgovci. Proces upravljanja s trgovci potrebuje dostop do WCF spletnega servisa in WCF storitev. Prav zaradi tega dani razred potrebuje komponento, ki je del tehnologije WCF RIA Services ter se imenuje ServiceReference. Komponenta ServiceReference omogoča klientu dostop do WCF spletnega servisa ter do WCF storitev.

Uporabniški vmesnik/pogled »PoslovanjeView«:

Uporabniški vmesnik »PoslovanjeView« uporabnikom omogoča pregledovanje in iskanje strank.
Poleg tega vsebuje povezavo do modalnega okna, ki omogoča statistični pregled nakupov

za posamezno stranko. Sestavljen je po principu, ki ga določa načrtovalski vzorec MVVM. »View« oziroma grafični uporabniški vmesnik prikazuje delovno površino, ki uporabnikom omogoča pregledovanje in iskanje strank. Združuje razred »PoslovanjeView.cs«, ki predstavlja kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »PoslovanjeViewModel«.

Razred »PoslovanjeViewModel« določa predstavitveno in poslovno logiko upravljanja s strankami. Proces upravljanja s strankami potrebuje dostop do WCF spletnega servisa in WCF storitev. Prav zaradi tega dani razred potrebuje komponento, ki je del tehnologije WCF RIA Services ter se imenuje ServiceReference. Komponenta ServiceReference omogoča klientu dostop do WCF spletnega servisa ter do WCF storitev.

6.2.4.1 Opis glavnih aktivnosti modula »Poslovodja«

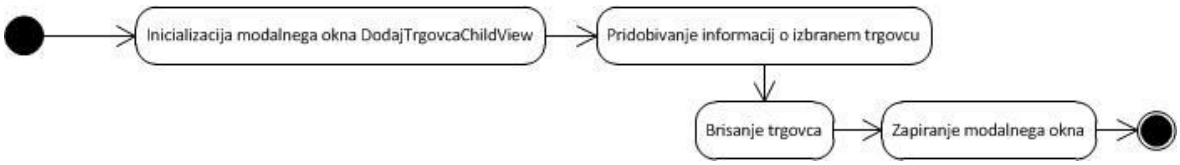
Modul »Poslovodja« vsebuje dva osnovna pregleda, pogled za navigacijo in dve modalni okni. Glavne aktivnosti, ki jih modul »Poslovodja« določa, so: aktivnost pregledovanja ali iskanja trgovcev, aktivnost dodajanja in brisanja trgovcev, aktivnost pregledovanja in iskanja strank, ter aktivnost pregledovanja statističnih podatkov o nakupih za posamezne stranke.

V zaključni nalogi sta opisani zgolj dve glavni aktivnosti, ki pa določata osnovo vseh aktivnosti v modulu »Poslovodja«.

6.2.4.1.1 Aktivnost brisanja obstoječega trgovca

Udeleženi razredi ali storitve:

- uporabniški pogled »TrgovecView«;
- poslovni razred »TrgovecViewModel«;
- modalno okno »DodajTrgovcaChildView«;
- poslovni razred »DodajTrgovcaChildViewModel«;
- WCF spletna storitev »VrniTrgovecID«;
- WCF spletna storitev »DeleteTrgovec«.



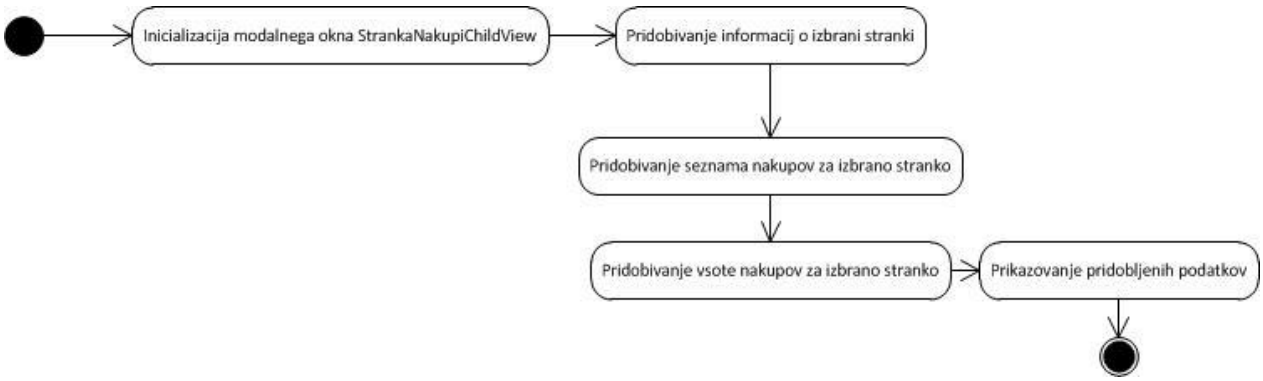
Slika 21: UML diagram aktivnosti za proces brisanja obstoječega trgovca

Pri procesu brisanja obstoječega trgovca bo razred »TrgovecViewModel« inicializiral modalno okno »DodajTrgovecChildView« skupaj z informacijo o identifikatorju trgovca. Modalno okno je implementirano po principu načrtovalskega vzorca MVVM. »View« oziroma grafični uporabniški vmesnik vsebuje razred »DodajTrgovecChildView.cs«, ki predstavlja kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »DodajTrgovecChildViewModel«. Dani razred bo uporabil spletni servis »VrniTrgovecID«, preko katerega bo pridobil vse potrebne informacije o izbranem trgovcu. Storitev »VrniTrgovecID« omogoča iskanje trgovca v podatkovni bazi s točno določenim identifikatorjem. Slednjega bo razred »DodajTrgovecChildViewModel« pridobil kot parameter pri inicializaciji. Po uspešno pridobljenem trgovcu bo sledil korak brisanja danega trgovca. Razred »DodajTrgovecChildViewModel« bo za brisanje obstoječega trgovca uporabil WCF spletno storitev, imenovano »DeleteTrgovec«. Ta storitev omogoča brisanje obstoječega trgovca iz tabele Trgovec. Po uspešno izbrisanem trgovcu razred »DodajTrgovecChildViewModel« pokliče privatno metodo »Close«, ki zapre obstoječe modalno okno.

6.2.4.1.2 Aktivnost pregledovanja statističnih podatkov o nakupih za stranke

Udeleženi razredi ali storitve:

- uporabniški pogled »PoslovanjeView«;
- poslovni razred »PoslovanjeViewModel«;
- modalno okno »StrankaNakupiChildView«;
- poslovni razred »StrankaNakupiChildViewModel«;
- WCF spletna storitev »VrniStrankaID«;
- WCF spletna storitev »VrniNakupStranka«;
- WCF spletna storitev »VrniVsotoNakupStranka«.



Slika 22: UML diagram aktivnosti pregledovanja statističnih podatkov o nakupih

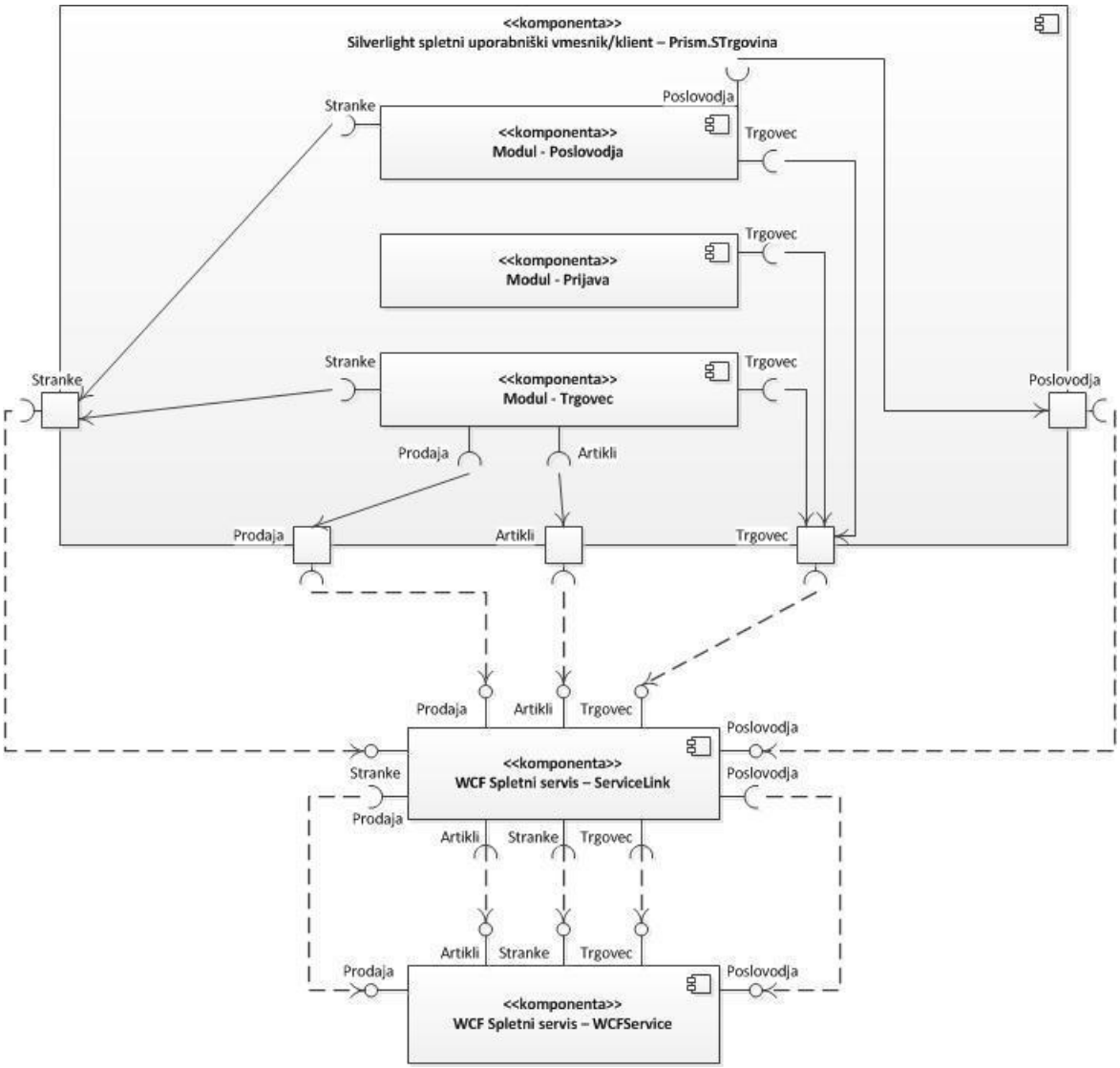
Pri procesu pregledovanja statističnih podatkov o nakupih posameznih strank bo razred »PoslovanjeViewModel« inicializiral modalno okno »StrankaNakupiChildView« skupaj z informacijo o identifikatorju izbrane stranke. Modalno okno je implementirano po principu načrtovalskega vzorca MVVM. »View« oziroma grafični uporabniški vmesnik vsebuje razred »StrankaNakupiChildView.cs«, ki predstavlja kodo v ozadju. Le-ta je namenjen inicializaciji poslovnega razreda »StrankaNakupiChildViewModel«. Dani razred bo uporabil spletni servis »VrniStrankaID«, preko katerega bo pridobil vse potrebne informacije o izbrani stranki. Storitev »VrniStrankaID« omogoča iskanje stranke v podatkovni bazi s točno določenim identifikatorjem. Slednjega bo razred »StrankaNakupiChildViewModel« pridobil kot parameter pri inicializaciji. Po uspešno pridobljeni stranki bo sledil korak pridobivanja statističnih podatkov o nakupih za dano stranko. Razred »StrankaNakupiChildViewModel« bo za pridobitev statističnih podatkov uporabil WCF spletni storitvi, imenovani »VrniNakupStranka« in »VrniVsotoNakupStranka«. Storitev »VrniNakupStranka« omogoča iskanje nakupov v podatkovni bazi po identifikatorju stranke in izbranem časovnem obdobju. V primeru, da v podatkovni bazi obstajajo iskani nakupi, dana storitev podatke o nakupih shrani v seznam nakupov in le-tega vrne klientu. Klient pridobi seznam nakupov ter ga posreduje grafičnemu uporabniškemu vmesniku »StrankaNakupiChildView«, ki podatke prikaže uporabniku. Storitev »VrniVsotoNakupStranka« pa omogoča izračunavanje vsote nakupov za določeno stranko in v določenem časovnem obdobju. Storitev klientu vrne vsoto nakupov v obliki decimalnega števila, ki ga klient posreduje grafičnemu uporabniškemu vmesniku.

6.3 Spletni strežnik IIS – Prism.STrgovina.Web

Poslovna aplikacija STrgovina bo za gostovanje spletnega uporabniškega vmesnika uporabljala Microsoftov spletni strežnik IIS (Internet Information Services), ker omogoča gostovanje Microsoft Silverlight spletnih aplikacij. Vseboval bo WCF spletni servis ter tri pomembne datoteke. WCF spletni servis, imenovan »ServiceLink«, bo omogočal spletnemu uporabniškemu vmesniku dostop do WCF spletnih storitev, ki jih ponuja WCF spletni servis »WCFService«. Med datoteke sodijo STrgovina.html, STrgovina.aspx in STrgovina.xap, ki se ob vsakem zagonu poslovne aplikacije na novo kreirajo.

WCF spletni servis »ServiceLink« bo dostopen na spletnem naslovu »http://84.255.226.182/ Prism.STrgovina/ServiceLink.svc« ter bo spletnemu uporabniškemu vmesniku omogočal dostop do WCF spletnih storitev, ki jih določa WCF spletni servis, imenovan »WCFService«. Prav zaradi tega dani spletni servis potrebuje komponento, ki je del tehnologije WCF RIA Services ter se imenuje ServiceReference.

Komponenta ServiceReference omogoča klientu dostop do WCF spletnega servisa ter do WCF storitev.

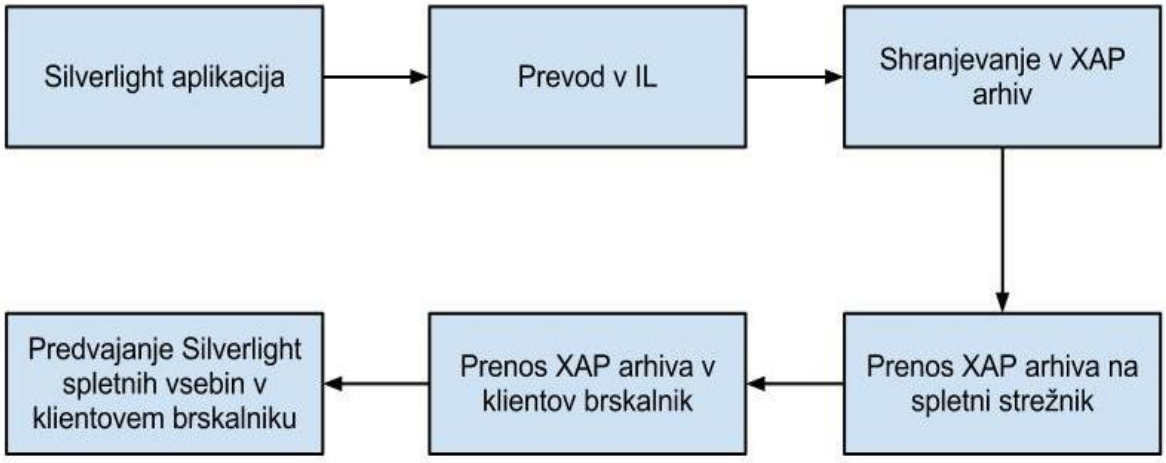


Slika 23: UML komponentni diagram poslovne aplikacije STrgovina

Datoteki STrgovina.html in STrgovina.aspx vsebujeta ogrodje za postavitev Silverlight spletnega uporabniškega vmesnika znotraj spletne strani, ki jo prikaže spletni brskalnik.

Datoteka STrgovina.xap predstavlja ZIP arhiv, ki združuje vse datoteke, ki so nujno potrebne za delovanje Silverlight spletnega uporabniškega vmesnika. Pri poganjanju Silverlight spletnih aplikacij v fazi izvedbe se Silverlight spletni uporabniški vmesnik prevede in shrani v xap arhiv, ki se naloži na spletni strežnik. Ko spletni brskalnik dostopi do danega spletnega naslova, se xap arhiv prenese v klientov brskalnik. Klientov brskalnik nato poskrbi za pravilno predvajanje Silverlight vsebine. Edini pogoj pri tem je, da ima klientov brskalnik nameščen Silverlight vtičnik, ki omogoča poganjanje Silverlight

spletnih aplikacij [3].



Slika 24: Koraki pri poganjanju Silverlight spletnih aplikacij v fazi razvoja

6.4 WCF Spletni servis – WCFService

Plast pristopa k podatkom aplikacije STrgovina bo vsebovana v WCF spletnem servisu, imenovanem »WCFService«. Spletni servis predstavlja metodo komunikacije med dvema elektronskima napravama preko svetovnega spleta. Določa programsko funkcijo, ki je dostopna na določenem spletnem naslovu.

WCF spletni servis bo dostopen na spletnem naslovu »http://84.255.226.182/Prism.STrgovina.WS/ /WCFService.svc«. Vseboval bo pet skupin WCF spletnih storitev, posebne poslovne objekte, imenovane Upravitelji, ter ORM tehnologijo, imenovano »ADO.net Entity Framework«. Slednja skrbi za dostop in upravljanje s podatkovno bazo. Za pravilno upravljanje z dano tehnologijo pa skrbijo poslovni razredi, imenovani Upravitelji. Med slednje sodijo razred UpraviteljArtikel, razred UpraviteljTrgovec, razred UpraviteljStranka ter razred UpraviteljPoslovanja.

6.4.1 WCF spletne storitve

WCF spletne storitve so določene z dvema razredoma. Prvi razred predstavlja vmesnik in se imenuje IWCFService, drugi razred pa implementira dani vmesnik ter se imenuje WCFService. Razred WCFService vsebuje pet sklopov storitev. Med slednje sodijo storitve, namenjene upravljanju z artikli, storitve, namenjene upravljanju s strankami, storitve, namenjene upravljanju s trgovci, storitve, namenjene upravljanju z blagajno, ter storitve, namenjene upravljanju s statističnimi podatki o poslovanju trgovine. Vsaka storitev vsebuje dostop do določenega Upravitelja, ki vsebuje poslovno logiko upravljanja

z ORM tehnologijo »ADO.net Entity Framework«.

Tabela 1: WCF spletne storitve vsebovane v razredu WCFService

Artikel	**Poslovanje**
- ArtikliList(); - VrniArtikel(); - VrniArtikelID(); - SaveArtikel(); - DeleteArtikel(); - GetNewArtikel();	- UstvariNakup(); - DodajArtikelVNakup(); - VrniDolocenNakup(); - PopraviNakup(); - VrniNakup(); - VrniInfoNakup(); - VrniListoNakupov(); - StornirajNakup(); - VrniVsotoNakupa();
Stranka	**Poslovodja**
- StrankaList(); - VrniStranka(); - VrniStrankaID(); - SaveStranka(); - DeleteStranka(); - GetNewStranka();	- VrniNakupStranka(); - VrniVsotoNakupStranka()
Trgovec	
- TrgovecList(); - VrniTrgovec(); - VrniTrgovecID(); - SaveTrgovec(); - DeleteTrgovec(); - GetNewTrgovec(); - TrgovecPrijava();	

6.4.2 Uporaba WCF spletnih storitev na primeru storitve VrniArtikelID

Storitev »VrniArtikelID« omogoča iskanje artikla v podatkovni bazi po točno določenem identifikatorju. Prav zaradi tega storitev kot parameter prejme identifikator, ki je tipa celo število, vrne pa objekt Artikel, ki ustreza iskanemu pogoju. Storitev pri iskanju artikla pokliče razred UpraviteljArtikel ter njegovo metodo, imenovano »VrniArtikelID«. UpraviteljArtikel nato uporabi ORM tehnologijo »ADO.net Entity Framework«, s katero se poveže na podatkovno bazo ter iz tabele Artikel pridobi iskani rezultat.

6.4.3 Poslovni razredi, imenovani Upravitelji

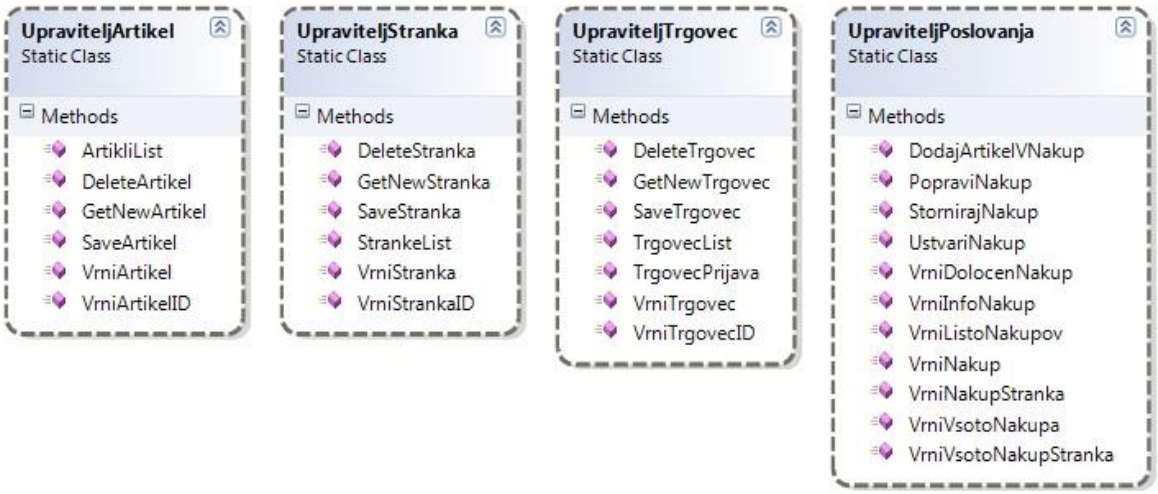
Poslovni razredi, imenovani Upravitelji, definirajo poslovno logiko in formalna pravila uporabe ORM tehnologije, imenovane »ADO.net Entity Framework«, ki skrbi za dostop in upravljanje s podatkovno bazo. Glavni namen uporabe sledečih poslovnih razredov je v razumljivejši programski kodi. Implementacijo metod, ki jih vsebujejo Upravitelji, bi lahko združili v WCF spletne storitve, vendar bi se kompleksnost razreda, ki implementira WCF spletni servis, povečala. Posledično bi bilo vzdrževanje takega razreda zahtevnejše. Z Upravitelji smo dosegli, da se kompleksnost WCF servisa porazdeli, in tako omogočili enostavnejši razvoj in kasnejše vzdrževanje. Med poslovne razrede, imenovane Upravitelji, sodijo UpraviteljArtikel, UpraviteljStranka, UpraviteljTrgovec in UpraviteljPoslovanja.

UpraviteljArtikel je namenjen upravljanju s poslovnim objektom Artikel. Omogoča dodajanje, urejanje, brisanje in poizvedovanje po tabeli Artikel.

UpraviteljStranka je namenjen upravljanju s poslovnim objektom Stranka. Omogoča dodajanje, urejanje, brisanje in poizvedovanje po tabeli Stranka.

UpraviteljTrgovec je namenjen upravljanju s poslovnim objektom Trgovec. Omogoča dodajanje, urejanje, brisanje in poizvedovanje po tabeli Trgovec.

UpraviteljPoslovanja je namenjen upravljanju s poslovnima objektoma Nakup in Nakup_Artikel. Omogoča razne poizvedovalne in vnosne metode, ki delujejo nad tabelama Nakup in Nakup_Artikel.



Slika 25: UML razredni diagram poslovnih objektom imenovanih Upravitelji

6.5 MS SQL podatkovna baza – Prism.STrgovina.DB

MS SQL ali »Microsoft SQL Server« je sistem za upravljanje z relacijskimi podatkovnimi bazami. Poslovna aplikacija STrgovina uporablja MS SQL relacijsko podatkovno bazo, imenovano Prism.STrgovina.DB. Podatkovna baza bo vsebovala tabele: Artikel, Stranka, Trgovec, Nakup in Nakup_Artikel.

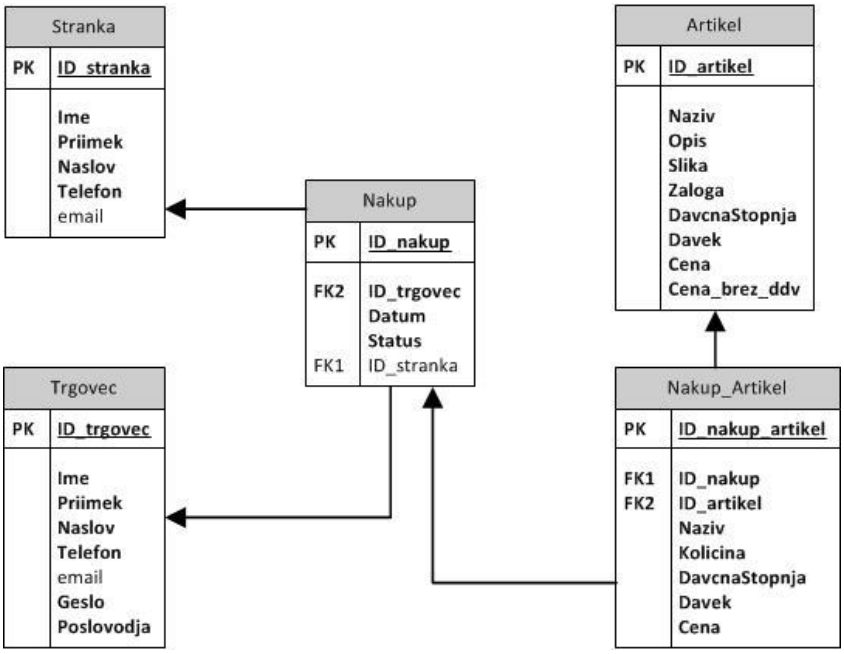
Tabela »Stranka«: hrani podatke o strankah. Identifikator in primarni ključ dane tabele je ID_stranka.

Tabela »Trgovec«: hrani podatke o trgovcih. Identifikator in primarni ključ dane tabele je ID_trgovec.

Tabela »Artikel«: hrani podatke o artiklih. Identifikator in primarni ključ dane tabele je ID_artikel.

Tabela »Nakup«: hrani podatke o nakupih. Identifikator in primarni ključ dane tabele je ID_nakup. Poleg tega dana tabela vsebuje še dva tuja ključa, in sicer ID_trgovec ter ID_stranka.

Tabela »Nakup_Artikel«: predstavlja razmerje med tabelama »Nakup« in »Artikel«. Identifikator in primarni ključ dane tabele je ID_nakup_artikel. Poleg tega vsebuje še dva tuja ključa, in sicer ID_nakup ter ID_artikel.



Slika 26: UML razredni diagram podatkovne baze Prism.STrgovina.DB

7 IZVEDBA

Izvedba predstavlja šesto fazo razvoja programskih produktov. Razvijalci na osnovi načrta programa implementirajo programski produkt. Predstavlja eno izmed najzahtevnejših faz razvoja, saj napake storjene v predhodnem načrtovanju, krojijo sam razvoj. Pomembno je, da se v fazah načrtovanja definirajo realne zahteve in časovne omejitve, ki omogočajo razvijalcem oziroma programerjem uspešno implementacijo programskega produkta.

V tem poglavju so predstavljeni praktični primeri implementacije poslovne aplikacije STrgovina ter problemi, ki so se pri tem pojavljali.

7.1 Spletni uporabniški vmesnik – Prism.STrgovina

Spletni uporabniški vmesnik aplikacije STrgovina predstavlja vmesnik med uporabniki in sistemom. Namenjen je grafičnemu upravljanju s poslovno pomembnimi podatki, kot so podatki o artiklih, strankah, trgovcih in nakupih. Spletni uporabniški vmesnik bo uporabnikom omogočal dodajanje, urejanje, pregledovanje in brisanje artiklov, strank in trgovcev ter upravljanje z blagajno.

Implementacija lupine aplikacije:

Lupina aplikacije določa osnovno postavitev oziroma izgled. Deklarirana je v formatu XAML, ki ga uporablja aplikacijsko ogrodje Microsoft Silverlight. Vsebuje tri regije oziroma področja, ki so določena s pomočjo komponente, imenovane RegionManager, ki je del okolja Microsoft PRISM.

Način uporabe komponente RegionManager za določanje regij znotraj lupine aplikacije STrgovina (primer za regijo »MenuRegion«):

```
<ItemsControl prism:RegionManager.RegionName="MenuRegion"
                    Grid.Row="1"
                    Grid.Column="0"
                    HorizontalAlignment="Stretch"
                    VerticalAlignment="Stretch"
                    Margin="10, 4, 0, 0">

</ItemsControl>
```

7.1.1 Razred Bootstrapper

Bootstrapper je razred, ki povezuje aplikacijo z Microsoft PRISM programskimi knjižnicami. Med te sodijo MEF ali Unity programske knjižnice, ki omogočajo razvoj razširljivih aplikacij. V primeru poslovne aplikacije STrgovine smo uporabili MEF programsko knjižnico.

Primer dedovanja razreda MefBootstrapper:

```
public class Bootstrapper : MefBootstrapper
{
}
```

Glavne metode, ki jih vsebuje razred Bootstrapper poslovne aplikacije STrgovina:

1. Metoda CreateShell() pridobi instanco Lupine (ang. Shell) aplikacije STrgovina.

```
protected override DependencyObject CreateShell()
{
        _regionManager = this.Container.GetExportedValue<IRegionManager>();
        return this.Container.GetExportedValue<Shell>();
}
```

2. Metoda InitializeShell inicializira Lupino aplikacije STrgovina.

```
protected override void InitializeShell()
{
    base.InitializeShell();
    Application.Current.RootVisual = (UIElement)this.Shell;
}
```

3. Metoda ConfigureAggregateCatalog kreira komponento katalog modulov, katero aplikacija uporabi za nalaganje modulov.

```
protected override void ConfigureAggregateCatalog()
{
        base.ConfigureAggregateCatalog();

        this.AggregateCatalog.Catalogs.Add(new TypeCatalog(new[]
        {
                typeof(Shell)
        }));

       this.AggregateCatalog.Catalogs.Add(newAssemblyCatalog(typeof(InitPrijava).Assembly));
       this.AggregateCatalog.Catalogs.Add(newAssemblyCatalog(typeof(InitTrgovec).Assembly));
       this.AggregateCatalog.Catalogs.Add(newAssemblyCatalog(typeof(InitPoslovodja).Assembly));
}
```

7.1.2 »Model – View – ViewModel« načrtovalski vzorec

»Model–View–ViewModel« je načrtovalski vzorec, ki loči poslovno in predstavitveno logiko aplikacije od grafičnega uporabniškega vmesnika. V poslovni aplikaciji STrgovina so vsi pogledi, ki jih moduli vsebujejo implementirani s pomočjo opisanega načrtovalskega vzorca.

V nadaljevanju je predstavljen način uporabe načrtovalskega vzorca MVVM na primeru uporabniškega pogleda »TrgovecView«, ki ga vsebuje modul »Poslovodja«.

Uporabniški pogled »TrgovecView« je sestavljen iz dveh delov. Prvi del predstavlja grafični uporabniški vmesnik »TrgovecView.xaml« ki je zapisan v XAML formatu, in razred »TrgovecView.cs«, ki določa kodo v ozadju. Drugi del predstavlja razred »TrgovecViewModel.cs«, ki določa komponento »View–Model«.

1. XAML notacija grafične komponente »gumb« in povezovanja na akcijo »IsciTrgovecCommand«, ki je implementirana v razredu »TrgovecViewModel.cs«:

```
<Button Content="Išči" IsEnabled="{Binding Enabled, Mode=TwoWay}" Height="23" HorizontalAlign-
ment="Left" Command="{Binding IsciTrgovecCommand}" Margin="468,78,0,0" Name="button1" Vertica-
lAlignment="Top" Width="75">
        <Button.Effect>
        <DropShadowEffect Color="Gray"
                          Opacity=".50"
                ShadowDepth="8">
        </DropShadowEffect>
        </Button.Effect>
</Button>
```

2. Primer inicializacije in povezave grafičnega uporabniškega vmesnika »TrgovecView.xaml« s predstavitveno logiko, ki se nahaja v razredu »TrgovecViewModel.cs« - sledeča programska koda je implementirana v razredu »TrgovecView.cs«, ki predstavlja kodo v ozadju:

```
public TrgovecView(IRegionManager regionManager)
{
        InitializeComponent();
        TrgovecViewModel vm = new TrgovecViewModel(regionManager);
        this.DataContext = vm;
}
```

3. Primer implementacije akcije »IsciTrgovecCommand« v razredu »TrgovecViewModel.cs«:

```
//Deklaracija komponente DelegateCommand --> akcije IsciTrgovca
public DelegateCommand IsciTrgovecCommand { get; set; }

//Kosntruktor
public TrgovecViewModel(IRegionManager _regionManager)
{
        //Inicializacija akcije IsciTrgovecCommand
        IsciTrgovecCommand = new DelegateCommand(IsciTrgovec, () => true);
}

//Implementacija metode za iskanje trgovca
private void IsciTrgovec()
{
        //Vsebina metode
}
```

7.2 Moduli

Modul je zaokrožen paket funkcionalnosti, ki je lahko neodvisno načrtovan, implementiran in testiran. V večini primerov se moduli razvijajo in vzdržujejo v ločenih razvojnih skupinah. Module lahko uporabimo za predstavitev specifične poslovne funkcionalnosti, saj združujejo vso potrebno poslovno logiko, uporabniške vmesnike in podatkovne modele [2].
Poslovna aplikacija STrgovina vsebuje tri module, med katere sodijo modul »Prijava«, modul »Trgovec« in modul »Poslovodja«.

7.2.1 Inicializacija modulov

Modul je lahko inicializiran ob zagonu aplikacije ali pa na zahtevo. V aplikaciji STrgovina se modul »Prijava« inicializira ob zagonu, modula »Trgovec« in »Poslovodja« pa na zahtevo. Vsak modul vsebuje razred, ki je namenjen inicializaciji. Le-ta mora implementirati vmesnik IModule.

Primer inicializacijskega razreda modula »Prijava«, ki omogoča inicializacijo ob zagonu:

```
[ModuleExport(typeof(InitPrijava), InitializationMode=InitializationMode.WhenAvailable)]
public class InitPrijava : IModule
{
    public void Initialize()
    {
                //Implementacija …
     }
}
```

Opomba:

- zastavica »InitializationMode=InitializationMode.WhenAvailable« omogoča inicializacijo modula ob zagonu.

Primer inicializacijskega razreda modula »Trgovec«, ki omogoča inicializacijo na zahtevo:

```
[ModuleExport(typeof(InitTrgovec), InitializationMode=InitializationMode.OnDemand)]
public class InitTrgovec : IModule
{
        public void Initialize()
    {
      //Implementacija …
    }
}
```

Opomba:

- zastavica »InitializationMode=InitializationMode.OnDemand« omogoča inicializacijo modula na zahtevo.

7.2.2 Proces nalaganja modulov s pomočjo komponente ModuleManager

V poslovni aplikaciji STrgovini se modul »Trgovec« in modul »Poslovodja« inicializirata na zahtevo. Zahteva za inicializacijo nastane med akcijo prijave uporabnika v modulu »Prijava«. Akcija prijave vključuje komponento »ModuleManager«, ki omogoča nalaganje modulov v aplikacijo.

Primer nalaganja modula »Trgovec« v aplikacijo s pomočjo komponente »ModuleManager«:

```
this.moduleManager.Value.LoadModule("InitPoslovodja");
```

Opomba:

- komponenta »ModuleManager« vsebuje metodo »LoadModule«, ki naloži določen modul v aplikacijo. Kot parameter sprejme ime inicializacijskega razreda določenega modula.

7.2.3 Registracija in nalaganje uporabniških pogledov v regije

Moduli poslovne aplikacije STrgovina svoje uporabniške poglede registrirajo in nalagajo v določene regije. Komponenta, ki jo pri tem uporabijo, se imenuje RegionManager.

Primer uporabe komponente RegionManager za registracijo pogleda »StartView« v modulu »Prijava«:

```
this.regionManager.RegisterViewWithRegion(RegionNames.MainRegion, typeof(StartView));
```

Opomba:
- komponenta »RegionManager« vsebuje metodo »RegisterViewWithRegion«, ki omogoča registracijo in nalaganje uporabniških pogledov v določeno regijo na lupini. Kot paramater sprejme ime regije ter uporabniški pogled.

7.2.4 Uporaba komponente »EventAggregator«

V poslovni aplikaciji STrgovina modul »Prijava« pridobi informacije o trenutno prijavljenem uporabniku – trgovcu. Le-te za svoje delovanje potrebujeta tudi modul »Trgovec« in modul »Poslovodja«. Kot rešitev dane komunikacije med moduli smo uporabili komponento Microsoft PRISM knjižnice, imenovano »EventAggregator«. Slednja implementira mehanizem objavljanja in naročanja na dogodke.

Dogodek v primeru aplikacije STrgovina nosi informacijo o identifikatorju trenutno prijavljenega uporabnika. Identifikator smo poimenovali »IDTrgovecEvent«.

Implementacija dogodka »IDTrgovecEvent«:

```
[Export]
public class IDTrgovecEvent : CompositePresentationEvent<int>
{
}
```

Razred »IDTrgovecEvent« mora razširjati razred CompositePresentationEvent, saj le-ta omogoča metodi objavljanja in naročanja. Kot vsebino sporočila smo določili celo število.

V splošnem se komponente aplikacije (npr.: razredi, moduli …) delijo na založnike, torej tiste ki dogodke objavljajo, ter na naročnike, ki se na dogodek naročajo. V aplikaciji STrgovina založnika predstavlja modul »Prijava«, naročnika pa modula »Trgovec« in »Poslovodja«. Pri implementaciji naročanja in objavljanja dogodkov je potrebno paziti na vrstni red akcij. V splošnem se morajo naročniki najprej registrirati na dogodek, šele nato ga lahko založniki objavijo.

Primer naročanja modula »Trgovec« na dogodek »IDTrgovecEvent«:

```
public InitTrgovec(IEventAggregator _eventAggregator)
{
        this.eventAggregator = _eventAggregator;
        this.eventAggregator.GetEvent<IDTrgovecEvent>().Subscribe(GetIDTrgovec, ThreadOpti-
        on.UIThread, true);
}

public void GetIDTrgovec(int id_trgovec)
{
        //Implementacija …
}
```

Opis: v konstruktorju razreda InitTrgovec (predstavlja inicializacijski razred modula »Trgovec«) se pokliče komponenta »EventAggregator«. Ta sproži proces naročanja na dogodek »IDTrgovecEvent«, rezultat prejetih dogodkov shrani v metodo GetIDTrgovec. Slednja omogoči modulu »Trgovec« upravljanje z identifikatorjem trenutno prijavljenega trgovca.

Primer objavljanja dogodka »IDTrgovecEvent« v metodi »Prijava()« razreda »PrijavaViewModel«:

```
this.eventAggregator.GetEvent<IDTrgovecEvent>().Publish(Trgovec.ID_trgovec);
```

Opis: Po uspešno naloženem modulu »Trgovec« ali »Poslovodja« se pokliče komponenta »EventAggregator«, ki sproži proces objavljanja dogodka »IDTrgovecEvent«.

7.2.5 Uporaba komponente »ServiceReference«

V poslovni aplikaciji STrgovina moduli do WCF spletnih storitev dostopajo preko komponente, imenovane »ServiceReference«. Sledenja je del tehnologije WCF RIA Services, ki omogoča hitro implementacijo večplastnih Silverlight spletnih aplikacij, brez da bi se pri tem osredotočali na povezavo storitev med klienti in strežniki za upravljanje s podatki.

Primer inicializacije in uporabe komponente »ServiceReference« za dostop do WCF spletnih storitev, ki jih ponuja WCF spletni servis imenovan »ServiceLink«:

```
//Inicializacija komponente »ServiceReference« za dostop do WCF spletnega servisa »ServiceLink«
private ServiceReference.ServiceLinkClient srv = new ServiceLinkClient();

//Uporaba komponente »ServiceReference« za dostop do storitve »VrniArtikel«
srv.VrniArtikelCompleted += new EventHan-
dler<VrniArtikelCompletedEventArgs>(srv_VrniArtikelCompleted);
srv.VrniArtikelAsync(ArtikelFilter);
```

7.3 WCF spletni servis »ServiceLink«

WCF spletni servis, imenovan »ServiceLink« je del spletnega strežnika »Prism.STrgovina.Web«, ki je namenjen gostovanju Silverlight spletnega uporabniškega vmesnika. Spletni servis »ServiceLink« omogoča Silverlight klientu dostop do WCF spletnih storitev, ki jih ponuja WCF spletni servis »WCFService«. Slednji predstavlja plast pristopa k podatkom poslovne aplikacije STrgovina.

Primer implementacije WCF spletnega servisa »ServiceLink«:

```
[ServiceContract]
public class ServiceLink
{
    //Inicializacija komponente »ServiceReference« za dostop do WCF spletnega servisa  »WCFService«
    private ServiceReferenceWS.IWCFService srv = new WCFServiceClient();

    //WCF spletna storitev »VrniArtikel«
    [OperationContract]
     public IEnumerable<Artikel> VrniArtikel(string filter)
     {
        return srv.VrniArtikel(filter);
     }

    //Sledijo ostale WCF spletne storitve, ki so implementirane na podoben način …
}
```

Opomba:

- zastavica [ServiceContract] označuje dani razred kot WCF spletni servis;
- zastavica [OperationContract] označuje metodo »VrniArtikel« kot WCF spletno storitev.

7.4 WCF spletni servis »WCFService«

WCF spletni servis »WCFService« je del spletnega strežnika »Prism.STrgovina.WS«. Slednji definira podatkovno plast poslovne aplikacije STrgovina. Sestavljen je iz dveh razredov. Prva razred, imenovan »IWCFService«, predstavlja vmesnik, drugi razred »WCFService« pa vsebuje implementacijo spletnih storitev. Za upravljanje in dostop do podatkovne baze skrbi ORM tehnologija, imenovana »ADO.net Entity Framework«. Pravila uporabe dane tehnologije so definirana v poslovnih razredih »Upravitelji«. Spletne storitve svoje naloge posredujejo »Upraviteljem«, ki z uporabo tehnologije »ADO.net Entity Framework« dano nalogo opravijo.

Primer implementacije razreda »IWCFService« in »WCFService«:

```
[ServiceContract]
public interface IWCFService
{
        [OperationContract]
        IEnumerable<Artikel> VrniArtikel(string filter);
}

public class WCFService : IWCFService
{
        public IEnumerable<Artikel> VrniArtikel(string filter)
    {
                return UpraviteljArtikel.VrniArtikel(filter);
    }
}
```

7.4.1 Poslovni objekti imenovani »Upravitelji«

WCF spletni servis »WCFService« vsebuje štiri poslovne razrede, imenovane »Upravitelji«, ki definirajo poslovna pravila uporabe ORM tehnologije »ADO.net Entity Framework«. Med »Upravitelje« sodijo: razred »UpraviteljArtikel«, razred »UpraviteljStranka«, razred »UpraviteljTrgovec« in razred »UpraviteljPoslovanja«.

Primer implementacije poslovnega razreda »UpraviteljStranka« in metode »VrniStrankaID«:

```
public static class UpraviteljStranka
  {
        //Metoda, ki išče stranko po identifikatorju
    public static Stranka VrniStrankaID(int id)
    {
      try
      {
        using (var dataContext = new Entities())
        {
          Stranka str = dataContext.Stranka.FirstOrDefault(cRole => cRole.ID_stranka == id);

          return str;
        }
      }
      catch (Exception ex)
      {
        throw ex;
      }
    }
}
```

Opomba:

- metoda »VrniStrankaID« sprejme kot parameter identifikator stranke, rezultat iskanja pa vrne kot objekt Stranka;
- objekt Stranka predstavlja virtualno različico tabele Stranka v podatkovni bazi, kreirali pa smo ga s pomočjo ORM tehnologije »ADO.net Entity Framework«.

*Primer uporabe ORM tehnologije »ADO.net Entity Framework« za iskanje strank po določenem identifikatorju v metodi »*VrniStrankaID*«:*

```
using (var dataContext = new Entities())
{
        Stranka str = dataContext.Stranka.FirstOrDefault(cRole => cRole.ID_stranka == id);

        return str;
}
```

Opomba:

- spremenljivka dataContext predstavlja instanco ORM razreda Entities;
- razred Entities predstavlja virtualno podatkovno bazo.

8 ZAKLJUČEK

Skozi načrtovanje in razvoj poslovne aplikacije STrgovina sem spoznal glavne faze razvoja programskih produktov ter glavne smernice, ki jih pri tem določa okolje Microsoft PRISM. Slednje razvijalcem omogoča razvoj bogatih, fleksibilnih in enostavno razširljivih spletnih aplikacij.

Razvoj poslovne aplikacije STrgovina je sestavljen iz šestih faz. Med slednje sodijo: faza definicije problema, faza študije izvedljivosti, faza analize in definiranja zahtev, faza načrtovanja sistema, faza načrtovanja programa in faza izvedbe. Ugotovili smo, da faze omogočajo natančen pregled nad razvojem aplikacije ter posledično prispevajo k uspešnosti projekta. S pomočjo smernic, ki jih določa okolje Microsoft PRISM, smo implementirali modularno aplikacijo, ki omogoča enostavno dodajanje novih funkcionalnosti ter vzdrževanje obstoječih. Pri razvoju poslovne aplikacije STrgovina smo prišli do nekaterih izzivov, ki so se pojavljali predvsem pri začetni postavitvi PRISM aplikacijske infrastrukture. Na splošno pa lahko povzamemo, da razvoj poslovnih aplikacij v okolju Microsoft PRISM predstavlja bogato izkušnjo, ki razvijalcem poda celovit pregled nad implementacijo modularnih aplikacij.

Poslovna aplikacija STrgovina predstavlja zgolj osnovo, na kateri se lahko gradi v prihodnosti. Ena izmed zanimivih in pomembnih funkcionalnosti, ki bi jih bilo potrebno v prihodnosti implementirati, je plačilni sistem.

V zaključku lahko še dodam, da sem pri razvoju poslovne aplikacije STrgovina pridobil veliko izkušenj, s katerimi bom lažje prodiral na poslovni trg.

Literatura

[1] Anja Jablanovec, Matej Zdovc, Kako postaviti spletni strežnik, Seminarska naloga pri predmetu strokovna informatika in statistične metode vrednotenja, Celje, 2011.

[2] Bob Brumfield, Geoff Cox, David Hill, Brian Noyes, Michael Puleio, Karl Shifflett, Developer's Guide To Microsoft PRISM 4, Building Modular MVVM Applications using Windows Presentation Foundation and Microsoft Silverlight.

[3] Chris Anderson, Pro Business Applications with Silverlight 5, Second Edition, 2012.

Viri

[4] Microsoft Developer Network, Event Aggregator , http://msdn.microsoft.com/en-us/library/ff921122(v=pandp.20).aspx, 2013.

[5] Microsoft DreamSpark for Academic Institutions, msdn academic alliance, Software Center , http://msdn62.e-academy.com/uprim_foed, 2013.

[6] Microsoft Developer Network , WCF RIA Services, http://msdn.microsoft.com/en-us/library/ee707344(v=VS.91).aspx, 2013.

[7] Microsoft Developer Network, Entity Framework Overview, http://msdn.microsoft.com/en-us/library/bb399567.aspx, 2013.

[8] W3C Working Group Note, Web Service Architecture, http://www.w3.org/TR/ws-arch/#whatis, 2004.

[9] W3C, Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383486, 2000

Priloge

- Zgoščenka s celotno izvorno kodo poslovne aplikacije STrgovina.