# Projekt izrade aplikacije za upravljanje skladištem

**Zlatko, Popić**

**VELEUČILIŠTE VERN'**

Zagreb

POSLOVNA INFORMATIKA

**ZAVRŠNI RAD**

# Projekt izrade aplikacije za upravljanje skladištem

Zlatko Popić

Zagreb, 2017.

**VELEUČILIŠTE VERN'**

Preddiplomski stručni studij

Poslovna informatika

**ZAVRŠNI RAD**

# Projekt izrade aplikacije za upravljanje skladištem

Mentor: Krešo Vargec, dipl. ing.

Student: Zlatko Popić

Zagreb, listopad 2017.

VELEUČILIŠTE VERN'
Zagreb, Trg bana Josipa Jelačića 3
Poslovna informatika

Broj 3067

# *ZADATAK ZAVRŠNOGA RADA*

***Student/ica:*** Zlatko Popić

***Zadatak:*** Projekt izrade aplikacije za upravljanje skladištem

***U radu je potrebno razraditi sljedeće:***

- Opisati poslovnu ideju
- Izraditi analizu i specifikaciju aplikacije
- Opisati modele podataka i tokove podataka
- Opisati implementaciju aplikacije
- Izraditi zaključke i preporuke za praksu

***Napomena:*** *Pri izradi završnoga rada kandidat/kinja ima obvezu pridržavati se i uvažavati primjedbe, sugestije i naputke mentora/ice, koristiti i primjenjivati znanja i umijeća stečena tijekom studija, upotrebljavati informacije i podatke prikupljene vlastitim istraživanjem te spoznaje i činjenice iz odgovarajuće znanstvene i stručne literature uz ispravno navođenje korištenih izvora.*

Zadatak zadan 31. 8. 2017.
Rok predaje 2. 11. 2017.

**Mentor/ica:**

Pred. Krešo Vargec, dipl.ing

REPUBLIKA HRVATSKA * VELEUČILIŠTE VERN' * ZAGREB 2

**Voditelj/ica studija:**

V.pred. mr.sc. Ivo Beroš

Veleučilište VERN'

**IZJAVA**

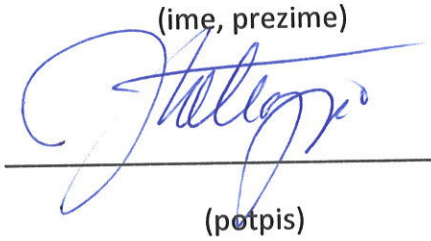
kojom izjavljujem da sam završni rad pod naslovom

Projekt izrade aplikacije za upravljanje skladištem,

izradio/la samostalno. Svi dijelovi rada, nalazi ili ideje koje su u radu citirane ili se temelje na drugim izvorima, bilo da su u pitanju knjige, znanstveni ili stručni članci, internetske stranice, propisi i sl. u radu su jasno označeni kao takvi te adekvatno navedeni u popisu literature.

Zagreb, 9. 11. 2017.

ZLATKO POPIĆ
(ime, prezime)



(potpis)

# SADRŽAJ

# SAŽETAK

Predmetni rad naslova „Projekt izrade aplikacije za upravljanje skladištem“ prikazuje najvažnije i neophodne korake pri planiranju i izradi programske podrške poslovnim procedurama kroz primjer izrade jednostavne aplikacije za vođenje skladišnog poslovanja.

Izrada jednostavne aplikacije podrazumijeva da autor ne prikazuje u detalje sve aspekte ove znanstvene discipline, već ukazuje na opće smjernice upravljanja prikazanim projektom te naglašava iznimnu važnost uloge programskog inženjerstva čija kvalitetna primjena ima dalekosežno pozitivne učinke na poslovanje potpomognuto programskom podrškom.

Rad počinje uvodom u općenite smjernice odabranih poglavlja programskog inženjerstva i upravljanja projektima, a nastavlja se ilustriranjem svake faze razvoja aplikacije pogodnim primjerom praktične izvedbe aplikacije za skladišno poslovanje izrađene za potrebe ovog rada.

Ključne riječi: programsko inženjerstvo, programska podrška, upravljanje projektima, modeliranje sustava, baze podataka

## SUMMARY

The aim of this thesis titled „A warehouse management application design project“ is to provide an insight into the fundamental and crucial steps of software engineering and software project planning processes through a simple inventory management application.

Building a simple application implies that the author did not want to cover all the aspects of the software engineering discipline, as he was striving to present the general guidelines and emphasise the importance of the software engineering discipline which can exert extensive and positive impact on business processes if applied properly.

The thesis starts by addressing selected general software engineering and software project management topics. It subsequently elaborates on all the steps of application development through relevant illustration from the actual inventory management application prepared primarily for this thesis.



Keywords: Software engineering, software, project management, system modelling, database systems

# 1. UVOD

Cilj rada je istaknuti važnost uloge sustavnog pristupa izradi kvalitetnih programskih rješenja kao i središnje uloge struke programskog inženjerstva u tom procesu. Nerazumijevanje položaja programskog inženjerstva u razvoju programske podrške može prouzročiti znatnu poslovnu štetu i s čak pogubnim posljedicama, pogotovo pri razvoju visoko složenih sustava (Leveson, 2004).

Ovaj rad objašnjava temeljne principe upravljanja projektom razvoja aplikacija s gledišta programskog inženjerstva kroz praktičan primjer izrade jednostavne aplikacije za skladišno poslovanje koja čini jezgru većeg informacijskog sustava za skladište mješovite robe. Rad neće prikazati sve aspekte programskog inženjerstva što daleko prelazi osnovnu autorovu namjeru kao i raspoloživ obavezni format ovoga rada.

U početku se rad osvrće na kratku povijest programskog inženjerstva kako bi dao smjernice za razumijevanje discipline kao i temeljnih pojmova kojima se ona služi. Temeljni pojmovi obuhvaćaju objašnjenje što je programska podrška[1] odnosno softver (engl. *Software*), što su poslovni procesi te na koji način su programsko inženjerstvo i poslovni procesi povezani u cjelinu koja se može nazvati upravljanjem izrade programske podrške poslovnim procesima.

Rad nastavlja teoretskim prikazom opće arhitekture informacijskog sustava za skladište mješovite robe, zatim se određuje opseg u kojem će se podrobnije objasniti ključne faze u izradi pojedinačnih dijelova programske podrške. Prikazat će se tehnički i metodološki pristupi izrade aplikacija na općenitoj i specifičnoj razini. Svi spomenuti pristupi i prikazi u izradi programske podrške bit će potkrijepljeni praktičnim primjerom skladišne aplikacije obrađujući temeljne poslovne procedure skladišnog poslovanja kao što je upravljanje primkama i otpremnicama, osnovnim proračunima te osnovnim sustavom izvještavanja i upozoravanja.

Ovaj rad stavlja težište na tehnički aspekt programskog inženjerstva, a manje na, također vrlo važan, organizacijski aspekt, odnosno upravljanje programskim projektom, o čemu će rad dati kratko očitovanje i prikaz u smislu osnovne organizacije vođenja projekta. Upravljanje projektom prikazat će se kroz prikladan dijagram s vremenskom raspodjelom zadataka i ljudskih resursa.

[1] Vidjeti izraz prema *Hrvatskom pravopisu*: http://pravopis.hr/pravilo/pisanje-opcih-rijeci-i-sveza/46/

# 2. UVOD U PROGRAMSKO INŽENJERSTVO

U ovom poglavlju bit će riječi o razlozima nastanka struke programskog inženjerstva i nastojat će se objasniti važna poveznica između programske podrške i poslovnih procesa te na koji način obje stavke međusobno djeluju.

## 2.1. Kratka povijest programskog inženjerstva i kratka definicija

Termin „programsko inženjerstvo“ (engl. *Software engineering*) prvi puta se spominje tijekom konferencije NATO-ova saveza 1969. godine, čija je tema bila rješavanje nagomilanih problema programske podrške velikim poslovnim sustavima (Sommerville, 2009). Ta konferencija, kao i sve nastavne, trebala je doprinijeti rješavanju onog što se nazivalo „kriza programske podrške“ (engl. *Software crisis*) (Naur and Randell, 1969. prema Sommerville, 2009.) izazvane problemima loše kontrole razvoja programske podrške poslovnim sustavima.

Razvojem internetskih usluga od 1990. godine i sve bržim razvojem multimedijalnih sadržaja povećava se i potreba za kvalitetnijim, financijski discipliniranijim i korisnijim programskim podrškama. Razvoj takve programske podrške sve više zahtijeva kvalitetnije upravljanje razvojem aplikacija kao i praćenje poslovnih procesa programskim kodom.

Novi milenij donosi i povećane zahtjeve za prilagodljivijim programskim rješenjima u većim poslovnim organizacijama koje prvi puta koriste programska rješenja okrenuta ne samo korisnicima, već i unutarnjoj organizaciji, odnosno mikrookruženju. Primjer takvih zahtjeva su sustavi vezani za aplikacije financijskih institucija i njihovih bankomatskih mreža ili razvoj specifičnih programskih alata u organizaciji poslovanja različitih odjela poduzeća.

Danas se koriste brojni načini upravljanja razvojem programske podrške, a proizvođači programske podrške često koriste i kombinacije već poznatih, u praksi ustanovljenih i provjerenih modela razvoja programske podrške.

Ukratko, programsko inženjerstvo, s povijesnog i funkcionalnog stajališta, moglo bi se definirati kao sustav teorijskih i tehnoloških procedura pri razvoju programske podrške poslovnim procesima. Iako je točno da se metode programskog inženjerstva mogu koristiti i na najjednostavnijim primjerima aplikacija, najpotrebnije su pri izradi aplikacija čiji je prvenstveni cilj informatizacija poslovnih procesa.

## 2.2. Svrha programskog inženjerstva i upravljanja projektima

Programsko inženjerstvo daleko nadilazi trivijalnu frazu „izrada programa“. Ne samo da je nadilazi, već bi takva pomalo habitualna definicija bila posve netočna. Vjerojatno najkorektniju definiciju donosi Somerville, koji jasno ističe da je programsko inženjerstvo sistematski pristup proizvodnji programske podrške što uključuje cijenu izrade, terminski plan izrade te sve sudionike procesa – kako one koji izrađuju, tako i one koji će je i koristiti, odnosno korisnike (Sommerville, 2009).

Važno je istaknuti da je programsko inženjerstvo – inženjerstvo, pristup koji odabranim metodama i postupcima rješava zadatke kako bi sastavio prije svega funkcionalna rješenja, ali u isto se vrijeme brine o kvaliteti rješenja, ponovnoj iskoristivosti te mogućnosti proširenja za rješavanje i drugih zadataka.

Potonji zahtjev za proširenjem i funkcionalnošću, pogotovo po uključivanju većih poslovnih institucija u informatizaciju, postaje imperativ, a ne mogućnost. Tako je danas već i u laičkoj praksi korištenja programske podrške prvo pitanje korisnika vezano za raspoložive mogućnosti nadogradnje. Kako bi sustav bio fleksibilan, nadogradiv i općenito održiv, presudno je kvalitetno definirati poslovne procese poduzeća u dijelu u kojem se traži programska potpora.

Za kvalitetno definiranje poslovnih procesa potrebno je načelno poznavati unutarnje funkcioniranje poslovne organizacije. Historiografski rečeno, poslovni procesi kao samostalan termin poslovanja korijen imaju u pojavi manufaktura, a da je problematika poslovnih procesa stara koliko i sama proizvodnja, odlično ilustrira i tekst o ranoj manufakturi Dubrovačke Republike. Posebno je zanimljiv sljedeći dio teksta: „Sam Pantela nije mogao neprestano i bez izbivanja biti u radionici, jer mu je u dio pala obaveza da prati cijeli proizvodni proces i da ne dopušta zastoje u radionici ni prekid između triju proizvodnih faza. Da bi ispunio uvjete ugovora s komunom, Pantela je morao ne samo da rukovodi radionicom, nego da i zanatlije prisili na sinhronizirani rad u I i u II fazi proizvodnje“ (Manančikova, 1977. str. 349, 350).

Ovo je bilo 1430. godine. Praćenje proizvodnog ciklusa, upravljanje terminskim planom (zastoji), ispunjavanje uvjeta ugovora, sinkronizacija rada, odabir najboljih zanatlija za svaku fazu poslovanja (danas se koristi termin *ljudski resursi*), sve su odreda izričaji koji se i danas koriste tijekom razvoja programskih rješenja i upravljanja projektima.

## 2.3. Što je softver[2]

Riječ „softver“ je anglizam i u praksi se najčešće koristi fonetski kao neprilagođeni neologizam koji je vrlo teško jednoznačno prevesti. Prijedlozi struke su „programska podrška“ kao dulji naziv ili „napudbina/napudba“ kao kraći (Mihaljević:, 2007). Autor ovog rada predlaže definiciju što programska podrška jest s funkcionalnog stajališta. Takvo stajalište donosi definiciju softvera kao skupa računalnih programa i prateće dokumentacije razvijenih za potrebe savladavanja zadataka radnih procesa (prilagođeno prema: Sommerville, I. (2009), *Software engineering*, str. 6.).

## 2.4. Što su poslovni procesi

Prvu sistematsku pažnju na poslovne procese kao dio „lanca vrijednosti“ iznosi Michael M. Porter u svojoj knjizi *Competitive Advantage: Creating and Sustaining Superior Performance* u kojoj na detaljan način objašnjava stvaranje vrijednosti usmjereno klijentu ili općem tržištu. Prema Porteru (1985) stvaranje takvih vrijednosti dio je najuže jezgre poslovanja svake organizacije, a nastaje kao rezultat strukturiranih, analitičkih i logički povezanih aktivnosti. Autor rada, prema različitim izvorima te vlastitom radnom iskustvu, može pridonijeti definiranju poslovnih procesa kao precizno definiranim skupinama međuovisnih zadataka koji u relativno konstantnim intervalima pod uvjetom jednakih ulaznih parametara proizvodi planiranu vrijednost za tržište.

## 2.5. Poslovni procesi i programsko inženjerstvo

Dva su posve različita gledišta s kojih se može promatrati poslovne procese. S jedne strane su voditelji poslovnih procesa u poduzećima, stručni u upravljanju poslovnim procesima, a s druge strane razvojni timovi zaduženi za izradu programske podrške poslovnim procesima koji su stručni u upravljanju razvojem programske podrške.

Obje strane su na udaljenim pozicijama između kojih je ono što autor označava kao „kontekstualni jaz“. Prvospomenuti stručnjaci upravljanja poslovnim procesima nisu stručni i u razvoju programskih rješenja, dok su drugospomenuti, razvojni timovi za izradu programske

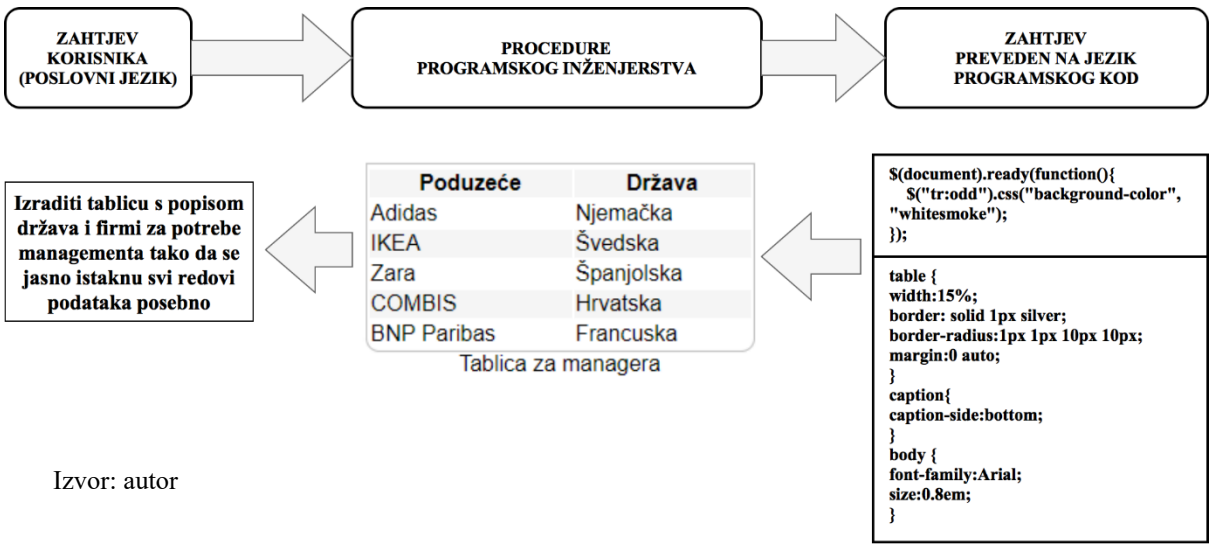
[2] Ibid, str. 1.

podrške često posve nesvjesni kompleksnosti poslovnih procesa i specifičnosti zahtjeva poslovnih procedura (Bubaš, Hutinski, i Kermek, 2000.).

Autor rada želi naglasiti da, u skladu s prethodno definiranim „kontekstualnim jazom“, programsko inženjerstvo premošćuje spomenuti jaz i svojim utvrđenim procedurama prevodi poslovne procedure iz domene upravljanja poslovnim procesima u programsku podršku poslovnim procesima. Samo o kvalitetno strukturiranim projektima pomoću programskog inženjerstva ovisi i kvaliteta isporučene programske platforme. Pojednostavljeno, programsko inženjerstvo prevodi poslovne procedure s jezika upravljača (menadžera) na jezik programskoga koda (programera).

Ukratko, jednostavnije prikazano, odnos i položaj poslovnih procesa, procedura programskog inženjerstva i samog razvoja programske platforme može se prikazati kako slijedi:

Slika 2.1. Pojednostavljeni prikaz uloge programskog inženjerstva




Izvor: autor

Prikaz na Slici 2.1. na vrlo jednostavnom primjeru prikazuje mjesto u kojem se nalaze procedure i pravila programskog inženjerstva u odnosu suradnje razvojnog tima stručnjaka s jedne strane i klijenta odnosno poslovnih procedura, s druge strane.

## 2.6. Procesi programskog inženjerstva, metode i alati

Današnje poimanje programske podrške podrazumijeva, bez iznimke, prije svega kvalitetu, a preciznije, kvaliteta u ovom smislu označava neophodne atribute (Manger, 2005):

1. Mogućnost održavanja – označava unutarnju arhitekturu programske podrške kao prilagodljivog sustava koji se može i mora moći mijenjati, odnosno prilagođavati izmjenama poslovnih procedura.
2. Pouzdanost i sigurnost – označava ponašanje programske podrške na predvidiv način.
3. Efikasnost – označava da sustav programske podrške mora djelovati štedljivo, racionalizirati resurse što se označava kao zadovoljavajuće postignuće.
4. Upotrebljivost – možda i najvažniji atribut odražava se u stupnju zadovoljenja poslovnih procesa na očekivan način s razumljivim korisničkim sučeljem i kvalitetnom dokumentacijom.

## 2.7. Procesi, aktivnosti i modeli programske podrške

Procesi programske podrške obuhvaćaju aktivnosti i rezultate čiji je cilj razvoj ili evolucija programske podrške. Iako postoje različiti procesi programske podrške različitih modela, svi moraju obuhvaćati najmanje sljedeće osnovne aktivnosti:

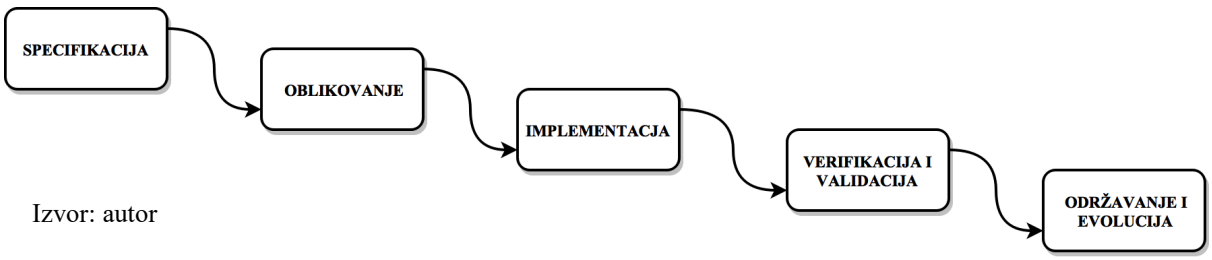
1. Specifikacija – u izradi specifikacije analiziraju se zahtjevi klijenta i određuje se što programska podrška mora raditi i koje zadaće obavljati.
2. Oblikovanje – u ovoj se fazi uređuju odnosi te izgled i rad sučelja među dijelovima programske podrške te se određuje na koji način će programska podrška raditi.
3. Implementacija – često se naziva i programiranje, faza u kojoj se načelna arhitektura sustava prevodi u programski kod pomoću programskih jezika i alata.
4. Verifikacija i validacija – važna je faza u kojoj se provjerava radi li programska podrška u skladu sa specifikacijom. Najčešće je sinonim za testiranje.
5. Održavanje ili evolucija – odnosi se na sve naknadne izmjene i nadogradnje programske podrške.

Osnovne aktivnosti procesa povezane su na unaprijed oblikovan način kroz idealizaciju međusobnih odnosa i veza i nazivaju se objedinjenim nazivom „model procesa programske podrške“.

Ovaj će rad opisati dva temeljna modela iz kojih su izvedeni svi ostali, a čine u osnovi dva načina razmišljanja, odnosno konceptualizacije i apstrakcije u razvoju programskih rješenja. Razvoj se može savladavati korak po korak, zatim završava cjelina i na kraju testira ili se paralelno razvijaju manji dijelovi programske podrške koji prolaze osnovne aktivnosti te na kraju uglavljuju u cjelinu.

1. **Model vodopada** – najjednostavniji je model razvoja programske podrške razvijen 70-ih godina 20. stoljeća. Odlikuje ga vrlo jasna struktura u kojoj svaka osnovna aktivnost prati drugu, a najveći nedostatak dotičnog modela jest u tome što se testiranjem započinje tek u završnoj fazi razvoja. (Vargec, Softverski modeli, 2017.)

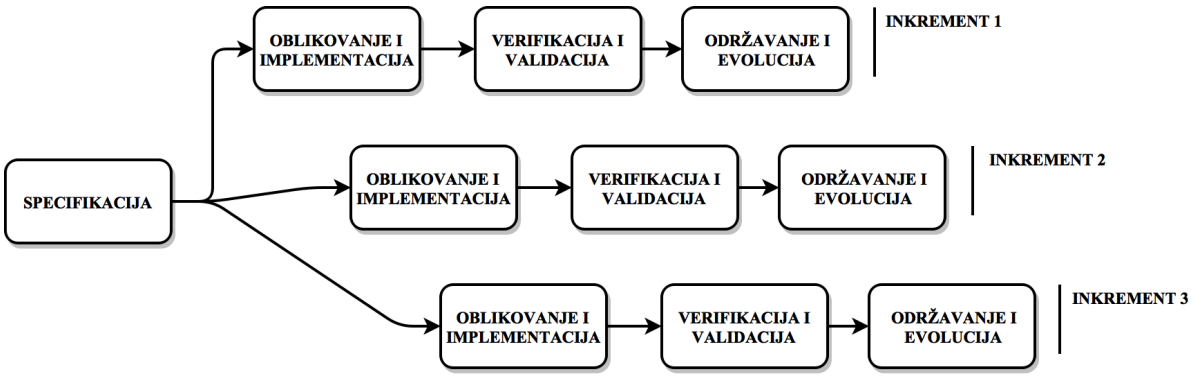
Slika 2.2. Model vodopada




Izvor: autor

2. **Model inkrementalnog razvoja** – razvoj u usporednim fazama različitih dijelova koji će se uklopiti u cjelinu na kraju razvojnog procesa. Svaka završena faza isporučuje završeni inkrement koji je prošao sve faze osnovnih aktivnosti (Vargec, Softverski modeli, 2017.).

Slika 2.3. Model inkrementalnog razvoja




Izvor: autor

Tijekom vremena razvijen je velik broj modela, a dalje se ukratko spominju još dva, suvremena načina razvoja programske podrške koji proizlaze iz prethodno navedenih osnovnih modela: model agilnog razvoja i spiralni model. Ovaj rad se u dijelu razvoja skladišne aplikacije vodi osnovnim inkrementalnim modelom.

Model agilnog razvoja – tip inkrementalnog razvoja s manjim razvojnim zahtjevima i brzim razvojnim tempom, a za razvoj aplikacije u ovom radu odabran je princip testiranjem vođenog razvoja. Ekstremno programiranje (engl. *XP – Extreme programming*) je najpoznatija verzija agilnog razvoja (Vargec, *Rapid software development*, 2017.).

Spiralni model – verzija modela koji se odvija u fazama: planiranje, analiza rizika, razvoj i vrednovanje. Faza rizika je ključna u spiralnom modelu razvoja do te mjere da su potrebna ekspertiza i znanje u tom segmentu ključni za uspjeh projekta. Ovaj model je najpodesniji za posve nove linije programskih proizvoda ili proizvoda visoko kompliciranih zahtjeva s izričitim naglaskom na upravljanje rizikom (Vargec, Softverski modeli, 2017.).

## 2.8. Metode i alati razvoja programske podrške

Metoda razvoja programske podrške odnosi se na detaljniju i konkretniju razradu procesa razvoja programske podrške što uključuje podjelu procesa na aktivnosti, podaktivnosti i zadatke te jasno propisuje način dokumentiranja (dijagrami, pseudojezik, tabele) te organizaciju i stil rada kao i odabrani način programiranja.

Razvoju programske podrške pomažu i aplikacije objedinjene pod imenom CASE[3], a pružaju rad s osnovnim aktivnostima odabrane metode prema vlastitom dizajnu ili predlošku.

*Upper* CASE je zajednički naziv za računalne aplikacije kojima se može vizualno i tekstualno organizirati sam početak razvoja programske podrške u smislu izrade specifikacije i oblikovanja kao što je aplikacija Microsoft Visio za crtanje UML[4] dijagrama.

*Lower* CASE su također računalne aplikacije koje služe za sam razvoj programske podrške kao što je Microsoft Visual Studio kao podrška u implementaciji i testiranju.

[3] CASE (engl. *Computer Aided Software Engineering* – računalno potpomognuto programsko inženjerstvo).
[4] UML (engl. *Unified Modelling Language* – standardizirani jezik za modeliranje).

# 3. PLANIRANJE PROGRAMSKE PODRŠKE

U ovom poglavlju rad se osvrće na planiranje programske podrške, važnu fazu u smislu utemeljenja općenitih parametara, metodologije izrade, opis te samu svrhu projekta koji će se izraditi. Od ovog poglavlja nadalje temeljne procedure programskog inženjerstva prikazuju se kroz primjer izrade programske podrške za skladišno poslovanje.

## 3.1. Opis projekta

*Skladište* je jednostavna aplikacija za vođenje osnovnih procedura skladišnog poslovanja. Na uvodnom sastanku korisnik je izričito naglasio da aplikacija mora biti jednostavna pa je prema tome dogovoren opseg projekta te je na lokaciji korisnika utvrđeno sadašnje stanje i preuzeti su klijentski zahtjevi.

## 3.2. Opseg projekta

Zaključak je uvodnog sastanka da projekt obuhvaća izradu, instalaciju i kratku edukaciju o korištenju skladišne aplikacije. Projekt ne uključuje održavanje aplikacije, ali će se predočiti odvojeni ugovor za održavanje bude li potrebno. Projekt se odnosi na izradu programske podrške za rad u odjelima narudžbe, nabave i dostave robe.

Pod izradom se podrazumijeva izrada dijelova programa grupiranih u module:

a) Modul *Šifrarnici* – mjesta, mjera, statusa i korisničkih uloga.

b) Modul izdavanja primki i otpremnica.

c) Moduli *Kupci*, *Dobavljači*, *Proizvođači* – u obliku jednostavnog adresara.

d) Modul *Administracija* – upisivanje korisnika aplikacije.

e) Modul *Izvještaji* – izvještaj za pojedinu otpremnicu i primku kao i izvještajni popis za sve primke i otpremnice.

f) *Glavni modul* – glavno sučelje koje povezuje sve module s prezentacijom osnovnih podataka skladišta.

Pod instalacijom se smatra:

a) izrada jednostavne instalacijske procedure (*one click* instalacija)

b) instalacija aplikacije kod korisnika.

Pod kratkom edukacijom smatraju se:

a) prezentacija i vježbe u radu korisnika, voditelja i operatera s aplikacijom nakon instalacije u trajanju od 90 minuta.

## 3.3. Svrha projekta

Svrha ovog projekta je izraditi aplikaciju za vođenje skladišnog poslovanja koje se do sada vodilo ručno, odnosno na papiru. Svrha aplikacije je da se temeljne procedure skladišnog poslovanja pojednostave i ubrzaju te da se na taj način ograniči mogućnost ljudske pogreške, a ako se ona i dogodi, da se na jednostavan i dostupan način može uočiti i ispraviti.

## 3.4. Metodologija

S obzirom na zaključke uvodnog sastanka, određeno je da će se aplikacija razvijati prema modelu sa središnjim repozitorijem prema principu testiranjem vođenog razvoja. Model središnjeg repozitorija označava arhitekturu programske podrške koja se temelji na središnjoj bazi podataka. Model testiranjem vođenog razvoja označava metodu razvoja programske podrške pri kojoj se svaka razvojna cjelina testira prije uključivanja u programsku cjelinu.

Slijed izrade aplikacije podijeljen je na sljedeći način:

1. Razvoj dijela aplikacije za unos i pregled kontakata – Kupci, Dobavljači i Proizvođači
2. Razvoj dijela aplikacije za unos i pregled poslovanja – Proizvodi, Primke i Otpremnice
3. Razvoj dijela aplikacije za kalkulacije poslovanja – Proizvodi, Primke i Otpremnice
4. Razvoj dijela aplikacije za izvještavanje – pregled svih i pojedinačnih primki i otpremnica.

Odlučeno je da će se koristiti inkrementalni model razvoja aplikacije s inkrementima za stavke 1. i 2. s nazivom CRUD inkrement, te za stavke 3. i 4. – FUNKCIONAL inkrement.

# 4. ANALIZA

U ovom će se poglavlju raspravljati o stanju postojećeg sustava, sadašnjim poslovnim procesima te će se na temelju organiziranog i sustavnog prikupljanja podataka analizirati klijentski zahtjevi i prema tome izraditi specifikacija programske podrške.

## 4.1. Analiza postojećeg sustava

Tijekom uvodnog sastanka korisnik je ukratko predstavio trenutni način rada u skladišnom poslovanju. Predstavljeni su djelatnici čiji je posao povezan sa skladišnim poslovanjem. Dogovoren je posjet u kojem će razvojni tim provesti planirano vrijeme od jednog radnog dana (osam radnih sati) na lokaciji skladišta poduzeća u svrhu bilježenja radnih procesa kako bi se funkcionalnosti buduće aplikacije prilagodile načinu rada korisnika. Dogovoreno je da će za taj posjet korisnik prirediti podatke o proizvođačima, dobavljačima, kupcima i robi koje posjeduje u digitalnom obliku (Microsoft Word dokumenti i Microsoft Excel tablice).

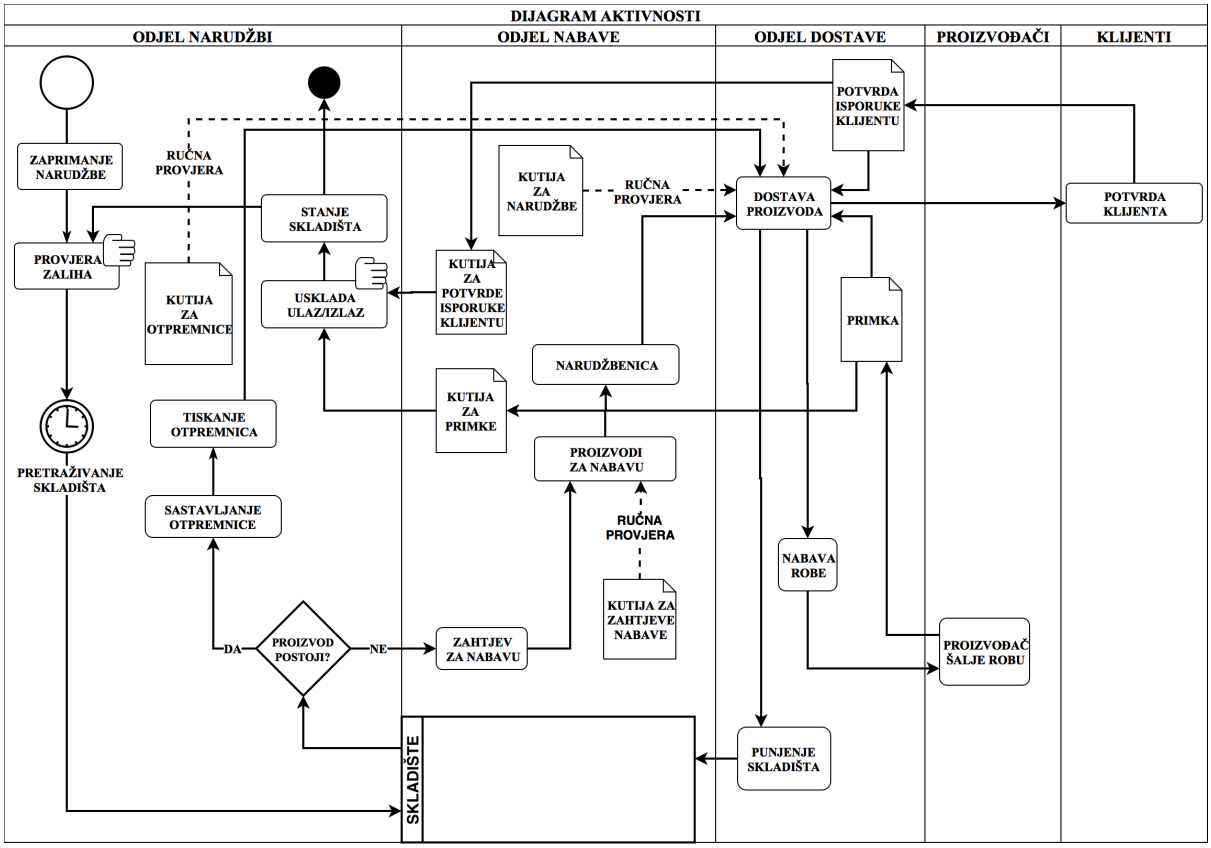
## 4.2. Sadašnji poslovni procesi vezani za skladišno poslovanje

Nakon provedene analize na lokaciji klijenta utvrđeno je da se skladišno poslovanje obavlja ručno, odnosno na papiru, pa se svi dokumenti otpremnica i primki ispunjavaju ručno i polažu u kutije te čuvaju u arhivi. Informacije o proizvođačima, dobavljačima i kupcima nalaze se na nekoliko različitih mjesta, u digitalnom obliku, u papirnatom obliku na stolovima operatera, dio u elektroničkoj pošti, a dio u drugom poslovnom sustavu. Ne postoji postojano vođenje zalihe pa se često događa da se izdaje roba sa skladišta koje nema. Svaki operater koji izdaje robu sa skladišta mora otići u skladište i prebrojati robu svaki puta kada se roba izdaje sa skladišta. Zbog brzine rada, velika količina vremena troši se na dohvat informacija o stanju skladišta pa je i kontrola izračuna u skladištu gotovo nepostojeća i ovisi isključivo o ažurnosti pojedinog operatera. Korisnik je u međuvremenu dostavio podatke u Microsoft Excel tablicama koji će se pokušati iskoristiti za početno punjenje šifrarničkih tablica u bazi podataka.

Prema obavljenim razgovorima s korisnikom te prikupljenim podacima u dijelu poslovanja koje će se pratiti programskom podrškom, izrađen je dijagram postojećih poslovnih procesa kako je prikazano na Dijagramu 4.1.

Topografija dijagrama pokazuje izrazitu složenost s nekoliko kritičnih točaka koje će programska podrška nastojati efikasno pojednostaviti. Važno je primijetiti da postoje čak tri skladišta podataka u obliku kutija za odlaganje papirnate dokumentacije odložene u različitim odjelima, da se sve provjerava ručno, da se ručno radi usklađivanje stanja robe te da se ručno provjerava stanje artikala u odjelu narudžbi. Kako se znatan dio vremena bespotrebno troši na traženje, slaganje i ažuriranje papirnate dokumentacije, tako je odlučeno da se prvo ustroji središnji repozitorij podataka te oko njega sastavi programska podrška primjerena prikazanim poslovnim procesima.

Slika 4.1. Dijagram postojećih procedura u poslovnom procesu skladišnog poslovanja




Izvor: autor

## 4.3. Prikupljanje i početno punjenje podataka

Nakon obrade dostavljenih korisničkih podataka nije bilo moguće uspješno iskoristiti spomenute podatke na predviđen način. Zbog toga je s korisnikom dogovoreno da će se prikupljeni i obrađeni podaci upisati u Microsoft Excel tablice koje po strukturi odgovaraju šifrarničkim tablicama u bazi podataka. Korisnik će u roku u kojem traje razvoj aplikacije u tim tablicama po potrebi ispraviti podatke i nadopuniti podacima koji ne postoje u digitalnom obliku te će se prije isporuke programske podrške ti podaci učitati u šifrarničke tablice.

Slike 4.2., 4.3. i 4.4. prikazuju tablice s podacima koji se upisuju u šifrarničke tablice baze:

Slika 4.2. Tablica dobavljača iz Microsoft Excel datoteke

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Dobavljac.Nazi | UlicaBroj | OIB | Email | Opis | Grad.Nazi |
| 2 | LIDL | Zadarska ul. 79 | 12345678912 | lidl@lidl.hr | super VIP status | Zagreb |
| 3 | KONZUM | Slavonska avenija 6 | 25987654321 | konz@konzum.hr | super VIP status | Zagreb |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |

Izvor: autor

Slika 4.3. Tablica kupaca iz Microsoft Excel datoteke

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Naziv | OIB | UlicaBr | Grad | PostanskiBroj |
| 2 | Virna | 13467986568 | Domo 2 | Zagreb | 10000 |
| 3 | Mirna | 12549587951 | Domo 34 | Zagreb | 10000 |

Izvor: autor

Slika 4.4. Tablica proizvoda iz Microsoft Excel datoteke

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Sifr | Naziv | Komad | JedCijen | JednicaMjereI | JedCijenaPD | PD | UkupnoPD | Proizvodjac |
| 2 | 10 | MLIJEKO - domaće | | 5,5 | 6 | 5,775 | 5 | | OPG - Zlayo |
| 3 | 20 | KRUH - raženi | | 8,5 | 4 | 9,775 | 15 | | OPG - Zlayo |
| 4 | 30 | NUTELLA - domaća | | 2,5 | 4 | 2,825 | 13 | | OPG - Zlayo |
| 5 | 40 | KAVA - mljevena | | 1,2 | 4 | 1,38 | 15 | | OPG - Zlayo |
| 6 | 50 | WHEY - protein | | 6 | 5 | 7,5 | 25 | | OPG - Zlayo |

Izvor: autor

Nakon dostave zaključnih tablica, daljnje ispravljanje ili nadopuna podataka obavljat će se isključivo unosom kroz aplikaciju. Početno stanje zaliha robe također će se unijeti kroz aplikaciju na temelju inventure koju će klijent napraviti prije isporuke aplikacije.

# 5. SPECIFIKACIJA

Podaktivnosti faze specifikacije odnose se na studiju izvodljivosti, otkrivanje i analizu zahtjeva, utvrđivanje zahtjeva i validaciju zahtjeva. Obradit će se samo klijentski zahtjevi i podijelit će se u dvije grupe, kao funkcionalni i nefunkcionalni. Funkcionalni i nefunkcionalni zahtjevi prikupljeni su prema analizi postojećeg poslovanja.

## 5.1. Funkcionalni zahtjevi

Funkcionalni zahtjevi govore o tome koje bi usluge trebao obavljati sustav, koje bi rezultate sustav trebao proizvesti na temelju unesenih podataka te način obrade i prikaza podataka na osnovi uvjeta koji su predani sustavu.

Prema analizi, prepoznati su sljedeći funkcionalni zahtjevi:

1. Ulazak u aplikaciju bit će omogućen isključivo pomoću korisničkog imena i lozinke.
2. Aplikacija mora omogućiti unos, ispravak, brisanje, pregled podataka za kupce, dobavljače, proizvođače, primke, otpremnice, korisnike aplikacije te šifrante mjesta, mjera, statusa i uloga.
3. Pregled osnovnih podataka na glavnoj formi aplikacije – za primke, otpremnice i proizvode potrebno je prikazati broj dokumenta, datum izdavanja i šifru operatera, ukupnu vrijednost, ukupni PDV, prosječnu nabavnu cijenu te prosječnu prodajnu cijenu, a za nefinancijske dokumente, dakle, kupce, dobavljače i proizvođače, najviše je potrebno prikazati ime, prezime i kontakt-podatak. Više od toga nije potrebno, ali nije ni zabranjeno.
4. Korištenje i rad aplikacije bit će dozvoljeni samo zaposlenicima koji će imati dozvolu za rad – operaterima, uvid u preglede podataka imat će zaposlenici sa statusom zaposlenika, a izmjene matičnih podataka (šifranata) bit će vidljive samo voditelju skladišta sa statusom admina.
5. Temeljni izvještaji za svaku izrađenu primku i otpremnicu kao i skupni izvještaj za sve primke i otpremnice moraju biti dostupni s glavne forme, ali i s mjesta za unos primki i otpremnica.
6. U radu se moraju poštovati načela zakonitosti i općih računovodstvenih načela (tako da temeljna „matematika štima“, cit. korisnika).

## 5.2. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi su oni koji se ne tiču rada same aplikacije, već njezina međudjelovanja s operativnim sustavom na koji je instalirana. To označava realna ograničenja sustava kao što su brzina, pouzdanost, zauzeće memorije, poštovanje standarda u obradi poslovnih procedura i slično. Na primjer, korištenje programa isključivo na Microsoft Windows operativnom sustavu, nadzor nad CLR[5] sustavom u svrhu upravljanja memorijom na .net platformi i slično.

Prema analizi, prepoznati su sljedeći funkcionalni zahtjevi:

1. Da postoji grafička istoznačnost u izgledu svih ekrana za lakše korištenje zaposlenicima koji nisu iskusni u radu s računalima niti informatički iskusni.
2. Aplikacija će se razviti kao desktop-aplikacija, dakle, bit će dostupna samo na računalu na kojem je instalirana s lokalnom bazom podataka.
3. Aplikacija se sastoji od baze podataka Microsoft Access i formi korisničkog sučelja.
4. Sustav će se kodirati pomoću programskog jezika C# i izvještajnog sustava unutar aplikacije MS Visual Studio Community Edition 2015.
5. Sigurnost rada u aplikaciji omogućit će se unosom korisničkog imena i lozinke koje će u sustav unijeti administrator sustava, odnosno voditelj skladišta.
6. Aplikacija će se snimiti na računalo klijenta opremljenog s najmanje 2 GB radne memorije, najmanje Windows 7 operativnim sustavom i najmanje 1 GB prostora na tvrdom disku.

[5] CLR – (engl. *Common Language Runtime*), https://en.wikipedia.org/wiki/Common_Language_Runtime (Wikipedia, 13. 8. 2017.)

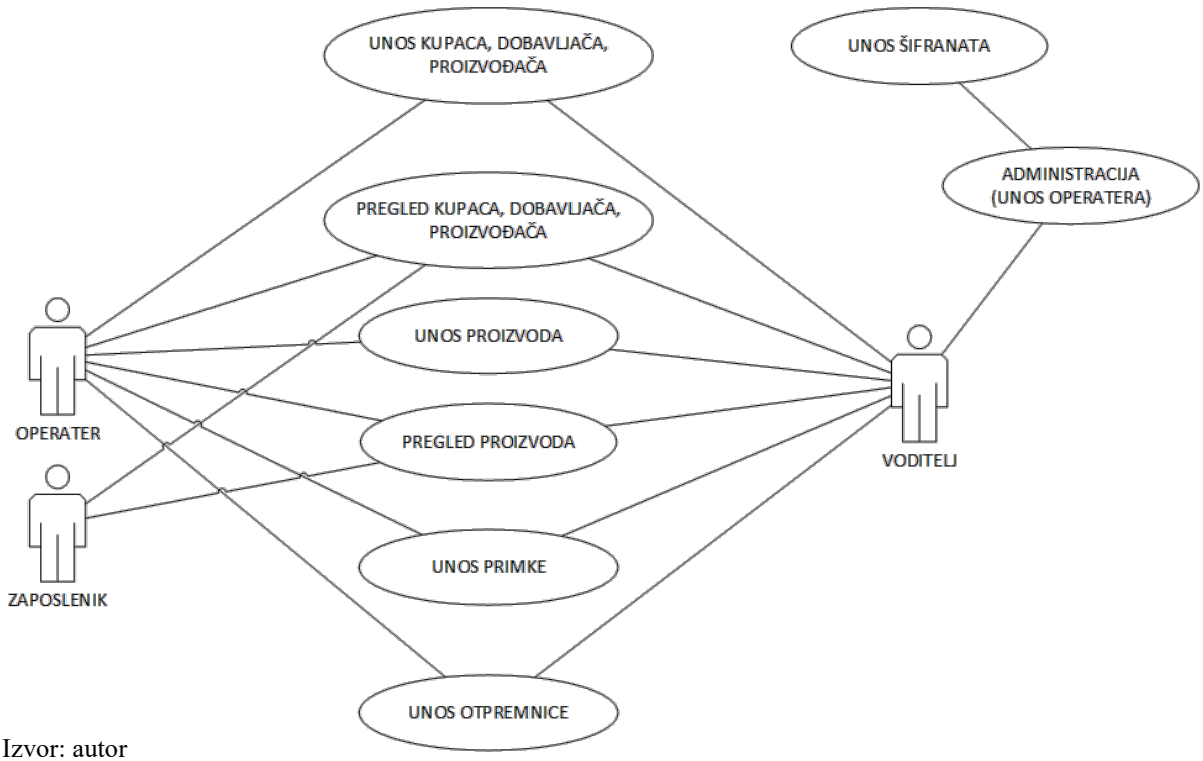
## 5.3. Modeliranje sustava programske podrške

Nakon ustanovljenih zahtjeva i razgovora s klijentom te prikupljenih početnih podataka, važno je krenuti prema prvom koraku realizacije razvoja programske podrške. Za svaki kvalitetan početak bilo kakvog razvoja neophodno je posjedovati kvalitetan plan. Planiranje u ovoj fazi odnosi se na grafički prikaz topografije sustava lišenog detalja i samo s naglascima na ključne poslovne procese i zadatke te njihove međuveze. Tako se znatno olakšava komunikacija s korisnikom zato što grafički prikaz smanjuje složenost prikaza sustava u cjelini.

Za potrebe modeliranja sustava u ovoj fazi koristi se UML[6]. UML sadrži brojne dijagrame za predstavljanje modela sustava, a u ovom će se radu koristiti dijagram slučajeva korištenja[7], dijagram slijeda[8] i dijagram aktivnosti[9].

Dijagram slučajeva korištenja na Slici 5.1. pripada statičkom pregledu uloga glumaca[10] (ovdje su to operater, zaposlenik i voditelj) i njihovih odnosa u slučajevima korištenja unutar sustava.

Slika 5.1. Dijagram slučajeva

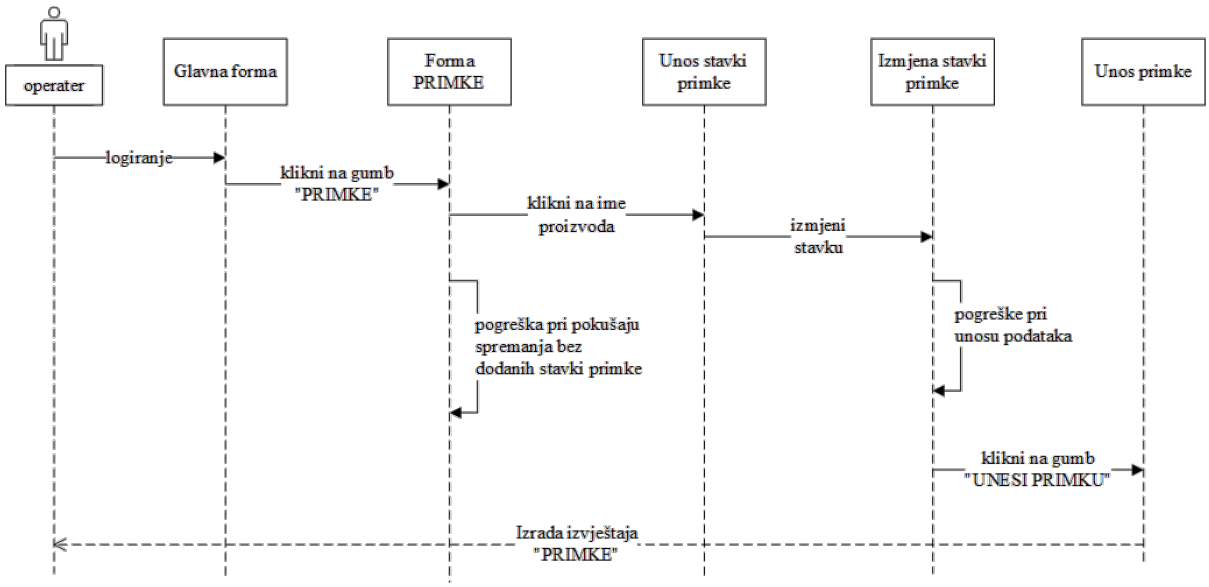



Izvor: autor

[6] Ibid. str. 8.
[7] Dijagram slučajeva korištenja (aut. prijevod s engl. – *Use Case diagram*).
[8] Dijagram slijeda (aut. prijevod s engl. – *Sequence diagram*).
[9] Dijagram aktivnosti (aut. prijevod s engl. – *Activity diagram*).
[10] Glumci (engl. *Actors* – specijalan tip klase).

Dijagram slijeda na Slici 5.2. prikazuje tijek operacija koje se zbivaju tijekom poslovne procedure obrade primke te poruke i konačni rezultat. Ovaj interakcijski tip dijagrama naglašava međudjelovanje operatera s programskom podrškom i dijelovima programske podrške međusobno.

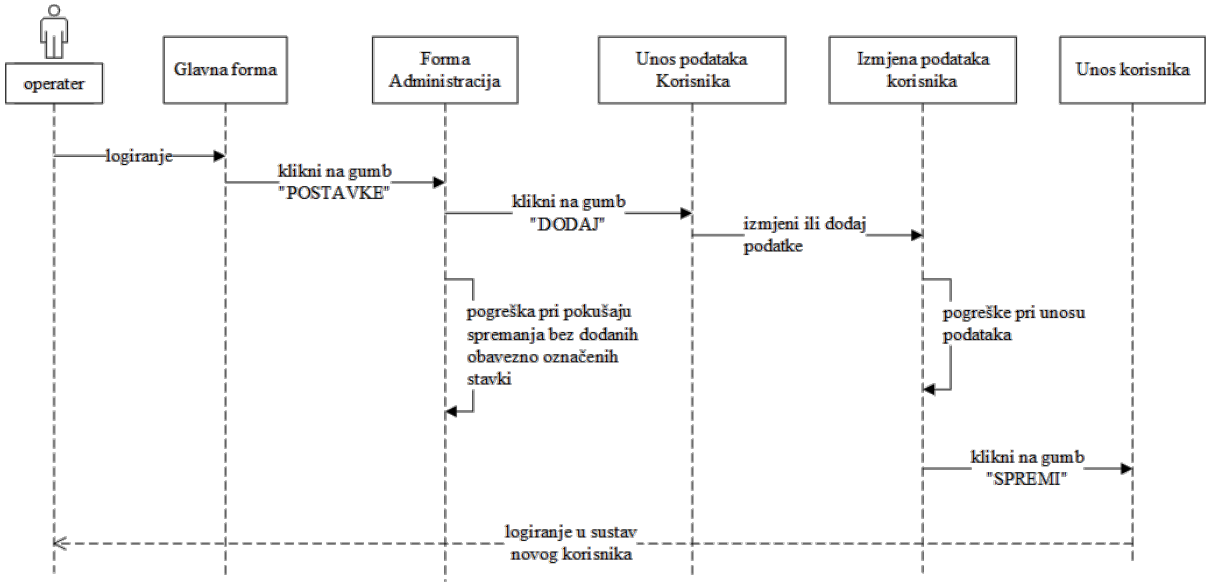
Slika 5.2. Dijagram slijeda – unos primke




Izvor: autor

Drugačiji primjer dijagrama slijeda prikazuje važnu funkciju u poslovnom procesu – dodavanje novoga korisnika programske podrške kako prikazuje Slika 5.3.

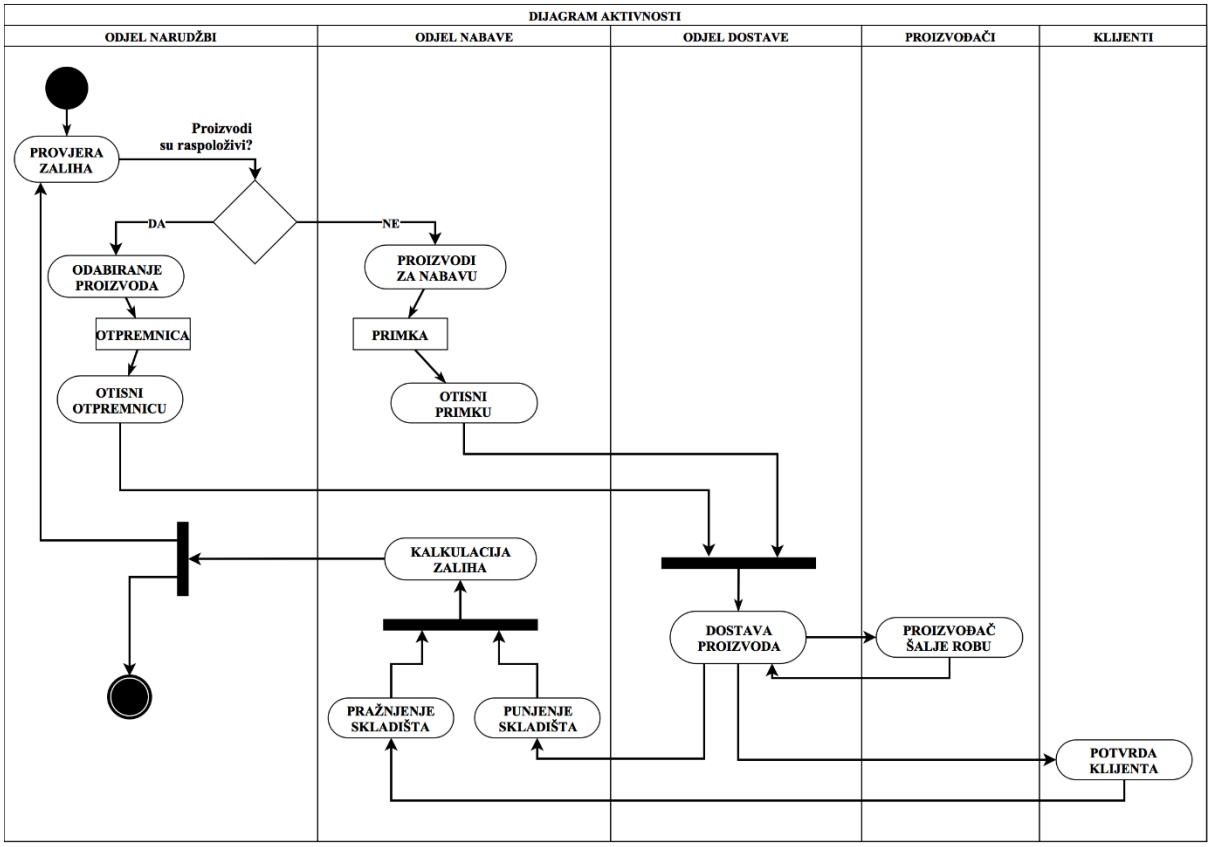
Slika 5.3. Dijagram slijeda – dodavanje novoga korisnika




Izvor: autor

Dijagram aktivnosti na Slici 5.4. prikazuje tijek aktivnosti s više detalja i logičku organizaciju programske podrške na razini jednog ili svih poslovnih procesa. Ovaj dijagram je izrazito koristan i pri samom početku izrade programske podrške zbog jasnog i jednostavnog prikaza tijeka poslovnih procesa.

Slika 5.4. Dijagram aktivnosti




Izvor: autor

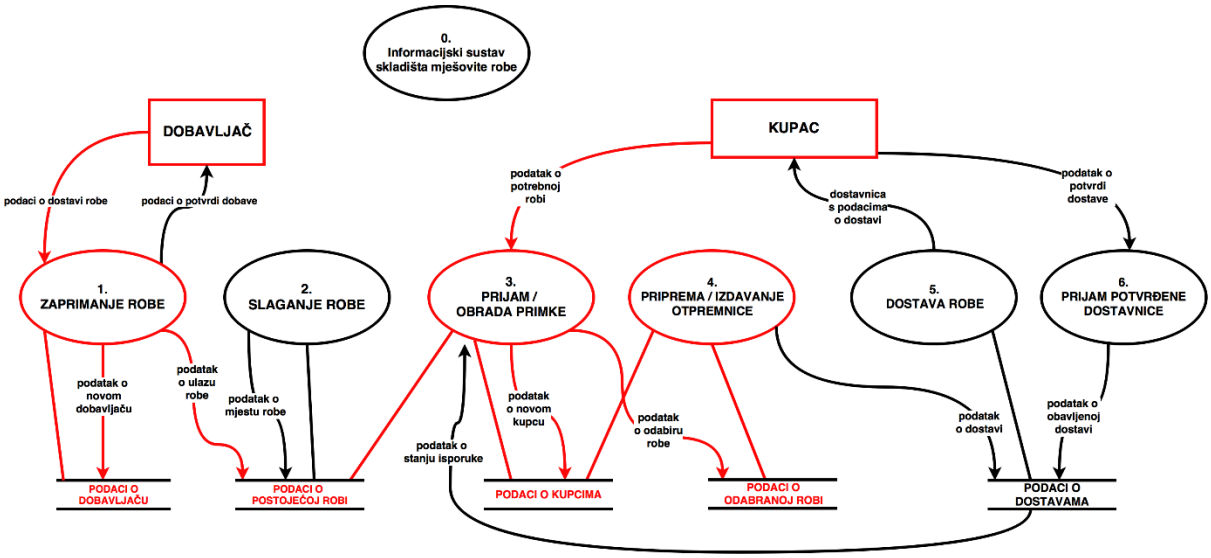
# 6. OBLIKOVANJE I IMPLEMENTACIJA

Nakon što je specifikacijom utvrđeno što sustav mora raditi, odnosno koji se poslovni procesi obuhvaćaju programskom podrškom, potrebno je detaljnije razraditi na koji način će se svi zadaci ispuniti. Kako će sustav raditi, ovisi o fazi oblikovanja u najmanje četiri značajnije stavke, a to su model procesa, model podataka, obrada sučelja između dijelova te dizajn korisničkog sučelja. Da bi oblikovanje bilo uspješno, potrebno je opisati svaki pojedinačni izdvojeni proces na gotovo atomaran[11] način. Temeljna pretpostavka za spomenuti opis procesa je kvalitetna dekompozicija sustava, odnosno rastavljanje sustava na manje jedinice procesa čime se savladava složenost razmatranog sustava.

## 6.1. Opći model poslovnih procesa – dijagram toka podataka

Opći model procesa na jednostavan i pregledan način pojednostavljuje otkrivene poslovne procese koji su opisani i uvršteni u specifikaciju. Prema zahtjevu korisnika, dio poslovnih procesa obuhvatit će se programskom podrškom, a dio neće. Crvenom bojom označeni su oni procesi koji će se razmatrati u razvoju programske podrške. Pogledati Prilog 1. za veći prikaz.

Slika 6.1. Dijagram toka podataka



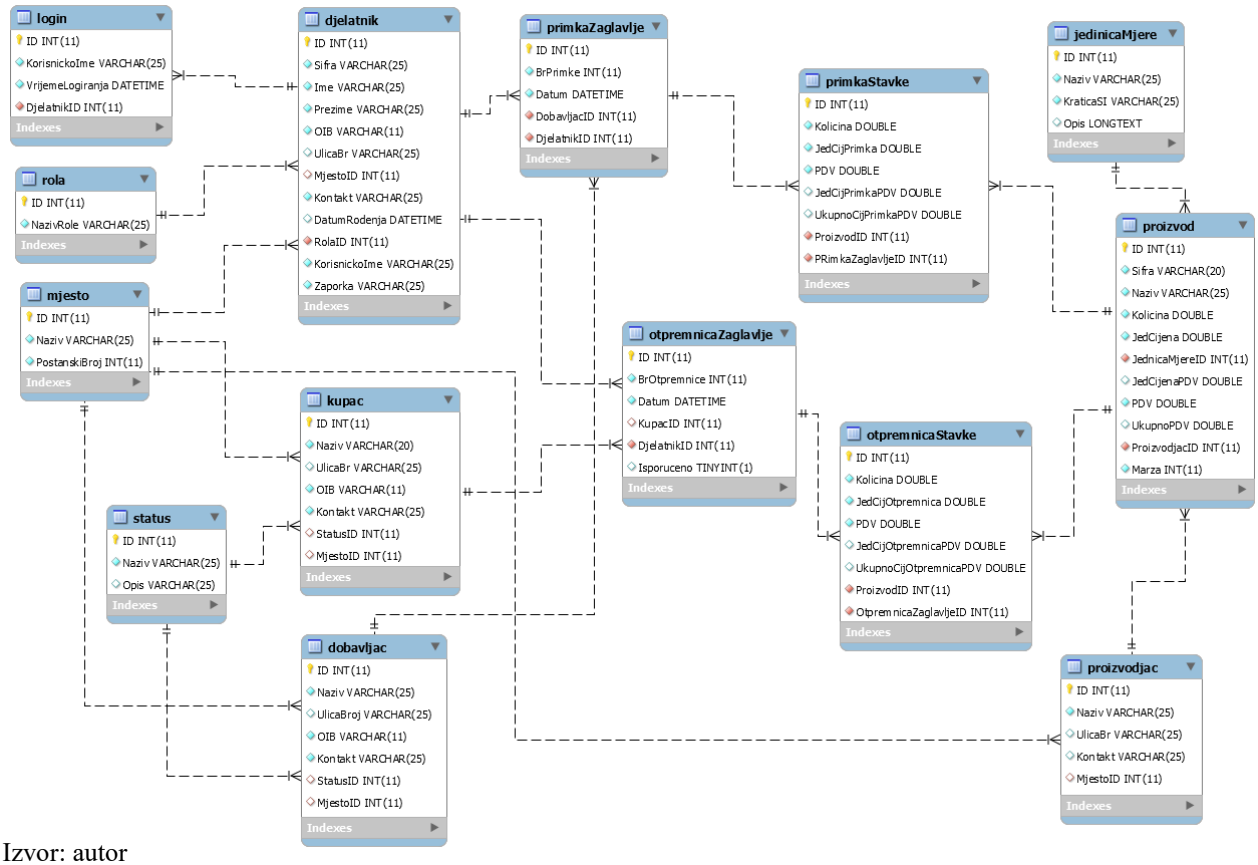

Izvor: autor

[11] Atomaran (grč. *ἄτομος* (*átomos*)) – onaj koji se ne može dijeliti u manje dijelove; nedjeljiv. U ovom smislu poslovni procesi koji su toliko jednostavni i samorazumljivi svim dionicima razvoja programske podrške da ih nije potrebno detaljnije opisivati ili raščlanjivati. To ne znači da se ne mogu dalje raščlanjivati, ali time bi se složenost nacrta arhitekture sustava povećala umjesto smanjila.

## 6.2. Model podataka – relacijski model

Kvalitetno sastavljen opći model procesa predstavljen dijagramom toka podataka nadalje se obrađuje i na temelju dobivenih informacija izrađuje se dijagram relacijskog modela podataka, takozvani ER dijagram (engl. *ER – Entitiy Relationship* – odnos entiteta) prikazan na Slici 6.2.

Slika 6.2. Relacijski (ER) dijagram (Pogledati Prilog 2. za veći prikaz)




Izvor: autor

Dijagram relacijskog modela podataka koji je sravnjen s tri osnovne *normalne forme* (Riordan, 2005) sadržava niz važnih podataka u razvoju programske podrške i to kako slijedi:

1. prikaz poslovnih entiteta – prikazani su u tablicama

2. prikaz odnosa među entitetima i *logiku*[12] tih odnosa – prikazani su linijama koje ih spajaju

3. tipovi podataka – prikazani u tabličnim poljima s podacima o veličini polja i nazivu.

[12] U ovom kontekstu *logika odnosa* označava postavku relacije među entitetima, odnosno radi li se o odnosu jedan prema jedan, više prema više, više prema jedan ili jedan prema više kao i razloge zašto su relacije tako postavljene.

## 6.3. Oblikovanje korisničkog sučelja

Iako oblikovanje korisničkog sučelja, strogo uzevši, pripada fazi oblikovanja, u praksi se često koriste crteži ili mulaže za prikaz „ekrana“ koje će programska podrška sadržavati kako bi se već na početku specifikacije razriješila razina prihvatljivosti aplikacije korisniku te kako bi se znatan dio korisničkih zahtjeva savladao već u najranijoj fazi. Oblikovanje korisničkog sučelja tijekom komunikacije s korisnikom može uputiti na važne stavke unutar poslovnih procesa koje bi trebalo poboljšati ili izmijeniti i time znatno skratiti vrijeme i olakšati način modeliranja podataka. Ova faza je toliko bitna da je moguće cijeli dio projekta posvetiti, posebice pri izradi većih i zahtjevnijih sustava, izradi prototipa. Rezultat prototipiranja je početna, ograničeno djelatna verzija, manjih pojedinačnih funkcionalnosti programske podrške u cilju prikaza rada spomenutih funkcionalnosti. Usklađivanje rezultata prototipiranja s krajnjim korisnikom važna je stavka dobro organiziranog razvoja programske podrške jer rješava cijeli niz kasnijih korisničkih problema u radu. Dapače, preporuča se da brzo prototipiranje s uključivanjem krajnjega korisnika bude uvijek način razvoja programske podrške (Sommerville, 2009), ali se autor u tom dijelu slaže samo djelomično i preporuča takav način samo pri razvoju većih sustava programske podrške ili sustava gdje programska podrška imperativno prati, korisničkim sučeljem, obveze prema zakonu[13].

Za potrebe razvoja aplikacije za skladišno poslovanje u primjeru koji koristi ovaj rad izrađeni su prijedlozi korisničkog sučelja uključivanjem korisnika u svakoj fazi razvoja. Izrada sučelja podijeljena je u četiri faze koje prate razvoj cjelokupne programske podrške kako slijedi:

1. faza izrade sučelja za kontakte aplikacije – Kupci, Dobavljači, Proizvođači

2. faza izrade sučelja za temeljno poslovanje – Proizvodi, Primke, Otpremnice

3. faza izrade sučelja za kontrolu aplikacije – Postavke, Šifranti

4. faza izrade sučelja izvještavanja – standardni izgled i sadržaj izvještaja.

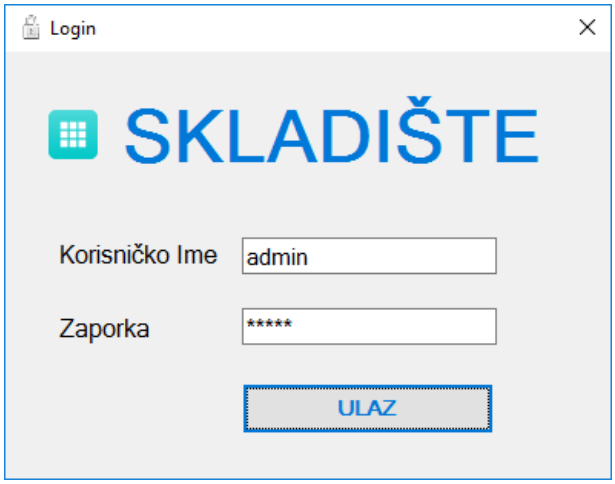
Nadalje će se prikazati samo najvažnija rješenja korisničkih sučelja aplikacije važna za rješavanje ključnih korisničkih zahtjeva.

[13] U spomenutom slučaju prototipiranje mora proizvesti apsolutno precizno korisničko sučelje jer u suprotnom može prouzročiti štetu zbog protuzakonitog rada.

### 6.3.1. Sučelje za prijavu u aplikaciju, osnove sigurnosti sustava

Forma za prijavu u aplikaciju je vrlo jednostavna i osnovne elemente čine dva polja za upis (tekstualne kutije[14]) od kojih ona za upis zaporke poštuje osnovno pravilo sigurnosti, odnosno sakrivanja podataka pa se upisivanje lozinke prikazuje znakom zvjezdice (*).

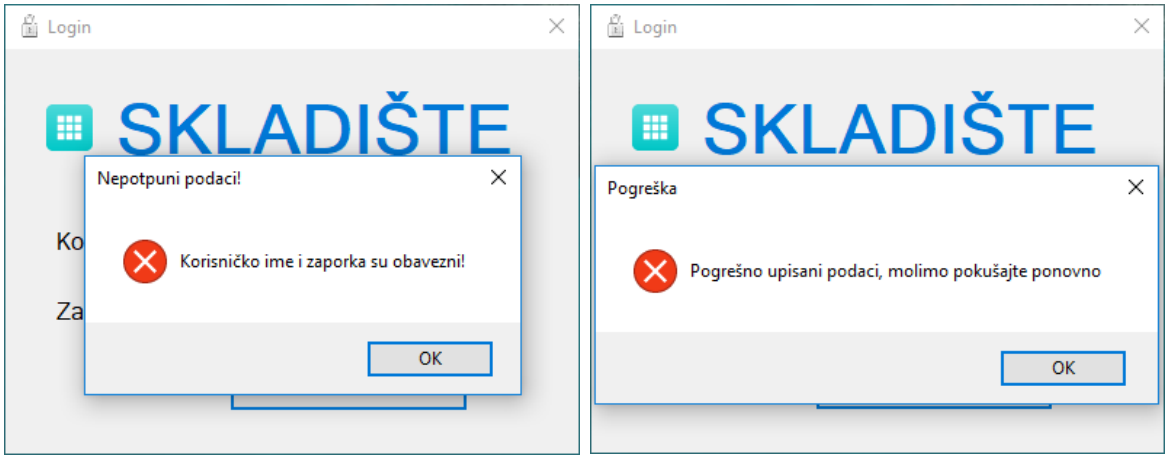
Slika 6.3 Primjer forme za ulazak u aplikaciju




Izvor: autor

Sučelje za prijavu u aplikaciju, kako je prikazano, rješava važno pitanje sigurnosti prijave koje je jedno od najvažnijih zahtjeva koji se postavljaju pri radu aplikacije. Ovdje je prikazana sposobnost aplikacije da provjeri u bazi podataka postoji li odobrenje za rad operateru te jesu li uneseno korisničko ime i zaporka ispravno upisani.

Slika 6.4. Primjeri mogućih pogrešaka tijekom prijave u aplikaciju



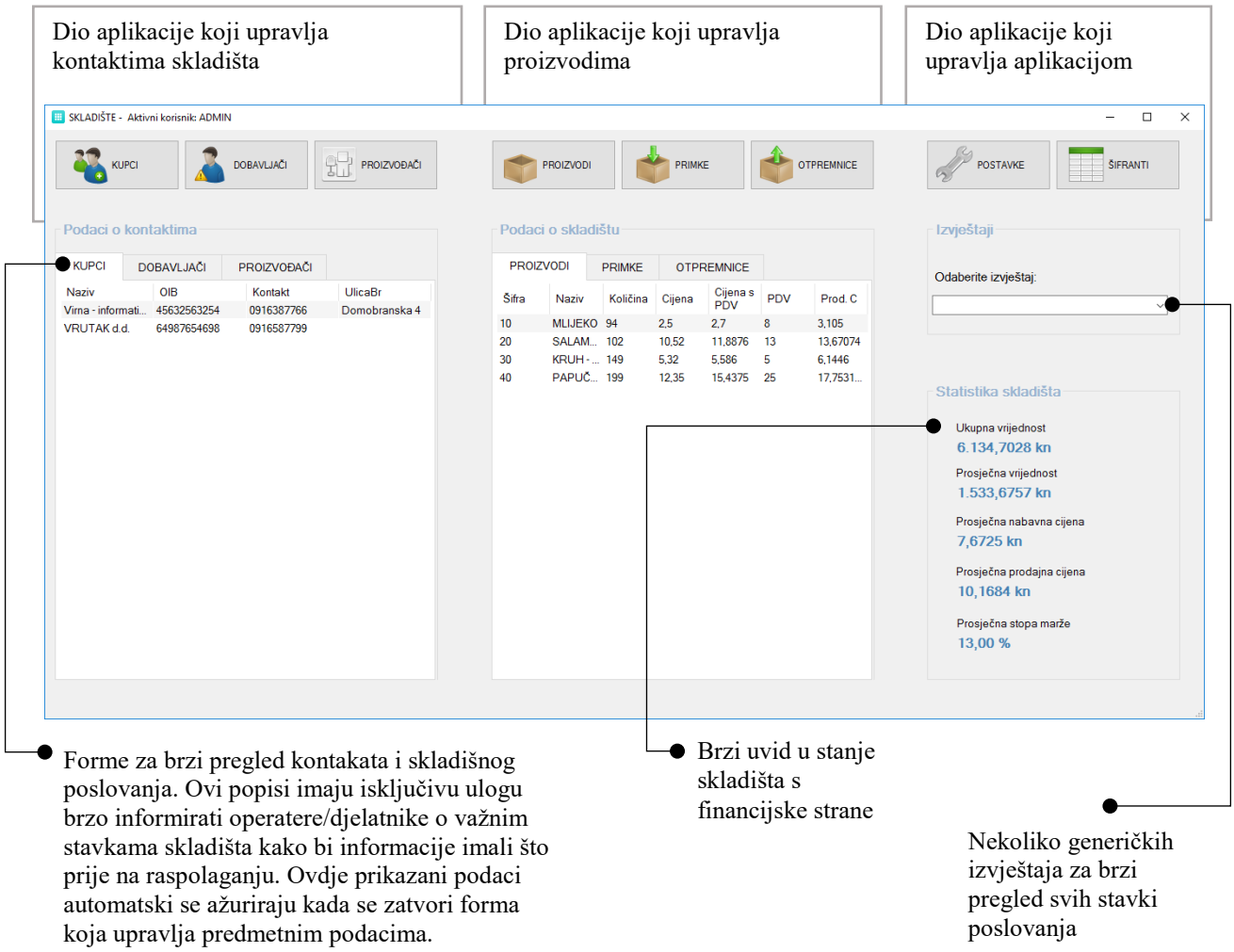

Izvor: autor

[14] Tekstualna kutija (engl. *Text box*), termin koji se koristi da bi se označilo mjesto za upisivanje podataka.

### 6.3.2. Početna forma programa, opći pregled funkcionalnosti

Nakon prijave u aplikaciju, otvara se početna forma aplikacije kako je prikazano na Slici 6.5.

Slika 6.5. Početna forma aplikacije




Izvor: autor

Na početnoj formi nalaze se postavljeni gumbi umjesto često korištenih izbornika[15] kojima se otvaraju forme poslovnih procesa skladišta. Na glavnoj formi nalaze se još priređeni izvještaji kao i osnovni izračuni stanja skladišta.

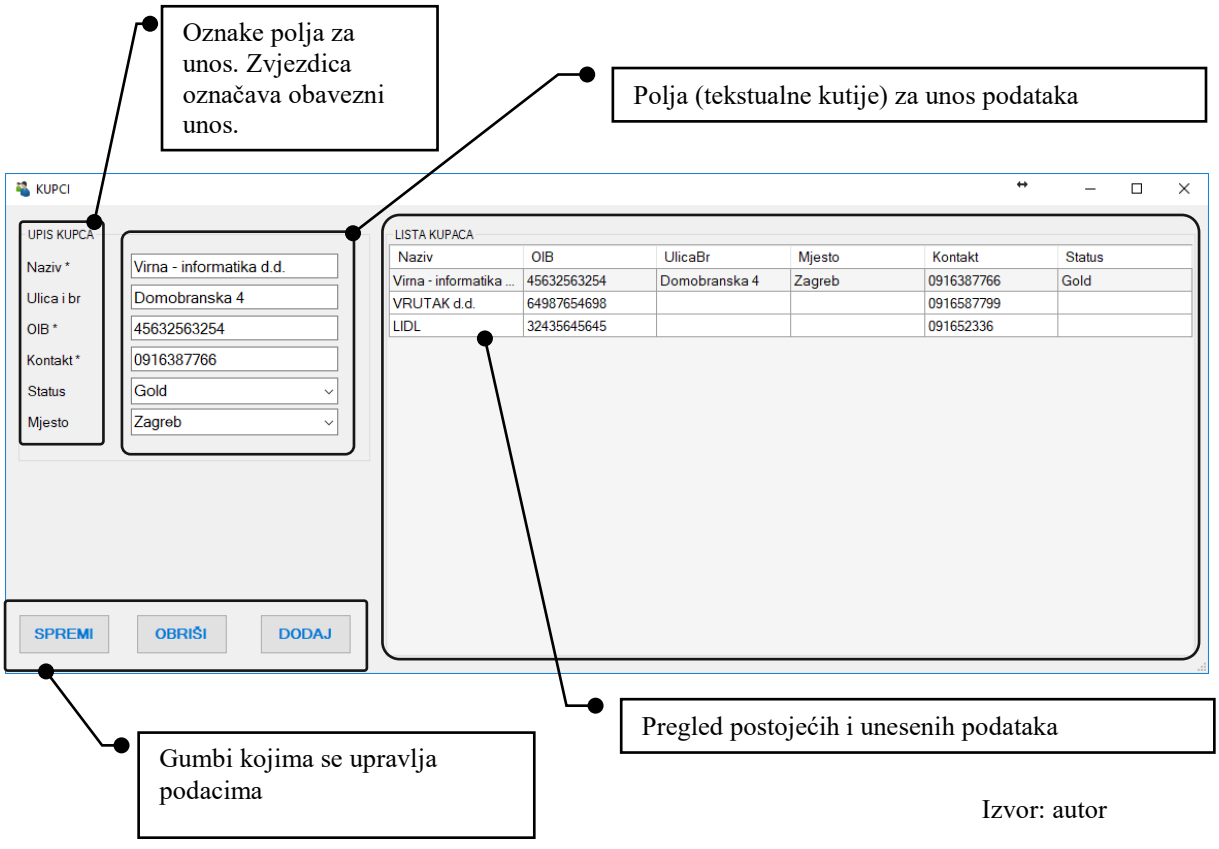
Cilj je na glavnoj formi dati odmah vidljive i ažurne podatke o poslovanju skladišta koji bi mogli biti od koristi operaterima brinući se o jednostavnosti, preglednosti i brzini dostupa informacijama s obzirom na to da je rad u skladištu često vrlo dinamičan posao.

[15] Izbornik (engl. *Menu)* označava specifičan način odabira ponuđenih mogućnosti koji nalikuje meniju.

### 6.3.3. Standardne forme aplikacije, postojanost podataka

Standardne forme za osnovne operacije unosa, ažuriranja, brisanja i kontrole podataka jednake su za unos proizvoda, proizvođača, dobavljača i kupaca. Jednoobraznost sučelja spomenutih formi olakšava snalaženje i utječe na brže savladavanje osnovnih pravila i ograničenja u radu aplikacije. Tako su sve forme podijeljene na tri osnovna dijela: mjesto s gumbima osnovnih operacija, mjesto za tablični prikaz unesenih stavki i mjesto za neposredni unos podataka u obliku tekstualnih polja i padajućih izbornika. Sve obavezne stavke pored imena stavke, a lijevo od prostora za unos, označene su znakom zvjezdice (*).

Slika 6.6. Izgled standardnih formi aplikacije




Izvor: autor

Standardne forme, kako je prikazano, vrlo konzervativno pristupaju slobodi upisa podataka što je važno ograničenje koje jamči da će upisani podaci biti ispravni. Ispravnost upisanih podataka je ključni uvjet kvalitetne obrade podataka kao i vjerodostojnosti sustava općenito, a aplikacija kojom se ne kontrolira kvalitetno proizvest će i loše izvještajne podatke[16].

[16] U informatičkom žargonu koristi se rečenica *Garbage in, garbage out.* U prijevodu: „Smeće ulazi, smeće izlazi“, a upućuje na poremećaj postojanosti podataka i rušenje vjerodostojnosti sustava u cjelini.

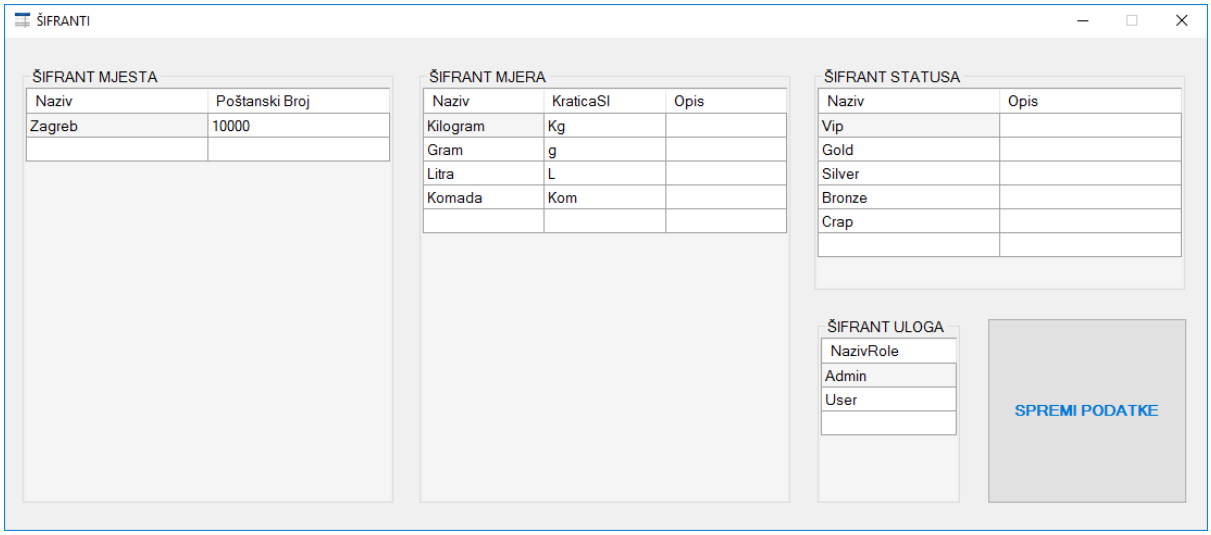
### 6.3.4. Forme poslovnih procesa

Spomenute forme prate postojeće poslovne procese koji su odabrani za obradu unutar programske podrške prema uputi i zahtjevu korisnika.

#### *Šifranti*

Da bi se pojednostavio unos i održavanje šifranata, korisnik upisuje sve podatke bez prekida te pritiskom na gumb „SPREMI PODATKE“ svi se unosi spremaju u bazu odjednom.

Slika 6.7. Forma šifranata

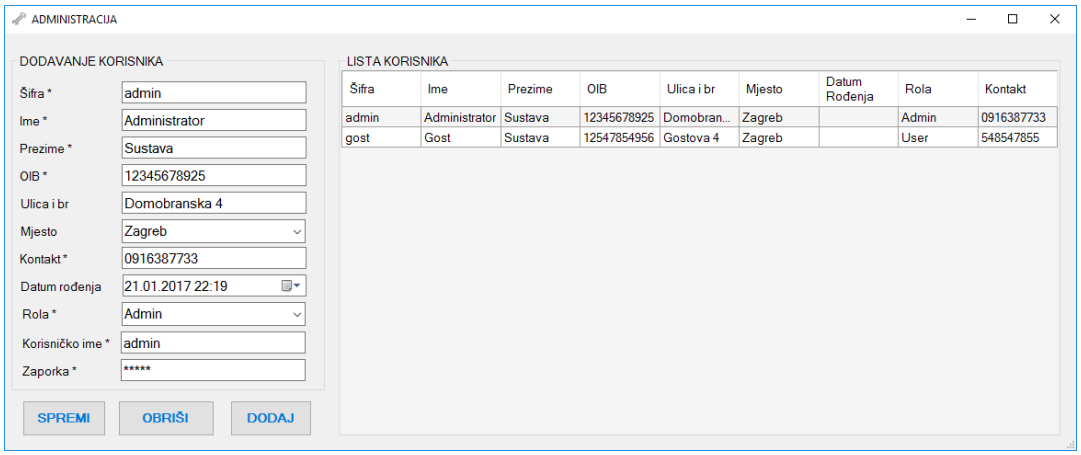



Izvor: autor

#### *Administracija*

Forma za unos podataka djelatnika s dodjelom korisničkog imena i zaporke.

Slika 6.8. Forma administracije aplikacije

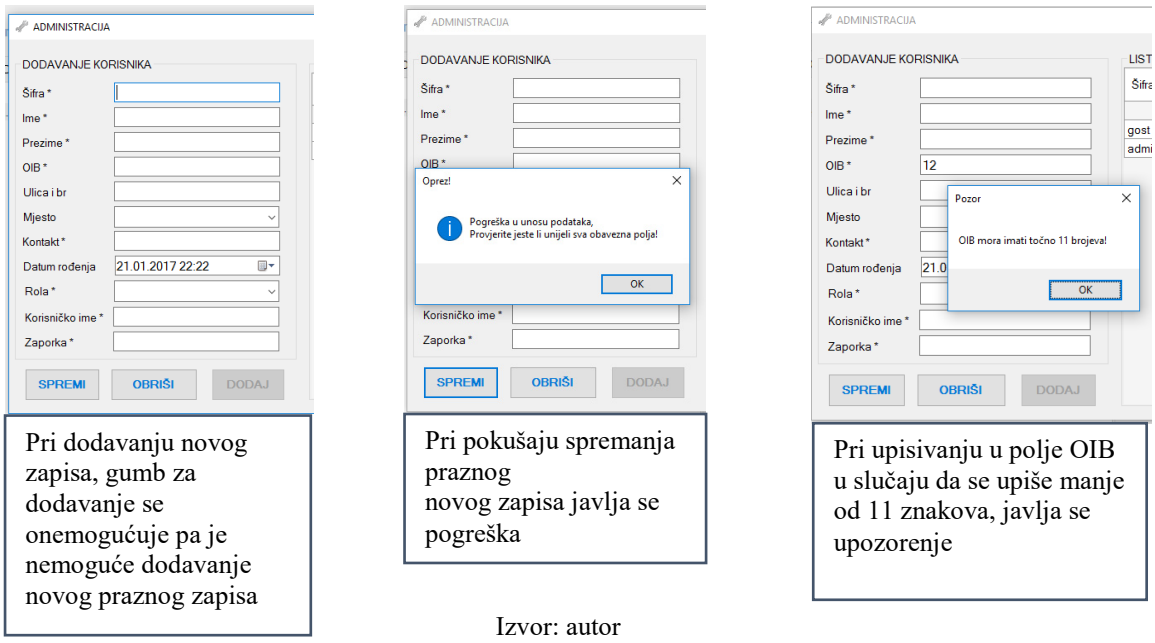



Izvor: autor

### *Primjer kontrole upisa te poruka među sučeljima aplikacije*

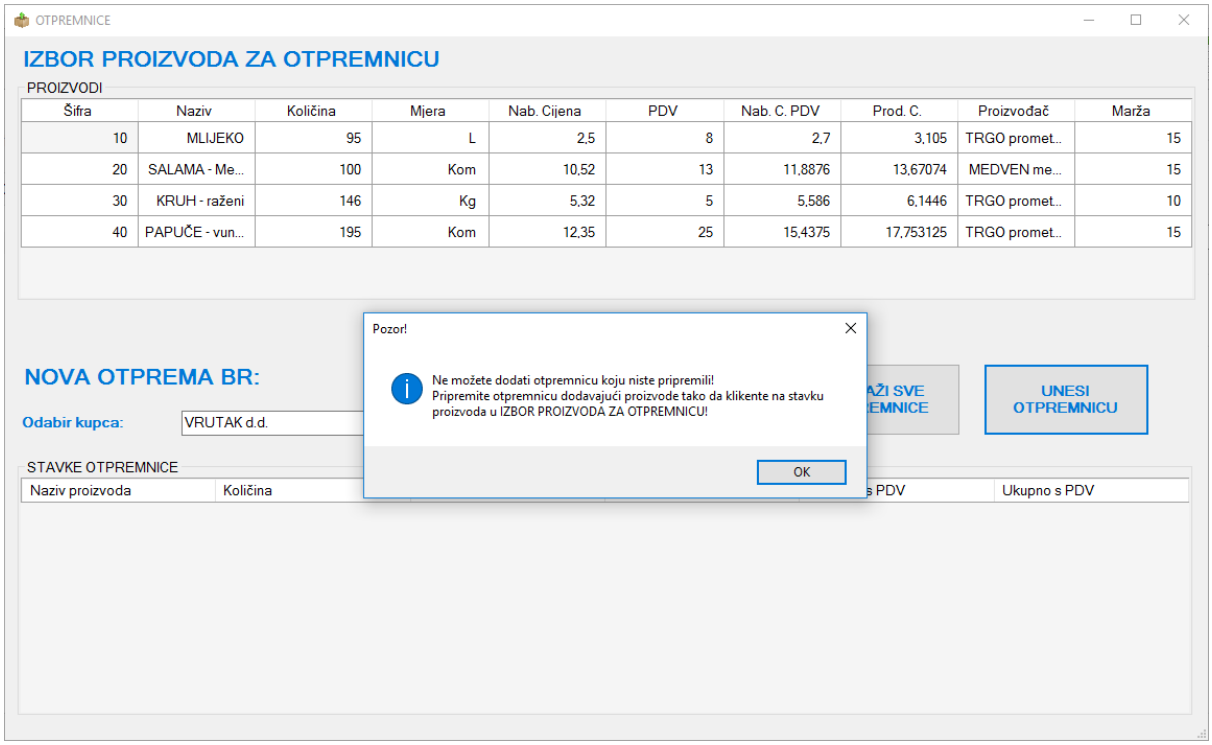
Kontrola upisa može se primjerom pokazati za polje OIB kao i za kontrolu poslovne procedure izdavanja otpremnica. U oba slučaja, neispravni ili nedostajući podaci pokreću upozorenja.

Slika 6.9. Primjer kontrole upisa podataka




Izvor: autor

Slika 6.10. Primjer kontrole poslovne procedure u izradi otpremnice

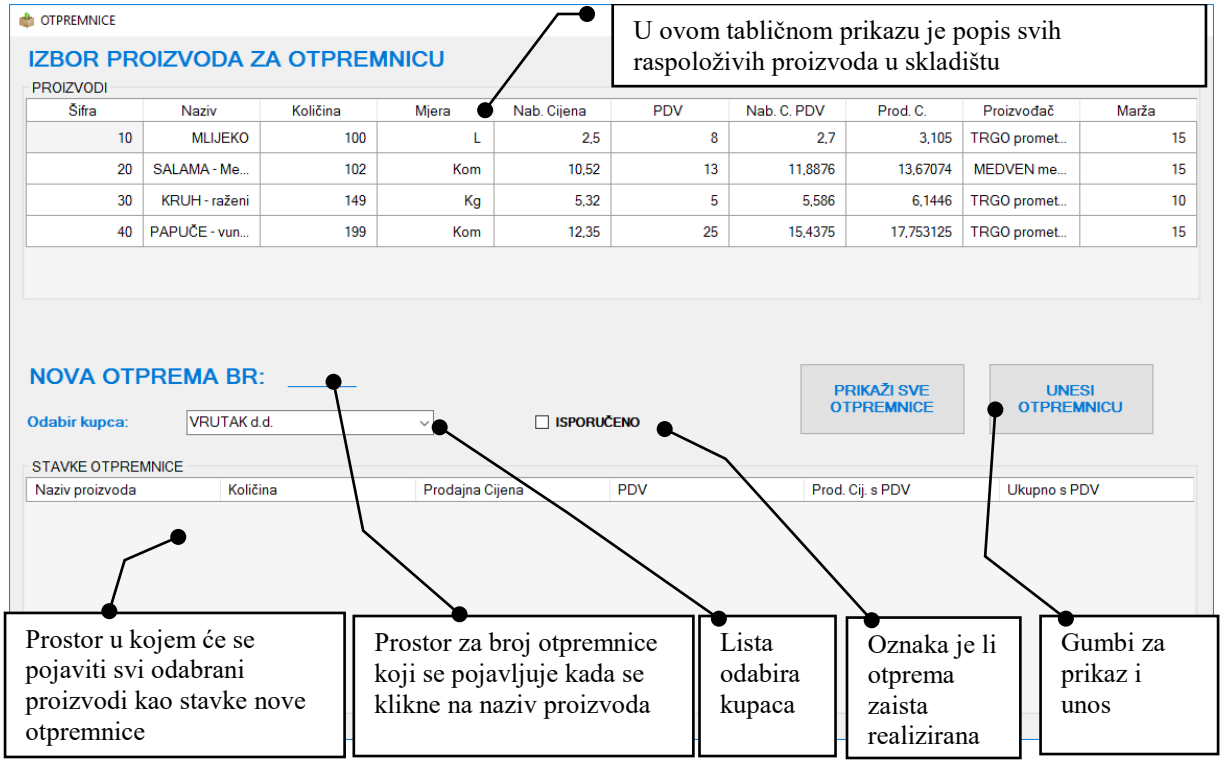



Izvor: autor

### *Otpremnice*

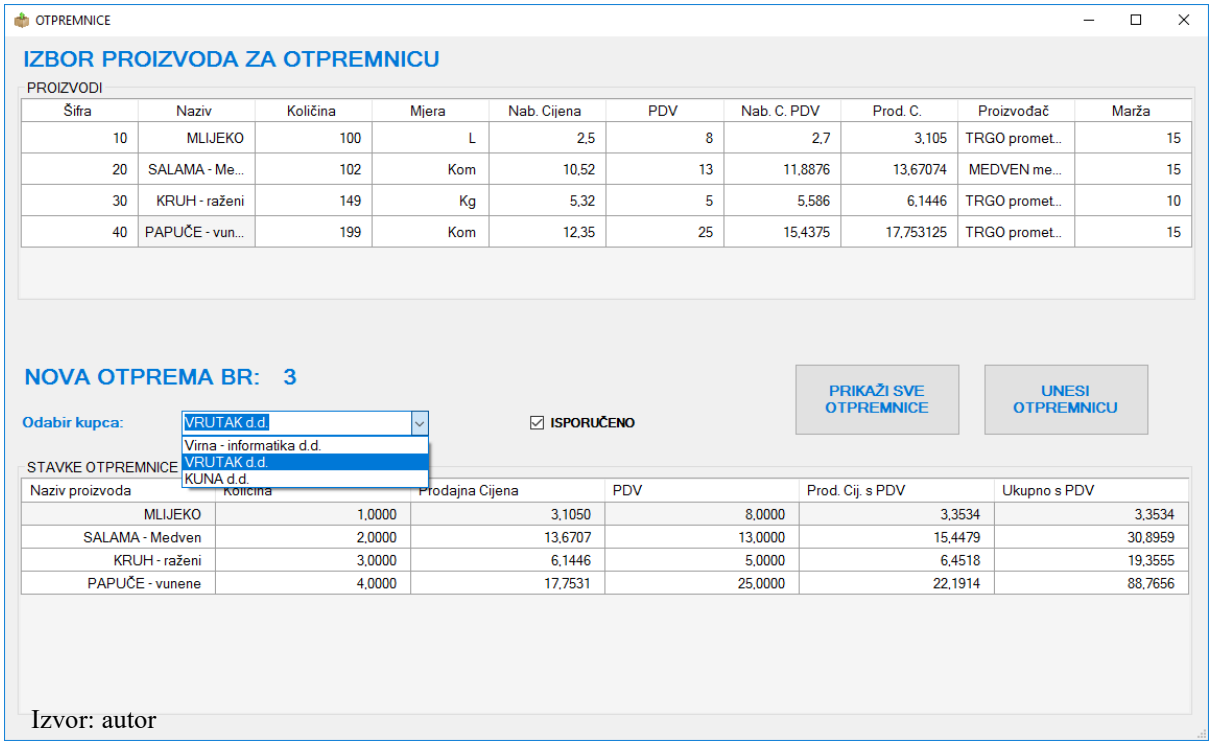
Na formi „Otpremnice“ unose se podaci sa skladišta u obliku „Otpremnica – račun“.

Slika 6.11. Forma otpremnica – objašnjenje sučelja




Izvor: autor

Slika 6.12. Izgled ispravno unesenih podataka otpremnice

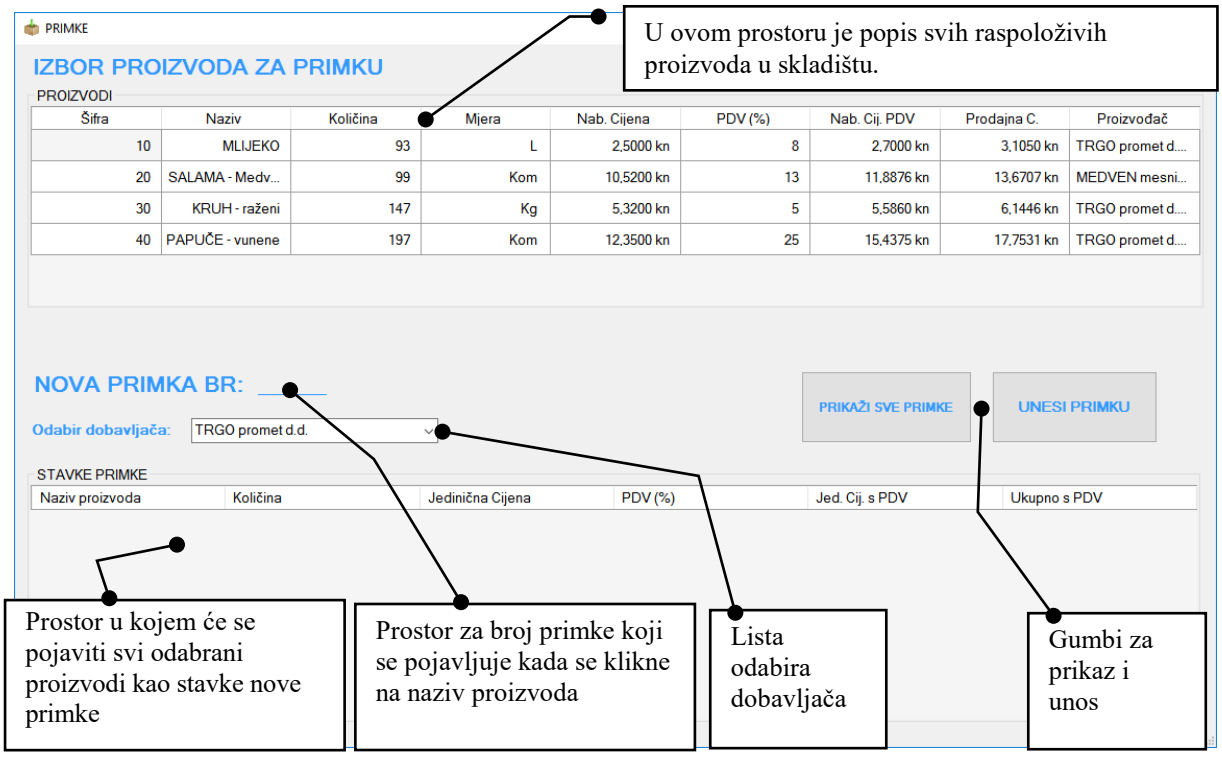



Izvor: autor

### *Primke*

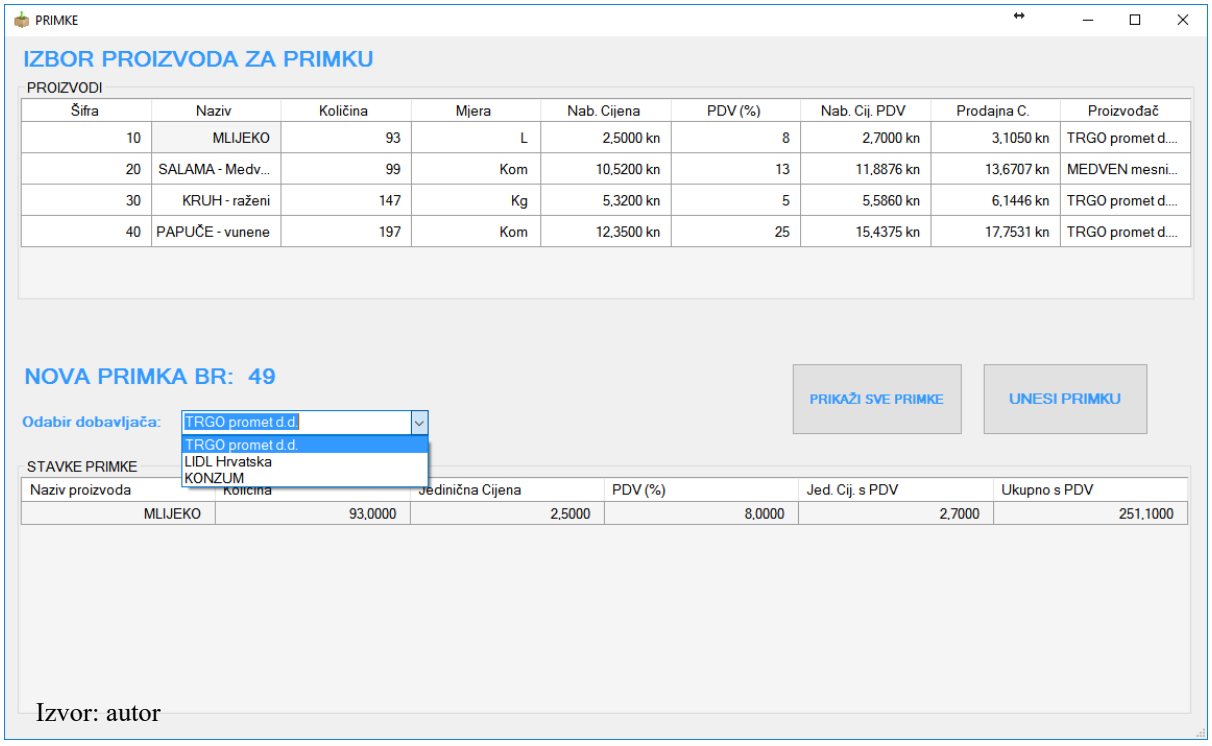
Na formi „Primke“ unosi se preuzimanje proizvoda u skladište kako je prikazano na Slici 6.13.

Slika 6.13. Forma primki – objašnjenje sučelja




Izvor: autor

Slika 6.14. Izgled ispravno unesenih podataka primke

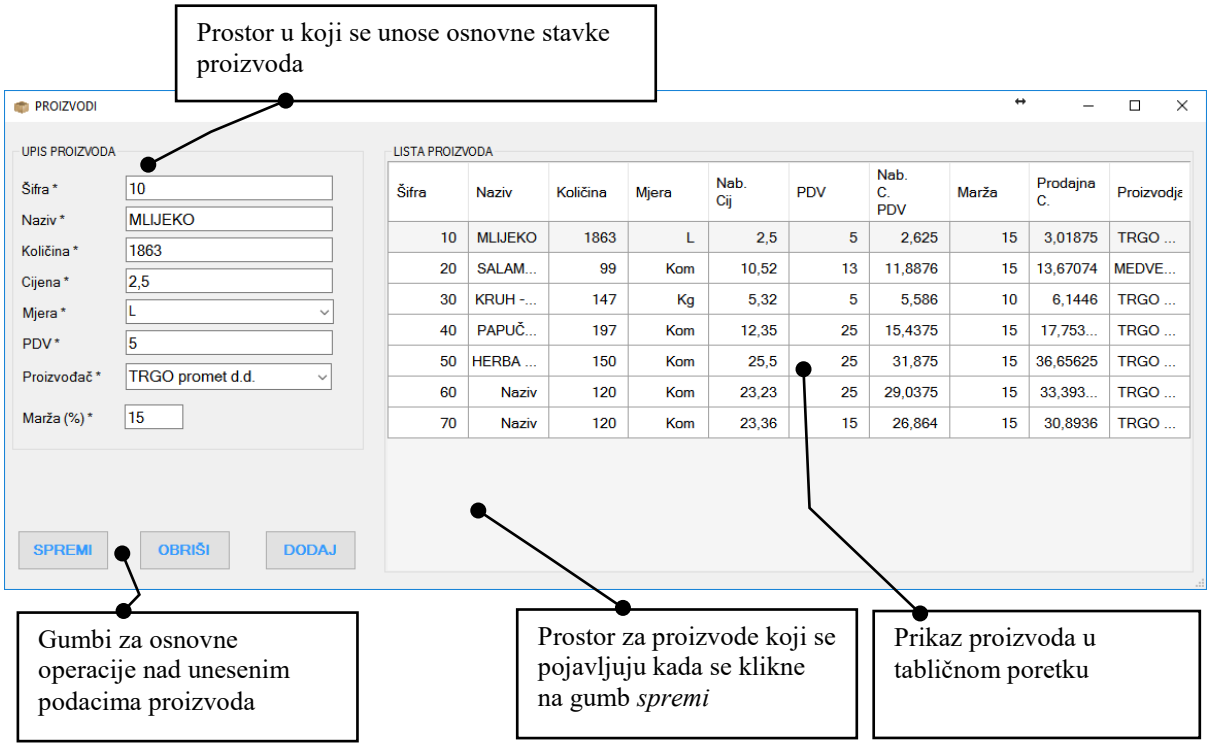



Izvor: autor

### *Proizvodi*

Na formi „Proizvodi“ upravlja se proizvodima u skladištu. To znači dodavanje novog te izmjenu ili brisanje postojećeg zapisa. Obavezna polja koja se moraju unijeti nakon imena imaju znak zvjezdice (*).

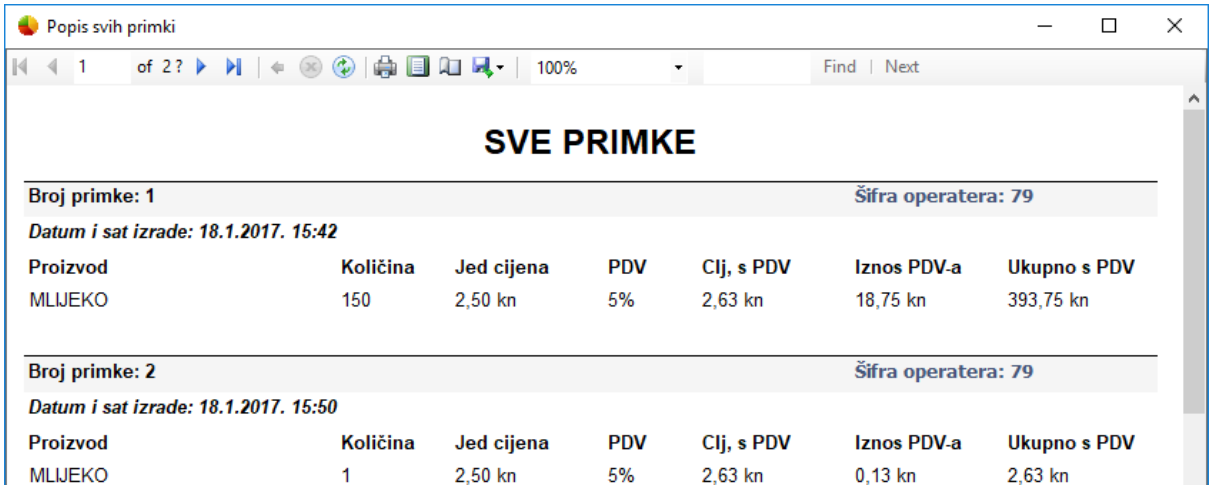
Slika 6.15. Prikaz standardnog izgleda formi za upis podataka




Izvor: autor

### *Primjer izvještaja*

Slika 6.16. Prikaz standardnog načina izvještavanja koje pruža aplikacija




Izvor: autor

## 6.4. Organizacija implementacije projekta

U ovom poglavlju ukratko će se prikazati organizacijski model vođenja projekta s naglaskom na tim koji će sudjelovati u razvoju aplikacije, terminski plan i prikaz implementacijskih koraka.

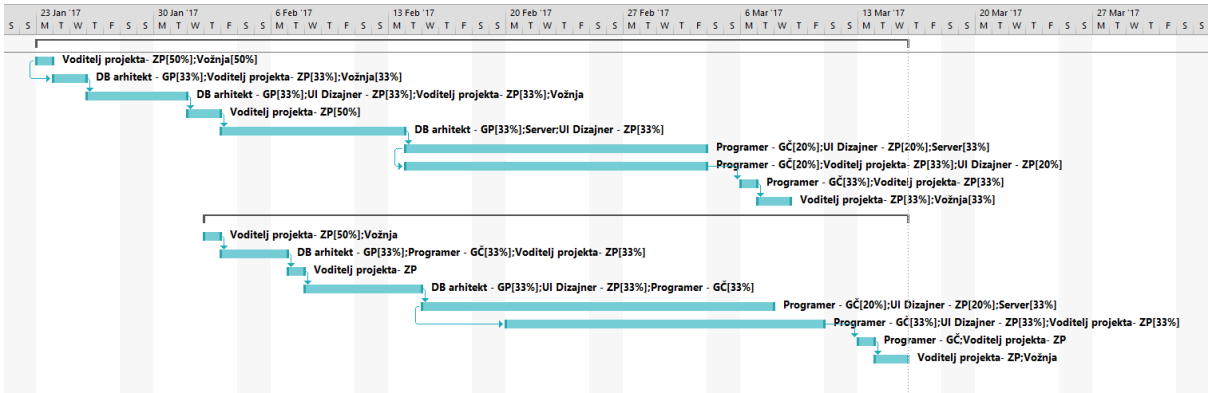
### *Projektni tim*

Nakon svih opisanih koraka u razvoju aplikacije skladišnog poslovanja važno je odrediti ljudske resurse s valjanom ekspertizom koji će implementirati aplikaciju. Za potrebe primjera u ovom radu će se uključiti jedan voditelj projekta (ZP), jedan inženjer baze podataka (GP), jedan razvojni inženjer (GČ) te jedan oblikovatelj korisničkog sučelja (ZP). Oblikovatelj korisničkog sučelja je ujedno i voditelj projekta.

### *Tijek implementacije – prikaz Ganttovim[17] grafikonom*

Ganttov grafikon (često nazivan i gantogram) je prikaz, vodoravnim stupičastim crtama, tijeka i položaja projektnih zadataka ovisnih o vremenu. Ovaj grafikon prikazuje – kako se vidi na Slici 6.17. – koji ljudski potencijali sudjeluju s kojim postotkom vremenskog udjela u savladavanju kojih zadataka, vremensku dimenziju savladavanja zadataka kao i ključne točke u savladavanju razvoja pojedinih zadataka ili grupe zadataka koji se nazivaju *milestone* (engl. *Milestone* – miljokaz, prekretnica). Uočiti na grafikonu dva inkrementa.

Slika 6.17. Ganttov dijagram (pogledati Prilog 3. za veći prikaz )




Izvor: autor

[17] Henry Gantt sastavio je ovaj grafikon na samom početku 20. stoljeća kao važan alat za prikaz upravljanja projektima. Gantt je bio američki inženjer mehanike i radio je u polju znanstvenog menadžmenta.

### *Terminska raspodjela projekta*

Terminska raspodjela zadataka odnosi se na vremensku dimenziju razvoja projekta u cjelini. Kako se može vidjeti na Slici 6.18., terminska raspodjela obuhvaća dva predviđena inkrementa s popisom zadataka te ostale bitne informacije za tijek projekta. Obratiti pozornost na to da svaki inkrement sadržava korake razvoja kako su objašnjeni u ovom radu prema temeljnim principima programskog inženjerstva.

Slika 6.18. Prikaz terminskog plana projekta

| | Task Mode | Task Name | Duration | Start | Finish | Predecessors | Resource Names |
|---|---|---|---|---|---|---|---|
| 1 | | **CRUD** | **38 days** | **Mon 23.1.17** | **Wed 15.3.17** | | |
| 2 | | crud - Uvodni satanak | 1 day | Mon 23.1.17 | Mon 23.1.17 | | Voditelj projekta- ZP[50%];Vožnja[50%] |
| 3 | | crud - Sastanak za tehničku specifikaciju | 2 days | Tue 24.1.17 | Wed 25.1.17 | 2SS+1 day | DB arhitekt - GP[33%];Voditelj projekta- ZP[33%];Vožnja[33%] |
| 4 | | crud - Analiza I specifikacija | 4 days | Thu 26.1.17 | Tue 31.1.17 | 3 | DB arhitekt - GP[33%];UI Dizajner - ZP[33%];Voditelj projekta- ZP[33%];Vožnja |
| 5 | | crud - Odobrenje specifikacije | 2 days | Wed 1.2.17 | Thu 2.2.17 | 4 | Voditelj projekta- ZP[50%] |
| 6 | | crud - Oblikovanje | 7 days | Fri 3.2.17 | Mon 13.2.17 | 5 | DB arhitekt - GP[33%];Server;UI Dizajner - ZP[33%] |
| 7 | | crud - Implementacija | 14 days | Tue 14.2.17 | Fri 3.3.17 | 6 | Programer - GČ[20%];UI Dizajner - ZP[20%];Server[33%] |
| 8 | | crud - Testiranje | 14 days | Tue 14.2.17 | Fri 3.3.17 | 7SS | Programer - GČ[20%];Voditelj projekta- ZP[33%];UI Dizajner - ZP[20%] |
| 9 | | crud - Integracija | 1 day | Mon 6.3.17 | Mon 6.3.17 | 8 | Programer - GČ[33%];Voditelj projekta- ZP[33%] |
| 10 | | crud - Distribucija klijentu | 2 days | Tue 7.3.17 | Wed 8.3.17 | 9 | Voditelj projekta- ZP[33%];Vožnja[33%] |
| 11 | | **FUNKCIONAL** | **30 days** | **Thu 2.2.17** | **Wed 15.3.17** | | |
| 12 | | fnc - Sastanak s računovodstvom | 1 day | Thu 2.2.17 | Thu 2.2.17 | | Voditelj projekta- ZP[50%];Vožnja |
| 13 | | fnc - Analiza I specifikacija | 2 days | Fri 3.2.17 | Mon 6.2.17 | 12 | DB arhitekt - GP[33%];Programer - GČ[33%];Voditelj projekta- ZP[33%] |
| 14 | | fnc - Odobrenje specifikacije | 1 day | Tue 7.2.17 | Tue 7.2.17 | 13 | Voditelj projekta- ZP |
| 15 | | fnc - Oblikovanje kalkulacija I izvještaja | 5 days | Wed 8.2.17 | Tue 14.2.17 | 14 | DB arhitekt - GP[33%];UI Dizajner - ZP[33%];Programer - GČ[33%] |
| 16 | | fnc - Implementacija | 15 days | Wed 15.2.17 | Tue 7.3.17 | 15 | Programer - GČ[20%];UI Dizajner - ZP[20%];Server[33%] |
| 17 | | fnc - Testiranje | 15 days | Mon 20.2.17 | Fri 10.3.17 | 16SS+3 days | Programer - GČ[33%];UI Dizajner - ZP[33%];Voditelj projekta- ZP[33%] |
| 18 | | fnc - Integracija | 1 day | Mon 13.3.17 | Mon 13.3.17 | 17 | Programer - GČ;Voditelj projekta- ZP |
| 19 | | fnc - Distribucija klijentu | 2 days | Tue 14.3.17 | Wed 15.3.17 | 18 | Voditelj projekta- ZP;Vožnja |

Izvor: autor

Prvi inkrement je nazvan CRUD (akronim s engl. CRUD – *Create, Read, Update, Delete; izradi/unesi, čitaj, osvježi, izbriši*) odnosi se na izradu pripremne faze projekta u kojem se uz poslovne procedure savladavaju i osnovne mogućnosti koje će buduća programska podrška pružati u smislu manipulacije (unos, čitanje, osvježavanje i brisanje podataka), čuvanja i prikaza podataka. Ovo je podatkovno orijentirani inkrement u kojem se priprema relacijska baza podataka kao osnovni model programske podrške tipa modela s repozitorijem, zatim se prikupljaju osnovni podaci za punjenje baze te izgradnja temeljne arhitekture sustava s korisničkim i unutaraplikacijskim sučeljima.

Drugi inkrement naziva Funkcional odnosi se na sve poslovne procedure skladišne aplikacije koje imaju poveznicu s izračunima te je potrebno uspostaviti legalnu i proceduralnu kontrolu izračuna kao i ispravnost svih dokumenata koji proizlaze iz aplikacije, a prikazuju spomenute izračune. Važno je primijetiti da spomenuti inkrement počinje sastankom s odjelom/osobom računovodstvene struke kako bi se osigurao najkvalitetniji pristup rješavanju zadataka, ali i zadržala razina formalne procedure važne za računovodstveno izvještavanje.

# 7. VERIFIKACIJA I VALIDACIJA

Verifikacija podrazumijeva aktivnosti čiji je cilj utvrditi koliko su oblikovanje i implementacija u skladu sa specifikacijom i razvija li se programska podrška na ispravan način, a validacija provjerava koliko je programska podrška razvijena na način da zadovoljava stvarne potrebe krajnjih korisnika. To svakako ne znači da će izrađena programska podrška biti lišena apsolutno svih pogrešaka, već da će biti spremna za korištenje u produkcijskom, realnom sustavu te da će omogućiti rad s poslovnim procedurama za koje je programska podrška u početku i organizirana.

Metode koje se koriste pri validaciji i verifikaciji u načelu se mogu podijeliti u dvije temeljne grupe, a to su statička verifikacija i validacija te testiranje.

Statička validacija i verifikacija upražnjava se bez izvođenja programa kako bi se provjerili dokumenti, modeli sustava, oblikovanje i implementacija.

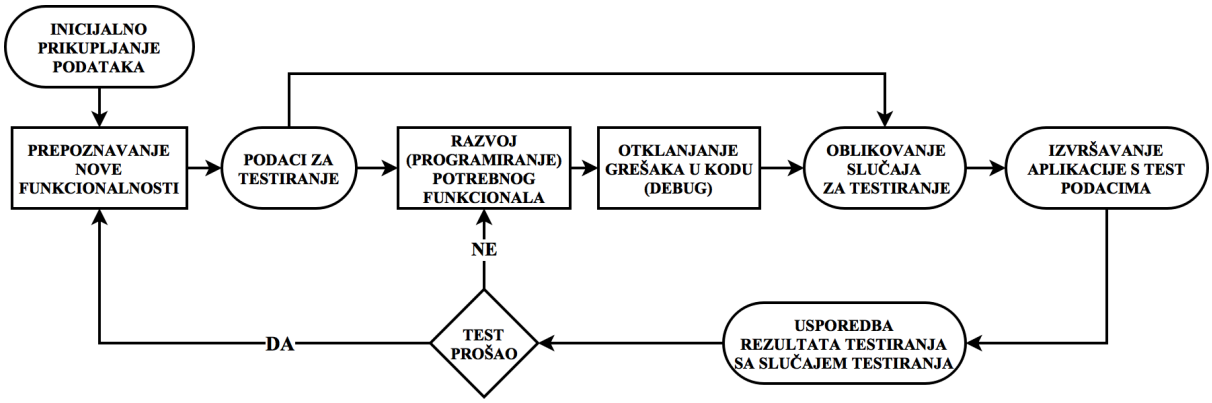
Testiranje, kao gotovo sinonim za verifikaciju i validaciju, najčešća je metoda verifikacije i validacije i upražnjava se pokretanjem programske podrške ili njezinih dijelova s izmišljenim ili stvarnim produkcijskim, ali pokusnim podacima. U ovom radu na primjeru skladišne aplikacije prikazat će se temeljna metodologija testiranja.

## 7.1. Testiranje skladišne aplikacije

U slučaju primjera skladišne aplikacije ovog rada još je u fazi analize dogovoreno prikupljanje podataka koji će sada poslužiti za testiranje. Za razvoj aplikacije skladišnog poslovanja općenito će se koristiti metoda testiranjem vođenog razvoja što je prikazano na Slici 7.1., a to znači da se razvoj skladišne aplikacije temelji na izmjenjivanju faze kodiranja i testiranja te se u novi razvojni ciklus implementacije kreće tek onda kada je prethodna funkcionalnost uspješno prošla fazu testiranja, odnosno kada je usporedba rezultata testiranja s oblikovanim slučajem za testiranje zadovoljavajuća.

Sažeti prikaz položaja testiranja u razvoju skladišne aplikacije prikazan je na Slici 7.1.

Slika 7.1. Testiranje, prikaz tijeka




Izvor: autor

U prvoj fazi testiranja testiraju se greške, odnosno, inicijalno prikupljeni podaci prilagođavaju se ovoj fazi testiranja tako da se otkriju pogreške koje upućuju na kritična odstupanja razvoja programske podrške od specifikacije. Prisustvo takvih pogrešaka otklanja se procesom koji se naziva *debugging* (engl. *debug* – otkloniti pogrešku ili *bug*[18]).

U sljedećoj fazi se, nakon otklanjanja razvojnih pogrešaka, formira oblikovanje slučaja za testiranje pa se i podaci za testiranje u ovoj fazi koriste bez prilagodbe preuzeti od klijenta te se nastavlja test s izvršavanjem aplikacije implementiranog funkcionala.

Ako je i ovaj implementacijski test prošao, razvojni ciklus kreće u prepoznavanje novog funkcionala, a ako nije, potrebno je ponovno pokrenuti fazu razvoja funkcionala, popraviti pogreške i odstupanja sve dok rezultat testiranja nije zadovoljavajući.

Važna kvaliteta razvoja vođenog testiranjem jest između ostaloga i regresijsko testiranje koje pruža mogućnost izvršavanja testa u bilo kojoj fazi razvoja programske podrške kako bi se otklonila mogućnost pojavljivanja novih pogrešaka te evidentirali slučajevi kada se one pojavljuju.

Oba inkrementa aplikacije nakon faze implementacije nastavljaju se fazom testiranja. U ovim fazama testira se cijeli inkrement kao objedinjeni proizvod spreman za upogonjenje i testiranje kod korisnika.

[18] Vidjeti podrobnije objašnjenje: https://en.wikipedia.org/wiki/Software_bug (Wikipedia, 6. 8. 2017.).

## 7.2. Testiranje dijelova skladišne aplikacije

U nastavku ovaj rad donosi najvažnije zadatke vezane za testiranje gdje će se primijeniti ključne stavke procesa verifikacije, validacije i drugih, u praksi čestih načina testiranja kao i važna stavka uključivanja korisnika u proces razvoja programske podrške skladišne aplikacije.

Prije testiranja sastavljeno je nekoliko tablica prema kojima će se sve provjeravati, a u tablicama će se koristiti kratice *viz* za vizualnu provjeru jesu li korištene ispravne boje, oblici i pozicije kontrola na formama te *fn* za provjeru funkcionalnosti kontrola, odnosno rade li sve kontrole u testu onako kako je određeno specifikacijom.

Svaka tablica prikazuje naziv modula i forme, postoji li uopće element koji se testira te očekivani i dobiveni rezultat funkcionalnosti elementa. Ako dobiveni rezultat nije u skladu s očekivanim, crvenim slovima s uskličnikom upisuje se problem na predloženu poziciju „dobiveni“ u tablici. Na slikama 7.2., 7.3. i 7.4. prikazat će se organizacija testiranja u CRUD fazi prvog inkrementa.

Slika 7.2. Prikaz testa ulaska u aplikaciju i njezine glavne forme

| FORMA | | | | postoji | REZULTAT | |
|---|---|---|---|---|---|---|
| naziv modula | naziv forme | tip testa | naziv elementa | | očekivani | dobiveni |
| Login modul | Login forma | viz | ikona | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | naziv forme | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | labele | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | gumb | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | tekst k. - Korisničko ime | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | tekst k. - Zaporka | DA | ispravne boje, oblici, pozicija | ispravno |
| | | fn | tekst k. - Korisničko ime | | dozvoljen upis znakova | ispravno |
| | | | tekst k. - Zaporka | | upis znakova koji se ne vide | ispravno |
| | | | tekst k. - Korisničko ime | | upis krivog korisničkog imena javlja grešku | DA |
| | | | tekst k. - Zaporka | | upis krive zaporke javlja grešku | DA |

| | | | | | | |
|---|---|---|---|---|---|---|
| Glavni modul | Skladište glavna | viz | ikone | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | naziv forme | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | labele | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | gumbi | DA | ispravne boje, oblici, pozicija | ispravno |
| | | fn | gumbi menija | | otvaraju se ispravno sve forme | DA |
| | | | kartice | DA | prikazuju se svi podaci | na tabu "Primke" se nepotrebno prikazuje "idPrimke"! |
| | | | padajući izbornik | DA | prikazuju se dostupni izvještaji i automatski se otvaraju na odabir | DA |
| | | | statistika | DA | prikazuje se kalkulacija skladišta na 4 decimale i postotak prosječne marže | DA |
| | | | prikaz aktivnog korisnika | DA | nakon imena forme prikazuje se korisničko ime aktivnog korisnika aplikacije | DA |

Izvor: autor

Prikaz na Slici 7.3. odnosi se na rezultate testiranja poslovnih procedura koje se odnose na rad s kupcima, dobavljačima i proizvođačima adresarskog tipa te popis proizvoda koji su momentalno na raspolaganju u skladištu. Prikaz na Slici 7.4. odnosi se na rezultate testiranja rada s primkama i otpremnicama. U fazi CRUD testiraju se samo vizualni i funkcionalni aspekti aplikacije, a u fazi FUNKCIONAL testirat će se i ispravnost kalkulacija.

Slika 7.3. Test poslovnih procesa registracije kupaca, dobavljača, proizvođača i proizvoda

| FORMA | | | | postoji | REZULTAT | |
|---|---|---|---|---|---|---|
| naziv modula | naziv forme | tip testa | naziv elementa | | očekivani | dobiveni |
| Podaci o kontaktima | Kupci, Dobavljači, Proizvođači | viz | ikone | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | naziv forme | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | labele | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | gumbi | DA | ispravne boje, oblici, pozicija | ispravno |
| | | fn | gumbi za upravljanje | | akcije kôda su ispravno povezane s pritiskom | DA |
| | | | padajući izbornici | DA | prikazuju se svi podaci | DA |
| | | | obavezna polja su označena | | polja s obaveznim upisom u označena zvjezdicom | DA |
| | | | upis u obavezna polja | | spremanje upisa bez obaveznih podataka javlja grešku | DA |
| | | | provjera OIB-a | | upis neodgovarajućeg broja znamenki u broju OIB-a javlja pogrešku | DA |
| Podaci o skladištu | Proizvodi | viz | ikone | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | naziv forme | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | labele | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | gumbi | DA | ispravne boje, oblici, pozicija | ispravno |
| | | fn | gumbi za upravljanje | | akcije kôda su ispravno povezane s pritiskom | DA |
| | | | padajući izbornici | DA | prikazuju se svi podaci | DA |
| | | | obavezna polja su označena | | polja s obaveznim upisom su označena zvjezdicom | DA |
| | | | upis u obavezna polja | | spremanje upisa bez obaveznih podataka javlja grešku | DA |
| | | | provjera šifre | | šifra proizvoda je jedinstvena, pri upisu novog proizvoda s istom šifrom javlja se greška | DA |

Izvor: autor

Slika 7.4. Test poslovnih procedura rada s primkama i otpremnicama

| FORMA | | | | postoji | REZULTAT | |
|---|---|---|---|---|---|---|
| naziv modula | naziv forme | tip testa | naziv elementa | | očekivani | dobiveni |
| Primke i Otpremnice | Primke, Otpremnice | viz | ikone | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | naziv forme | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | labele | DA | ispravne boje, oblici, pozicija | ispravno |
| | | | gumbi | DA | ispravne boje, oblici, pozicija | ispravno |
| | | fn | gumbi za upravljanje | | akcije kôda su ispravno povezane s pritiskom | DA |
| | | | padajući izbornik | DA | prikazuju se svi podaci | DA |
| | | | tablice za prikaz podataka na formi | DA | tablice prikazuju sve propisane podatke | DA |
| | | | stavka proizvoda | DA | dvoklik na stavku proizvoda prenosi propisane podatke u stavke primke | DA |
| | | | stavka primke | DA | stavke primke se mogu obrisati tipkom za brisanje na tipkovnici | DA |
| | | | broj primke i otpremnice | DA | pri prijenosu podataka za prvi proizvod, automatski se upisuje broj primke i otpremnice | DA |
| | | | oznaka isporučenosti | DA | kvadratna kontrola za oznaku isporučenosti bilježi promjenu stanja | DA |

Izvor: autor

# 8. ODRŽAVANJE I EVOLUCIJA

Programska podrška skladišne aplikacije u primjeru ovog rada je dovršena, instalirana na klijentsko računalo i nakon svih otklonjenih pogrešaka i provedenog testiranja spremna je za rad. To nikako ne znači da će na ovom koraku biti i kraj angažmana u pogledu predmetne aplikacije. Nakon uspješnog upogonjenja i rada aplikacije, potrebno ju je održavati izmjenama prateći promjene u poslovnim i legalnim procedurama poslovanja, a može se opisati i s najmanje dva od tri Lehmanova zakona (Vargec, Održavanje i evolucija, 2017.), odnosno temeljnih zajedničkih karakteristika evolucije svih praktično primijenjenih programskih rješenja. Prvi Lehmanov zakon ustanovljuje činjenicu da će izmjene u programskoj podršci biti nužne zbog dinamike promjena okoline. Izmjene u aplikaciji utjecat će i na okolinu pa će se proces kružno nastaviti. Drugim zakonom rečeno je da tijek izmjena u programskoj podršci utječe na sve veću složenost strukture što znatno umanjuje njezinu početnu strukturalnu transparentnost i jednostavnost.

Općenito uzevši, možemo reći da se tek u ovoj fazi može govoriti o ***sustavu*** programske podrške za rad u skladištu s obzirom na to da tek u ovom dijelu nastupa intenzivan međuodnos aplikacije, korisnika i računalnog sustava na koji je aplikacija instalirana. Imajući rečeno na umu, jasno je da su u primjeru skladišne aplikacije ovog rada primjenjiva oba Lehmanova zakona i stoga je potrebno unaprijed ugovoriti strategiju održavanja i evolucije programske podrške.

S obzirom na to da je klijent zatražio izradu jednostavne aplikacije koja će imati ograničen doseg, ugovorena je strategija u izmjenama programske podrške koja se naziva *održavanje*, a označava mijenjanje funkcionalnosti pri čemu struktura aplikacije ostaje ista. (Vargec, Održavanje i evolucija, 2017.).

Prema vrsti održavanja ugovoreno je *korekcijsko* i *perfekcijsko* održavanje (Vargec, Održavanje i evolucija, 2017.). Korekcijsko održavanje označava proces u kojem se ispravljaju uočene pogreške, a u perfekcijskom održavanju razvijaju se posve novi dijelovi programske podrške s novim funkcionalnim ili nefunkcionalnim zahtjevima. Ovakvo održavanje je ugovoreno zbog činjenice što je za sada korisnik zadovoljan razvijenom programskom podrškom, ali i svjestan mogućnosti širenja poslovanja skladišta što znači i povećanje broja novih poslovnih procedura koje će trebati opskrbiti programskom podrškom.

# 9. ZAKLJUČAK

Rad je pružio uvid u osnove struke programskog inženjerstva i upravljanja projektima kroz primjer izrade jednostavne aplikacije za skladišno poslovanje. Rad do poglavlja (uključujući i njega) „Procesi programskog inženjerstva, metode i alati“ objašnjava općenite procedure i zakonitosti programskog inženjerstva, povezanu terminologiju, kratak povijesni pregled i porijeklo te mjesto programskog inženjerstva u pogledu suvremenih poslovnih sustava. Rad nastavlja objašnjavati metode, alate i osnovne procese programskog inženjerstva i upravljanja procesima te stavlja važan i izdvojen naglasak na razumijevanje načina kojim se poslovne procedure iz svijeta poslovanja dodiruju s procedurama programskog inženjerstva.

Od poglavlja „Planiranje programske podrške“ nadalje rad nastavlja detaljnije objašnjavati osnovne principe razvoja programske podrške na primjeru skladišne aplikacije. S obzirom na to da je korisnik aplikacije pristao na razvoj programske podrške samo za dio poslovnih procedura, rad donosi model cijelog sustava poslovnih procedura s jasno naznačenim dijelovima koji će se obraditi programskom podrškom.

Za potrebe analize pregledano je postojeće stanje skladišnog poslovanja te je obavljeno inicijalno prikupljanje podataka procesa poslovanja. Nakon analize krenulo se u izradu specifikacije na temelju koje su predstavljeni i zabilježeni korisnički zahtjevi te se uspostavilo temeljno modeliranje programske podrške koja se kani upogoniti.

U poglavlju o oblikovanju i implementaciji, rad predstavlja osnovne pojmove vezane za modeliranje podataka te prevođenje poslovnih procedura i zahtjeva korisnika u model relacijske baze podataka. U ovom poglavlju predstavlja se dijagram toka podataka kao i dijagram relacijske baze podataka. Unutar tog poglavlja ukratko se obrađuje i izrada korisničkog sučelja na primjeru skladišne aplikacije s pregledom najvažnijih uvjeta pri oblikovanju spomenutog sučelja.

Rad završava dvama važnim poglavljima programskog inženjerstva, pogotovo pri razvoju aplikacija suvremenih sustava, a to su verifikacija i validacija te održavanje i evolucija. Spomenuta poglavlja donose ne samo općeniti pregled navedenih procesa, već i detaljan pregled koraka u verifikaciji i validaciji te važnost i motivaciju za održavanje i evoluciju programske podrške.

# LITERATURA (knjige, članci, internetski izvori, ostalo)

## Knjige

Porter, M. E. (1985). *Competitive Advantage: Creating and Sustaining Superior Performance.* New York: The Free Press, A division of Simon & Schuster Inc.

Riordan, M. R. (2005). Database structure. U *Designing effective database systems* (str. 25-45). New York: Addison-Wesley.

Sommerville, I. (2009). *Software engineering* (9. izd.). (M. Hirsch, Ur.) Boston, Massachusetts, USA: Pearson Education, Inc., publishing as Addison-Wesley.

### Članci

Bubaš, G., Hutinski, Ž., i Kermek, D. (2000.). Communication problems in information system and software development. *Journal of Information and Organizational Sciences, Vol.24* (No.1), 43-53.

Leveson, N. G. (2004). The Role of Software in Spacecraft Accidents. *Journal of Spacecraft and Rockets, Vol. 41*, str. 11-25.

Manančikova, N. (1977.). Rana manufaktura i socijalni aspekti povijesti zanatskog stanovništva Dubrovnika u XV. i na početku XVI. stoljeća. (N. Stančić, Ur.) *Radovi Zavoda za hrvatsku povijest Filozofskoga fakulteta Sveučilišta u Zagrebu*, str. 341-356.

Mihaljević, M. (2007). Problemi hrvatskoga računalnoga nazivlja (s jezikoslovnog motrišta). *Studia lexicographica*, str. 68.

### Ostalo

Manger, R. (2005). Softversko inženjerstvo - skripta. *Prirodoslovno Matematički Fakultet, Sveučilište u Zagrebu*. Zagreb.

Vargec, K. (2017). Održavanje i evolucija. *Nastavni materijal - Veleučilište Vern'*. Zagreb.

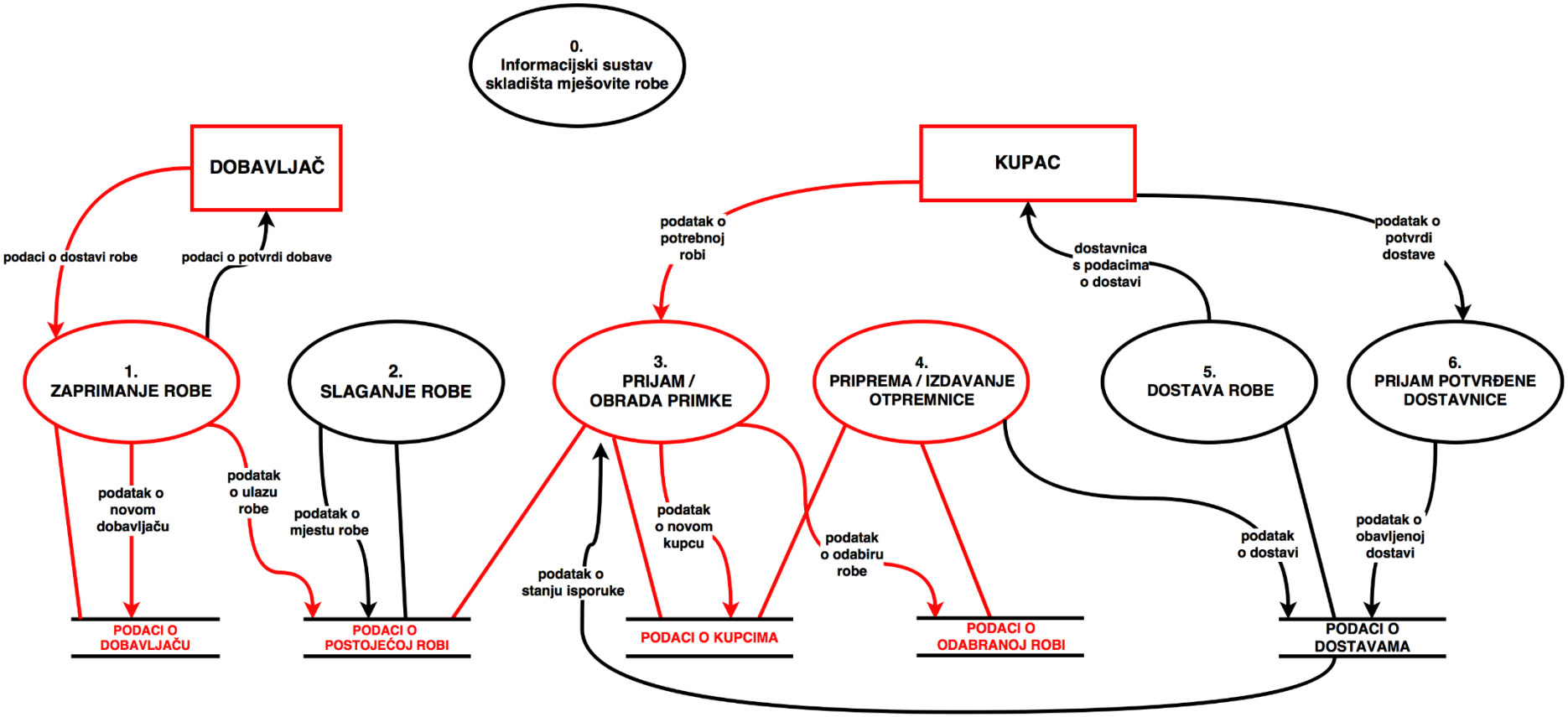
Vargec, K. (2017). Rapid software development. *Nastavni materijal - Veleučilište Vern'*. Zagreb.

Vargec, K. (2017). Softverski modeli. *Nastavni materijal - Veleučilište Vern'*. Zagreb.

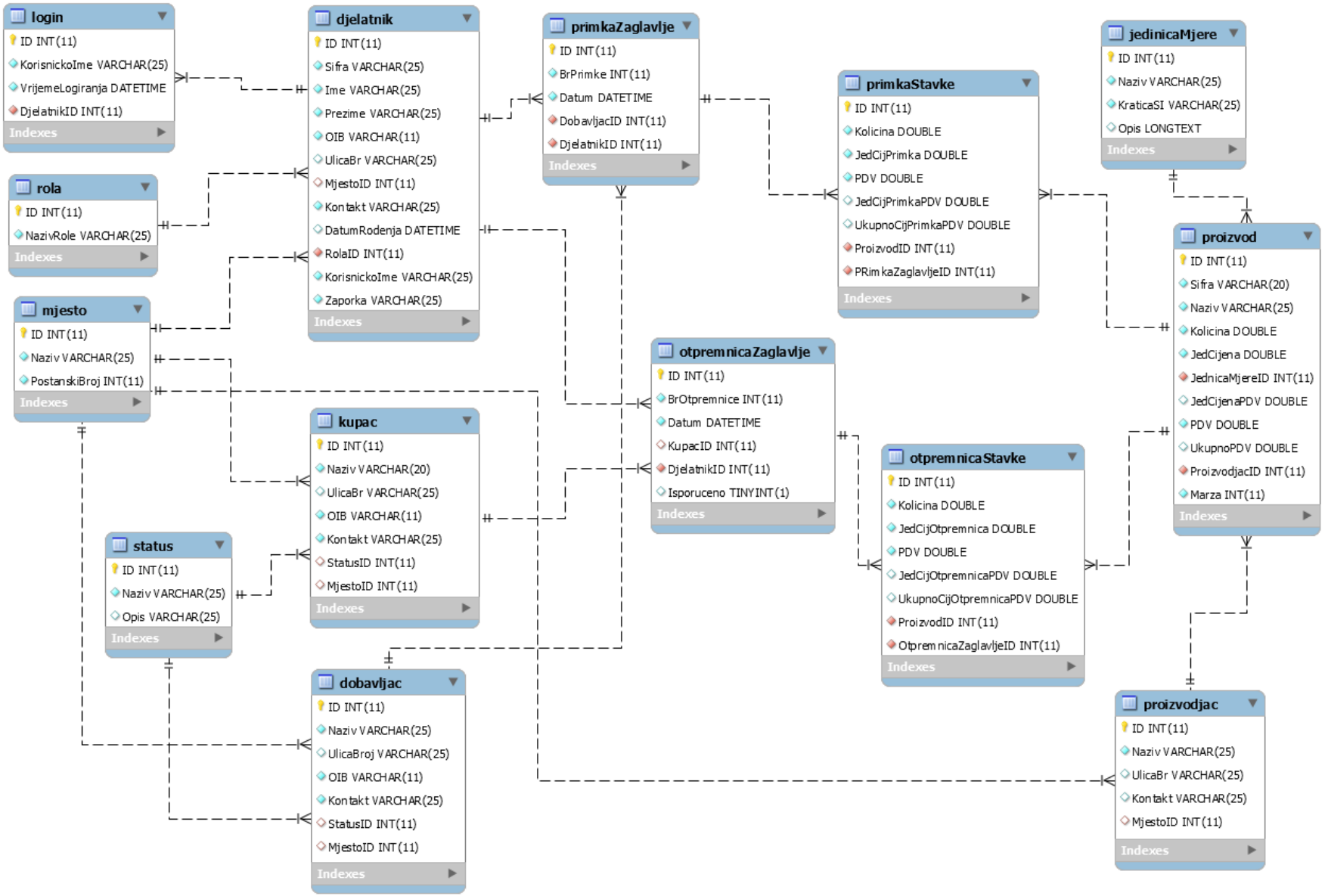
# POPIS SLIKA

# PRILOZI

Prilog 1. Dijagram toka podataka



Prilog 2. Relacijski (ER) dijagram



Prilog 3. Ganttov dijagram