SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

***Tim 21***

Tea Jarčov, 39983/11-R

Luka Klancir, 39998/11-R

Josip Kolarić, 40000/11-R

Ivana Koren, 40004/11-R

Filip Rafajec, 40080/11-R

Appoteka.NET

Projekt iz kolegija programsko inženjerstvo

Varaždin, 2014.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tea Jarčov, 39983/11-R, PS

Luka Klancir, 39998/11-R, IS

Josip Kolarić, 40000/11-R, PS

Ivana Koren, 40004/11-R, IS

Filip Rafajec, 40080/11-R, IS

Redoviti studenti

Preddiplomski studij

Appoteka.NET

PROJEKT IZ KOLEGIJA PROGRAMSKO INŽENJERSTVO

Mentor:

Ivan Švogor, mag. inf.

Varaždin, travanj 2014.

Sadržaj

[1. Uvod 2](#_Toc386388860)

[2. Specifikacija zahtjeva 3](#_Toc386388861)

[2.1 Zahtjevi vanjskog sučelja 3](#_Toc386388862)

[2.1.1 Korisnička sučelja 3](#_Toc386388863)

[2.1.2 Sklopovska sučelja 3](#_Toc386388864)

[2.1.3 Programska sučelja 3](#_Toc386388865)

[2.1.4 Komunikacijska sučelja 4](#_Toc386388866)

[2.2 Sistemska svojstva 4](#_Toc386388867)

[2.2.1 Obavijest o minimalnom stanju zaliha lijeka 4](#_Toc386388868)

[2.2.2 Prikaz stanja i lokacije skladištenja lijeka 5](#_Toc386388869)

[2.3 Izvedbeni zahtjevi 6](#_Toc386388870)

[2.4 Ograničenje dizajna 7](#_Toc386388871)

[2.5 Atributi programskog rješenja 7](#_Toc386388872)

[2.6 Ostali zahtjevi 7](#_Toc386388873)

[3. Projektni plan 8](#_Toc386388874)

[3.1. Projektni tim 9](#_Toc386388875)

[3.2. Terminski plan 9](#_Toc386388876)

[3.3. Proračun i budžet 12](#_Toc386388877)

[3.4. Ponuda naručitelju 13](#_Toc386388878)

[4. Opis dizajna sustava 15](#_Toc386388879)

[4.1 Arhitektura sustava 15](#_Toc386388880)

[4.2. Dijagram slučajeva korištenja (Use Case Diagram) 16](#_Toc386388881)

[4.3. Dijagrami aktivnosti (Activity Diagrams) 17](#_Toc386388882)

[4.3.1. Naručivanje lijekova 17](#_Toc386388883)

[4.3.2. Unos lijekova 18](#_Toc386388884)

[4.3.3. Ažuriranje stanja lijekova 18](#_Toc386388885)

[4.3.4. Prikaz stanja i lokacije skladištenja lijeka 19](#_Toc386388886)

[4.3.5. Provjera klijenta 20](#_Toc386388887)

[4.3.6. Izrada i ispis računa 20](#_Toc386388888)

[4.3.7. Obavijest o minimalnom stanju zaliha lijeka 21](#_Toc386388889)

[5. Dijagram klasa (Class Diagram) 22](#_Toc386388890)

[6. Dijagram podataka 23](#_Toc386388891)

# Uvod

Poslovanje svake ljekarne na prvi pogled je vrlo slično poslovanju u maloprodaji s kakvim se susrećemo svakodnevno, ali nakon detaljnijeg proučavanja možemo zaključiti da u poslovanju ljekarni postoje brojna pravila i ograničenja kojima se onemogućavaju zloupotrebe u sustavu zdravstva kao što su primjerice podizanje lijekova s doktorskim receptom kojemu je istekao rok trajanja ili pak pokušaj kupnje lijekova čija je kupnja nedozvoljena bez doktorskog recepta. Ta pravila i ograničenja koje nalazimo u poslovanju ljekarne su uzrok složenijem poslovanju ljekarni. Prilikom stvaranja ideje našeg projekta imali smo prilike promotriti sustav poslovanja ljekarne te način na koji on funkcionira. Tijekom detaljne poslovne analize smo uvidjeli mnoge propuste i greške u poslovanju za koje smatramo da bi ovim projektom mogli korigirati i optimizirati, te programsko rješenje nastalo kao rezultat našeg projekta ponuditi određenoj ljekarni s ciljem poboljšanja poslovanja.

Naša vizija je da ovim projektom uvelike ubrzamo i poboljšamo cjelokupan sustav naručivanja, prodaje kao i evidencije zaliha unutar poslovanja. Tu viziju realizirati ćemo na način da sve bitne aspekte poslovanja jedne ljekarne objedinimo u jedno jedinstveno programsko rješenje, to jest aplikaciju za upravljanje poslovanjem. Poslovna aplikacija koju bi proizveli po završetku ovoga projekta bila bi pregledna i jednostavna za korištenje, te bi također bila prilagođena svim zahtjevima koje korisnik navede prije početka izrade same aplikacije.

Prvobitni cilj ovog projekta je ostvariti pozitivan rezultat iz oba gledišta, tako da krajnji korisnik aplikacije bude zadovoljan isporučenim proizvodom ali isto tako da se za uloženi trud ostvari određeni profit za naš projektni tim. Zadaci i strategije s kojima želimo ostvariti taj cilj su stvoriti aplikaciju upućenu ljekarnama sa svim zamišljenim funkcionalnostima i opcijama koje bi uvelike pomogle prilikom poslovanja svake ljekarne. Samim time, želimo ukloniti neučinkovito i nekvalitetno praćenje stanja i lokacija skladištenja lijekova uvođenjem tehnoloških trendova.

# Specifikacija zahtjeva

## 2.1 Zahtjevi vanjskog sučelja

### 2.1.1 Korisnička sučelja

Aplikacija je zamišljena kao sustav sa jednim glavnim sučeljem i drugim (izlaznim) sučeljima do kojih se može doći, ovisno o izabranoj radnji, putem glavnog sučelja. Svi korisnici (zaposlenici) pripadati će istoj grupi korisnika, tj. svi su opunomoćeni u radu s aplikacijom te dodatna rješenja u svrhu selekcije neće biti potrebna.

Prilikom implementacije, prvi korisnik koji će biti registriran u aplikaciji biti će ujedno i u mogućnosti dodavati nove korisnike. Pretpostavka je da će prvi korisnik biti voditelj ljekarne te će on kao takav brinuti o drugim korisnicima aplikacije.

Sve mogućnosti i svojstva aplikacije biti će dostupna svim korisnicima iste i će oni moći samostalno upravljati cijelim sustavom (prodaja/narudžba).

Detaljnije o mogućnostima i svojstvima aplikacije nalazi se na točci 4. Opis dizajna sustava.

### 2.1.2 Sklopovska sučelja

Sklopovsko sučelje za ovaj sustav podijeljen je u dvije kategorije: klijentska računala i web servis i server. Klijentska računala nalaze se u poslovnici u kojoj je aplikacija implementirana, tj. na računalima koje zaposlenici koriste.

Web servis i server sa bazom podataka nalazi se ili na kupljenom serveru od providera ili na našem računalu koje smo mi zamislili kao server. Ukoliko je server iznajmljen tada se smanjuje sama sklopovska podrška potrebna za cijeli sustav ali se povećavaju vremenski troškovi za sustav. Ukoliko je server računalo u vlasništvu ljekarne tada se u povećava održavanje cijele infrastrukture.

### 2.1.3 Programska sučelja

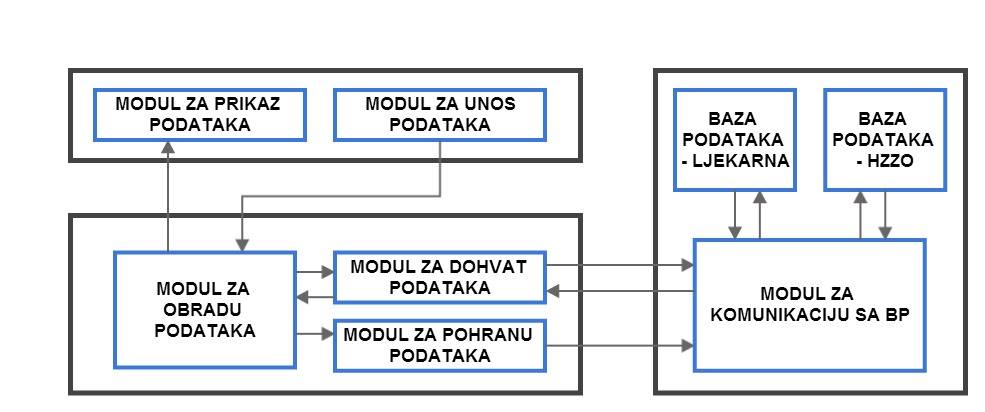
Programska sučelja podijeljena su u dva dijela: programska rješenja na strani servera i programska rješenja na klijentskoj strani.

Sučelja na strani servera zadužena su za obradu zahtjeva klijentskih računala (aplikacija) u koje ulazi ažuriranje baze podataka, komunikacija za HZZO bazom podataka, detaljno vođenje evidencije o svakom lijeku i arhiviranje poslovnih aktivnosti.

Sučelja na strani klijenta zadužena su za podršku zaposlenicima, tj. njihova namjena je komunikacija sa web servisom sustava i front-end prikaz svih podataka koje sustav sadržava.

### 2.1.4 Komunikacijska sučelja

Komunikacijska sučelja implementirana su na serverskoj i klijentskoj strani zajedno sa modulima. Njihov detaljniji prikaz moguće je vidjeti na slici 2.1 Arhitektura sustava.



**Slika 2.1 Arhitektura sustava**

## 2.2 Sistemska svojstva

### 2.2.1 Obavijest o minimalnom stanju zaliha lijeka

Automatizirano svojstvo koje s obzirom na postavljeni okidač u sustavu obavještava zaposlenika (korisnika) o zalihama lijeka i služi kao podsjetnik i sigurnosna granica za naručivanje lijeka.

#### 2.2.1.1 Svrha svojstva

Glavna svrha svojstva je poboljšanje i ubrzanje rada cijelog sustava ljekarne. Svaka ljekarna trguje sa puno lijekova i po svakom lijeku ne drži istu količinu na skladištu. Takav način rada zahtjeva detaljnu evidenciju trenutnog stanja količine lijeka u ljekarni što otežava i oduzima vrijeme zaposlenicima. Svojstvo minimalnog stanja zaliha omogućuje zaposlenicima postavljanje donje granice (količine lijeka), te kada do nje dođe, aplikacija će obavijestiti zaposlenika putem poruke da je lijek došao do donje granice. Daljnja svrha svojstva omogućuje zaposleniku da nakon pročitane obavijesti automatski označi lijek za naručivanje ili donese odluku o promjeni praćenja lijeka.

#### 2.2.1.2 Stimulacijski slijed

Rezultat ovog svojstva trebao bi voditi zaposlenika do jednostavnijeg i bržeg naručivanja samog lijeka kao i pomoć pri odluci kod njegovog praćenja.

Odmah pri unosu lijeka u sustav ljekarne, zaposlenik će biti u mogućnosti postaviti na kojoj količini će željeti ponovo naručiti lijek (kada želi biti obaviješten o zalihi), što će stvoriti zatvoreni krug praćenja svih lijekova za koje će svojstvo biti definirano.

#### 2.2.1.3 Funkcionalni zahtjevi

Svojstvo će biti uključeno i vidljivo pri svakom unosu lijeka te je direktno sa samim lijekom i povezano. Jedini dodatni zahtjev za ovo svojstvo je sama akcija zaposlenika, tj. njegovo definiranje svojstva na svakom lijeku pojedinačno.

##### 2.2.1.3.1 Postojanje lijeka u sustavu

Kako svaki lijek ima mogućnost praćenja zaliha, glavni zahtjev će biti postojanje lijeka na sustavu, tj. postojanje lijeka u određenoj količini na skladištu. Ukoliko lijek ne postoji ili ga nemamo na skladištu, postavljanje evidencije lijeka nema smisla.

##### 2.2.1.3.2 Korisnička inicijativa

Prilikom unošenja lijeka u sustav po prvi puta, nakon dostavljene narudžbe ili kod promjene odluke o vođenju lijeka, zaposlenik će biti u mogućnosti postaviti granicu (broj) na kojeg će aplikacija reagirati. Zaposlenik je taj koji mora donijeti odluku o granici i postaviti ju.

### 2.2.2 Prikaz stanja i lokacije skladištenja lijeka

Svojstvo koje omogućuje zaposleniku precizan uvid gdje se lijek nalazi i u kojoj količini.

#### 2.2.2.1 Svrha svojstva

U sustavu svake ljekarne nalazi se puno lijekova koji se ovisno o svojim količinama i veličini pakiranja u kojem dolaze razlikuju. Iako svaka ljekarna vodi neki svoj sustav evidencije i razvrstavanja, on se uvijek temelji na pamćenju samih zaposlenika. Svojstvo lokacije i stanja lijeka olakšati će zaposlenicima rad i ubrzati izvršavanje zahtjeva kupaca. Zaposlenik će moći u bilo kojem trenutku provjeriti dali lijek postoji i u kojoj količini, te će mu aplikacija također pokazati lokaciju traženog lijeka.

#### 2.2.2.2 Stimulacijski slijed

Svaki lijek kada se unosi u sustav zahtjeva unos količine i lokaciju na koju će se spremiti. Prilikom prodaje lijeka (prije nego se dosegne minimalna granica) zaposlenik ne zna u kojoj se količini lijek nalazi osim fizičkim pretraživanjem i prebrojavanjem lijeka. Nadalje taj problem na sebe nadovezuje još jedan problem, a to je lokacija ili lokacije lijeka. Ovim svojstvom zaposlenik će bez imalo truda saznati potrebne informacije.

#### 2.2.2.3 Funkcionalni zahtjevi

Svojstvo količine i lokacije ažurirati će se prilikom intervencije zaposlenika. Podaci o količini će se ažurirati prilikom bilo kakve promjene, tj. narudžbe ili prodaje dok će se lokacija mijenjati ili proširivati prilikom narudžbe ili odluke o promjeni lokacije. Za realizaciju ove funkcionalnosti potrebno je definirati „spremište“ po već predviđenim sektorima i prilikom ažuriranja definirati tražene podatke.

##### 2.2.2.3.1 Ažuriranje količine

Kako bi se mogla pratiti evidencija o lijeku potrebno je prilikom svake promjene ažurirati podatke o lijeku, tj. njegovu količinu. Ažuriranje, ili uvođenje novog lijeka obavlja se prilikom naručivanja lijeka ili prilikom prodaje istog.

##### 2.2.2.3.2 Klasifikacija „spremišta“

Kako bi smo bili u mogućnosti pratiti lijek, ljekarna zahtjeva jednu razinu organizacijske strukture. U cijelom sustavu ljekarne, lijek je najmanja jedinica. Za ovo svojstvo predviđene su dvije karakteristike koje će jednoznačno definirati svaki lijek i time omogućiti cijelu funkcionalnost. Prva karakteristika je grupa u koju taj lijek pripada, tj. kategorija lijeka. Kategorija će nam omogućiti klasifikaciju lijeka po njegovim svojstvima i djelovanjima i samim time grupiranje lijekova iste kategorije u grupe. Druga karakteristika je „ladica“, tj. fizički prostor na kojem se lijek i/ili grupa lijeka nalaze. Kada je barem jedna karakteristika ispunjena rad sustava postaje znatno precizniji. Kada su obje karakteristike ispunjene zaposlenik zna točno na kojoj se lokaciji lijek nalazi.

## 2.3 Izvedbeni zahtjevi

Zbog fizičke podjele aplikacije na dva dijela: serverski i klijentski dio, same performanse tražene od sustava se povećavaju.

Od klijentskog računala se očekuje podrška Windows OS-a sa barem 2gb radne memorije, i 120gb statične memorije. Kako se sva radnja odvija na web servisu, tj. podaci putuju preko Internet veze, najveći zahtjev ovog sustava je kvalitetna Internet veza. U slučaju proširenja sustava zahtjevi za vezom proporcionalno rastu.

Web servis i server bi trebali imati snagu proporcionalnu sustavu kojeg podržavaju. Ukoliko servis obrađuje podatke samo jedne podružnice na jednom do dva računala performanse neće trebati biti visoke dok se kod većih sustava proporcionalno traži više. Sam server i baza podataka, tj. prostor na kojem se pohranjuju podaci ovisan je direktno o veličini sustava i starosti samog sustava. Ukoliko se podaci o radu arhiviraju trajno tada će s vremenom sustav konstantno rasti. Shodno tome ukoliko se uvede pravilo zastare, sustavu će se optimizirati performanse.

## 2.4 Ograničenje dizajna

Aplikacija ima svega par ograničenja. Prvo i najvažnije je ograničenje s obzirom na Internet vezu i samim time web servis i server. Aplikacija da bi mogla nesmetano obavljati sve funkcije mora biti povezana na Internet kako bi web servis na koji se nadovezuje baza mogao voditi evidenciju o radu sustava. Drugo ograničenje je zakonsko i to se odnosi na fiskalizaciju. Propisanim zakonom sustav mora imati fiskalizaciju i svoje poslovanje voditi shodno tim propisima.

## 2.5 Atributi programskog rješenja

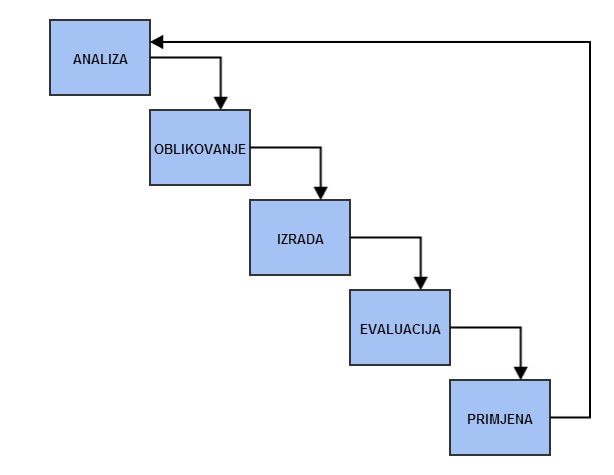
Programsko rješenje ponuditi će svim ljekarskim sustavima brži, precizniji i samim time kvalitetniji rad. Osnovne značajke vezane za sustav kao što su upozorenja kod kritičnih količina zaliha, precizno određivanje količine i lokacije lijeka i kvalitetno uređena baza u svrhu pretraživanja daju aplikaciji na težini i modernosti. Aplikaciju je moguće implementirati u bilo koji sustav što joj daje veliku adaptabilnost i zbog zakonske obveze za fiskalizacijom i Internet vezom, servis koji će obrađivati podatke biti će lako proširiv na više podružnica ukoliko sustav ima potrebu za tim.

## 2.6 Ostali zahtjevi

Aplikacija kao takva može raditi bez ijednog dodatnog svojstva, tj. bez njihovog specificiranja. Međutim bit ove aplikacije jest unaprjeđenje sustava što sa sobom nosi neke posljedice. Iz tih razloga glavni zahtjev ove aplikacije je uključenost zaposlenika sustava u aplikaciju i njeno održavanje.

# Projektni plan

Za izradu projektnog plana izabrali smo vodopadni model, čija struktura je prikazana na sljedećoj slici.



**Slika 3.1. Vodopadni model**

Kod vodopadnog modela, faze razvoja se realiziraju strogo sekvencijalno, istovremeno za cijeli programski proizvod. Prednost ovog modela je dobra dokumentacija te istovremeni završetak svih podsustava. Dobra dokumentacija je temelj za jednostavno održavanje aplikacija. Isto tako, ovaj pristup daje garanciju da će u konačnom vremenu, doći do zadovoljavajućeg rješenja programskog proizvoda, a samim time i smanjenja rizika od neuspjeha.

Ovaj model smo izabrali iz razloga jer najviše opisuje naš rad na ovom projektu. Kako je i navedeno, prije početka rada bilo je potrebno napraviti kvalitetnu analizu. Analiza je obuhvaćala područje rada, određenu zahtjevnost projekta, specifikacije te prednosti i nedostatke pojedinog područja. Analizom smo stekli uvid u stvarno stanje određenih ljekarni te na temelju toga definirali naše ciljeve, odnosno zadatke koje planiramo provesti kako bi poboljšali njihov sustav poslovanja. Nakon analize slijedi oblikovanje, koje obuhvaća pripremu određenih dijelova projekta, izradu dijagrama i stvaranje osnovnog koncepta za rad. Sljedeća faza je sama izrada gdje se primjenjuje sve iz prethodne dvije faze. Kod faze evaluacije, potrebno je testirati određene dijelove projekta prije same primjene. U našem slučaju, aplikaciju je potrebno ispitati i testirati kako bi se utvrdile eventualne pogreške u radu. Nakon ispravne evaluacije, proizvod, odnosno aplikacija je spremna za primjenu.

Drugim riječima, aplikacija je spremna za korištenje od strane ljekarne. Ukoliko korisnici aplikacije nisu zadovoljni ili imaju dodatnih zahtjeva za promjenama, proces se ponavlja ispočetka počevši ponovno s prvom fazom – analizom.

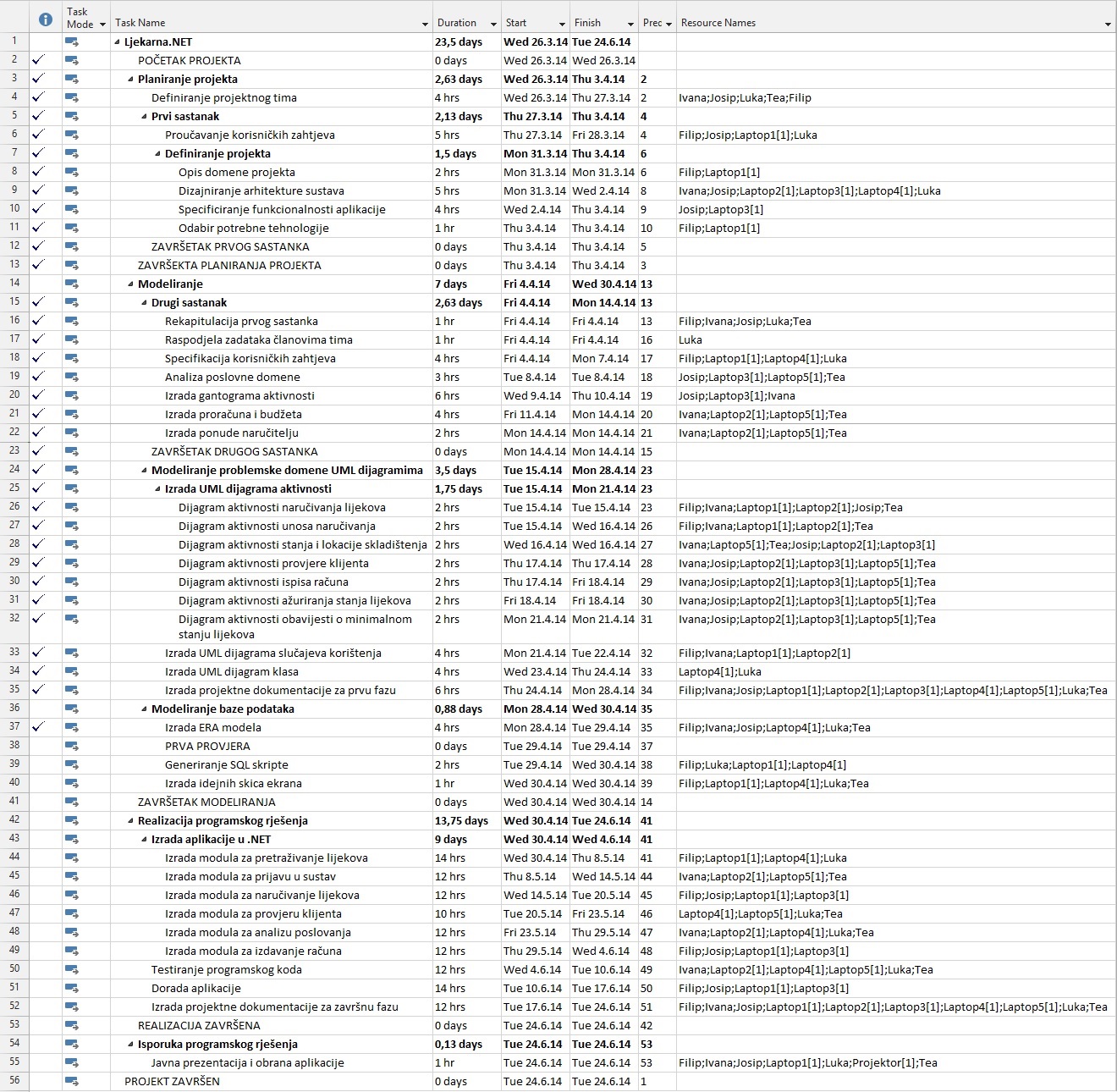
## 3.1. Projektni tim

Članovi tima navedeni su na početku dokumentacije, podijeljeni u nekoliko uloga:

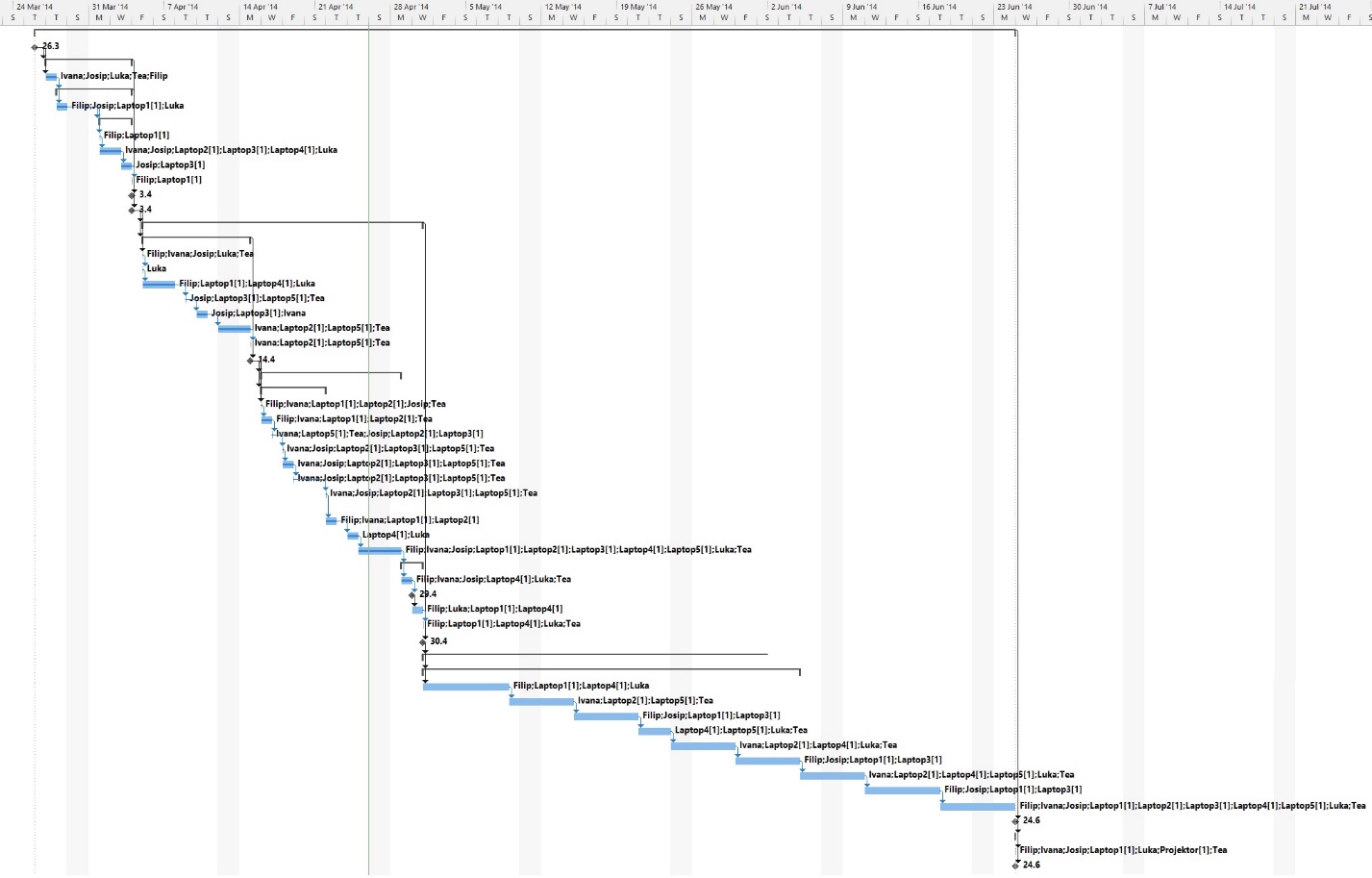
* Arhitekt – izrađuje SQL skriptu i UML dijagrame na temelju odluka analitičara
* Analitičar – donosi odluke o tome što će se modelirati, programirati i testirati te proučava korisničke zahtjeve i prilagođava ih realnoj implementciji
* Dizajner – bavi se „vizualnom komunikacijom“; definira izgled i uređuje dokumentaciju, osmišlja i izrađuje dizajn aplikacije, vodi računa o tome da aplikacija bude pregledna, jednostavna i razumljiva korisnicima
* Programer – kreira pseudokod i piše programski kod, uređuje dio dokumentacije koji se odnosi na rješenje programa i testira aplikaciju, održava je i otklanja pogreške
* Voditelj projekta – izrađuje projektni plan te ga ažurira u skladu sa svim promjenama, definira potrebne resurse i raspoređuje ih po dostupnosti, dodjeljuje zadatke članovima tima i prati izvedbu projekta

Navedene uloge nisu specifično određene, već svi članovi tima sudjeluju u svim dijelovima projekta podjednako. Izrada projekta temelji se na dogovoru svih članova i pokušaja da svi utroše podjednako vremena na izradu.

## 3.2. Terminski plan

Terminski plan našeg projekta izrađen je u Microsoftovom alatu MS Project koji nam je omogućio da bilježimo svaku aktivnost koju smo izvršili ili planiramo izvršiti u budućnosti te je na taj način uvelike olakšao planiranje vremena i raspodjelu zadataka u timu. Aktivnosti u terminskom planu su zapisane slijedno kako smo ih izvršavali i u stvarnosti, dok su one aktivnosti koje još uvijek nisu završene zapisane na način da prikazuju plan izvršavanja aktivnosti u budućnosti. Između pojedinih aktivnosti prikazana je i njihova međuovisnost, što znači da određena aktivnost ne može započeti sa izvršavanjem ako je uvjet njenog početka da prethodna aktivnost bude u potpunosti završena. Na svakoj aktivnosti određeno je njeno vrijeme trajanja, dok se u samom terminskom planu nalaze i brojne aktivnosti čije vrijeme trajanja nije određeno nego proračunato kao zbroj trajanja svih aktivnosti koje se izvršavaju kao pod aktivnosti unutar te aktivnosti. Datum početka prve aktivnosti zadan je kao datum kada je projekt odobren. Datum završetka projekta nije zadan nego izveden, a dobiven je zbrojem trajanja svih aktivnosti u projektu. Aktivnostima su također dodani i resursi koji se koriste ili rade na tim aktivnostima, resurse smo podijelili na rad i materijal. Na slici 3.2. moguće je vidjeti sve aktivnosti koje se nalaze u projektu te podatke koji su vezani za te aktivnosti, dok se na slici 3.3. može vidjeti Ganttov dijagram projekta.

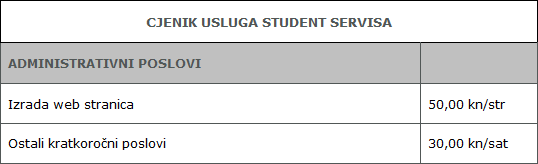
**Slika 3.2. Terminski plan aktivnosti**



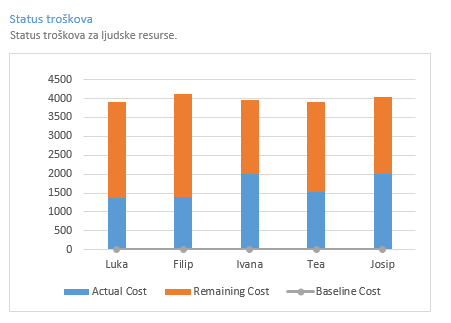
**Slika 3.3. Ganttov dijagram**

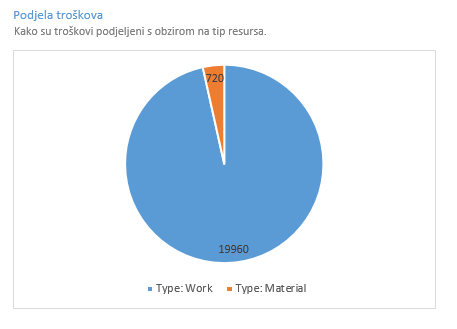
## 3.3. Proračun i budžet

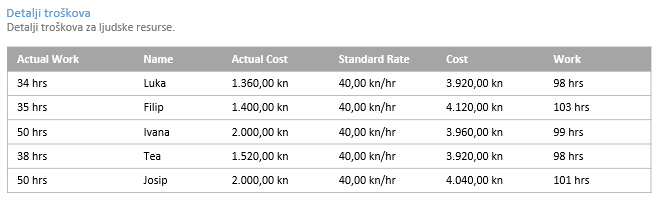
Proračun i budžet projekta dobiveni su također uz pomoć Microsoftovog alata MS Project koji pruža mogućnost kreiranja izvještaja koristeći podatke iz unesenih aktivnosti. Proračun i budžet našeg projekta dobiven je na način da smo odredili koliki su troškovi pojedinih resursa s obzirom na vrijeme njihova korištenja. Tako smo primjerice za plaću ljudskih resursa odredili iznos od 40 kuna po satu, dok je trošak korištenja materijalnih resursa 10 kuna po satu za svaki od laptopa koji se koriste u projektu te za projektor koji je potreban za prezentaciju projekta. Cijena troška ljudskih resursa po satu je određena sukladno sa cjenikom koji je postavljen na web stranicama studentskog servisa Varaždin, iz razloga što su kategorije navedene u cjeniku bile preopširne, našu cijenu odredili smo kao aritmetičku sredinu kategorija koje najviše odgovaraju opisu našeg projekta. Te kategorije su „Izrada Web stranica“ gdje je cijena troška ljudskih resursa po satu 50 kuna, i „Ostali kratkoročni poslovi“ gdje je cijena troška ljudskih resursa po satu 30 kuna. Na slici 3.4. prikazane su kategorije navedene u cjeniku student servisa Varaždin, dok su na slici 3.5., 3.6. i 3.7. prikazani troškovi resursa u projektu.



**Slika 3.4. Cjenik usluga student servisa**



**Slika 3.5. Status troškova za ljudske resurse**

**Slika 3.6. Podjela troškova s obzirom na tip resursa**

**Slika 3.6. Detalji troškova za ljudske resurse**

## 3.4. Ponuda naručitelju

Nakon što smo definirali terminski plan i proračun projekta, izradili smo ponudu za naručitelja, koji je u našem slučaju ljekarna. U ponudi je navedena okvirna cijena aplikacije, module i usluge koje obuhvaća aplikacija.

*Tim 21*

*Ludbreška 3*

*42 000 Varaždin*

*Sveučilište u Zagrebu*

*Fakultet organizacije i informatike*

*Pavlinska 2*

*42 000 Varaždin*

U Varaždinu, 26. ožujka 2014.

**PONUDA**

Ponudili bismo Vam aplikaciju koja je trenutno u izradi i za koju smatramo da bi Vam bila veoma korisna te omogućila brže i kvalitetnije poslovanje. Neke od glavnih specifikacija koje bi aplikacija omogućavala:

* Mogućnost narudžbe određenih proizvoda
* Provjeru stanja i lokacije skladištenja pojedinog proizvoda
* Obavijest o minimalnom stanju zaliha
* Prilagodbu cijene klijentima s dopunskim osiguranjem

Nakon izvedbe plana projekta, predviđena cijena aplikacije je 21.000 kn, a u cijenu su uključeni troškovi analize poslovnog slučaja, izrada svih potrebnih dijagrama i baze podataka te sama realizacija i izrada aplikacije. Predviđeno trajanje projekta je tri mjeseca.

Za sva pitanja možete nam se obratiti putem e-maila ili možemo ugovoriti sastanak na kojem bi se rješavale nedoumice vezane uz projekt.

Spremni smo na dogovore i radujemo se budućoj suradnji.

S poštovanjem,

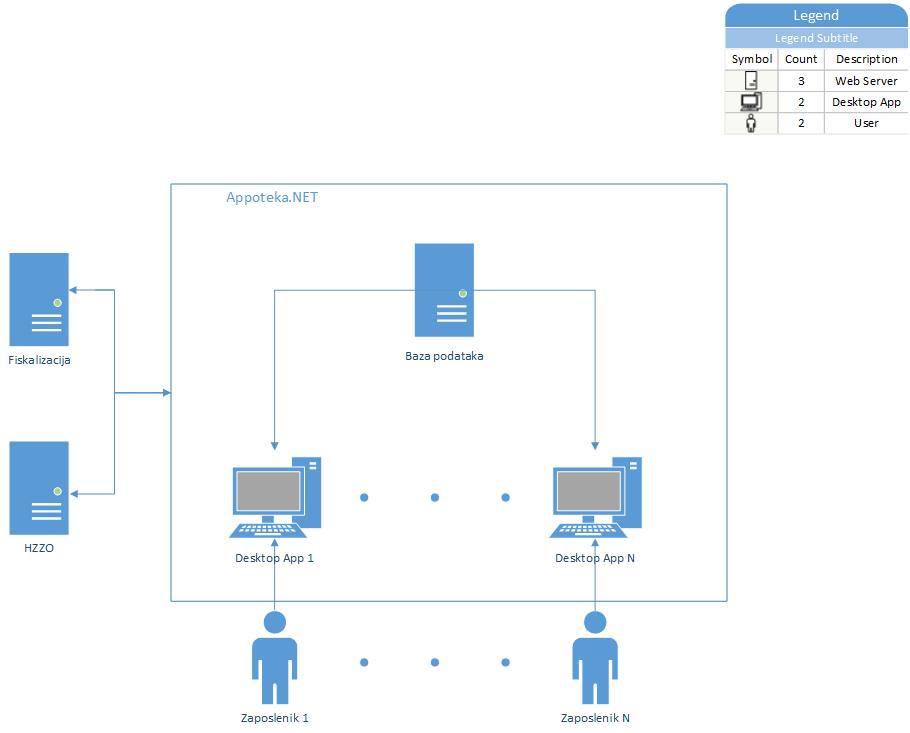
*Tea Jarčov (tjarcov@foi.hr)*

*Luka Klancir (lklancir@foi.hr)*

*Josip Kolarić (jkolaric@foi.hr)  
Ivana Koren (ikoren@foi.hr)  
Filip Rafajec (frafajec@foi.hr)*

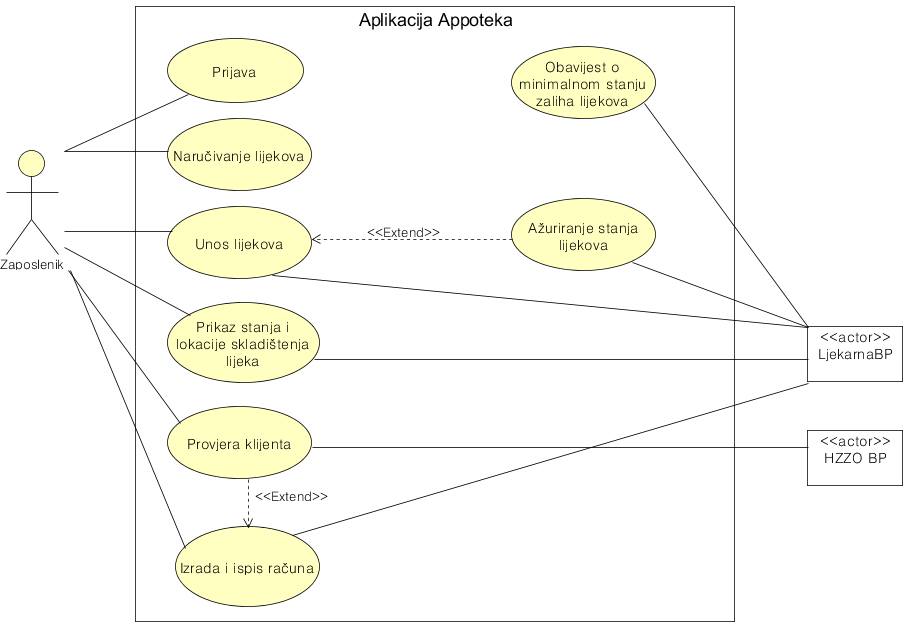
# Opis dizajna sustava

## 4.1 Arhitektura sustava



Sljedeća arhitektura sustava prikazuje našu aplikaciju Appoteka.NET koja je instalirana na desktop računalu te povezana sa web bazom podataka. Odlučili smo se za bazu podataka na web serveru zbog veće mogućnosti skalabilnosti i integracije samog sustava unutar grupacija određenih ljekarni. Također sama aplikacija komunicira sa bazom podataka HZZO zbog preuzimanja prepisanih recepata naših klijenata. Zbog potrebe fiskalnog računa aplikacija po potrebi komunicira sa poreznom upravom.

## 4.2. Dijagram slučajeva korištenja (Use Case Diagram)



**Slika 4.2. Dijagram slučajeva korištenja**

Primarni i jedini učesnik u našem sustavu je zaposlenik, jer samo on koristi i ima pristup aplikaciji. Slučajevi korištenja unutar sustava su *Prijava*, *Naručivanje lijekova*, *Unos lijekova*, *Ažuriranje stanja lijekova*, *Prikaz stanja i lokacije skladištenja lijeka*, *Provjera klijenta*, *Izrada i ispis računa* te *Obavijest o minimalnom stanju zaliha lijekova*.

Slučaj korištenja *Ažuriranje stanja lijekova* je mogućnost proširenja slučaja *Unos lijekova*, a slučaj korištenja *Provjera klijenta* je mogućnost proširenja slučaja *Izrada i ispis računa.*

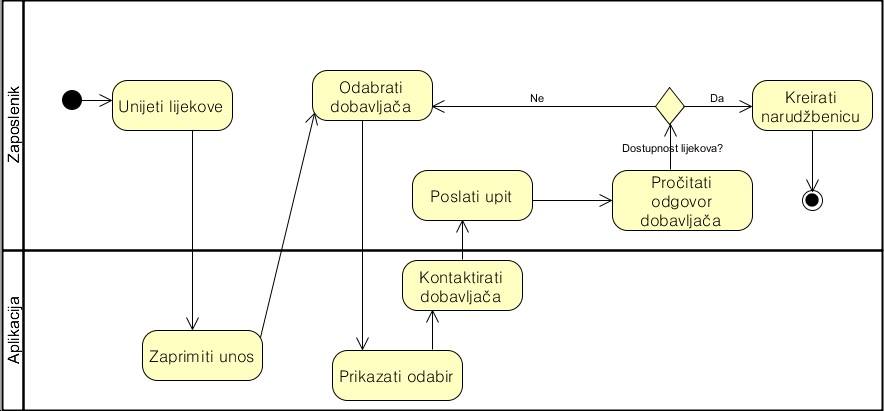
U međudjelovanju s aplikacijom su dvije baze, koje su međusobno simulirane. To su baza ljekarne u kojoj se nalaze podaci o lijekovima, zaposleniku, dobavljačima i slično te se unutar te baze može izvršavati dohvaćanje, ažuriranje i brisanje određenih podataka. Druga baza je HZZO baza podataka iz koje ljekarna može samo dohvaćati određene podatke kao što su podaci o klijentu, njegovim receptima te propisanim lijekovima.

Kod pristupanja aplikaciji *Appoteka*, zaposlenik mora izvršiti *Prijavu* u sustav. Kod prijave on unosi svoje korisničko ime i lozinku kako bi mogao pristupiti ostalim slučajevima aplikacije, koji će biti opisani u nastavku.

## 4.3. Dijagrami aktivnosti (Activity Diagrams)

### 4.3.1. Naručivanje lijekova

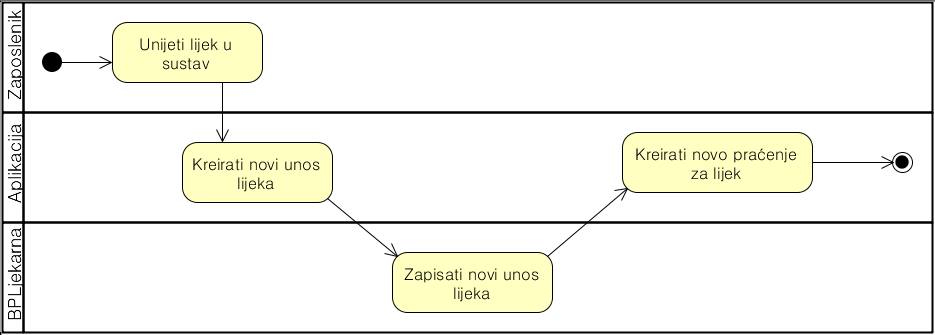
U aktivnosti *Naručivanje lijekova* sudjeluju zaposlenik i aplikacija. Sve kreće od zaposlenika koji kreće s unosom potrebnih lijekova za narudžbu u aplikaciju, nakon čega aplikacija zaprima taj unos i sprema ga. Nakon unosa lijekova, zaposlenik unosi, odnosno odabire dobavljača te aplikacija prikazuje odabir. Nakon odabira aplikacija nudi mogućnost kontakta s dobavljačem putem e-maila. Korisnik odabirom kontaktiranja šalje upit dobavljaču. Nakon poslanog upita čeka odgovor te ga čita. U odgovoru zaposlenik može uočiti jesu li naručeni lijekovi dostupni te ukoliko jesu može kreirati narudžbenicu, odnosno završiti sa ispunjavanjem svih podataka za narudžbu. U slučaju da dobavljač nema tražene lijekove, postupak se ponavlja od mjesta odabira dobavljača.



**Slika 4.3.1 Dijagram aktivnosti *Naručivanje lijekova***

### 4.3.2. Unos lijekova

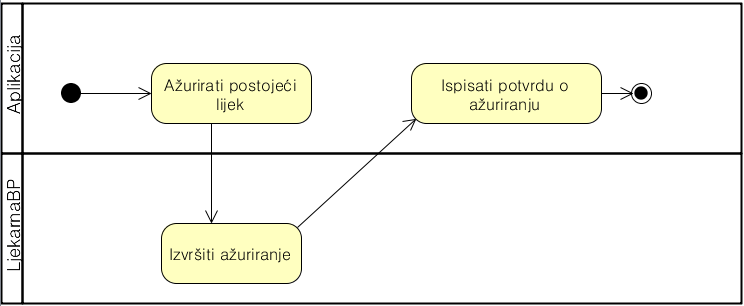
Za aktivnost *Unos lijekova* učesnici su zaposlenik, aplikacija i baza podataka ljekarne. Zaposlenik inicira aktivnost s unosom lijeka u sustav, nakon čega aplikacija bilježi unos i automatski ga kreira. Nakon toga, unešeni podaci šalju se u bazu podataka gdje se zapisuje novi lijek i njegovi podaci. Nakon toga, aplikacija kreira novo praćenje za lijek, odnosno postavlja određeni okidač na količinu lijeka kako bi mogao javiti minimalno stanje.



**Slika 4.3.2 Dijagram aktivnosti *Unos lijekova***

### 4.3.3. Ažuriranje stanja lijekova

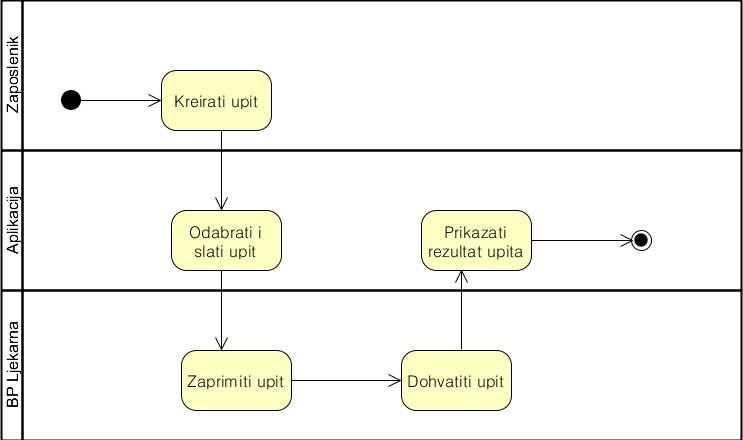
Učesnici u aktivnosti su aplikacija i baza podataka ljekarne. S obzirom na to da je navedena aktivnost proširena mogućnost aktivnosti *Unos lijekova*, korisnik je već tamo izvršio potreban unos lijekova tako da ovdje aplikacija sama izvršava ažuriranje već postojećeg lijeka. Osim aplikacije, ažuriranje je potrebno i izvršiti unutar baze podataka, nakon čega aplikacije ispisuje potvrdu o uspješnom ažuriranju.



**Slika 4.3.3 Dijagram aktivnosti *Ažuriranje stanja lijekova***

### 4.3.4. Prikaz stanja i lokacije skladištenja lijeka

Aktivnost počinje kreiranjem upita od strane zaposlenika. On unosi na primjer naziv lijeka za kojeg želi provjeriti stanje ili lokaciju skladištenja. Nakon toga aplikacija odabire kreirani upit i šalje ga prema bazi podataka u kojoj se nalaze traženi podaci. U bazi se zaprimaju i dohvaćaju podaci vezani za upit te se šalju do aplikacije koja prikazuje sam rezultat upita zaposleniku.

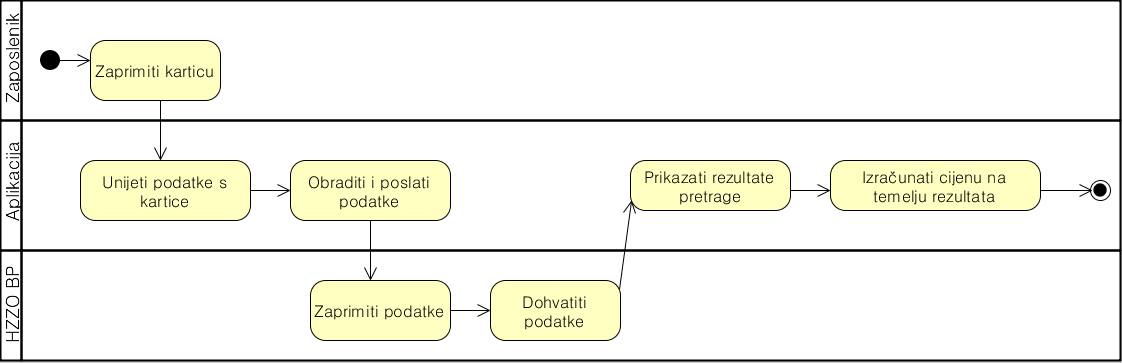


**Slika 4.3.4 Dijagram aktivnosti *Prikaz stanja i lokacije skladištenja lijeka***

### 4.3.5. Provjera klijenta

Klijent koji dođe u ljekarnu, može ali i ne mora biti unutar sustava HZZO. Ukoliko je klijent unutar sustava HZZO, odnosno ukoliko ima dopunsko osiguranje, ima i određeni popust na određene lijekove koji isto tako moraju biti unutar sustava. Prilikom dolaska klijenta u ljekarnu, zaposlenik zaprima njegovu karticu, odnosno broj kartice koji unosi u aplikaciju. Aplikacija bilježi zaprimljene podatke te ih šalje na obradu unutar baze podataka HZZO. Iz te baze može dohvatiti podatke o osiguranju klijenta te njegovim receptima, nakon čega aplikacija prikazuje rezultate pretrage. Ukoliko klijent posjeduje dopunsko osiguranje, potrebno je izračunati cijenu lijeka za dopunsko osiguranje.

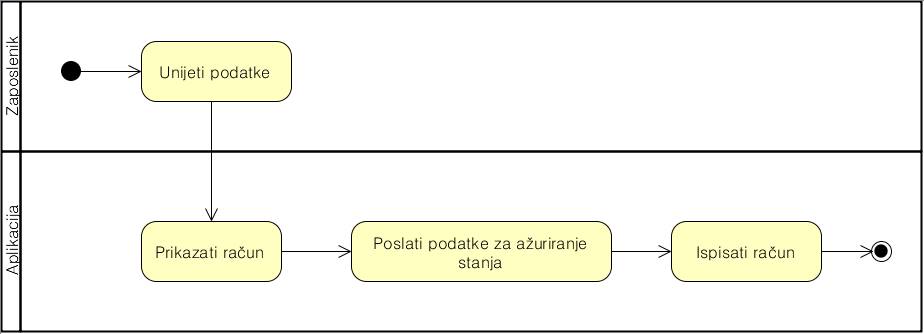
Aktivnost može biti izvedena samostalno, ukoliko klijent dođe s ciljem da samo provjeri stanje dopunske kartice ili recepta. S druge strane, aktivnost se može izvršiti kao nadopuna aktivnosti *Izrada i ispis računa*.



**Slika 4.3.4 Dijagram aktivnosti *Provjera klijenta***

### 4.3.6. Izrada i ispis računa

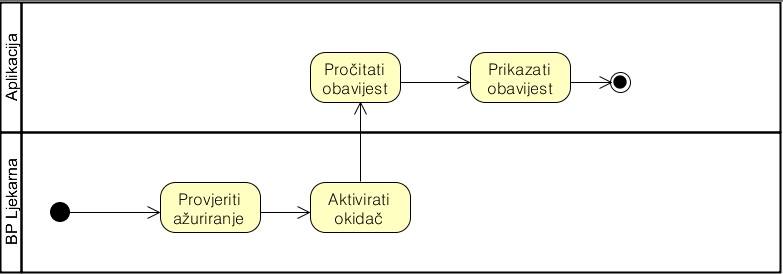
Zaposlenik započinje s unosom podataka potrebnih za izradu računa, nakon čega aplikacija prikazuje izgled računa te šalje podatke za ažuriranje stanja. Nakon toga se račun ispisuje. Ova aktivnost se dodatno može proširiti aktivnošću *Provjera klijenta* koja je prethodno opisana. Račun se može izdati i klijentu koji nije u sustavu HZZO te stoga nije potrebno provjeravati njegove podatke, već samo izdati račun za kupljenu robu.



**Slika 4.3.6 Dijagram aktivnosti *Izrada i ispis računa***

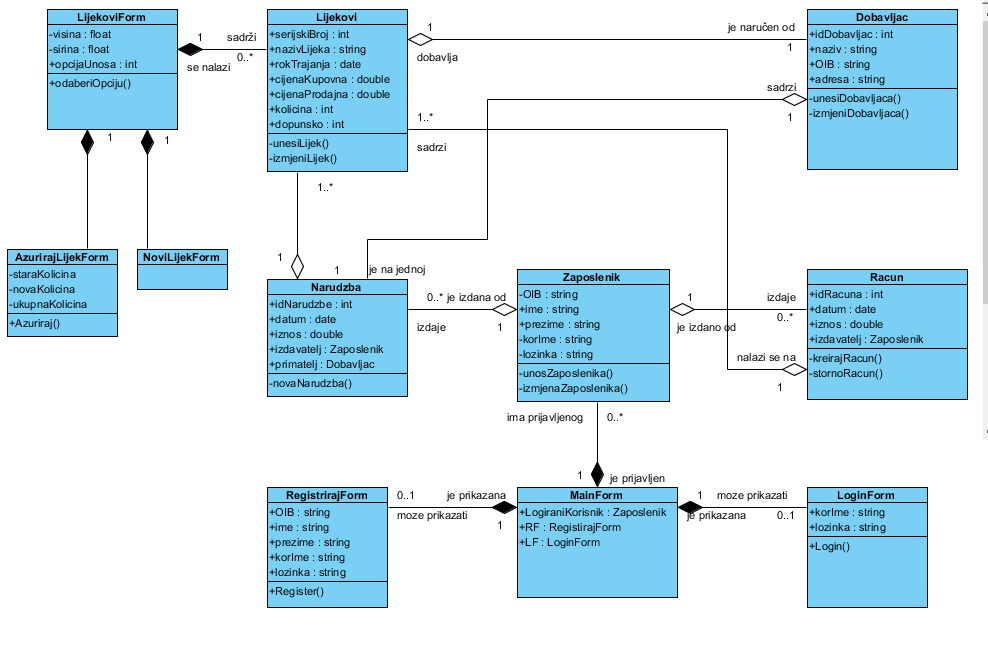
### 4.3.7. Obavijest o minimalnom stanju zaliha lijeka

Ova aktivnost vezana je samo uz aplikaciju i bazu podataka, ne inicira je zaposlenik. Unutar baze podataka ljekarne vrši se provjera svakog ažuriranja stanja zaliha pojedinog lijeka, nakon čega se aktivira okidač na minimalno stanje zaliha. Ukoliko se dogodi minimalno stanje zaliha, aplikacija čita obavijest iz baze podataka te prikazuje tu obavijest zaposleniku. Obavijest daje do znanja zaposleniku da je potrebno naručiti nove zalihe lijeka.



**Slika 4.3.7 Dijagram aktivnosti *Obavijest o minimalnom stanju zaliha lijeka***

# Dijagram klasa (Class Diagram)



**Slika 5.1. Dijagram klasa aplikacije Appoteka.NET**

Dijagram klasa prikazuje nam strukturu aplikacije te klase koje ćemo koristit kod izrade iste.

Klasa **Zaposlenik** sadrži informacije o zaposlenicima koji su korisnici aplikacije. Zaposlenik mora prvotno biti registriran da bi se mogao ulogirati u sustav i koristiti funkcionalnosti. To radi preko forme **RegistrirajForm**, a nakon registracije se svaki puta potrebno prijaviti putem **LoginForm** u sustav. Nadalje zaposlenik se koristi u klasama **Narudzba** i **Racun** gdje se kod izdavanja bilo kojeg od tih dvoje poslovnih dokumenata mora naznačiti koja osoba odnosno zaposlenik ju je izdao. Klasa **Lijekovi** sadrži sve informacije o lijekovima, a povezana je sa klasom **Dobavljac** koja klasificira određene dobavljače za određene lijekove.

**LijekoviForm** je forma za unos/provjeru/ažuriranje lijekova.

*\*Dijagram klasa nije finalni te je podložan ažuriranju tokom razvoja aplikacije*

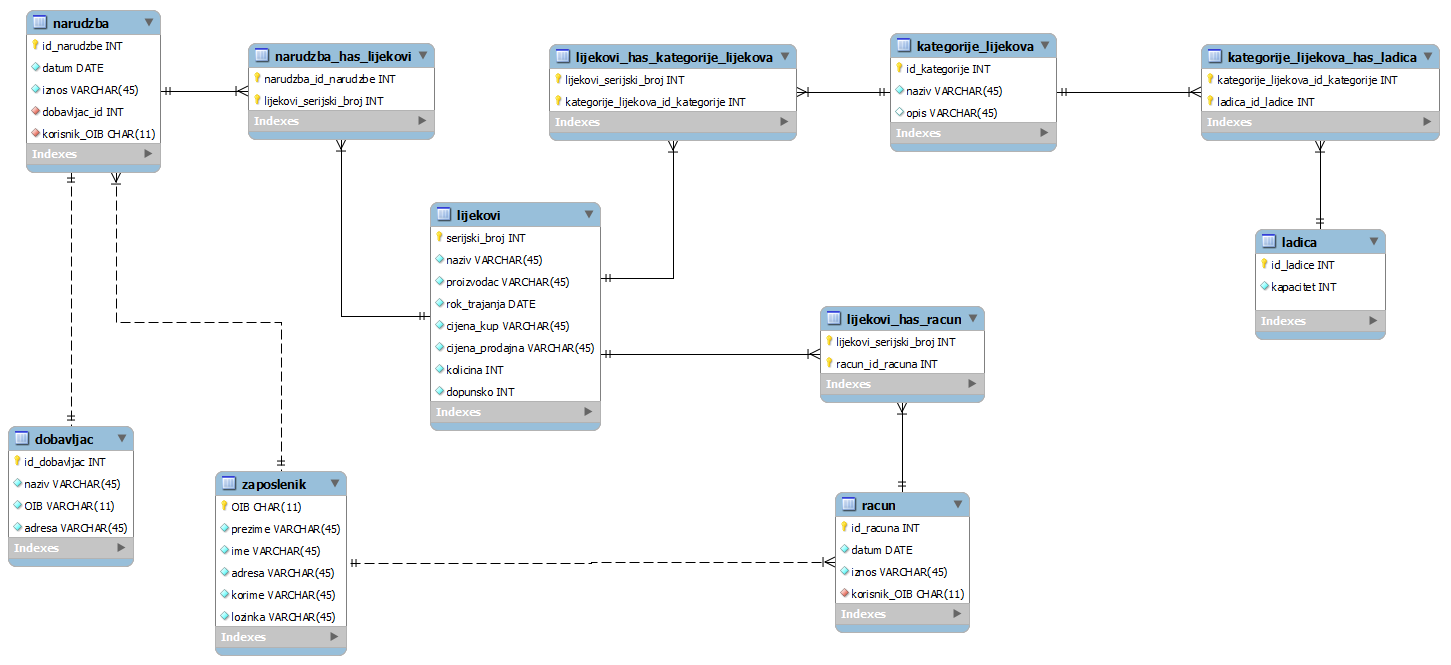
# Dijagram podataka

Sljedeći dijagram podataka je ERA model podataka izrađen u alatu MySQL Workbench.

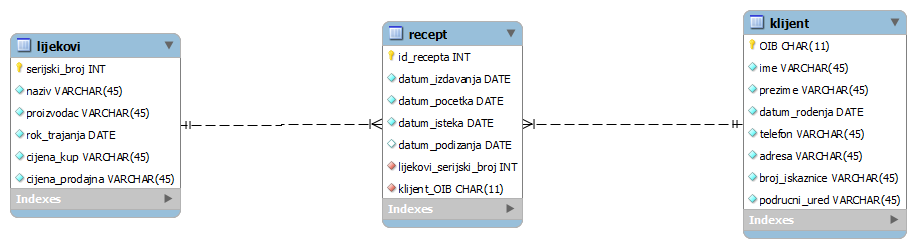
ERA model podataka sastoji se od sljedećih entiteta:

* Lijekovi – svaki lijek spada pod određenu kategoriju koja spada pod određenu kategoriju, a ista kategorija može biti raspoređena kroz više spremišta (ladica) ovisno o količinama koje se naručuju
* Zaposlenici – korisnici aplikacije
* Narudzbe – narudžbe lijekova
* Racun – izdavanje računa
* Dobavljac – dobavljači u našem sustavu nabave

Također smo zbog potrebe funkcionalnosti simulirali smo bazu HZZO koja nam je potrebna za povlačenje klijenata i njihovih prepisanih recepata za lijekove.



**Slika 6.1 ERA model podataka za bazu Appoteka.NET**



**Slika 6.2 ERA model podataka sa simulaciju baze HZZO**