Univerzitet u Kragujevcu

Fakultet inženjerskih nauka



**Projektovanje informacionih sistema i baza podataka**

Projektni zadatak:

Apoteka

Student: Predmetni nastavnik:

Tijana Tanasković 656/2017 Milan Erić

**Sadržaj:**

**1 Uvod . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .** 3

* 1. Relevantni dokumenti i potrebe realnog sistema . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

**2 Dekompozicioni dijagram sistema. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4**

2.1 Dijagram konteksta. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

2.2 Dekompozicioni dijagrami. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.3 Stablo aktivnosti. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

**3 ER dijagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8**

3.1 Entiteti . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .8

3.2 Veze . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

3.3 Kompletan Er dijagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

**4 Logicka šema relacione baze podataka . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18**

4.1 Entiteti . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .18

4.2 Veze . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .19

4.3 Međurelaciona ograničenja . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

4.4 Kompletna logička šema . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

**5 Fizička šema . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .21**

**6 Implementacija projektovane baze sa testnim podacima . . . . . . . . . . . 21**

**7 Aplikacija. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33**

**8 Literatura . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .39**

**1. Uvod**

Apoteka je zdravstvena ustanova, koja nabavlja lekove i sanitetski materijal, ispituje ih i njima snabdeva stanovništvo. U nekim apoteka se takođe i izrađuju i testiraju lekovi.

Apoteka je radno mesto većine farmaceuta, koji u ovom slučaju, pored uloge farmaceuta, imaju i ulogu prodavca. U apoteci tokom čitavog radnog vremena, mora biti prisutan kvalifikovan farmaceut.

Većina apoteka je danas svoj asortiman proširila sa lekova na kozmetiku, zdravu hranu (hranu za ljude sa intolerancijom, posebna brašna, formule za bebe i sl.), naočare za vid i slične proizvode.

U ovom projektnom zadatku biće objašnjen način modelovanja baze podataka informacionog sistema koji je pogodan za skladištenje informacija u apotekama.

* 1. **Relevantni dokumenti i potrebe realnog sistema**

U našoj bazi podataka odlučili smo da čuvamo informacije bitne za svaku apoteku, informacije o zaposlenima, proizvodima koje nabavljaju i prodaju, dobavljače koji sve te proizvode obezbeđuju i kupce (pacijente) koji uzimaju lekove na recept.

Bitne tačke ovog sistema su:

-apoteka

-zaposleni u toj apoteci

-dobavljač

-proizvodi

-pacijenti

Veze između tačaka:

-trenutni dobavljači obezbeđuju proizvode

-proizvodi su oni koji su trenutno dostupni u apoteci

-zaposleni prodaje proizvode kupcima

-kupci mogu kupiti raznovrsne proizvode

**2. Dekompozicioni dijagram sistema**

Dekompozicioni dijagram prikazuje najbitnije procese koji se odvijaju u sistemima takođe i tokove podataka kojima su ti procesi povezani.

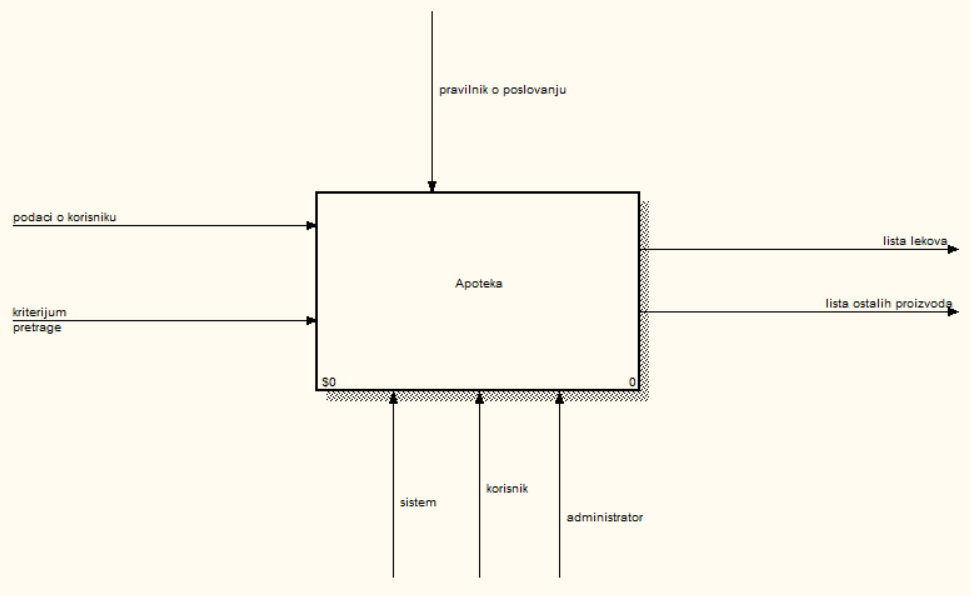
**2.1 Dijagram konteksta**

Dijagram konteksta je najviši nivo apstrakcije koji se dekompozicionim dijagramima prevodi u niže nivoe apstrakcije. Strelice sa leve strane pravougaonika predstavljaju ulaze koji se koriste u sistemu. U ovom slučaju to su podaci o korisniku i kriterijum pretrage. Podaci o korisniku su bitni prilikom zavođenja radnika, a na osnovu kriterijuma pretražujemo bazu podataka.

Strelice na gornjoj strani pravougaonika su kontrole ili uslovi izvođenja aktivnosti. U našem slučaju je to pravilnik o poslovanju.

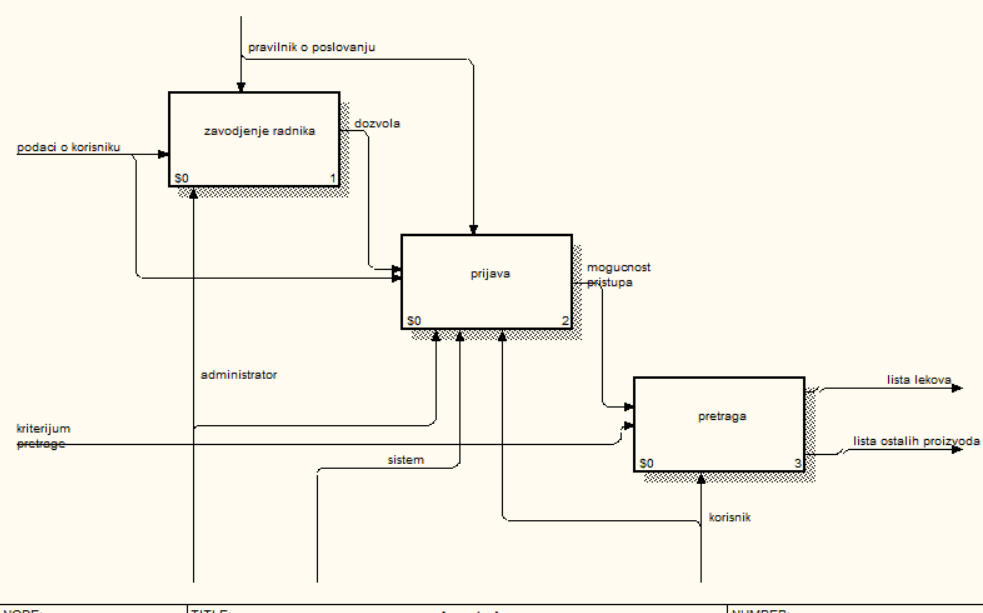
Strelice na desnoj strani pravougaonika definišu izlaze, tj rezultate izvođenja aktivnosti. Nakon pretrage mi dobijama izlaze za pretragu po leku i ostalim porizvodima, u ovom slučaju lista lekova i lista ostalih porizvoda.

Strelice na donjoj strani pravougaonika su mehanizmi, nešto što koristimo u aktivnostima ali se ne menja. To su sistem, administrator i korisnik.



Slika 1: Dijagram konteksta

**2.2 Dekompozicioni dijagrami**

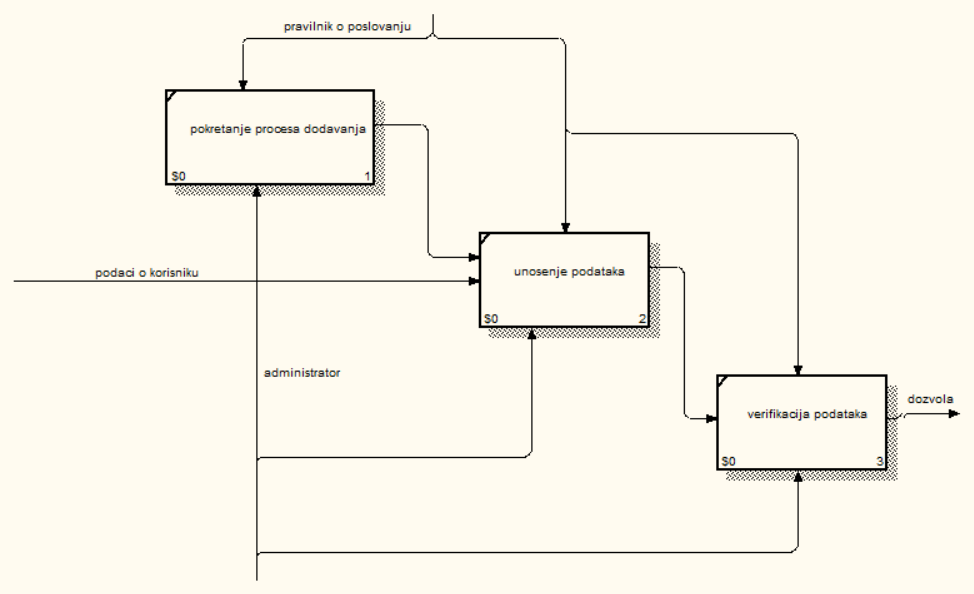


Slika 2: Prvi nivo dekompozije dijagrama konteksta

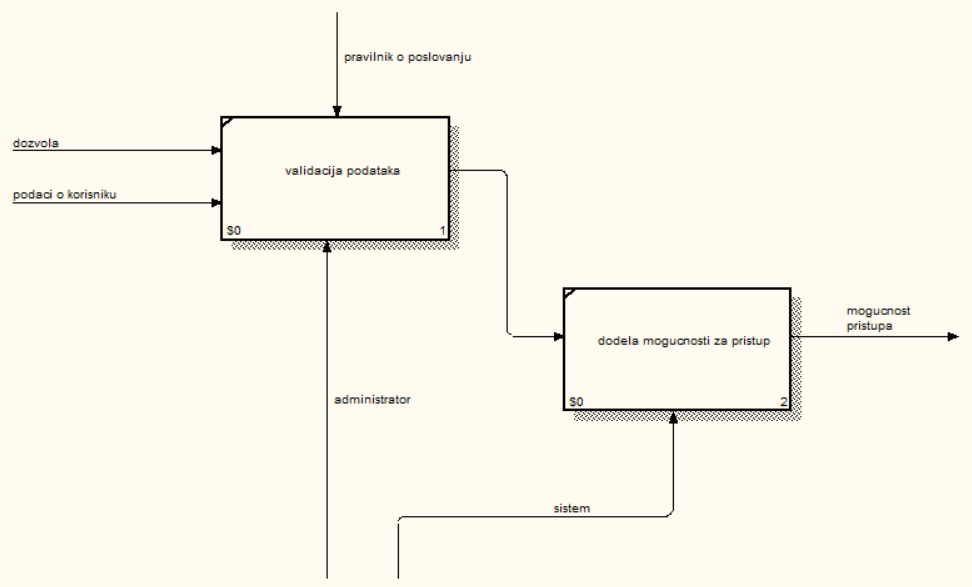
Procesi definisani prvim nivoom dekompozicije dijagrama konteksta su zavođenje radnika, prijava i pretraga.

Zavođenje radnika je bitan proces koji administratori moraju da odrade poštujući pravilnike, zakon i privatnost radnika. Samo radnici koji rade u toj apoteci imaju pristuj njenoj bazi podataka. Ako dobiju dozvolu, mogu da se prijave. Prijava je obavezan proces da bi se sistemu moglo pristupiti.

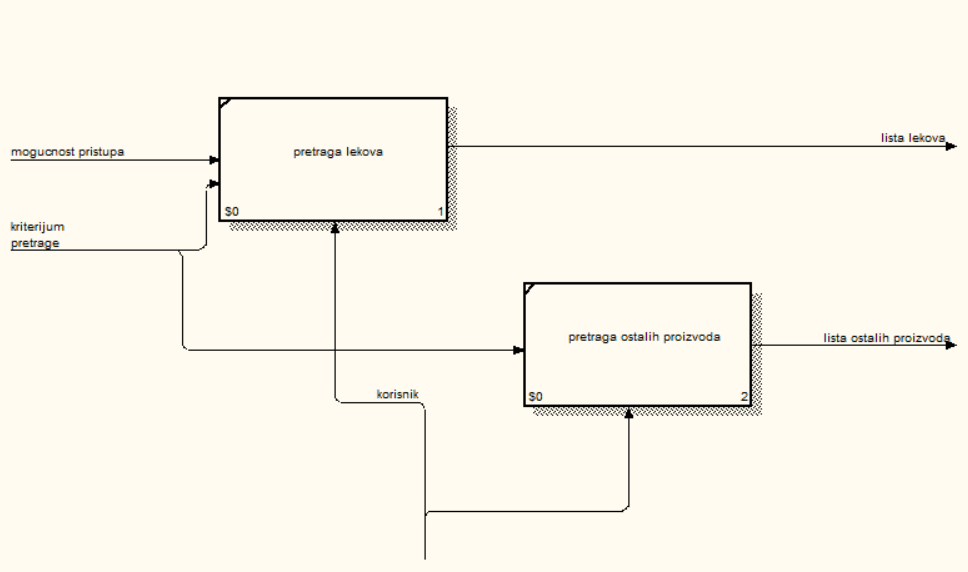
Procesu pretrage pristupa se sa mogućnošću pristupa istoj zajedno sa kriterijumima pretrage, na osnovu koje dobijamo izlaz.



Slika 3: Dekompozicija procesa zavođenja radnika



Slika 4: Dekompozicija procesa prijave



Slika 5: Dekompozicija procesa pretrage

Slike 2, 3, 4 su dijagrami dekompozije drugog reda.

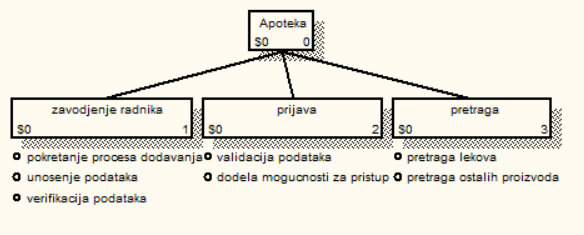
Dekompozicijom procesa zavođenje radnika dobijaju se procesi: pokretanje procesa dodavanja, unošenje podataka, verifikacija podataka.

Dekompozicijom procesa prijave dobijaju se procesi:validacija podataka i dodela mogućnosti za pristup.

Dekompozicijom procesa pretrage dobijaju se procesi: pretraga lekova i pretraga ostalih proizvoda.

**2.3 Stablo aktivnosti**

Stablo aktivnosti predstavlja složenu aktivnost rastavljenu na više podređenih aktivnosti, predstavljenu strukturom stabla. Dobija se na osnovu dijagrama dekompozicije i predstavlja hijerarhiju definisanih aktivnosti, očišćenu od strelica.



Slika 6: Stablo aktivnosti

**3. ER dijagram**

Modelovanje sistema započinjemo konstruisanjem ER dijagrama. Zatim identifikujemo entitete, atribute i sve veze između entiteta. ER dijagram je kontruisan u programu Edraw Max.

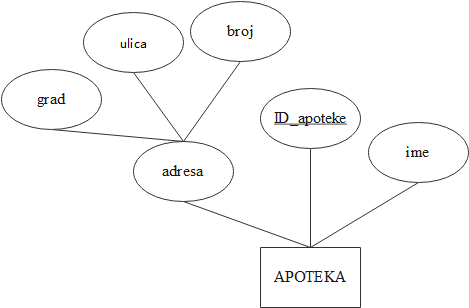
U daljem tekstu su opisani i prikazani svi entiteti i atributi, kao i veze između entiteta.

**3.1 Entiteti**

Identifikovali smo sledeće entitete:

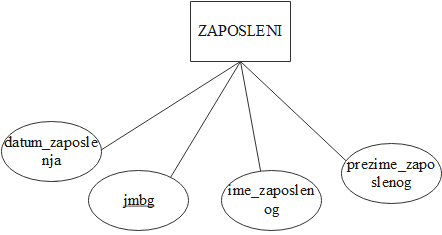
**Apoteka:**

Entitet *apoteka* predstavlja jednu apoteku koja sadrži identifikator *ID\_apoteke*, atribute *ime* i *adresu* (koju čine podaci g*rad, ulica i broj*).



Slika 7: ER dijagram entiteta *apoteka*

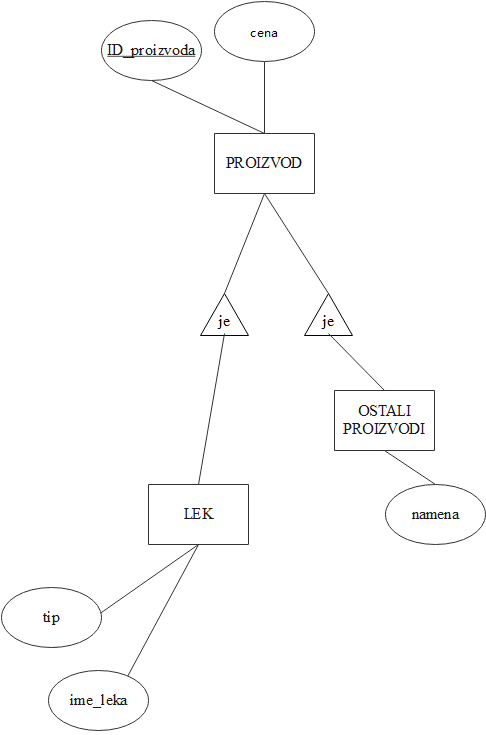
**Zaposleni:**  
Entitet *zaposleni* ima svoj identifikator *jmbg* i dodatne atribute *ime\_zaposlenog*, *prezime\_zaposlenog* i *datum*\_*zaposlenja*.



Slika 8: ER dijagram entiteta *zaposleni*

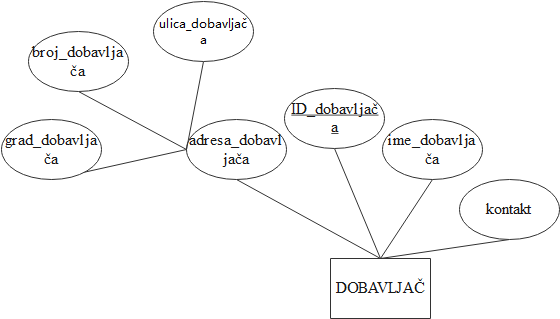
**Proizvodi i tipovi proizvoda:**

Entitet *proizvod* sadrži svoj identifikator *ID\_proizvoda* i atribut *cena.* Ovaj entitet ne sadrži nijedan više atribut o proizvodu, jer proizvod može da bude lek ili neki drugi proizvod (u nasem dijagramu označeni kao ostali proizvodi) i oni predstavljaju posebne entitete, tj podentitete entiteta *proizvod*. Podentiteti *lek* i *ostali proizvodi*, nasleđuju ključ entiteta *proizvod* i njegove atribute. Takođe imaju i svoje atribute. Entitet *lek* sadrži podatke proizvoda kao što je *ime\_leka* i *tip.* Pod tipom se podrazumeva, da li taj lek ide na recept ili se kupuje. Entitet *ostali proizvodi* ima svoj atribut *namena.* Namena u ovom entitetu određuje da li je ovaj prozvod namenjen ishrani (zdrava hrana, posebna hrana za ljude sa intolerancijom i sl.), negi lica i kože (različita vrsta kozmetike), medicinskoj opremi (medicinske papuče i mantili) ili su to proizvodi poput žvaka, naočara za vid, čajeva itd.



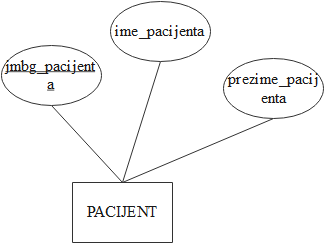
Sika 9: ER dijagram tipova entiteta *proizvod*

**Dobavljač:**  
Entitet dobavljač predstavlja dobavljača proizvoda koji ima svoj identifikator *ID\_dobavljača,* kao i atribute *ime\_dobavljača, adresa\_dobavljača* (*ulica\_dobavljača, broj\_dobavljača i grad\_dobavljača*)i *kontakt.* Pod kontaktom podrazumevamo broj telefona.



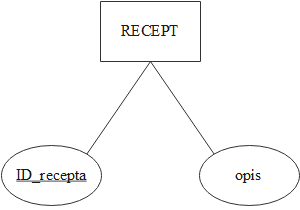
Slika 10: ER dijagram entiteta *dobavljač*

**Pacijent:**Entitet *pacijent*pored svog identifikatora *jmbg\_pacijenta***,** sadrži i atribute *ime\_pacijenta* i *prezime\_pacijenta.*



Slika 11: ER dijagram entiteta *pacijent*

**Recept:**Entitet *recept* ima svoj identifikator *ID\_recepta* i atribut *opis*. Pod opisom se podrazumeva preporučena doza koju je lekar propisao (na koliko sati i koliko tačno uzima određeni lek).

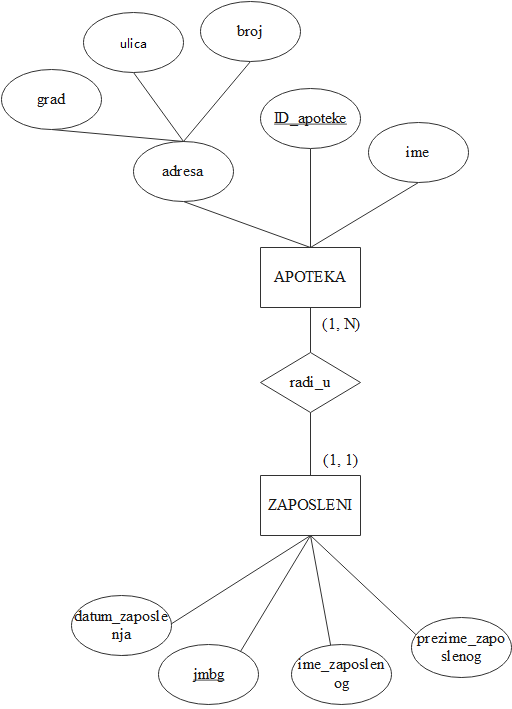


Slika 12: ER dijagram entiteta *recept*

**3.2 Veze**

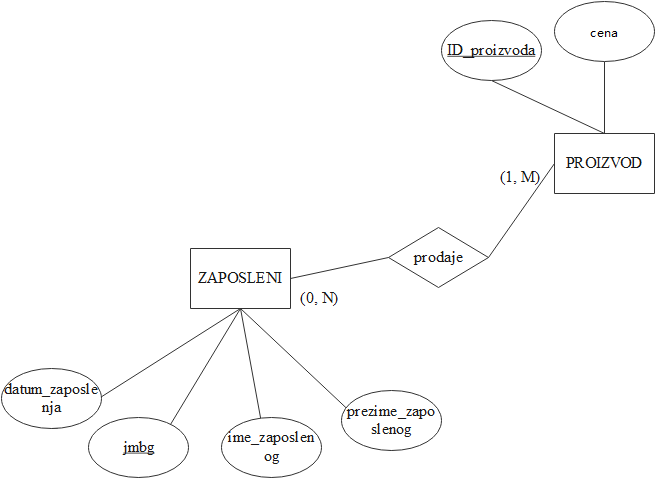
Identifikovali smo sledeće veze:

**Veza *apoteka --- zaposleni*:** Veza sa strane entiteta *apoteka* je 1,N i u ovom slučaju ovaj entitet mora da ima najmanje jednog zaposlenog, a najčešće ima više.  
Veza sa strane entiteta *zaposleni* je 1,1. To znači da jedan zaposleni radi tačno i samo u jednoj apoteci. Kardinalnost veze je 1:N. Nazv ove veze je *radi\_u.*



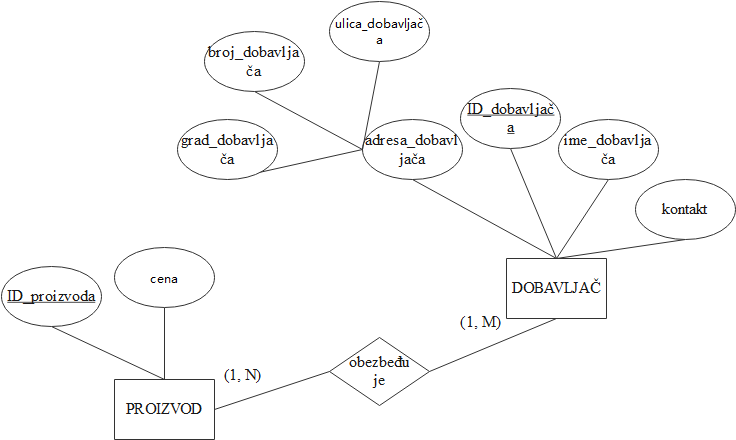
Slika 13: ER dijagram veze *radi\_u* entiteta *apoteka* i *zaposleni*

**Veza *zaposleni* --- *proizvod*:** Svaki proizvod (koji se proda) će prodati bar jedan ili više farmaceuta ali svaki farmaceut može da ne proda nijedan proizvod a može da ih proda i više. Veza je parcijalna sa strane entiteta *zaposleni*. Naziv veze je *prodaje* a njena kardinalnost je 1:N.



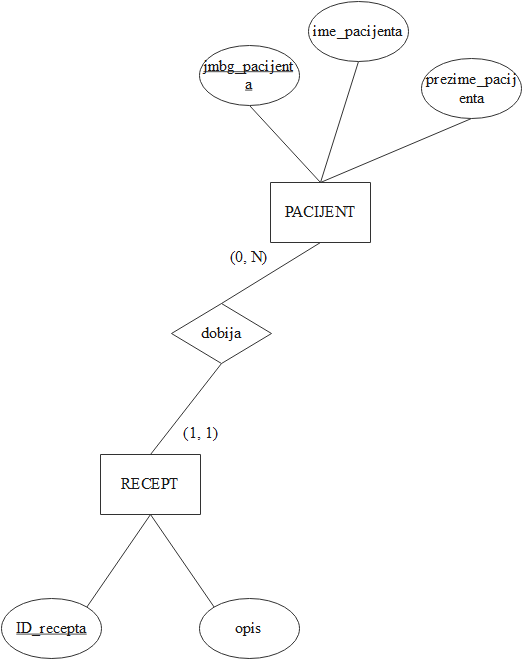
Slika 14: ER dijagram veze *prodaje* entiteta *zaposleni* i *proizvod*

**Veza*****proizvod* --- *dobavljač***: Određeni proizvod (npr lek ili ortopedske papuče), može da obezbedi više različitih dobavljača, dok jedan dobavljač, može da obezbeđuje više različitih proizvoda, ali mora bar jedan ako želimo da on bude u našoj bazi podataka. Kardinalnost ove veze je N:M. Naziv veze je *obezbeđuje*.



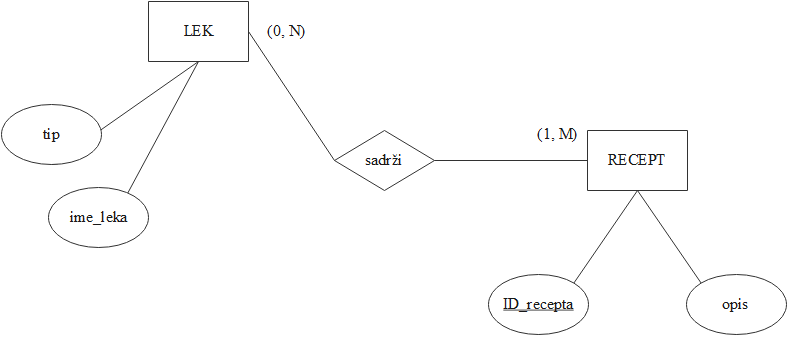
Slika 15: ER dijagram veze *obezbeđuje* entiteta *proizvod* i *dobavljač*

**Veza *pacijent*--- *recept*:** Veza sa strane entitet *pacijent* je 0,N, što znači da pacijentu može da se prepiše lek na recept ili više lekova, ali i ne mora nijedan ukoliko se npr ti lekovi kupuju. Veza sa strane *recept* je 1,1 što znači da je jednom pacijentu na tačno jednom receptu propisani tačno jedan lek. Kardinalnost veze je 1:N, dok je njen naziv *dobija*.



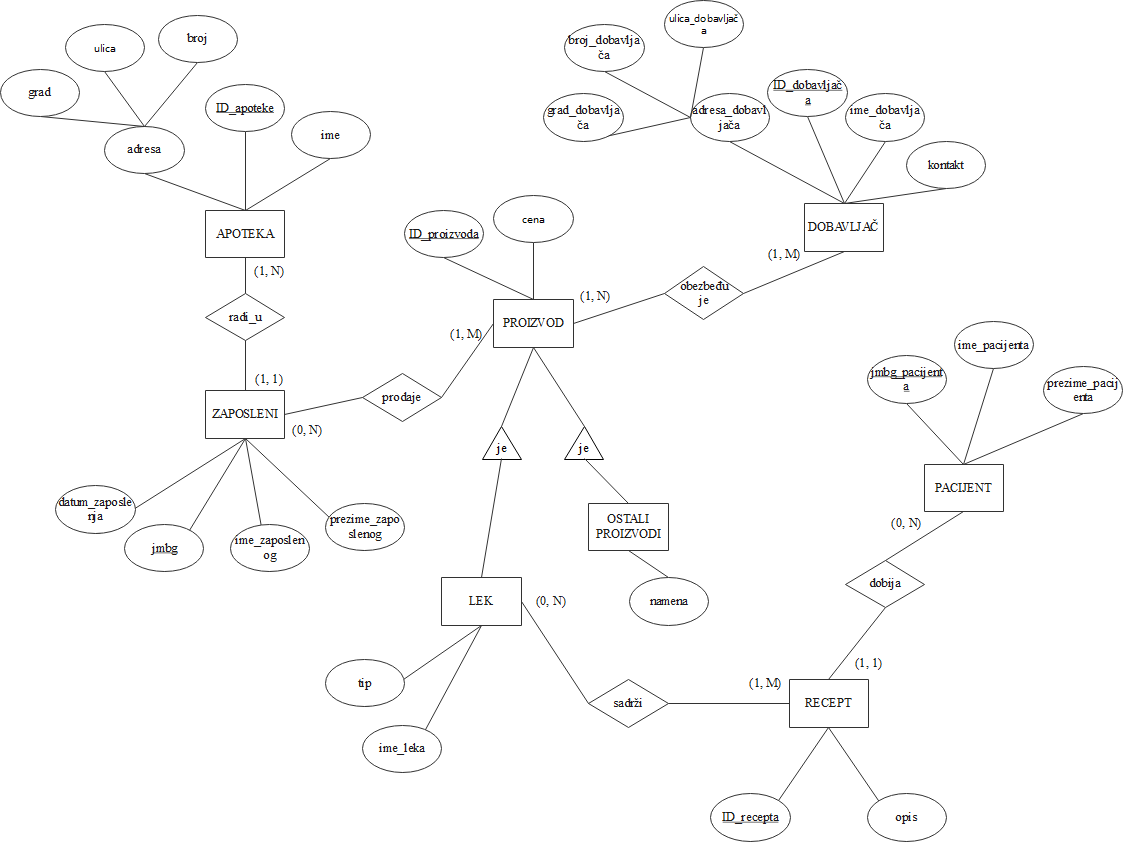
Slika 16: ER dijagram veze *dobija* entiteta *pacijent* i *recept*

**Veza *recept* --- *lek* :** Recept da bi postojao u našoj bazi, mora da propiše bar jedan ili više lekova, a neki lek može ali i ne mora da bude propisan receptom. Kardinalnost ove veze je N:M. Njen naziv je *sadrži*.



Slika 17: ER dijagram veze *sadrži* entiteta *recept* i *lek*

**3.3 Kompletan ER dijagram**



Slika 18: ER dijagram komplentnog sistema

**4. Logička šema relacione baze podataka**

Potrebno je da prvo sve entitete i veze iz ER modela prevedemo u relacioni model. Formiraćemo relacije i međurelaciona ograničenja.

**4.1 Entiteti**

Svaki enitet u našoj šemi postaje nezavisna šema relacije. Ime entiteta posta ime šeme relacije. Identifikator entiteta postaje *primarni ključ* šeme relacije. Obeležja tipa objekta su obeležja šeme relacije. U slučaju da imamo složen atribut (npr *adresa*), dolazi do pretvaranja u 1NF. Svaki atribut je atomičan.

Dobijamo sledeće:

Apoteka(ID\_apoteke, ime, grad, ulica, broj)

Zaposleni(jmbg, datum\_zaposlenja, ime\_zaposlenog, prezime\_zaposlenog, **ID\_apoteke**)

Proizvod(ID\_proizvoda, cena)

Dobavljač(ID\_dobavljača, grad\_dobavljača, ulica\_dobavljača, broj\_dobavljača, ime\_dobavljača, kontakt)

Pacijent(jmbg\_pacijenta, ime\_pacijenta, prezime\_pacijenta)

Recept(ID\_recepta, opis, **jmbg\_pacijenta**)

Ukoliko imamo entitete sa podentitetima, podentitet postaje šema relacije. Ime podentiteta postaje ime šeme relacije, a identifikator entiteta predstavlja primarni ključ podentiteta.

Lek(ID\_proizvoda, tip, ime\_leka)

OstaliProizvodi(ID\_proizvoda, namena)

**4.2 Veze**

U zavisnosti od kardinalnosti, na veze se primenjuju različita pravila prevođenja.

Relacija *Prodaje* nastaje od veze *prodaje* entiteta  *zaposleni* i *proizvod*. Kardinalnost ove veze je (0,N):(1,M), što znači da u ovom slučaju veza postaje posebna šema relacije. Obeležja ove šeme relacija su identifikatori objekata koji su u vezi, a ključ šeme relacije je složeni ključ koji se sastoji od identifikatora objekata koji su u vezi.

Prodaje(jmbg, ID\_proizvoda)

Sledeća relacija koja nastaje prebacivanjem veze entiteta *proizvod* i *dobavljač* sa kardinalnošću (1,N):(1,M), nastaje na gotovo identičan način kao prethodni primer. Ključ ove relacije je složeni ključ koji se sastoji od identifikatora entiteta *proizvod* i *dobavljač.*

Obezbeđuje(ID\_proizvoda, ID\_dobavljača)

Relacija *Sadrži* nastaje od veze *sadrži* entiteta *lek* i *recept*. Njena kardinalost je (0,N):(1,M) i relacija se dobija na potpuno isti način kao prethodne dve.

Sadrži(ID\_proizvoda, ID\_recepta)

**4.3 Međurelaciona ograničenja**

Pošto su dve relacije normalizovane, mogu se definisati sledeća međurelaciona ograničenja.

Zaposleni[ID\_apoteke]⊆Apoteka[ID\_apoteke]

Recept[jmbg\_pacijenta]⊆Pacijent[jmbg\_pacijenta]

Prodaje[jmbg]⊆Zaposleni[jmbg]

Prodaje[ID\_proizvoda]⊆Proizvod[ID\_proizvoda]

Obezbeđuje[ID\_proizvoda] ⊆ Proizvod[ID\_proizvoda]

Obezbeđuje[ID\_dobavljača] ⊆ Dobavljač[ID\_dobavljača]

Sadrži[ID\_proizvoda] ⊆ Proizvod[ID\_proizvoda]

Sadrži[ID\_recepta] ⊆ Recept[ID\_recepta]

**4.4 Kompletna logička šema**

Na osnovu dosadašnjeg izlaganja dolazi se do kompletnog relacionog modela sistema:S={

Apoteka(ID\_apoteke, ime, grad, ulica, broj)

Zaposleni(jmbg, datum\_zaposlenja, ime\_zaposlenog, prezime\_zaposlenog, **ID\_apoteke**)

Proizvod(ID\_proizvoda, cena)

Dobavljač(ID\_dobavljača, grad\_dobavljača, ulica\_dobavljača, broj\_dobavljača, ime\_dobavljača, kontakt)

Pacijent(jmbg\_pacijenta, ime\_pacijenta, prezime\_pacijenta)

Recept(ID\_recepta, opis, **jmbg\_pacijenta**)

Prodaje(jmbg, ID\_proizvoda)

Obezbeđuje(ID\_proizvoda, ID\_dobavljača)

Sadrži(ID\_proizvoda, ID\_recepta)

Lek(ID\_proizvoda, tip, ime\_leka)

OstaliProizvodi(ID\_proizvoda, namena)

}

I={

Zaposleni[ID\_apoteke]⊆Apoteka[ID\_apoteke]

Recept[jmbg\_pacijenta]⊆Pacijent[jmbg\_pacijenta]  
Prodaje[jmbg]⊆Zaposleni[jmbg]  
Prodaje[ID\_proizvoda]⊆Proizvod[ID\_proizvoda]  
Obezbeđuje[ID\_proizvoda] ⊆ Proizvod[ID\_proizvoda]

Obezbeđuje[ID\_dobavljača] ⊆ Dobavljač[ID\_dobavljača]

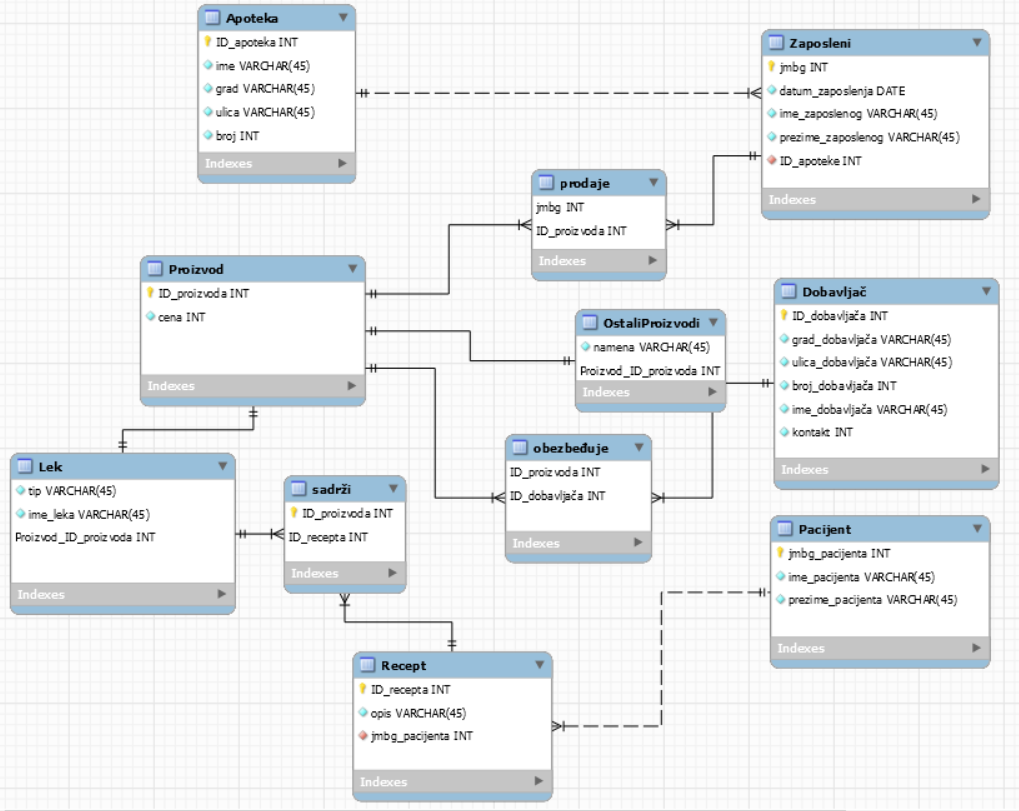
Sadrži[ID\_proizvoda] ⊆ Proizvod[ID\_proizvoda]

Sadrži[ID\_recepta] ⊆ Recept[ID\_recepta]

}

**5. Fizička šema relacione baze podataka**

Fizička šema predstavlja najniži nivo apstrakcije podataka u bazi. Ona je jedan korak iza implementacije samog sistema. Za izadu fizičke šeme korišćen je program MySQL Workbench.



Slika 19: Fizička šema realnog sistema

**6. Implementacija projektovane baze podataka sa testnim podacima u MySQL**

Pošto MySQL ima opciju da sam generiše SQL kod koji implementira projektovanu fizičku šemu baze podataka, to će biti prikazano na sledećim stranama, zajedno sa testnim podacima koji su ručno ubačeni.

-- MySQL Script generated by MySQL Workbench

-- Sun Jun 28 15:52:39 2020

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`Apoteka`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Apoteka` (

`ID\_apoteka` INT NOT NULL,

`ime` VARCHAR(45) NOT NULL,

`grad` VARCHAR(45) NOT NULL,

`ulica` VARCHAR(45) NOT NULL,

`broj` INT NOT NULL,

PRIMARY KEY (`ID\_apoteka`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Zaposleni`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Zaposleni` (

`jmbg` INT NOT NULL,

`datum\_zaposlenja` DATE NOT NULL,

`ime\_zaposlenog` VARCHAR(45) NOT NULL,

`prezime\_zaposlenog` VARCHAR(45) NOT NULL,

`ID\_apoteke` INT NOT NULL,

PRIMARY KEY (`jmbg`),

INDEX `ID\_apoteke\_idx` (`ID\_apoteke` ASC) VISIBLE,

CONSTRAINT `ID\_apoteke`

FOREIGN KEY (`ID\_apoteke`)

REFERENCES `mydb`.`Apoteka` (`ID\_apoteka`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Proizvod`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Proizvod` (

`ID\_proizvoda` INT NOT NULL,

`cena` INT NOT NULL,

PRIMARY KEY (`ID\_proizvoda`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`prodaje`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`prodaje` (

`jmbg` INT NOT NULL,

`ID\_proizvoda` INT NOT NULL,

PRIMARY KEY (`jmbg`, `ID\_proizvoda`),

INDEX `ID\_proizvoda\_idx` (`ID\_proizvoda` ASC) VISIBLE,

CONSTRAINT `jmbg`

FOREIGN KEY (`jmbg`)

REFERENCES `mydb`.`Zaposleni` (`jmbg`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `ID\_proizvoda`

FOREIGN KEY (`ID\_proizvoda`)

REFERENCES `mydb`.`Proizvod` (`ID\_proizvoda`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Dobavljač`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Dobavljač` (

`ID\_dobavljača` INT NOT NULL,

`grad\_dobavljača` VARCHAR(45) NOT NULL,

`ulica\_dobavljača` VARCHAR(45) NOT NULL,

`broj\_dobavljača` INT NOT NULL,

`ime\_dobavljača` VARCHAR(45) NOT NULL,

`kontakt` INT NOT NULL,

PRIMARY KEY (`ID\_dobavljača`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`obezbeđuje`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`obezbeđuje` (

`ID\_proizvoda` INT NOT NULL,

`ID\_dobavljača` INT NOT NULL,

PRIMARY KEY (`ID\_proizvoda`, `ID\_dobavljača`),

INDEX `ID\_dobavljača\_idx` (`ID\_dobavljača` ASC) VISIBLE,

CONSTRAINT `ID\_proizvoda`

FOREIGN KEY (`ID\_proizvoda`)

REFERENCES `mydb`.`Proizvod` (`ID\_proizvoda`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `ID\_dobavljača`

FOREIGN KEY (`ID\_dobavljača`)

REFERENCES `mydb`.`Dobavljač` (`ID\_dobavljača`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Pacijent`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Pacijent` (

`jmbg\_pacijenta` INT NOT NULL,

`ime\_pacijenta` VARCHAR(45) NOT NULL,

`prezime\_pacijenta` VARCHAR(45) NOT NULL,

PRIMARY KEY (`jmbg\_pacijenta`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Recept`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Recept` (

`ID\_recepta` INT NOT NULL,

`opis` VARCHAR(45) NOT NULL,

`jmbg\_pacijenta` INT NOT NULL,

PRIMARY KEY (`ID\_recepta`),

INDEX `jmbg\_pacijenta\_idx` (`jmbg\_pacijenta` ASC) VISIBLE,

CONSTRAINT `jmbg\_pacijenta`

FOREIGN KEY (`jmbg\_pacijenta`)

REFERENCES `mydb`.`Pacijent` (`jmbg\_pacijenta`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Lek`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Lek` (

`tip` VARCHAR(45) NOT NULL,

`ime\_leka` VARCHAR(45) NOT NULL,

`Proizvod\_ID\_proizvoda` INT NOT NULL,

PRIMARY KEY (`Proizvod\_ID\_proizvoda`),

CONSTRAINT `fk\_Lek\_Proizvod1`

FOREIGN KEY (`Proizvod\_ID\_proizvoda`)

REFERENCES `mydb`.`Proizvod` (`ID\_proizvoda`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`sadrži`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`sadrži` (

`ID\_proizvoda` INT NOT NULL,

`ID\_recepta` INT NOT NULL,

PRIMARY KEY (`ID\_proizvoda`, `ID\_recepta`),

INDEX `ID\_recepta\_idx` (`ID\_recepta` ASC) VISIBLE,

CONSTRAINT `ID\_proizvoda`

FOREIGN KEY ()

REFERENCES `mydb`.`Lek` ()

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `ID\_recepta`

FOREIGN KEY (`ID\_recepta`)

REFERENCES `mydb`.`Recept` (`ID\_recepta`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`OstaliProizvodi`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`OstaliProizvodi` (

`namena` VARCHAR(45) NOT NULL,

`Proizvod\_ID\_proizvoda` INT NOT NULL,

PRIMARY KEY (`Proizvod\_ID\_proizvoda`),

CONSTRAINT `fk\_OstaliProizvodi\_Proizvod1`

FOREIGN KEY (`Proizvod\_ID\_proizvoda`)

REFERENCES `mydb`.`Proizvod` (`ID\_proizvoda`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

-- -----------------------------------------------------

-- Data for table `mydb`.`Apoteka`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Apoteka` (`ID\_apoteka`, `ime`, `grad`, `ulica`, `broj`) VALUES (1, 'Nana', 'Kagujevac', 'Stojana Protića', 70);

INSERT INTO `mydb`.`Apoteka` (`ID\_apoteka`, `ime`, `grad`, `ulica`, `broj`) VALUES (2, 'Kalenić', 'Beograd', 'Maksima Gorkog ', 21);

INSERT INTO `mydb`.`Apoteka` (`ID\_apoteka`, `ime`, `grad`, `ulica`, `broj`) VALUES (3, 'Laurus', 'Novi Sad', 'Futoška', 19);

INSERT INTO `mydb`.`Apoteka` (`ID\_apoteka`, `ime`, `grad`, `ulica`, `broj`) VALUES (4, 'Zegin', 'Niš', 'Bulevar Nemanjića', 27);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`Zaposleni`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Zaposleni` (`jmbg`, `datum\_zaposlenja`, `ime\_zaposlenog`, `prezime\_zaposlenog`, `ID\_apoteke`) VALUES (0510998725028, '20.01.2016.', 'Tijana', 'Tanasković', 1);

INSERT INTO `mydb`.`Zaposleni` (`jmbg`, `datum\_zaposlenja`, `ime\_zaposlenog`, `prezime\_zaposlenog`, `ID\_apoteke`) VALUES (0202999725018, '05.06.2017.', 'Mirko', 'Nikolić', 4);

INSERT INTO `mydb`.`Zaposleni` (`jmbg`, `datum\_zaposlenja`, `ime\_zaposlenog`, `prezime\_zaposlenog`, `ID\_apoteke`) VALUES (1012989725011, '27.07.2017.', 'Milica', 'Živadinović', 2);

INSERT INTO `mydb`.`Zaposleni` (`jmbg`, `datum\_zaposlenja`, `ime\_zaposlenog`, `prezime\_zaposlenog`, `ID\_apoteke`) VALUES (2212336987512, '08.12.2018.', 'Andrijana', 'Tomašević', 3);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`Proizvod`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Proizvod` (`ID\_proizvoda`, `cena`) VALUES (1, 550);

INSERT INTO `mydb`.`Proizvod` (`ID\_proizvoda`, `cena`) VALUES (2, 340);

INSERT INTO `mydb`.`Proizvod` (`ID\_proizvoda`, `cena`) VALUES (3, 100);

INSERT INTO `mydb`.`Proizvod` (`ID\_proizvoda`, `cena`) VALUES (4, 1350);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`prodaje`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`prodaje` (`jmbg`, `ID\_proizvoda`) VALUES (0202999725018, 2);

INSERT INTO `mydb`.`prodaje` (`jmbg`, `ID\_proizvoda`) VALUES (0510998725028, 1);

INSERT INTO `mydb`.`prodaje` (`jmbg`, `ID\_proizvoda`) VALUES (1012989725011, 4);

INSERT INTO `mydb`.`prodaje` (`jmbg`, `ID\_proizvoda`) VALUES (2212336987512, 3);

INSERT INTO `mydb`.`prodaje` (`jmbg`, `ID\_proizvoda`) VALUES (2212336987512, 1);

INSERT INTO `mydb`.`prodaje` (`jmbg`, `ID\_proizvoda`) VALUES (0510998725028, 4);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`Dobavljač`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Dobavljač` (`ID\_dobavljača`, `grad\_dobavljača`, `ulica\_dobavljača`, `broj\_dobavljača`, `ime\_dobavljača`, `kontakt`) VALUES (1, 'Kragujevac', 'Dr Jovana Ristića', 67, 'Hemofarm', 034363094);

INSERT INTO `mydb`.`Dobavljač` (`ID\_dobavljača`, `grad\_dobavljača`, `ulica\_dobavljača`, `broj\_dobavljača`, `ime\_dobavljača`, `kontakt`) VALUES (2, 'Novi Sad', 'Radnička', 100, 'Krka', 034333012);

INSERT INTO `mydb`.`Dobavljač` (`ID\_dobavljača`, `grad\_dobavljača`, `ulica\_dobavljača`, `broj\_dobavljača`, `ime\_dobavljača`, `kontakt`) VALUES (3, 'Niš', 'Apatinska', 12, 'Galenika', 034359882);

INSERT INTO `mydb`.`Dobavljač` (`ID\_dobavljača`, `grad\_dobavljača`, `ulica\_dobavljača`, `broj\_dobavljača`, `ime\_dobavljača`, `kontakt`) VALUES (4, 'Beograd', 'Borova', 54, 'AbelaPharm', 034336959);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`obezbeđuje`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`obezbeđuje` (`ID\_proizvoda`, `ID\_dobavljača`) VALUES (1, 2);

INSERT INTO `mydb`.`obezbeđuje` (`ID\_proizvoda`, `ID\_dobavljača`) VALUES (3, 4);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`Pacijent`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Pacijent` (`jmbg\_pacijenta`, `ime\_pacijenta`, `prezime\_pacijenta`) VALUES (2536958214523, 'Milos', 'Todorović');

INSERT INTO `mydb`.`Pacijent` (`jmbg\_pacijenta`, `ime\_pacijenta`, `prezime\_pacijenta`) VALUES (4251326987521, 'Jelena', 'Jevremović');

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`Recept`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Recept` (`ID\_recepta`, `opis`, `jmbg\_pacijenta`) VALUES (1, '3 puta na dan po jedan lek', 2536958214523);

INSERT INTO `mydb`.`Recept` (`ID\_recepta`, `opis`, `jmbg\_pacijenta`) VALUES (2, 'jednom ujutru jednom uvece', 4251326987521);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`Lek`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Lek` (`tip`, `ime\_leka`, `Proizvod\_ID\_proizvoda`) VALUES ('recept', 'Diane 35', 1);

INSERT INTO `mydb`.`Lek` (`tip`, `ime\_leka`, `Proizvod\_ID\_proizvoda`) VALUES ('kupuje se', 'Rupafin', 2);

INSERT INTO `mydb`.`Lek` (`tip`, `ime\_leka`, `Proizvod\_ID\_proizvoda`) VALUES ('kupuje se', 'Momensa', 3);

INSERT INTO `mydb`.`Lek` (`tip`, `ime\_leka`, `Proizvod\_ID\_proizvoda`) VALUES ('recept', 'Bromazepam', 3);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`sadrži`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`sadrži` (`ID\_proizvoda`, `ID\_recepta`) VALUES (2, 1);

INSERT INTO `mydb`.`sadrži` (`ID\_proizvoda`, `ID\_recepta`) VALUES (4, 2);

COMMIT;

-- -----------------------------------------------------

-- Data for table `mydb`.`OstaliProizvodi`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`OstaliProizvodi` (`namena`, `Proizvod\_ID\_proizvoda`) VALUES ('hrana', 5);

INSERT INTO `mydb`.`OstaliProizvodi` (`namena`, `Proizvod\_ID\_proizvoda`) VALUES ('kozmetika', 6);

INSERT INTO `mydb`.`OstaliProizvodi` (`namena`, `Proizvod\_ID\_proizvoda`) VALUES ('naočare za vid', 7);

COMMIT;

Primeri za upite:

1. Prikazati podate o zaposlenima koji rade u apoteci Nana:

SELECT \* FROM Zaposleni INNER JOIN Apoteka  
ON Apoteka.ID\_apoteke = Zaposleni.ID\_apoteke  
WHERE ime= “Nana”;

2. Prikazati lek na recept koji je uzeo pacijent Miloš Todorović:

SELECT ime\_leka, tip FROM Lek INNER JOIN  
Sadrži ON Sadrži.ID\_proizvoda = Lek.ID\_proizvoda  
INNER JOIN Recept ON Recept.ID\_recepta = Sadrži.ID\_recepta  
INNER JOIN Pacijent ON Pacijent.jmbg\_pacijenta = Recept.jmbg\_pacijenta  
WHERE ime\_pacijenta = “Miloš”  
WHERE prezime\_pacijenta = “Todorović”;

3. Prikazati lekove koji se dobavljaju iz Hemofarma:

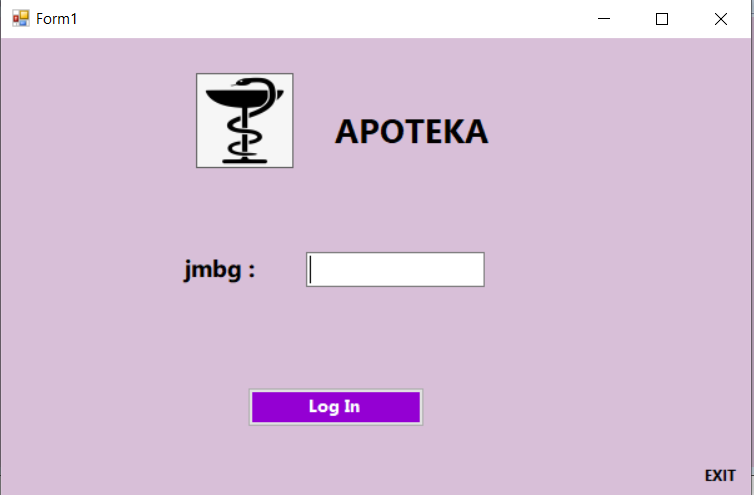
SELECT ime\_leka FROM Lek INNER JOIN Obezbeđuje  
ON Obezbeđuje.ID\_proizvoda = Lek.ID\_proizvoda  
INNER JOIN Dobavljača ON Dobavljač.ID\_dobavljača = Obezbeđuje.ID\_dobaljača  
WHERE ime\_dobavljača= “Hemofarm”;

4. Prikazati cene proizvoda koji se prodaju u apoteci Zegin u Nišu:

SELECT cena FROM Proizvod INNER JOIN Prodaje  
ON Prodaje.ID\_proizvoda = Proizvod.ID\_proizod  
INNER JOIN Zaposleni ON Zaposleni.jmbg = Prodaje.jmbg  
INNER JOIN Apoteke ON Apoteke.ID\_apoteke = Zaposleni.ID\_apoteke  
WHERE grad = “Niš” and ime = “Zegin”  
ORDER BY cena ASC;

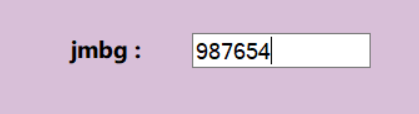
**9. Aplikacija**

Aplikacija je kreirana u jeziku c#(.NET) u okruženju Vsual Studio 2019. Fokus je na pretrazi baze podataka. Mogu je koristiti radnici u apotekama prilikom pretrage proizvoda koje imaju u istoj.

****

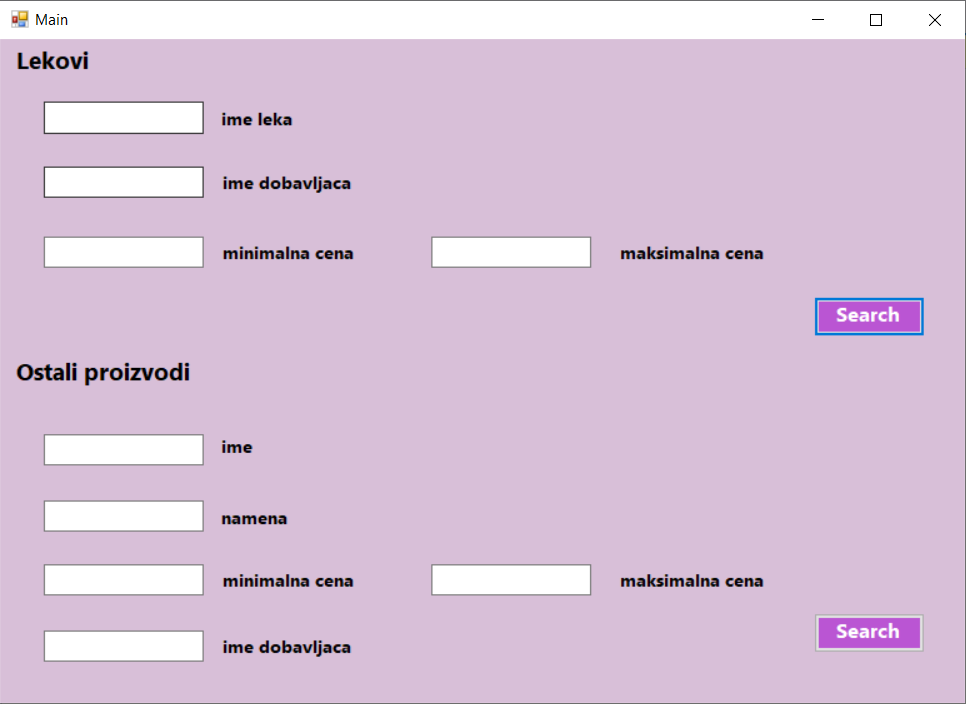
Slika 20: Izgled početnog ekrana – log in stranica

Kada se program pokrene prikazuje se ekran za prijavu. U ovu aplikaciju se mogu prijaviti samo radnici apoteke, koje je administrator već uneo u bazu podataka i to sa njihovim jedinstvenim matičnim brojem građana. Taj broj je kao što i ime kaže jedinstven, shodno time, idealan za prijavljivanje radnika, jer dva radnika nikada neće imati isti.



Slika 21: Primer unosa

Kada korisnik (radnik) unese jmbg koji je potvrđeno tačan i odgovara radniku apoteke, radnik ima pristup sledećoj strani aplikacije.



Slika 22: Stranica pretrage

Nakon prijave, korisnik može da vidi stranu pretrage. Na ovoj strani imamo dve kategorije. Kao što u bazi postoji entitet proizvod koga čine lekovi i ostali proizvodi, tako u aplikaciji imamo mogučnosti pretrage lekova i pretrage ostalih proizvoda.

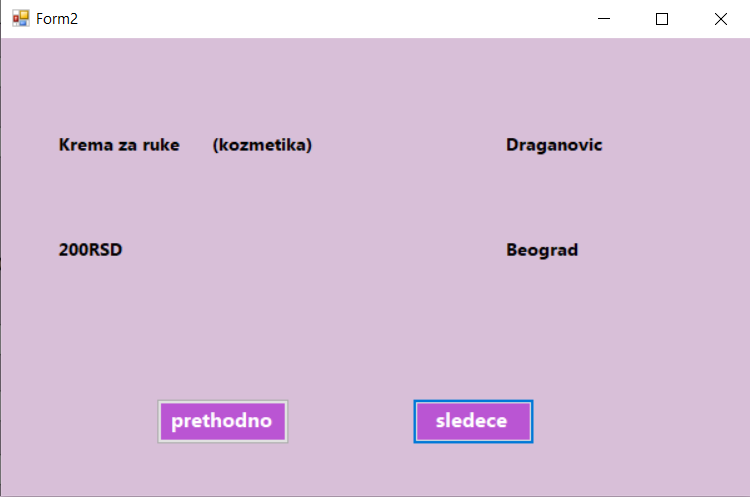
U polja sa slike možemo uneti vrednost koja je kriterijum pretrage i na osnovu nje dobijemo izlaz iz sistema. Ukoliko ne unesemo nijedan parametar klikom na dugme Search (i prvo i drugo) moći ćemo da izlistavamo sve proizvode koji se nalaze trenutno u bazi.

Ukoliko postavimo ime leka na neku vrednost (unosom) dobićemo povratnu informaciju u vidu prozora sa informacijama tog leka ili obaveštenje o tome da taj rezultat nemamo u bazi, što znači da predmet pretrage ne postoji.

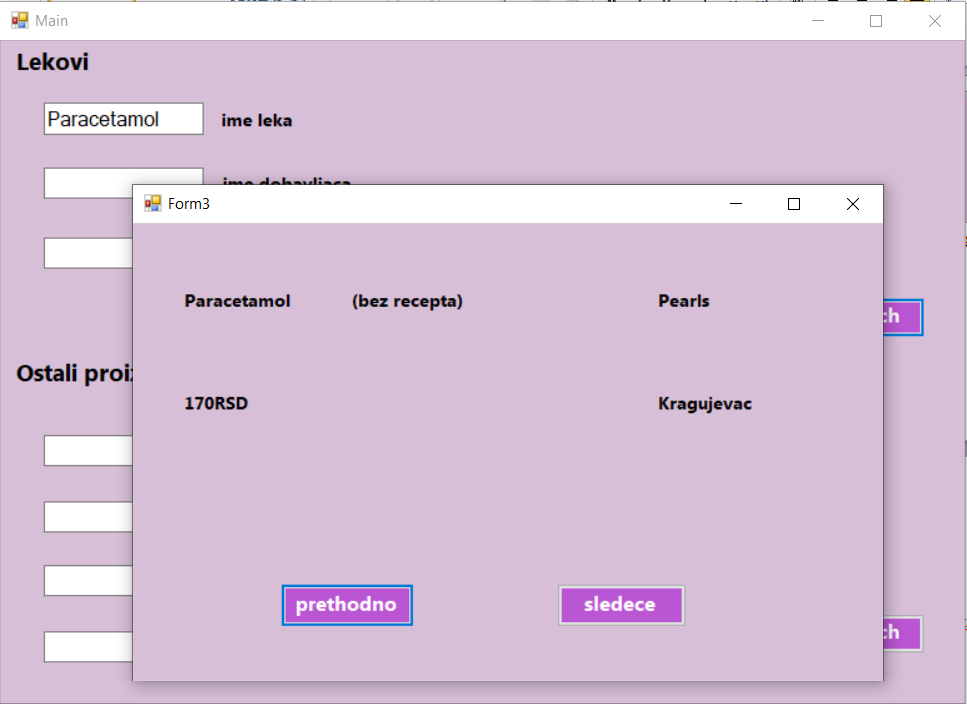
U narednim slikama ćemo detaljnije prikazati prikaze ekrana za navedene slučajeve.

Takođe prozore sa informacijama proizvoda koji zadovolje uslov menjamo tako što kliknemo na dugme prethodno (za prethodni prozor) i sledece (za naredni).

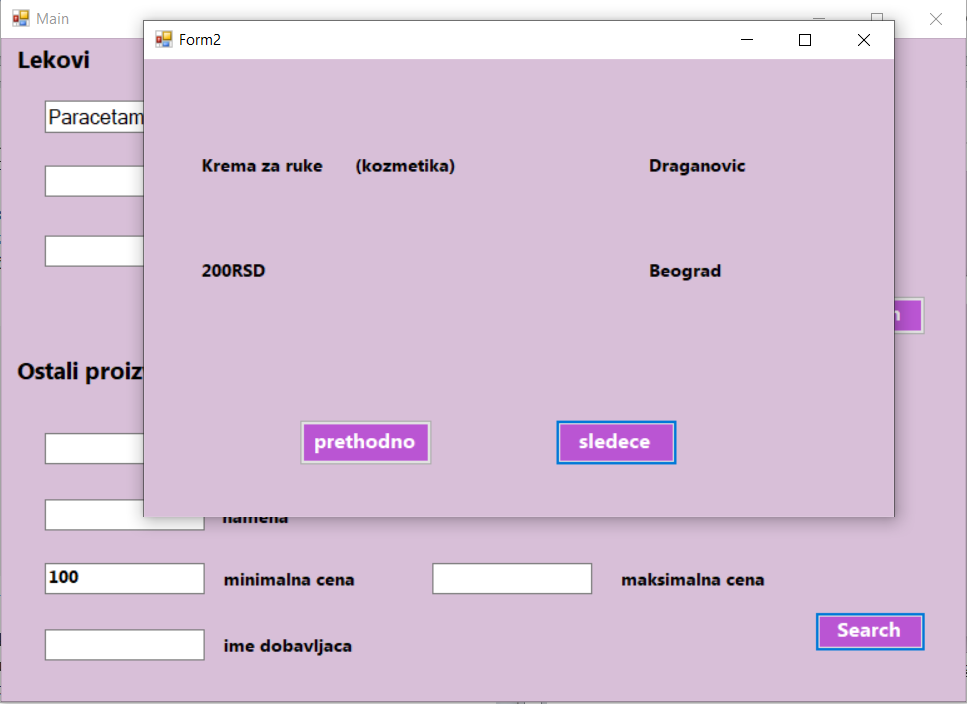
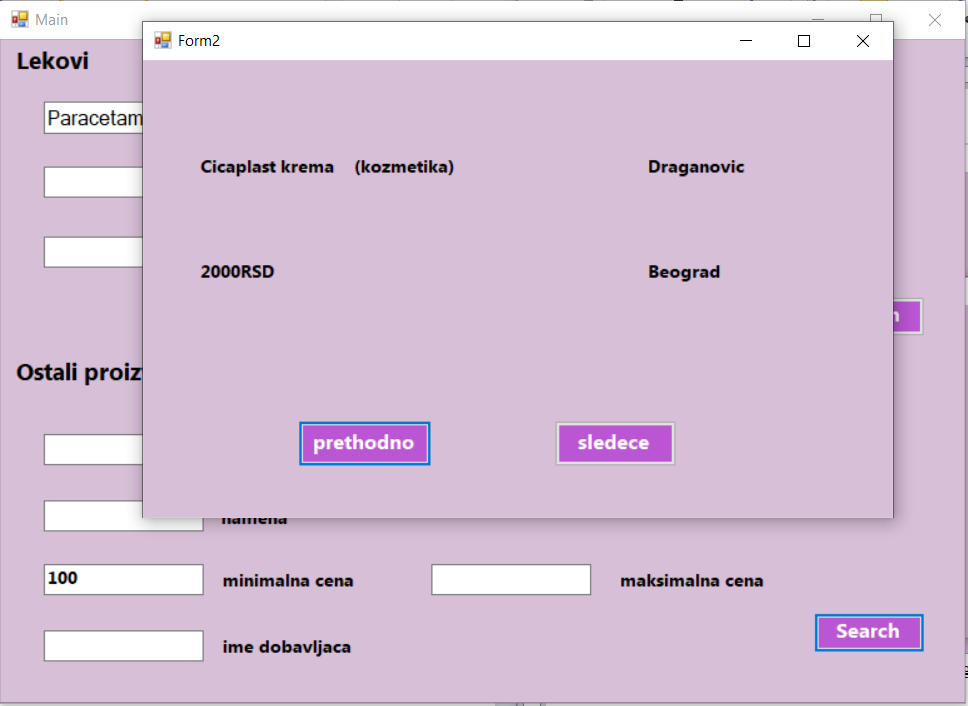




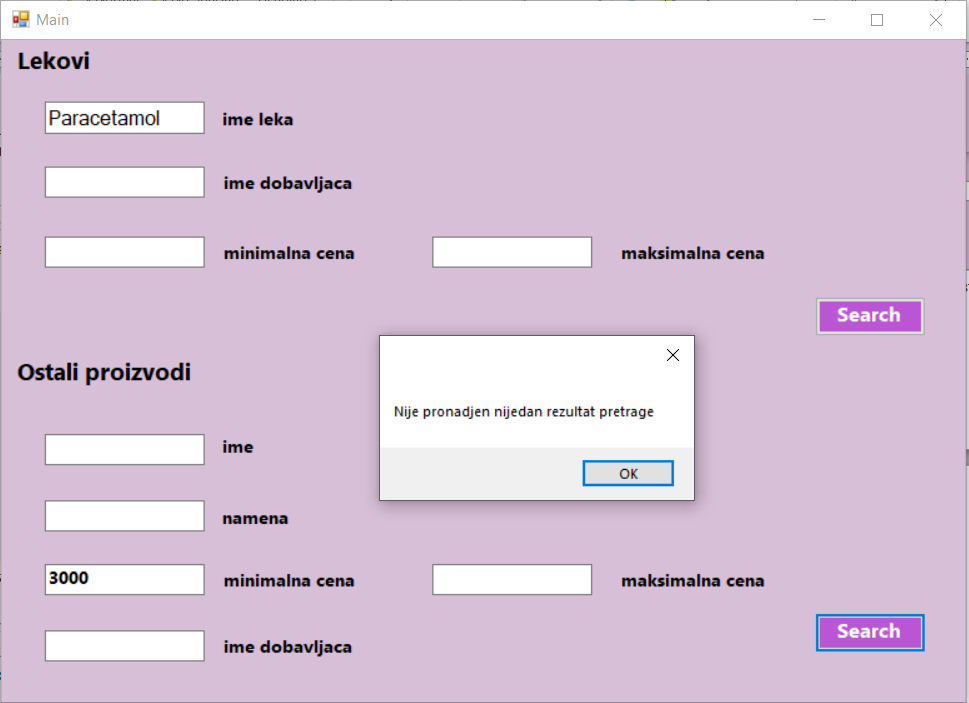
Slike 23 i 24: neki od rezultata pretrage ukoliko ne unesemo ništa u polja

****

Slika 25: Rezultat pretrage za ime leka Paracetamol

****

Slike 26 i 27 : Rezultati pretrage ostalih porizvoda za minimalnu cenu 100



Slika 28: Primer rezultata pretrage koji trenutno ne postoji u bazi a da zadovoljava uslov

**8. Literatura**

[1] Kurs: Baze podataka, 01.06.2020. <http://moodle.fink.rs/enrol/index.php?id=977>

[2] <https://www.farmacijamedicina.rs/>

[3] Kurs: Projektovanje informacionih sistema i baza podataka, 05.08.2021. <http://moodle.fink.rs/course/view.php?id=986>