

Veb aplikacija za zubarsku ordinaciju

DOKUMENTACIJA PROJEKTA



MaJ 10, 2021 , Beograd , Republika Srbija

Svetozar stanković br. indeksa 2018203764

UROŠ LJUBIMIROVIĆ BR. INDEKSA 2018203861

Sadržaj :

[1) Kratak opis projekta 2](#_Toc73532919)

[2) Postupak razvoja softvera 2](#_Toc73532920)

[3) Analiza zahteva 4](#_Toc73532921)

[4) Projektovanje sistema 11](#_Toc73532922)

[5) Implementacija softvera 24](#_Toc73532923)

[6) Testiranje softvera 24](#_Toc73532924)

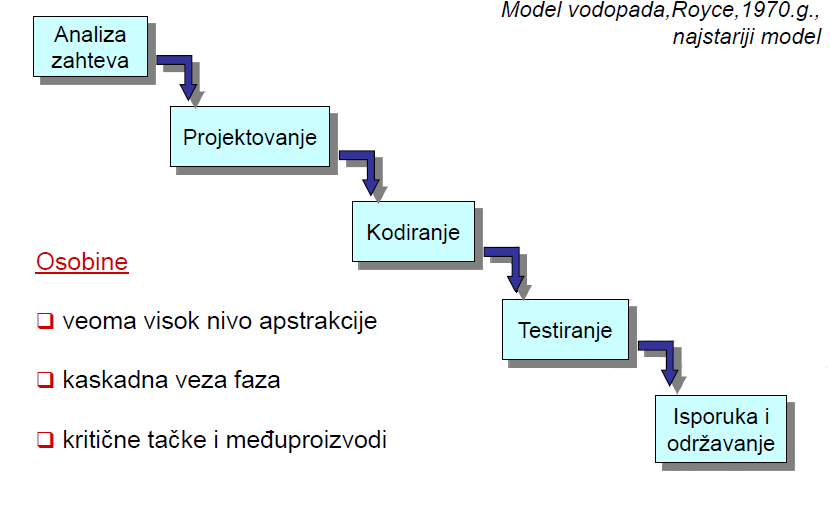
[7) Isporuka softvera 25](#_Toc73532925)

# Kratak opis projekta

Tema projekta je realizacija veb bazirane aplikacije za zubarsku ordinaciju. Kada se korisnik prijavi na aplikaciju, u panelu za uređivanje sadržaja može da doda,izmeni ili obriše određenu uslugu zubarske ordinacije. Svaka usluga ima naziv, opis, kataloški broj, kategoriju usluge (preventivna intervencija, redovna intervencija, hirurška intervencija), cenu za izdvojeno pružanje usluge, cenu za pružanje usluge u paketu, cenu za pružanje usluge deci i cenu za pružanje usluge penzionerima. U sistemu korisnik ima pristup podacima o pacijentima. Može da doda podatke o pacijentu ili da menja već unete podatke, kao i da odjavi pacijenta iz spiska aktivnih pacijenata ordinacije. Svakom pacijentu se otvara elektronski karton u kojem administrator ima prikaz zuba u vilici u obliku šeme (raspored). Korisnik (zubar) po završenoj intervenciji u aplikaciju unosi spisak pruženih usluga i obeležava na šemi vilice nad kojim zubom ili zubima je vršio intervenciju i obeležava koja je usluga u pitanju, a informacioni sistem na kraju računa automatski cenu na osnovu toga da li je pružena jedna usluga ili više , pa ih računa na osnovu cene u paketu i da li je pacijent dete,penzioner ili ne spada u jednu od te dve kategorije pacijenata. U kartonu pacijenta se mogu videti sve intervencije i pružene usluge koje su pružane od kako je pacijent upisan u informacioni sistem veb aplikacije.

# Postupak razvoja softvera

Izabrana i primenjena u procesu razvoja softvera je tradicionalna metoda modelovanja koja se zove KASKADNI MODEL (model vodopada).



U “ Kaskadnom modelu (Model vodopada) ” razvojne faze su povezane kaskadnom vezom koja se ostvaruje tako što se na narednu fazu prelazi tek nakon završetka prethodne faze. Izlaz iz prethodne faze se prosleđuje narednoj fazi kao ulaz. Ovaj model sadrži 5 faza koje su prikazane na prethodnoj slici. Svaka faza je praćena obimnom dokumentacijom, pa se često kaže da je ovaj model vođen dokumentima.

U svakoj fazi, mogu se definisati **kritične tačke**, koje predstavljaju repere na osnovu kojih se lako može pratiti izvođenje projekta. Kritične tačke mogu da budu , na primer, sastanci u zakazanom terminu na kojima se prezentuju rezultati, ili informacije da su neki moduli završeni. Osim kritičnih tačaka, u svakoj fazi mogu se definisati i međuproizvodi, na osnovu kojih se dobija uvid u trenutni stepen gotovosti projekta. Za razliku od kritičnih tačaka, koje su više informativnog karaktera, međuproizvodi predstavljaju konkretne celine, na primer, istestiran programski kod.

**Prednosti kaskadnog modela su:** jednostavnost , lako praćenje projekta, lak primena modela.

**Nedostaci kaskadnog modela su:** ne podržava povratne sprege, ne ukazuje na način povezivanja faza ,razvoj softvera se ne posmatra kao rešavanje problema, ima ograničenu interakciju sa korisnikom.

**Kritične tačke (datum početka i planiranog kraja faze) :**

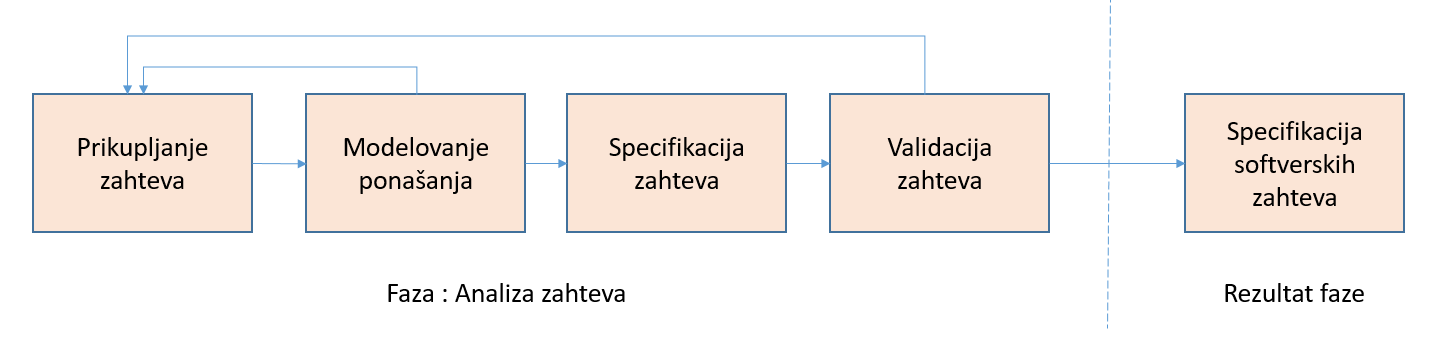
* Opis projekta u dokumentaciji **(10.05.2021. - 11.05.2021.)**
* Opis postupka razvoja softvera u dokumentaciji **(10.05.2021. - 11.05.2021.)**
* Analiza zahteva **(15.05.2021. - 25.05.2021.)**
  + Opis faza tokom prikuplanja zahteva
  + ER dijagram i objašnjenje
  + Spisak aktivnosti po ulogama (Use-case diagram)
  + Šema i prikaz svih identifikovanih entiteta sa njihovim atributima
  + Skiciran izgled aplikacije
* Projektovanje **(26.05.2021. - 29.05.2021.)**
  + Opis korišćene arhitekture i stila
  + Opis korišćenih tehnologija za realizaciju svakog sloja arhitekture
  + Dizajn baze podataka
* Kodiranje **(30.05.2021. - 2.06.2021.)**
  + Implementacija frontend-a i backend-a veb aplikacije
  + Implementacija baze podataka po projektovanom dizajnu (modelu)
* Testiranje **(3.06.2021. - 3.06.2021.)**
* Isporuka i održavanje **(3.06.2021. - 3.06.2021.)**

# Analiza zahteva

Zahtev predstavlja izraz željenog ponašanja softvera. Analizu zahteva izvodi analitičar zahteva. **Tokom analize, analitičar treba da obavi sledeće aktivnosti :**

* **Prikupljanje zahteva** (ovaj korak će biti preskočen zato što već postoji kao tekst zadatka )
* **Modelovanje ponašanja** (formiranje modela ili prototipa ponašanja sistema)
* **Specifikacija zahteva** (izrada specifikacije u okviru koje se definiše koji će delovi zahtevanog ponašanja biti implementirani u softveru)
* **Validacija i verifikacija zahteva** (provera da li specifikacija odgovara onome što naručilac očekuje od softverskog proizvoda . Ovo će takođe biti preskočen korak jer je u pitanju školski zadatak i ne postoji realan klijent da odobri specifikaciju)

Rezultat analize zahteva sprovedene kroz navedene korake je **Specifikacija softverskih zahteva ( Software Requirements Specificaton - SRS )** koja se dalje koristi za komunikaciju sa razvojnim timom.



**ER dijagram (Entity – Relationship diagram)** je grafička notacija za predstavljanje konceprualnog modela. Ovaj sistem se može korsititi ne samo za modelovanje zahteva, već i za modelovanje dizajna sistema, strukture softverskog proizvoda, baze podataka, ...

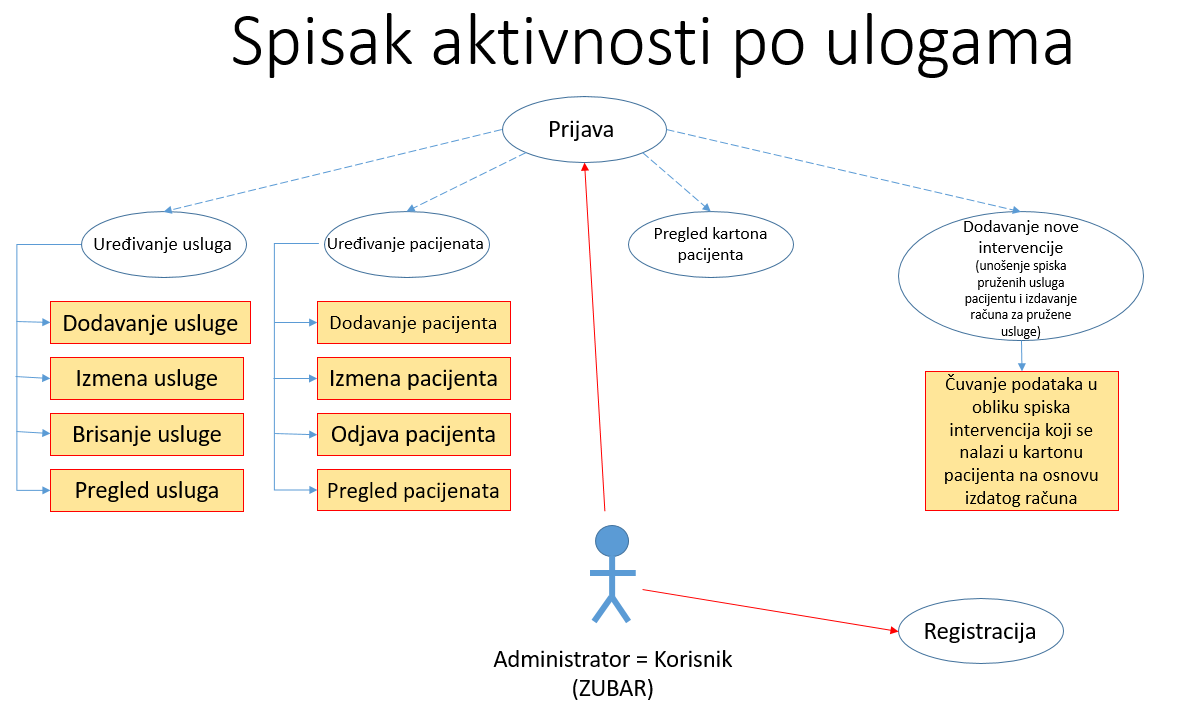
**Osnovni elementi** na ER dijagramu su : **entitet,atribut** i **relacija**. Pod entitetom se podrazumeva skup objekata (klasa objekata) iz realnog sveta koji imaju zajedničke osobine i ponašanje. Oni se na ER dijagramu predstavljaju pravugaonicima. Atributima se opisuju svojstva entiteta. Relacije definišu tip odnosa između dva entiteta i na dijagramu se predstavljaju linijom koja spaja dva entiteta na čijoj sredini se nalazi romb u kome je naveden tip veze. Na krajevima relacija često se prikazuje kardinalnost veze koja označava broj entiteta koji mogu učestvovati u vezi.

ER notacija se često koristi u složenijim pristupima. Na primer, u UML jeziku za modelovanje, ER notacija se kosristi u dijagramu klasa , pri čemu klase predstavljaju realne entitete bitne za rešavanje problema.

**Izdvojeni zahtevi iz teksta zadataka :**

1. Aplikaciji može da pristupi samo prijavljeni korisnik sa korisničkim imenom i lozinkom.
2. Svaki korisnik sistema ima svoje pristupne parametre.
3. Aplikacija vodi evidenciju o svim prijavama korisnika, kao i o svim neuspelim prijavama na sistem (pogrešno uneta lozinka naloga ili pogrešno uneto korisničko ime u formularu za prijavu) .
4. Kada se korisnik prijavi na aplikaciju, u panelu za uređivanje sadržaja može da doda,izmeni ili obriše određenu uslugu zubarske ordinacije.
5. Svaka usluga ima naziv, opis, kataloški broj, kategoriju usluge (preventivna intervencija, redovna intervencija, hirurška intervencija), cenu za izdvojeno pružanje usluge, cenu za pružanje usluge u paketu, cenu za pružanje usluge deci i cenu za pružanje usluge penzionerima.
6. U sistemu korisnik ima pristup podacima o pacijentima.
7. Može da doda podatke o pacijentu ili da menja već unete podatke, kao i da odjavi pacijenta iz spiska aktivnih pacijenata ordinacije.
8. Svakom pacijentu se otvara elektronski karton u kojem administrator ima prikaz zuba u vilici u obliku šeme (raspored).
9. informacioni sistem na kraju izvršenih intervencija nad pacijentom računa automatski cenu na osnovu toga da li je pružena jedna usluga ili više , pa ih računa na osnovu cene u paketu i da li je pacijent dete,penzioner ili ne spada u jednu od te dve kategorije pacijenata.
10. U kartonu pacijenta se mogu videti sve intervencije i pružene usluge koje su pružane od kako je pacijent upisan u informacioni sistem veb aplikacije.
11. Grafički interfejs veb sajta je realizovan sa „responsive dizajnom“ .

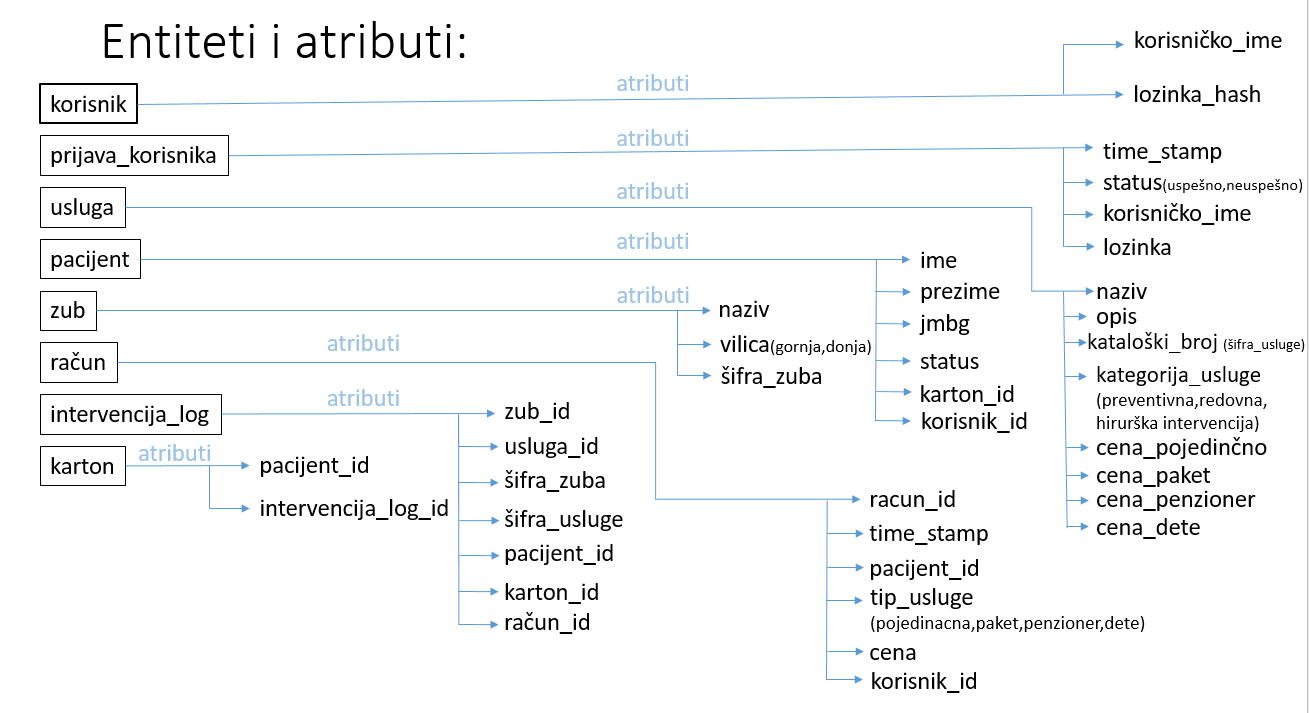
**Na sledećoj slici je prikazan dijagram slučajeva korišćenja :**

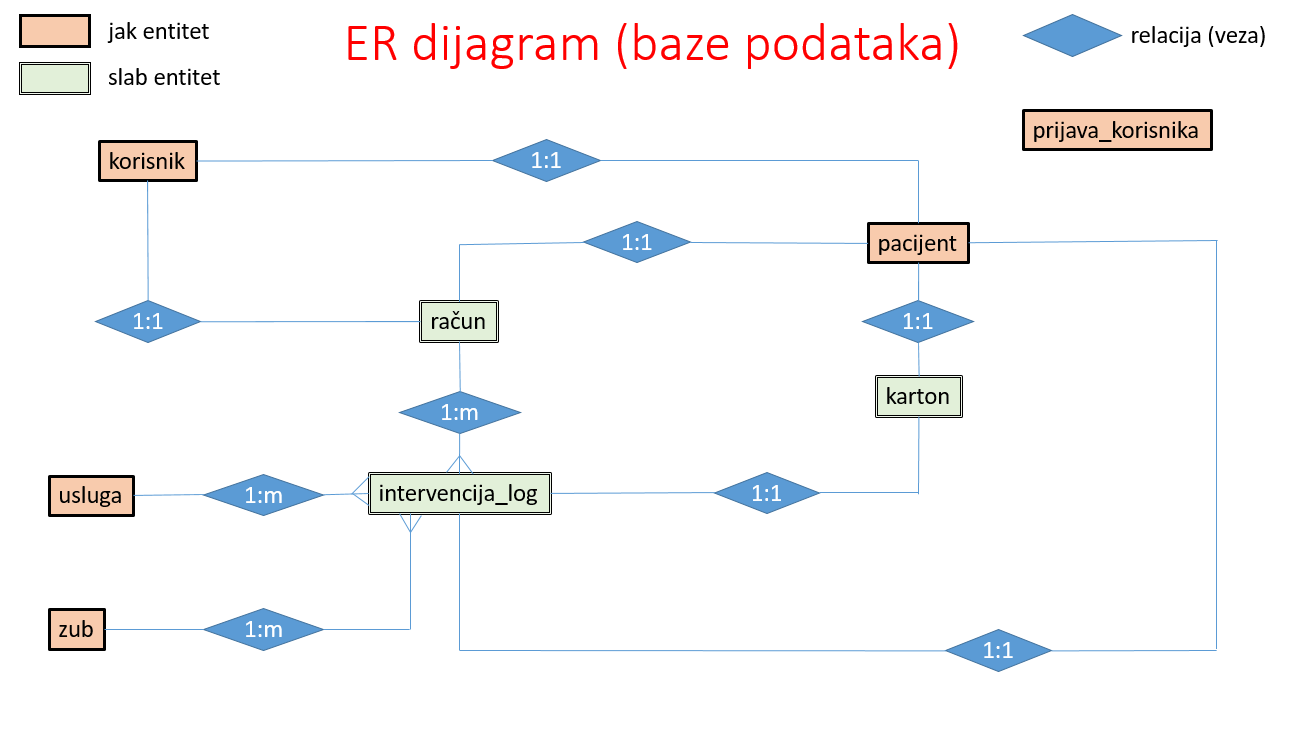


**ULOGE:**

U našem konkretnom slučaju ulogu administratora i korisnika ima jedna osoba (zubar) . On sve kontroliše i radi.

* Korisnik = Administrator (zubar)
  + Registracija
  + Prijava
  + Uređivanje usluga - (dodavanje,izmena,brisanje,pregled)
  + Uređivanje pacijenata -(dodavanje,izmena,brisanje,pregled)
  + Pregled kartona pacijenta
  + Dodavanje nove intervencije (unošenje spiska pruženih usluga pacijentu) – (čuvanje podataka u obliku izdatog računa koji se nalazi u kartonu pacijenta)





**Dodatno objašnjenje ER dijagrama :**

**prijava\_korisnika** je jak i potpuno nezavisan entitet . Nema releacije ni sa jednim drugim entitetom zato što on služi da se u njemi skladište podaci o logovanju korsinika na sistem , a ti podaci nisu od važnosti za ostale entitete.

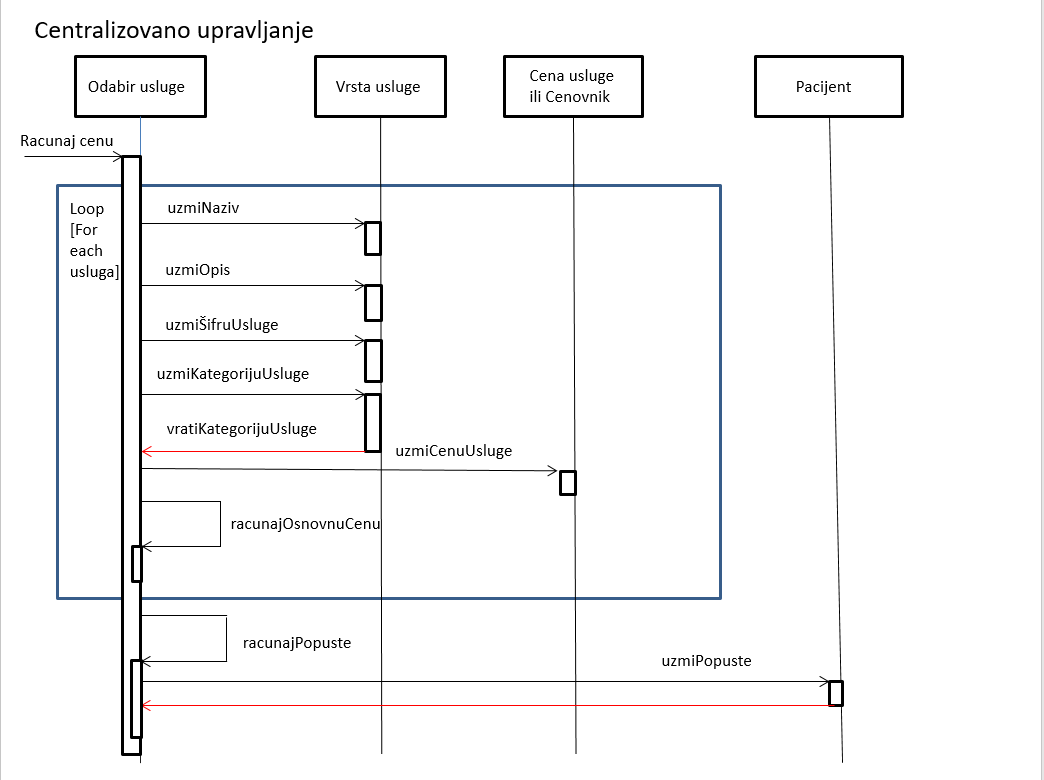
Takođe , ovaj ER dijagram će biti iskorišćen za modelovanje baze podataka za ovu veb aplikaciju.

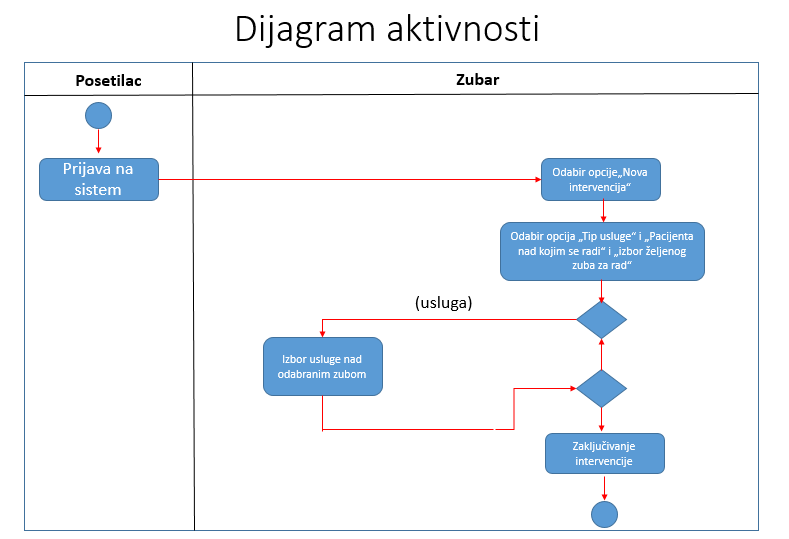
Na ER dijagramu nisu prikazani atributi konkretnih entiteta radi preglednosti , ali su zato prikazani na slici pre dijagrama.

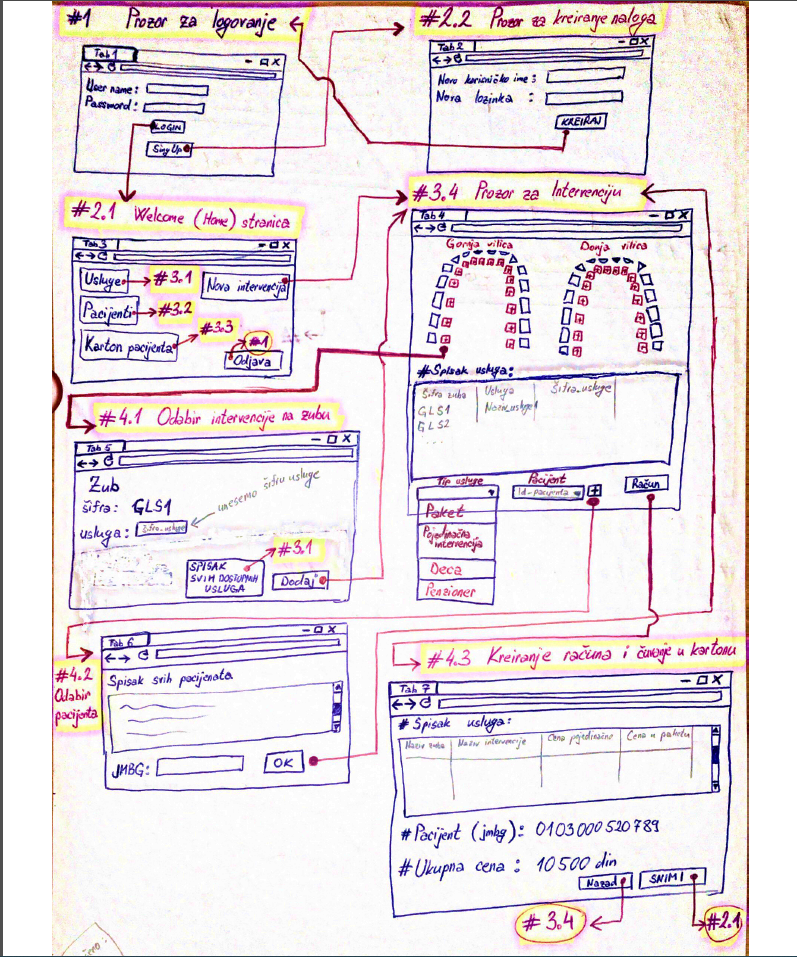
Na šemi koja prikazuje entitete i njihove atribute nisu prikazani atributi koji će biti kreirani za potrebe primarnog ključa pri implementaciji baze podataka , ali su zato prikazani atributi koji će biti iskorišćeni za potrebe spoljašnjeg ključa pri implementaciji radi lakšeg uočavanja veza u daljem radu (ovo smo mogli takođe da izostavimo pa u konkretnom dizajnu baze podataka da prikažemo)

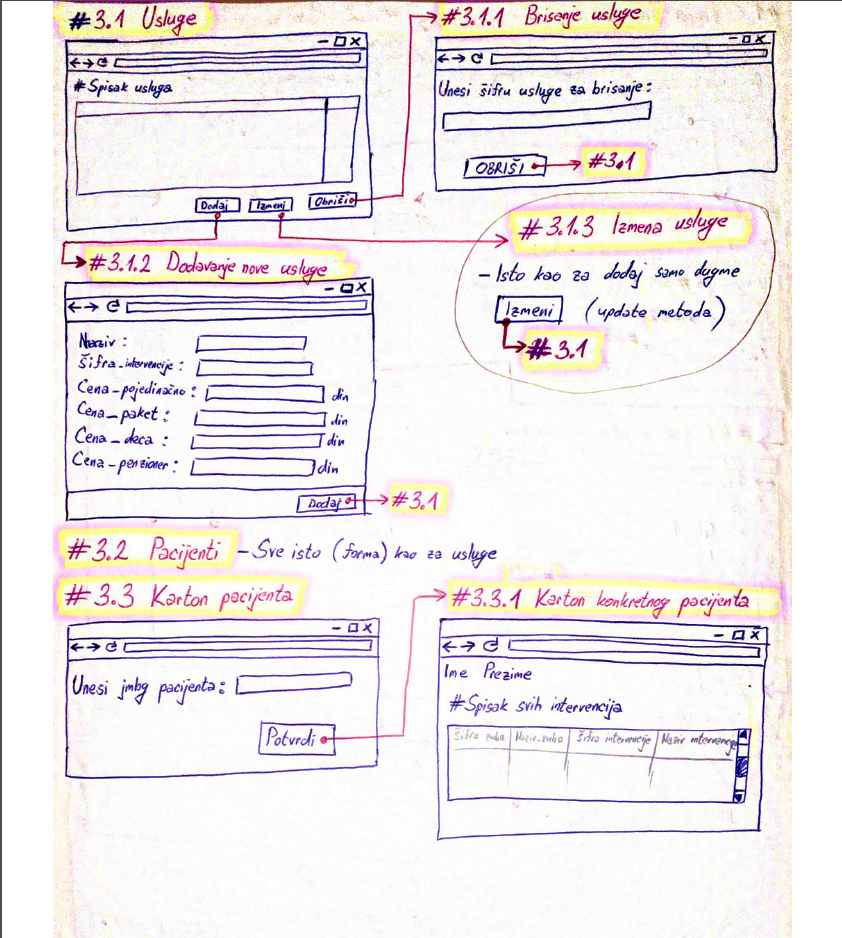
U nastavku dokumentacije biće prikazan model ponašanja veb aplikacije koja ujedno može predstavljati i skicu predloga za korisnički interfejs.

Na skici je detaljno prikazan način korišćenja i tok veb aplikacije sa slikama svih mogućih prozora i prikaza u toku korišćenja aplikacije.









# Projektovanje sistema

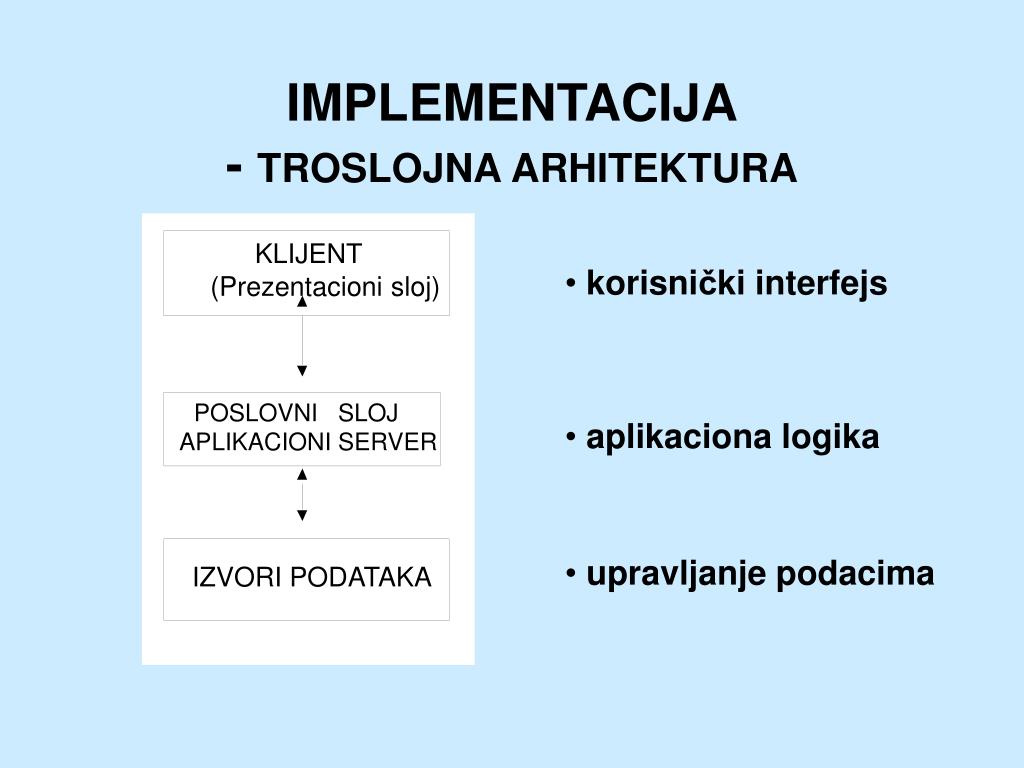
U ovom projektu će se koristiti **troslojna klijent-server** **arhitektura** koja podrazumeva **sloj baze podataka** gde se skladište podaci, **sloj poslovne logike** koji će biti smešten na serverskoj strani koji će služiti za procesiranje zahteva upućenih od strane klijenta i zaštite podataka važnih za korsinika i

**sloj prezentacije** koji predstavlja korisnički interfejs koji će se nalaziti na klijentskoj strani i upućivaće zahteve u vidu zahteva preko http protokola serverskoj strani.

Stil arhitekture označava način povezivanja , uz eventualna ograničenja. Veza između klijenta i servera se ostvaruje tako što klijent postavlja pitanja, a server odgovara na njih. Izabrani stil koji će biti implementiran u ovoj veb aplikaciji je **klijent-server (client-server) arhitektura.**

U klijent-server arhitekturi, serverska komponenta pruža usluge većem broju klijentskih komponenata. Klijentske komponente zahtevaju usluge od servera. Serveri su stalno aktivni i osluškuju da li ima zahteva od klijenata. Zahtevi se šalju komunikacionim kanalima koji zavise od mašina na kojima se server i klijent nalaze (najčešće putem interneta). Prispeli zahtevi se obično opslužuju u odvojenim nitima na serveru. U komunikaciji često ima „praznog hoda“ koji nastaje kako zbog saobraćaja na mreži, tako i zbog neophodnog transformisanja zahteva i rezultata u formate koji su često različiti na klijentskoj i serverskoj strani.

**Opšti model troslojne arhitekture** je prikazan na slici ispod :



Tehnologija koja će biti korišćena na sloju baze podataka je **RDBMS (Relational Database Management System) MySQL – MySQL Workbench** i **MySQL server** koji služe za kreiranje i upravljanje realcionom bazom podataka. Ovaj izbor je napravljen zato što je MySQL stabilan i ima odličnu integraciju sa drugim tehnologijama na ostalim slojevima arhitekture i zato što je opensource, odnosno ima besplatnu licencu za korišćenje.

Tehnologija koja je izabrana za poslovnu logiku je **Spring framework** koji je baziran na **Java programskom jeziku** koji je dobro usklađen za realizaciju CRUD (Create Read Update Delete) funkcionalnosti nad bazom podataka i ostalih funkcionalnosti i servisa koji zavise od potreba klijentske strane. Takođe je u pitanju opensource tehnologija ,a koristi se pomoću Eclipse IDE razvojnog okruženja.

Za potrebe pravljenja korisničkog interfejsa na klijentskoj strani će biti korišćen **Angular js framework** koji se bazira na **Javascript jeziku** ili **Typescript jeziku** (koji se naknadno prevodi na Javascript). Korisnicki interfejs koji će biti realizovan ovom tehnologijom će pomoću HTTP metoda (PUT,GET,POST i DELETE) za RESTful API komunicirati sa serverskom stranom. Ova veb aplikacija ce biti izvedena u obliku SPA (Single Page Application) za čije kreiranje Angular radni okvir i služi.

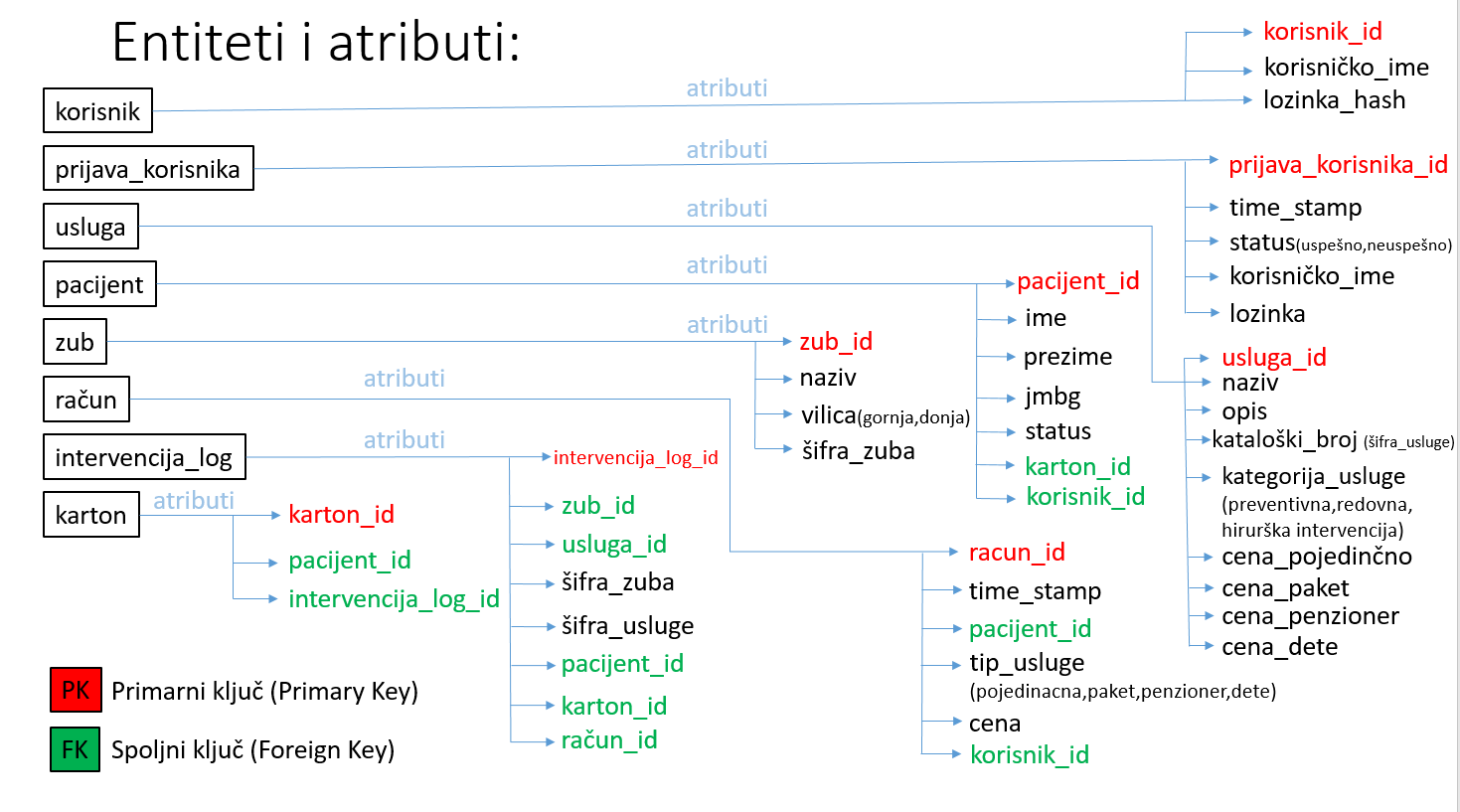
RESTful API je stil arhitekure za API (Aplication Programm Interface) koji koristi gore pomenute metode u obliku HTTP zahteva (requests) za pristup i korićenje podataka. REST (Represantional State Transfer) se češće koristi od drugih sličnih tehnologija zbog toga što koristi manju propusnost (bandwith), što ga čini pogodnijim za efikasno korišćenje Interneta.

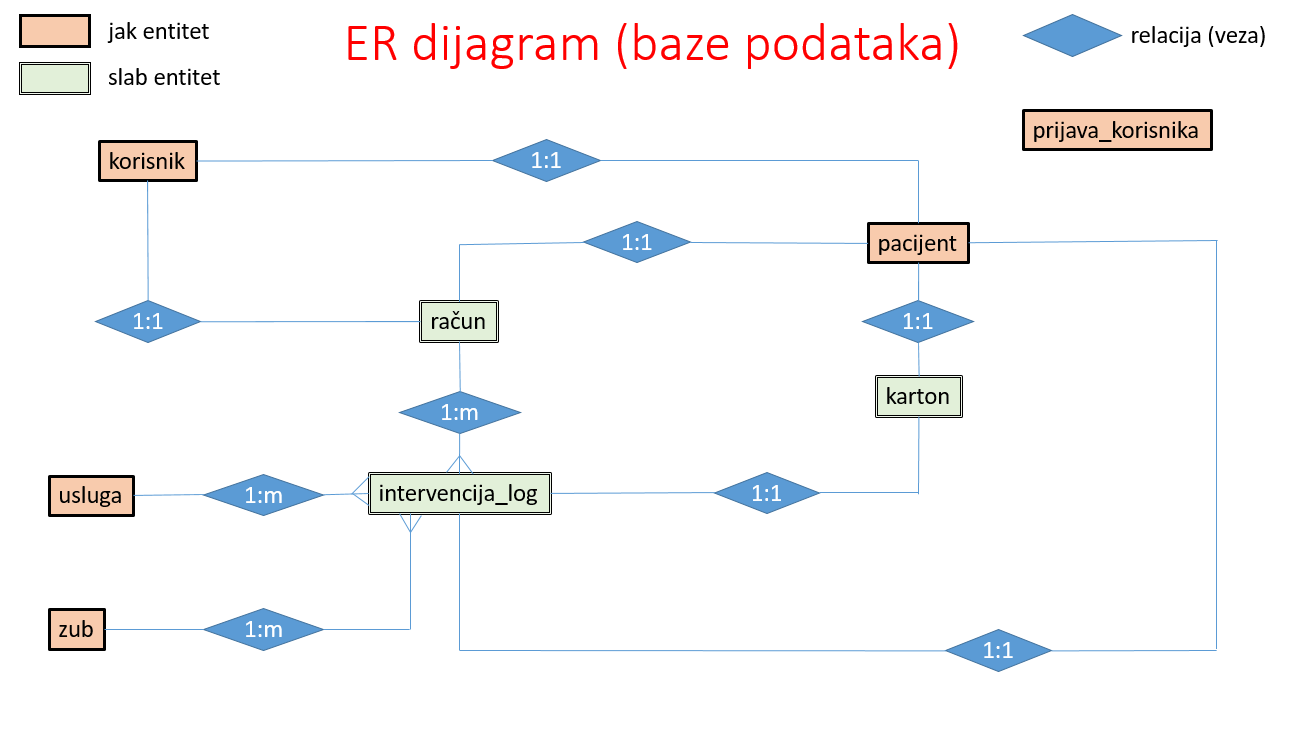
Izabrani format koji podržava REST a koji će biti kotišćen u projektu je JSON format. Treba napomenuti da REST takođe podržava i XML format koji je drugi najzastupljeniji format za prikazivanje i prenos zahtevanih resursa.

RESTful veb servisi su stateless što znači da ne pamte prethodna stanja (upućene zahteve) .

Angular framework koji se koristio za frontend se bazira na stilu arhitekture MV\* (Model VIew) i sve je izdvojeno u zasebnim komponentama koje se sastoje od .html/.css/.ts/test.ts fajlova koji remo služe za kreiranje šablona (view) i prikaz , za stilizovanje šablona,za kreiranje funkcionalnosti i modela, četvrti fajl služi za potrebe testiranja.

Na narednim slikama će biti prikazan model (dizajn) baze podataka i šema sa entitetima i njihovim atriburima :





Na narednim slikama ce biti prikazana fizicka implementacija baze podataka u MySQL Workbench-u :

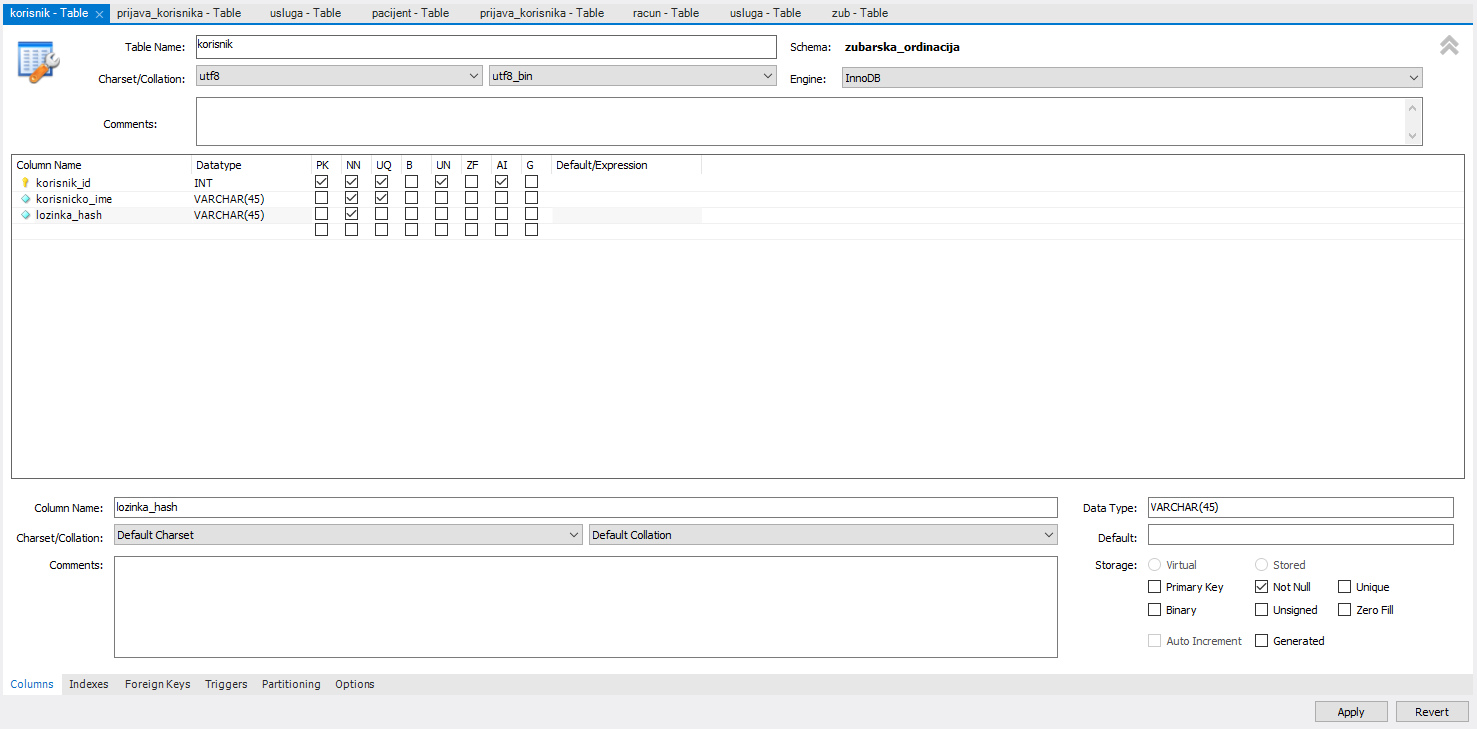
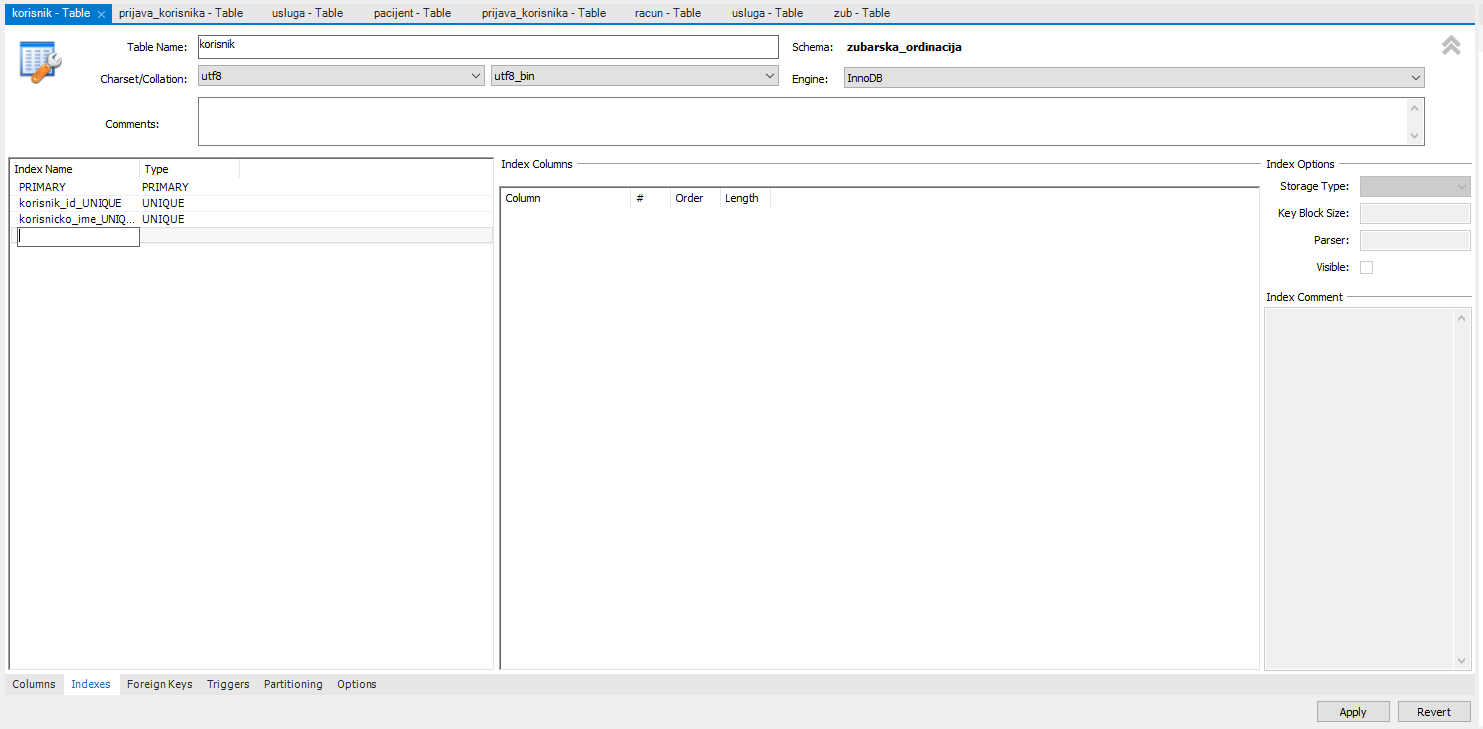


Tabela za entitet „korisnik“



Indeksi za entitet „korisnik“

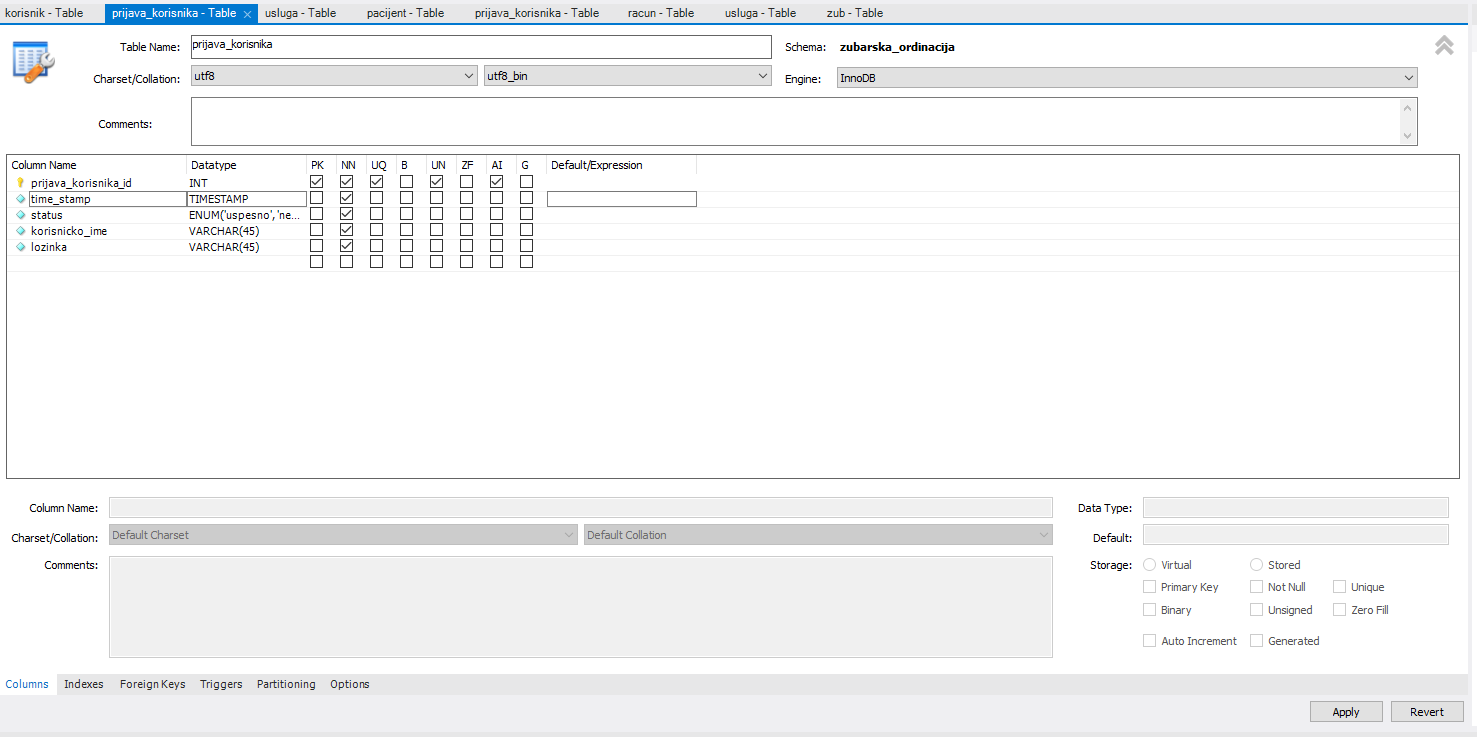
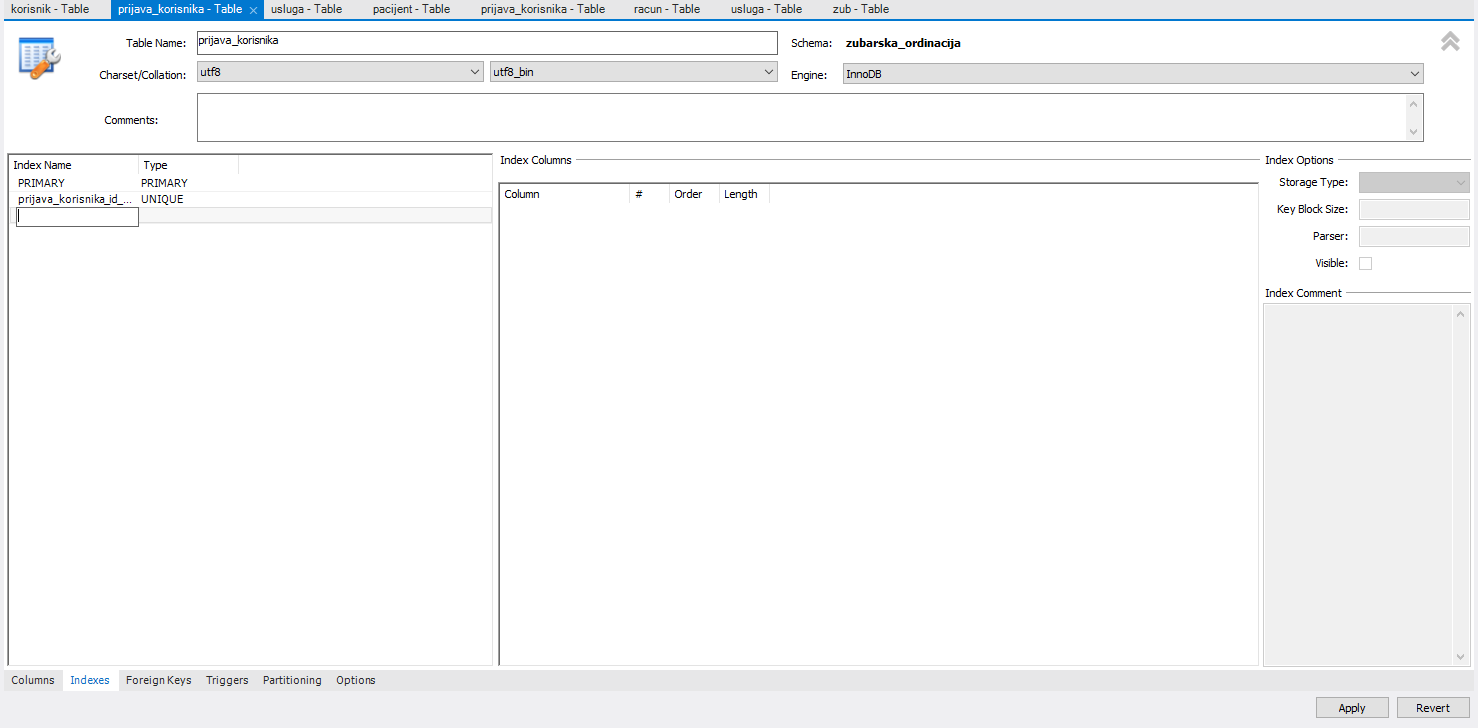


Tabela za entitet „prijava\_korisnika“



Indeksi za entitet „prijava\_korisnika“

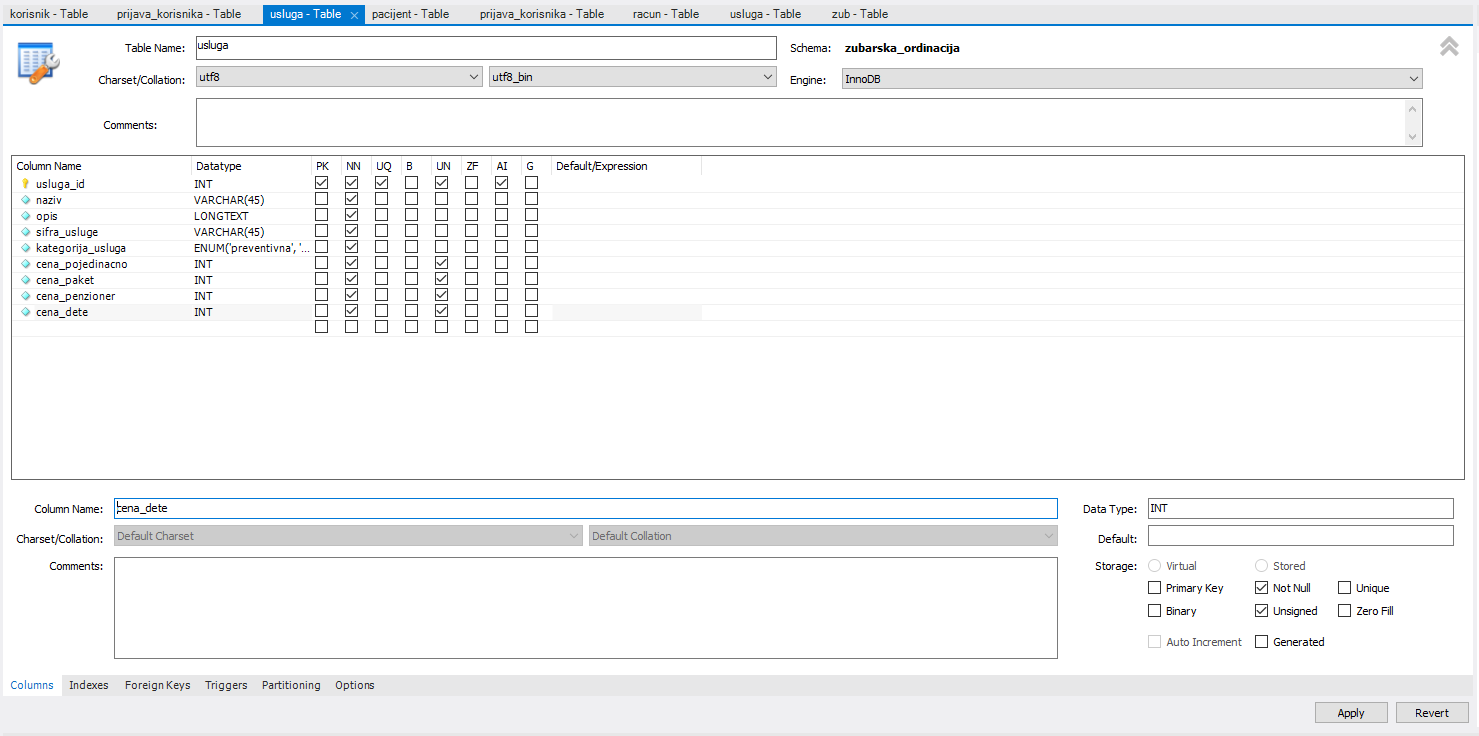
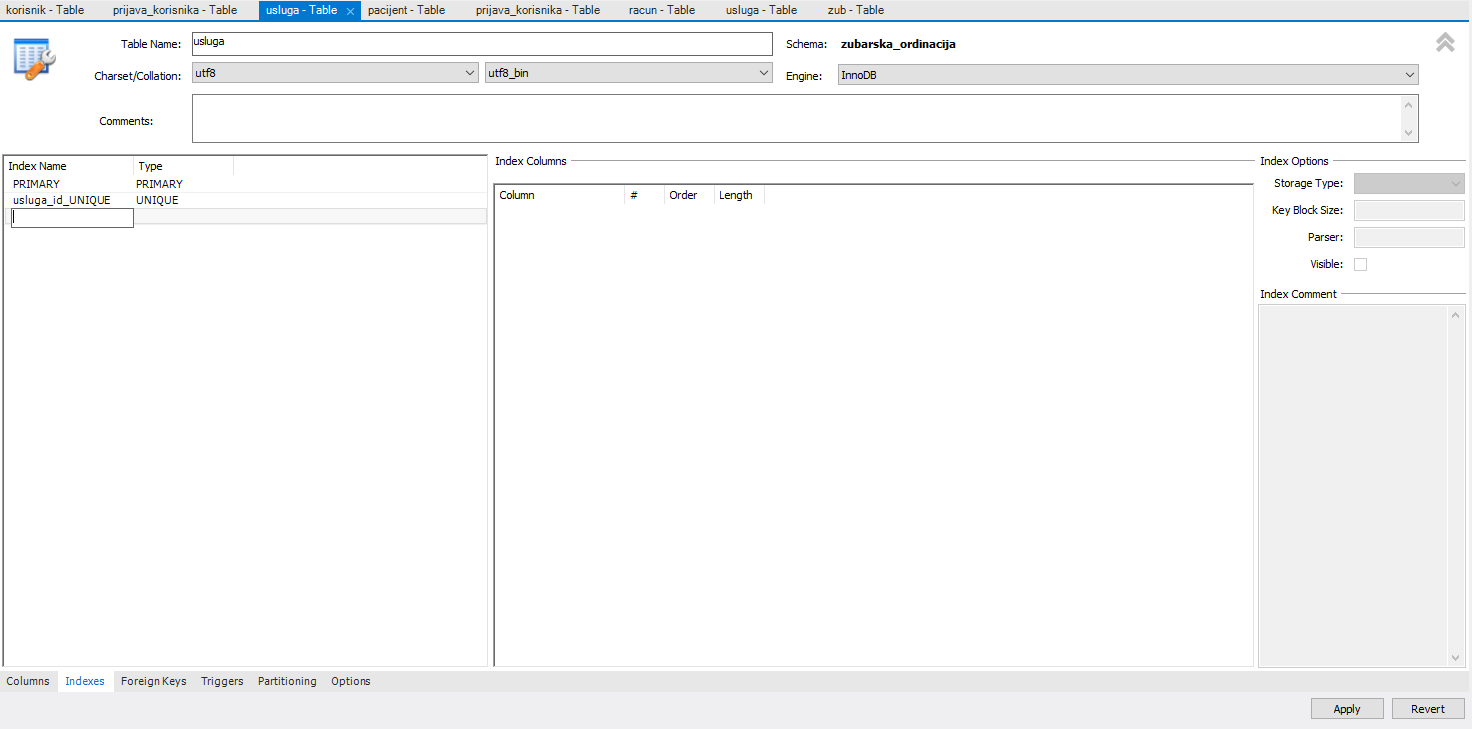


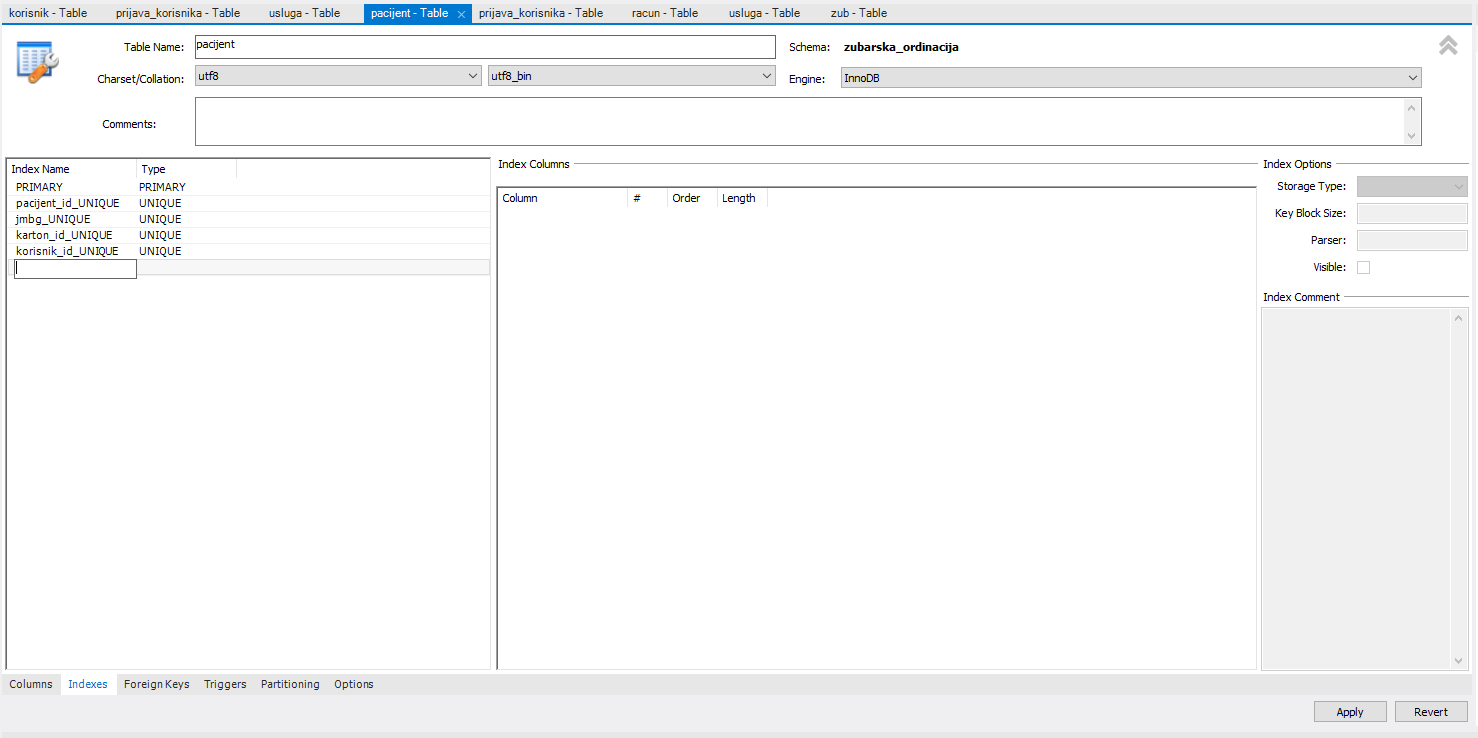
Tabela za entitet „usluga“



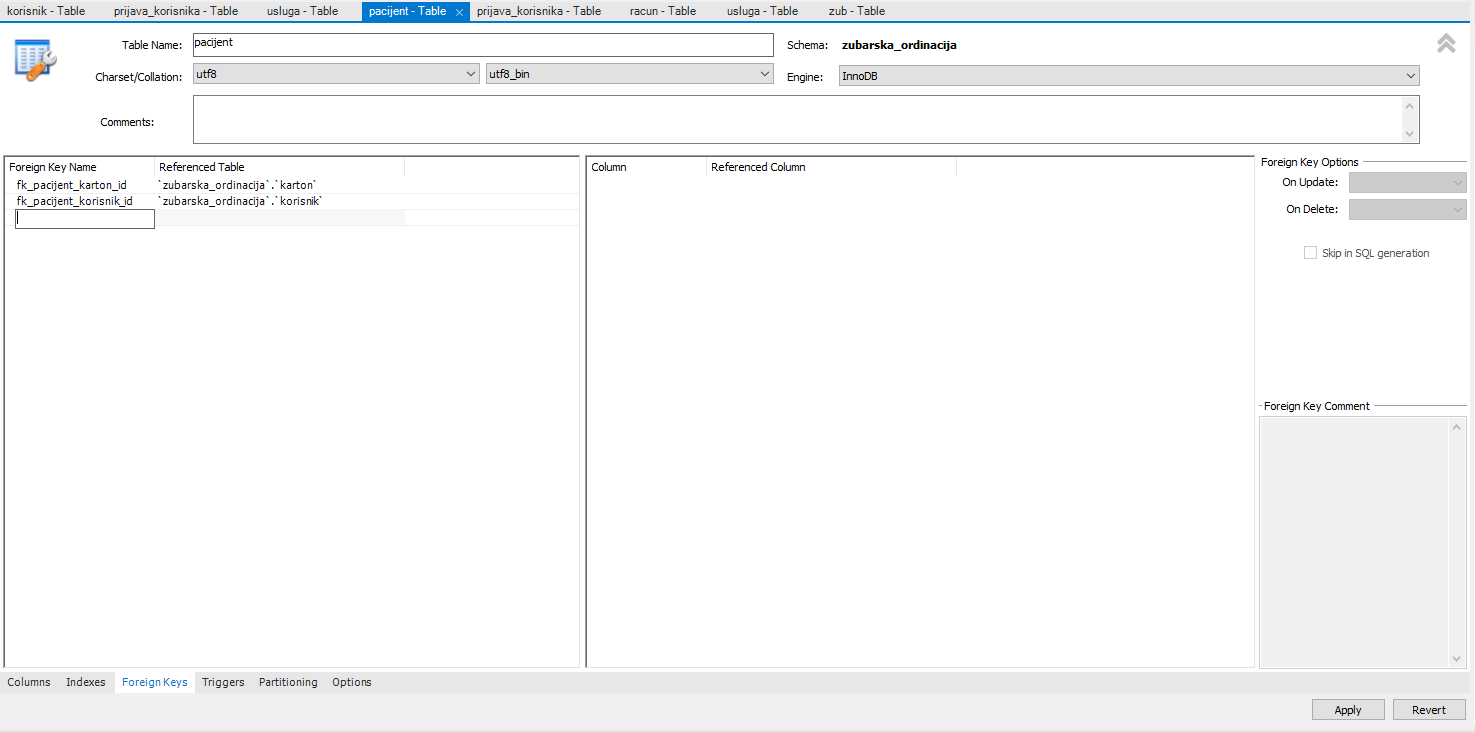
Indeksi za entitet „usluga“



Tabela za entitet „pacijent“



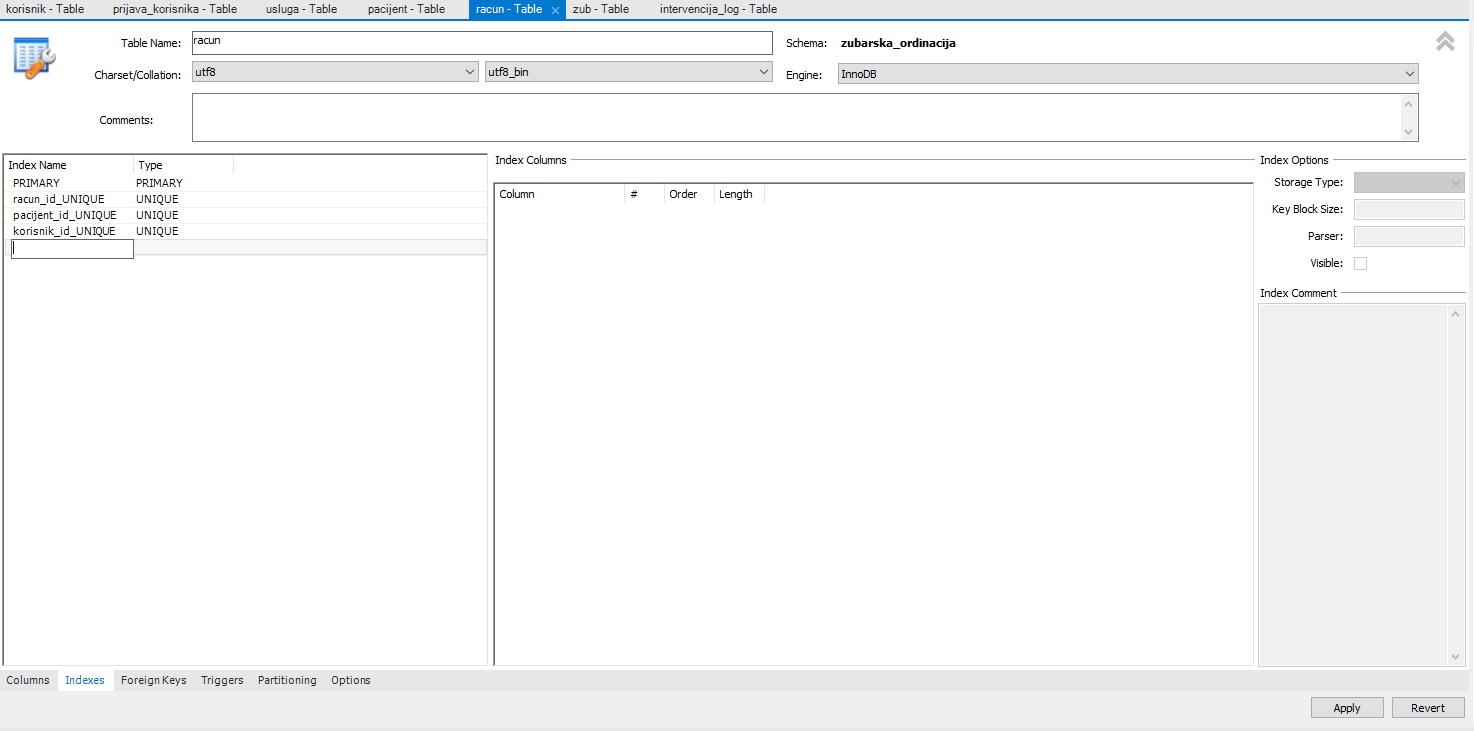
Indeksi za entitet „pacijent“



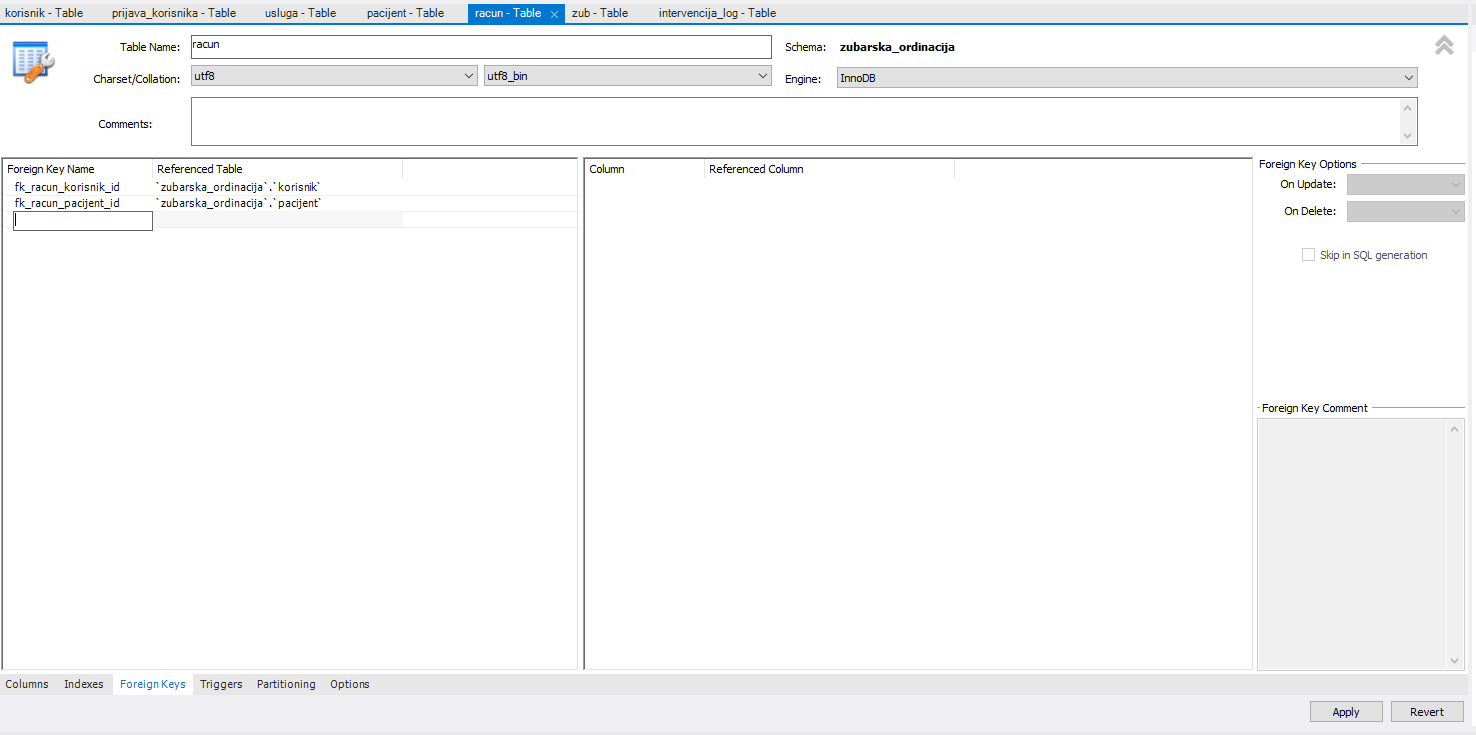
Spoljni ključevi za entitet „pacijent“



Tabela za entitet „racun“



Indeksi za entitet „racun“



Spoljni ključevi za entitet „racun“

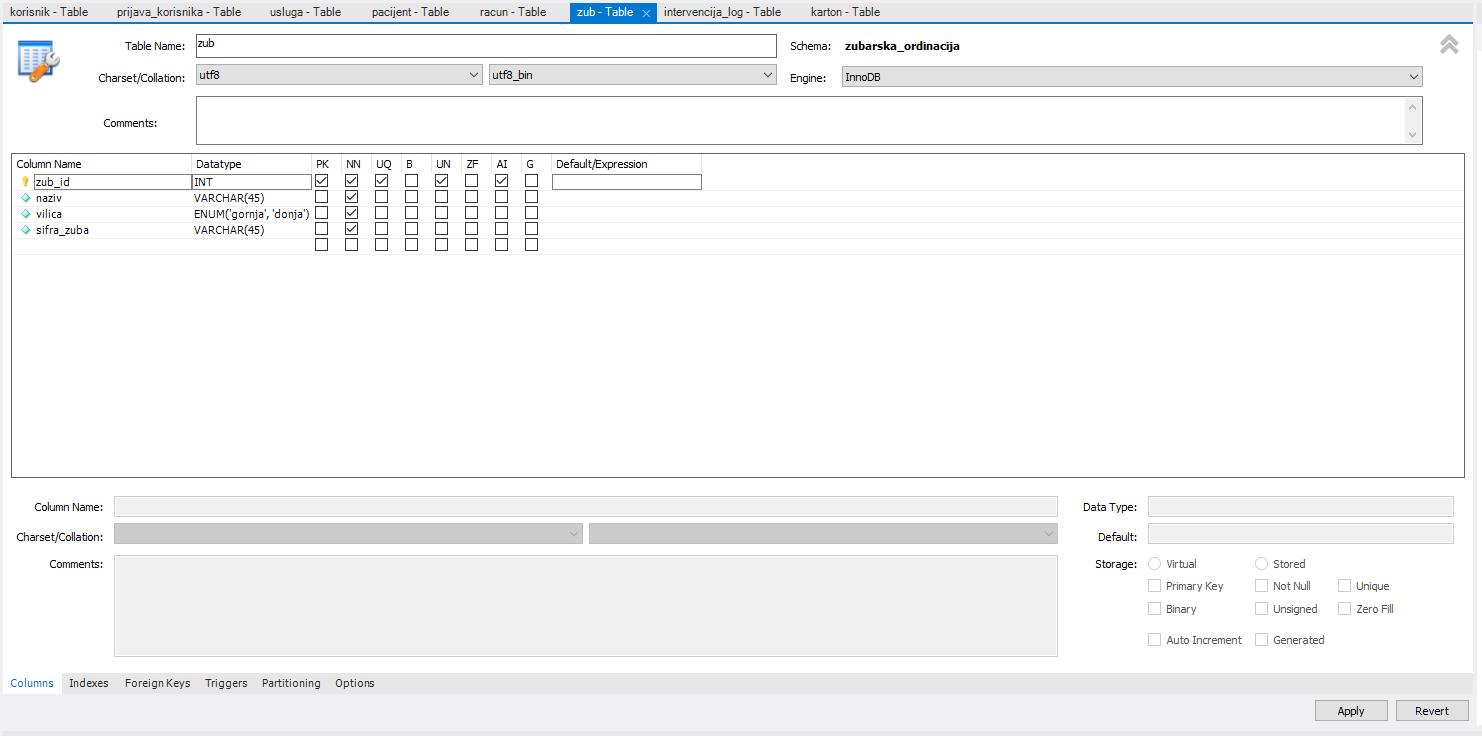
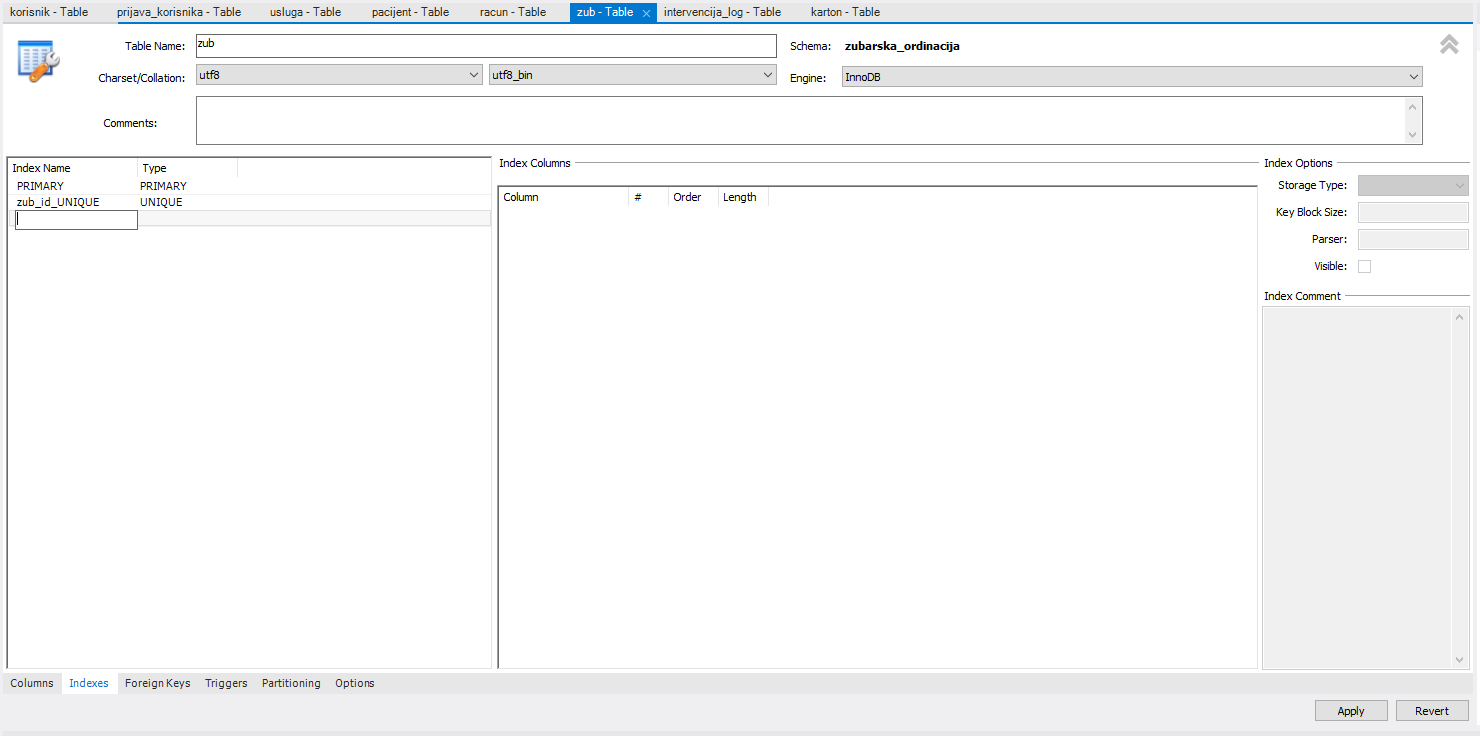


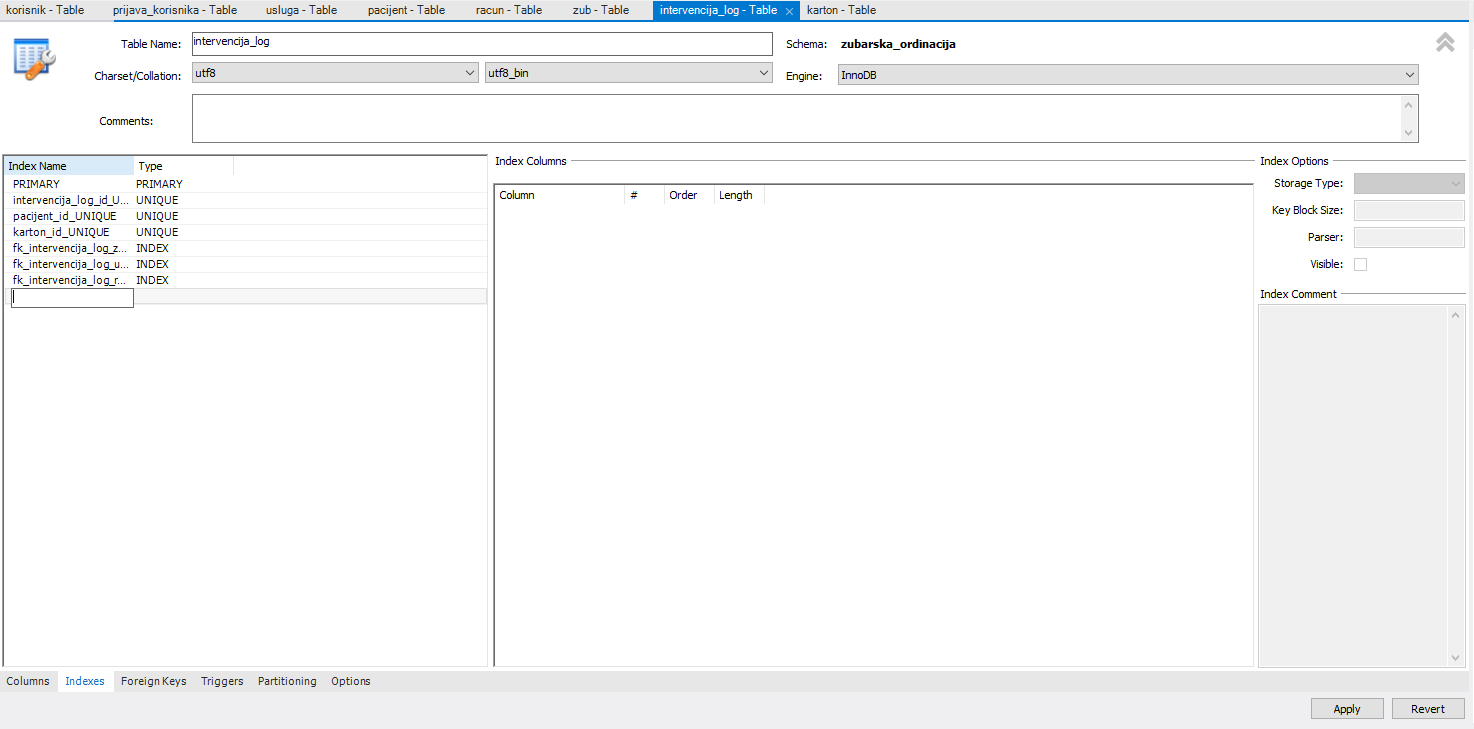
Tabela za entitet „zub“



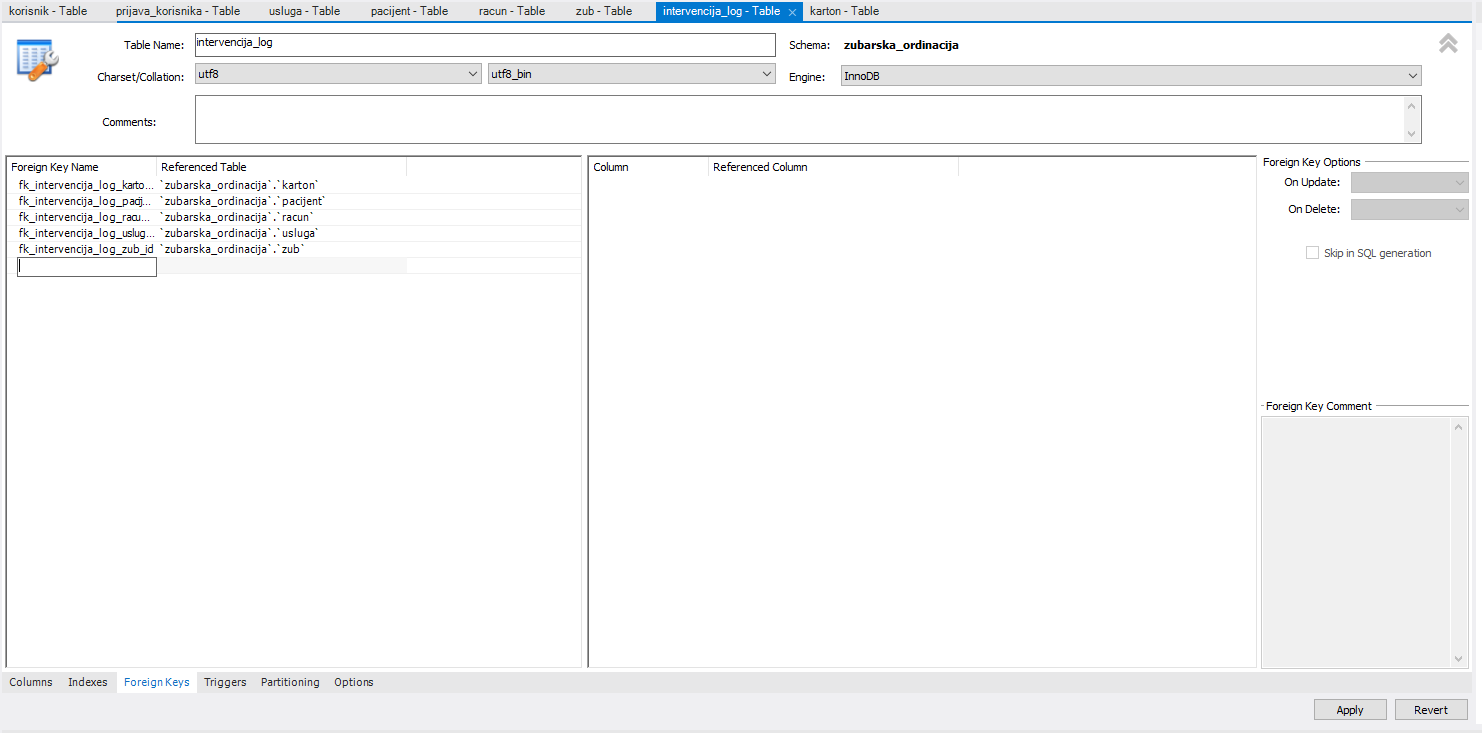
Indeksi za entitet „zub“



Tabela za entitet „intervencija-log“



Indeksi za entitet „intervencija-log“



Spoljni ključevi za entitet „intervencija-log“

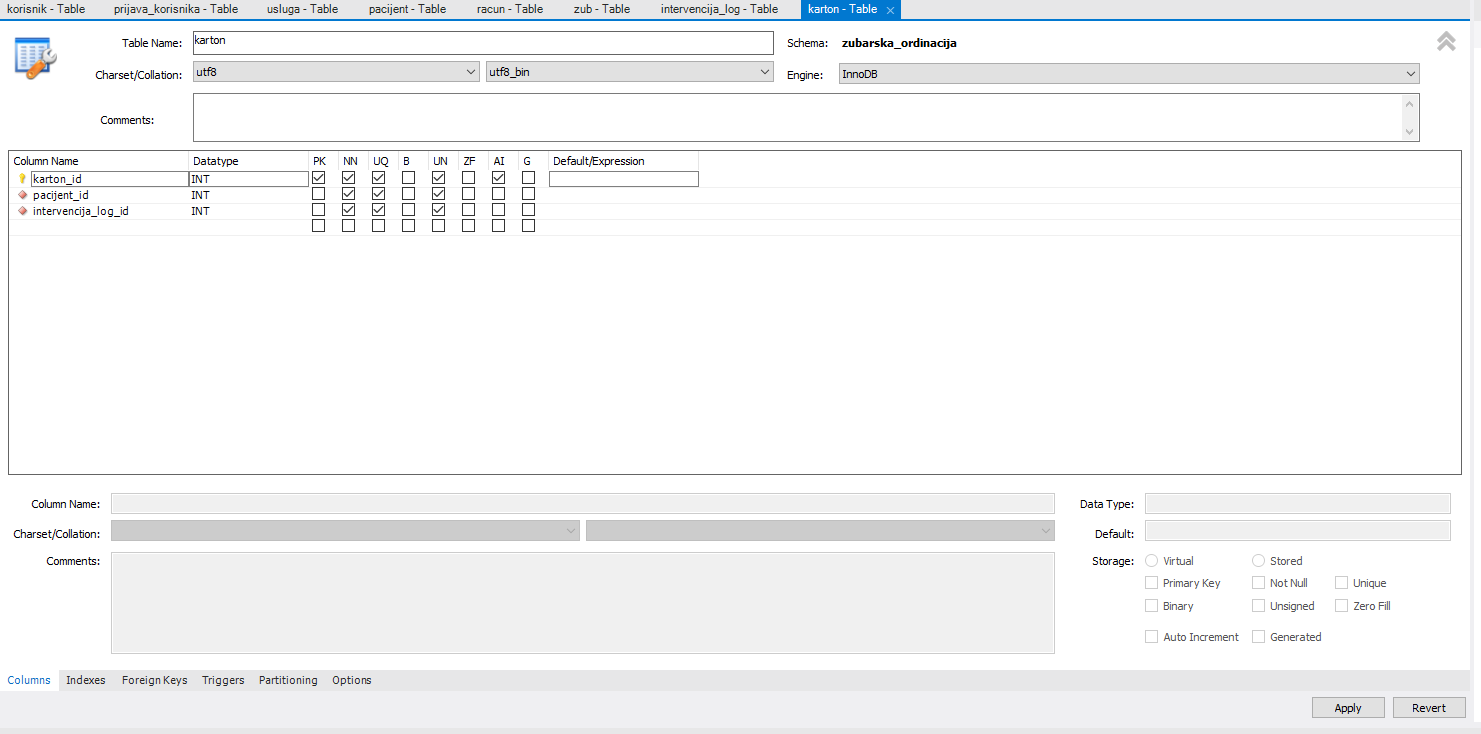
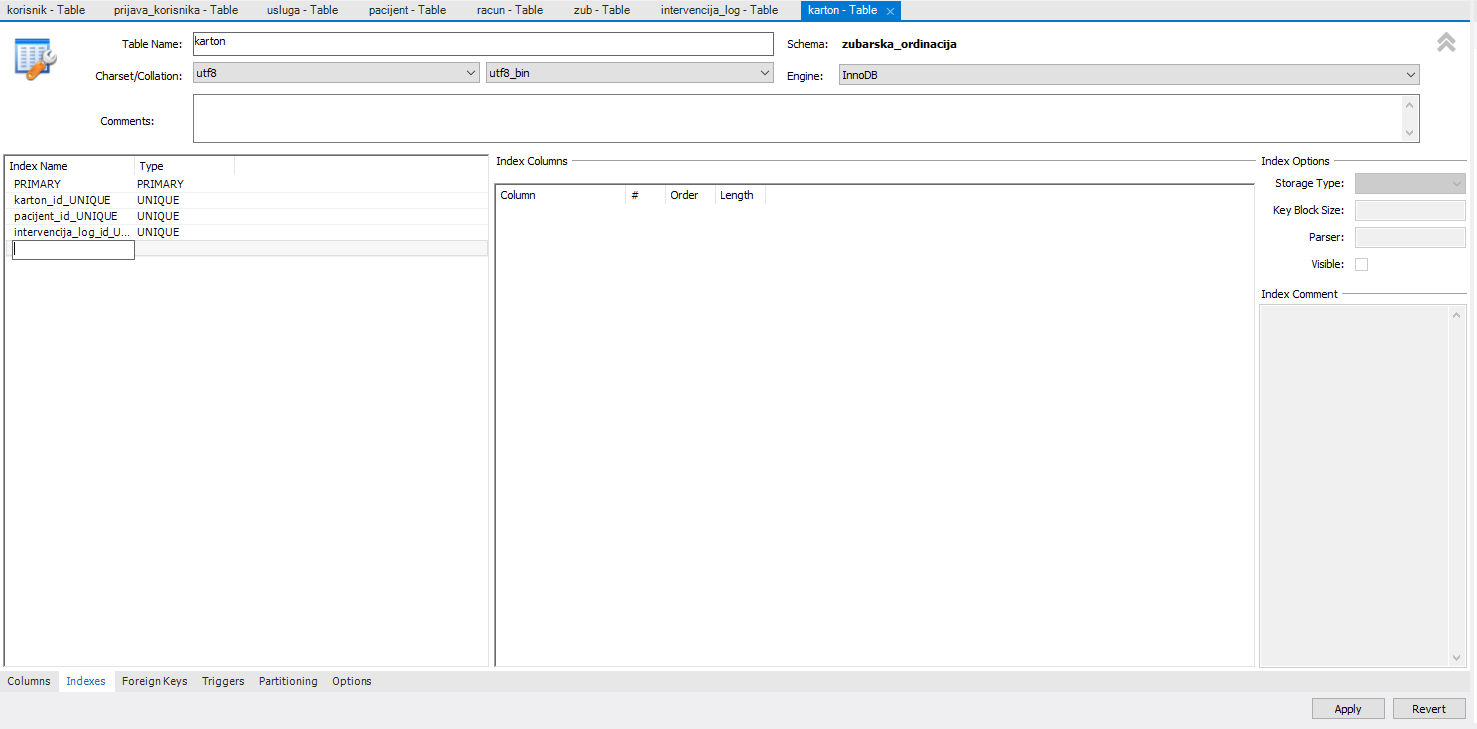
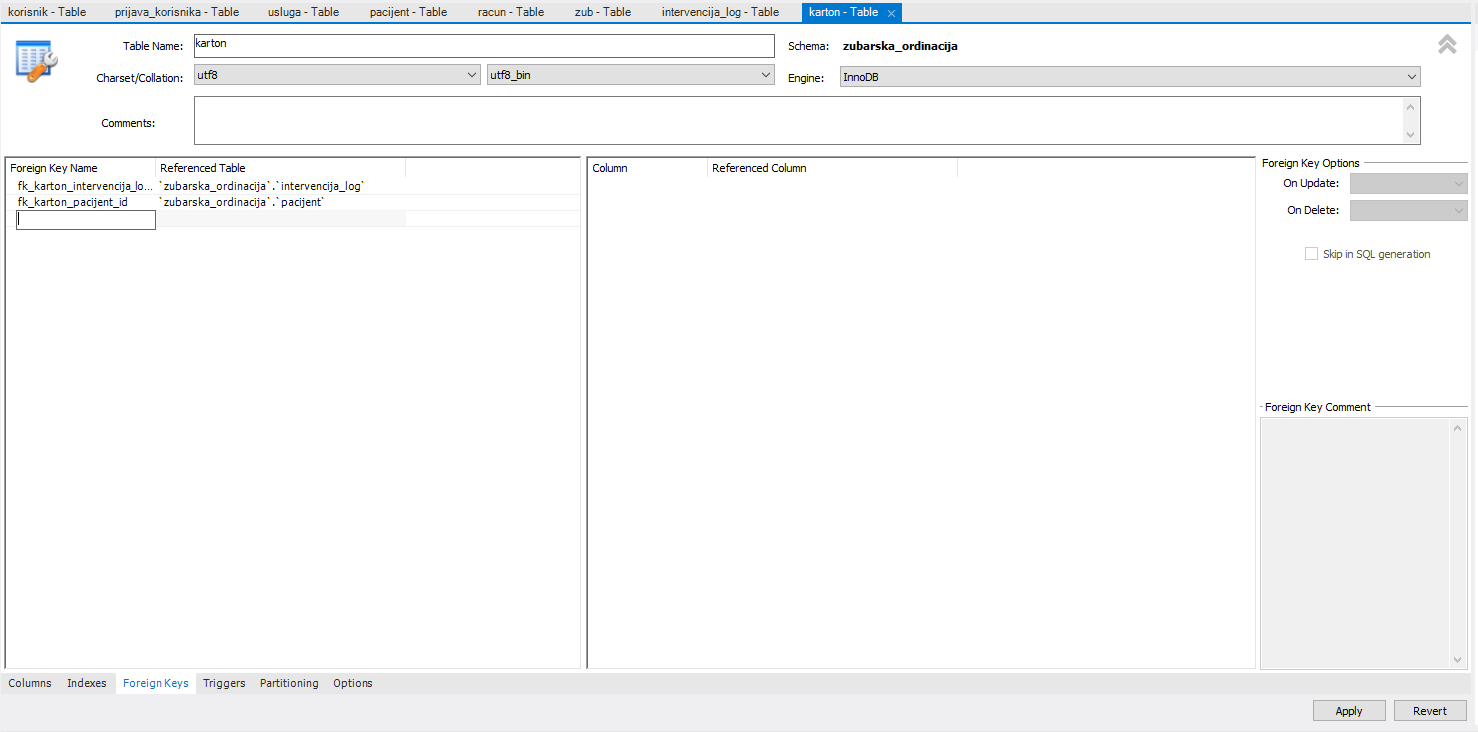


Tabela za entitet „karton“



Indeksi za entitet „karton“



Spoljni ključevi za entitet „karton“

# Implementacija softvera

**Napomena:** Zbog nedostatka vremena projekat nije do kraja implementiran , samo je sloj baze podataka urađen u potpunosti , dok frontend i backend je samo započet. Tako da za ovu stavku nije bilo materijala za prikaz kako bi prikazali dobro komentarisan kod u procesu pisanja.

# Testiranje softvera

**Testiranje sistema** je formalni proces koji izvodi tim za testiranje sa ciljem da utvrdi logičku ispravnost i svrsishodnost testiranog programa.

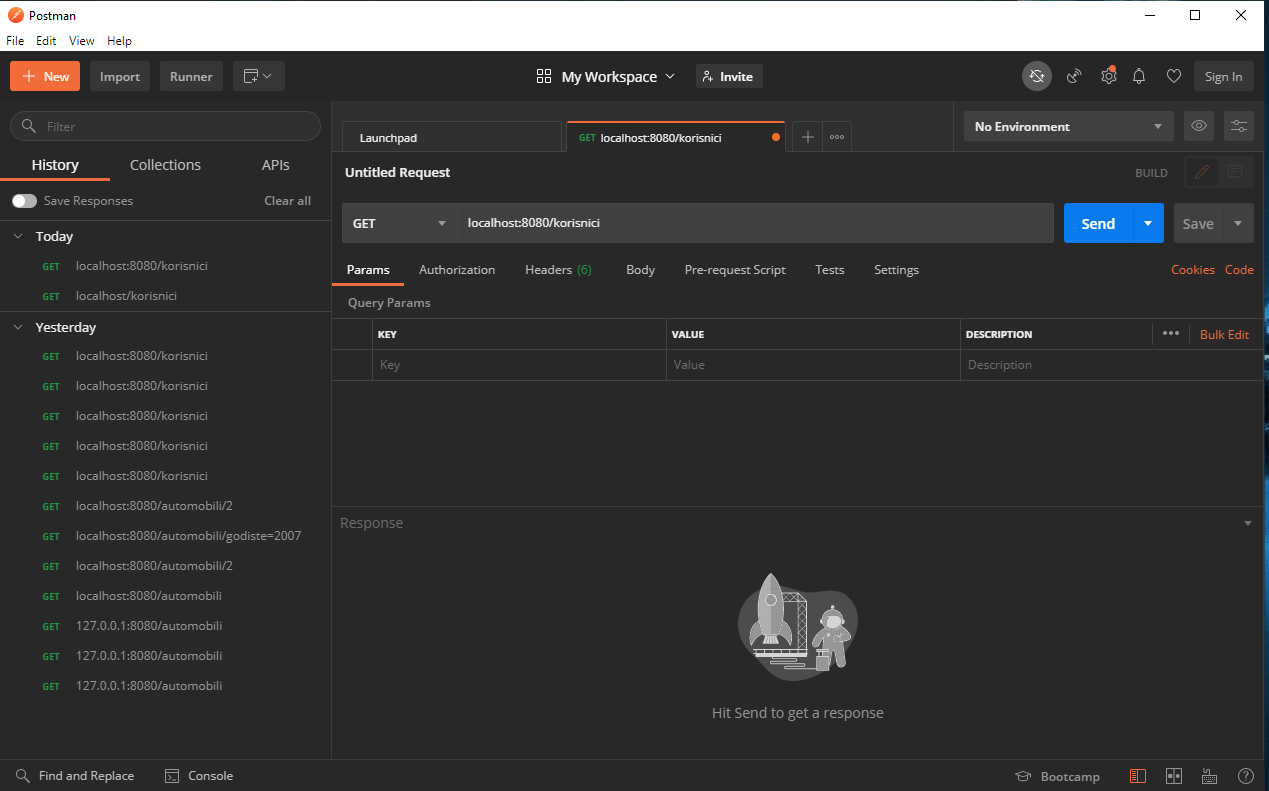
**Vrste testiranja su :**

1. **Jedinično** – je testiranje funkcionalnosti svake pojedinačne programske komponente (modula) sistema nezavisno od ostatka sistema
2. **Integraciono** – je testiranje veza tj. saradnje i komunikacije između komponenata sistema sa ciljem dobijanja kompletnog (integrisanog) sistema
3. **Sistemsko** – je testiranje da li sistem u potpunosti ispunjava sve zahteve korisnika, tako da može da bude isporučen

Za potrebe **testiranja backend dela aplikacije** možemo koristiti **Postman program** koji simulira http zahteve sa frontenda i ispusuje dostavljene rezultate sa backend-a i baye podataka za konkretan zahtev (npr. **GET metoda localhost:8080/korisnici** koji bi treba da u rezultatu ,obično u vidu JSON ili XML formata, dostavi sve postojeće korisnike iz baze podataka)

Za potrebe **testiranja frontend dela aplikacije** , Angular framework ima mogućnost korišćenja sopstvenih test klasa. To se postiže pokretanjem komande **ng test** u konzoli pravi se izdvojeno okruženje , koje radi na drugom portu nezavisno od aplikacije. Struktura testiranja je potpomognuta Karma+Jasmine okruženjima. Takođe Angular framwork omogućava i pokretanje aplikacije da radi u poyadini i u procesu kodiranja i vrši osvežavanje u realnom vremeno, pa tako možemo pratiti izmene koje su posledica dodavanjai i izmene koda. Ova mogućnost se pokreće komandom **ng serve** u komandnoj konzoli.

Na sledecoj slici je prikazan radni prozor Postman programa



# Isporuka softvera

U cilju lakšeg razumevanja uputstvo za korišćenje veb aplikacije najbolje je prikazati grafički u vidu slika mogućih prozora i dugmića i polja za unos podataka. Imajući ovo u vidu , skica dizajna veb aplikacije koja je u prethodnim koracima prikazana može se iskoristi kao korisničko uputstvo.

**Opis prijave i registrovanja korisnika (zubara) na veb aplikaciju :**

Upisati u polja za unos podataka korisničko ime i lozinku , a zatim kliknuti na dugme LOGIN (kao što je prikazano na **slici #1** ). Ako nemate nalog ,možete kreirati nalog klikom na dugme Sing Up i tada će vam se otvoriti prozor koji je prikazan na **slici** **#2.2 .** Tu treba da unesete vaše novo korisničko ime i lozinku za Vaš nalog.

**Opis korišćenja aplikacije nakog uspešnog procesa logovanja :**

Nakon prijave pojaviće se prozor početne stranice gde su ponuđene mogućnosti ( **slika #2.1** ) . Od opcija imamo dugme **Usluge** kojim upravljamo svim uslugama u zubarskoj ordinaciji (kreiranje novih,izmena postojećih, brisanje postojećih usluga i prikaz svih dostupnih usluga), dugme **Pacijenti** gde imamo iste mogućnosti kao za usluge samo vezano za pacijente , dugme **Karton pacijenta** kojim imamo mogućnost da unosom broja jmbg pacijenta pristupimo kartonu odnosno spisku svih rađenih intervencija na pacijentu, dugme **Nova intervencija** kojim započinjemo novu intervenciju dodavanjem na spisak usluga koje su izvšene na određenom zubu pacijenta i obračun cene za navedene usluge (taj spisak usluga se nakon završene intervencije beleži u karton pacijenta) , i na kraju imamo dugme **Odjava** za odjavi sa sistema.

Sve navedene mogućnosti su prikazane postupno na slikama radi lakšeg razumevanja.

Napomena: Kod **Nove intervencije** ,kada se prikaže prozor **#3.4** imamo prikazanu šemu gornje i donje vilice i pored svakog zuba po jedno dugme koje se odnosi na taj zub za dodavanje odrađenih usluga nad tim zubom (slika **#4.1**). Takođe u dnu prozora **#3.4** imamo padajući meni da odaberemo tip usluge (pojedinacna intervencija,paket,deca ,penzioneri) radi obračunavanja cene odrađenih usluga . Pored toga imamo i polje za unos pacijenta koga radimo i imamo malo dugmence pored da izlistamo spisak svih registrovanih pacijenata ako ne možemo da se setimo tačnog jmbg-a radi identifikacije pacijenta . Na kraju imamo dugme **Račun** koje nam formira račun i ukupnu cenu sa spiskom izvršenih usluga za odrađenu intervenciju.

