VELEUČILIŠTE U RIJECI
POSLOVNI ODJEL RIJEKA

KRISTIJAN MIFKA

Ticketing sustav

PROJEKTNA DOKUMENTACIJA

Rijeka, 2020.

VELEUČILIŠTE U RIJECI
POSLOVNI ODJEL RIJEKA

Ticketing sustav

PROJEKTNA DOKUMENTACIJA

Kolegij: IZGRADNJA OBJEKTNO ORIJENTIRANIH APLIKACIJA
Mentor: Vlatka Davidović, viši predavač
Student: Kristijan Mifka

Rijeka, svibanj 2020.

Sadržaj

1 Opis sustava (Ticketing sustav)

Ticketing sustav ima svrhu za bilježenje aktivnosti koje se zadaju zaposleniku-administratoru. Korisnik šalje upit, zaposlenik ga riješava, te nakon riješenog upita, šalje povratnu informaciju korisniku. Naposljetku zaposlenik zatvara riješeni upit. Naručitelj zahtjeva je BKS bank. Sustav će se isključivo koristiti unutar IT odjela, te će mu svrha biti brza interna komunikacija i riješavanje postojećih zahtjeva. Sustav se ne zamjenjuje za postojeći sustav (JIRA), iz razloga jer je to sofisticiran sustav koji je već implementiran već isključivo za razmjenu informacija između odjela. Sustav se uklapa u potrebe/poslovanje korisnika tako što će mu omogućiti bržu komunikaciju isključivo radi svoje jednostavnosti ali i potrebnih funkcionalnosti koje zahtjeva sam korisnik.

2 Specifikacija zahtjeva

Kratak opis sustava

Ticketing sustav ima svrhu za bilježenje aktivnosti koje se zadaju zaposleniku. Korisnik šalje upit, zaposlenik ga riješava, te nakon riješenog upita, šalje povratnu informaciju korisniku.Naposljetku zatvara ticket.

Akteri
- korisnik
- zaposlenik

Popis funkcionalnosti prema pojedinom akteru

Za korisnika:
-Uočava problem
-Prijavljuje grešku-šalje upit

Za zaposlenik:
- Dobiva upit
- Riješava problem / šalje odgovor korisniku

Scenariji za korisnika:

Rbr.	Naziv funkcionalnosti	Opis	Tijek događaja (koraci)
1	Uočava problem	Korisnik se susreće s novonastalim problemom	-Prijava u sustav -Susret s problemom -Sastavljanje upita
2.	Prijavljivanje greške	Korisnik prijavljuje grešku na ticketing sustav	-Nakon sastvaljenog upita odabir vrste upita -prijava greške/slanje upita

Scenariji za zaposlenika:

Rbr.	Naziv funkcionalnosti	Opis	Tijek događaja (koraci)
1.	Dobivanje upita	Dobivanje upita na ticketing sistem u obliku teksta s priloženim datotekama	1 - Prijava u sustav 2 -Primanje upita 3 -Kategoriziranje upita
2.	Riješvanje upita/ uklanjanje problema	Otklanje softwerskog ili hardwerskog problema te dokumentiranje izvršenog upita	-Otklanjanje problema -Slanje odgovora korisniku -Zatvaranje ticketa -Odjava iz sustava

Popis nefunkcionalnih zahtjeva:

Problemi na koje nailazi korisnik je dokumentiranost obavljenih zadataka koji se isključivo obavljaju unutar IT odjela -interno. Vremenski period, te vođenje podataka koliko svaki korisnik ticketing sustava obavlja ažurno (redovito) svoje zadatke.

Plan intervjua

Sustav: ITHelpdesk	
Projekt: Ticketing sustav	
Učesnik(ci):	Zaposlenik banke
Datum:	25.3.2020.
Vrijeme:	8:45
Mjesto:	Rijeka
Trajanje:	2h
Namjena:	Dobiti podatke koji će kasnije koristiti za kreiranje funkcionalnosti
Dokumenti:	Interni akti odjela, Interni podaci interne revizije

Detalji o korisniku sustava

Ime i prezime:	Saša Hodžić
Firma/odjel:	BKS bank
Uloga u sustavu:	Sistem administrator

Zadaci korisnika

Na koje načine akter koristi sustav?
Akter koristi sustav kako bi autmatizirao sustav poslovanja u banci.
Koje zadatke izvršava/za koje je odgovoran?
Izvršava sistemske zadatke, rad s serverskom infrastrukturom, izdavanjem certifikata.
Kome je odgovoran za izvršavanje zadataka?
Nadređenom – Marko Gustović
Na koji način se izvršavaju zadaci? Opis!
Zadaci se izvšavaju kroz postojeći ticketing sustav, kroz mailove, preko komunikacijskih kanala:cisco jabber itd. Ticketing sustav - interno za sve sektore prikazuje taskove (zadatke). Mailovi- (outlook), Jabber-video konferencije, chat kanal. TeamViewer, Fast master konekcija s klijentom, zaposlenikom.
Postoje li problemi i koji su u izvršavanju zadataka?
Ne postoje, zadaci se izvršavaju fluidno, ali rade bolje preglednosti riješenih zadataka potreban novi ticketing sutav koji bi se koristio za internu oprabu IT sektora kao testni primjer- mogučnost dorade i stavljanja na produkciju.

Identifikacija problema

Pitanja ponoviti više puta!

Na koje probleme korisnik nailazi tijekom posla?

Na koje probleme korisnik nailazi tijekom posla?
Nedovoljna dokumentiranost zahtjeva na koje mora odgovoriti. Kratak vremenski period za implementaciju.
Postoje li standardni načini rješavanja tih problema?
Produženje roka za implementaciju.
Postoji li bolji način za riješiti problem?
Bolja dokumentiranost zahtjeva.

Sumarizacija problema

- Grupirati probleme, opisati ih vlastitim riječima
- Pročitati to korisniku!
- Slaže li se ovo što je napisano s korisnikovim očekivanjima?
- Navesti 3-4 najbitnije stavke za implementaciju

Problemi na koje nailazi korisnik je dokumentiranost obavljenih zadataka koji se isključivo obavljaju unutar IT odjela -interno. Vremenski period, te vođenje podataka koliko svaki korisnik ticketing sustava obavlja ažurno (redovito) svoje zadatke. Prikaz grafikonski ako je potrebno radi predočenja reviziji. Najbitnije stavke za implementaciju su: Dobro preispitivanje okoline Razgovor s nadređenima Odabir adekvatnog alata za razvoj Dobra dokumentacija sustava Potrebe koje zahtjeva okolina Poštivanje rokova

Identificiranje nefunkcionalnih zahtjeva

Koju razinu edukacije ima korisnik?

Visoka stručna sprema.

Koju razinu vještina rada na računalu ima korisnik kojem je sustav namijenjen?

Visoka razina rada na računalu.

Koji drugi informatički sustavi se koriste u firmi i na kojim platformama?

Ticketing sustav - Jira

Kako se novi sustav može povezati s postojećim IT sustavima? Postoji li potreba za tim?

S postojećim ne pošto jira ima svoj razrađeni sustav, ali za internu uporabu da na tesnoj oklini koja bi služila za ravoj.

Postoje li planovi za nadogradnju postojećih sustava ili platformi?

Postoje.

Koja su očekivanja korisnika od novog sustava?

Da će se moći koristiti interno u IT odjelu na testnoj oklini koja bi se mogla kasnije razviti u produkcijsku.

Kakvu vrstu dokumentacije korisnik očekuje na kraju?

Dokumentaciju razvoja web aplikacije.

U kojoj mjeri bi sustav trebao biti dostupan?

Sustav bi bio dostupan interno samo izmedu IT odjela.

Koja su očekivanja korisnika vezana uz performanse sustava?

Napraviti jednostavniji ticketing sustav nego sadašnji, te mogučnost grafičkog prikaza.

Tko će održavati i konfigurirati sustav?

IT support banke.

Kako bi se sustav trebao instalirati i konfigurirati?

Instalirati se na server banke a konfiguracija preko apacha.

Koji su planovi za backup podataka?

Backup na dnevnoj bazi.

Koji su sigurnosni zahtjevi?

Nisu potrebni certifikati posto će se koristi interno u tesnoj oklini, stavljenje u produkciju zahtjevati će ssl.

Kako će se sustav distribuirati?

Distribucija će biti izravno na suse linux server.

Postoje li još neke specifičnosti ili zahtjevi o kojima bi trebalo voditi računa?

Zahtjevi će se dodatno dogovoriti interno.

3 Analiza sustava

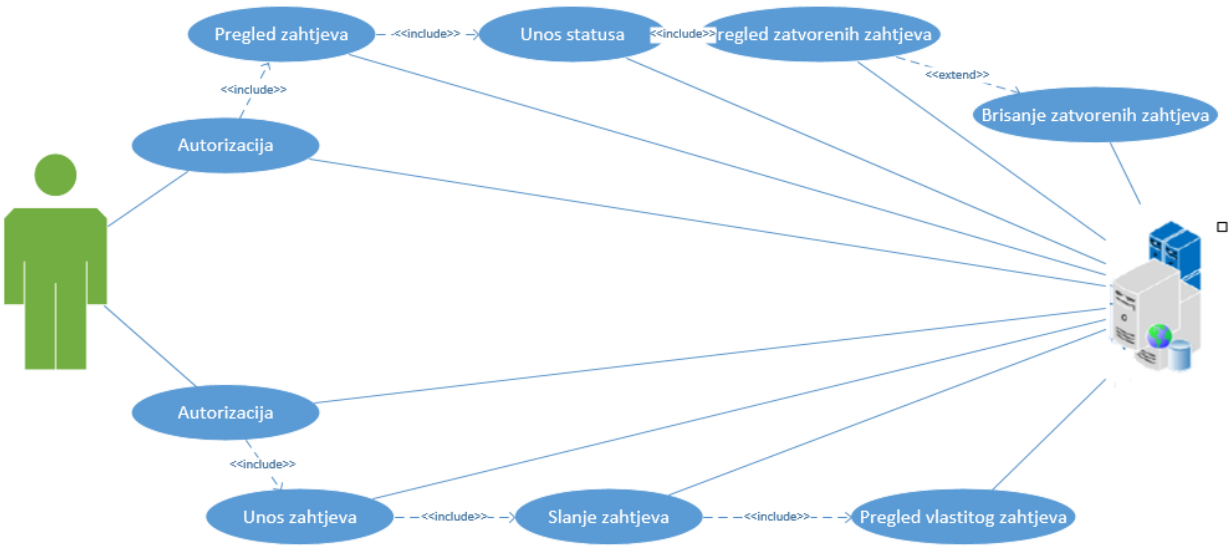
3.1 Analiza zahtjeva i korištenje sustava

Iz opisa sustava izdvojene su slijedeće funkcionalnosti koje sustav mora zadovoljiti s aspekta korisnika sustava:

Korisnik sustava (podnositelj zahtjeva):

- Unosi nove upite
- Pregledava upite
- Briše vlastite upite
- Objašnjava problem/ažurira

Slika 1: Dijagram slučajeva korištenja



3.1.2 Scenarij: Unos novog ticketa/upita

Identifikacijski sažetak: Korisnik unosi novi tiket. Za unos ticketa, evidentira se naziv, te vrijeme unesenog ticketa.

Datum zadnje izmjene: 13.7.2020.

Akteri: Korisnik

Tijek događaja:

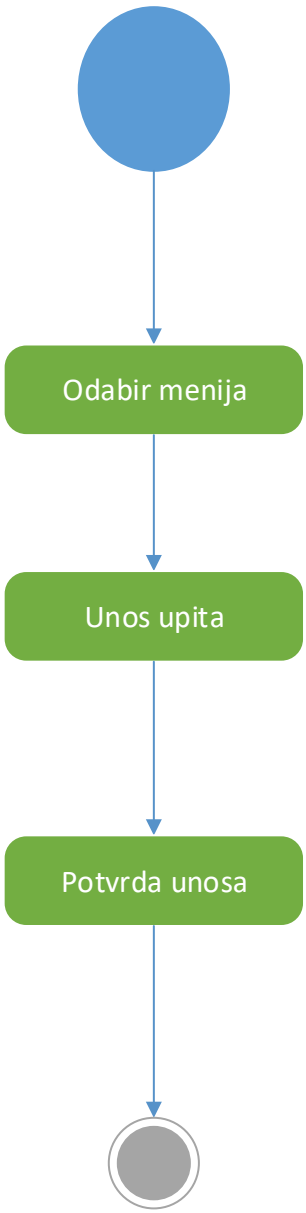
Glavni uspješni scenarij (G):

1. Korisnik odabire meni za unos ticketa
2. Unos upita
3. Odabir razine upita
4. Potvrda unosa

Alternativne sekvence:
A1: iz koraka G2 => Korisnik odabire poslani ticket te ima mogučnost ažuriranja
A2: iz koraka G2 => Korisnik ima mogučnost zatvaranja upita ako u međuvremenu ukloni problem

Sekvence s greškom:
Ulazno-izlazni zahtjevi:
Dostupna web aplikacija. Kroz tipkovnicu osigurati brzo biranje usluga.
Nefunkcionalni uvjeti (ograničenja):
Korisnik je ažurirao sve tickete.

Slika 2: Dijagram aktivnosti za unos novih ticketa



3.1.3 Scenarij: Pregled ticketa

Identifikacijski sažetak: Zaposlenik pregledava sve unesene tickete od korisnika, njihov sadržaj. Osigurava se kategorizacija upita, te se nakon toga riješava ticket. Ticketi se mogu pregledati po nazivu i datumu kreiranja, te osobi koja je zadala i riješilla zahtjev.

Datum zadnje izmjene: 13.07.2020.

Akteri: Zaposlenik

Preduvjeti: Ticket postoji u sutavu

Tijek događaja:

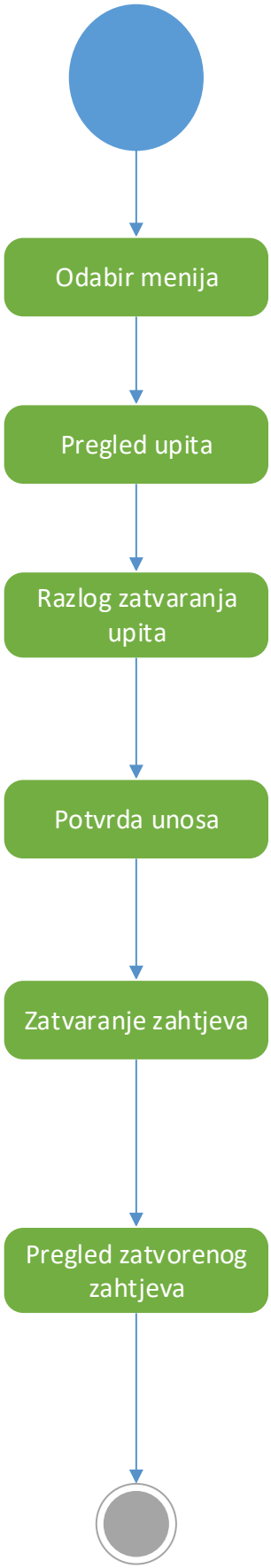
Glavni uspješni scenarij (G):

1. Zaposlenik help deska odabire iskreirani ticket od strane korisnika
2. Sustav prikazuje sve tickete u sustavu
3. Zaposlenik (administrator) otklanja poteškoću, te zatvara zahtjev.

Alternativne sekvence:
A1: iz koraka G2 => može odbrati ticket i ažuirati ga
A2: iz koraka G2 => može odabrati ticket, te ga pregledati kada je i zatvoren.

Sekvence s greškom:
Ulazno-izlazni zahtjevi:
Nefunkcionalni uvjeti(ograničenja):

Slika 3:Dijagram aktivnosti za pregled ticketa

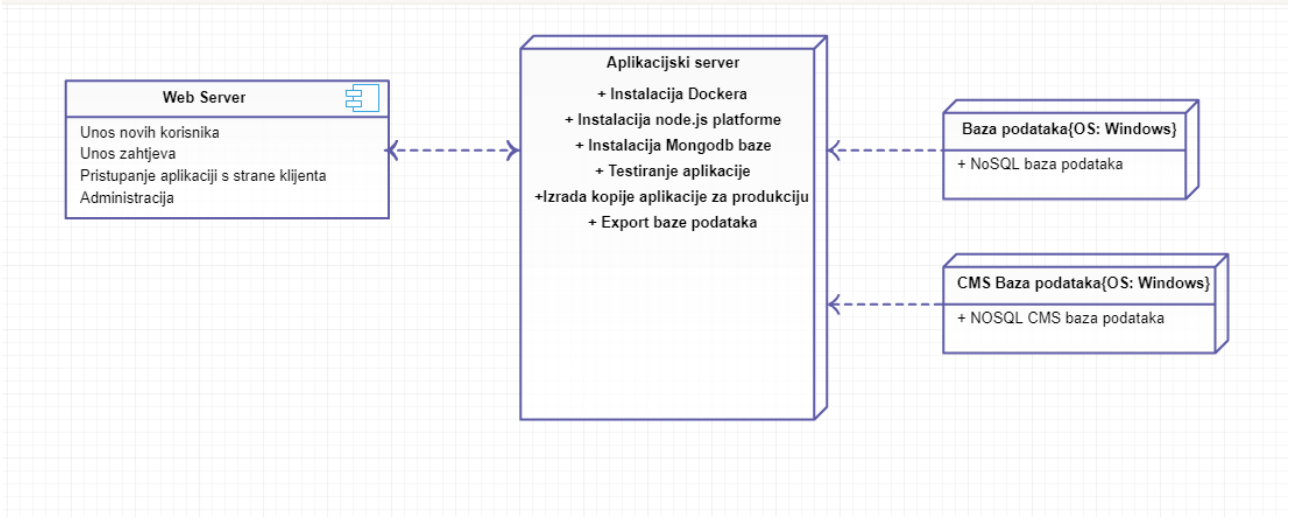


3.2 Odabir tehnologija

Programsko rješenje se izrađuje za web infrastrukturu. Podaci se pohranjuju u relacijsku bazu podataka. Korisničko front-end sučelje izrađeno je Bootstrap frameworku,
koristeći responzivni dizajn kako bi aplikacija bila prilagođena za korištenje na mobilnim uređajima i računalima. Backend za komunikaciju korisničkog frontenda i baze podataka pisan je html-u. Programsko rješenje se instalira na poslužitelja. Na istom poslužitelju instalira se baza podataka.

Korištene aplikacije za izradu:

1. Aplikacija node.js v14.3.0.
2. Aplikacija za unos baze podataka – Mongodb v3.4.10



Slika 4:Dijagam isporuke

NoSql baza podataka nalazi se na windows platformi. Također isto vrijedi za CMS bazu podataka. Kod aplikacijskog servera vrši se instalacija i testiranje aplikacije. Prvenstveno bitno je dignuti docker platformu. Također za pokretanje same aplikacije potrebno je instalirati node.js instancu. Nakon toga možemo izvršiti export baze. Sljedeći korak je testiranje aplikacije, te nakon utvrđenih funkcionalnosti izrada kopije aplikacije za produkciju. Aplikacijski server komunicira s web serverom. Na web serveru se vrši unos novih korisnika, unos zahtjeva, pristupa se aplikaciji s strane klijenta, te se vrši administracija.

4 Dizajn sustava

4.1 Dizajn korisničkih sučelja

Slika 5. Prikaz prozora korisnika

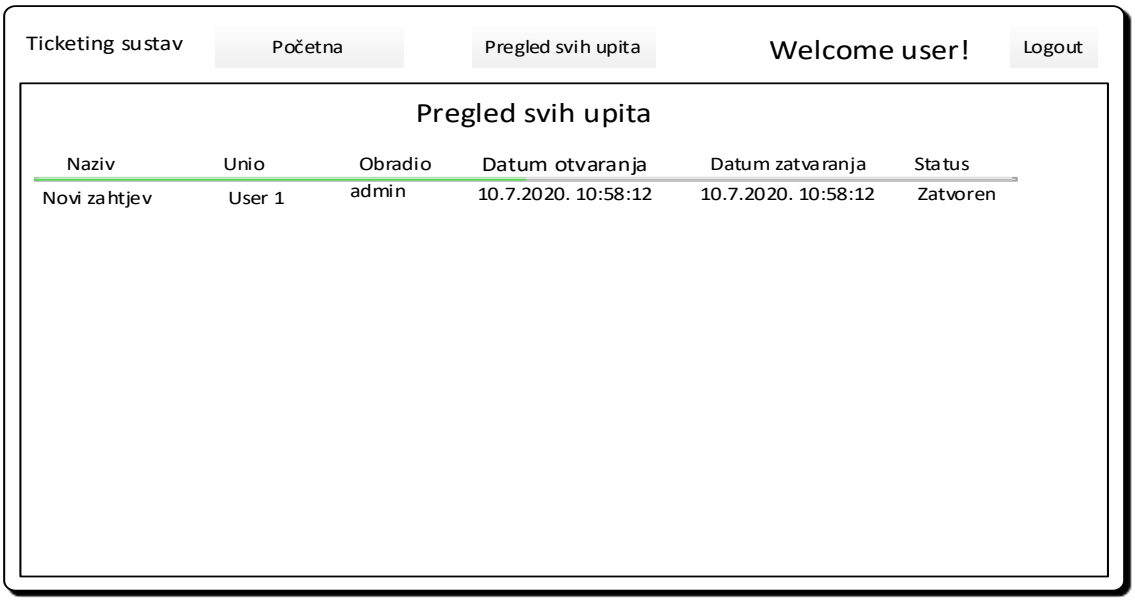
Unutar prozora od korisnika vidljive su opcije za unos novog upita. Novi upit se prikazuje na početnoj stranici gdje korisnik ima pregled svih upita koje je zadao zaposleniku- administratoru sustava. Za svaki upit vidljiv je naslov, osoba koja je zatvorila upita, te status upita. Korisnik ima mogučnost izlaza.

Slika 6. Prikaz unosa novog upita

Korisnik zadaje novi upit tako da odaber unos novog upita. Upisuje naziv – naslov upita, te opisuje zahtjev koji zadaje zaposleniku- administratoru. Nakon unesenog upita odabire kreiraj upit, te se automatski upit šalje zaposlenik- administartoru na pregled.

Slika 7. Prikaz početnog sučelja zaposlenika- administatora

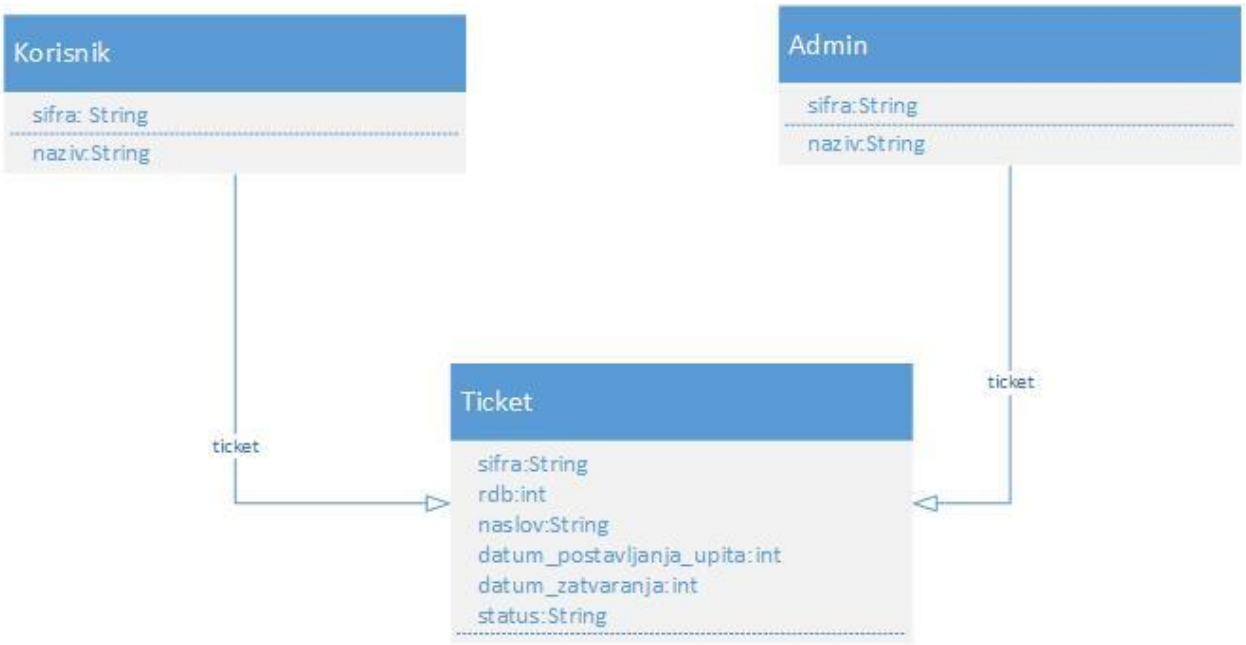
Administrator odabire zadani upit, te odgovara na njega. Nakon odgovorenog upita zatvara ticket odabirom na drop down meni. Zatvoreni ticket odlazi u pregled svih upita gdje se vrši pohrana svih izvršenih zahtjeva.



Slika 8. Prikaz pregleda svih zatvorenih upita

Unutar izbornika pregled svih upita prikazuju se svi zatvoreni upiti. Prikazano je naziv upita, tko je unio, obradio upit, datum otvaranja, zatvaranja upita, te status zahtjeva- Zatvoren.

4.2 Dijagram klasa



Slika 9. Prikaz dijagrama klasa

Dijagram klasa se sastoji od:

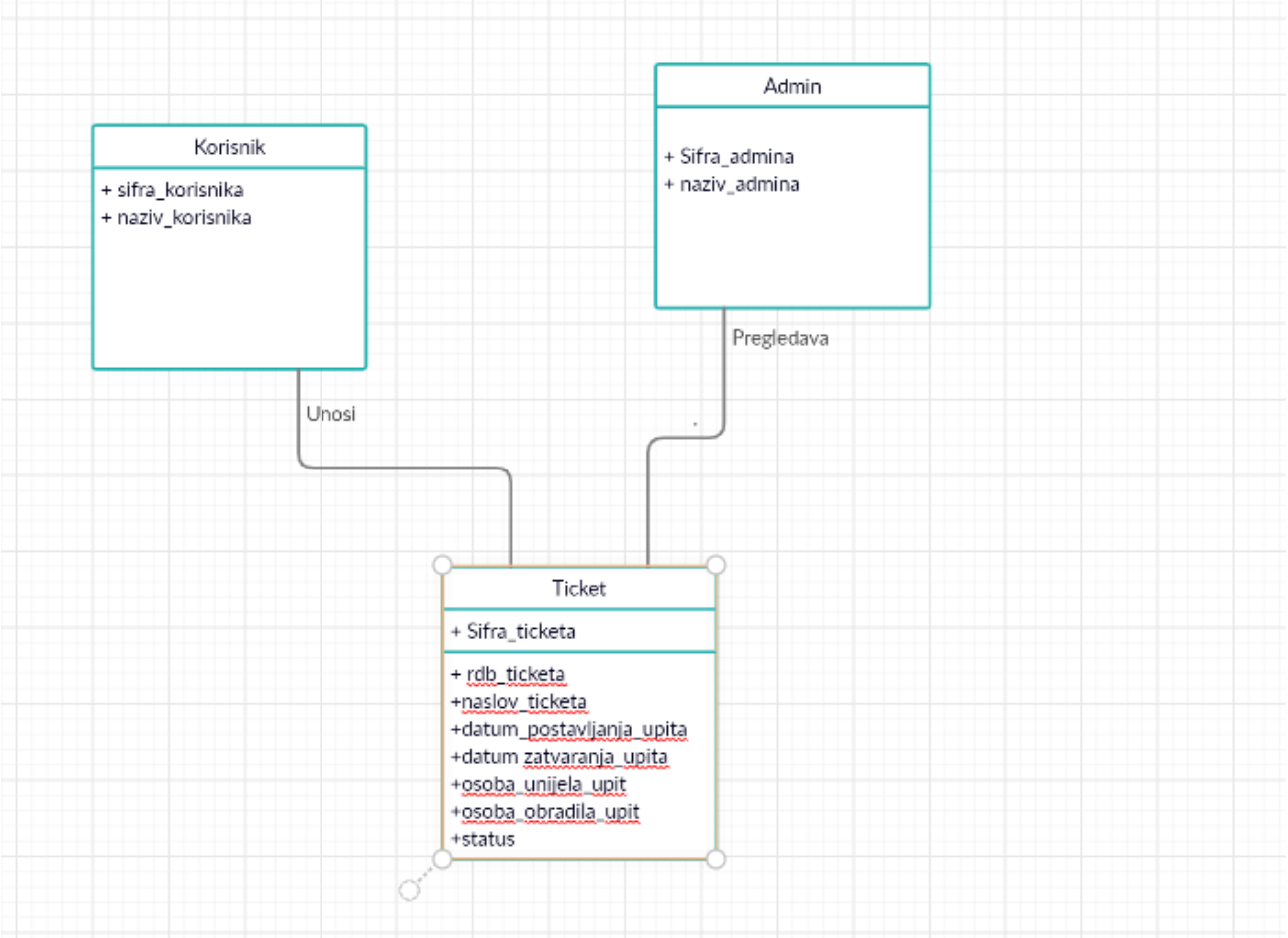
1. Korisnika sustava – osoba koja zadaje upit. Svaki korisnik ima zadano svoj naziv, te dobiva šifru korisnika.

2. Ticketa- zahtjeva koji se sastoji od šifre ticketa, rednog broja-rdb, naslova ticketa,datuma postavljanja i zatvranaj ticketa, statusa izvšenja, te opcionalno osobe koja je unijela i riješila zahtjev.

3. Administrator- osoba koja je zaposlenik u lancu- ima zadano svoj naziv, te šifru s kojom je prepoznatljiv u sustavu.

Sustav funkcionira na sljedeči način da korisnik se autorizira s svojim podacima. Nakon toga zadaje ticket koji dobiva zadane parametre, te administrator kao osoba zadužena za riješavanje zahtjeva zatvara lanac- sustav s svojim podacima- sifra, naziv administratora.

4.3 Model podataka



Slika 10. Prikaz nerelacijskog modela podataka

Korišten je nerelacijski model podataka. Baza je rađena u Nosql bazi podataka. Nerelacijski model se sastoji od: Korisnika koji zadaje ticket, te administratora koji pregledava -->odgovara na postavljeni upit.

5 Implementacija sustava

5.1 Postavljanje radnog okruženja

Za radno okruženje za izradu aplikacije izabrani su MongoDB Compass kao baza podataka te Microsoft Visual Studio Code za izradu tijela aplikacije.

Verzija alata koje su korištene za izradu aplikacije:

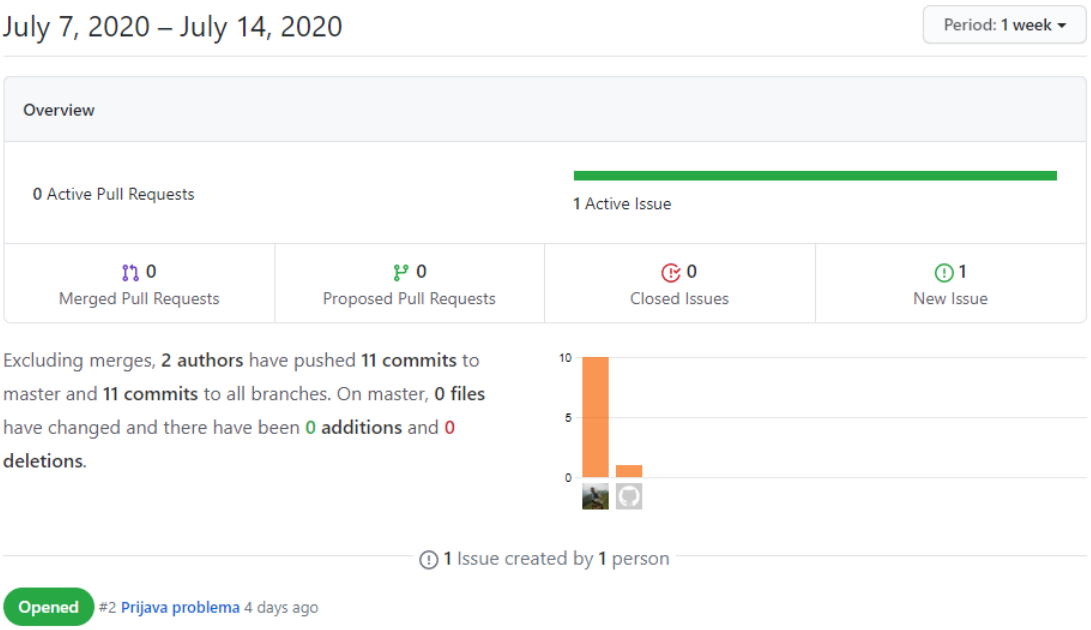
1. Node.js v14.3.0.
2. Aplikacija za unos baze podataka – Mongodb v3.4.10

5.2 Verzije aplikacije

Link prema aplikaciji na GitHubu:

https://github.com/KristijanMifka/Ticketing_sustav/tree/master/ticketing-sustav-13.7.2020

Datum prvog postavljanja: 09.07.2020.



Slika 11. Prikaz učestalosti i vremena postavljanja i nadogradnji

5.3 Prikaz dijelova programskog koda

```
const bcrypt = require("bcryptjs");
const mongoose = require("mongoose");
const { user } = require("./schema");

const User = mongoose.model("User", user);
const prompt = require("prompt");

prompt.start();

console.log("Dobrodošli u CLI za kreiranje korisnika! \n");

mongoose
  .connect(
    "mongodb://127.0.0.1:27017/ticketing-sustav",
    { useNewUrlParser: true, useUnifiedTopology: true }
  )
  .then(() => {
    console.log("DB Spojen!");
    prompt.get(["username", "password", "role"], function (err, result) {
      if (err) {
        return onErr(err);
      }

      const username = result.username;
      const password = result.password;
      const role = result.role.toLowerCase();

      const newUser = new User({
        username,
        password,
        role,
      });

      bcrypt.genSalt(10, (err, salt) => {
        bcrypt.hash(newUser.password, salt, (err, hash) => {
          console.log("Hashing password...");
          if (err) {
            console.log("error");
            throw err;
```

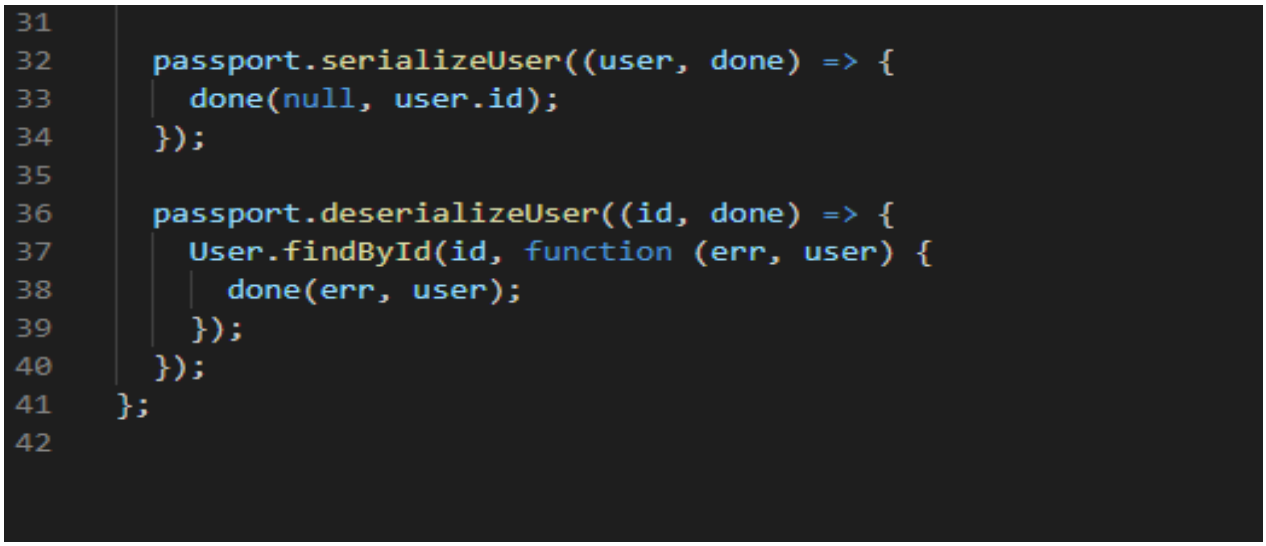
Slika 12. Prikaz cli.js klase- Linija koda 1-30

```
      bcrypt.genSalt(10, (err, salt) => {
        bcrypt.hash(newUser.password, salt, (err, hash) => {
          console.log("Hashing password...");
          if (err) {
            console.log("error");
            throw err;
          }
          newUser.password = hash;
          console.log("Password hash is:", hash);
          newUser
            .save()
            .then((user) => {
              console.log("Saving user to DB...");
              console.log("User created!", user);
              process.exit();
            })
            .catch((err) => console.error(err));
        });
      });
    });
  })
  .catch(() => console.error.bind(console, "Greška pri spajanju na DB:"));

function onErr(err) {
  console.error(err);
  return 1;
}

```

Slika 13. Prikaz cli.js klase- Linija koda 34-61

Klasa cli.js služi za kreiranje korisnika. Svaki korinik posjeduje naziv, sifru, te rolu. Prilikom kreiranja korisnika podaci se spremaju u bazu, te se lozinka u obliku hasha prikazuje u istoj. Ukoliko se ne unese jedno od polja program za kreiranje korisnika automatski izlazi, te ispisuje poruku: Greška pri spajanju na DB.

Slika 14. Prikaz passport.js klase- Linija koda 1-31



Slika 15. Prikaz passport.js klase- Linija koda 31-42

Klasa passport.js za autorizaciju postojećih korisnika- sinkronizacija s bazom. Klasa ima poziv na bazu gjde pregledava iskreirane korisnike, te vraća povratnu imformaciju u programu gitbash, da li postoji korisnik u bazi ili ne. Također prilikom unosa neispravne lozinke pojavljuje se poruka u internet preglednku da korisnik ne postoji.

```
router.post("/add", async (ctx) => {
  console.log(ctx.request.body);
  const { title, subject, resolvedReason } = ctx.request.body;
  const user = ctx.state.user;
  const data = await Ticket.create({
    title,
    subject,
    user,
  resolvedReason,
    status: "Otvoren",
    dateOfCreation: new Date(),
    closeDate: null,
  });
  console.log(`Kreiran novi ticket! \n ${data}`);
  ctx.redirect("/");
});

router.delete("/ticket/:id", async (ctx) => {
  const data = await Ticket.findByIdAndDelete(ctx.params.id);
  console.log(`Izbrisan ticket! \n ${data}`);
  ctx.body = { success: true };
  ctx.redirect("/");
});
```

Slika 16. Prikaz app.js klase- Linija koda 31-42

Unutar klase app.js prikazan je način kreiranja novog ticketa te koji su parametri potrebni za prikaz otvorenog ticketa. Također prikazano je izvođenje brisanja pojedinog zahtjeva.

```
ws > <> all.html > h2.text-center.mb-4
<h2 class="text-center mb-4">Pregled svih upita</h2>

<table class="table">
  <thead>
    <tr>
      <th scope="col">Naziv</th>
      <th scope="col">Unio</th>
      <th scope="col">Obradio</th>
      <th scope="col">Datum Otvaranja</th>
      <th scope="col">Datum Zatvaranja</th>
      <th scope="col">Status</th>
      <th scope="col"></th>
      <th scope="col"></th>
    </tr>
  </thead>
  <tbody>
  <% tickets.forEach(ticket => {%>
  <% if (ticket.status === 'Zatvoren'){%>
    <tr>
      <th><%=ticket.title%></th>
      <th><%=ticket.user.username%></th>
      <th><%=ticket.reviewer.username%></th>
      <th><%= moment(ticket.dateOfCreation).format('DD.MM.YYYY h:mm:ss')%></th>
      <th><%=moment(ticket.closeDate).format('DD.MM.YYYY h:mm:ss')%></th>
      <th><%=ticket.status%></th>
      <th class="text-center">
        <a href="/ticket/<%= ticket._id %>?referrer='all'">
          <button class="btn btn-primary w-100">Pregled</button>
        </a>
      </th>
      <th class="text-center"><button onclick="deleteTicket('<%=ticket._id%>')" class="btn btn-danger w-100">Obr
    </tr>
```

Slika 17. Prikaz all.html klase- Linija koda 1-32

Unutar all.html klase nalaze se svi parametri koji se prikazuju u prikazu zatvorenih zahtjeva: Naziv, osoba koja je unijela zahtjev, osoba koja je obradila zahtjev, datum otvaranja i zatvaranja zahtjeva, te status zatvorenog ticketa.

6 Isporuka i korištenje aplikacije

6.1 Pakiranje i isporuka aplikacije

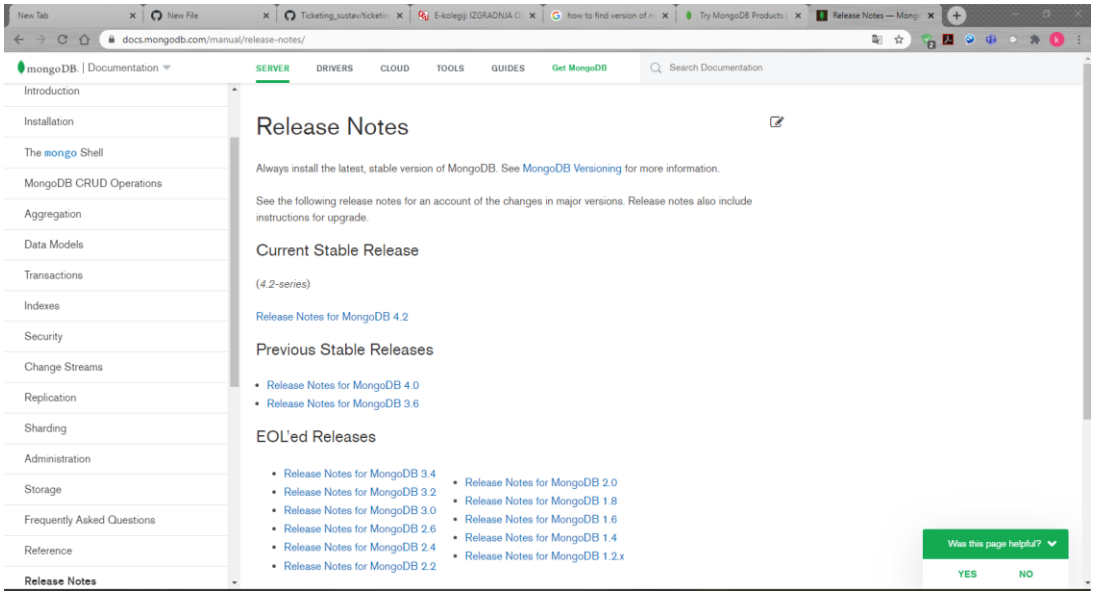
Aplikacija se će se u dogovoru s kolegama iz BKS bank AG postaviti na serversku infrastrukturu. Održavanje je ažuriranje baze podataka.

6.2 Korisničke upute za korištenje aplikacije

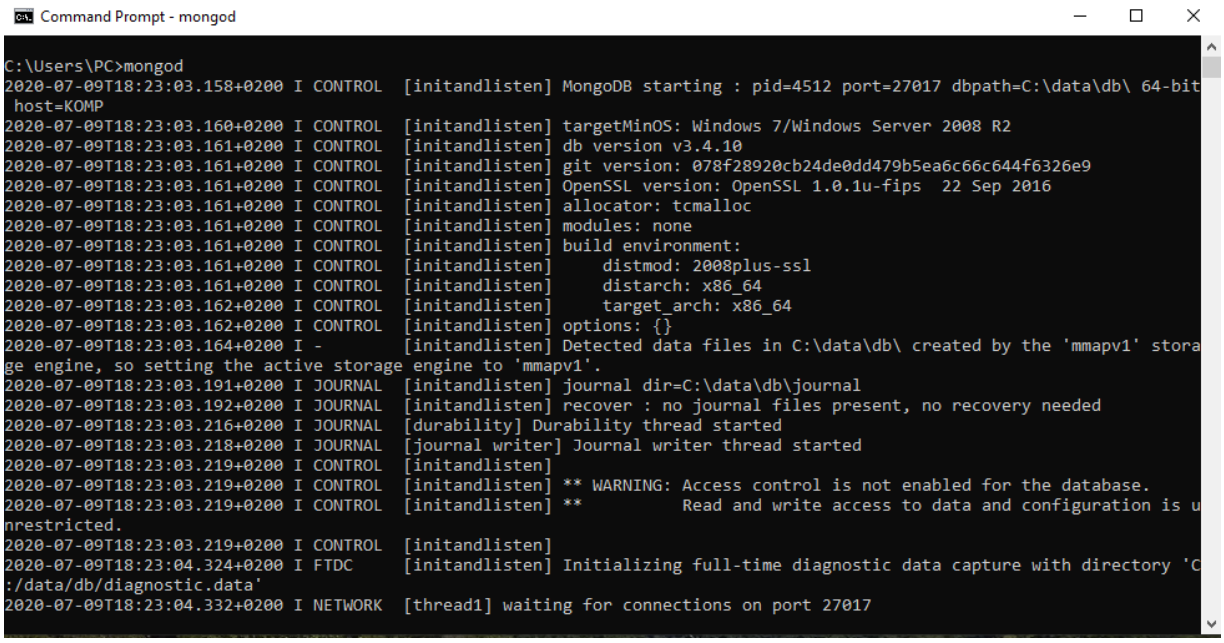
Za korištenje aplikacije potrebno instalirati:

1. Aplikacija node.js v14.3.0.
2. Aplikacija za unos baze podataka – Mongodb v3.4.10
3. Aplikacija Gitbash v 2.26.0.
4. Web preglednik: Mozila, Google Chrome, Edge.

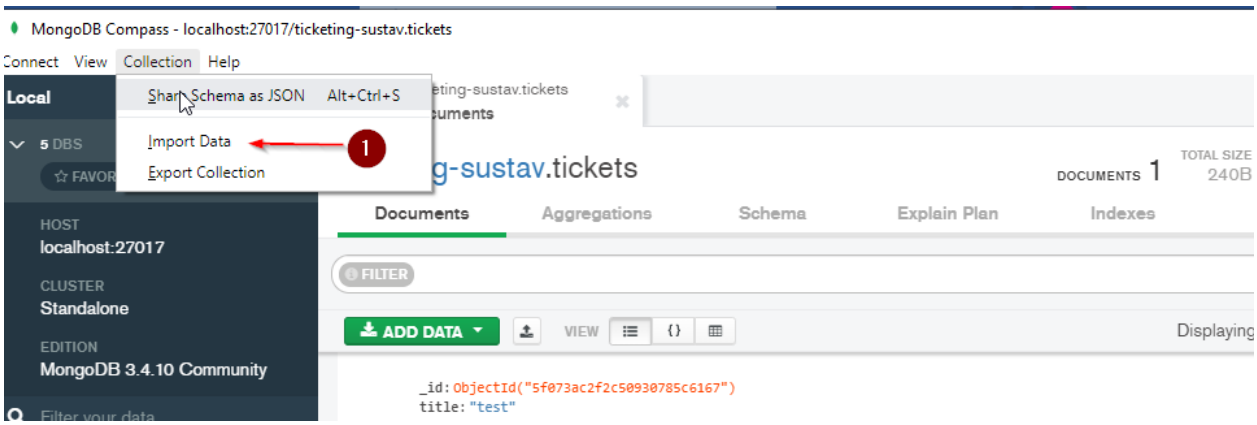
6.2.1 INSTALACIJA I IMPORT BAZE PODATAKA -MONGODB



1. Preuzeti MongoDB-→ url: https://docs.mongodb.com/manual/release-notes/ Verzija 3.4



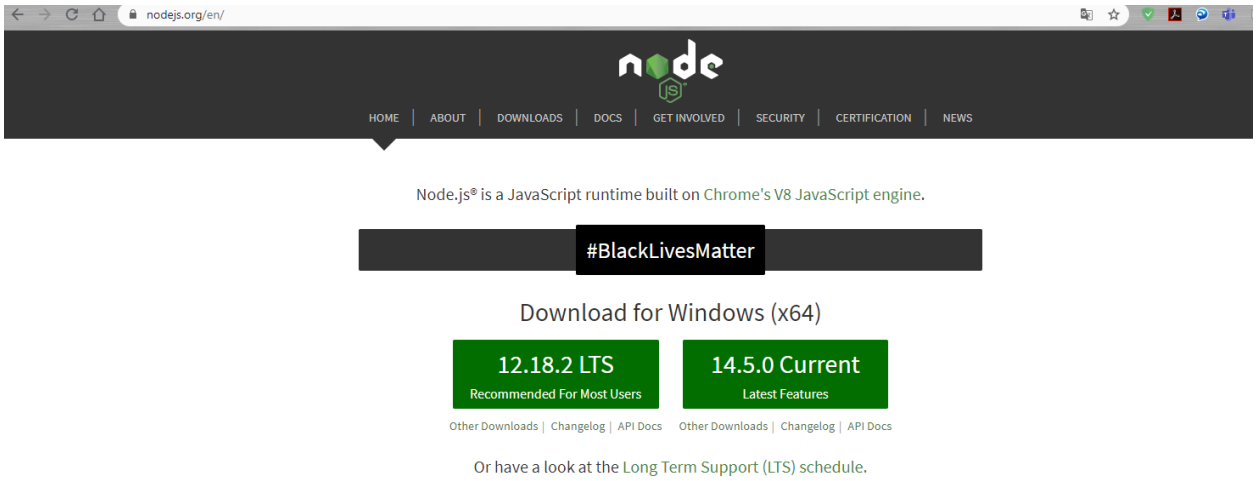
2. Pokrenuti bazu sa naredbom mongod u cmd-u



3.Skinuti bazu na github repozitoriju url: https://github.com/KristijanMifka/Ticketing_sustav/tree/master/ticketing-sustav%2009.07.2020/ticketing-sustav-najnovije --> [baza-ticketing sustav.json]

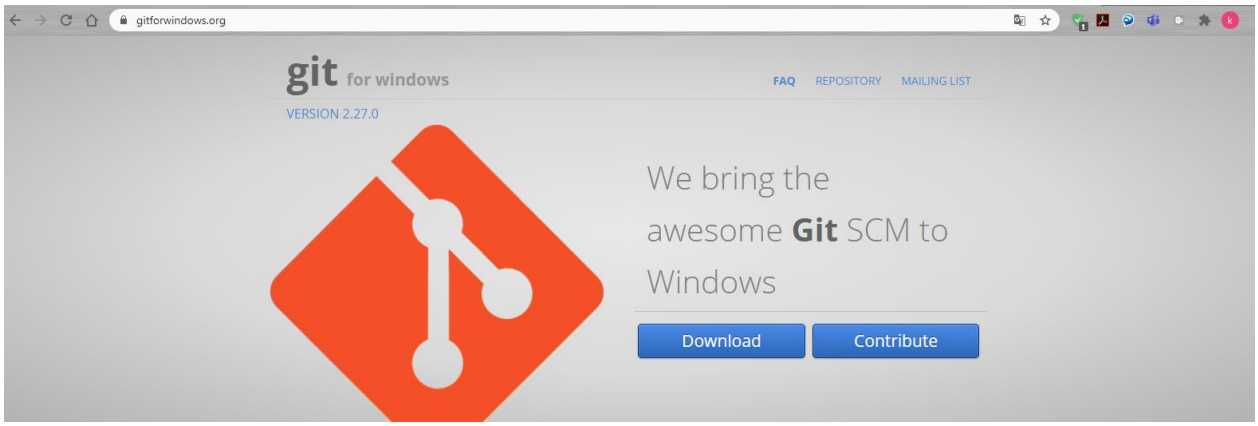
4. Unutar Mongodb Compass aplijacije napraviti import data → Collection – Import data,

6.2.2. INSTALACIJA NODE.JS APLIKACIJE

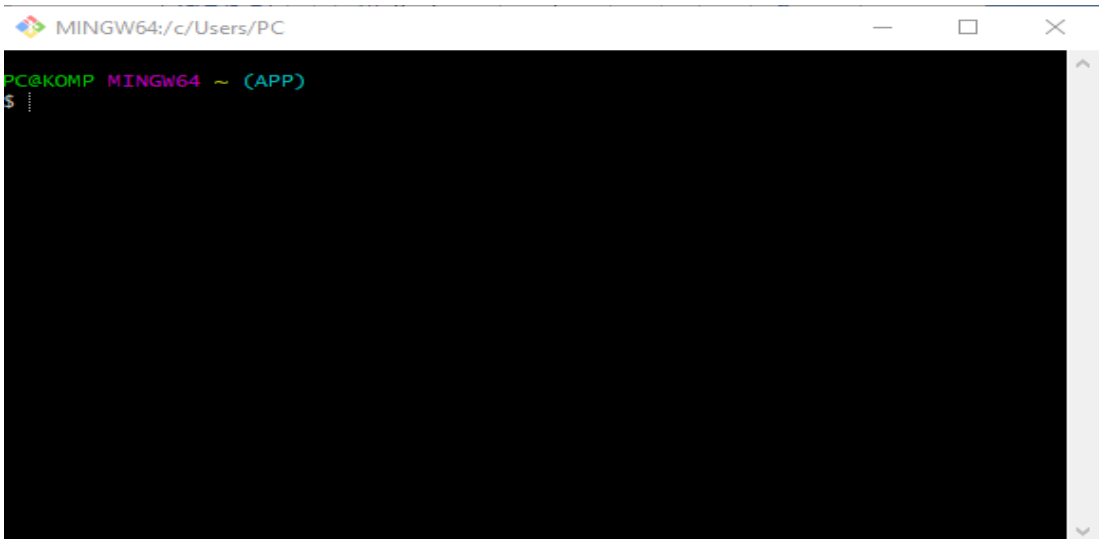


1. Aplikaciju preuzeti s url: https://nodejs.org/en/
2. Instalirati

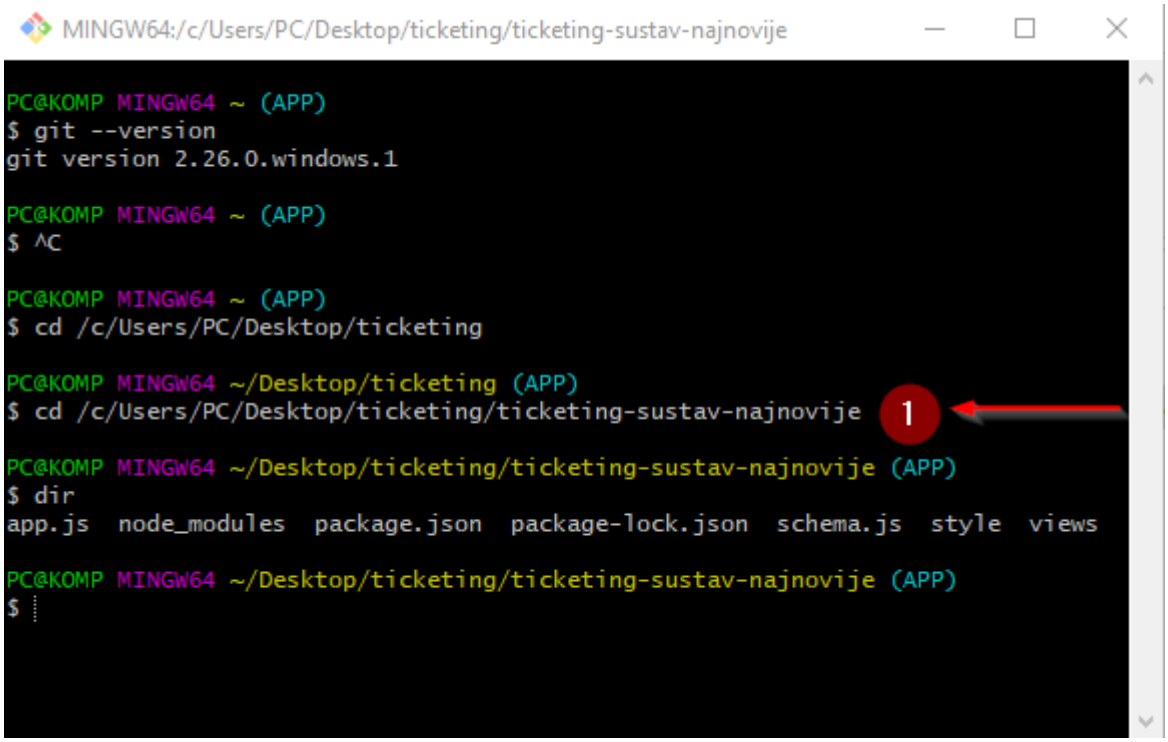
6.2.3 INSTALACIJA I POKRETANJE GITBASH APLIKACIJE



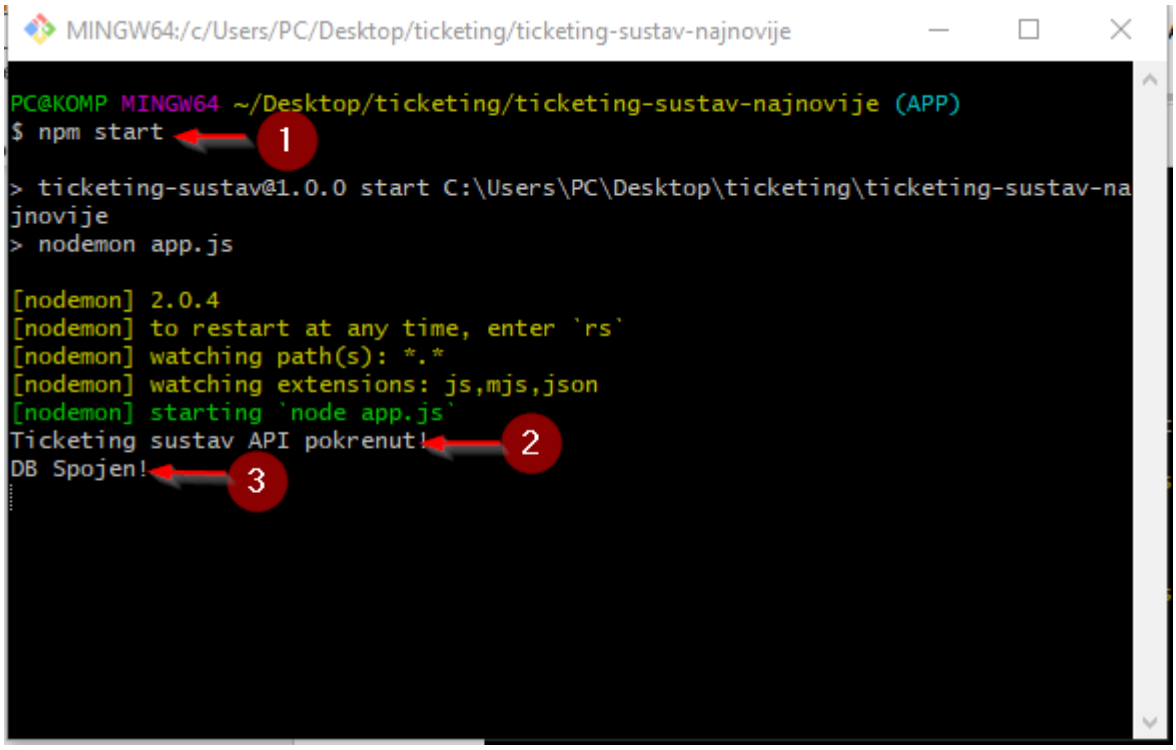
1. PREUZETI APLIKACIJU S URL: https://gitforwindows.org/
2. IZVRŠITI INSTALACIJU



3. POKRENUTI GITBASH APLIKACIJU



4. Pomoću naredbe cd pronači direktorij gdje ste preuzeli aplikaciju s github repozitorija

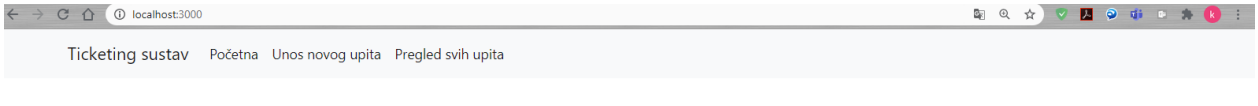


5. Upisati naredbu ***npm start***
6. Ispravno pokrenuta aplikacija→ispis poruka:

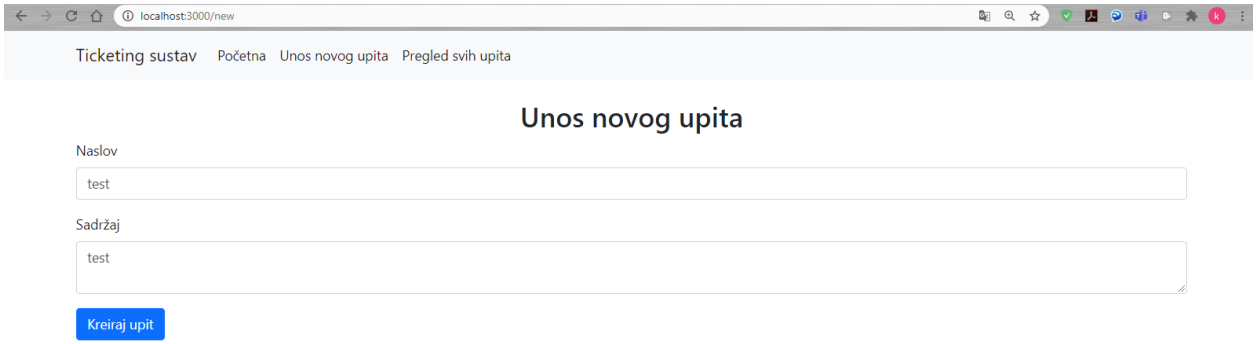
Ticketing sustav API pokrenut!

DB Spojen!

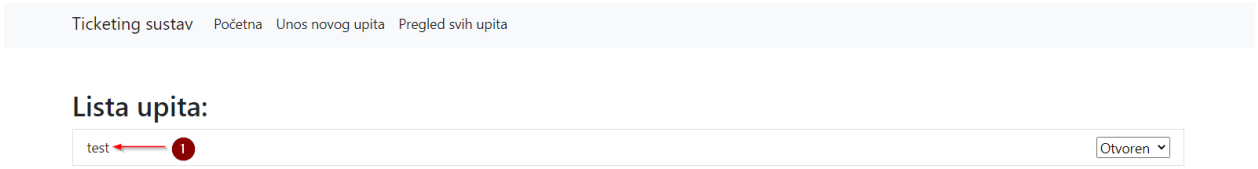
6.2.4 POKRETANJE APLIKACIJE U WEB PREGLEDNIKU TE KORIŠTNJE



1. Upisati u url: http://localhost:3000/

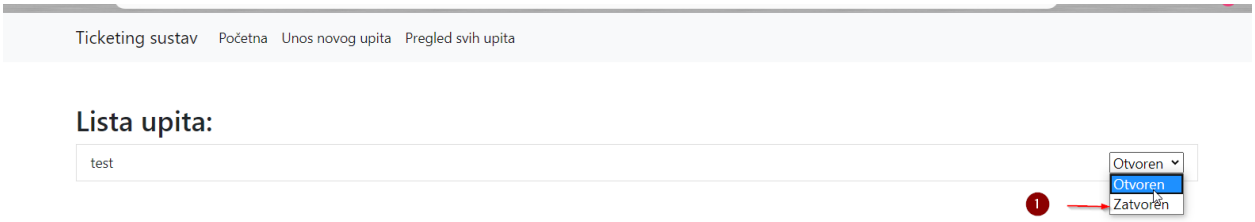


2. Pod unos novog upita unijeti traženi upit

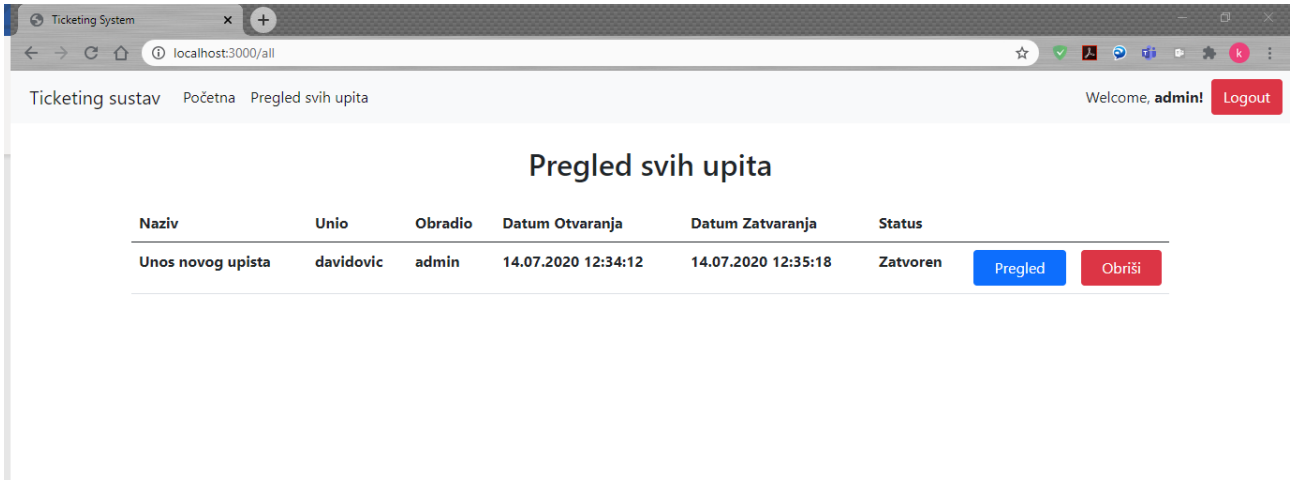


3. Odabrati uneseni upit

4. Upisati u polje odgovor za zatvaranje upita
5. Kliknuti izmjeni upit



6. Unutar početne stranice kliknuti “Zatvoren”→ tako zatvaramo riješeni upit korisnika



Unutar polja pregled svih upita mogučnost pregleda, te brisanja zatvorenog zahtjeva

7 Zaključak

Aplikacija je rađena u Node.js skriptnom jeziku. Node.js je serverska JavaScript platforma koja se sadrži od minimalne core biblioteke pored bogatog ekosistema. Radi na V8 JavaScript engine-u, što znači da je veoma brz u izvršavanju. Tradicionalno, programiranje se obavlja sinhrono: Linija koda se izvršava, sistem čeka rezultat, rezultat se procesuira i zatim se izvršavanje programa nastavlja. Ponekad taj sistem izvršavanja zahteva dugo čekanje; na primer čitanje sa baze podataka.

Aplikativni sustav ticketing funkcionira da se upit/problem šalje zaposleniku help deska. Aplikacija će biti integrirana kroz web sučelje. Korisnik posjeduje lozinku, te naziv korisnika. Može pregledavati ticket kojega je otvoro, te vidjeti trenutni status da li je otvoren ili zatvoren.Ticket se sastoji od šifre ticketa, rdb, naslova ticketa,datum postavljanja upita,datum zatvaranja, korisnika koji je zadao zahtjev, korisnika koji je zatvorio zahtjev, te statusa . Adminstrator ima pregled nad svim ticketima, a posjedovat će sljedeće parametre: sifra, te naziv admina. Zadovoljan sam konačnim riješenjem jer sam uspio pokriti sve funkcionalnosti koje su se od mene očekivale i tražile. Preporuke za daljnu nadogradnju su nadogranja css-a.

8 Literatura i izvori

1. https://stackoverflow.com/tags/mongodb/info
2. https://stackoverflow.com/questions/2353818/how-do-i-get-started-with-node-js
3. https://www.w3schools.com/nodejs/
4. https://www.guru99.com/mongodb-tutorials.html

9 Popis slika