

VELEUČILIŠTE U RIJECI
POSLOVNI ODJEL RIJEKA

KRISTIJAN MIFKA

Ticketing sustav

PROJEKTNNA 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)	4
2	Specifikacija zahtjeva	5
	Plan intervjua	6
	Detalji o korisniku sustava	7
	Zadaci korisnika	7
	Identifikacija problema	7
	Sumarizacija problema	8
	Identificiranje nefunkcionalnih zahtjeva	9
3	Analiza sustava	11
	3.1 Analiza zahtjeva i korištenje sustava	11
	3.2 Odabir tehnologija	16
4	Dizajn sustava	17
	4.1 Dizajn korisničkih sučelja	17
	4.2 Dijagram klasa	19
	4.3 Model podataka	20
5	Implementacija sustava	21
	5.1 Postavljanje radnog okruženja	21
	5.2 Verzije aplikacije	21
	5.3 Prikaz dijelova programskog koda	22
6	Isporuka i korištenje aplikacije	26
	6.1 Pakiranje i isporuka aplikacije	26
	6.2 Korisničke upute za korištenje aplikacije	26
7	Zaključak	32
8	Literatura i izvori	33
9	Popis slika	34

1 Opis sustava (Ticketing sustav)

Ticketing sustav ima svrhu za bilježenje aktivnosti koje se zadaju zaposleniku-administratoru. Korisnik šalje upit, zaposlenik ga rješava, 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 rješavanje 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 rješava, 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
- Rješava 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 sastavljenog 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.	Riješavanje upita/ uklanjanje problema	Otklanje softverskog ili hardverskog 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 automatizirao 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 izvršavaju 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- mogućnost 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 predocjenja 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 razvoj.

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 između IT odjela.

Koja su očekivanja korisnika vezana uz performanse sustava?

Napraviti jednostavniji ticketing sustav nego sadašnji, te mogućnost 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 koristiti 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

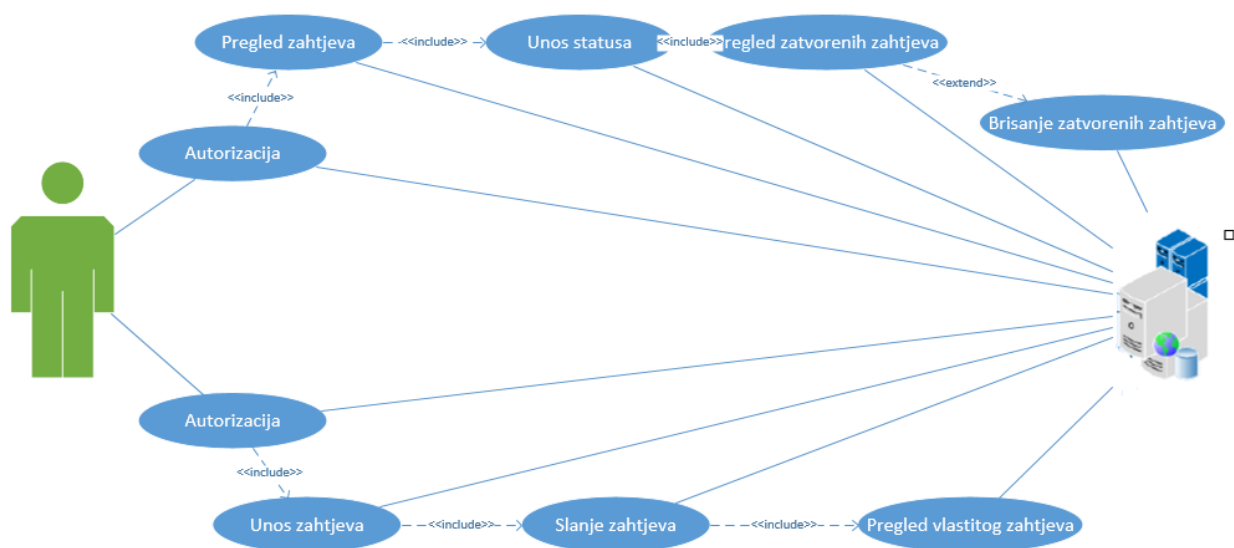
3.1 Analiza zahtjeva i korištenje sustava

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 mogućnost ažuriranja

A2: iz koraka G2 => Korisnik ima mogućnost zatvaranja upita ako u međuvremenu ukloni problem

Sekvence s greškom:

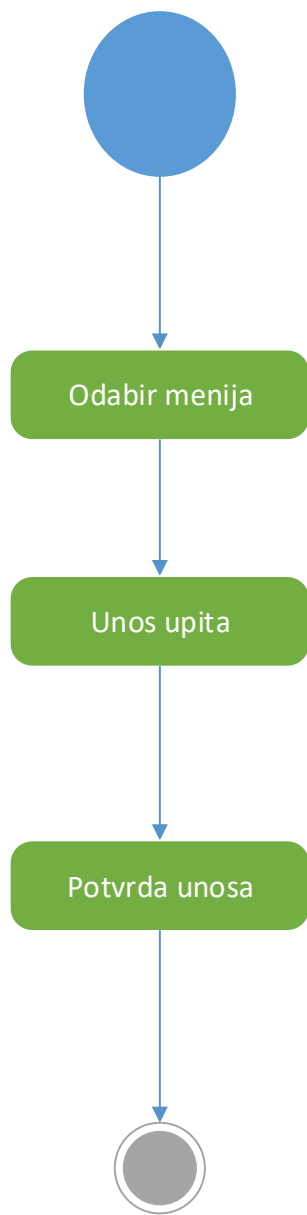
Ulazno-izlazni zahtjevi:

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 rješava ticket. Ticketi se mogu pregledati po nazivu i datumu kreiranja, te osobi koja je zadala i riješila zahtjev.

Datum zadnje izmjene: 13.07.2020.

Akteri: Zaposlenik

Preduvjeti: Ticket postoji u sustavu

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 ažurirati ga

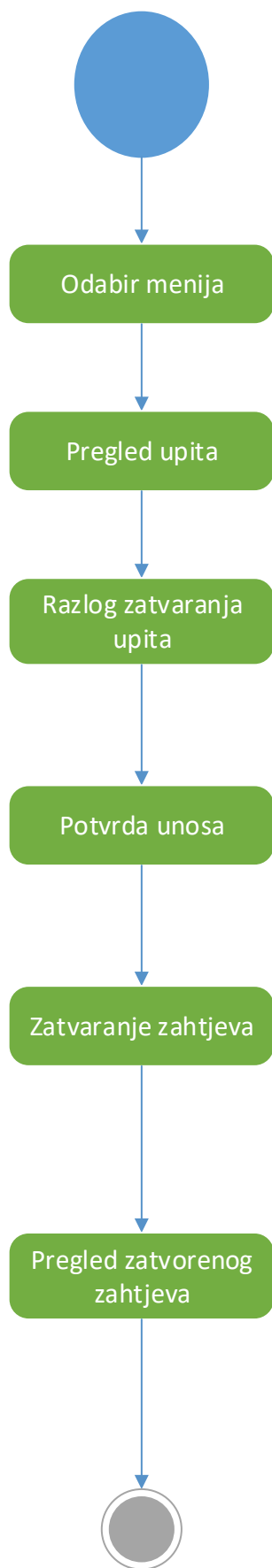
A2: iz koraka G2 => može odabrati ticket, te ga pregledati kada je i zatvoren.

Sekvence s greškom:

Ulazno-izlazni zahtjevi:

Nefunkcionalni uvjeti(ograničenja):

Slika 3: Dijagram aktivnosti za pregled ticketa

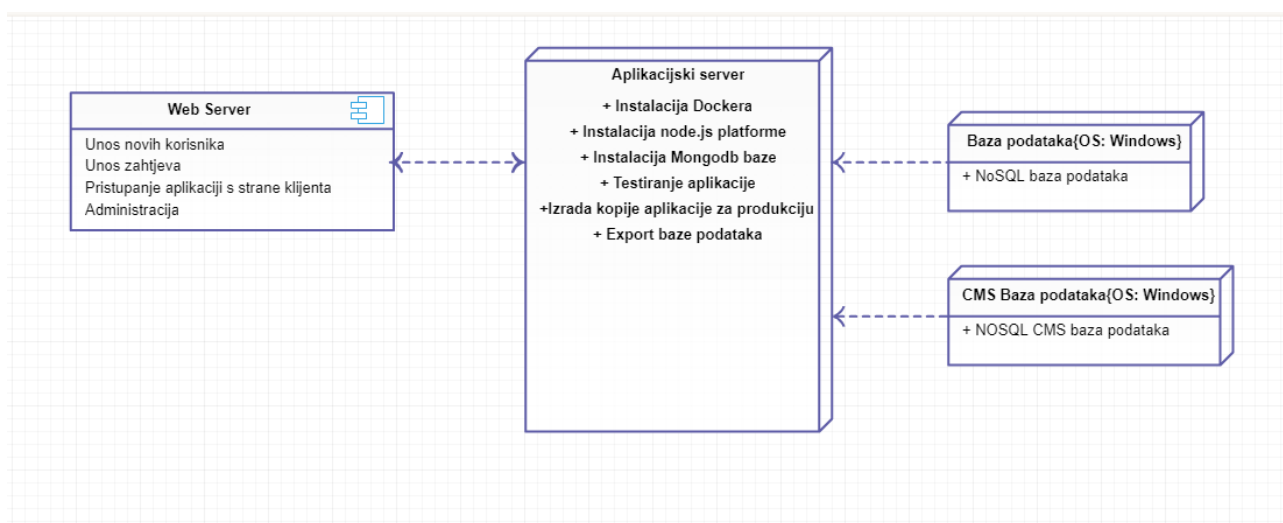


3.2 Odabir tehnologija

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 – Mongoddb v3.4.10



Slika 4: Dijagram 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

The screenshot displays a web application interface for a ticketing system. At the top, there is a navigation bar with the text "Ticketing sustav" on the left, and three buttons: "Početna", "Unos novog upita", and "Logout". On the right side of the navigation bar, it says "Welcome user!". Below the navigation bar, the main content area is titled "Lista upita:". It contains a table with three columns: "Naslov:", "Zatvorio", and "Status". The table has two rows of data. The first row shows "Unos novog upita" as the title, "Admin" as the person who closed it, and "Zatvoren" as the status. The second row shows "Unos novog upita 2" as the title and "Otvoren" as the status.

Naslov:	Zatvorio	Status
Unos novog upita	Admin	Zatvoren
Unos novog upita 2		Otvoren

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 mogućnost izlaza.

The screenshot shows the "Unos novog upita" (Enter new ticket) form. It is part of the same web application as the previous screenshot, with the same navigation bar. The form has two input fields: "Naslov" (Title) and "Sadržaj" (Content). Below these fields is a button labeled "Kreiraj upit" (Create ticket).

Unos novog upita

Naslov

Sadržaj

Kreiraj upit

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.

Naslov:	Zatvorio	Status
Novi upit		Otvoren <input checked="" type="checkbox"/>

Slika 7. Prikaz početnog sučelja zaposlenika- administratora

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.

Naziv	Unio	Obradio	Datum otvaranja	Datum zatvaranja	Status
Novi zahtjev	User 1	admin	10.7.2020. 10:58:12	10.7.2020. 10:58:12	Zatvoren

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



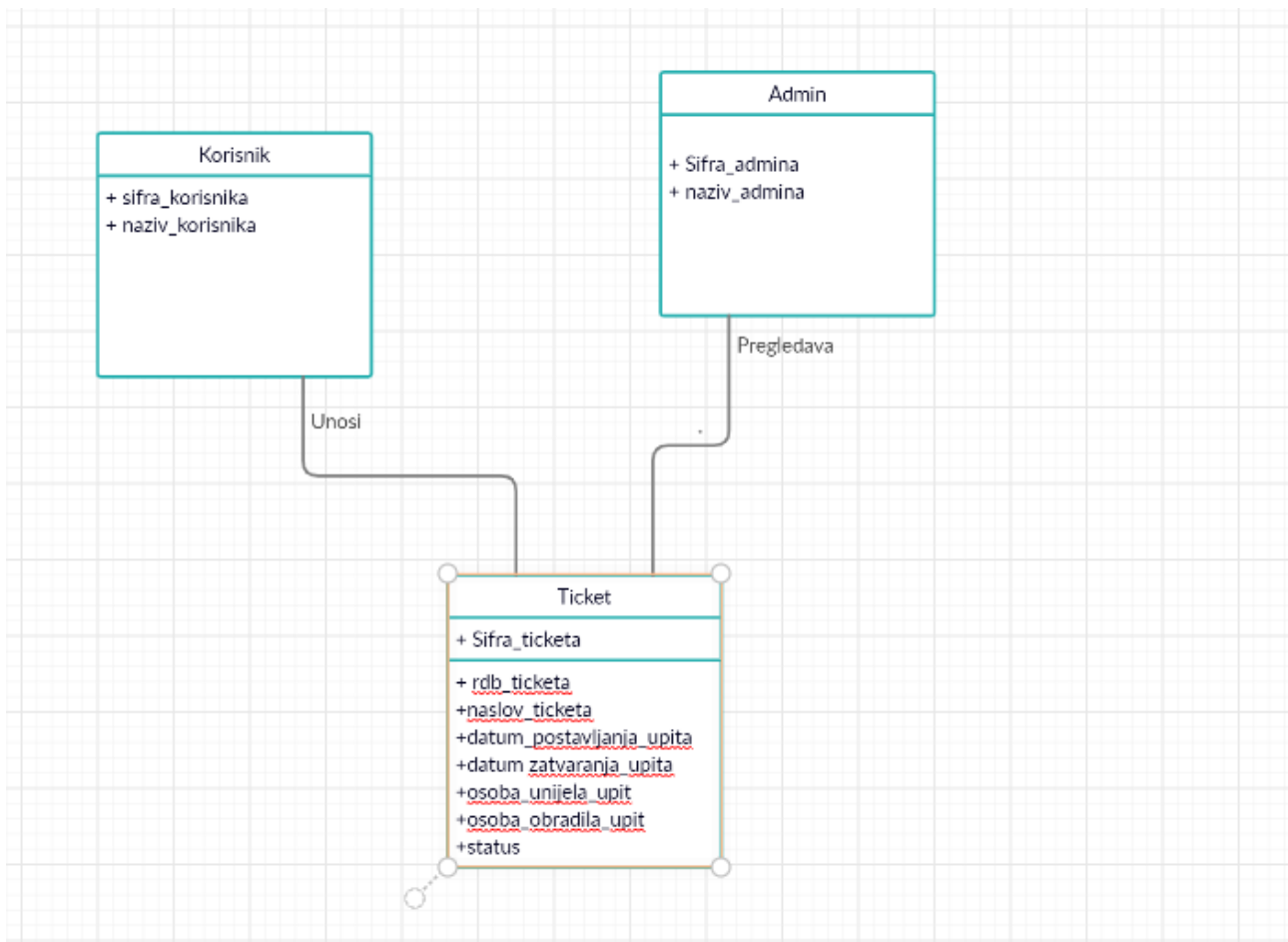
Slika 9. Prikaz dijagrama klasa

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 zatvaranja ticketa, statusa izvršenja, 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 funkcioniše na sljedeći način da korisnik se autorizira s svojim podacima. Nakon toga zadaje ticket koji dobiva zadane parametre, te administrator kao osoba zadužena za rješavanje 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

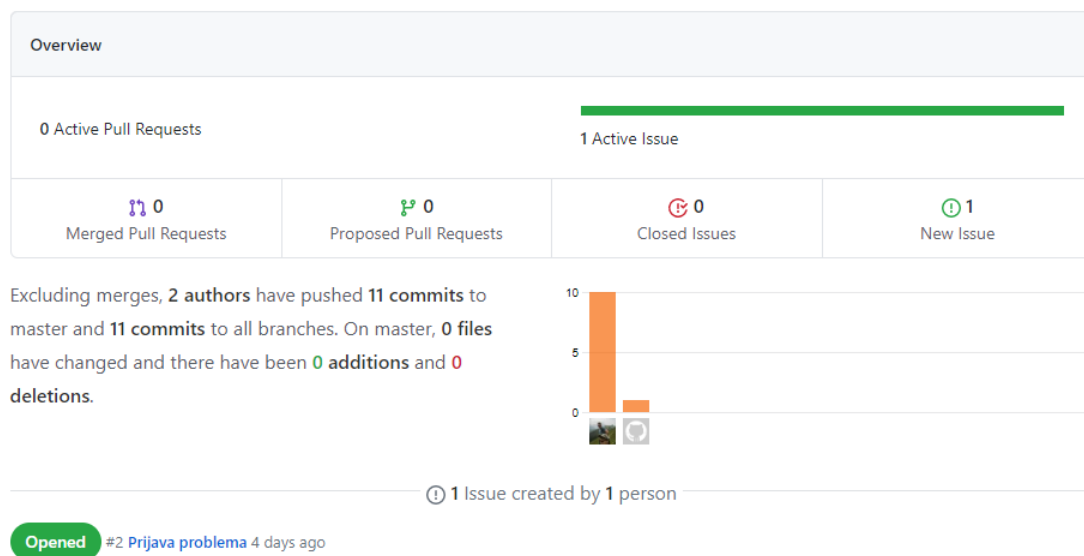
Link prema aplikaciji na GitHubu:

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

Datum prvog postavljanja: 09.07.2020.

July 7, 2020 – July 14, 2020

Period: 1 week ▼



Slika 11. Prikaz učestalosti i vremena postavljanja i nadogradnji

5.3 Prikaz dijelova programskog koda

```
1  const bcrypt = require("bcryptjs");
2  const mongoose = require("mongoose");
3  const { user } = require("../schema");
4
5  const User = mongoose.model("User", user);
6  const prompt = require("prompt");
7
8  prompt.start();
9
10 console.log("Dobrodošli u CLI za kreiranje korisnika! \n");
11
12 mongoose
13   .connect(
14     "mongodb://127.0.0.1:27017/ticketing-sustav",
15     { useNewUrlParser: true, useUnifiedTopology: true }
16   )
17   .then(() => {
18     console.log("DB Spojen!");
19     prompt.get(["username", "password", "role"], function (err, result) {
20       if (err) {
21         return onErr(err);
22       }
23
24       const username = result.username;
25       const password = result.password;
26       const role = result.role.toLowerCase();
27
28       const newUser = new User({
29         username,
30         password,
31         role,
32       });
33
34       bcrypt.genSalt(10, (err, salt) => {
35         bcrypt.hash(newUser.password, salt, (err, hash) => {
36           console.log("Hashing password...");
37           if (err) {
38             console.log("error");
39             throw err;
40           }
41         });
42       });
43     });
44   });
45 }
```

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

```
33
34     bcrypt.genSalt(10, (err, salt) => {
35       bcrypt.hash(newUser.password, salt, (err, hash) => {
36         console.log("Hashing password...");
37         if (err) {
38           console.log("error");
39           throw err;
40         }
41         newUser.password = hash;
42         console.log("Password hash is:", hash);
43         newUser
44           .save()
45           .then((user) => {
46             console.log("Saving user to DB...");
47             console.log("User created!", user);
48             process.exit();
49           })
50           .catch((err) => console.error(err));
51       });
52     });
53   });
54 }
55 .catch(() => console.error.bind(console, "Greška pri spajanju na DB:"));
56
57 function onErr(err) {
58   console.error(err);
59   return 1;
60 }
61 }
```

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

Klasa cli.js služi za kreiranje korisnika. Svaki korisnik 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.

```
config > JS passportjs > ...
1  const LocalStrat = require("passport-local").Strategy;
2  const bcrypt = require("bcryptjs");
3
4  const { User } = require("../schema");
5
6  module.exports = function (passport) {
7    passport.use(
8      new LocalStrat(
9        { usernameField: "username" },
10        async (username, password, done) => {
11          console.log("passport data:", username, password, done);
12          try {
13            const user = await User.findOne({ username: username });
14            if (!user) {
15              throw done(null, false, { message: "Korisnik ne postoji!" });
16            }
17
18            const match = await bcrypt.compare(password, user.password);
19
20            if (match) {
21              return done(null, user);
22            } else {
23              return done(null, false, { message: "Lozinka neispravna!" });
24            }
25          } catch (err) {
26            console.error(err);
27          }
28        }
29      )
30    );
31  }
```

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

```
31
32  passport.serializeUser((user, done) => {
33    done(null, user.id);
34  });
35
36  passport.deserializeUser((id, done) => {
37    User.findById(id, function (err, user) {
38      done(err, user);
39    });
40  });
41 };
42
```

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 gdje pregledava iskrenirane 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
1 <h2 class="text-center mb-4">Pregled svih upita</h2>
2
3 <table class="table">
4   <thead>
5     <tr>
6       <th scope="col">Naziv</th>
7       <th scope="col">Unio</th>
8       <th scope="col">Obradio</th>
9       <th scope="col">Datum Otvaranja</th>
10      <th scope="col">Datum Zatvaranja</th>
11      <th scope="col">Status</th>
12      <th scope="col"></th>
13      <th scope="col"></th>
14    </tr>
15  </thead>
16  <tbody>
17    <% tickets.forEach(ticket => {%)
18    <% if (ticket.status === 'Zatvoren'){%)
19      <tr>
20        <th><%=ticket.title%></th>
21        <th><%=ticket.user.username%></th>
22        <th><%=ticket.reviewer.username%></th>
23        <th><%= moment(ticket.dateOfCreation).format('DD.MM.YYYY h:mm:ss') %></th>
24        <th><%=moment(ticket.closeDate).format('DD.MM.YYYY h:mm:ss') %></th>
25        <th><%=ticket.status%></th>
26        <th class="text-center">
27          <a href="/ticket/<%= ticket._id %>?referrer='all'">
28            <button class="btn btn-primary w-100">Pregled</button>
29          </a>
30        </th>
31        <th class="text-center"><button onclick="deleteTicket('<%=ticket._id%>')" class="btn btn-danger w-100">Obr
32      </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

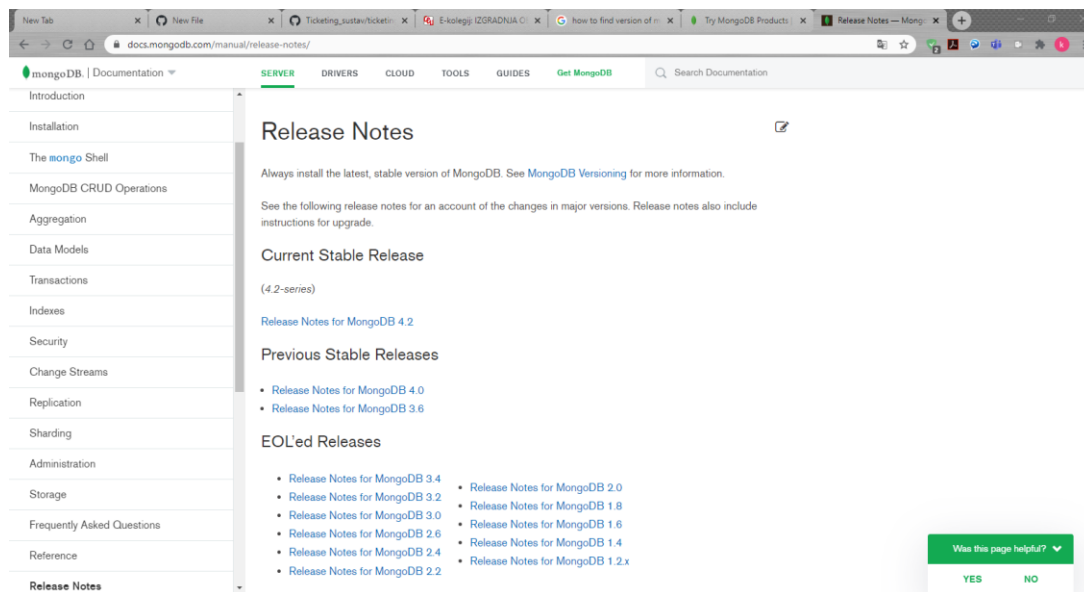
Aplikacija se će se u dogovoru s kolegama iz BKS bank AG postaviti na serversku infrastrukturu. Održavanje je ažuriranje baze podataka.

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 – Mongoddb v3.4.10
3. Aplikacija Gitbash v 2.26.0.
4. Web preglednik: Mozila, Google Chrome, Edge.

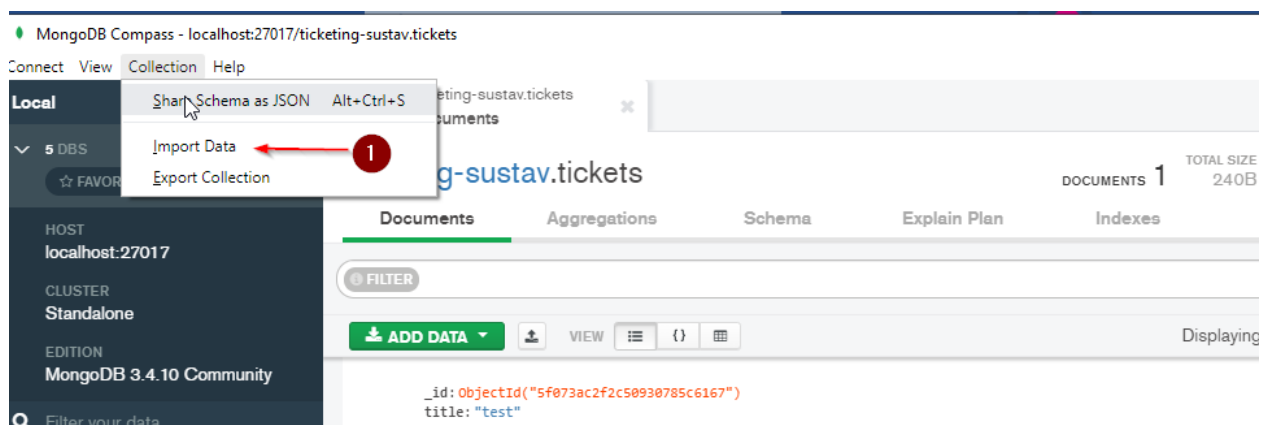
6.2.1 INSTALACIJA I IMPORT BAZE PODATAKA -MONGODB



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

```
Command Prompt - mongod
C:\Users\PC>mongod
2020-07-09T18:23:03.158+0200 I CONTROL [initandlisten] MongoDB starting : pid=4512 port=27017 dbpath=C:\data\db\ 64-bit
host=KOMP
2020-07-09T18:23:03.160+0200 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-07-09T18:23:03.161+0200 I CONTROL [initandlisten] db version v3.4.10
2020-07-09T18:23:03.161+0200 I CONTROL [initandlisten] git version: 078f28920cb24de0dd479b5ea6c66c644f6326e9
2020-07-09T18:23:03.161+0200 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1u-fips 22 Sep 2016
2020-07-09T18:23:03.161+0200 I CONTROL [initandlisten] allocator: tcmalloc
2020-07-09T18:23:03.161+0200 I CONTROL [initandlisten] modules: none
2020-07-09T18:23:03.161+0200 I CONTROL [initandlisten] build environment:
2020-07-09T18:23:03.161+0200 I CONTROL [initandlisten] distmod: 2008plus-ssl
2020-07-09T18:23:03.161+0200 I CONTROL [initandlisten] distarch: x86_64
2020-07-09T18:23:03.162+0200 I CONTROL [initandlisten] target_arch: x86_64
2020-07-09T18:23:03.162+0200 I CONTROL [initandlisten] options: {}
2020-07-09T18:23:03.164+0200 I - [initandlisten] Detected data files in C:\data\db\ created by the 'mmapv1' storage
engine, so setting the active storage engine to 'mmapv1'.
2020-07-09T18:23:03.191+0200 I JOURNAL [initandlisten] journal dir=C:\data\db\journal
2020-07-09T18:23:03.192+0200 I JOURNAL [initandlisten] recover : no journal files present, no recovery needed
2020-07-09T18:23:03.216+0200 I JOURNAL [durability] Durability thread started
2020-07-09T18:23:03.218+0200 I JOURNAL [journal writer] Journal writer thread started
2020-07-09T18:23:03.219+0200 I CONTROL [initandlisten]
2020-07-09T18:23:03.219+0200 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-07-09T18:23:03.219+0200 I CONTROL [initandlisten] ** Read and write access to data and configuration is u
nrestricted.
2020-07-09T18:23:03.219+0200 I CONTROL [initandlisten]
2020-07-09T18:23:04.324+0200 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C
:\data\db\diagnostic.data'
2020-07-09T18:23:04.332+0200 I NETWORK [thread1] waiting for connections on port 27017
```

2. Pokrenuti bazu sa naredbom mongod u cmd-u

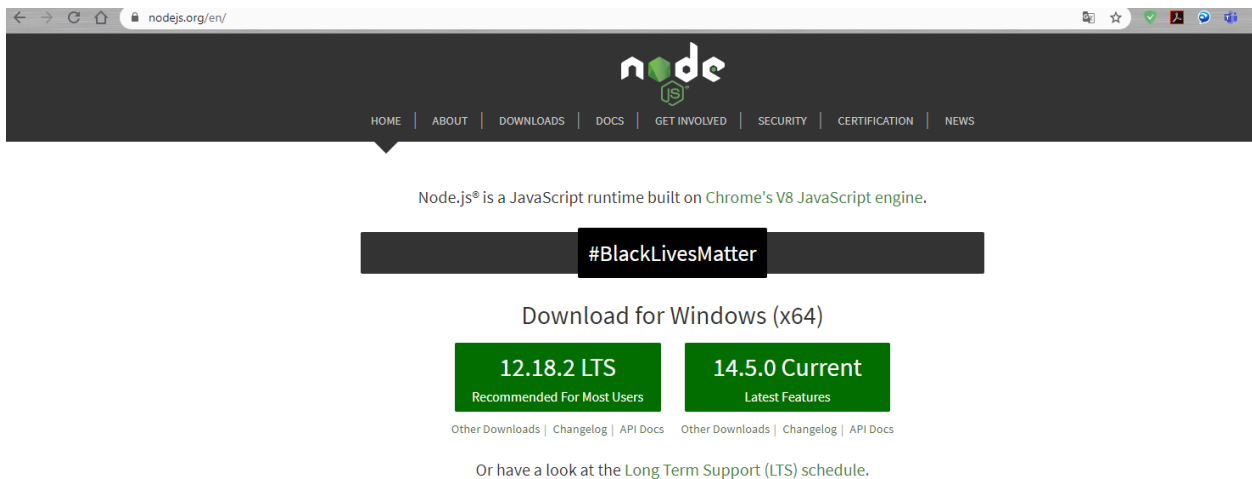


3. Skinuti bazu na github repozitoriju url:

https://github.com/KristijanMifka/Ticketing_sustav/tree/master/ticketing-sustav%2009.07.2020/ticketing-sustav-najnovije --> [baza-ticketing sustav.json](#)

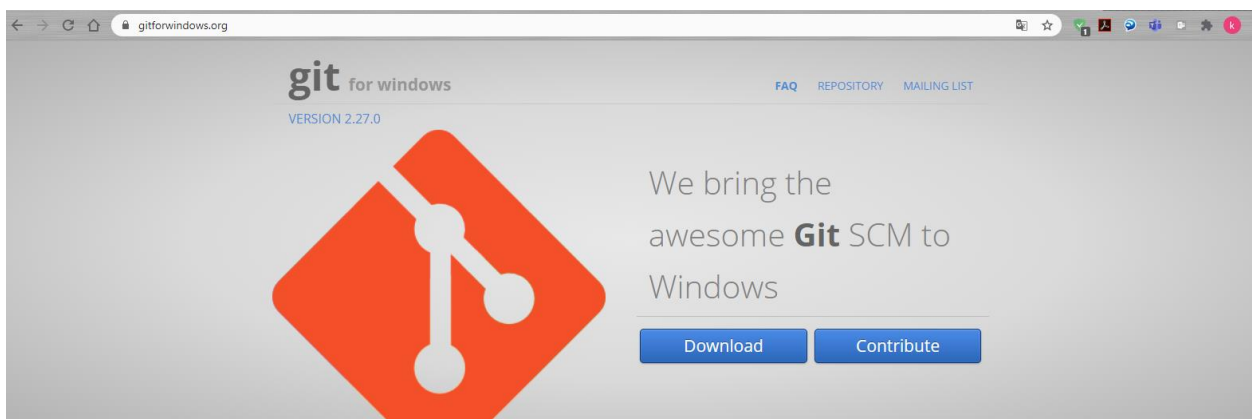
4. Unutar Mongoddb Compass aplikacije napraviti import data → Collection – Import data,

6.2.2. INSTALACIJA NODE.JS APLIKACIJE

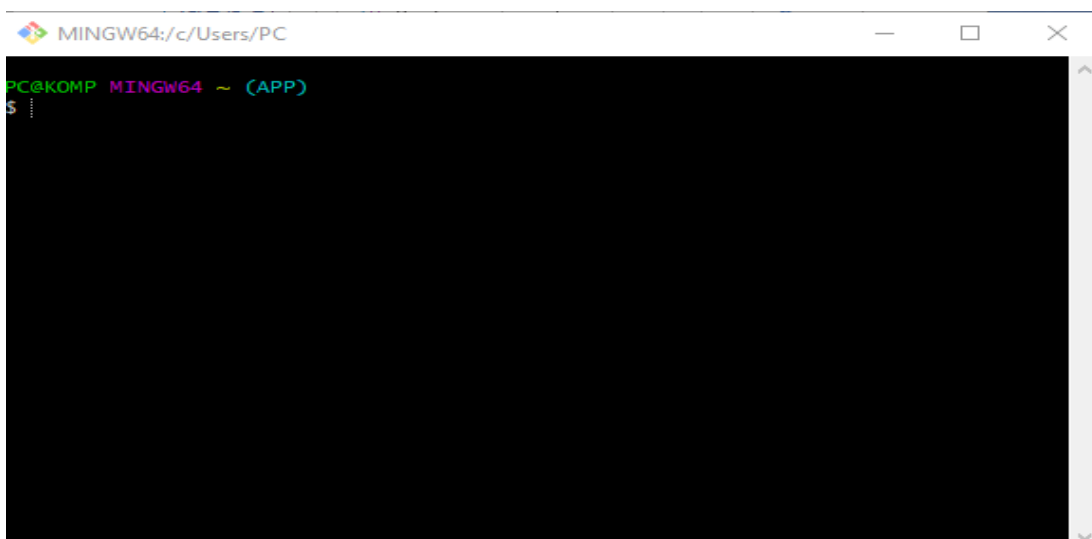


1. Aplikaciju preuzeti s url: <https://nodejs.org/en/>
2. Instalirati

6.2.3 INSTALACIJA I POKRETANJE GITBASH APLIKACIJE



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



3. POKRENUTI GITBASH APLIKACIJU

```
MINGW64:/c/Users/PC/Desktop/ticketing/ticketing-sustav-najnovije

PC@KOMP MINGW64 ~ (APP)
$ git --version
git version 2.26.0.windows.1

PC@KOMP MINGW64 ~ (APP)
$ ^C

PC@KOMP MINGW64 ~ (APP)
$ cd /c/Users/PC/Desktop/ticketing

PC@KOMP MINGW64 ~/Desktop/ticketing (APP)
$ cd /c/Users/PC/Desktop/ticketing/ticketing-sustav-najnovije 1
PC@KOMP MINGW64 ~/Desktop/ticketing/ticketing-sustav-najnovije (APP)
$ dir
app.js  node_modules  package.json  package-lock.json  schema.js  style  views

PC@KOMP MINGW64 ~/Desktop/ticketing/ticketing-sustav-najnovije (APP)
$
```

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

```
MINGW64:/c/Users/PC/Desktop/ticketing/ticketing-sustav-najnovije

PC@KOMP MINGW64 ~/Desktop/ticketing/ticketing-sustav-najnovije (APP)
$ npm start 1
> ticketing-sustav@1.0.0 start C:\Users\PC\Desktop\ticketing\ticketing-sustav-na
jnovije
> nodemon app.js

[nodemon] 2.0.4
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'node app.js'
Ticketing sustav API pokrenut! 2
DB Spojen! 3
.....
```

5. Upisati naredbu `npm start`

6. Ispravno pokrenuta aplikacija → ispis poruka:

Ticketing sustav API pokrenut!

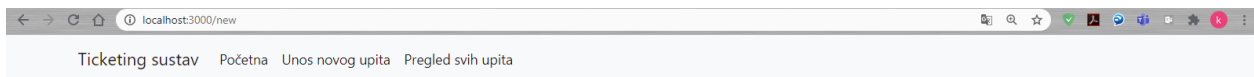
DB Spojen!

6.2.4 POKRETANJE APLIKACIJE U WEB PREGLEDNIKU TE KORIŠTNJE



Lista upita:

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



Unos novog upita

Naslov

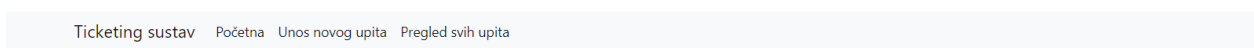
test

Sadržaj

test

Kreiraj upit

2. Pod unos novog upita unijeti traženi upit



Lista upita:



3. Odabrati uneseni upit

Ticketing sustav Početna Unos novog upita Pregled svih upita

Pregled upita: test

Naslov
test

Sadržaj
test2

Razlog zatvaranja upita
Riješeno 1

Izmjeni upit 2

4. Upisati u polje odgovor za zatvaranje upita

5. Kliknuti izmjeni upit

Ticketing sustav Početna Unos novog upita Pregled svih upita

Lista upita:

test

Otvoren
Otvoren
Zatvoren 1

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

Ticketing System

localhost:3000/all

Ticketing sustav Početna Pregled svih upita

Welcome, admin! Logout

Pregled svih upita

Naziv	Unio	Obradio	Datum Otvaranja	Datum Zatvaranja	Status
Unos novog upista	davidovic	admin	14.07.2020 12:34:12	14.07.2020 12:35:18	Zatvoren

Pregled Obriši

Unutar polja pregled svih upita mogućnost 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 procesira 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 otvorio, 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. Administrator 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 nadogradnja 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

Slika 1: Dijagram slučajeva korištenja.....	11
Slika 2: Dijagram aktivnosti za unos novih ticketa.	13
Slika 3: Dijagram aktivnosti za pregled ticketa.....	15
Slika 4: Dijagram isporuke.....	16
Slika 5. Prikaz prozora korisnika.....	17
Slika 6. Prikaz unosa novog upita.....	17
Slika 7. Prikaz početnog sučelja zaposlenika- administratora.....	18
Slika 8. Prikaz pregleda svih zatvorenih upita.....	18
Slika 9. Prikaz dijagrama klasa.....	19
Slika 10. Prikaz nerelacijskog modela podataka.....	20
Slika 11. Prikaz učestalosti i vremena postavljanja i nadogradnji.....	21
Slika 12. Prikaz cli.js klase- Linija koda 1-30.....	22
Slika 13. Prikaz cli.js klase- Linija koda 34-61.....	22
Slika 14. Prikaz passport.js klase Linija koda 1-31.....	23
Slika 15. passport.js klase Linija koda 31-42.....	23
Slika 16. Prikaz app.js klase- Linija koda 31-42.....	24
Slika 17. Prikaz all.html klase- Linija koda 1-32.....	24