Programsko inženjerstvo

Ak. god. 2021./2022.

Halo112

Dokumentacija, Rev. *2*

Grupa: *Hakeri*

Voditelj: *Luka Ivanković*

Datum predaje: *14. 01. 2022.*

Nastavnik: *Hrvoje Nuić*

Sadržaj

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Luka Ivanković	26.10.2021.
0.2	Napisan dio obrazaca upotrebe	Krunoslav Zadrić	27.10.2021.
0.3.1	Napisani svi obrasci uporabe.	Florijan Rusac	28.10.2021.
0.3.2	Dorada obrazaca upotrebe	Krunoslav Zadrić	29.10.2021.
0.3.3	Dijagram obrazaca uporabe	Leon Banko, Lukas Ujčić	29.10.2021.
0.3.4	Sekvencijski dijagrami	Leon Banko, Lukas Ujčić	30.10.2021.
0.6	Opis projektnog zadatka	Luka Ivanković	7.11.2021.
0.6.1	Slični primjeri	Mario Galić	8.11.2021.
0.7	Arhitektura baze podataka	Krunoslav Zadrić	9.11.2021.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.7.1	Započeli dijagrame razreda	Leon Banko, Lukas Ujčić	9.11.2021.
0.8	Dovršena arhitektura baze podataka	Florijan Rusac	10.11.2021.
0.8	Opis ostalih zahtjeva	Mario Galić	10.11.2021.
0.9	Opisana arhitektura	Florijan Rusac	19.11.2021.
0.10	Provjera pravopisa i općeg dojma	Mario Galić, Florijan Rusac	19.11.2021.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus		19.11.2021.
1.5	Manja revizija (dijagram komponenti, dijagram aktivnosti, korištene tehnologije i alati)	Leon Banko	12.01.2022.
1.6	Ispitivanje programskog rješenja	Leon Banko, Lukas Ujčić	13.01.2022.
1.7	Dijagram razmještaja i Dijagram stanja	Lukas Ujčić	13.01.2022.
1.8	Popravljeni dijagrami obrazaca uporabe, implementacijski dijagram razreda	Leon Banko, Lukas Ujčić	14.01.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
2.0	Konačni tekst predloška dokumentacije		14.01.2022.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije „Halo112“. To je aplikacija za olakšavanje koordinacije između rada svih spasilačkih službi. Neregistriranom korisniku se pojavljuje stranica za prijavu. Potrebno je unijeti korisničko ime i lozinku. Ispod polja za unos postoji hiperlink „registriraj se“ – vodi korisnika na stranicu za registraciju.

Na stranici za registraciju potrebno je unijeti osobne podatke:

- korisničko ime
- lozinka (potrebno je dva puta upisati lozinku na dva različita polja za tekst radi sigurnosti)
- ime
- prezime
- email adresa
- broj mobitela
- fotografija(opcionalno)

Sve osim podataka na kojima piše opcionalno su obavezni za unos, i ne može se nastaviti ako nisu popunjena. Osim osobnih podataka, treba se odabrati i uloga:

- dispečer
- doktor
- vatrogasac
- policajac

Korisnik

Nakon što je korisnik popunio sva potrebna polja, na dnu stranice postoji gumb „pošalji prijavnicu“. Ako se prelazi mišem preko gumba može se vidjeti: „Administrator mora odobriti vašu prijavu“. Klikom na gumb dešavaju se dvije stvari: administratoru dolazi obavijest o registraciji novog korisnika (on može odobriti ili onemogućiti prijavu i također vidi popis svih ostalih registriranih korisnika), korisnika se šalje na novu stranicu. Na stranici piše: „Obavijestit ćemo vas e-mailom

kada administrator odobri vašu prijavnicu.“. Ispod natpisa postoji hiperlink „vrati se na stranicu za prijavu“.

Administrator

Administratoru se nakon prijave u sustav prikazuje, na početnoj stranici, popis korisnika kojima se čeka potvrda registracije. On može odobriti registraciju ili odbiti, čime će se automatski poslati mail korisniku o odobrenju/otkazanju njegove registracije. Na posebnoj stranici ima popis korisnika u sustavu. Svakom korisniku ima pristup u osobne podatke i podatke o ulozi (dispečer, doktor, vatrogasac ili policajac). Sve od navedenog administrator može promijeniti. Na trećoj stranici postoji popis stanica u gradu koje može dodavati ili brisati.

Spasioci

Doktori, vatrogasci i policajci su spasioci. Svaki spasilac pripada nekoj stanici (npr. KBC Rebro, PP Trešnjevka, VP Dubrava.) Kojoj stanici pripada određuje voditelj te stanice (on je isto spasilac). Spasilac će trebati raditi određene akcije spašavanja, i ponekad neće biti dostupan (izvan radnog vremena, trenutno izvodi drugu akciju). On može ručno namjestit da li je spreman ili nije u postavkama profila, a dok izvodi drugu akciju automatski se postavlja da je nespreman za ostale akcije.

Ako spasilac ne izvodi trenutno nikakvu akciju i ako je slobodan prikazuje mu se tablica sa zahtjevima za uključivanje u akcije (Akcije zadaje dispečer u posebnom sučelju). On se može odazvati na jednu od ponuđenih akcija (ako ih uopće ima). Za svaku akciju postoje detalji. Prikazuju se neke opće informacije o opisu problema, kolika je razina hitnosti za obradu akcije i kako bi se spasilac kojemu je zahtjev poslan trebao odazvati (pješke, autom, motorom...). Mogu se prikazati i fotografije. Nakon što se odazvao na neku akciju, ne može odabrati drugu dok ova akcija nije završena ili dok ga dispečer nije uklonio s trenutne akcije.

Spasiteljima koji su aktivni na nekoj akciji na početnoj im se stranici ne prikazuje više tablica ponuđenih akcija, nego mapa grada. Na mapi grada nalazi se sve vezano uz trenutnu akciju, može vidjeti sve ostale spasioce koji trenutno sudjeluju u toj akciji i na karti mu se prikazuju zadaci koje može izvoditi. Svaki zadatak je statičan i prikazuje se na mapi kao točka. Ako se mišem stisne na zadatak mogu se vidjeti detalji zadatka. U detaljima piše kratki komentar što se treba obaviti i kako. Za neke zadatke može se na mapi prikazati ruta, kao put kojim bi spasilac trebao doći do lokacije.

Voditelj Stanice

Voditelj stanice ponaša se isto kao običan spasilac u gotovo svemu. Specijalno kod

njega je što ima posebnu stranicu gdje može odabrati koji spasioci su dio njegove stanice. Također definira na koji su način spasilaci osposobljeni voditi spašavanje. Doktor može biti osposobljen za vožnju motociklom ili kao putnik u kolima hitne pomoći. Vatrogasac može biti osposobljen za vožnju autocisterni, autoljestvi, zapovjednog vozila i šumskog vozila. Policajac se može kretati pješke kao kontaktni policajac, pomoću motocikla, automobila i oklopnog vozila.

Dispečer

Dispečer na temelju prijave otvara akcije spašavanja s dostupnim informacijama i fotografijama. Dispečer vidi broj dostupnih spasioca po stanicama i može poslati zahtjev za uključivanjem spasilaca u akciju spašavanja. Prilikom slanja zahtjeva dispečer definira na koji način bi spasilac trebao sudjelovati (auto, pješke..) i koja je razina hitnosti. Spasioci koji zadovoljavaju kriterije se mogu odazvati na akciju. Dispečer, ako je to potrebno, može spasioca ukloniti s akcije. Ako je akcija spašavanja završila, dispečer je u sustavu može označiti kao gotovom.

Dispečer preko karte spasiocima pojedinačno zadaje zadatke. Zadatak može tražiti prolazak određenom rutom i dolazak do određene lokacije. Svaki zadatak može imati i dodatan komentar od dispečera. Za izračun ruta koje prate staze i ceste koristi se vanjski servis OSRM2.

Dispečeru se na temelju trenutnih pozicija spasilaca prikazuje Voronojev dijagram. Dispečer može odabrati da se za izradu dijagrama koriste pozicije svih spasioca, ili svih dostupnih neaktivnih spasioca, ili aktivnih spasioca na određenoj akciji. Ovisno o načinu na koji spasilac sudjeluje u akciji, dispečeru se na karti prikazuje drugačija ikona.

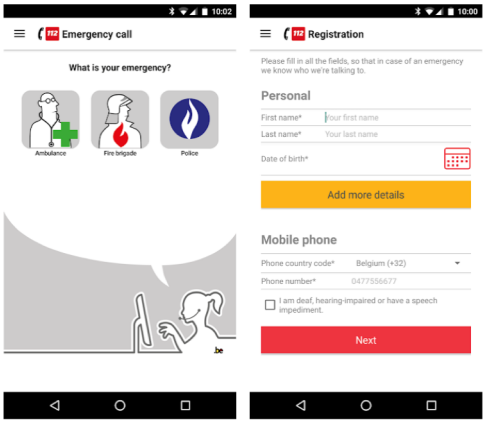
Potencijalna korist projekta

Ovaj projekt mogao bi biti od velike koristi bilo kojoj spasilačkoj službi jer jako olakšava komunikaciju između služba. Ako se dogodi nekakva nesreća/problem, puno prije i lakše spasioci mogu iskomunicirati tko se i gdje treba nalaziti da što prije otklone problem.

Postojeća slična rješenja

Slično rješenje u svijetu je belgijska aplikacija ”112 BE”. ”112 BE” je besplatna službena aplikacija belgijskog tima za hitne slučajeve (policija, vatrogasci, hitna pomoć). Nakon registracije u aplikaciju moguće je komentirati službu za korisnike koja uz pomoć vaše lokacije odabire najbliže spasioce koje treba poslati. U ovoj verziji se komunicira preko interneta no ukoliko nema interneta postoji i mogućnost SMS poruka koja je u ovom slučaju besplatna. Također ukoliko osoba

nije u mogućnosti govoriti otvara se chat kako bi se moglo komunicirati



Slika 2.1: Primjer postojećeg rješenja

Skup korisnika koji bi mogao biti zainteresiran za ostvareno rješenje
Ovaj projekt može koristiti bilo kakav spasilac. Vrlo vjerojatno bi voditelj stanice određivao bi li se koristila aplikacija ili ne. Ako se odluči da bi aplikacija bila korisna mogao bi ju uvesti kao obavezno sredstvo za komuniciranje između svojih spasilaca. Organizacija bi im se odmah poboljšala.

Mogućnost prilagodbe rješenja
Ovaj program je jako fleksibilan i može se uvesti jako puno različitih funkcionalnosti. Ako se tijekom izrade programa neka funkcionalnost čini redundantna, može se lako zamjeniti drugom ili ukloniti. Dodatne funkcionalnosti se također vrlo lako mogu dodavati.

Opseg projektnog zadatka
Ovaj zadatak se može implementirati na bezbroj različitih načina od kojih postoje i teži i lakši načini. Naravno cilj je napraviti ispravan i koristiv program, a ne si olakšavati ili otežavati zadatak radi posla. Svaku funkcionalnost potrebno je dodati na prvenstveno ispravan način bez nepotrebnog otežavanja zadatka.

Moguće nadogradnje projektnog zadatka
Ovaj zadatak se može vrlo jednostavno urediti naknadno. Uz ideju nekog voditelja stanice mogle bi se dodati nove funkcionalnosti za tu stanicu ili za sve korisnike.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Neregistrirani/neprijavljeni korisnik
2. Spasilac
3. Doktor(Spasilac)
4. Vatrogasac(Spasilac)
5. Policajac(Spasilac)
6. Voditelj stanice
7. Dispečer
8. Administrator
9. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik može:
   (a) poslati zahtjev za registraciju s željenom ulogom za koju se prijavljuje, a potrebni su korisničko ime, fotografija, lozinka, ime, prezime, broj mobitela i email adresa
   (b) Aktor 1 (inicijator) može:
   (c) prijaviti se za ulogu: 112 dispečer, doktor, vatrogasac i policajac
2. Spasilac može:
   (a) osvježiti podatak o dostupnosti za akcije (dostupan i spreman za akciju ili nije)
   (b) odazvati se na akciju spašavanja
3. Doktor(Spasilac) može:
   (a) biti osposobljen za vožnju motociklom ili kao putnik u kolima hitne pomoći

4. Vatrogasac(Spasilac) može:
   (a) biti osposobljen za vožnju autocisterni, autoljestvi, zapovjednog vozila i šumskog vozila
5. Policajac(Spasilac) može:
   (a) kretati se kao kontaktni policajac, pomoću motocikla, automobila ili oklopnog vozila
6. Voditelj stanice može:
   (a) definirati koji su spasioci dio njegove stanice
   (b) definirati na koji način su spasioci iz njegove stanice osposobljeni voditi spašavanje
   (c) odazvati se na akciju spašavanja
7. Dispečer može:
   (a) na temelju prijave otvarati akcije spašavanja s dostupnim informacijama i fotografijama
   (b) vidjeti broj dostupnih spasilaca po stanicama
   (c) poslati zahtjev za uključivanjem spasilaca u akciju spašavanja
   (d) prilikom slanja zahtjeva definirati na koji način bi spasilac trebao sudjelovati (auto, pješke..)
   (e) definirati razinu hitnosti zahtjeva
   (f) ako je potrebno spasioca ukloniti s akcije
   (g) ako je akcija spašavanja završila, označiti ju u sustavu kao gotovom
   (h) preko karte spasiocima pojedinačno zadati zadatke
   (i) pristupiti trenutnim pozicijama svih spasilaca zajedno s prikazom Voronojevog dijagrama
   (j) odabrati da se za izradu dijagrama koriste pozicije svih spasioca, ili svih dostupnih neaktivnih spasioca, ili aktivnih spasioca na određenoj akciji
8. Administrator može:
   (a) vidjeti popis svih registriranih korisnika i njihovih osobnih podataka
   (b) potvrditi zahtjeve za registraciju
   (c) kreirati stanice
   (d) mijenjati dodijeljena prava, osobne podatke i pripadnost stanici

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 -Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Registrirati se
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za se registrirati
  2. Korisnik unosi potrebne podatke
  3. Korisnik dobiva potvrdu da mu je registracija odobrena
- **Opis mogućih odstupanja:**

  2.a Odabir zauzetog korisničkog imena ili e-mail adrese, unos potrebnih podatak u krivom formatu
  1. Sustav obavještava korisnika o grešci
  2. Korisnik ispravlja podatke ili odustaje od registracije

UC2 -Prijava

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
  1. Unos korisničkog imena i lozinke
  2. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**

  1.a Unos krivog korisničkog imena ili lozinke
  1. Sustav obavještava korisnika o grešci
  2. Korisnik ispravlja podatke ili odustaje od registracije

UC3 -Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Vidjeti popis registriranih korisnika i njihove podatke
- **Sudionici:** Baza podataka

- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju pregledavanja korisnika
  2. Prikaže se lista registriranih korisnika s njihovim podacima

UC4 -Promjena prava i podataka registriranih korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Mijenjati prava, podatke i pripadnost stanici registriranih korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Administrator odabire željenog korisnika
  2. Administrator mijenja željene podatke i prava

UC5 -Obrada zahtjeva za registraciju

- **Glavni sudionik:** Administrator
- **Cilj:** Odobriti ili odbiti zahtjev za registraciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav, ima novih zahtjeva za registraciju
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju pregledavanja zahtjeva za registraciju
  2. Administrator potvrđuje ili odbija registraciju

UC6 -Kreiranje stanice

- **Glavni sudionik:** Administrator
- **Cilj:** Napraviti novu stanicu
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju stvaranja nove stanice
  2. Administrator unosi potrebne podatke i stvara stanicu

UC7 - Osvježavanje dostupnosti

- **Glavni sudionik:** Spasilac
- **Cilj:** Ažurirati dostupnost za akciju

- **Sudionici:** Baza podataka
- **Preduvjet:** Spasilac prijavljen
- **Opis osnovnog tijeka:**
  1. Spasilac odabire postavke računa
  2. Spasilac u svojim postavkama odabire opciju za dostupnost
  3. Odabirom jedne od dviju opcija se postavlja dostupnost spasioca
  4. Spasilac postaje dostupan ili nedostupan za akciju

UC8 - Dodavanje spasilaca

- **Glavni sudionik:** Voditelj
- **Cilj:** Dodati spasioca stanici
- **Sudionici:** Baza podataka
- **Preduvjet:** Voditelj prijavljen
- **Opis osnovnog tijeka:**
  1. Voditelj odabire opciju pregleda stanice
  2. Voditelj dobiva pregled stanice i spasilaca pripadnika stanice
  3. Voditelj odabire opciju dodavanja spasilaca
  4. Voditelj odabire spasioca iz baze registriranih spasilaca te mu dodjeljuje ulogu
- **Opis mogućih odstupanja:**

  4.a Nema slobodnih spasilaca
    * Sustav obavještava voditelja da nema dostupnih spasilaca

UC9 - Promjena uloge spasioca

- **Glavni sudionik:** Voditelj
- **Cilj:** Promijeniti ulogu spasioca
- **Sudionici:** Baza podataka
- **Preduvjet:** Voditelj prijavljen, spasilac član voditeljeve stanice
- **Opis osnovnog tijeka:**
  1. Voditelj odabire opciju pregleda stanice
  2. Voditelj dobiva pregled stanice i spasilaca pripadnika stanice
  3. Voditelj odabire željenog spasioca
  4. Voditelj odabire opciju mijenjanja uloge te ostvaruje tu promjenu
- **Opis mogućih odstupanja:**

  4.a Voditelj nije odabrao niti jednu ulogu

    * Sustav obavještava voditelja da nije odabrao ulogu te ga traži da odabere ulogu

UC10 - Otvaranje akcije

- **Glavni sudionik:** Dispečer
- **Cilj:** Otvoriti akciju i poslati zahtjeve spasiocima
- **Sudionici:** Baza podataka
- **Preduvjet:** Dispečer prijavljen
- **Opis osnovnog tijeka:**
    1. Dispečer odabire opciju otvaranja nove akcije
    2. Dispečer otvara akciju sa dostupnim informacijama i fotografijama
    3. Dispečer dobiva uvid o broju spasilaca po stanicama
    4. Dispečer šalje zahtjev pojedinim spasiocima sa definiranim načinom sudjelovanja i hitnosti
- **Opis mogućih odstupanja:**

    2.a Nema dostupnih spasilaca za akciju
        * Sustav obavještava dispečera o nedostupnosti spasilaca

UC11 - Završavanje akcije

- **Glavni sudionik:** Dispečer
- **Cilj:**Označiti akciju kao završenom
- **Sudionici:** Baza podataka
- **Preduvjet:** Akcija završena
- **Opis osnovnog tijeka:**
    1. Dispečer odabire završenu akciju
    2. Dispečer označava akciju kao završenom

UC12 - Uklanjanje spasilaca s akcije

- **Glavni sudionik:** Dispečer
- **Cilj:**Ukloniti spasioca s akcije
- **Sudionici:** Baza podataka
- **Preduvjet:** Spasilac trenutno na akciji
- **Opis osnovnog tijeka:**
    1. Dispečer otvara popis spasilaca na odabranoj akciji
    2. Dispečer odabire spasioca te ga miče s akcije
    3. Spasilac biva obaviješten o uklanjanju s akcije

UC13 - Odazivanje na akciju

- **Glavni sudionik:** Spasilac
- **Cilj:** Odazivanje na akciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Akcija pokrenuta i dispečer poslao zahtjev spasiocu
- **Opis osnovnog tijeka:**
  1. Spasilac dobiva zahtjev od dispečera za uključenje u akciju
  2. Spasilac potvrđuje zahtjev ili ga odbija
  3. Nakon potvrde vanjski servis izračunava rutu do lokacije

UC14 - Zadavanje pojedinačnih zadataka

- **Glavni sudionik:** Dispečer
- **Cilj:** Zadati pojedinačan zadatak spasiocu
- **Sudionici:** Baza podataka, Spasilac
- **Preduvjet:** Spasilac dostupan za akciju
- **Opis osnovnog tijeka:**
  1. Dispečer otvara kartu
  2. Na karti odabire dostupnog spasioca
  3. Dispečer upisuje informacije o zadatku, moguć je zahtjev prolaska određenom rutom do lokacije
  4. Dispečer po potrebi dodaje i dodatni komentar
  5. Spasilac dobiva informacije o zadatku te izračunatu rutu od strane vanjskog servisa
- **Opis mogućih odstupanja:**

  2.a Nema dostupnih spasilaca za zadatak
  - Sustav obavještava dispečera porukom o nedostupnosti spasilaca

UC15 - Odabir moda izračuna dijagrama

- **Glavni sudionik:** Dispečer
- **Cilj:** Odabrati način izračuna Vornojevog dijagrama
- **Sudionici:** Baza podataka, vanjski servis
- **Preduvjet:** Dispečer prijavljen u aplikaciju
- **Opis osnovnog tijeka:**
  1. DIspečer odabire opciju pottavki karte
  2. Dispečer u postavkama karte odabire opciju ”način izračuna”

3. Dispečer odabire jednu od tri opcije:
   (a) Uključiti sve spasioce
   (b) Uključiti sve dostupne neaktivne spasioce
   (c) Uključiti sve spasioce aktivne na određenoj akciji
4. Vanjski servis obavlja izračun na temelju odabrane opcije

UC16 - Prikaz spasilaca

- **Glavni sudionik:** Dispečer
- **Cilj:** Adekvatno prikazati spasioce na karti
- **Sudionici:** Baza podataka
- **Preduvjet:** Dispečer prijavljen u aplikaciju
- **Opis osnovnog tijeka:**
  1. Ovisno o ulozi spasioca u akciji, sustav spasioca na karti treba prikazati adekvatnom ikonom
  2. Prilikom odabira karte, dispečeru se spasioci koji sudjeluju u akciji prikazuju adekvatnim ikonama

UC17 -Pregled zadataka

- **Glavni sudionik:** Spasilac
- **Cilj:** Pregled zadataka koje treba obaviti
- **Sudionici:** Baza podataka
- **Preduvjet:** Spasilac je prijavljen u sustav, dispečer je zadao barem jedan zadatak
- **Opis osnovnog tijeka:**
  1. Spasilac pristupa karti na kojoj mu se prikazuju zadaci
  2. moguć je i odabir prikaza popisa svih zadataka

UC18 -Pregled pozicije ostalih spasilaca

- **Glavni sudionik:** Spasilac
- **Cilj:** Pregledati trenutnu poziciju ostalih spasilaca aktivnih na istoj akciji
- **Sudionici:** Baza podataka
- **Preduvjet:** Spasilac je prijavljen u sustav, ima ostalih spasilaca na istoj akciji
- **Opis osnovnog tijeka:**
  1. Spasilac pristupa karti na kojoj mu se prikazuje lokacija ostalh spasioca

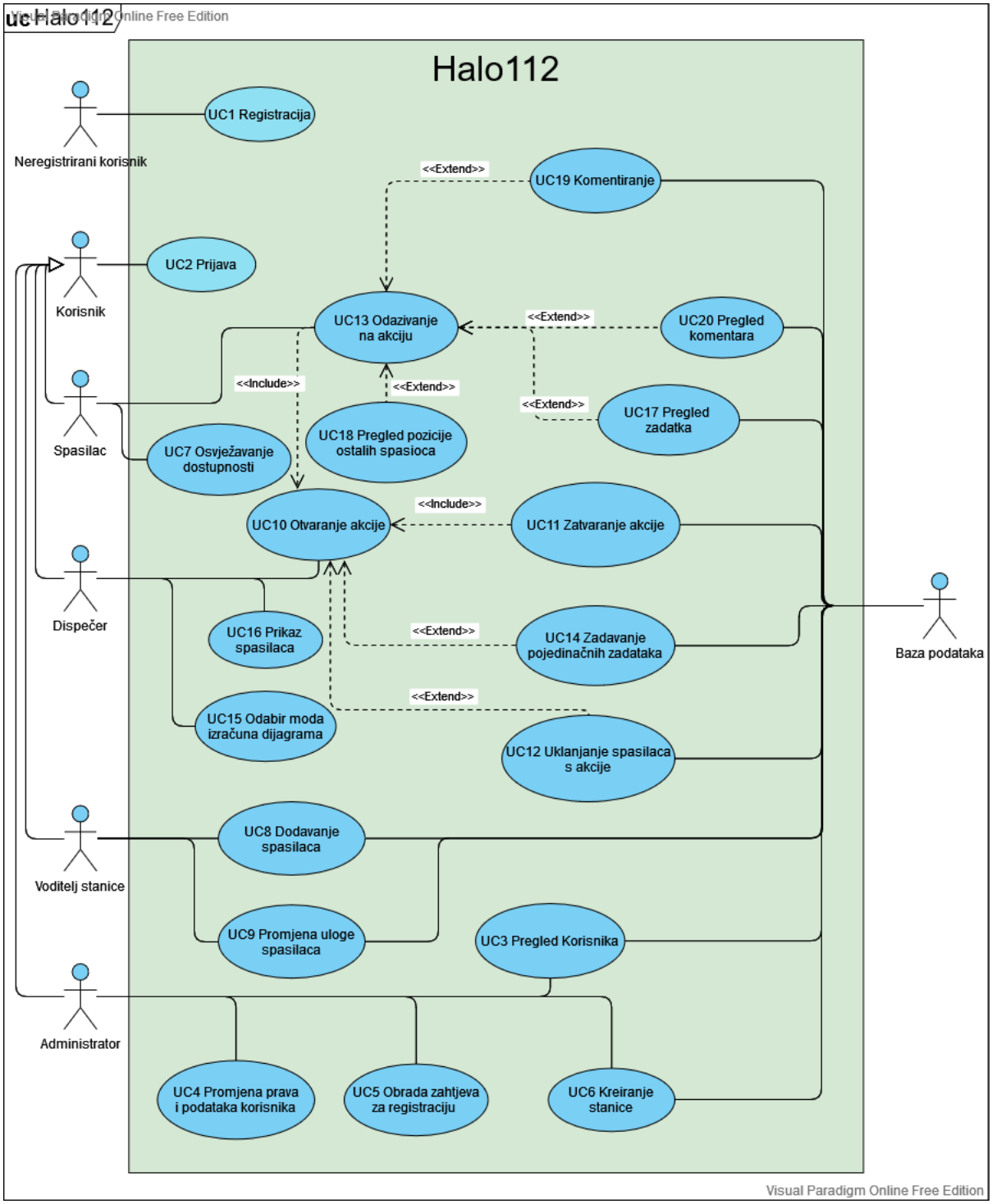
UC19 -Komentiranje

- **Glavni sudionik:** Spasilac
- **Cilj:** Ostaviti komentar na karti za ostale sudionike u akciji
- **Sudionici:** Baza podataka
- **Preduvjet:** Spasilac je prijavljen u sustav i sudjeluje u akciji
- **Opis osnovnog tijeka:**
    1. Spasilac pristupa karti te odabire akciju
    2. Spasilac upisuje svoj komentar o akciji

UC20 -Pregled komentara

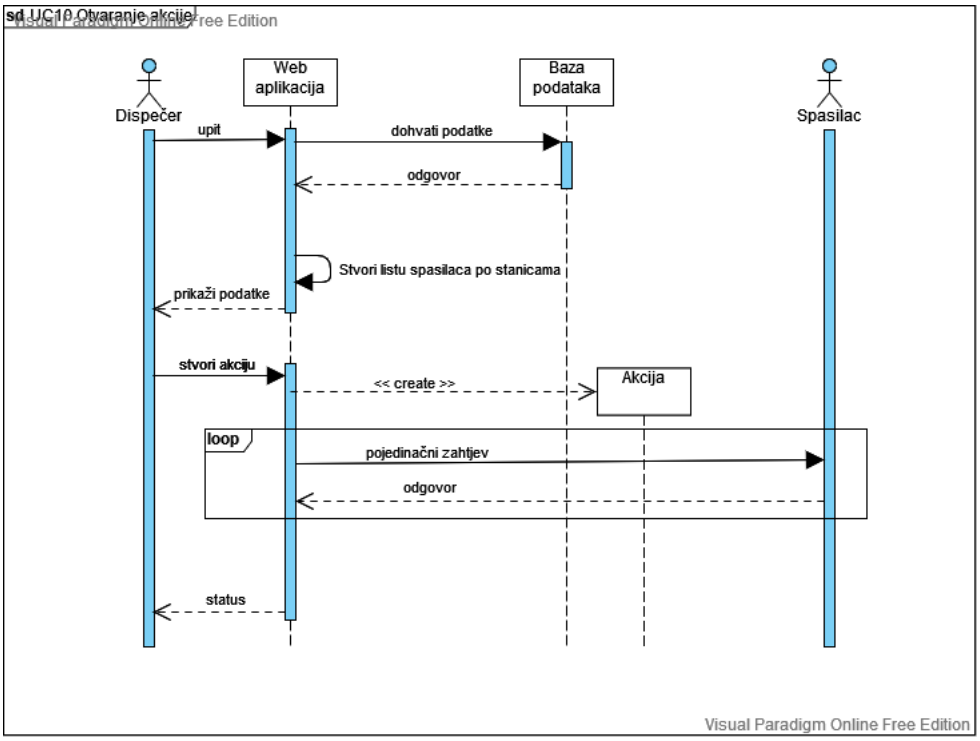
- **Glavni sudionik:** Spasilac
- **Cilj:** Pregledati komentar ostavljen na karti
- **Sudionici:** Baza podataka
- **Preduvjet:** Spasilac je prijavljen u sustav, sudjeluje u akciji i akcija ima ostavljenih komentara
- **Opis osnovnog tijeka:**
    1. Spasilac pristupa karti te odabire akciju
    2. Spasilac dobiva uvid u informacije akcije te ostavljene komentare

Dijagrami obrazaca uporabe



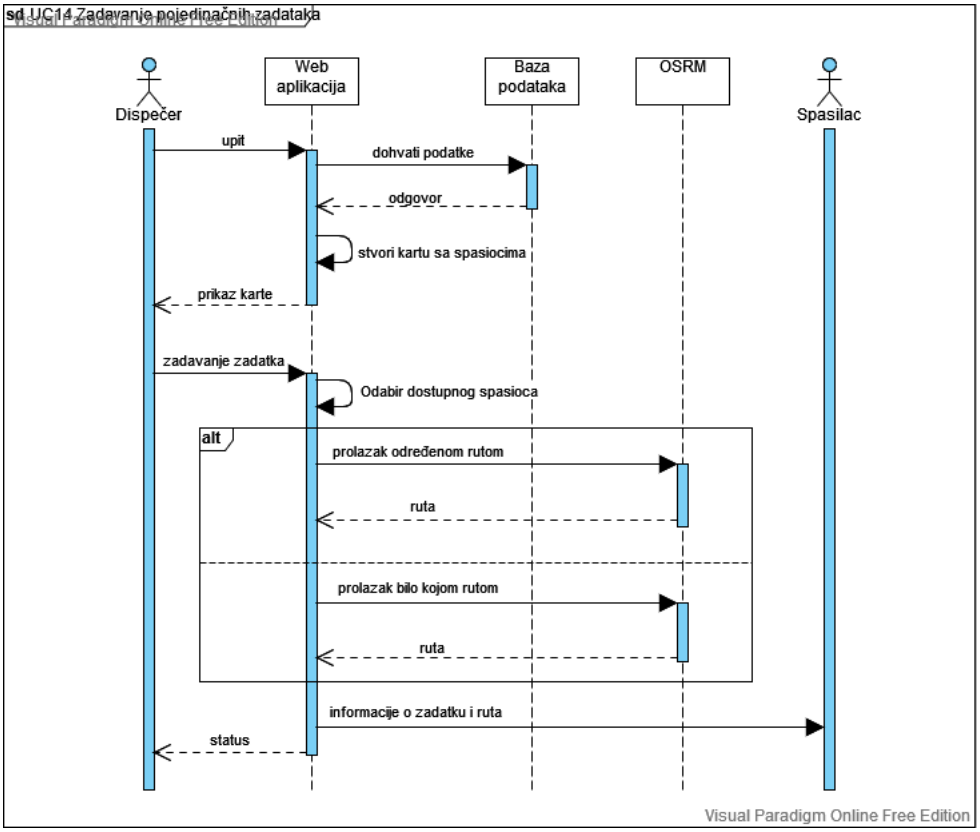
Slika 3.1: Dijagram obrazaca uporabe

3.1.2 Sekvencijski dijagrami



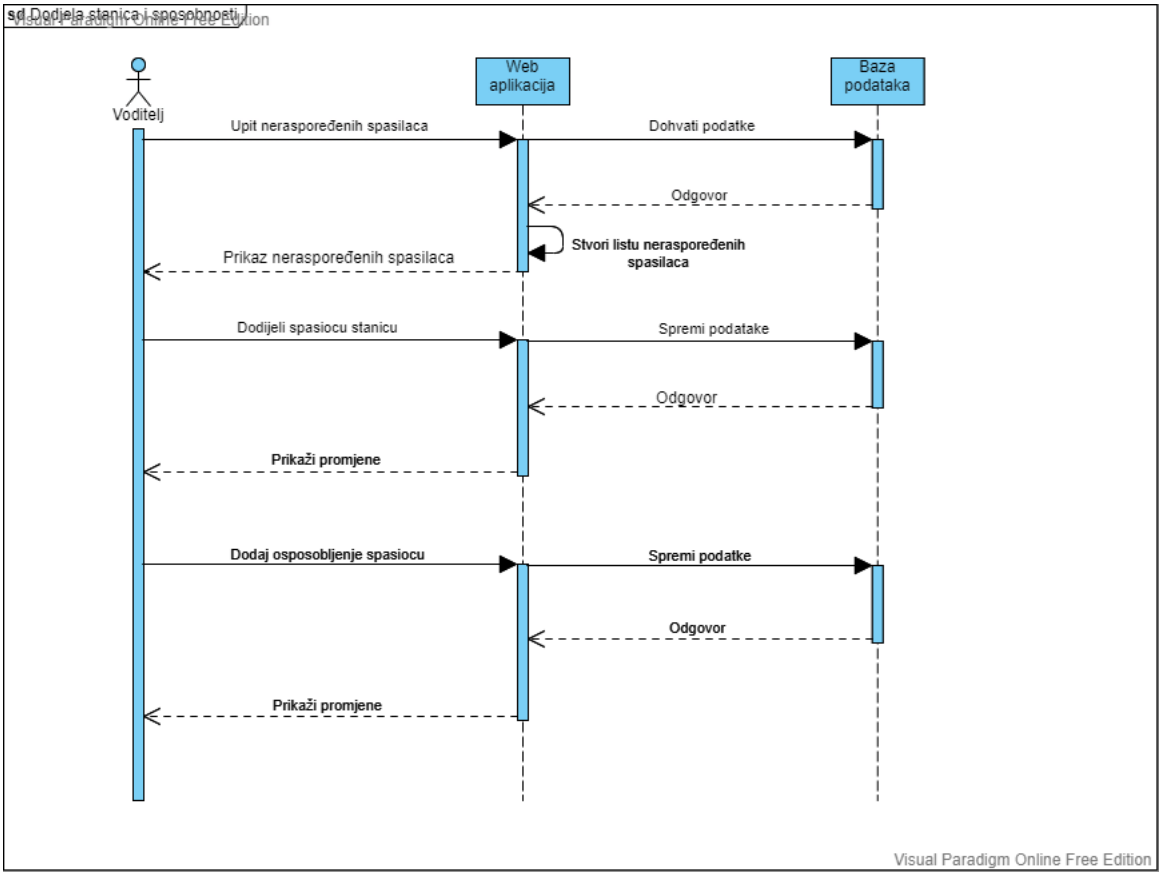
Slika 3.2: Sekvencijski dijagram - Otvaranje akcije (UC10)

Prikaz postupka u kojem dispečer stvara akciju i šalje zahtjeve pojedinim spasiocima. Prije stvaranja akcije dispečeru se prikazuje popis broja dostupnih spasilaca po stanicama.



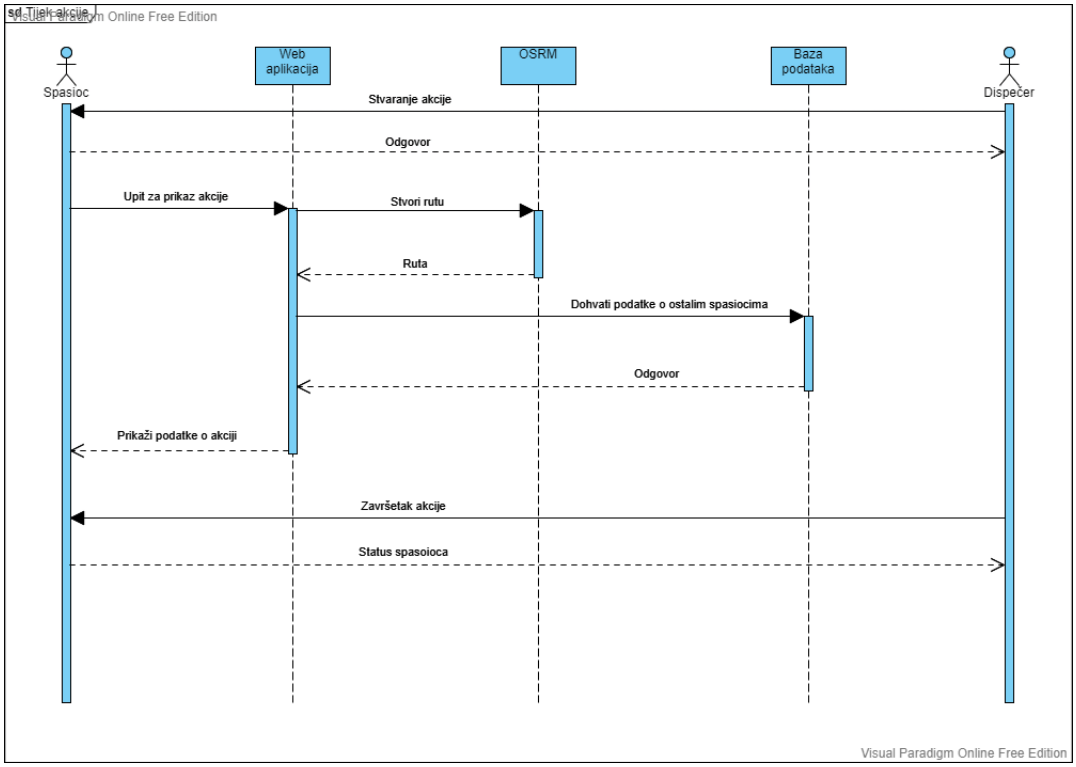
Slika 3.3: Sekvencijski dijagram - Zadavanje pojedinačnih zadataka (UC14)

Prikaz postupka u kojem dispečer pomoću karte spasilaca zadaje pojedinačni zadatak određenom spasiocu. Dispečer može odrediti rutu kojom spasilac prolazi.



Slika 3.4: Sekvencijski dijagram - Dodjela stanica i sposobnosti

Prikaz postupka u kojem voditelj dodaje neraspoređene spasioce u svoju postaju i dodjeljuje im za što su osposobljeni



Slika 3.5: Sekvencijski dijagram - Tijek akcije

Prikaz postupka dispečera i spasioca od početka do kraja akcije. Kada je spasiocu dodijeljena akcija, ili on stvara upit za prikaz karte i ostalih sudionika u akciji ili odbija dodijeljenu akciju. kada se spasioc vrati s akcije, dispečer ju zatvara i status spasioca se vraća na dostupan.

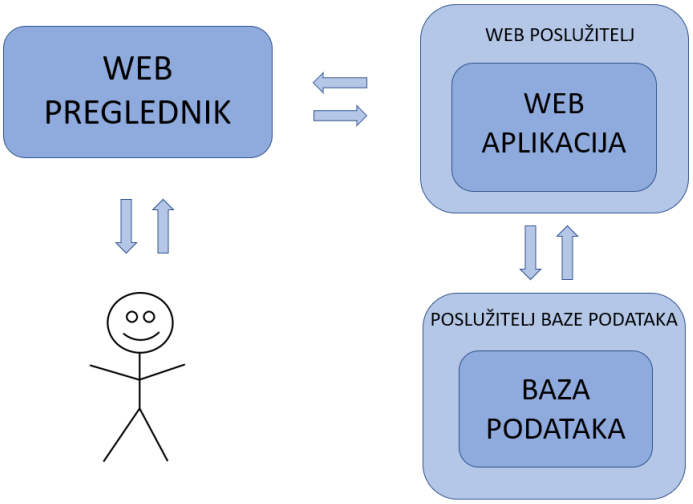
3.2 Ostali zahtjevi

- Sustav treba podržavati spajanje više korisnika u vremenu
- Sustav treba biti svima jednostavan za korištenje uz minimalne upute te bez potrebnog korisničkog iskustva
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
- Vrijeme odziva treba biti što prije moguće (u nekoliko sekundi)
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orijentirane jezike
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Promjene i nadogradnje sustava ne smiju poremetiti rad sustava
- Sustav treba imati ograničen broj spasilaca
- Mjerna jedinica za udaljenost je metar

4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na tri podsustava:

- Web poslužitelj
- Web aplikacija
- Baza podataka



Slika 4.1: Arhitektura sustava

Web preglednik je program koji služi za pregledavanje web-stranica. Uloga web preglednika je da kada korisnik zatraži određenu stranicu dohvati potreban sadržaj s web poslužitelja i prikaže ga korisniku.

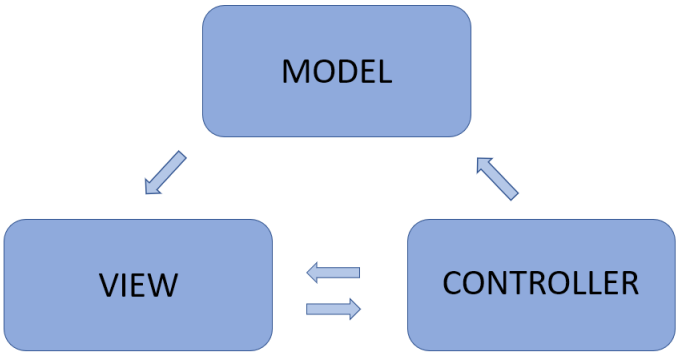
Web poslužitelj omogućuju komunikaciju klijenta s aplikacijom tako da proslijedi zahtjev klijenta aplikaciji i odgovor kojeg je stvorila aplikacija natrag klijentu. Komunikacija se odvija preko HTTP-a, internetskog protokola aplikacijskog sloja koji služi za prijenos dokumenata poput onih pisanih u HTML-u.

Web aplikacija omogućuje klijentu postavljanje zahtjeva na koje ona stvara odgovore. Ovisno o zahtjevu može pristupati bazi podataka te onda generira odgovor kojeg šalje u obliku HTML dokumenta pregledniku kojeg koristi klijent.

Poslužitelj baze podataka omogućuje web aplikaciji dohvaćanje sadržaja iz baze podataka i zapisivanje novih stavki u bazu podataka.

Za izradu frontenda korišten je programski jezik JavaScript i radni okvir React. React je JavaScript biblioteka koja služi za izradu korisničkih sučelja. Za izradu backenda korišten je programski jezik Java i radni okvir Spring Boot koji omogućuje brži i efikasniji razvoj aplikacije. Korišteno razvojno okruženje je Microsoft Visual Studio. Arhitektura sustava temeljiti će se na MVC (Model-View-Controller) konceptu.

Svrha MVC obrasca je odvajanje programske logike u tri međusobno povezana dijela, čime se poštuje načelo odvajanja briga i olakšava razvoj aplikacije.



Slika 4.2: MVC dijagram

MVC se sastoji od:

- Model - centralna komponenta koja upravlja podacima, logikom i pravilima aplikacije
- View - prikazuje podatke modela i šalje radnje korisnika kontroleru
- Controller - prima radnje korisnika na temelju kojih vrši daljnu interakciju s modelom i pogledom

4.1 Baza podataka

Za potrebe našeg sustava koristit ćemo relacijsku bazu podataka koja svojom strukturom olakšava modeliranje stvarnog svijeta. Gradivna jedinka baze je relacija, odnosno tablica koja je definirana svojim imenom i skupom atributa. Zadaća baze podataka je brza i jednostavna pohrana, izmjena i dohvat podataka za daljnju obradu. Baza podataka se sastoji od idućih entiteta:

- User
- Responder

- Station
- Call
- CallRequest
- Photo
- Task
- CallComments

4.1.1 Opis tablica

User Ovaj entitet sadrži općenite informacije o svim korisnicima aplikacije. Sadrži atribute: Username, PhotoURL, PasswordHash, Name, Surname, PhoneNumber, eMail, Role i Confirmed. Ovaj entitet je u vezi *One-to-one* s entitetom Responder preko atributa Username, u vezi *One-to-many* s entitetom CallRequest preko atributa DispatcherUsername te je u vezi *One-to-many* s entitetom Task preko atributa DispatcherUsername.

Username		
Username	VARCHAR	jedinstveno korisničko ime
PhotoURL	VARCHAR	url fotografije
PasswordHash	VARCHAR	hash lozinke
Name	VARCHAR	Ime korisnika
Surname	VARCHAR	Prezime korisnika
eMail	VARCHAR	e-mail korisnika
PhoneNumber	VARCHAR	Broj mobitela korisnika
Role	VARCHAR	Uloga korisnika
Confirmed	BIT	Status registracije korisnika

Responder Ovaj entitet sadrži sve važne informacije o spasiocima. Sadrži atribute: Username, Qualification, LocationLatitude, LocationLongitude, Availability, CallName i StationName. Ovaj entitet je u vezi *One-to-one* sa entitetom User preko atributa Username, u vezi *One-to-many* sa entitetom CallRequest preko atributa ResponderUsername, u vezi *One-to-one* s entitetom Station preko atributa StationName, u vezi *One-to-one* s entitetom Station preko atributa StationChief, u vezi

One-to-Many s entitetom Task preko atributa ResponderUsername, u vezi *One-to-Many* s entitetom CallComents preko atributa Username te u vezi *One-to-one* s entitetom Call preko atributa CallName.

Responder		
Username	VARCHAR	Korisničko ime spasioca
Qualification	VARCHAR	Osposobljenje spasioca
Location Latitutude	VARCHAR	Geografska širina lokacije spasioca
Location Longitude	VARCHAR	Geografska dužina lokacije spasioca
Availability	BIT	Status zauzetosti spasioca
CallName	VARCHAR	Ime akcije u kojoj spasilac trenutno sudjeluje
StationName	VARCHAR	Ime stanice kojoj spasilac pripada

Call Ovaj entitet sadrži sve važne informacije o akciji. Sadrži atribute: CallName, Address, Finished i Description. Ovaj entitet je u vezi *One-to-one* s entitetom Responder preko atributa CallName, u vezi *One-to-many* s entitetom Photo preko atributa CallName, u vezi *One-to-many* s entitetom CallComments preko atributa CallName te u vezi *One-to-many* s entitetom CallRequest preko atributa CallName.

Call		
CallName	VARCHAR	Ime akcije
Address	VARCHAR	Adresa
Finished	BIT	Oznaka završetka akcije
Description	VARCHAR	Kratak opis akcije

CallRequest Ovaj entitet sadrži sve važne informacije o pozivu na akciju pojedinačnom spasiocu. Sadrži atribute: CallName, ResponderUsername, DispatcherUsername, ResponderResponse, ResponderRole, Urgency i Comment. Ovaj en-

titet je u vezi *Many-to-one* s entitetom Responder preko atributa ResponderUsername, u vezi *Many-to-many* s entitetom User preko atributa DispatcherUsername te u vezi *Many-to-one* s entitetom Call preko atributa CallName.

CallRequest		
CallName	VARCHAR	Ime akcije za koju se šalje poziv
Respoder Username	VARCHAR	Korisničko ime spasioca kojem je poslan poziv
Dispatcher Username	VARCHAR	Korisničko ime dispečera koji je poslao poziv
Responder Response	BIT	Prihvat poziva ili odbijanje
Responder Role	VARCHAR	Način sudjelovanja spasioca
Urgency	INT	Razina hitnosti
Comment	VARCHAR	Komentar dispečera

Task Ovaj entitet sadrži sve važne informacije o zadatku kojeg dispečer zadaje spasiocima. Sadrži atribute: IDTask, Comment, Description, Address, ResponderUsername i DispatcherUsername. Ovaj entitet je u vezi *Many-to-one* s entitetom User preko atributa DispatcherUsername te u vezi *Many-to-one* s entitetom Responder preko atributa ResponderUsername.

Task		
IDTask	INT	Identifikator zadatka
Comment	VARCHAR	Komentar kojeg može dati dispečer
Description	VARCHAR	Opis zadatka
Address	VARCHAR	Adresa na koju treba doći
Responder Username	VARCHAR	Korisničko ime spasioca koji je dobio zadatak

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Task		
Dispatcher Username	VARCHAR	Korisničko ime dispečera koji je zadao zadatak

Photo Ovaj entitet sadrži fotografije akcija. Sadrži atribute: PhotoURL i CallName. Ovaj entitet je u vezi *Many-to-one* s entitetom Call preko atributa CallName.

Photo		
PhotoURL	VARCHAR	Url fotografije akcije
CallName	VARCHAR	Ime akcije kojoj fotografija pripada

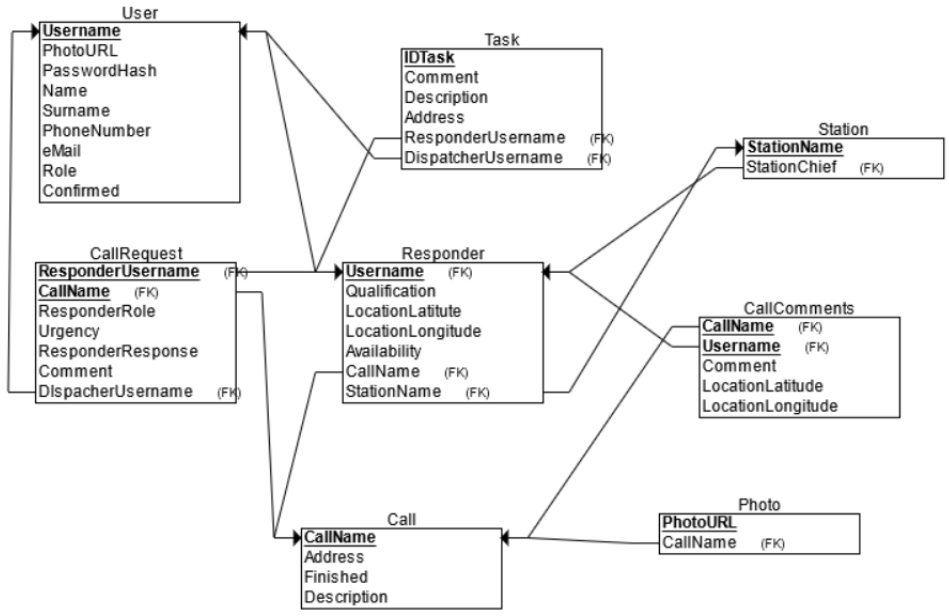
CallComments Ovaj entitet sadrži sve važne informacije vezane uz mogućnost spasilaca da komentiraju akcije. Sadrži atribute: CallName, Username, Comment, Locationlatitude i LocationLongitude. Ovaj entitet je u vezi *Many-to-one* s entitetom Call preko atributa CallName te u vezi *Many-to-one* s entitetom Responder preko atributa Username.

CallComments		
CallName	VARCHAR	Ime akcije za koju se ostavlja komentar
Username	VARCHAR	Korisnik koji ostavlja komentar
Comment	VARCHAR	Komentar kojeg je korisnik ostavio
Location Latitude	VARCHAR	Geografska širina lokacije komentara
Location Longitude	VARCHAR	Geografska dužina lokacije komentara

Station Ovaj entitet sadrži sve važne informacije vezane uz stanicu. Sadrži atribute: StationName i StationChief. Ovaj entitet je u vezi *One-to-one* s entitetom Responder preko atributa StationName te u vezi *One-to-one* s entitetom Responder preko atributa StationChief.

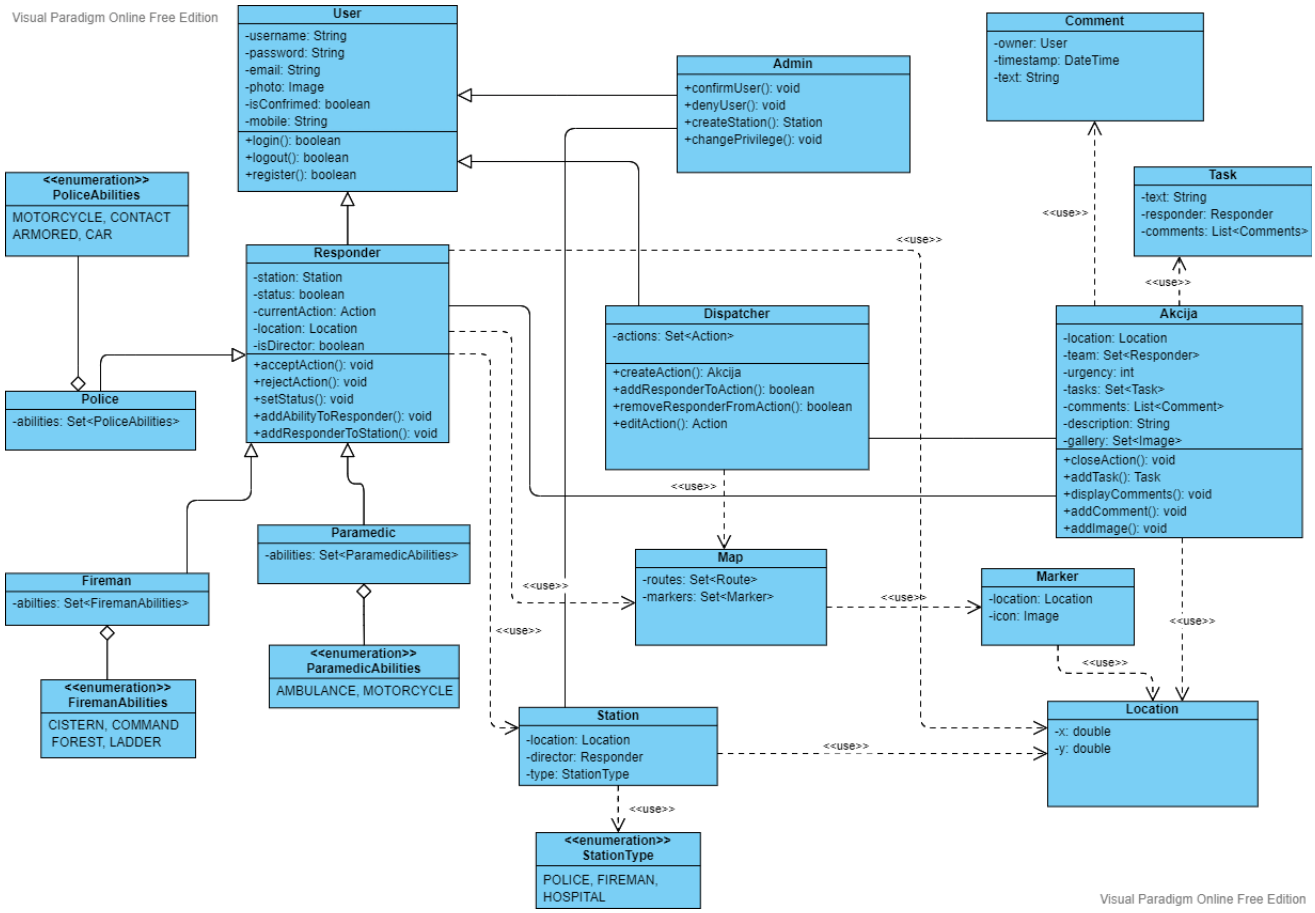
Station		
StationName	VARCHAR	Naziv stanice
StationChief	VARCHAR	Korisničko ime voditelja stanice

4.1.2 Dijagram baze podataka



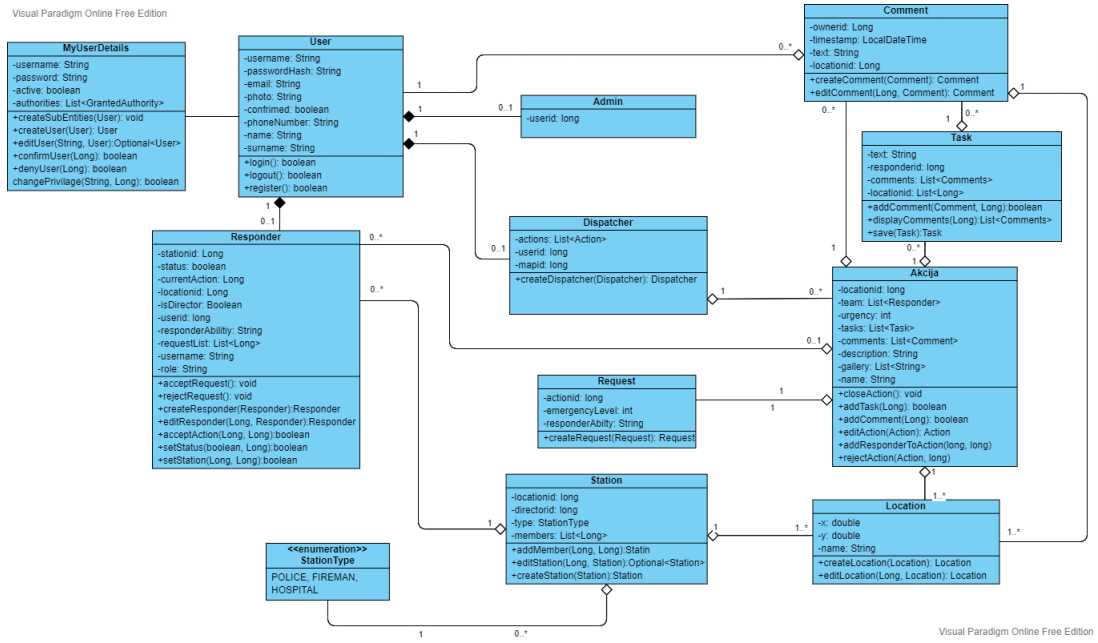
Slika 4.3: Dijagram baze podataka

4.2 Dijagram razreda



Slika 4.4: Dijagram razreda

Planirani dijagram razreda za aplikaciju Halo112. User je glavna klasa koju ostale klase extendaju.

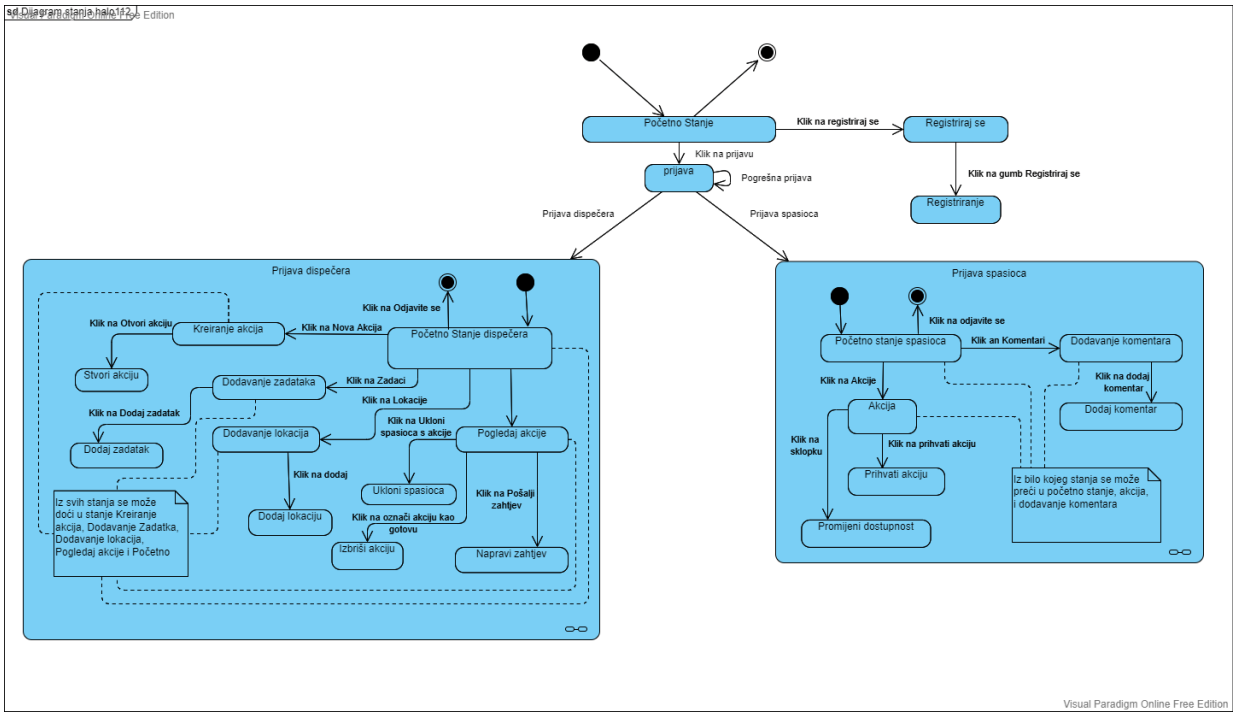


Slika 4.5: Implementacijski dijagram razreda

Konačni dijagram razreda za aplikaciju Halo112. Enumovi su zamijenjeni stringom i responder ima string koji označava njegovu ulogu. Svi razredi imaju svoj id koji nije prikazan na dijagramu.

4.3 Dijagram stanja

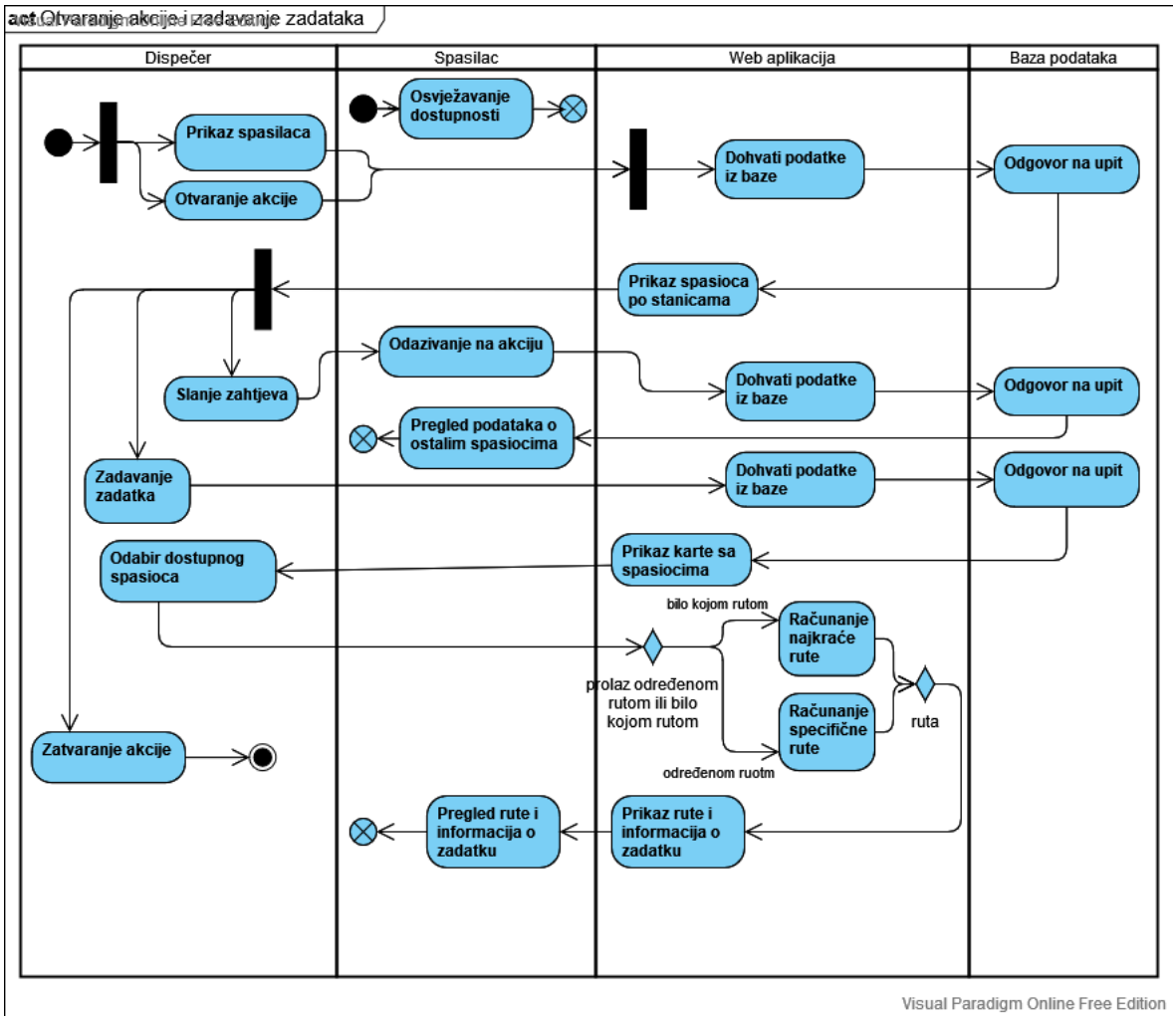
dio 2. revizije



Slika 4.6: Dijagram stanja

Prikaz mogućih stanja dispečera i spasioca. Sustav počinje od logina i prelazi u jednu od dvije grane ovisno o tome tko se prijavljuje.

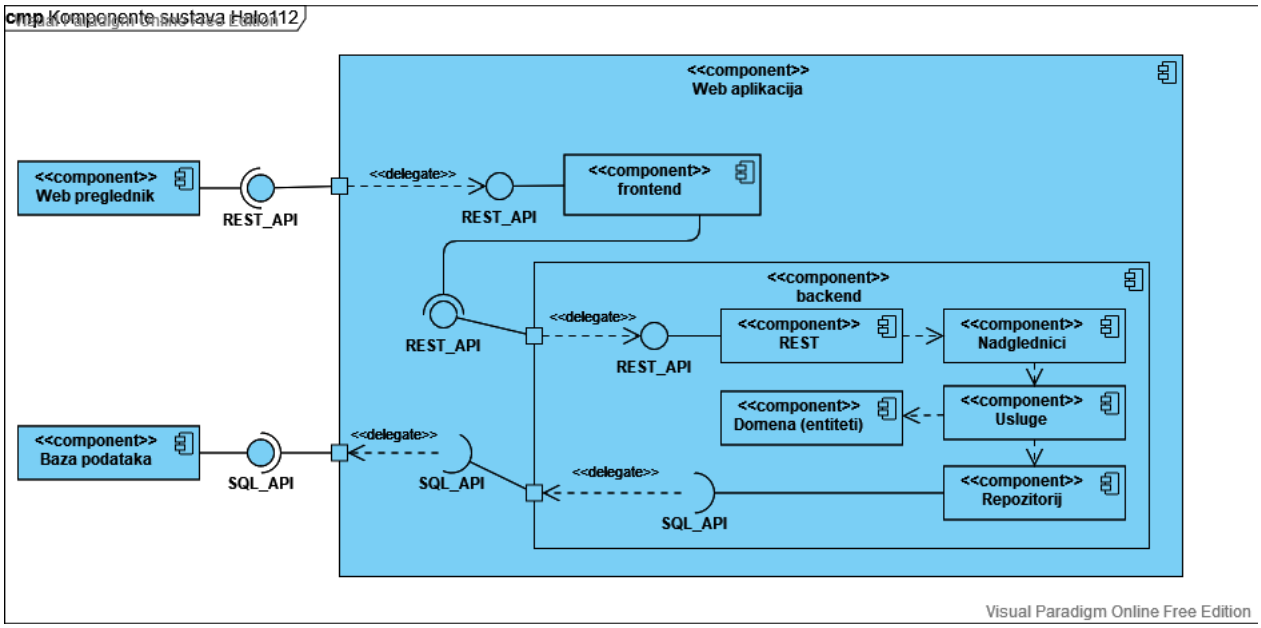
4.4 Dijagram aktivnosti



Slika 4.7: Dijagram aktivnosti

Prikaz akcije otvaranja akcije za spasioce i zadavanja zadatka. Dispečer otvara i zatvara akcije, te zadaje zadatke. Spasioc osvježava svoju dostupnost i odaziva se na akcije. Dispečeru se prikazuju spasioci po stanicama i na karti. Spasiocu se prikazuju informacije o zadatku i ostalim spasiocima.

4.5 Dijagram komponenti



Slika 4.8: Dijagram komponenti

Prikaz komponenti i strukture čitavog sustava. Korinik komunicira sa sustavom pomoću HTTP zahtjeva. Frontend i backend također komuniciraju HTTP porukama. Razmjenu HTTP poruka omogućuje REST API. Backend se sastoji od tri sloja. REST nadglednici komuniciraju s frontendom, a JPA repozitorij pohranjuje entitete sustava u bazu podataka korištenjem SQL API-a. Na sloju usluge je ostvarena temeljna funkcionalnost. U domeni su opisani svi entiteti.

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

MiKTeX

- **Opis:** MiKTeX je besplatna distribucija otvorenog koda TeX/LaTeX sustava za Microsoft Windows.
- **Mjesto primjene:** Projektna dokumentacija
- **Poveznica:** https://miktex.org/

Visual paradigm

- **Opis:** Visual paradigm je UML CASE alat koji se koristi za stvaranje UML dijagrama.
- **Mjesto primjene:** Projektna dokumentacija
- **Poveznica:** https://www.visual-paradigm.com/

Java Spring Framework

- **Opis:** Java Spring Framework je razvojni okvir otvorenog koda za stvaranje samostalnih aplikacija koje se pokreću na Java Virtual Machine-u (JVM).
- **Mjesto primjene:** Web aplikacija - backend
- **Poveznica:** https://spring.io/projects/spring-framework/

React

- **Opis:** React je besplatna JavaScript biblioteka otvorenog koda za izgradnju korisničkih sučelja.
- **Mjesto primjene:** Web aplikacija - frontend
- **Poveznica:** https://reactjs.org/

Leaflet

- **Opis:** Leaflet je JavaScript biblioteka otvorenog koda koja se koristi za izradu web aplikacija s kartama.
- **Mjesto primjene:** Web aplikacija - frontend

- **Poveznica:** http://leafletjs.com/

Heroku

- **Opis:** Heroku je cloud platforma kao usluga (PaaS) koja podržava nekoliko programskih jezika.
- **Mjesto primjene:** Puštanje aplikacije u pogon
- **Poveznica:** https://www.heroku.com/

Selenium

- **Opis:** Selenium je projekt otvorenog koda za niz alata i knjižnica čiji je cilj podrška automatizaciji web preglednika. Koristi se za testiranje.
- **Mjesto primjene:** Ispitivanje programskog rješenja
- **Poveznica:** https://www.selenium.dev/

JUnit

- **Opis:** JUnit je okvir za programski jezik Java koji služi za ispitivanje jedinica.
- **Mjesto primjene:** Ispitivanje programskog rješenja
- **Poveznica:** https://junit.org/

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

1. Test stvara novog korisnika i provjerava postoji li korisnik s istim imenom u repozitoriju, zatim ispituje potvrđivanje tog korisnika.

```
public void testUser() throws Exception {
    userService.createUser(new User(null, "test1", "", "secret",
        "Test", "Test", "", "", "policeman", false));

    assertTrue(userRepo.findUserByUserName("test1").isPresent());
    User user = userRepo.findUserByUserName("test1").get();
    assertFalse(user.isConfirmed());
    assertTrue(userService.confirmUser(user.getId()));
    user = userRepo.findById(user.getId()).get();
    assertTrue(user.isConfirmed());
}
```

2. Test stvara potvrđene korisnike s ulogama vatrogasca i dispečera, zatim provjerava jesu li stvoreni ostali potrebni entiteti (Responder, Dispatcher) i jesu li ispravno povezani.

```
public void testRole() throws Exception {
    User user1 = userService.createUser(new User(null, "test2", "",
        "secret", "Test", "Test", "", "", "fireman", true));
    Responder responder =
        responderService.findByUserId(user1.getId()).get();
    assertEquals("test2", responder.getUserName());
    assertEquals(responder.getUser_id(), user1.getId());

    User user2 = userService.createUser(new User(null, "test3", "",
        "secret", "Test", "Test", "", "", "dispatcher", true));
    assertEquals(user2.getId(),
        dispatcherService.listAll().get(0).getUser());
}
```

3. Test otvara akciju, provjerava ispravnost dodavanja spasioca i zatvaranja akcije.

```
public void testAction() throws Exception {
   User user = userService.createUser(new User(null, "test4", "",
       "secret", "Test", "Test", "", "", "doctor", true));
   Responder responder =
       responderService.findByUserId(user.getId()).get();

   Action act = new Action();
   act.setName("test action");
   act.setDescription("test...");
   act.setTeam(new ArrayList<Responder>());
   act.setTasks(new ArrayList<Task>());
   actionService.createAction(act);

   assertTrue(actionService.addResponderToAction(responder.getId(),
       act.getId()));
   assertTrue(actionService.displayResponders(act.getId()).contains(responder));

   assertTrue(actionService.closeAction(act.getId()));
}
```

4. Test stvara novu stanicu i dodaje joj dva spasioca (voditelja i člana), zatim provjerava uspješnost tih funkcionalnosti.

```
public void testStation() throws Exception {
   User user1 = userService.createUser(new User(null, "test5", "",
       "secret", "Test", "Test", "", "", "doctor", true));
   Responder director =
       responderService.findByUserId(user1.getId()).get();
   User user2 = userService.createUser(new User(null, "test6", "",
       "secret", "Test", "Test", "", "", "doctor", true));
   Responder responder =
       responderService.findByUserId(user2.getId()).get();

   Station station = stationService.createStation(new Station("Test
       station", director.getId(),
   locationService.createLocation(new Location()).getId(),
       StationType.HOSPITAL));
   station.setMembers(new ArrayList<Long>());
```

```
    station = stationService.addMember(station.getId(),
        responder.getId());

    assertTrue(station.getMembers().contains(responder.getId()));
    assertEquals(station.getDirector_id(), director.getId());
}
```

5. Test stvara novu akciju, dodaje joj zadatak i zadatku dodaje komentar. Test ispituje navedene funkcionalnosti pomoću povratnih vrijednosti.

```
public void testTask() throws Exception {
    User user = userService.createUser(new User(null, "test7", "",
        "secret", "Test", "Test", "", "", "policeman", true));

    Action act = new Action();
    act.setName("test action");
    act.setDescription("test...");
    act.setTeam(new ArrayList<Responder>());
    act.setTasks(new ArrayList<Task>());
    actionService.createAction(act);

    Task task = taskService.save(new Task(null, "test task",
        user.getId(), new ArrayList<Long>(), new ArrayList<Comment>()));
    assertEquals(user.getId(), task.getResponder_id());
    assertTrue(actionService.addTask(task.getId(), act.getId()));
    assertTrue(taskService.addComment(new Comment(), task.getId()));
}
```

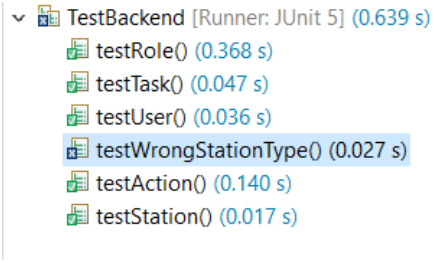
6. Test dodaje spasioca stanici kojoj ne pripada i pritom očekuje IllegalArgumentException. Funkcionalnost nije implementirana.

```
public void testWrongStationType() throws Exception {
    User user1 = userService.createUser(new User(null, "test8", "",
        "secret", "Test", "Test", "", "", "policeman", true));
    Responder director =
        responderService.findByUserId(user1.getId()).get();
    User user2 = userService.createUser(new User(null, "test9", "",
        "secret", "Test", "Test", "", "", "doctor", true));
    Responder responder =
```

```
        responderService.findByUserId(user2.getId()).get();

    Station station = stationService.createStation(new Station("Test
        station 2", director.getId(),
    locationService.createLocation(new Location()).getId(),
        StationType.POLICE));
    station.setMembers(new ArrayList<Long>());

    assertThrows(IllegalArgumentException.class, () ->
        stationService.addMember(station.getId(), responder.getId()));
}
```



Slika 5.1: Rezultati testova

testWrongStationType() je jedini test koji ne prolazi uspješno, jer funkcionalnost koju ispituje nije implementirana.

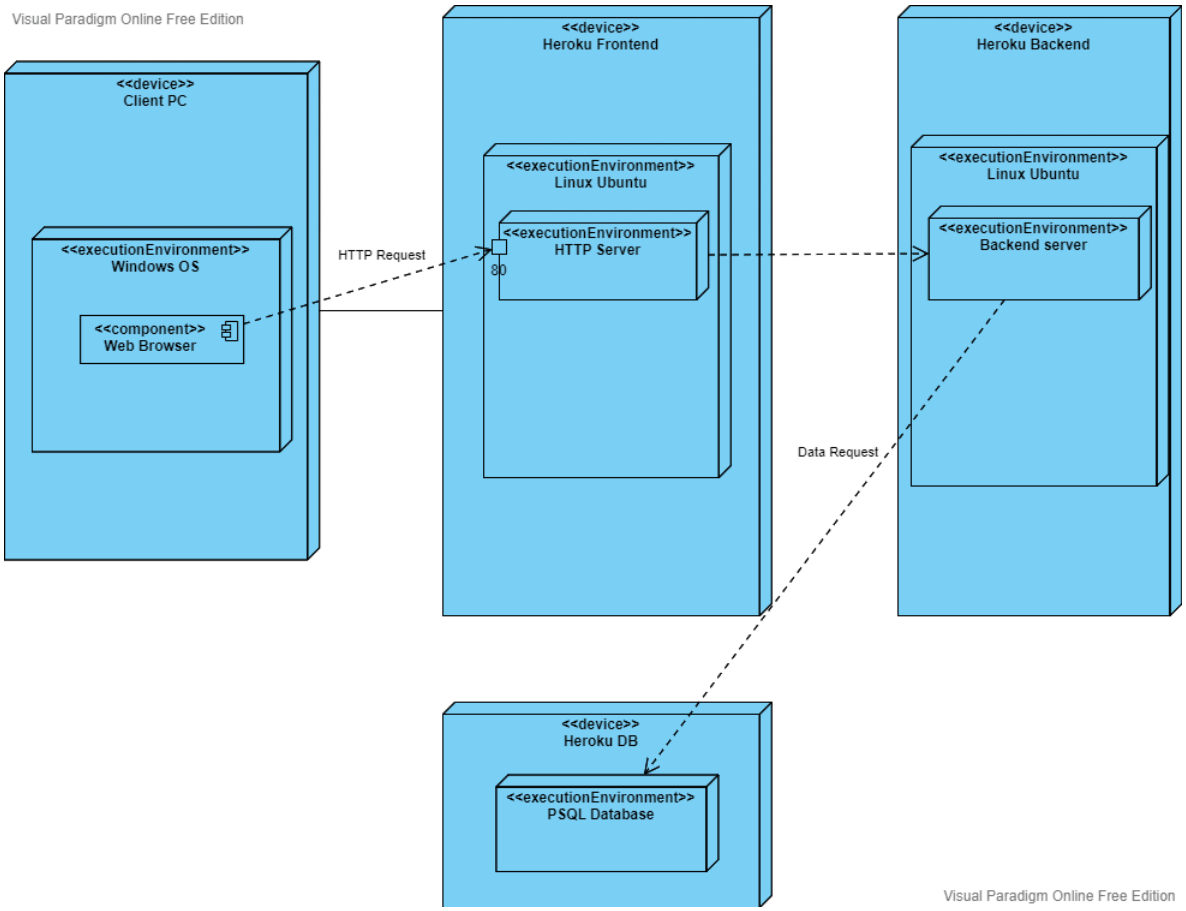
5.2.2 Ispitivanje sustava

Za pokretanje Selenium testova, potrebno je imati ekstenziju na pregledniku za pokretanje testova. Testovi se pokreću redom označenim brojevima.

- 1RegisterTest - Registrira 2 nova korisnika.
- 2AdminConfirmAndCreateStation - Potvrđuje dva nova korisnika i stvara stanicu.
- 3VoditeljTest - Prijava voditelja i dodavanje drugog korisnika u stanicu.
- 4DispatcherAction - Prijava dispatchera i stvaranje i zadavanje akcije.
- LoginFAIL - Neuspješna prijava u sustav.

5.3 Dijagram razmještaja

dio 2. revizije



Slika 5.2: Dijagram razmještaja

Prikaz dijagrama razmještaja. U sustavu postoje klijent, frontend server, backend server i baza podataka. Ovo je bio planirani razmještaj projekta međutim aplikacija radi samo lokalno. Na heroku je postavljena aplikacija međutim funkcionalnost prijave ne radi. Također koristi se lokalna baza podataka umjesto psql servera.

5.4 Upute za puštanje u pogon

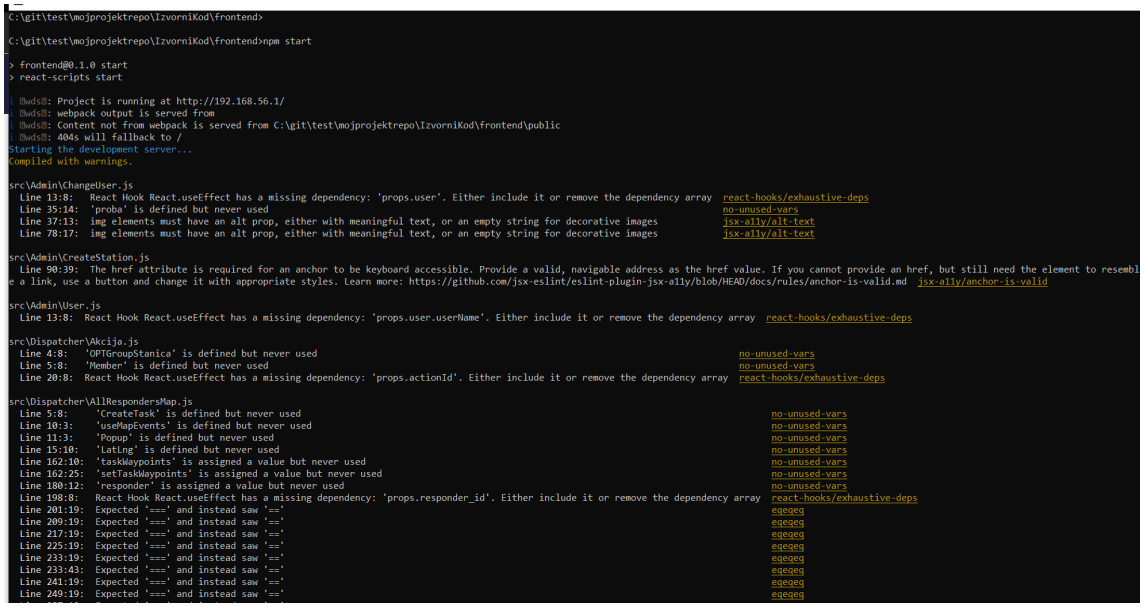
5.4.1 Priprema za puštanje u pogon

Kako bi aplikacija radila potrebno je instalirati NodeJs, Maven i Javu (minimum verzija 11). Moguće da je potrebno postaviti sustavske varijable okruženja što se može napraviti odlaskom u Control Panel - System and Security - System - Advanced System Settings - Enviromental Variables gdje u dolnjem izborniku treba pronaći PATH i ukoliko nedostaje dodati ../PUT-DO-NODEJS-DIREKTORIJA/nodejs i ..IME-MAVEN-DIREKTORIJA/bin.

5.4.2 Puštanje u pogon

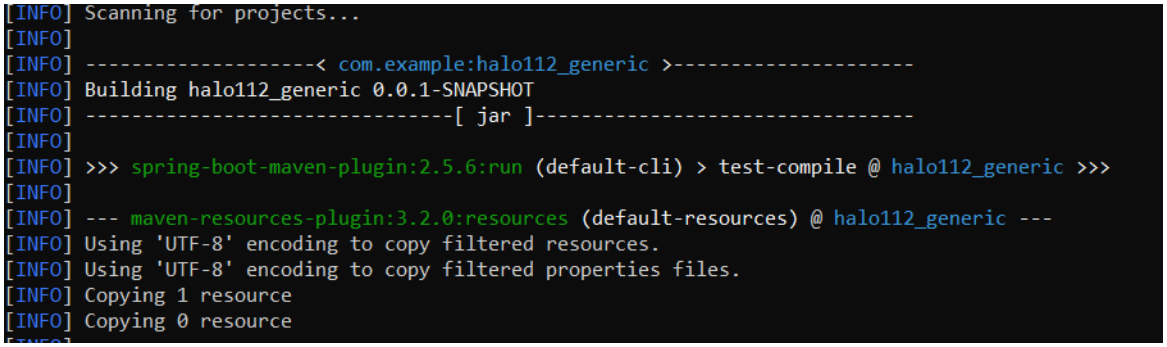
Kada su programi instalirani, potrebno je upaliti command prompt koji se može pokrenuti upisivanjem cmd u Windowsov pretraživač i premijestiti se u lokaciju projekta (korištenjem cd path naredbe). Zatim se premijestite u IzvorniKod/frontend te tamo pokrenite naredbu npm install koja će instalirati potrebne node pakete. Nakon toga pokrenite naredbu npm start koja će otvoriti vaš web preglednik i pokrenuti frontend. Nemojte zatvoriti konzolu jer time frontend završava s radom.

Slika 5.3: Otvoreni browser



Slika 5.4: Pokretanje frontenda iz cmd-a

Sada otvorite novi command prompt i premjestite se u lokaciju projekta te zatim u IzvorniKod/backend i pokrenite backend naredbom mvn spring-boot:run. Sada biste trebali vidjeti sljedeći prikaz u vašem command promptu:



Slika 5.5: Pokretanje backenda iz cmd-a

Ostavite konzolu otvorenu. Sada se možete vratiti na vaš preglednik i koristiti aplikaciju.

6. Zaključak i budući rad

Sve skupa zadatak je izrađen kvalitetno. Ne savršeno, ima mjesta za popravak ali tim je zadovoljan gotovim rezultatom. Kao tim smo većinom dobro surađivali. Početak projekta (pisanje dokumentacije) tekao je glatko. Prve probleme susreli smo pokušavajući isprogramirati generičke funkcionalnosti, sve je dobro radilo osim ulogiravanja korisnika. CORS greška se tada prvi puta pojavila i sve odtad ju nismo uspjeli rješiti. To je rezultiralo neuspjelim pokušajima deployanja aplikacije na javni poslužitelj (prvo smo isprobali HEROKU). Nekako smo zaobišli problem i uspjeli prijaviti korisnika bez obzira na CORS grešku. Za vrijeme međuispita nismo radili. Zato je bilo teško vratiti se natrag na posao odmah nakon završetka ispita i trebalo je cijelom timu dosta vremena da bi počeli opet kvalitetnije raditi. Oko Božića smo shvatili da nemamo još puno vremena za dovršiti aplikaciju, a da nam je još puno preostalo. No ipak praznici su bili i malo se radilo. Nakon završetka praznika malo pomalo pa je cijeli tim stisnuo i progurao nekako sve funkcionalnosti koje dotad nismo imali. 90 posto aplikacije smo uspjeli napraviti kako treba.

Od funkcionalnosti dugo nam je trebalo da smislimo rješenje za spremanje slika na backend i ta komunikacija (slanje slika sa frontenda na backend i obrnuto). Nakon što smo shvatili da se slika može jednostavno slati kao string unutar JSON-a, trebalo nam je samo koji dan da implementiramo i tu funkconalnost. Velik zalogaj je također bila mapa. Ali nakon što smo uspjeli postaviti mapu da se prikazuje, nije bilo većih problema sa prikazivanjem stvari na mapi.

Nismo našli kvalitetno rješenje za prikazivanje i postavljanje komentara. Teško je bilo postaviti pop-up gdje bi korisnik napisao komentar. Imali smo i problema sa prikazivanjem voronijevih dijagrama, ali smo uspjeli prikazati lokacije korisnika i zadano ih filtrirati (bez dijagrama).

Na backendu nije bilo previše „tehničkih“ poteškoća, više su bile poteškoće organizacije. Pošto je većina rađena u žurbi, nije bilo puno vremena za organizirati adrese za GET i POST. To je rezultiralo teško snalaženje frontend tima i time malo manje kvalitetan rad nego li je mogao biti. Definitivno je bilo puno truda uloženo u cjelokupno programiranje projekta, a time je stečeno i mnogo novih znanja (a

i prijateljstava). Na frontendu se najviše naučilo o asinkronosti jezika Javascript i kako pametno manevrirati kroz HTML bez da javascript baci error zbog ne dohvaćanja stavki sa backenda. Na backendu su se naučile osnove rađenja unutar programskog okvira (JavaBeans). Moglo se više toga naučiti o samoj komunikaciji sa frontendom (kako funkcioniraju protokoli i kako dobro organizirati adrese za bolje i sigurnije snalaženje). Mislim da nam je svima nedostajalo znanja o timskom radu u kodiranju, što prvenstveno uključuje uredno i organizirano pisanje kodova (tako da ne može samo autor znati kako bi kod trebao raditi, nego i bilo tko drugi koji čita kod nakon njega).

Kao voditelj tima, također mogu reći da mi je nedostajalo voditeljskog iskustva i da sam puno naučio o vođenju tima. Ima definitivno stvari koje bih učinio drugačije od početka da sam vodim ovaj projekt ispočetka. Na kraju je cijeli tim odlično radio na projektu i prije nastavka bismo definitivno uredili cijeli kod da radi kvalitetnije sad kada razumijemo neke nove stvari.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new

Indeks slika i dijagrama

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak
    - Datum: 27. listopada 2021.
    - Prisustvovali: Luka Ivanković, Ante Perković, Lukas Ujčić, Florijan Rusac, Krunoslav Zadrić, Leon Banko
    - Teme sastanka:
        - Upoznavanje sa GitLabom
2. sastanak
    - Datum: 28. listopada 2021.
    - Prisustvovali: Luka Ivanković, Ante Perković, Lukas Ujčić, Florijan Rusac, Krunoslav Zadrić, Leon Banko, Mario Galić
    - Teme sastanka:
        - Zadavanje zadataka
3. sastanak
    - Datum: 31. listopada 2021.
    - Prisustvovali: Luka Ivanković, Ante Perković, Lukas Ujčić, Florijan Rusac, Krunoslav Zadrić, Leon Banko, Mario Galić
    - Teme sastanka:
        - Provjera napretka i odrađenog
4. sastanak
    - Datum: 7. studenoga 2021.
    - Prisustvovali: Luka Ivanković, Ante Perković, Lukas Ujčić, Florijan Rusac, Krunoslav Zadrić, Leon Banko, Mario Galić
    - Teme sastanka:
        - Provjera napretka i odrađenog
        - Zadavanje zadataka

Tablica aktivnosti

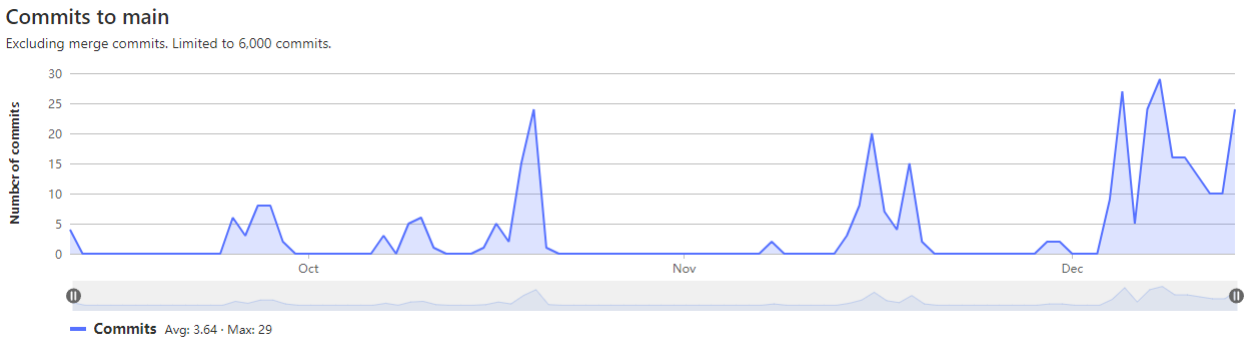
	Luka ivanković	Lukas Ujčić	Ante Perković	Florijan Rusac	Krunoslav Zadrić	Mario Galić	Leon Banko
Upravljanje projektom							
Opis projektnog zadatka						2	
Funkcionalni zahtjevi							
Opis pojedinih obrazaca				4	4		
Dijagram obrazaca		2					2
Sekvencijski dijagrami		3					4
Opis ostalih zahtjeva						2	
Arhitektura i dizajn sustava				1			
Baza podataka				3	4		
Dijagram razreda		3					3
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Korištene tehnologije i alati							
Ispitivanje programskog rješenja							
Dijagram razmještaja							
Upute za puštanje u pogon							
Dnevnik sastajanja						0.5	
Zaključak i budući rad							

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Luka ivanković	**Lukas Ujčić**	**Ante Perković**	**Florijan Rusac**	**Krunoslav Zadrić**	**Mario Galić**	**Leon Banko**
Popis literature							
Dodatne stavke kako ste podijelili izradu aplikacije							
npr. izrada početne stranice							
izrada baze podataka							
spajanje s bazom podataka							
back end							

Dijagrami pregleda promjena



Slika 6.1: Dijagram pregleda promjena