

Programsko inženjerstvo

Ak. god. 2021./2022.

Radno vrijeme

Dokumentacija, Rev. 2.0

Grupa: *Dugonogi Progi*

Voditelj: *Bernard Kazazić*

Datum predaje: *15. siječnja 2022.*

Nastavnik: *Miljenko Krhen*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	6
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	20
3.2 Ostali zahtjevi	22
4 Arhitektura i dizajn sustava	23
4.1 Baza podataka	24
4.1.1 Opis tablica	24
4.1.2 Dijagram baze podataka	28
4.2 Dijagram razreda	29
4.3 Dijagram stanja	39
4.4 Dijagram aktivnosti	40
4.5 Dijagram komponenti	53
5 Implementacija i korisničko sučelje	54
5.1 Korištene tehnologije i alati	54
5.2 Ispitivanje programskog rješenja	55
5.2.1 Ispitivanje komponenti	55
5.2.2 Ispitivanje sustava	71
5.3 Dijagram razmještaja	78
5.4 Upute za puštanje u pogon	79
6 Zaključak i budući rad	88
Popis literature	89
Indeks slika i dijagonama	92

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Raspudić	14.10.2021.
0.1.1	Dopisani dnevnički sastajanja.	Raspudić	15.10.2021.
0.2	Dodani nefunkcionalni zahtjevi.	Raspudić	16.10.2021.
0.3	Dodani funkcionalni zahtjevi.	Pašalić, Sušac	17.10.2021.
0.4	Dodan opis projekta.	Pavlović, Raspudić	21.10.2021.
0.5	(1) Dodani obrasci uporabe i sekvencijski dijagrami (2) Dodana arhitektura i dizajn sustava	(1) M.Erlić, A.Pašalić, V.Žunar (2) B.Kazazić, P.Sušac	24.10.2021.
0.5.1	Manje gramatičke izmjene.	Raspudić	04.11.2021.
0.6	Napravljeni dijagrami razreda.	M. Erlić, B. Pavlović, B. Kazazić, V. Žunar	18.11.2021.
0.6.1	Opis dijagrama razreda	M. Erlić, B. Pavlović, B. Kazazić, V. Žunar	19.11.2021.

Rev.	Opis promjene/dodataka	Autori	Datum
1.0	Ispravak dijagrama razreda	M. Erlić, B. Pavlović, B. Kazazić, V. Žunar	10.12.2021.
1.1	(1) Dodan dijagram stanja (2) Dodan dijagram aktivnosti (3) Dodan dijagram komponenti	(1) B. Pavlović (2) M. Erlić (3) M. Erlić, L. Raspudić, A. Pašalić	20.12.2021.
1.2	(1) Opis korištenih tehnologija i alata (2) Prikaz ispitivanja komponenata (3) Ispitivanje sustava (Selenium IDE) (4) Dodan dijagram razmještaja	(1) B. Pavlović, V. Žunar (2) M. Erlić, B. Pavlović (3) P. Sušac (4) V. Žunar	4.1.2022.
1.3	Stvorene upute za puštanje u pogon	B. Kazazić	6.1.2022.
1.4	Zaključak i budući rad	V. Žunar	7.1.2022.
1.5	Dodan popis literature	V. Žunar	8.1.2022.
1.6	Dodan dnevnik sastajanja	V. Žunar	10.1.2022.

Rev.	Opis promjene/dodataka	Autori	Datum
1.7	Tablica aktivnosti popunjena	V. Žunar, B. Pavlović, L. Raspudić, B. Kazazić, M. Erlić, A. Pašalić	11.1.2022.
1.8	Dodani dijagrami pregleda promjena	B. Kazazić	14.1.2022.
2.0	Završno uređivanje dokumentacije	A. Pašalić	14.1.2022.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za web aplikaciju "Radno vrijeme" koja će omogućiti djelatnicima poduzeća „Mi puno radimo“ praćenje realizacije pojedinih djelatnosti i radnih sati na razini svakog djelatnika, grupe i poduzeća, kao i praćenje zauzeća/raspoloživosti pojedinih djelatnika. Na taj će se način olakšati organizacija istovremenog obavljanja mnogih uslužnih djelatnosti.

Korisnike sustava možemo podijeliti u 4 grupe:

- vlasnika sustava
- voditelja grupa
- zaposlenike
- neregistrirane korisnike

Neregistrirani korisnici mogu samo vidjeti popis i opis djelatnosti koje poduzeće ima u svojem portfoliju. Direktoru je omogućeno registriranje korisnika u sustav kreiranjem novog računa. Za kreiranje novog računa potrebni su sljedeći podaci:

- korisničko ime
- lozinka
- ime
- prezime
- OIB
- adresa e-pošte

Svim djelatnicima kasnije korisničko ime i lozinka služe za prijavu u sustav, a adresa e-pošte kako bi ih se moglo kontaktirati.

Naknadno se djelatnicima, ako već nisu, mogu još i dodijeliti prava voditelja grupe. Ukoliko radnici idu na intervencije izvan lokacije sjedišta poduzeća, potrebno je za svakoga prikazati mjesto na karti za lokaciju na koju je bio upućen.

Djelatnik poduzeća kroz aplikaciju može vidjeti podatke samo o sebi. Njemu se prikazuju grupe u koje je raspodijeljen te zadatke koji su mu dodijeljeni. Jedan djelatnik može biti raspoređen i na nekoliko radnih zadataka unutar jedne djelatnosti, a i između više njih.

Ukoliko je djelatniku dodijeljen zadatak na nekoj lokaciji različitoj od sjedišta poduzeća, on tu promjenu u lokaciji mora unijeti na karti prije odlaska, ako nije voditelj grupe već unio u sustav.

Voditelji grupe mogu kroz aplikaciju dobiti podatke o sebi, ali i o članovima svoje grupe. Oni razrađuju plan rada svoje grupe te određuju zadatke i pridjeljuju ih zaposlenicima. Za svaki zadatak voditelj određuje očekivani broj potrebnih radnih sati te cijenu radnog sata na razini pojedine djelatnosti i/ili zadatka.

Vlasnik može stvarati grupe, pri čemu zaposlenika postavlja kao voditelja te grupi dodjeljuje djelatnost i članove, a također ima mogućnost brisanja grupe.

On također može definirati nove uslužne djelatnosti kojima se poduzeće bavi te će te promjene biti vidljive u popisu djelatnosti.

Vlasniku sustava se kroz aplikaciju prikazuju zauzetost i realizacija (stvarna i materijalna) za sve djelatnike te odnos planiranih i realiziranih troškova/dobiti. Informacija o zauzetosti djelatnika pomaže mu pri stvaranju grupa kako ne bi preopteretio zaposlenike prevelikom količinom zadataka te na taj način osigurao veću količinu i kvalitetu održenog posla.

Ukoliko postoje aktivnosti izvan sjedišta poduzeća, direktor u svakom trenutku mora moći vidjeti na karti gdje se nalazi (ili se nalazio) koji djelatnik. Podaci o adresi intervencije moraju biti prethodno uneseni u sustav.

Na kraju svakog radnog dana svaki pojedini djelatnik (uključujući i voditelje

grupa) upisuju broj odrđenih radnih sati taj dan.

Sustav mora omogućiti istovremeni rad svih korisnika sustava i mora omogućiti unos hrvatskih dijakritičkih znakova.

Sustav također mora biti prilagođen i u potpunosti poštivati sva prava radnika, uključujući radno vrijeme (početak i kraj), duljinu radnog vremena, slobodne dana i ostala prava koja radnik ima temeljem općih i posebnih uvjeta definiranih zakonom.

Gledajući dostupne programe koji bi omogućili praćenje produktivnosti zaposlenika možemo vidjeti razne kategorije. Od najpopularinijih, najskupljih, najjednostavnijih pa sve do najjeftinijih. Ujedno je problem što je toliko veliki izbor između svih programa. To otežava voditeljima poduzeća posao odmjeravanja jakosti i slabosti svakog programa. Pošto su svi ti programi namjenjeni široj publici, svaki od tih programa će doći sa svojim opcijama koje nisu nužne niti potrebne poduzeću "Mi puno radimo". Ukoliko odaberu neki program, tu dolazi još jedan problem. Učenje korištenja tog programa (engl. *learning curve*). Potrebno je vrijeme i trud prije nego što počnu koristiti taj program optimizirano.

Tu dolazimo mi. Mi dajemo priliku poduzeću "Mi puno radimo" sa krojenim programom prema njihovim potrebama. Ne trebaju plaćati za opcije koje neće koristiti, ne trebaju odustati od svojih zahtjeva kako bi koristili jeftinija i/ili lošija rješenja. Dajemo im priliku da napravimo program koji će oni sami intuitivno znati koristiti jer će sudjelovati u svakom koraku korisničkog sučelja i njegovih funkcionalnosti.

Projekt je uvijek moguće nadograditi. Ovaj projekt je savršen za male, srednje i velike tvrtke. Možemo uz dogovor s klijentima povezati program s njihovim drugim servisima za koje drugi programi nemaju mogućnost, a niti im je u planu jer većina ne bi koristila. Trenutni cilj projekta nije specifičan da ga može samo jedno poduzeće koristiti. Na primjer, trenutna ideja je da zaposlenici sami unesu broj radnih sata za pojedini dan, moguće je nadograditi da se to automatski računa uz pametne kartice, optionalno koje bismo mi enkriptirali. Također, ukoliko neke tvrtke imaju već brojače sati za zaposlenike, ali im treba većina naših opcija, uz

dogovor možemo proširiti naš program da dobije podatke iz njihovog brojača te samo proširimo program za te servise koje oni traže.

Prednost našeg rješenja je što ne zahtjeva instalaciju na svaki uređaj koji treba pristupiti servisu. Svatko može sa svojeg pametnog mobitela ili računala doći do svojih zadataka, preinaka i novih informacija. Samo trebaju znati svoje korisničko ime i lozinku. Iako ova prednost može doći kao i mana, tu opet dolazimo do naše prethodne snage. Mi možemo, ukoliko je potrebno, ojačati sigurnost programa. Na način da program dozvoli pristup samo ovlaštenim računalima/mobitelima. Na primjer, zabrana pristupa osim dozvoljenim MAC adresama, dozvola pristupa računalima i mobitelima sa instaliranim certifikatom koji mi napišemo i sl. Naša tvrtka kroji proizvod svojim klijentima.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik sustava (direktor poduzeća/naručitelj)
2. Djelatnici poduzeća
 - Voditelji grupa
 - Ostali djelatnici
3. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Vlasnik sustava (direktor poduzeća) može:
 - (a) definirati uslužne djelatnosti koje će poduzeće raditi
 - (b) dodijeliti djelatnosti voditeljima grupa
 - (c) registrirati novog djelatnika (uključujući i voditelja grupa) i pritom mu dodijeliti ulogu
 - (d) vidjeti zauzetost i realizaciju za sve djelatnike
 - (e) vidjeti trenutno i prošlu poticiju svih djelatnika koji su izašli na intervencije na karti
2. Voditelji grupa mogu:
 - (a) definirati zadatke i dodjeljivati ih djelatnicima
 - (b) zabilježiti procjenu radnih sati potrebnih za zadatak
 - (c) odrediti cijenu sata rada ovisno o djelatnosti ili zadatku
 - (d) pregledati podatke za sebe i svoju grupu
 - (e) upisati broj odrađenih sati za svaki dan

3. Ostali djelatnici mogu:

- (a) vidjeti koji su mu zadaci dodijeljeni i u kojim se grupama nalazi
- (b) pregledati vlastite podatke
- (c) upisati broj odrađenih radnih sati za svaki dan

4. Neregistrirani korisnik može:

- (a) vidjeti popis i opis djelatnosti poduzeća

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled djelatnosti

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati djelatnosti koje poduzeće obavlja
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik u aplikaciji odabire opciju "Prikaz djelatnosti"
 2. Prikaže se popis svih djelatnosti poduzeća

UC2 - Pregled opisa djelatnosti

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati opis djelatnosti koje poduzeće obavlja
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire djelatnost za koju želi vidjeti informacije
 2. Prikaže se stranica s opisom odabrane djelatnosti

UC3 - Registracija korisnika

- **Glavni sudionik:** Direktor
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Direktor odabire opciju za registraciju novih korisnika
 2. Direktor unosi sve potrebne korisničke podatke
 3. Direktor novom korisniku dodjeljuje ulogu
 4. Direktor prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 2.a Odabir već zauzetog korisničkog imena i/ili elektroničke pošte, unos podataka u nedozvoljenom formatu ili upis neispravnog imena elektroničke pošte

1. Sustav obavljaštava direktora o neuspjelom upisu i vraća ga na stranicu za registraciju
2. Direktor mijenja potrebne podatke te završava unos ili odustaje od registracije

UC4 - Prijava u sustav

- **Glavni sudionik:** Djetatnik
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Djetatnik odabire opciju za prijavu u sustav
 2. Djetatnik upisuje korisničko ime i lozinku
 3. Sustav potvrđuje ispravnost unesenih podataka
 4. Djetatnik dobiva pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Unos neispravnog imena i/ili lozinke
 1. Sustav obavljaštava korisnika o neispravnom upisu podataka i neuspjeloj prijavi te ga vraća na stranicu za prijavu

UC5 - Pregled svih grupa

- **Glavni sudionik:** Vlasnik sustava (direktor poduzeća)
- **Cilj:** Pregledati sve postojeće grupe u poduzeću
- **Sudionici:** baza podataka
- **Preduvjet:** vlasnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Vlasnik odabire opciju "Pregled grupa"
 2. Vlasniku se prikazuje popis grupa

UC6 - Stvaranje grupe

- **Glavni sudionik:** Vlasnik sustava (direktor poduzeća)
- **Cilj:** Stvoriti grupu djetatnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Vlasnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Vlasnik odabire opciju "Pregled grupa"

2. Vlasniku se prikazuje popis grupa
3. Vlasnik odabire opciju "Stvori novu grupu"
4. Vlasniku se prikazuje popis svih djelatnika
5. Vlasnik odabire djelatnike koji će biti raspodijeljeni u grupu
6. Vlasnik odabire opciju "Odaberi voditelja"
7. Vlasniku odabire jednoga od članova grupe s popisa kojega postavlja za voditelja
8. Vlasnik odabire opciju "Potvrđujem odabir"
9. Sustav javlja vlasniku da je grupa uspješno stvorena

UC7 - Brisanje grupe

- **Glavni sudionik:** Vlasnik
- **Cilj:** Obrisati grupu
- **Sudionici:** Baza podataka
- **Preduvjet:** Vlasnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Vlasnik odabire opciju "Pregled grupe"
 2. Vlasniku se prikazuje popis grupa
 3. Vlasnik odabire grupu koju želi obrisati
 4. Vlasnik odabire opciju "Brisanje grupe"
 5. Vlasniku se prikazuje poruka "Jeste li sigurni da želite obrisati grupu?"
 6. Vlasnik odabire opciju "Da"

UC8 - Definiranje uslužne djelatnosti

- **Glavni sudionik:** Vlasnik
- **Cilj:** Definirati uslužnu djelatnost
- **Sudionici:** baza podataka
- **Preduvjet:** Vlasnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Vlasnik odabire opciju "Pregled djelatnosti"
 2. Vlasniku se prikazuje popis djelatnosti
 3. Vlasnik odabire opciju "Stvori djelatnost"
 4. Vlasnik ispuni obrazac za stvaranje djelatnosti
 5. Vlasnik odabire opciju "Potvrdi unos"
- **Opis mogućih odstupanja:**
 - 4.a Neispunjena sva polja obrasca

1. Sustav šalje poruku "Molimo vas ispunite sva polja obrasca."
2. Vlasnik ispunjava obrazac do kraja i ponovno ga potvrđuje

UC9 - Dodjela zadataka (voditelji djelatnicima)

- **Glavni sudionik:** voditelj grupe
- **Cilj:** Dodijeliti djelatnost Razrada plana rada i dodjela zadataka pojedinim djelatnicima
- **Sudionici:** baza podataka
- **Preduvjet:** Korisnik je registriran i dodijeljena mu je uloga "voditelj grupe"
- **Opis osnovnog tijeka:**
 1. Voditelj grupe odabire djelatnika koji će obavljati zadatak.
 2. Voditelj grupe dodjeljuje zadatak odabranom djelatniku.
 3. Upisani podaci se spremaju pritiskom na opciju "Spremi"
 4. Zadaci postaju vidljivi pojedinim djelatnicima.
- **Opis mogućih odstupanja:**
 - 2.a Zauzetost djelatnika
 1. U trenutku dodjele zadatka odabranom djelatniku, ukoliko je već u tom vremenskom periodu istom zadan neki zadatak, sustav šalje poruku upozorenja o preklapanju.
 2. Voditelj grupe zatim zadaje drugi termin djelatniku.

UC10 - Upis odrđenih sati

- **Glavni sudionik:** djelatnici, voditelji grupa
- **Cilj:** Praćenje odrđenih radnih sati u danu za svakog djelatnika.
- **Sudionici:** baza podataka
- **Preduvjet:** Prijava korisnika u sustav.
- **Opis osnovnog tijeka:**
 1. Svaki djelatnik i voditelj grupa nakon radnog dana upiše odrđeno vrijeme (sati i minute) za taj dan.
 2. Uneseni podaci pohranjuju se u bazu klikom na gumb "Spremi".

UC11 - Pregled dodijeljenih zadataka

- **Glavni sudionik: djelatnik**
- **Cilj:** Pojedini djelatnik može vidjeti koji su mu zadaci dodijeljeni.
- **Sudionici:** baza podataka
- **Preduvjet:** Korisnik mora biti prijavljen u sustav i imati ulogu "djelatnika".

- **Opis osnovnog tijeka:**

1. Djelatnik pristupa stranici dodijeljenih zadataka.
2. S obzirom da je djelatnik prijavljen u sustav, poznati su nam njegovi identifikacijski podaci te na temelju njih prikazuje se raspored samo za tog djelatnika.

UC12 - Prikaz dodijeljenih grupa

- **Glavni sudionik:** djelatnik

- **Cilj:** Pojedini djelatnik može vidjeti u kojim se grupama nalazi

- **Sudionici:** baza podataka

- **Preduvjet:** Korisnik mora biti prijavljen u sustav i imati ulogu "djelatnika".

- **Opis osnovnog tijeka:**

1. Djelatnik odabire stranicu dodijeljenih grupa.
2. Na temelju njegovih identifikacijskih oznaka prikazuju mu se dodijeljene grupe.

UC13 - Pregled zauzetosti djelatnika

- **Glavni sudionik:** djelatnik, voditelj grupe, direktor

- **Cilj:** Korisnik može vidjeti predviđenu zauzetost djelatnika ovisno o ulozi. Direktor ima pristup zauzetosti svih djelatnika, voditelji grupe imaju pristup podacima za sebe i svoju grupu, a djelatnik može vidjeti samo podatke za sebe.

- **Sudionici:** baza podataka

- **Preduvjet:** Korisnik mora biti prijavljen u sustav.

- **Opis osnovnog tijeka:**

1. Korisnik, ako je direktor ili voditelj grupe odabire djelatnika čiju zauzetost želi provjeriti.
2. Odabire se vrijeme za koje se želi provjeriti zauzetost.
3. Na temelju identifikacijskih podataka odabranog djelatnika ili prijavljenog korisnika, ako je njegova uloga "djelatnik" ili "voditelj grupe", prikazuje se zauzetost u prethodno navedenom vremenu.

UC14 - Pregled odnosa planiranih i realiziranih troškova

- **Glavni sudionik:** Direktor

- **Cilj:** Evidencija finansijskih rashoda.

- **Sudionici:** baza podataka

- **Preduvjet:** Korisnik mora biti prijavljen u sustav i imati ulogu "direktor".
- **Opis osnovnog tijeka:**
 1. Direktor unosi planirane troškove u bazu.
 2. Tijekom obavljanja poslova, redovito se upisuju realizirani troškovi.
 3. Sustav na temelju unesenih podataka računa razliku.
 4. Prikazuje se odnos planiranih I realiziranih troškova.

UC15 - Pregled odnosa planiranih I realiziranih dobiti

- **Glavni sudsionik:** direktor
- **Cilj:** Evidencija finansijskih prihoda.
- **Sudsionici:** baza podataka
- **Preduvjet:** Korisnik prijavljen u sustav i ima ulogu "direktor"
- **Opis osnovnog tijeka:**
 1. Direktor unosi planiranu dobit u bazu.
 2. Tijekom obavljanja poslova, redovito se upisuju realizirane dobiti.
 3. Sustav na temelju unesenih podataka računa razliku.
 4. Prikazuje se odnos planiranih I realiziranih dobiti.

UC16 - Pregled podređene grupe (voditelj)

- **Glavni sudsionik:** Voditelj grupe
- **Cilj:** Voditelj treba moći vidjeti članove grupe kojoj je on nadređen. Može vidjeti članove grupe, njihove podatke i dodijeljene zadatke.
- **Sudsionici:** baza podataka
- **Preduvjet:** korisnik je prijavljen u sustav i ima ulogu "voditelj"
- **Opis osnovnog tijeka:**
 1. Voditelj odabire opciju "Moje podređene grupe".
 2. Prikazuje se popis svih grupa kojima je taj voditelj nadređen.
 3. Voditelj odabire grupu za koju želi vidjeti podatke.
 4. Prikazuju se članovi grupe i djelatnost koja joj je dodijeljena. Odabirom pojedinog djelatnika prikazuju se njegovi podaci i dodijeljeni zadaci.

UC17 - Pregled vlastitih podataka

- **Glavni sudsionik:** direktor, voditelj grupe, djelatnik
- **Cilj:** Svi djelatnici (uključujući i direktora i voditelje grupe) mogu vidjeti svoje osobne podatke, dodijeljene zadatke i grupe u kojima se nalazi.
- **Sudsionici:** baza podataka

- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Moji podaci".
 2. Prikazuju se korisnikovi osobni podaci, njegovi dodijeljeni zadaci te grupe u kojima se nalazi.

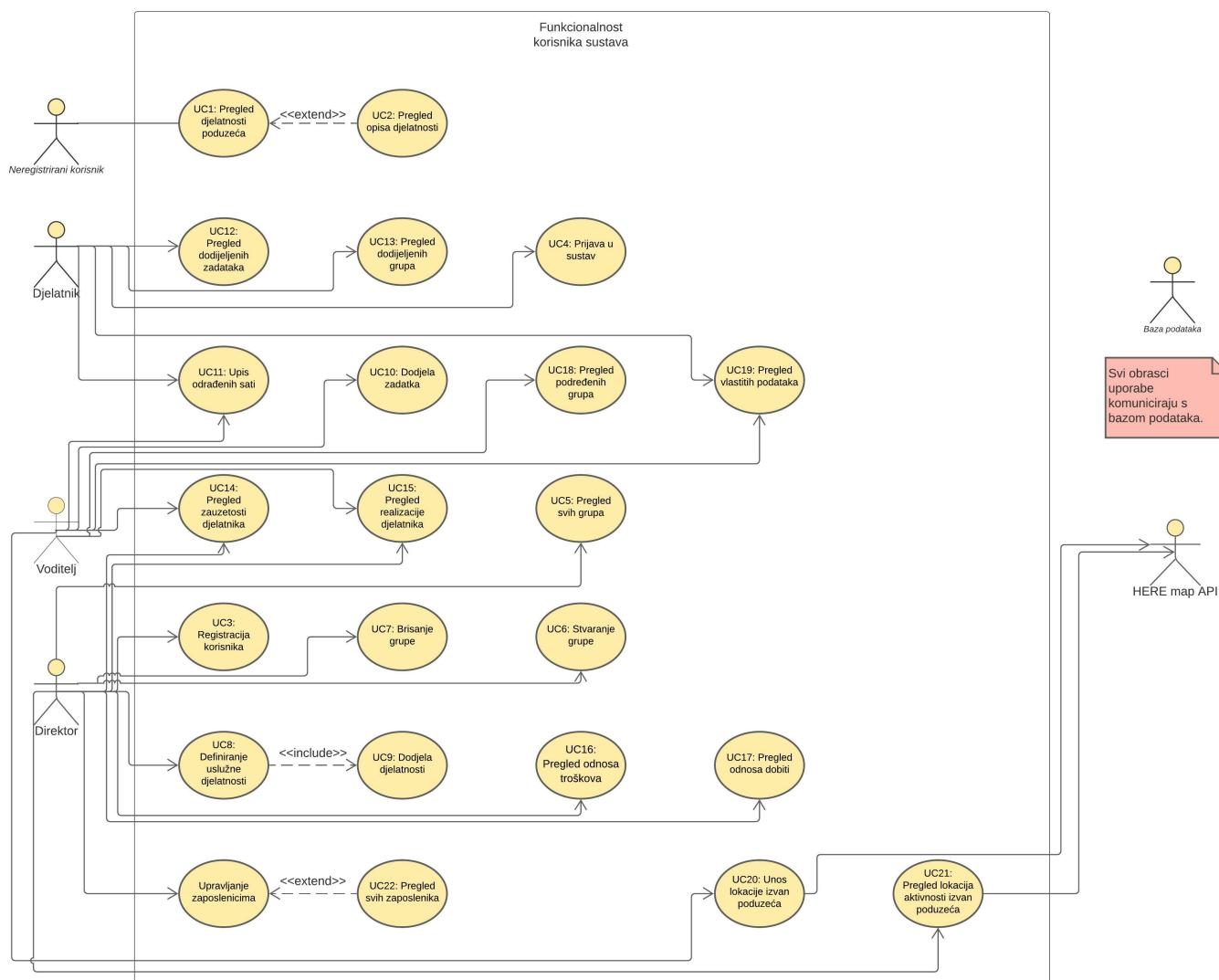
UC18 - Unos lokacije aktivnosti izvan poduzeća

- **Glavni sudionik:** voditelj grupe
- **Cilj:** Voditelj treba stvoriti zadatak koji se obavlja izvan poduzeća
- **Sudionici:** baza podataka
- **Preduvjet:** Voditelj je prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Voditelj grupe odabire djelatnika koji će obavljati zadatak.
 2. Voditelj grupe dodjeljuje zadatak izvan poduzeca odabranom djelatniku.
 3. Upisani podaci se spremaju pritiskom na opciju "Spremi"
 4. Zadaci postaju vidljivi pojedinim djelatnicima.
- **Opis mogućih odstupanja:**
 - 2.a Zauzetost djelatnika
 1. U trenutku dodjele zadatka odabranom djelatniku, ukoliko je već u tom vremenskom periodu istom zadan neki zadatak, sustav šalje poruku upozorenja o preklapanju.
 2. Voditelj grupe zatim zadaje drugi termin djelatniku.

UC19 - Pregled lokacija aktivnosti izvan poduzeća

- **Glavni sudionik:** direktor
- **Cilj:** Moguće je da djelatnici imaju aktivnosti koje moraju obaviti na nekoj lokaciji izvan poduzeća. U tom slučaju direktor mora moći vidjeti gdje su se nalazili.
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik je prijavljen u sustav i ima ulogu "direktor"
- **Opis osnovnog tijeka:**
 1. Direktor odabire opciju "Pregledajte aktivnosti izvan poduzeća".
 2. Prikazuje se karta s označenim lokacijama aktivnosti.
 3. Odabirom pojedine lokacije prikazuju se podaci o aktivnosti koja se obavljala na toj lokaciji I djelatniku koji je tu aktivnost obavljao.

Dijagrami obrazaca uporabe

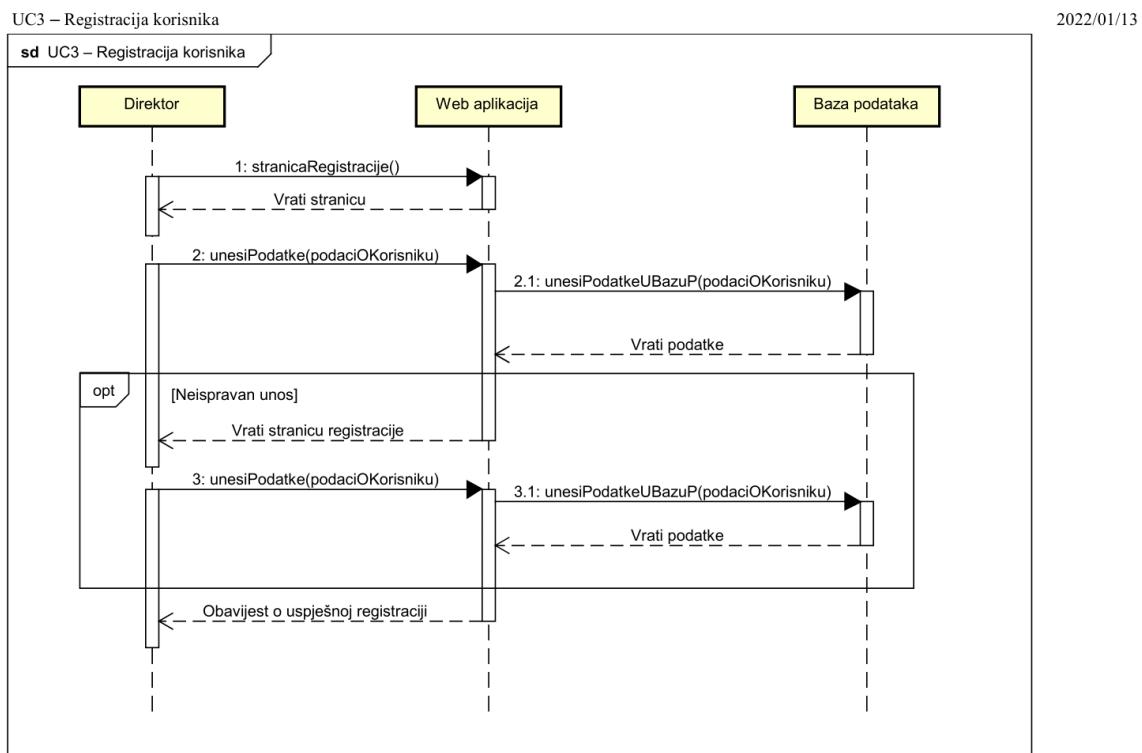


Slika 3.1 Funkcionalnosti korisnika sustava

3.1.2 Sekvencijski dijagrami

UC3 – Registracija korisnika

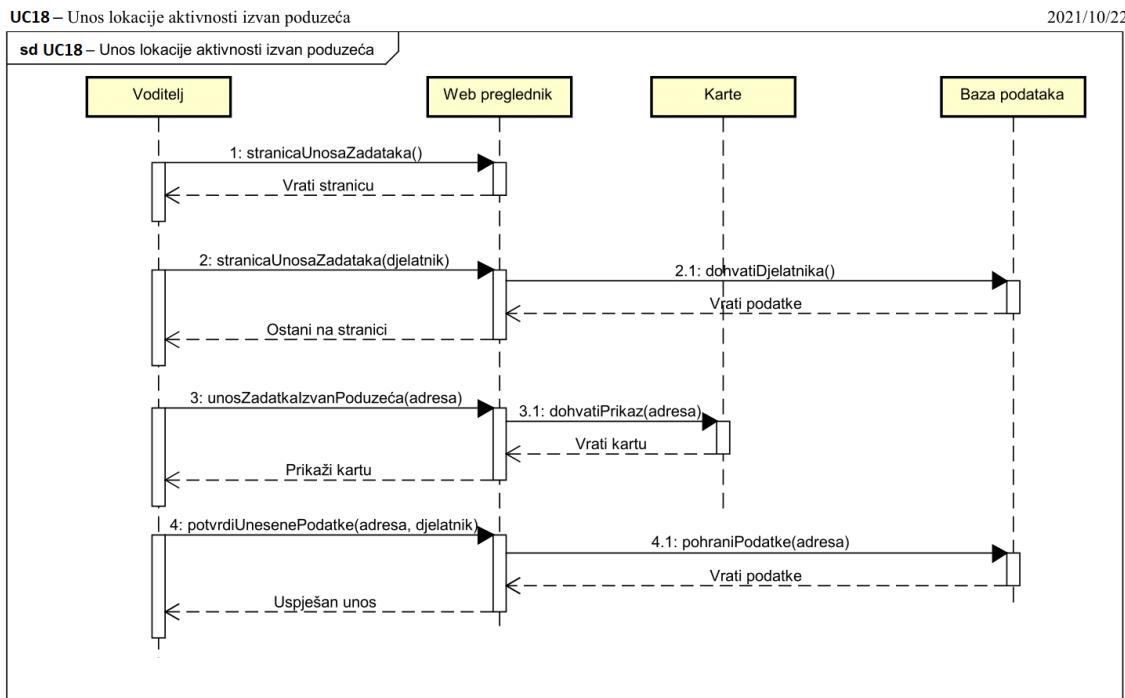
Direktor otvara stranicu za registraciju djelatnika. Obavljanje zahtjeva obavlja se na poslužitelju koji kao odgovor vraća zatraženu stranicu. Direktor unosi sve potrebne korisničke podatke za stvaranje potpuno novog profila djelatnika. Prilikom registracije direktor također dodjeljuje ulogu djelatniku. Tijekom unosa moguće su sljedeća odstupanja: već zauzeto korisničko ime i/ili elektronička pošta, unos podataka u neispravnom formatu (ilegalni znakovi, predugačak unos ili unos krivog tipa podataka). Ukoliko dođe do pogreške prilikom unosa, poslužitelj obavještava direktora o neuspjelom zahtjevu i vraća ga na stranicu za registraciju. U takvom slučaju direktor ispravlja unos te ponovno pokušava izvršiti unos ili odustaje od registracije. Svi uneseni podaci šalju se na poslužitelj te se spremaju u bazu podataka. Direktor prima obavijest o uspješnoj registraciji.



Slika 3.2 Registracija korisnika

UC18 – Unos lokacije aktivnosti izvan poduzeća

Djelatnik šalje poslužitelju zahtjev za unos aktivnosti izvan poduzeća. Na web pregledniku otvara se obrazac za upis adrese na kojoj je djelatnik obavljao aktivnost. Podaci se šalju pregledniku te se dohvaća karta s prikazom navedene adrese. Karta se prikazuje djelatniku. Djelatnik potvrđuje lokaciju pritiskom na gumb “Potvrди” te se adresa upisuje u bazu podataka.



Slika 3.5 Unos lokacije aktivnosti izvan poduzeća

3.2 Ostali zahtjevi

Sustav treba:

- omogućiti korištenje hrvatskih dijakritičkih znakova pri unosu i prikazu tekstuallnog sadržaja
- podržati višekorisnički rad u realnom vremenu
- dati odgovor na traženi upit unutar nekoliko sekundi kada se dohvaćaju podaci iz baze podataka
- imati intuitivno i jednostavno za korištenje korisničko sučelje
- biti implementiran kao web aplikacija koristeći objektno-orientirane jezike
- neispravno korištenje korisničkog sučelja ne smije narušiti rad sustava
- osigurati sigurnu, brzu i otpornu na vanjske greške vezu s bazom podataka

4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na tri podsustava:

- Web poslužitelj
- Web aplikacija
- Baza podataka

Web preglednik je program koji korisniku omogućuje pregled web stranica i multimedijalnih sadržaja vezanih uz njih. Svaki internetski preglednik je prevoditelj. Dakle, stranica je pisana u kodu koji preglednik nakon toga interpretira kako nešto svakome razumljivo. Korisnik putem web preglednika šalje zahtjev web poslužitelju.

Web poslužitelj osnova je rada web aplikacije. Njegova primarna zadaća je komunikacija klijenta s aplikacijom. Komunikacija se odvija preko HTTP (engl. Hyper Text Transfer Protocol) protokola, što je protokol u prijenosu informacija na webu. Poslužitelj je onaj koji pokreće web aplikaciju te joj prosljeđuje zahtjev.

Korisnik koristi web aplikaciju za obrađivanje željenih zahtjeva. Web aplikacija obrađuje zahtjev te ovisno o zahtjevu, pristupa bazi podataka nakon čega preko poslužitelja vraća korisniku odgovor u obliku HTML dokumenta vidljivog u web pregledniku. HTML elementi su stvoreni koristeći JavaScript biblioteku React. Odabранo razvojno okruženje je Visual Studio Code.

Programski jezik kojeg smo odabrali za izradu naše web aplikacije je Java Spring Boot. Odabrana razvojna okruženja su Eclipse IDE i IntelliJ IDEA. Arhitektura sustava temeljiti će se na MVC (Model-View-Controller) konceptu.

Karakteristika MVC koncepta je nezavisan razvoj pojedinih dijelova aplikacije što za posljedicu ima jednostavnije ispitivanje kao i jednostavnije razvijanje i dodavanje novih svojstava u sustav.

MVC koncept sastoji se od:

- Model - Središnja komponenta sustava. Predstavlja dinamičke strukture podataka, neovisne o korisničkom sučelju. Izravno upravlja podacima, logikom i pravilima aplikacije. Također prima ulazne podatke od Controllera

- View - Bilo kakav prikaz podataka, poput grafa. Mogući su različiti prikazi iste informacije poput grafičkog ili tabličnog prikaza podatak.
- Controller – Prima ulaze i prilagođava ih za proslijedivanje Model-u ili View-u. Upravlja korisničkim zahtjevima i na temelju njih izvodi daljnju interakciju s ostalim elementima sustava.

4.1 Baza podataka

dio 1. revizije

Baza podataka je predstavljena pomoću ER te relacijskog dijagrama. Od glavnih komponenti, primarno se pojavljuju:

- Employee
- Role
- Group
- EmployeeGroup
- Job
- Task
- EmployeeTask
- Location
- WorkHoursInput

4.1.1 Opis tablica

Employee

Sadrži podatke o djelatniku, uključujući osobne podatke i podatke za prijavu (korisničko ime i lozinka).

Employee		
pid	CHAR	Osobni identifikacijski broj djelatnika.

Employee		
name	VARCHAR	Ime djelatnika.
surname	VARCHAR	Prezime djelatnika.
email	VARCHAR	E-mail adresa.
username	VARCHAR	Jedinstveno korisničko ime djelatnika.
password	VARCHAR	Lozinka djelatnika.
idRole	INTEGER	Identifikator uloge djelatnika.

Role

Sadrži podatke o ulozi korisnika. Uloga može biti: direktor, vođa tima ili djelatnik.

Role		
idRole	INTEGER	Identifikator uloge.
name	VARCHAR	Naziv uloge (direktor, vođa tima, djelatnik).

Group

Djelatnici rade u grupama. Grupa ima svog voditelja - djelatnika kojem je direktor dodijelio ulogu voditelja. Grupa može biti stvorena bez članova, ali ne može biti bez voditelja. Svaka se grupa bavi nekom djelatnosti.

Group		
idGroup	INTEGER	Identifikator grupe.
name	VARCHAR	Naziv grupe.
idLeader	INTEGER	Identifikator voditelja grupe.
idJob	INTEGER	Identifikator djelatnosti kojom se grupa bavi.

EmployeeGroup

Tablica koja ostvaruje vezu više-na-više između tablica Employee i Group, tj. asocira djelatnike s grupama čiji su članovi. Djelatnik može istovremeno raditi u više grupa.

EmployeeGroup		
idEmployee	CHAR	Identifikator djelatnika.
idGroup	INTEGER	Identifikator grupe.

Job

Sadrži podatke o djelatnosti kojima se tvrtka bavi. Svakoj djelatnosti pridružen je naziv i tekstualni opis.

Job		
idJob	INTEGER	Identifikator djelatnosti.
name	VARCHAR	Naziv djelatnosti.
description	TEXT	Opis djelatnosti.

Task

Zadaci na kojima djelatnici rade, a zadaje ih voditelj grupe za djelatnike kojima je nadležan. Svaki zadatak ima naziv i opis, datum i vrijeme planiranog početka i završetka, te procjenu broja radnih sati koji će biti potrebni da bi se zadatak dovršio. Voditelj grupe može i pridružiti zadatak nekoj djelatnosti. Zadatak se može obavljati na nekoj lokaciji. Pri stvaranju zadatka, voditelj određuje planirani trošak i prihod. Kada djelatnik dovrši zadatak, stvarni trošak i prihod se upisuju u tablicu.

Task		
idTask	INTEGER	Identifikator zadatka.
name	VARCHAR	Naziv zadatka.
description	TEXT	Tekstualni opis zadatka.
dateTimeStart	TIMESTAMP	Datum i vrijeme početka zadatka. Mora biti manje vrijednosti od polja datumVrijemeZavrsetka.
dateTimeEnd	TIMESTAMP	Datum i vrijeme završetka zadatka. Mora biti veće vrijednosti od polja datumVrijemePocetka.
hoursNeededEstimate	SMALLINT	Procjena broja radnih sati potrebnih za dovršetak zadatka. Procjenu određuje voditelj tima koji je zadao zadatak. Procjena može, ali ne mora odgovarati razlici između dateTimeEnd i dateTimeStart.
plannedCost	FLOAT	Planirani trošak.

Task		
realizedCost	FLOAT	Realizirani trošak.
plannedProfit	FLOAT	Planirana dobit.
realizedProfit	FLOAT	Realizirana dobit.
idJob	INTEGER	Identifikator djelatnosti kojoj zadatak pripada.
idLocation	INTEGER	Identifikator lokacije na kojoj se zadatak obavlja.

EmployeeTask

Tablica koja ostvaruje vezu više-na-više između tablica Employee i Task, tj. određuje koji djelatnik radi na kojim zadacima. Zapisuje se i realizacija zadatka koja predstavlja postotak dovršenog posla. Na početku iznosi 0 i povećava se kako djelatnik napreduje s poslom.

EmployeeTask		
idEmployee	CHAR	Identifikator djelatnika koji radi na zadatku.
idTask	INTEGER	Identifikator zadatka na kojem djelatnik radi.
realized	SMALLINT	Postotak posla kojeg je djelatnik obavio na zadatku u odnosu na ukupan posao. Može biti u intervalu [0, 100]. Početna vrijednost je 0.

Location

Lokacije na kojima se obavljaju poslovi. Za svaku lokaciju zapisuje se adresa, mjesto i geografska širina i dužina radi prikaza na karti.

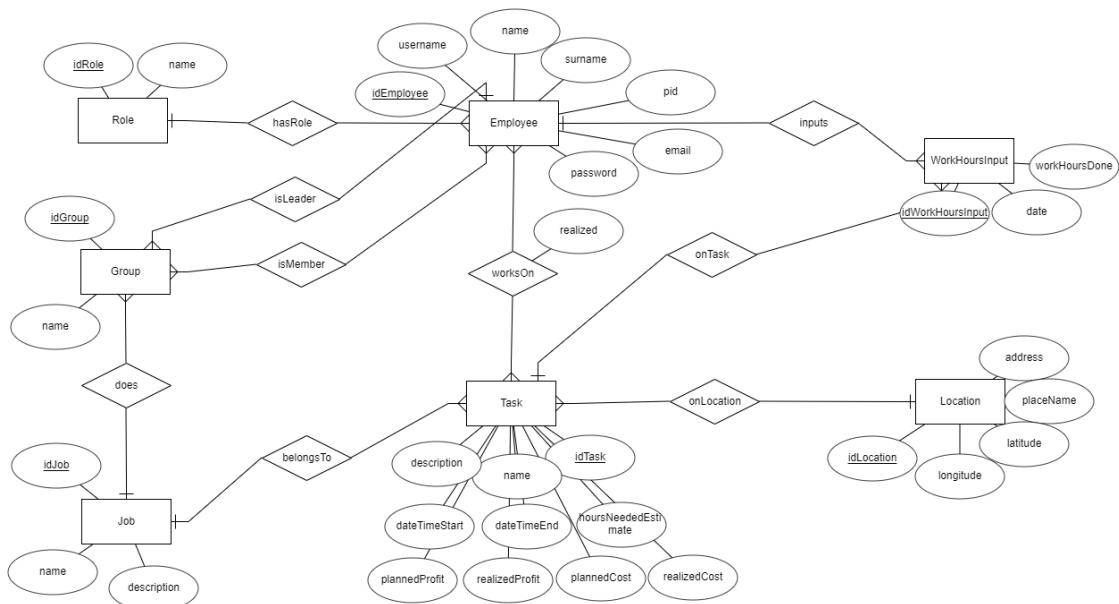
Location		
idLocation	INTEGER	Identifikator lokacije.
address	VARCHAR	Adresa na kojoj se lokacija nalazi.
placeName	VARCHAR	Mjesto u kojem se lokacija nalazi.
latitude	DOUBLE	Geografska širina lokacije.
longitude	DOUBLE	Geografska dužina lokacije.

WorkHoursInput

Tablica koja predstavlja evidenciju radnih sati djelatnika. Svaki djelatnik upisuje koliko je sati odradio na kojem zadatku svaki radni dan.

WorkHoursInput		
idWorkHoursInput	INTEGER	Identifikator unosa.
date	DATE	Datum unosa.
workHoursDone	SMALLINT	Broj radnih sati koje je djelatnik odradio na zadatku tog dana.
idEmployee	CHAR	Identifikator djelatnika koji obavlja unos.
idTask	INTEGER	Identifikator zadatka na kojeg se unos odnosi.

4.1.2 Dijagram baze podataka



Slika 4.1: ER dijagram baze podataka

Slika 4.2: Relacijski dijagram baze podataka

4.2 Dijagram razreda

Na slikama su prikazani razredi koji pripadaju backend dijelu aplikacije. Dijagramme sačinjavaju razredi te među njima uočavamo dijelove poslužiteljskog sloja, odnosno entitete, objekte za prijenos podataka (engl. Data Transfer Object), rezervorije, servise i kontrolere.

Entiteti su klase koje predstavljaju tablice spremljene u bazi podataka. Svaka instanca entiteta predstavlja jedan redak u toj tablici.

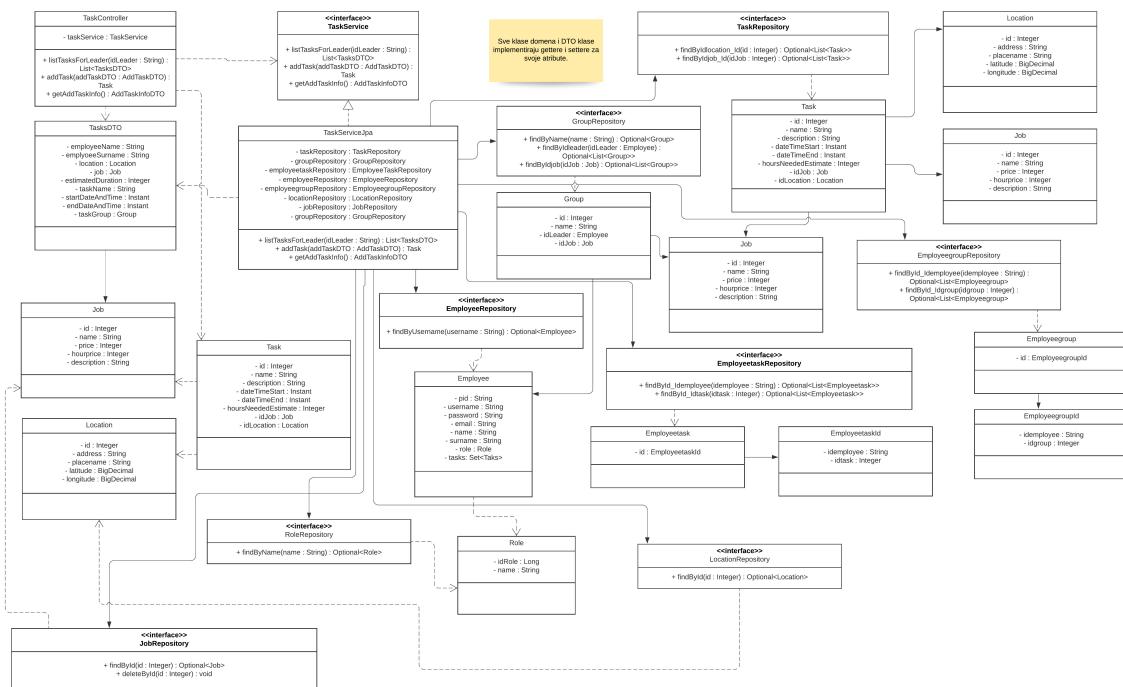
Objekti za prijenos podataka su objekti koji se koriste za komunikaciju između poslužiteljskog i klijentskog dijela aplikacije. Korisni su jer pomoću njih prenosimo samo ono što je bitno, isključivo nužne podatke o određenim entitetima, no ne i cijele entitete.

Rezervoriji su Java sučelja koje upravljaju tokom podataka između poslužitelja i baze podataka. Klase rezervorija služe kao mehanizam za enkapsulaciju pothrane, dohvatanja i pretraživanja podataka.

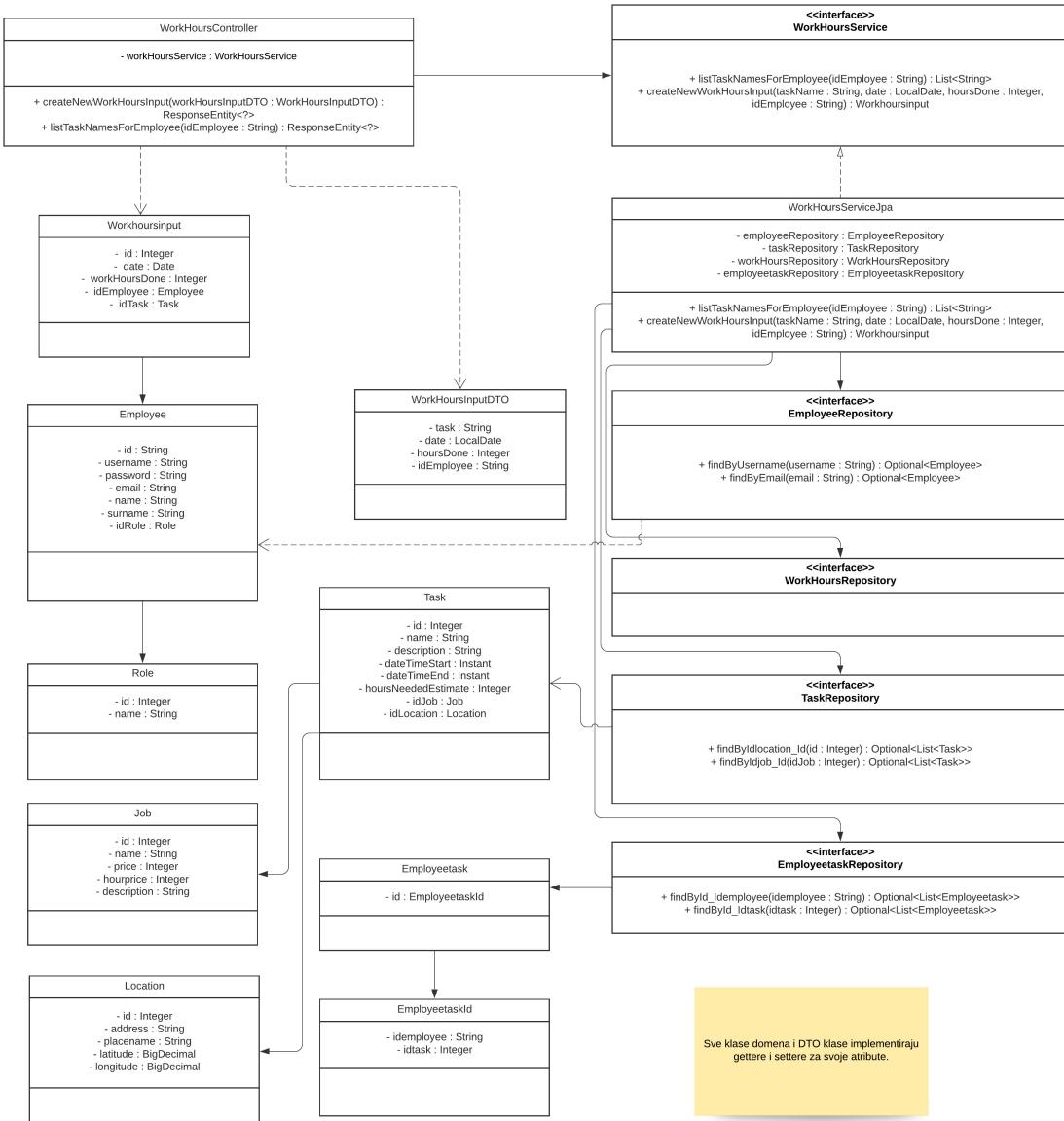
Servisi su klase koje sadrže poslovnu logiku aplikacije. U njima se nalaze metode čija je zadaća izvršavanje operacija nad bazom podataka, odnosno pozivanje metoda definiranih u rezervorijima.

Kontroleri, odnosno REST (engl. Representational State Transfer) kontroleri presreću dolazeće zahtjeve, pretvaraju podatke iz zahtjeva u unutarnju strukturu podataka, šalju podatke modelu na daljnju obradu te dobivaju određene podatke iz modela i prosljeđuju ih u klijentski dio aplikacije kako bi se prikazali korisniku.

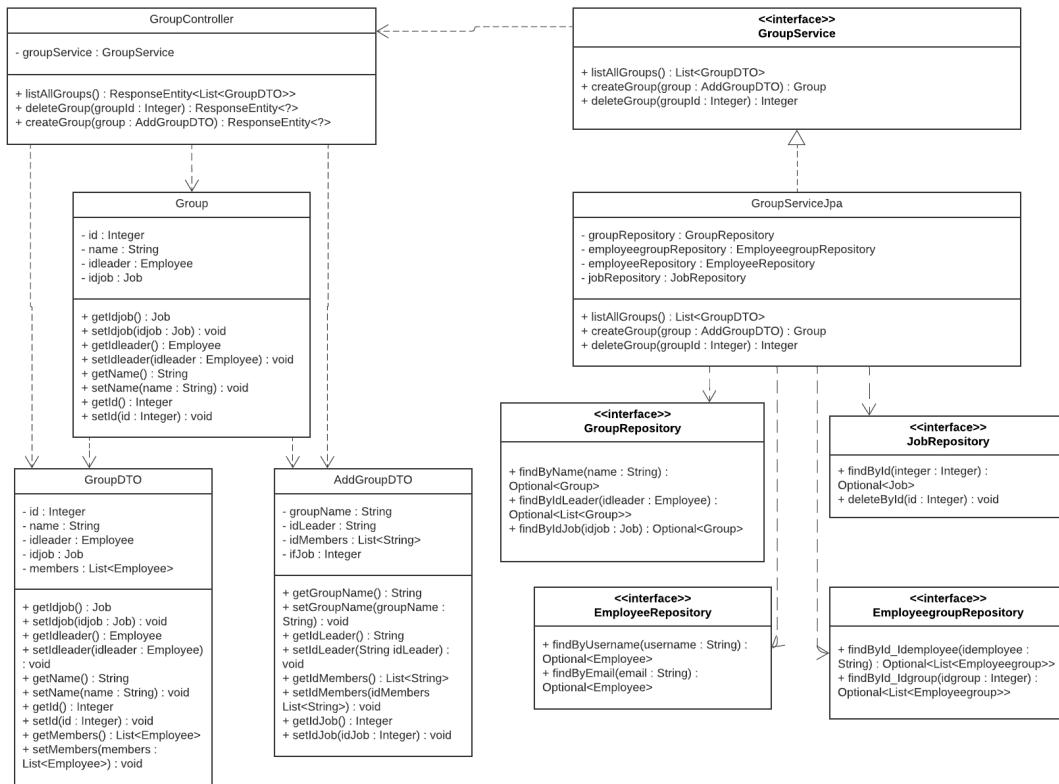
Iz naziva i tipova atributa u razredima može se zaključiti vrsta ovisnosti među različitim razredima prikazanima na priloženim dijagramima.



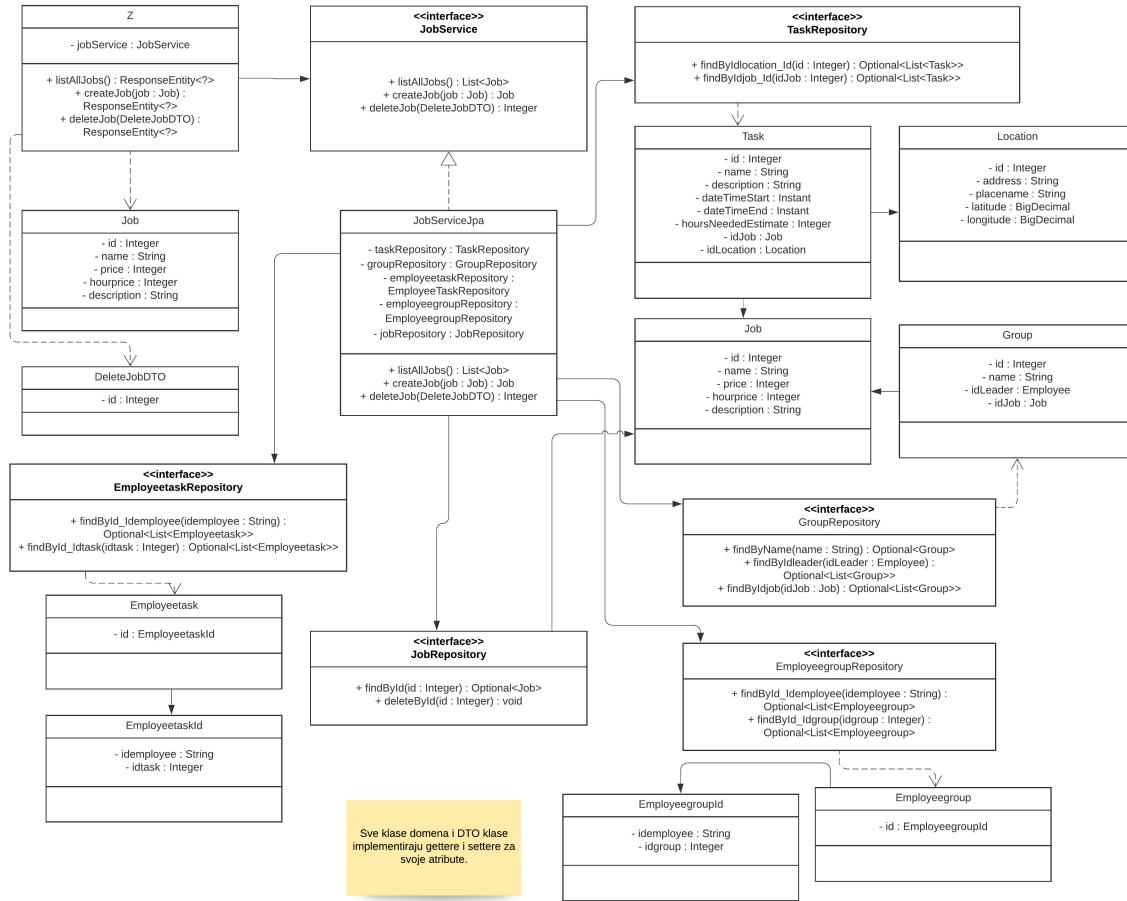
Slika 4.3: Dijagram razreda TaskController



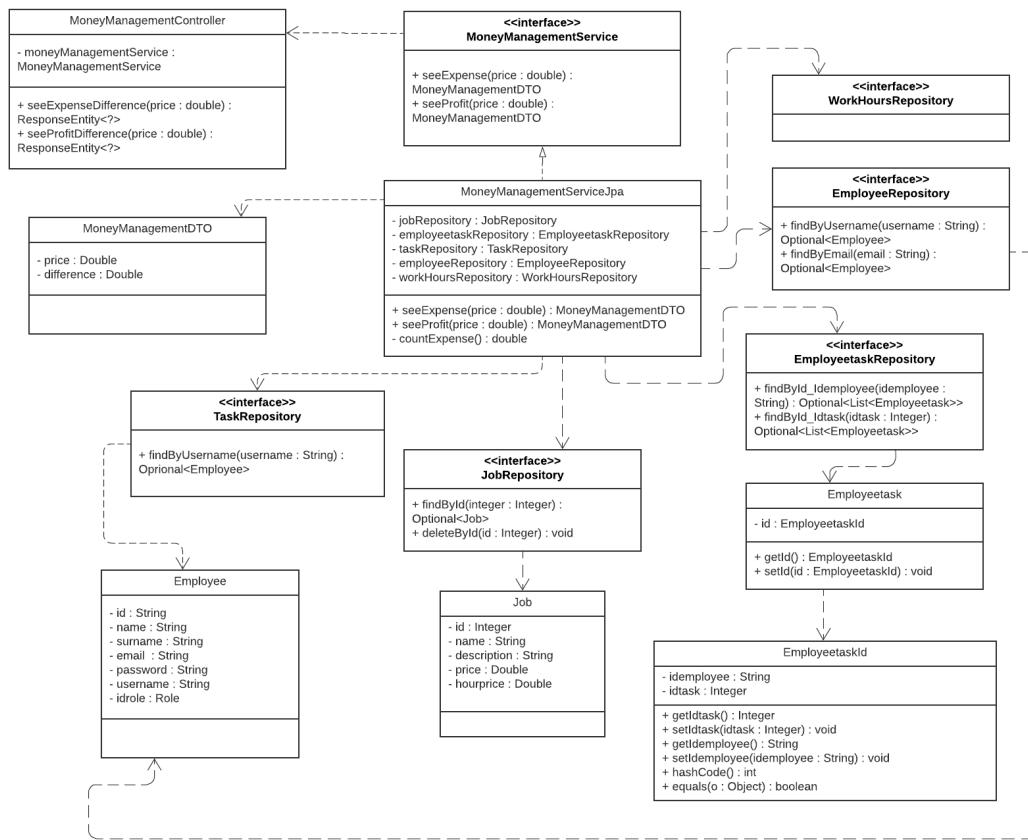
Slika 4.4: Dijagram razreda WorkHoursController



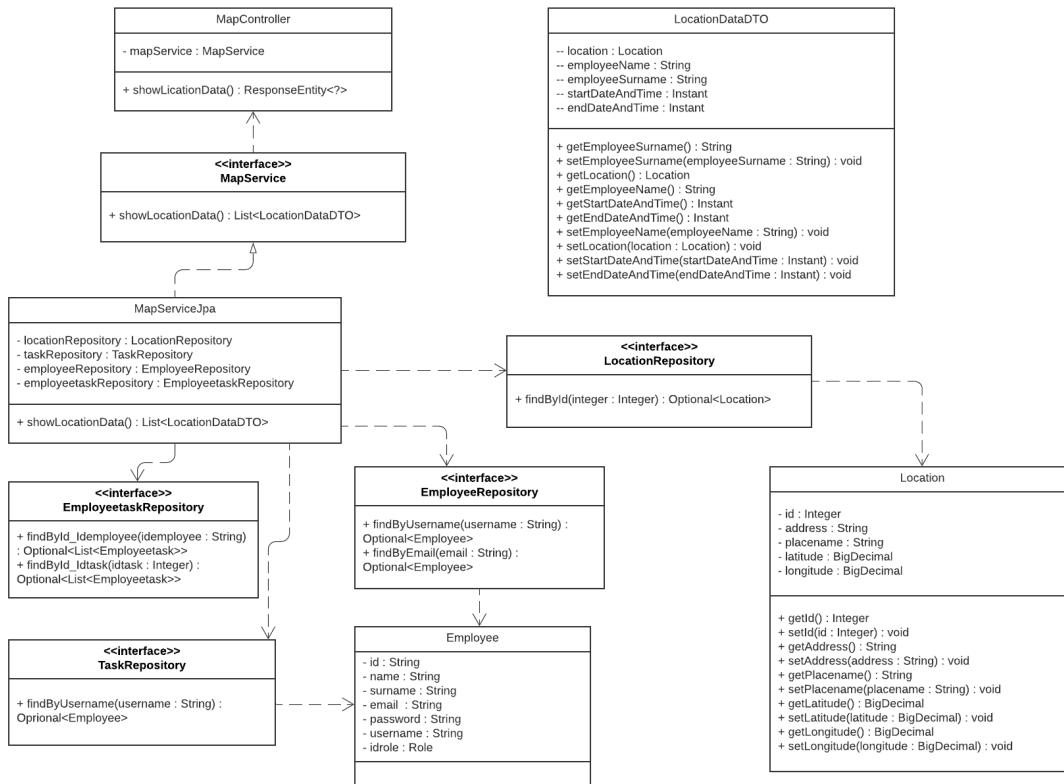
Slika 4.5: Dijagram razreda GroupController



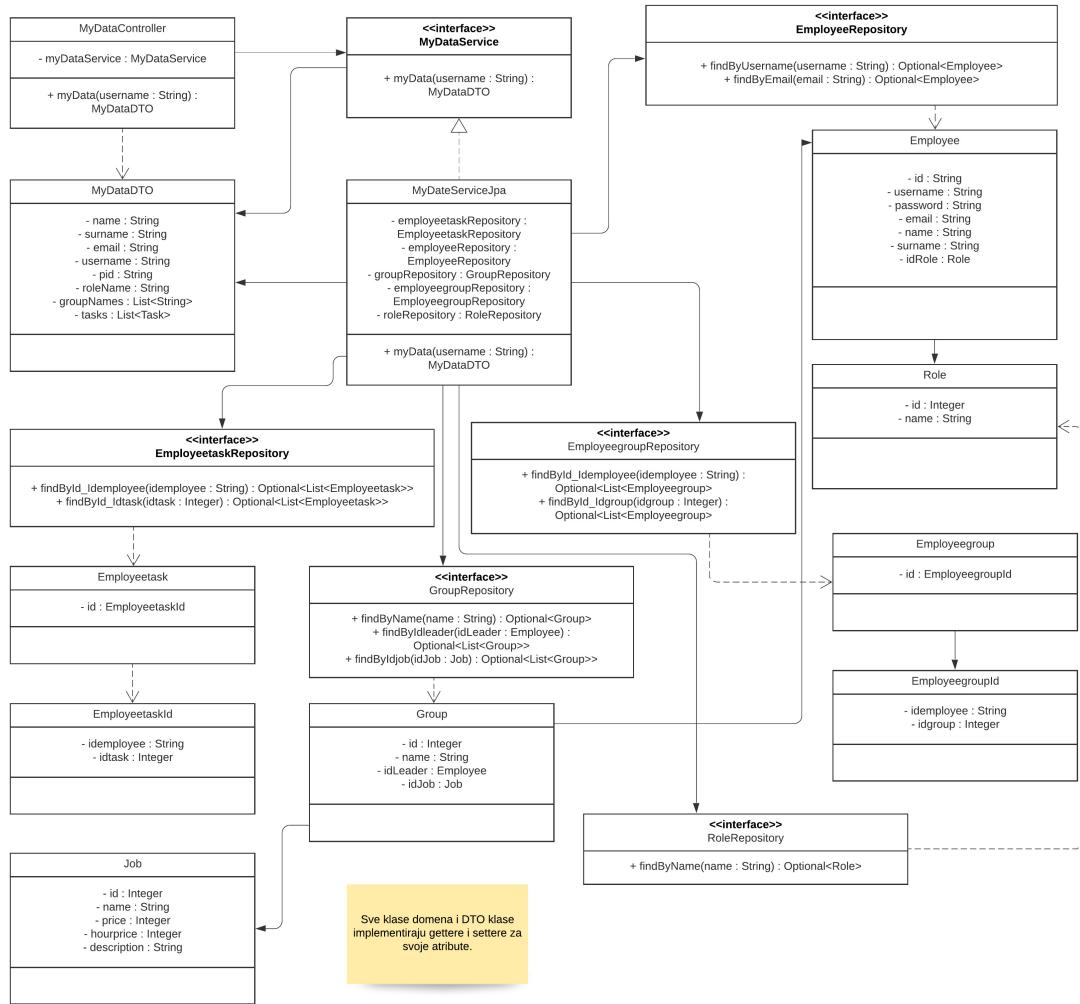
Slika 4.6: Dijagram razreda JobController



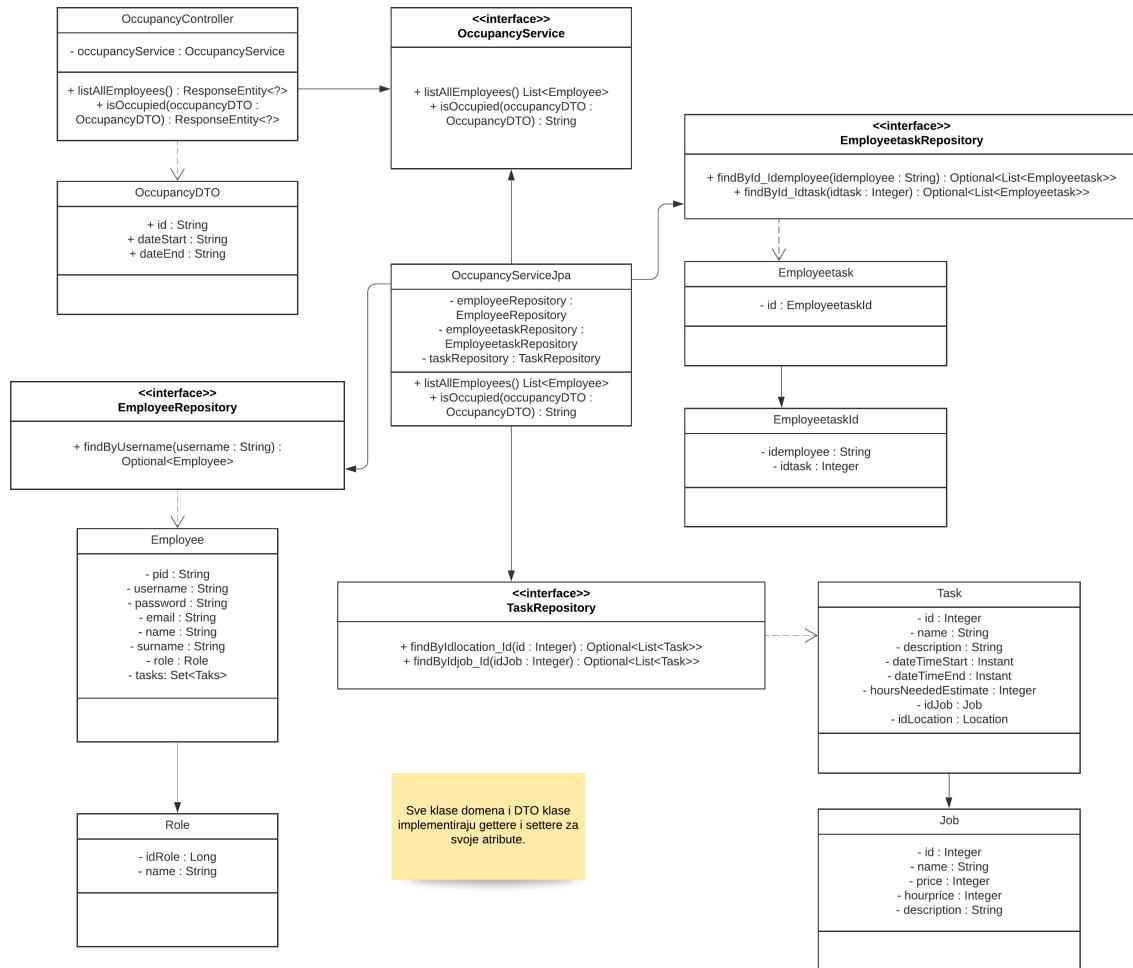
Slika 4.7: Dijagram razreda MoneyManagementController



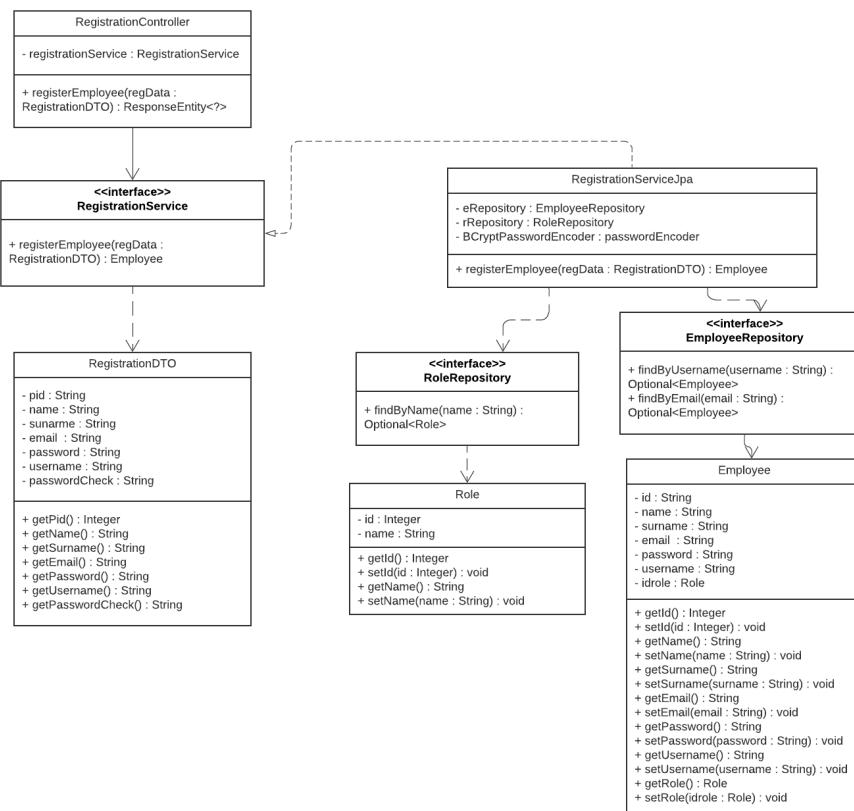
Slika 4.8: Dijagram razreda MapController



Slika 4.9: Dijagram razreda MyDataController



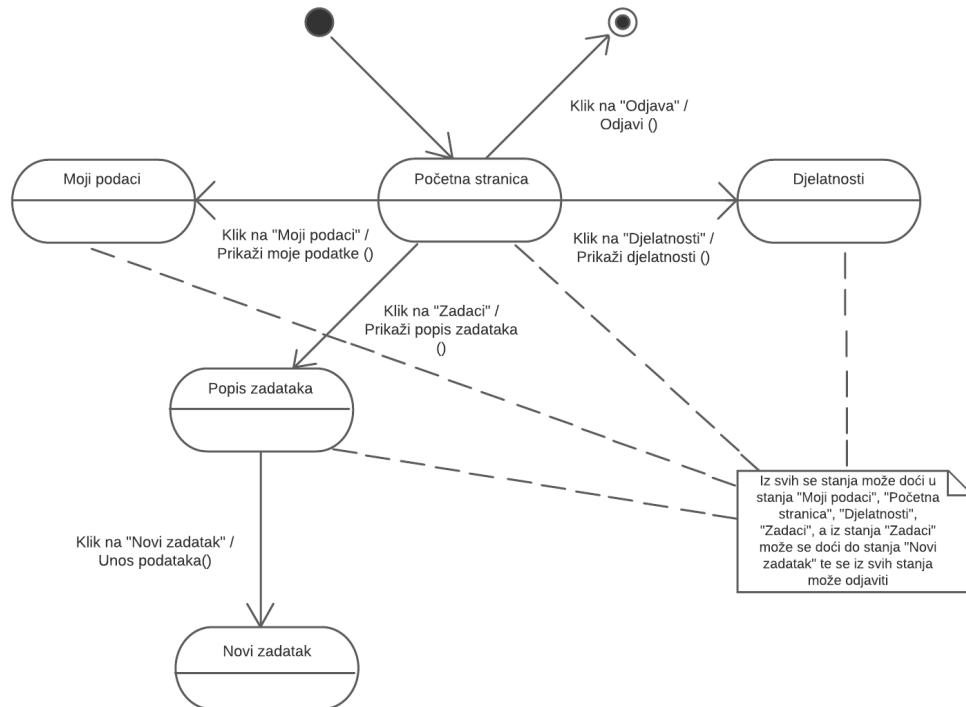
Slika 4.10: Dijagram razreda OccupancyController



Slika 4.11: Dijagram razreda RegistrationController

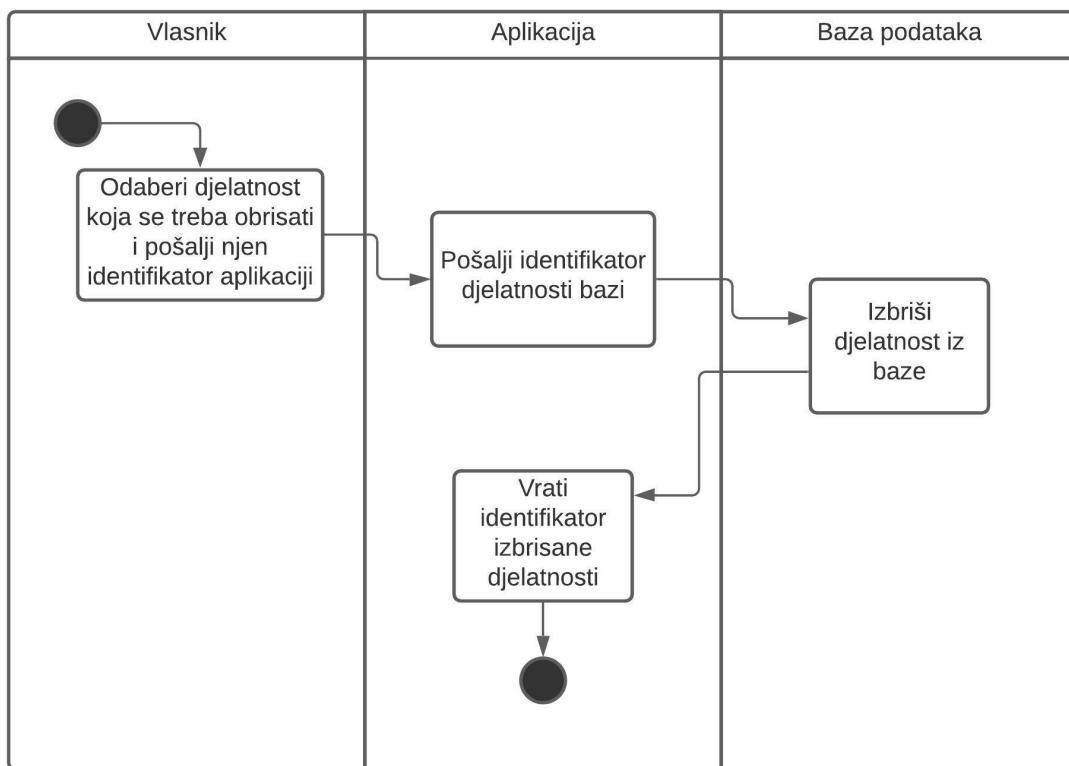
4.3 Dijagram stanja

stm Dijagram stanja : Voditelj

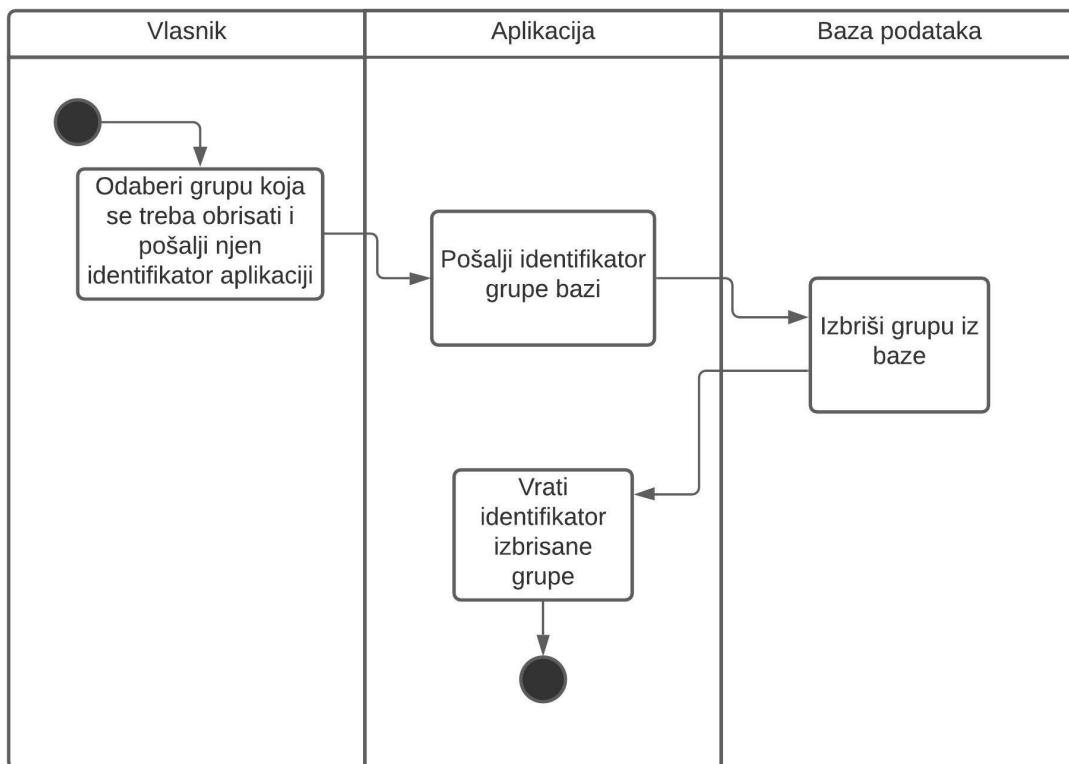


Slika 4.12: Dijagram stanja - Voditelj

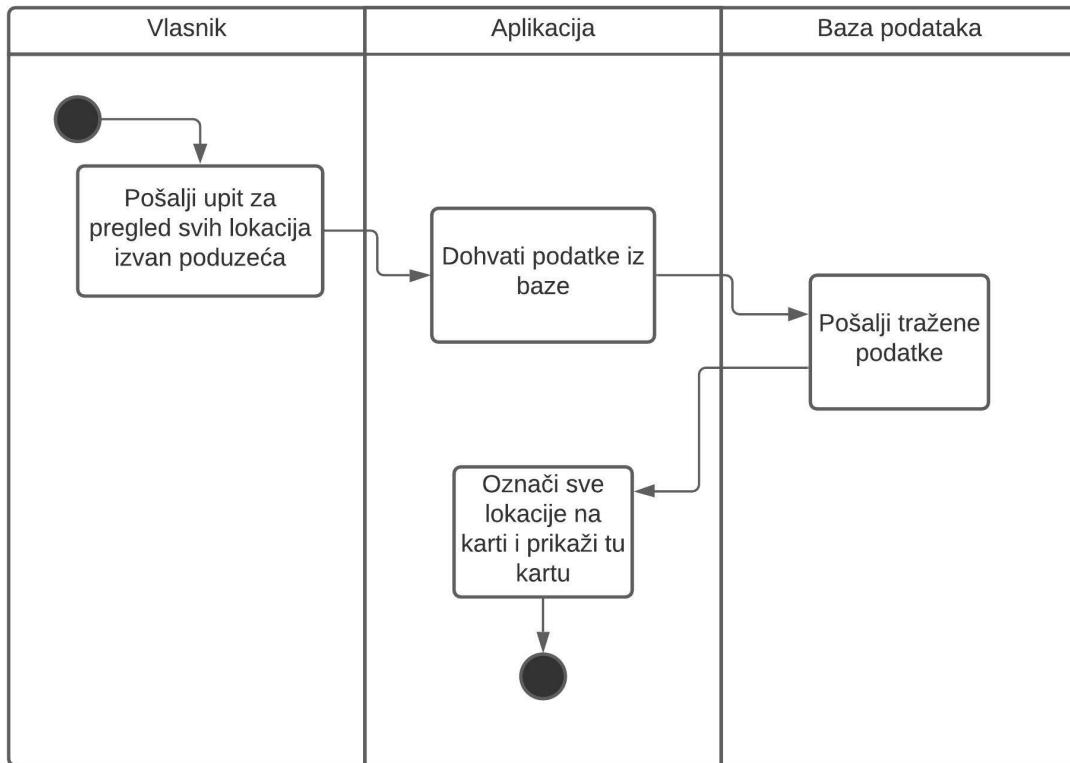
4.4 Dijagram aktivnosti



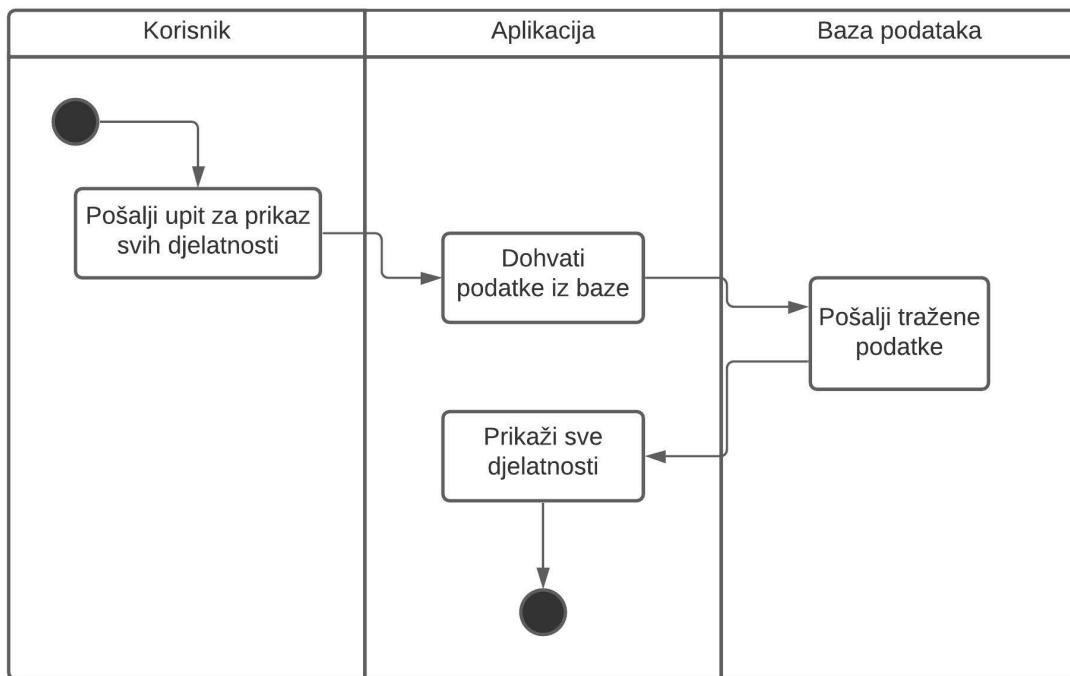
Slika 4.13: Brisanje djelatnosti



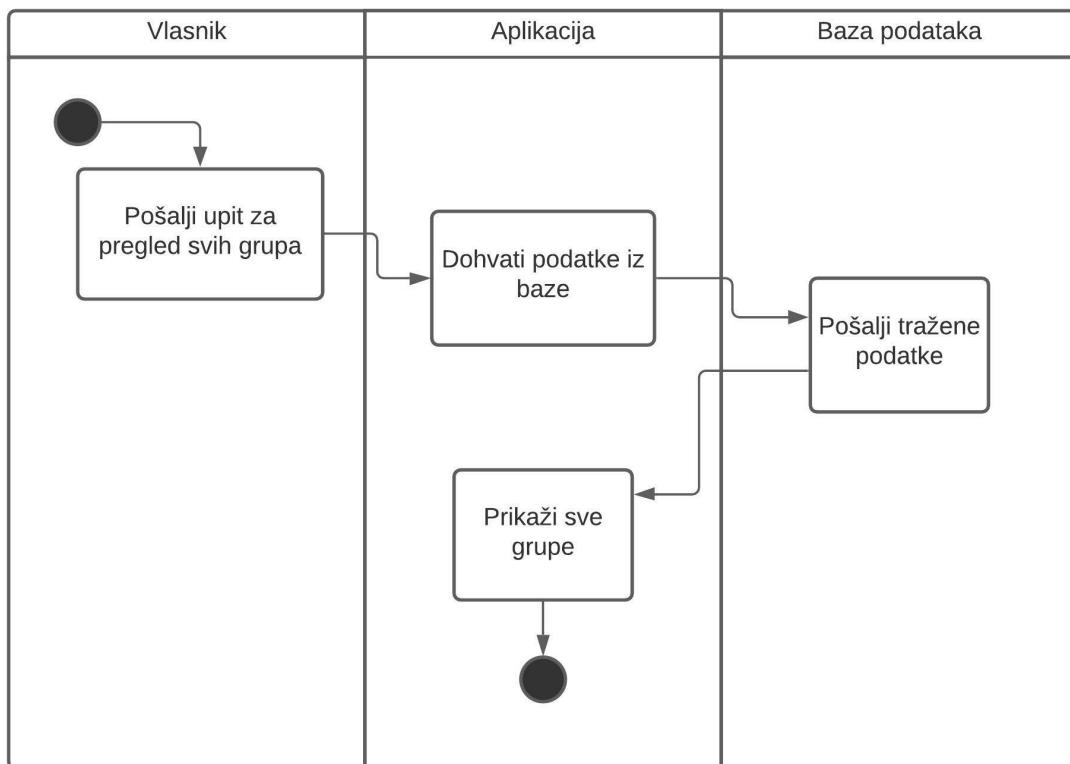
Slika 4.14: Brisanje grupe



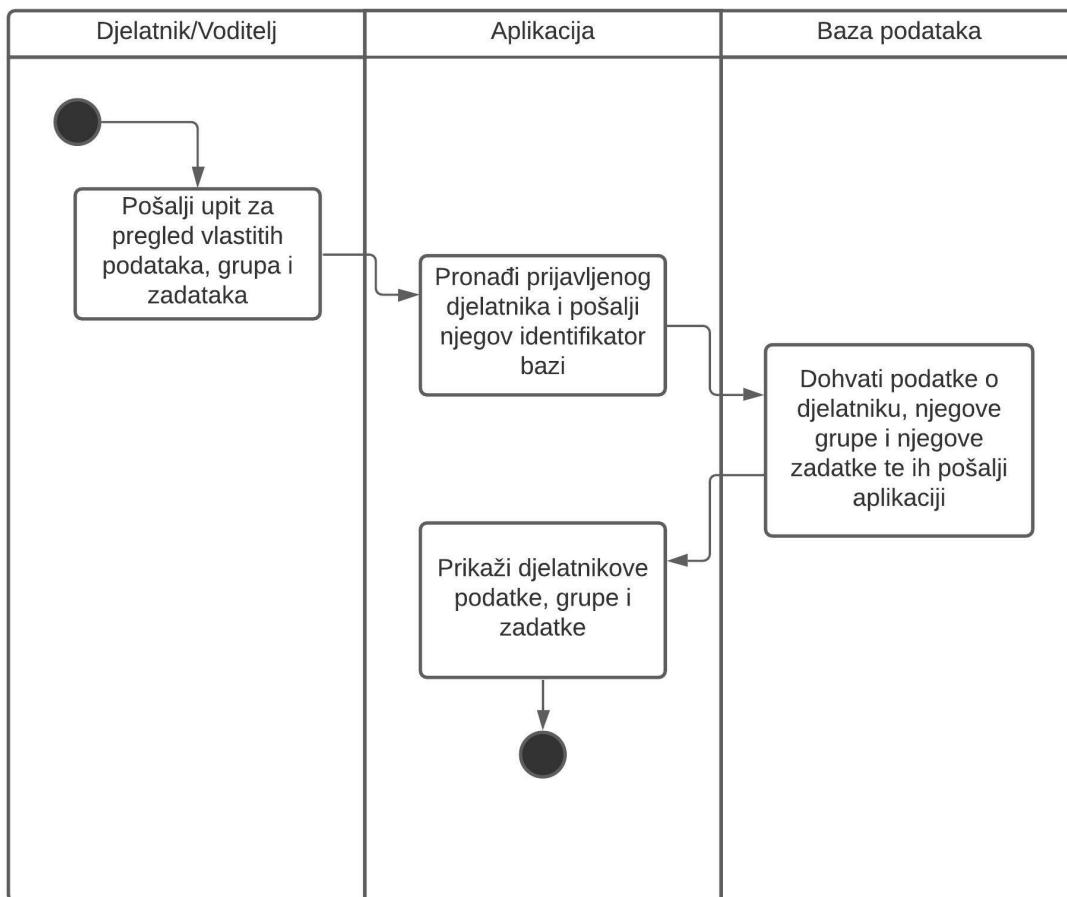
Slika 4.15: Pregled lokacija



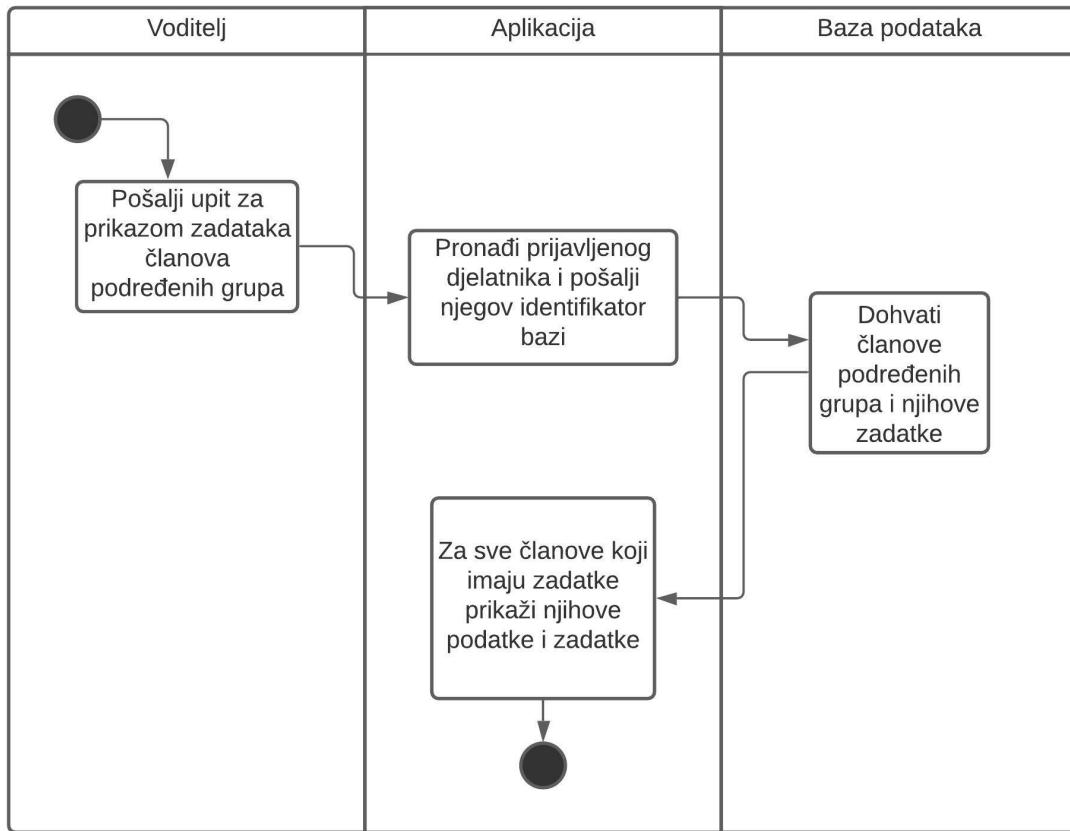
Slika 4.16: Pregled svih djelatnosti



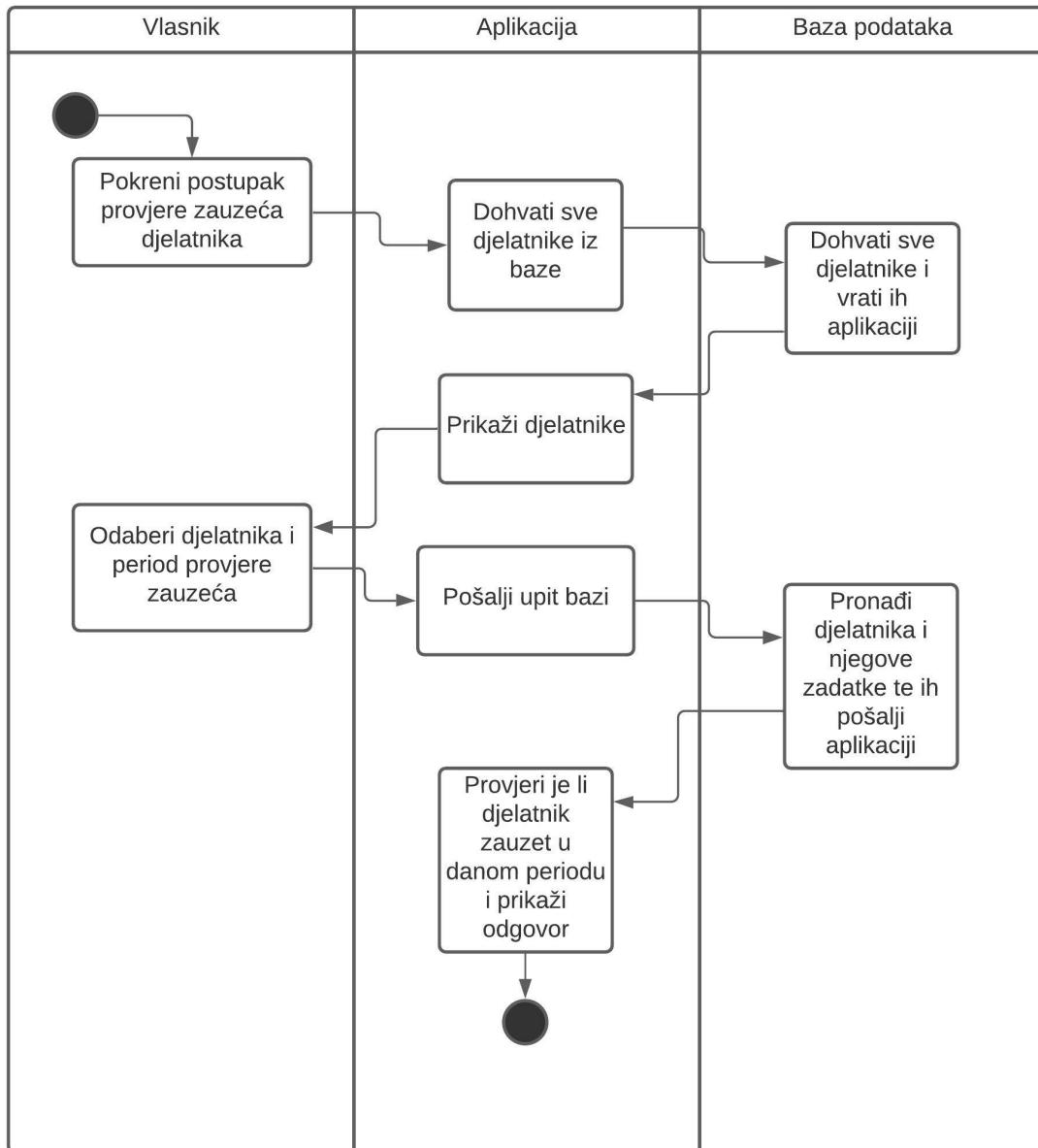
Slika 4.17: Pregled svih grupa



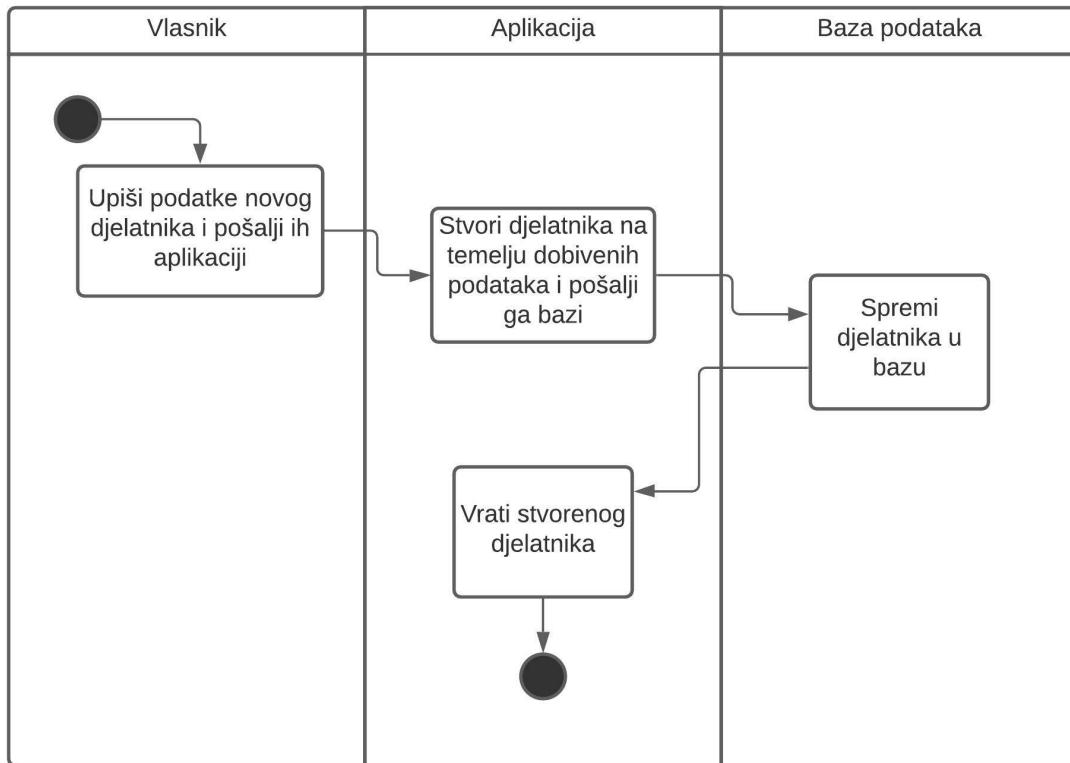
Slika 4.18: Pregled vlastitih podataka



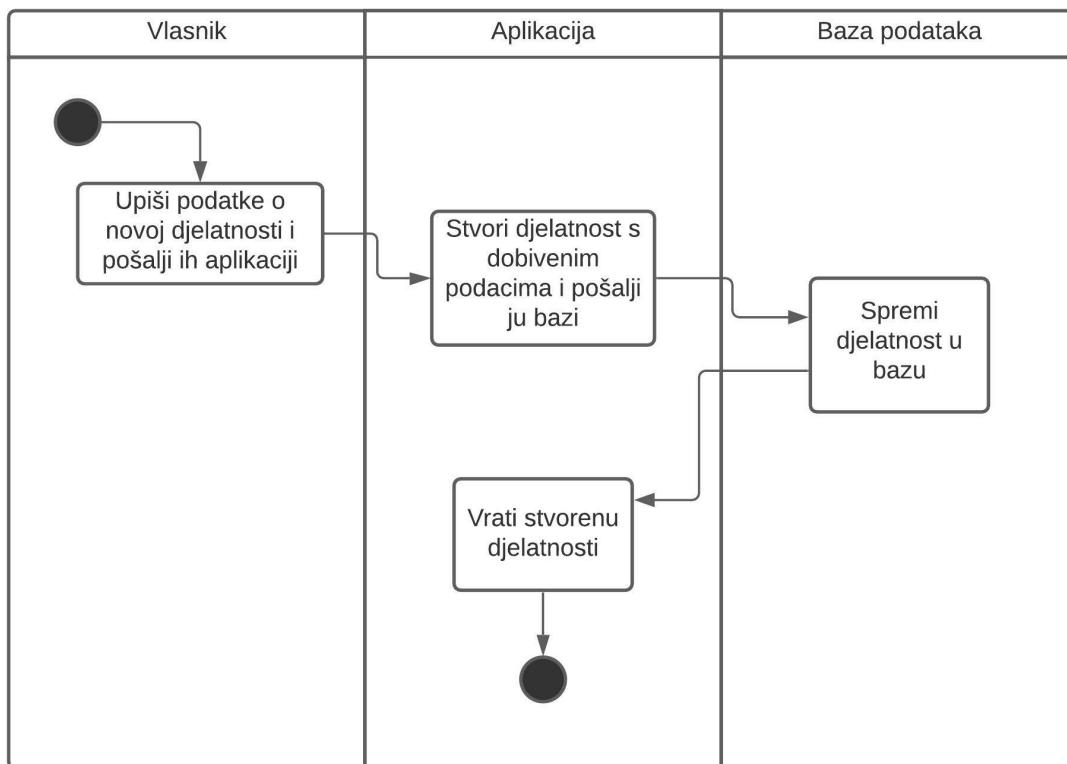
Slika 4.19: Pregled zadataka



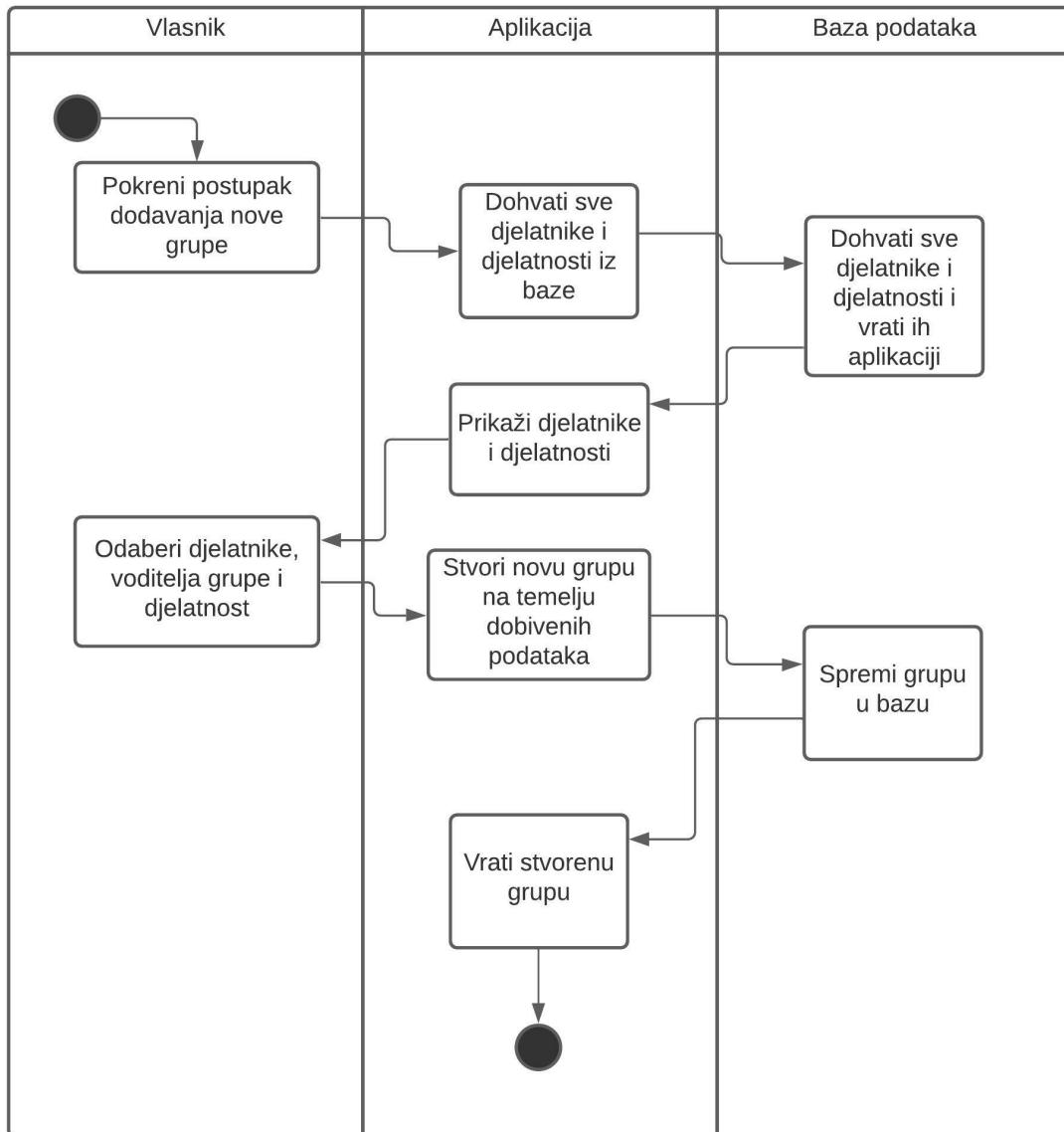
Slika 4.20: Provjera zauzeća djelatnika



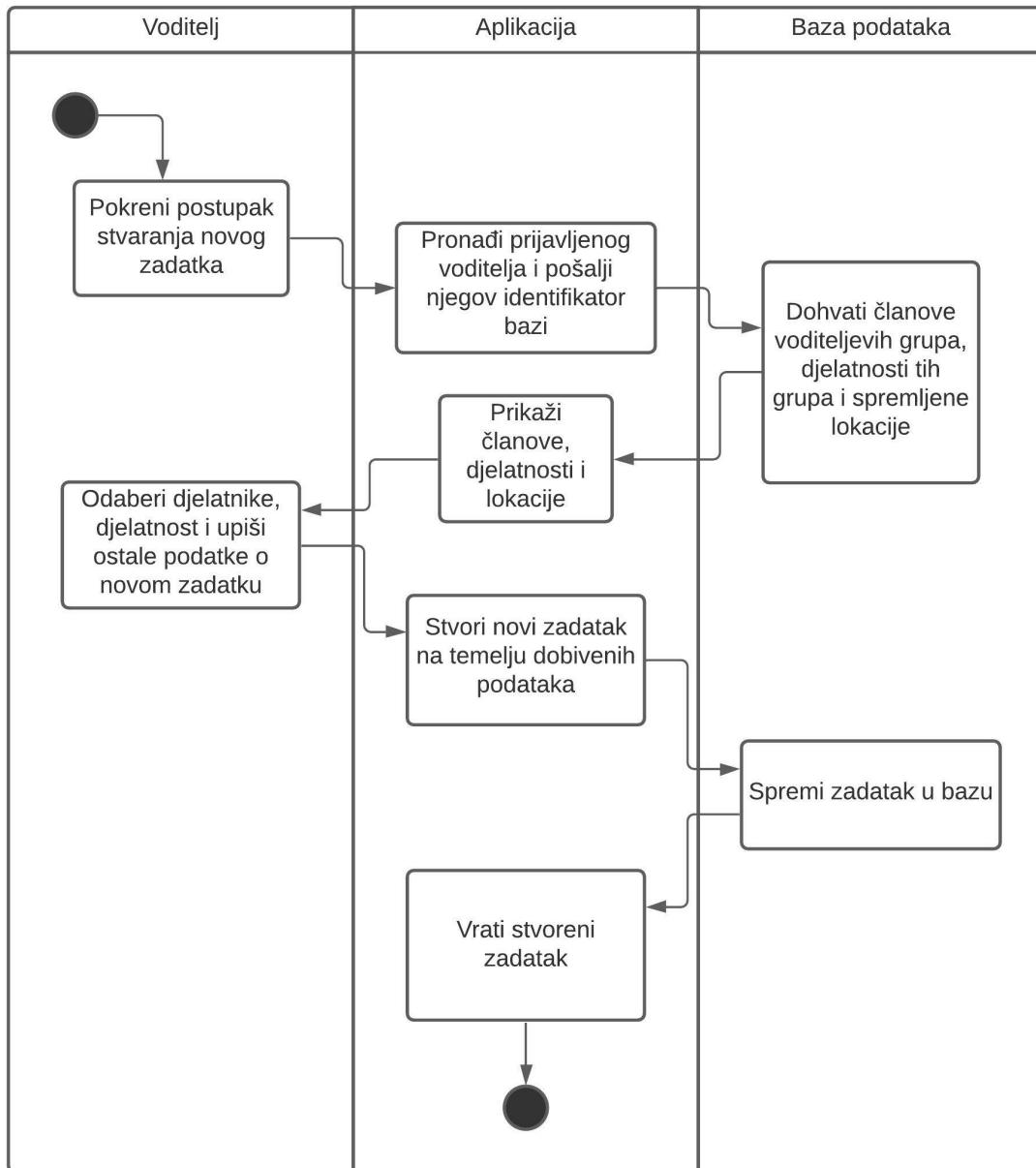
Slika 4.21: Registracija



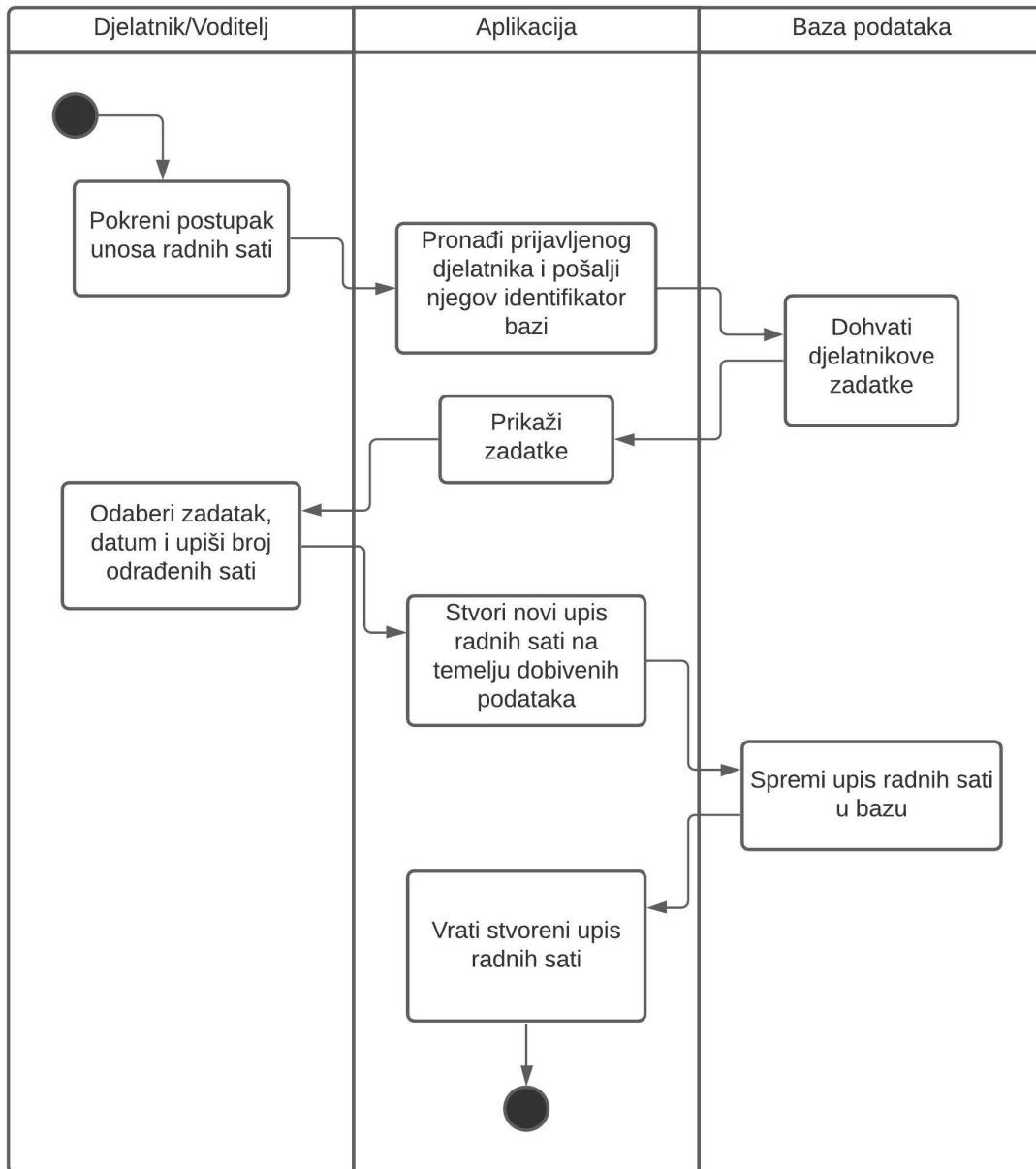
Slika 4.22: Stvaranje djelatnosti



Slika 4.23: Stvaranje grupe

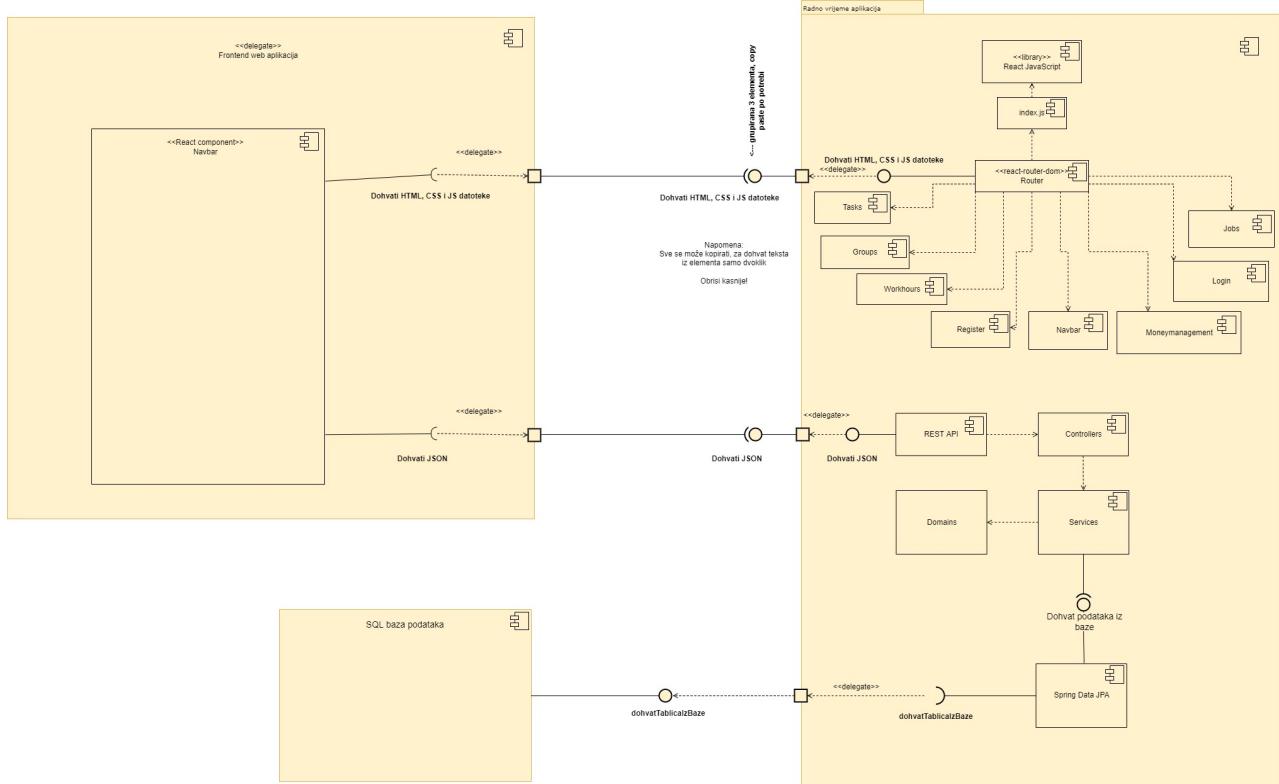


Slika 4.24: Stvaranje novog zadatka



Slika 4.25: Unos radnih sati

4.5 Dijagram komponenti



Slika 4.26: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u teamu realizirana je korištenjem aplikacija Whatsapp i Discord. Organizacija rada i podjela na zadatke među članovima projekta ostvarene su pomoću Microsoft SharePointa i Trello-a, a sastanci su održavani preko platforme Microsoft Teams. Za izradu UML dijagrama korišten je alat Astah UML, za izradu grafova aplikacija Lucidchart, a kao sustav za upravljanje izvornim kodom Git. Udaljeni je repozitorij projekta dostupan na web platformi GitLab. Korištena je i platforma Heroku kao servisni oblak koji nudi razvoj web aplikacija i usluga. Kao razvojno okruženje korišten je IntelliJ IDEA Ultimate. To je višestruka platforma za Java koja se može proširiti brojnim dodacima koji će ju učiniti još cjelovitijom. IntelliJ IDEA također pruža pomoć u kodiranju za širok spektar drugih jezika poput SQL-a, HTML-a i JavaScripta te predviđa potebe korisnika i automatizira zamorne i ponavljuće zadatke kako bi mogao ostati fokusiran na širu sliku projekta. Aplikacija je napisana koristeći radni okvir Spring Boot u programskom jeziku Javi za izradu backenda, React, skriptni programski jezik JavaScript, platforma Node.js, mrežni okvir s otvorenim kodom Bootstrap, prezentacijski jezik HTML te skriptni jezik CSS za izradu frontenda. Za izradu mapa korištena je aplikacija HERE WeGo. Upravljanje bazom podataka smo pomoću sustava PostgreSQL. React, također poznat kao React.js ili ReactJS, je biblioteka u JavaScriptu za izgradnju korisničkih sučelja. Održavana je od strane Facebooka. React se najčešće koristi kao osnova u razvoju web ili mobilnih aplikacija. Složene aplikacije u Reactu obično zahtijevaju korištenje dodatnih biblioteka za interakciju s API-jem. Neke od karakteristika radnog okvira Spring Boot su: unaprijed pripremljene funkcionalnosti (eng. Out of the box functionalities), nema generiranja klase i koda već se koriste unaprijed definirane biblioteke te set nefunkcionalnih alata i klasa u pogledu konfiguracije i pokretanja servera, sigurnosti, metrike te ostalih pomoćnih poslova.

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

```
@Test
public void testListAllGroups() {
    if (groupService.listAllGroups().size() != 4)
        Assertions.fail();
}

@Test
public void testNonExistingEmployee() {
    AddGroupDTO addGroupDTO = new AddGroupDTO();
    addGroupDTO.setGroupName("name");
    addGroupDTO.setIdJob(1);
    List<String> members = new ArrayList<>();
    members.add("1111111111");
    addGroupDTO.setIdMembers(members);
    addGroupDTO.setIdLeader("1111111111");
    try {
        groupService.createGroup(addGroupDTO);
    } catch (IllegalArgumentException e) {
        return;
    }
    Assertions.fail();
}

@Test
public void testDuplicateName() {
    AddGroupDTO addGroupDTO = new AddGroupDTO();
    addGroupDTO.setGroupName("Group 1");
    addGroupDTO.setIdJob(1);
    List<String> members = new ArrayList<>();
    members.add("0000000001");
    addGroupDTO.setIdMembers(members);
    addGroupDTO.setIdLeader("0000000001");
    try {
        groupService.createGroup(addGroupDTO);
    } catch (IllegalArgumentException e) {
        return;
    }
    Assertions.fail();
}
```

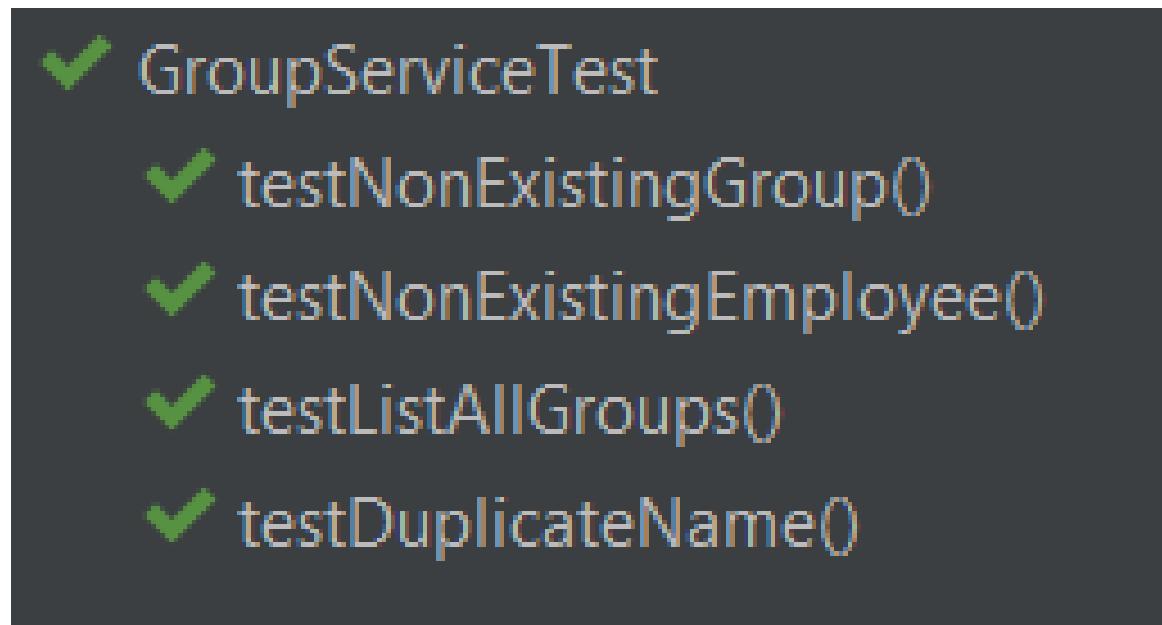
Slika 5.1: GroupServiceTest 1. dio

```
@Test
public void testCreateGroup() {
    AddGroupDTO addGroupDTO = new AddGroupDTO();
    addGroupDTO.setGroupName("newGroup2");
    addGroupDTO.setIdJob(1);
    List<String> members = new ArrayList<>();
    members.add("0000000001");
    addGroupDTO.setIdMembers(members);
    addGroupDTO.setIdLeader("0000000001");
    Group g = groupService.createGroup(addGroupDTO);
    if (!g.getName().equals("newGroup2"))
        Assertions.fail();
}

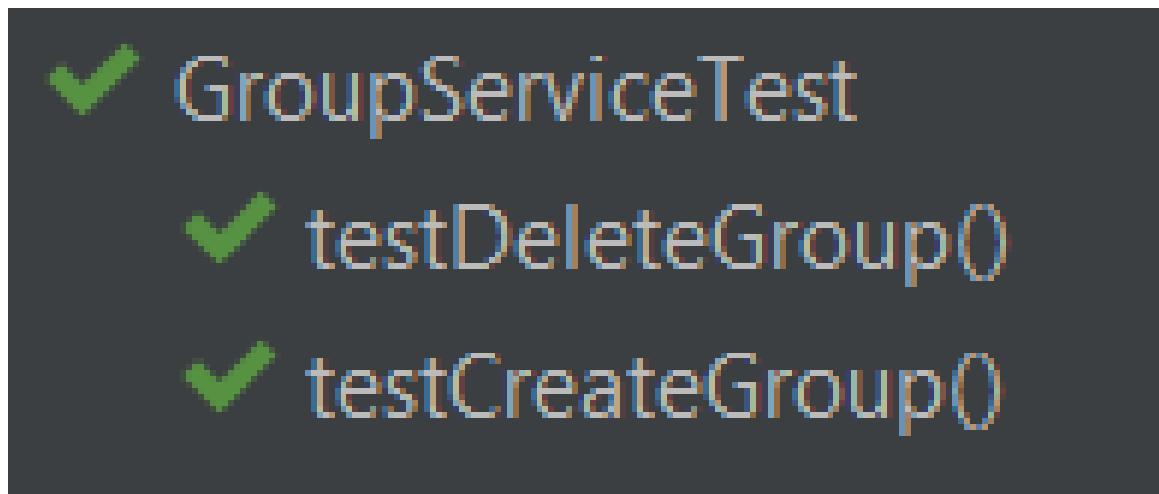
@Test
public void testNonExistingGroup() {
    Assertions.assertThrows(MissingGroupException.class, () -> groupService.deleteGroup(groupId: 10));
}

@Test
public void testDeleteGroup() {
    if (groupService.deleteGroup(groupId: 7) != 7)
        Assertions.fail();
}
```

Slika 5.2: GroupServiceTest 2. dio



Slika 5.3: GroupServiceTest rezultati 1. dio



Slika 5.4: GroupServiceTest rezultati 2. dio

```
@Test
public void testDeleteJob() {
    Integer id = jobService.deleteJob( id: 3);
    if (!id.equals(Integer.valueOf(3)))
        Assertions.fail();
}

@Test
public void testDeleteNonExistingJob() {
    Assertions.assertThrows(EntityMissingException.class, () -> jobService.deleteJob( id: 2));
}

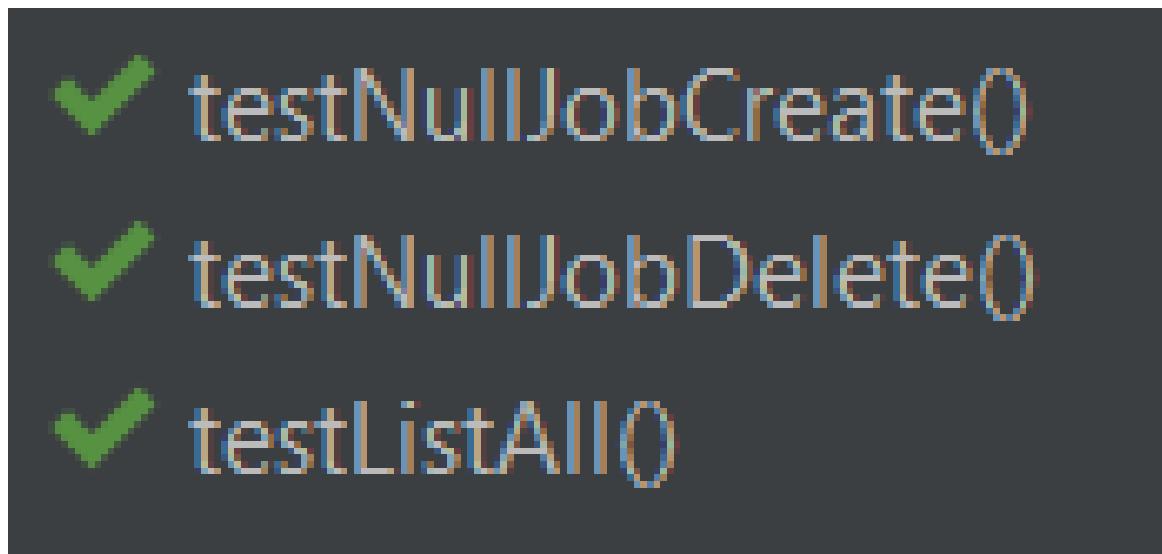
@Test
public void testCreateJob() {
    Job job = new Job();
    job.setDescription("desc");
    job.setHourprice(30.0);
    job.setName("name");
    job.setPrice(30.0);
    Integer id = jobService.createJob(job).getId();
    if (!id.equals(9))
        Assertions.fail();
}

@Test
public void testListAll() {
    if (jobService.listAll().size() != 5)
        Assertions.fail();
}

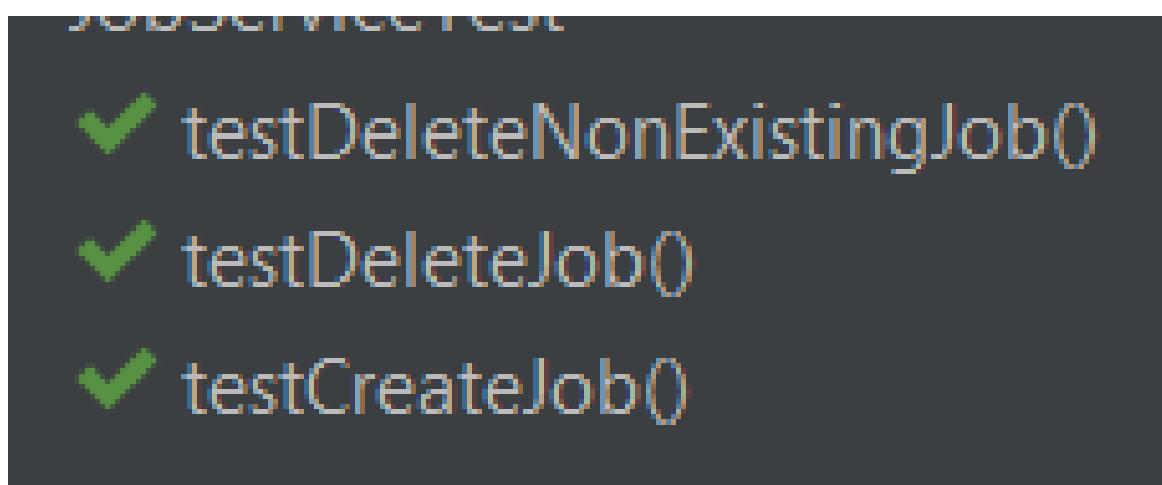
@Test
public void testNullJobCreate() {
    Assertions.assertThrows(InvalidDataAccessApiUsageException.class, () -> jobService.createJob(null));
}

@Test
public void testNullJobDelete() {
    Assertions.assertThrows(InvalidDataAccessApiUsageException.class, () -> jobService.deleteJob( id: null));
}
```

Slika 5.5: JobServiceTest



Slika 5.6: JobServiceTest rezultati 1. dio



Slika 5.7: JobServiceTest rezultati 2. dio

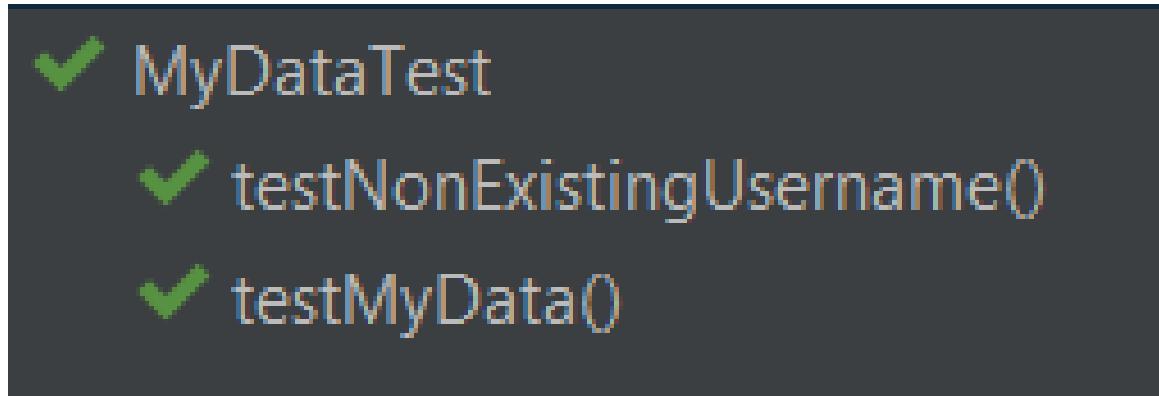
```

    @Test
    public void testNonExistingUsername() {
        Assertions.assertThrows(MissingEmployeeException.class, () -> myDataService.myData(username: "nonexistingusername"));
    }

    @Test
    public void testMyData() {
        MyDataDTO myDataDTO = myDataService.myData(username: "hWang");
        if (myDataDTO.getGroupNames().size() != 1 || myDataDTO.getTasks().size() != 4)
            Assertions.fail();
    }
}

```

Slika 5.8: MyDataServiceTest



Slika 5.9: MyDataServiceTest rezultati

```

    @Test
    public void testListAllEmployees() {
        if (occupancyService.listAllEmployees().size() != 21)
            Assertions.fail();
    }

    @Test
    public void testNonExistingEmployee() throws ParseException {
        SimpleDateFormat formatter = new SimpleDateFormat(pattern: "dd-MM-yyyy");
        Assertions.assertThrows(MissingEmployeeException.class, () -> occupancyService.isOccupied(id: "111111111111", formatter.parse(source: "01-01-2020"), formatter.parse(source: "01-03-2020")));
    }

    @Test
    public void testEndDateBeforeStartDate() throws ParseException {
        SimpleDateFormat formatter = new SimpleDateFormat(pattern: "dd-MM-yyyy");
        Assertions.assertThrows(TimePeriodException.class, () -> occupancyService.isOccupied(id: "00000000001", formatter.parse(source: "01-03-2020"), formatter.parse(source: "01-01-2020")));
    }

    @Test
    public void testOccupied() throws ParseException {
        SimpleDateFormat formatter = new SimpleDateFormat(pattern: "dd-MM-yyyy");
        if (!occupancyService.isOccupied(id: "00000000001", formatter.parse(source: "01-03-2020"), formatter.parse(source: "01-03-2020")).equals("Djelatnik je zauzet u odabranom periodu."))
            Assertions.fail();
    }

    @Test
    public void testNotOccupied() throws ParseException {
        SimpleDateFormat formatter = new SimpleDateFormat(pattern: "dd-MM-yyyy");
        if (!occupancyService.isOccupied(id: "00000000001", formatter.parse(source: "01-01-2010"), formatter.parse(source: "01-01-2011")).equals("Djelatnik je slobodan u odabranom periodu."))
            Assertions.fail();
    }
}

```

Slika 5.10: OccupancyServiceTest

```
✓ OccupancyServiceTest
  ✓ testEndDateBeforeStartDate()
  ✓ testNonExistingEmployee()
  ✓ testNotOccupied()
  ✓ testListAllEmployees()
  ✓ testOccupied()
```

Slika 5.11: OccupancyServiceTest rezultati

```
@Test
public void testPasswordDoesntMatch() {
    RegistrationDTO registrationDTO = new RegistrationDTO();
    registrationDTO.setPassword("pass");
    registrationDTO.setPasswordCheck("pass2");
    registrationDTO.setEmail("email@gmail.com");
    registrationDTO.setUsername("username");
    registrationDTO.setName("name");
    registrationDTO.setSurname("surname");
    registrationDTO.setPid("1111111111");
    Assertions.assertThrows(InvalidPasswordCheckException.class, () -> registrationService.registerEmployee(registrationDTO));
}

@Test
public void testDuplicatePID() {
    RegistrationDTO registrationDTO = new RegistrationDTO();
    registrationDTO.setPassword("pass");
    registrationDTO.setPasswordCheck("pass");
    registrationDTO.setEmail("email@gmail.com");
    registrationDTO.setUsername("username");
    registrationDTO.setName("name");
    registrationDTO.setSurname("surname");
    registrationDTO.setPid("0000000001");
    Assertions.assertThrows(IllegalArgumentException.class, () -> registrationService.registerEmployee(registrationDTO));
}

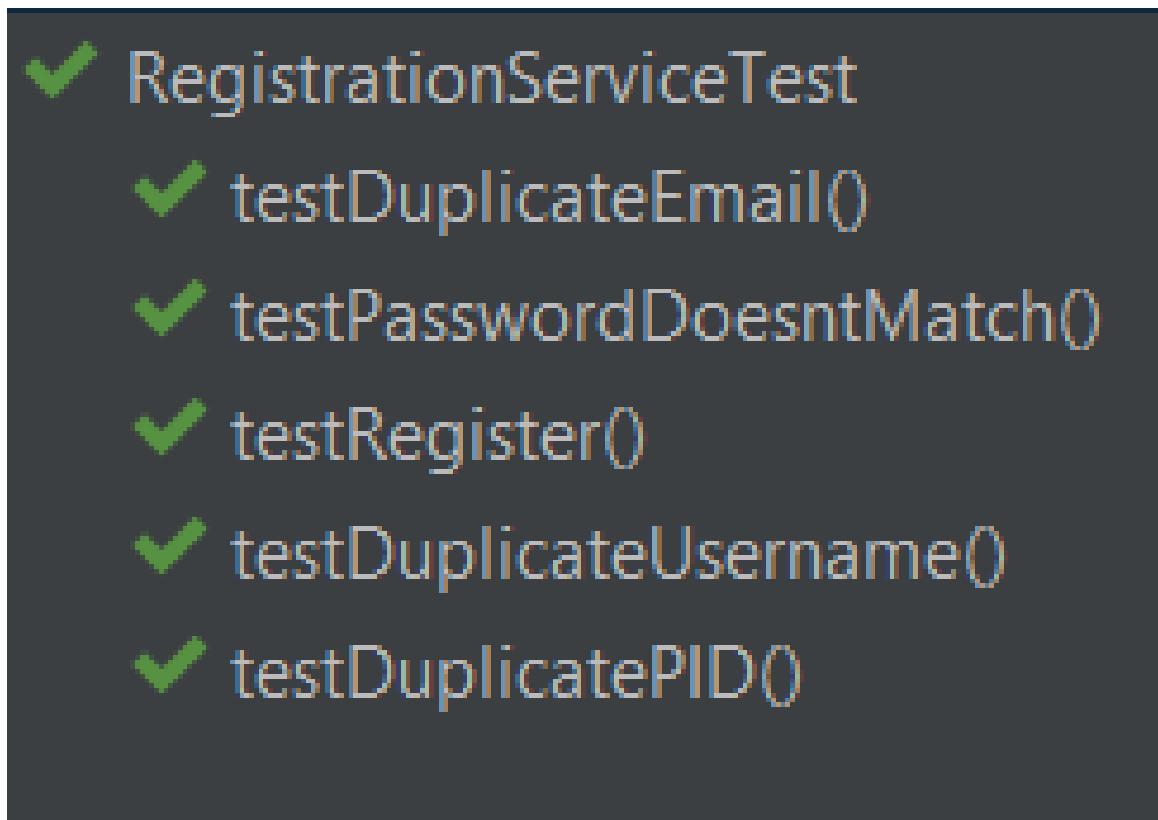
@Test
public void testDuplicateUsername() {
    RegistrationDTO registrationDTO = new RegistrationDTO();
    registrationDTO.setPassword("pass");
    registrationDTO.setPasswordCheck("pass");
    registrationDTO.setEmail("email@gmail.com");
    registrationDTO.setUsername("h\u0107ang");
    registrationDTO.setName("name");
    registrationDTO.setSurname("surname");
    registrationDTO.setPid("1111111111");
    Assertions.assertThrows(IllegalArgumentException.class, () -> registrationService.registerEmployee(registrationDTO));
}
```

Slika 5.12: RegistrationServiceTest

```
@Test
public void testDuplicateEmail() {
    RegistrationDTO registrationDTO = new RegistrationDTO();
    registrationDTO.setPassword("pass");
    registrationDTO.setPasswordCheck("pass");
    registrationDTO.setEmail("neena.wang@email.com");
    registrationDTO.setUsername("username");
    registrationDTO.setName("name");
    registrationDTO.setSurname("surname");
    registrationDTO.setPid("1111111111");
    Assertions.assertThrows(IllegalArgumentException.class, () -> registrationService.registerEmployee(registrationDTO));
}

@Test
public void testRegister() {
    RegistrationDTO registrationDTO = new RegistrationDTO();
    registrationDTO.setPassword("pass");
    registrationDTO.setPasswordCheck("pass");
    registrationDTO.setEmail("email@email.com");
    registrationDTO.setUsername("username");
    registrationDTO.setName("name");
    registrationDTO.setSurname("surname");
    registrationDTO.setPid("1111111111");
    if (!registrationService.registerEmployee(registrationDTO).getId().equals("1111111111"))
        Assertions.fail();
}
```

Slika 5.13: RegistrationServiceTest rezultati 1. dio



Slika 5.14: RegistrationServiceTest rezultati 2. dio

```

@Test
public void testListTasksForEmployee() {
    int size = workHoursService.listTaskNamesForEmployee( idEmployee: "0000000001").size();
    if (workHoursService.listTaskNamesForEmployee( idEmployee: "0000000001").size() != 0)
        Assertions.fail();
}

@Test
public void testNullEmployee() {
    Assertions.assertThrows(IllegalArgumentException.class, () -> workHoursService.listTaskNamesForEmployee( idEmployee: null));
}

@Test
public void testNonExistingEmployeeWhenListingTasks() {
    Assertions.assertThrows(MissingEmployeeException.class, () -> workHoursService.listTaskNamesForEmployee( idEmployee: "1111111111"));
}

@Test
public void testNonExistingEmployeeWhenCreatingInput() {
    Assertions.assertThrows(MissingEmployeeException.class, () -> workHoursService.createNewWorkHoursInput( taskName: "Sustav za navodnjavanje",
        LocalDate.of( year: 2020, month: 2, dayOfMonth: 10), hoursDone: 10, idEmployee: "1111111111"));
}

@Test
public void testInvalidNumberOfHours() {
    Assertions.assertThrows(IllegalArgumentException.class, () -> workHoursService.createNewWorkHoursInput( taskName: "Sustav za navodnjavanje",
        LocalDate.of( year: 2020, month: 2, dayOfMonth: 10), hoursDone: 30, idEmployee: "0000000001"));
}

@Test
public void testCreateInput() {
    if (workHoursService.createNewWorkHoursInput( taskName: "Nabava", LocalDate.of( year: 2020, month: 2, dayOfMonth: 10), hoursDone: 10, idEmployee: "0000000001").getId().intValue() != 2)
        Assertions.fail();
}

```

Slika 5.15: WorkHoursServiceTest

- ✓ WorkHoursTest
- ✓ testNonExistingEmployeeWhenListingTasks()
 - ✓ testCreateInput()
 - ✓ testNonExistingEmployeeWhenCreatingInput()
 - ✓ testListTasksForEmployee()
 - ✓ testInvalidNumberOfHours()
 - ✓ testNullEmployee()

Slika 5.16: WorkHoursServiceTest rezultati

```
package progi.dugonogiprogi.radnovrijeme.backend;

import ...

@ExtendWith(SpringExtension.class)
@RunWith(SpringRunner.class)
@SpringBootTest(classes = BackendApplication.class)
class TasksTest {

    @Autowired
    private TaskService taskService;

    @Autowired
    private GroupRepository groupRepository;

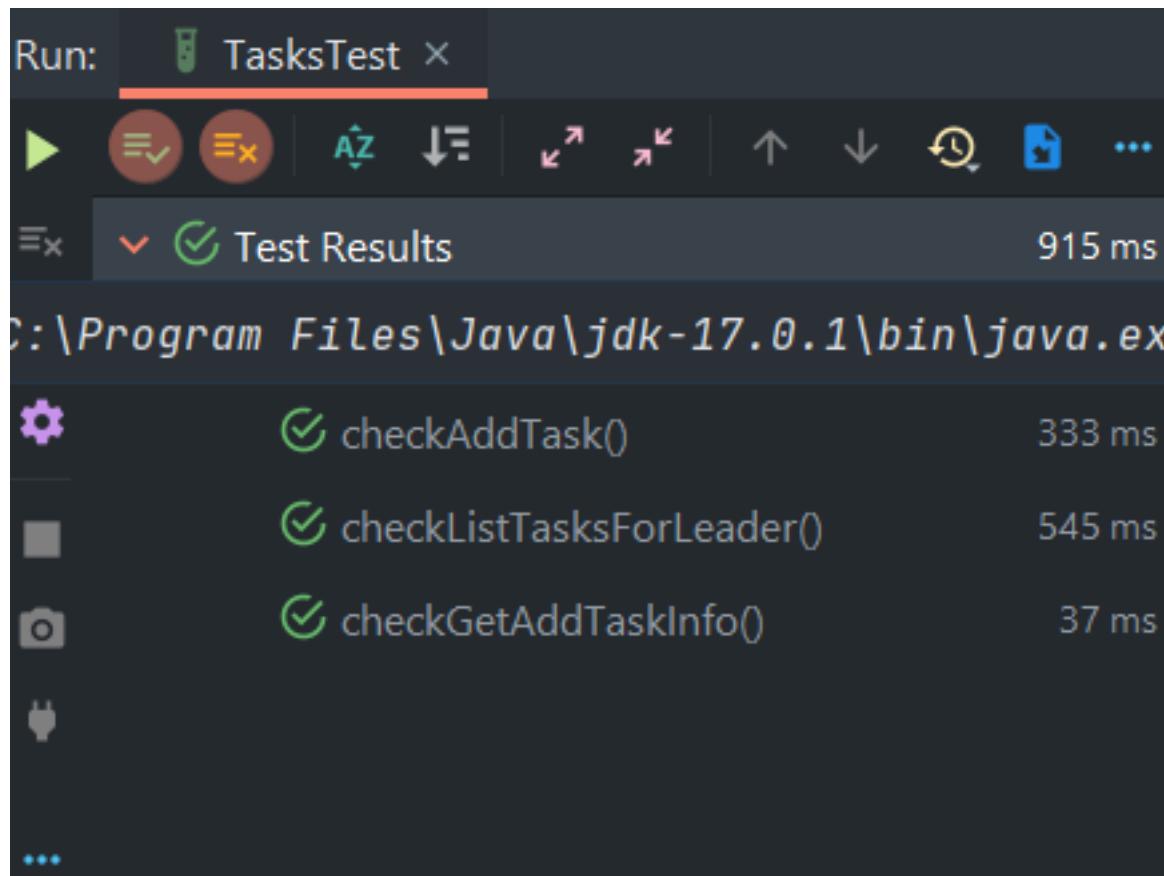
    @Autowired
    private JobRepository jobRepository;

    @Test
    public void checkListTasksForLeader() {
        List<TasksDTO> list = taskService.listTasksForLeader( idLeader: "00000000001");
        for(TasksDTO tDTO : list){
            Assertions.assertEquals(groupRepository.findByName("Group 1").get().getId(), tDTO.getTaskGroup().getId());
        }
        Assertions.assertThrows(NoSuchGroupException.class, () -> taskService.listTasksForLeader( idLeader: "00000000014"));
    }

    @Test
    public void checkAddTask(){
    }

    @Test
    public void checkGetAddTaskInfo(){
        Assertions.assertEquals( expected: 21,taskService.getAddTaskInfo().getEmployees().size());
        Assertions.assertEquals( expected: 1,taskService.getAddTaskInfo().getExistingLocations().size());
        Assertions.assertEquals( expected: 6,taskService.getAddTaskInfo().getJobs().size());
    }
}
```

Slika 5.17: TasksServiceTest



Slika 5.18: TasksServiceTest rezultati

```
package progi.dugonogiprogi.radnovrijeme.backend;

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import org.springframework.test.context.junit4.SpringRunner;
import progi.dugonogiprogi.radnovrijeme.backend.rest.dto.MoneyManagementDTO;
import progi.dugonogiprogi.radnovrijeme.backend.service.abstractService.MoneyManagementService;
import progi.dugonogiprogi.radnovrijeme.backend.service.impl.MoneyManagementServiceJpa;

@ExtendWith(SpringExtension.class)
@RunWith(SpringRunner.class)
@SpringBootTest(classes = BackendApplication.class)

class MoneyManagementTest {

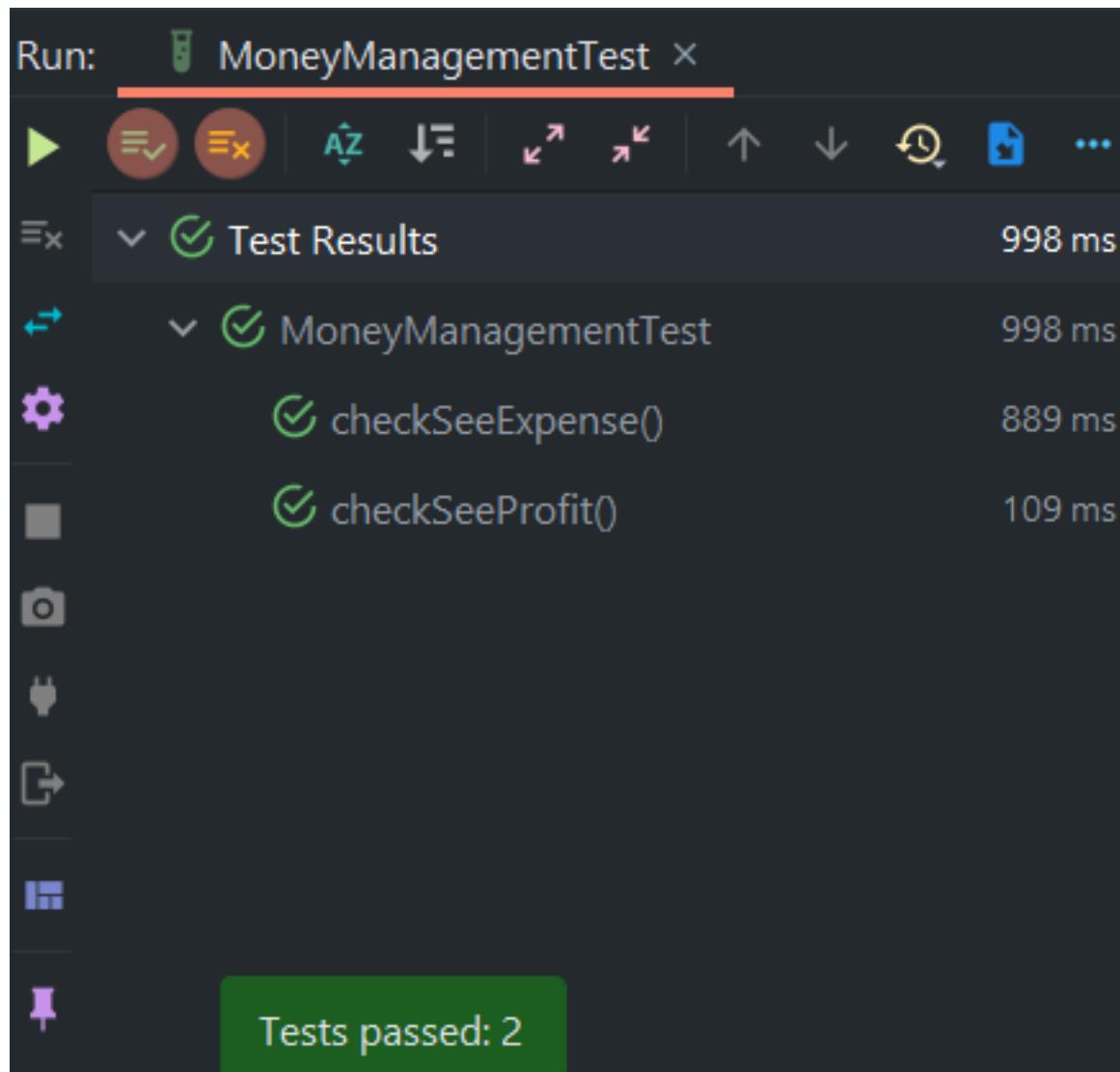
    @Autowired
    private MoneyManagementService moneyManagementService;

    @Autowired
    private MoneyManagementServiceJpa moneyManagementServiceJpa;

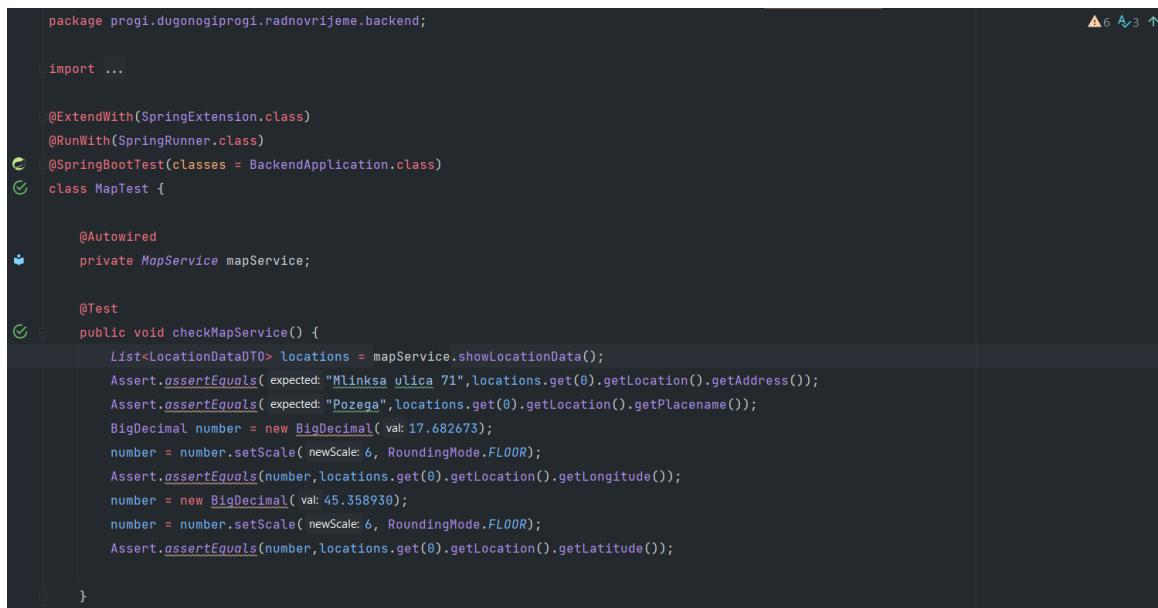
    @Test
    public void checkSeeProfit() {
        MoneyManagementDTO moneyManagementDTO = moneyManagementService.seeProfit( price: 85645745 );
        Assertions.assertEquals( expected: 85645745 - moneyManagementDTO.getPrice(), moneyManagementDTO.getDifference());
    }

    @Test
    public void checkSeeExpense() {
        MoneyManagementDTO moneyManagementDTO = moneyManagementService.seeExpense( price: 85645745 );
        Assertions.assertEquals( expected: 85645745 - moneyManagementDTO.getPrice(), moneyManagementDTO.getDifference());
        Assertions.assertEquals(moneyManagementServiceJpa.countExpense(), moneyManagementDTO.getPrice());
    }
}
```

Slika 5.19: MoneyManagementServiceTest



Slika 5.20: MoneyManagementServiceTest rezultati



```
package progi.dugonogiprogi.radnovrijeme.backend;

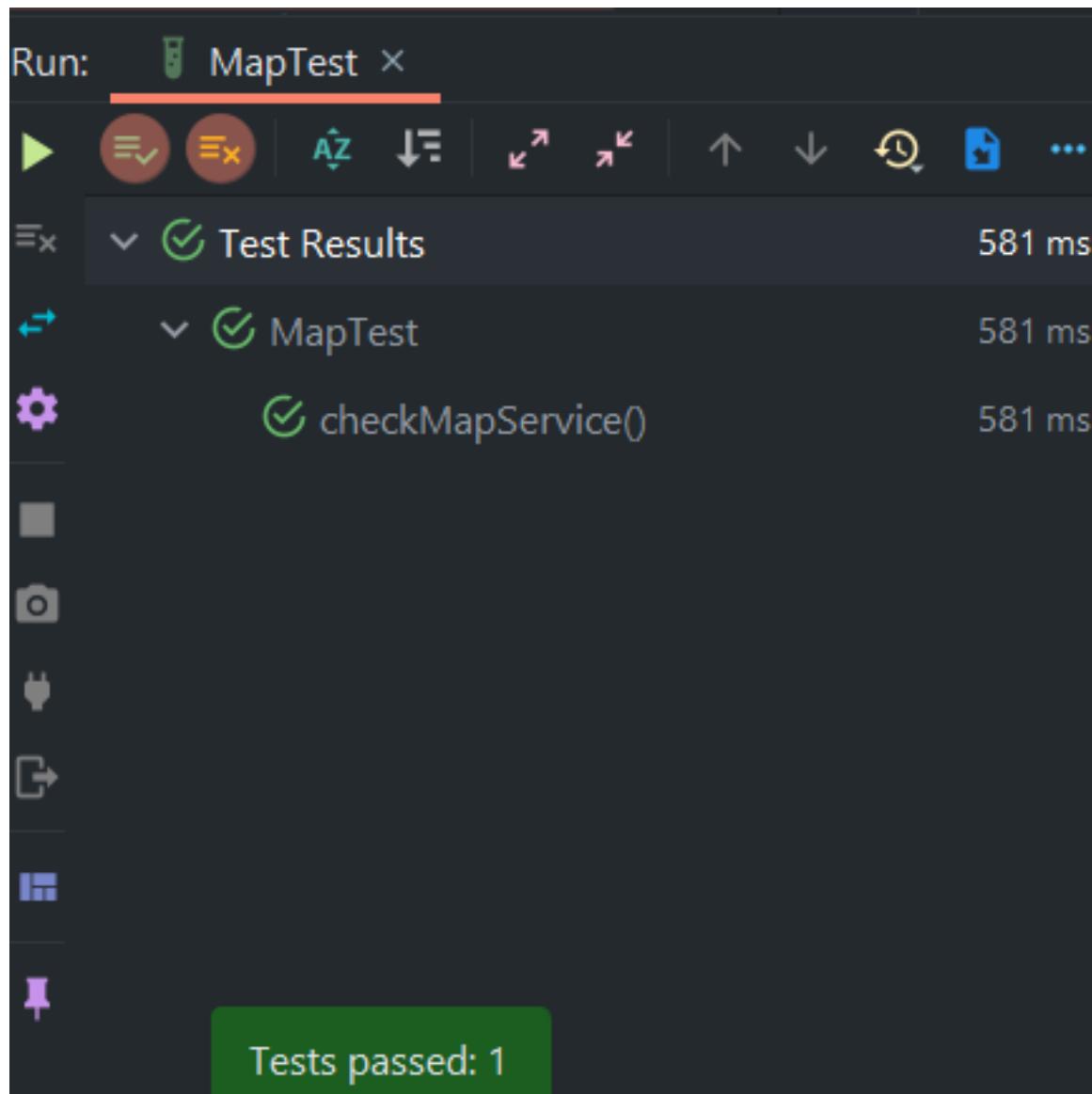
import ...

@ExtendWith(SpringExtension.class)
@RunWith(SpringRunner.class)
@SpringBootTest(classes = BackendApplication.class)
class MapTest {

    @Autowired
    private MapService mapService;

    @Test
    public void checkMapService() {
        List<LocationDataDTO> locations = mapService.showLocationData();
        Assert.assertEquals("Mlinčka ulica 71", locations.get(0).getLocation().getAddress());
        Assert.assertEquals("Pozega", locations.get(0).getLocation().getPlacename());
        BigDecimal number = new BigDecimal( val: 17.682673);
        number = number.setScale( newScale: 6, RoundingMode.FLOOR);
        Assert.assertEquals(number, locations.get(0).getLocation().getLongitude());
        number = new BigDecimal( val: 45.358930);
        number = number.setScale( newScale: 6, RoundingMode.FLOOR);
        Assert.assertEquals(number, locations.get(0).getLocation().getLatitude());
    }
}
```

Slika 5.21: MapServiceTest

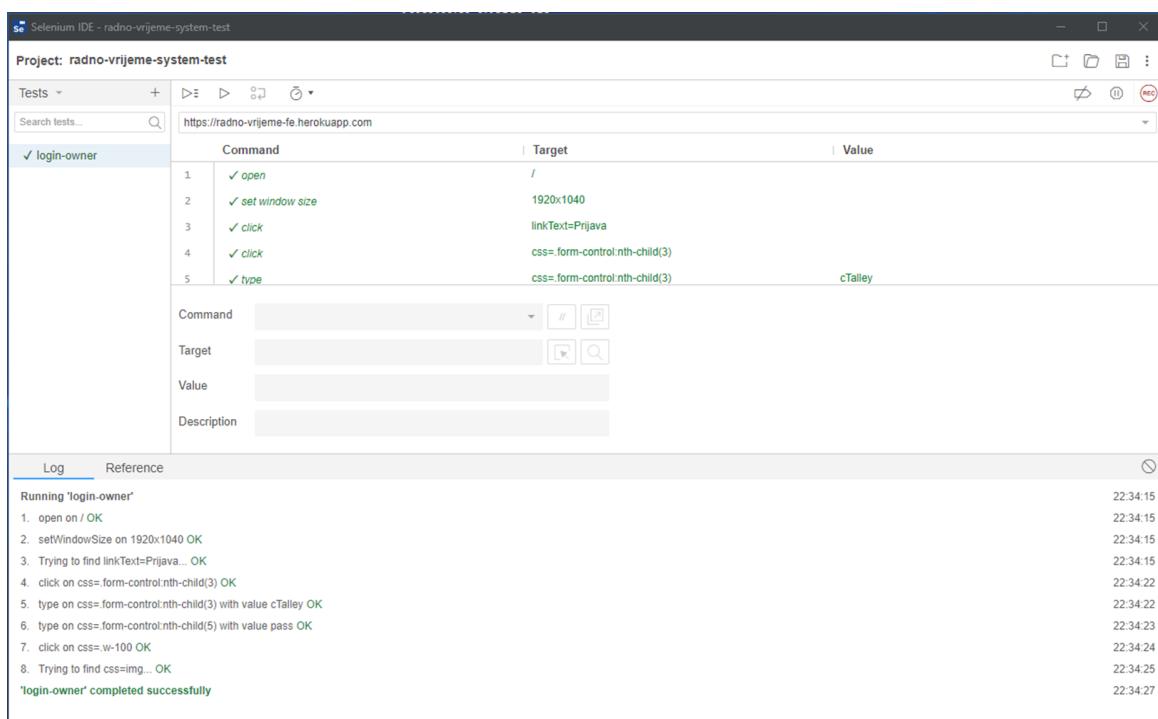


Slika 5.22: MapServiceTest rezultati

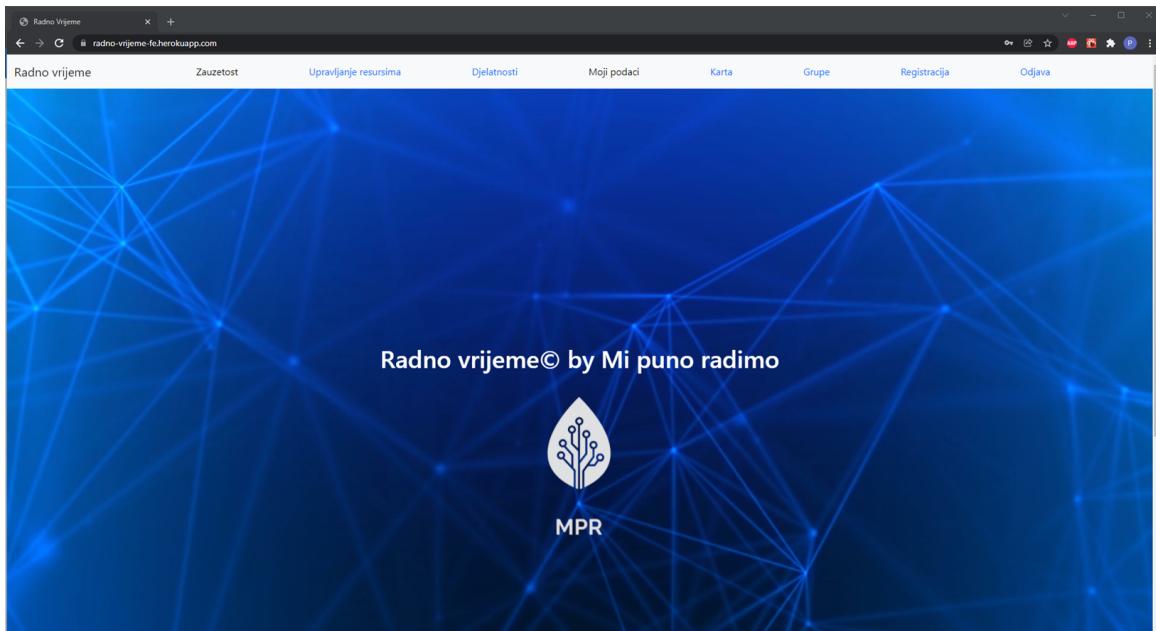
5.2.2 Ispitivanje sustava

Ispitni slučaj 1: Prijava vlasnika

- **Ulaz:** korisničko ime: cTalley, lozinka: pass
- **Očekivani izlaz:** uspješna prijava i preusmjeravanje na početnu stranicu, navigacijska traka prikazuje stranice dostupne vlasniku
- **Dobiveni izlaz:** uspješna prijava i preusmjeravanje na početnu stranicu, navigacijska traka prikazuje stranice dostupne vlasniku
- **Rezultat ispitivanja:** uspješno



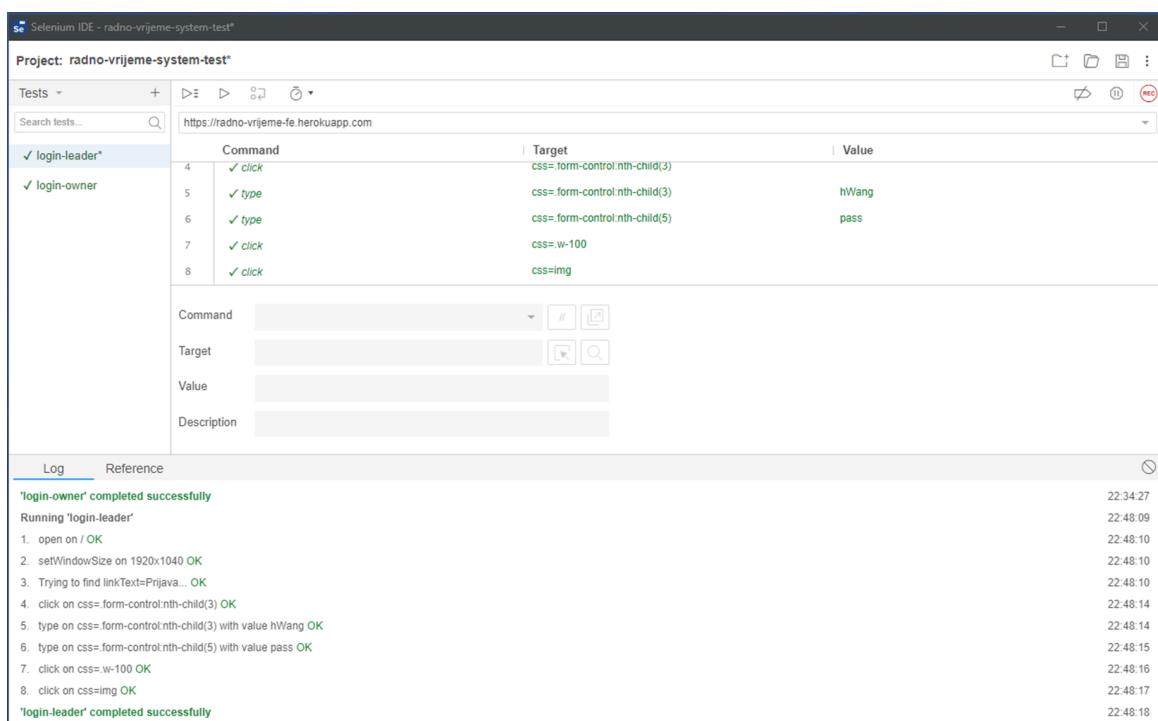
Slika 5.23: Selenium test 1



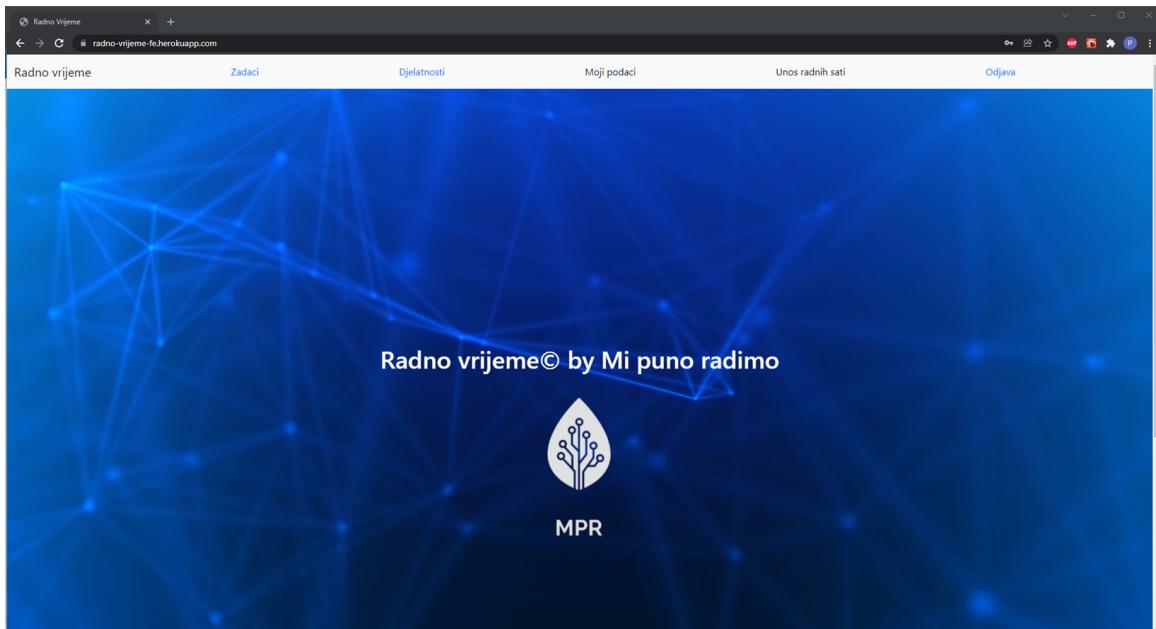
Slika 5.24: Rezultat 1

Ispitni slučaj 2: Prijava vođe tima

- **Ulaz:** korisničko ime: hWang, lozinka: pass
- **Očekivani izlaz:** uspješna prijava i preusmjeravanje na početnu stranicu, navigacijska traka prikazuje stranice dostupne vođi tima
- **Dobiveni izlaz:** uspješna prijava i preusmjeravanje na početnu stranicu, navigacijska traka prikazuje stranice dostupne vođi tima
- **Rezultat ispitivanja:** uspješno



Slika 5.25: Selenium test 2



Slika 5.26: Rezultat 2

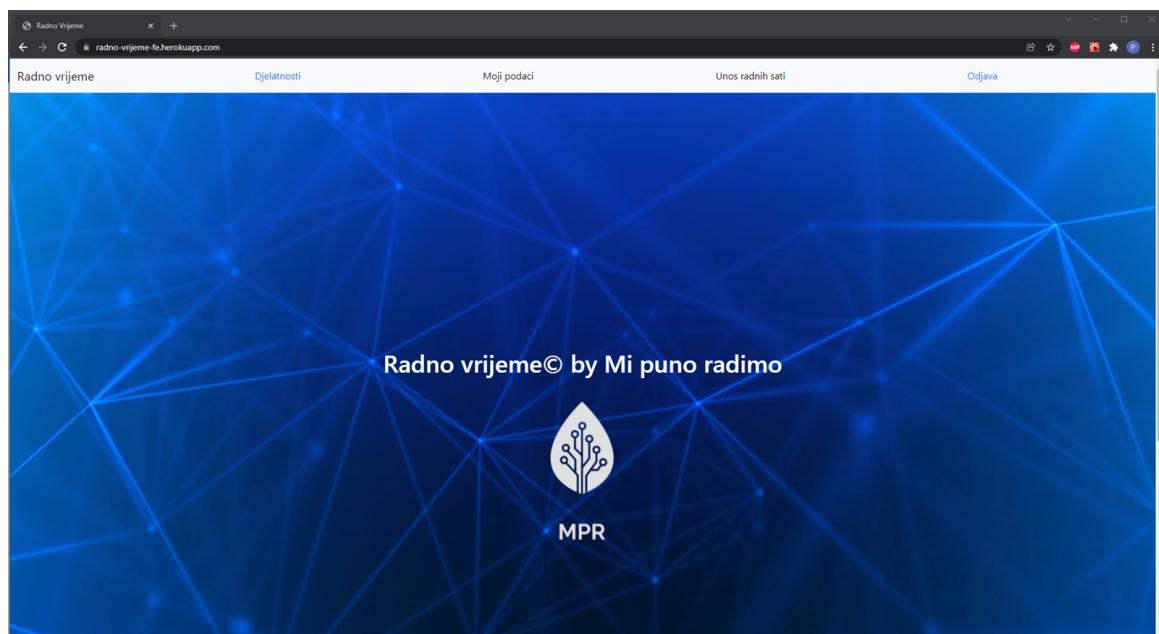
Ispitni slučaj 3: Prijava zaposlenika

- **Ulaz:** korisničko ime: *aBevan*, lozinka: *pass*
- **Očekivani izlaz:** uspješna prijava i preusmjeravanje na početnu stranicu, navigacijska traka prikazuje stranice dostupne zaposleniku
- **Dobiveni izlaz:** uspješna prijava i preusmjeravanje na početnu stranicu, navigacijska traka prikazuje stranice dostupne zaposleniku
- **Rezultat ispitivanja:** uspješno

The screenshot shows the Selenium IDE interface with the following details:

- Project:** radno-vrijeme-system-test*
- Test:** login-employee*
- URL:** https://radno-vrijeme-fe.herokuapp.com
- Test Log:**
 - 'login-owner' completed successfully
 - Running 'login-employee'
 - 1. open on / OK
 - 2. setWindowSize on 1920x1040 OK
 - 3. click on linkText=Prjava OK
 - 4. click on css=form-control:nth-child(3) OK
 - 5. type on css=form-control:nth-child(3) with value aBevan OK
 - 6. type on css=form-control:nth-child(5) with value pass OK
 - 7. click on css=w-100 OK
 - 8. click on css=img OK
- Log:** Shows the execution log with timestamps from 22:50:21 to 22:55:31.

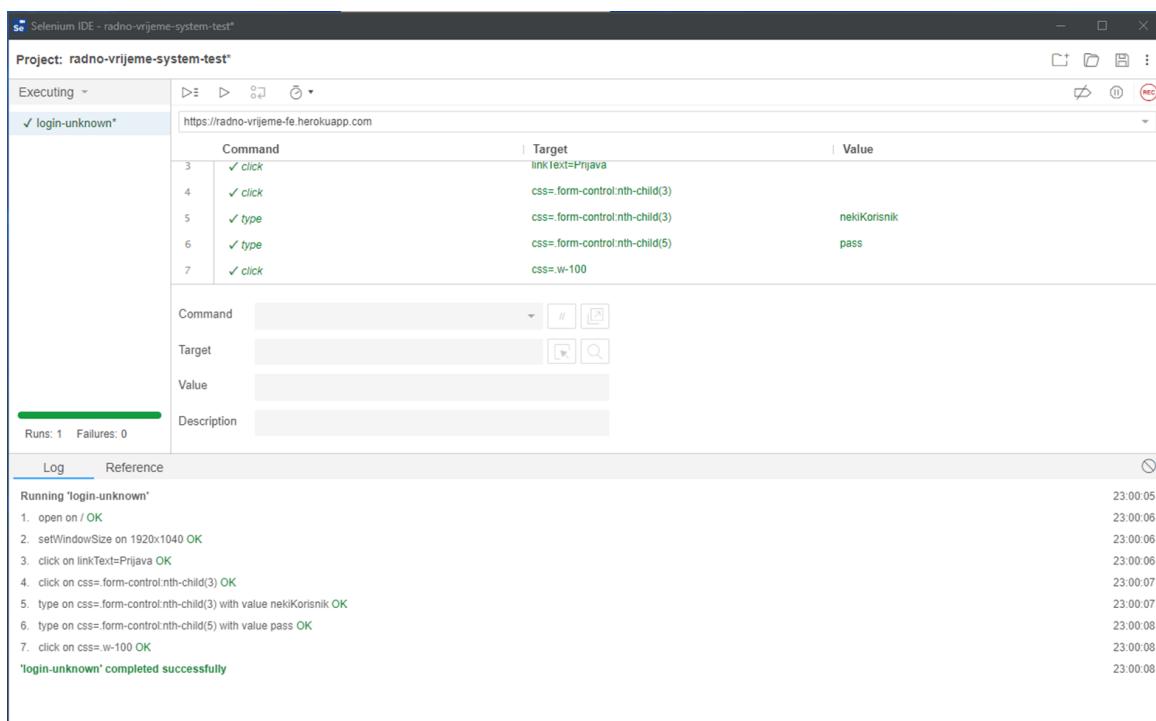
Slika 5.27: Selenium test 3



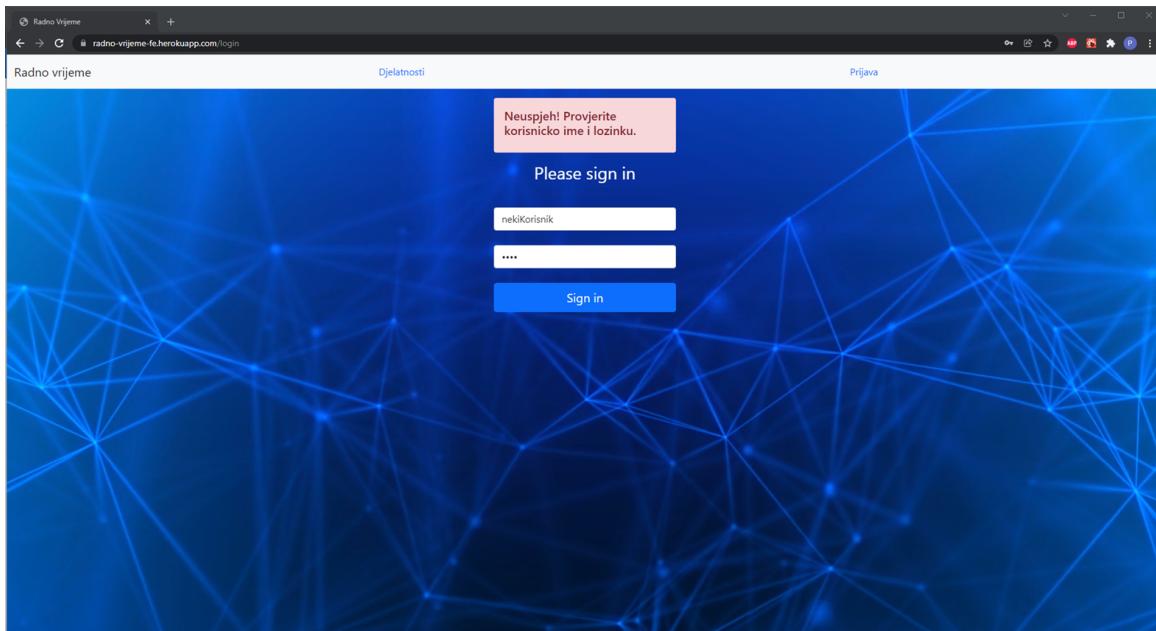
Slika 5.28: Rezultat 3

Ispitni slučaj 4: Prijava nepostojećeg korisnika

- **Ulaz:** korisničko ime: nekiKorisnik, lozinka: pass
- **Očekivani izlaz:** poruka o neuspješnoj prijavi
- **Dobiveni izlaz:** poruka o neuspješnoj prijavi
- **Rezultat ispitivanja:** uspješno



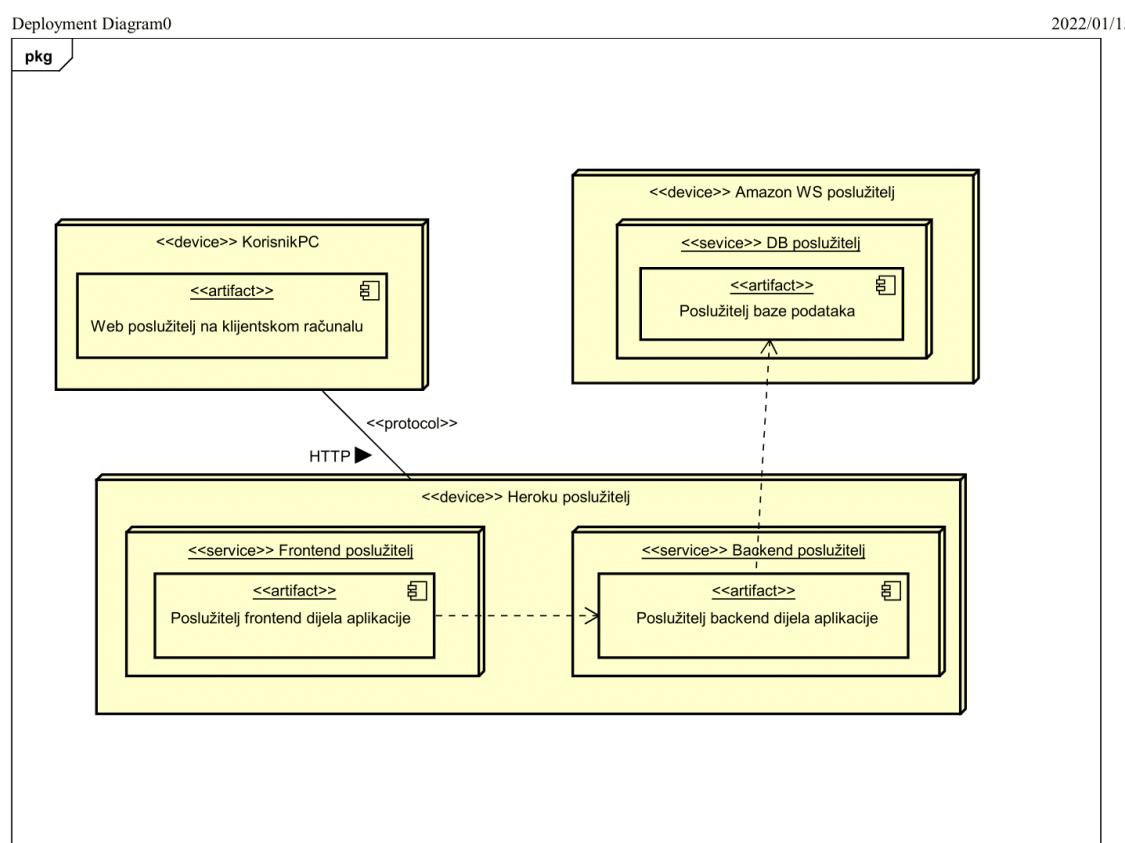
Slika 5.29: Selenium test 4



Slika 5.30: Rezultat 4

5.3 Dijagram razmještaja

Dijagram razreda prikazuje izvršnu arhitekturu sustava. Sustav je baziran na arhitekturi REST koja odvaja klijente od servera. Pomoću web preglednika, klijenti pristupaju poslužitelju frontend dijela naše aplikacije. Ta komunikacija odvija se putem HTTP veze. Poslužitelj frontend dijela zatim dohvata podatke u JSON formatu sa back-end poslužitelja također putem HTTP veze. Frontend i backend aplikacije, iako odvojene, obje se nalaze na Heroku poslužitelju. Backend poslužitelj u izravnoj je komunikaciji s poslužiteljem baze podataka koja se nalazi na Amazon WS poslužitelju.



Slika 5.31: Dijagram razmještaja

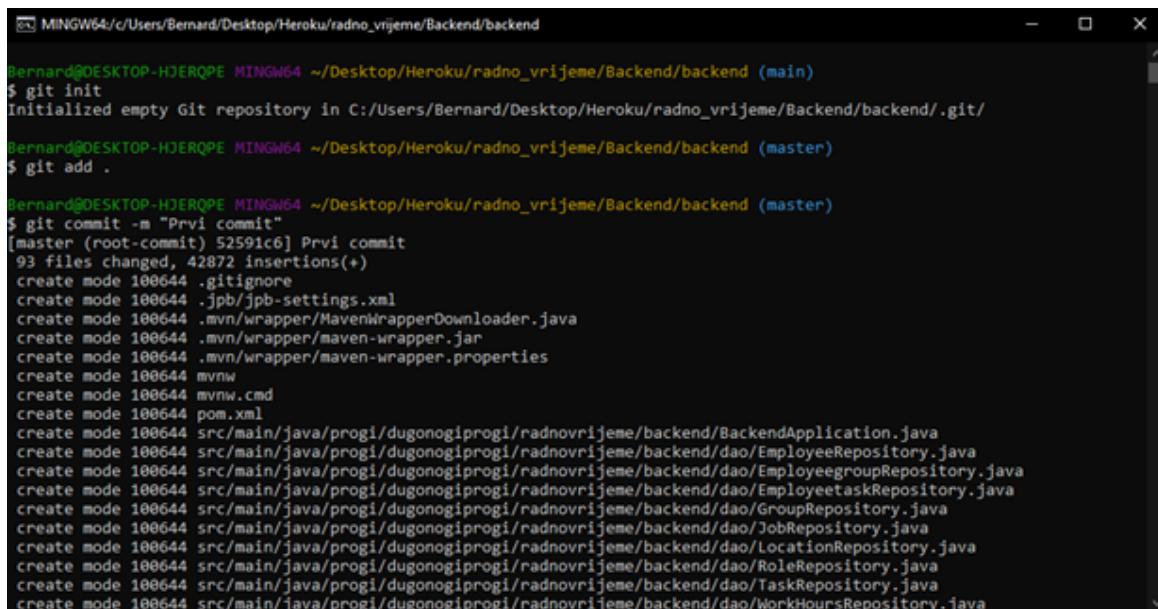
5.4 Upute za puštanje u pogon

Priprema alata za puštanje aplikacija u pogon na Heroku

Potrebno je instalirati Git i Heroku CLI pomoću windows installera koji se može jednostavno naći na internetu na njihovim službenim stranicama. Također treba napraviti korisnički račun na Heroku. Nakon što je to postavljeno potrebno je skinuti Repozitorij *radno vrijeme*.

Puštanje backend aplikacije u pogon

Prebacuje se u mapu .../*radno vrijeme*/Backend/backend i tamo se treba otvoriti Git Bash. Inicijaliziramo novi git repozitorij pomoću naredbe “git init”. Dodajemo i commitamo sve datoteke u tom folderu naredbama “git add .” i “git commit -m “Prvi commit””.



```

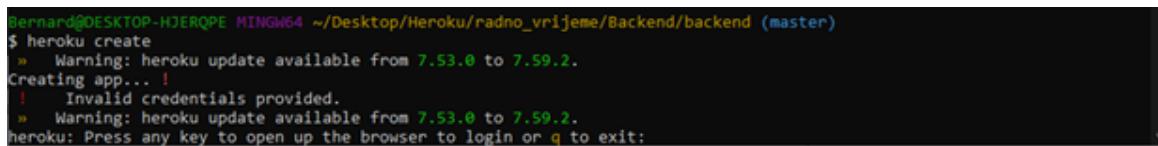
MINGW64:/c/Users/Bernard/Desktop/Heroku/radno_vrijeme/Backend/backend
$ git init
Initialized empty Git repository in C:/Users/Bernard/Desktop/Heroku/radno_vrijeme/Backend/backend/.git/
Bernard@DESKTOP-HJERQPE MINGW64 ~/Desktop/Heroku/radno_vrijeme/Backend/backend (master)
$ git add .

Bernard@DESKTOP-HJERQPE MINGW64 ~/Desktop/Heroku/radno_vrijeme/Backend/backend (master)
$ git commit -m "Prvi commit"
[master (root-commit) 52591c6] Prvi commit
 93 files changed, 42872 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .jpb/jpb-settings.xml
 create mode 100644 .mvn/wrapper/MavenWrapperDownloader.java
 create mode 100644 .mvn/wrapper/maven-wrapper.jar
 create mode 100644 .mvn/wrapper/maven-wrapper.properties
 create mode 100644 mvnw
 create mode 100644 mvnw.cmd
 create mode 100644 pom.xml
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/BackendApplication.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/EmployeeRepository.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/EmployeegroupRepository.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/EmployeeetaskRepository.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/GroupRepository.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/JobRepository.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/LocationRepository.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/RoleRepository.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/TaskRepository.java
 create mode 100644 src/main/java/progi/dugonogiprogi/radnovrijeme/backend/dao/WorkHoursRepository.java

```

Slika 5.32: Prvi commit za backend

Nakon toga upisujemo komandu “heroku create”. Heroku će reći da nisu validni podaci za login te će tražiti da se ulogiramo pomoću web preglednika.



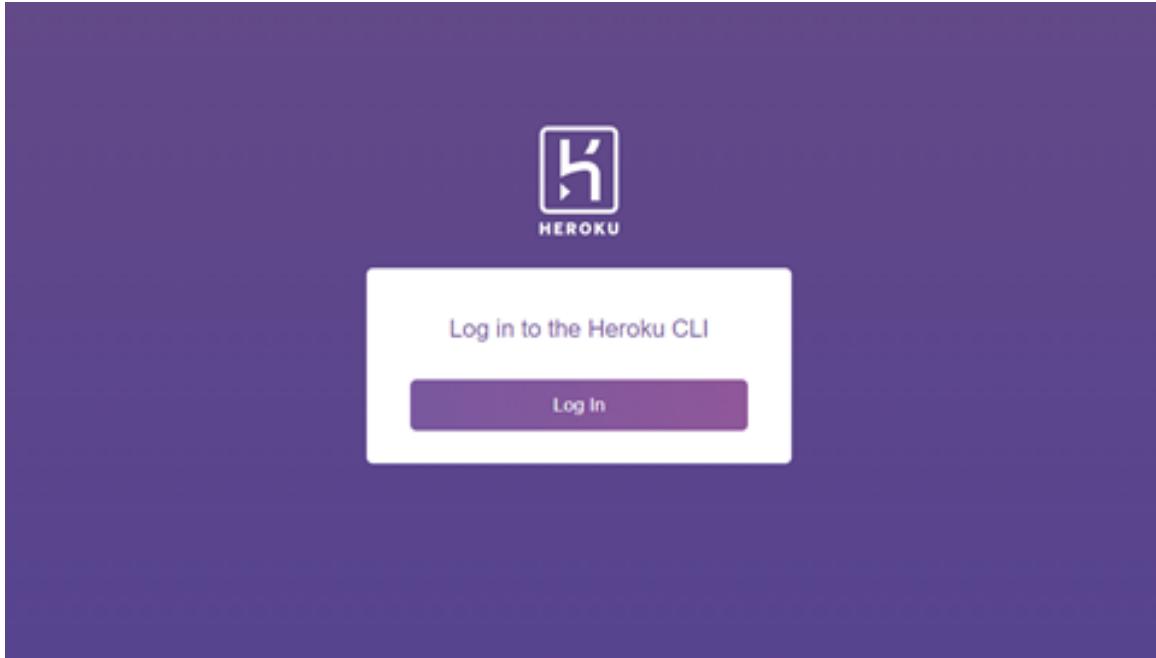
```

Bernard@DESKTOP-HJERQPE MINGW64 ~/Desktop/Heroku/radno_vrijeme/Backend/backend (master)
$ heroku create
» Warning: heroku update available from 7.53.0 to 7.59.2.
Creating app... !
! Invalid credentials provided.
» Warning: heroku update available from 7.53.0 to 7.59.2.
heroku: Press any key to open up the browser to login or q to exit:

```

Slika 5.33: Pokretanje Heroku u Git-u

Nakon što se upiše bilo koji znak u web pregledniku će nam se pojaviti ova stranica :



Slika 5.34: Heroku web-stranica

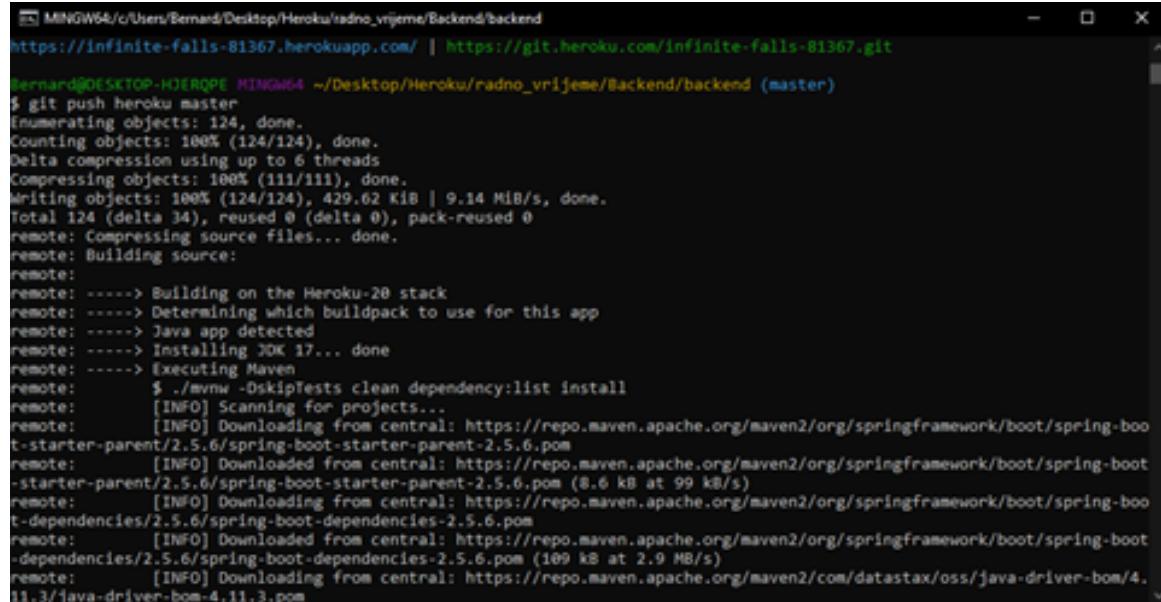
Prije puštanja u pogon moramo namjestiti varijable za bazu podataka u application.properties. Datoteka se nalazi u .../backend/src/main/resources/ i treba izgledati kao sljedeće:

```
java.runtime.version=17
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=${JDBC_DATABASE_URL}
spring.datasource.username=${JDBC_DATABASE_USERNAME}
spring.datasource.password=${JDBC_DATABASE_PASSWORD}
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Slika 5.35: Prikaz datoteke za varijable baze podataka

Mora se specificirati koju verziju jave koristimo u aplikaciji da Heroku zna kako ju pokrenuti. Url, korisničko ime i lozinku navodimo tako jer ih Heroku upisuje pomoću svojih varijabli okruženja. Koristi se također PostgreSQLDialect pošto heroku baze podataka stvara u Postgresql.

Nakon podešavanja application.properties u bash upisujemo “git push heroku master”



```
MINGW64 /c/Users/Bernard/Desktop/Heroku/radno_vrijeme/Backend/backend
https://infinite-falls-81367.herokuapp.com/ | https://git.heroku.com/infinite-falls-81367.git
Bernard@DESKTOP-HOERQPE MINGW64 ~ /Desktop/Heroku/radno_vrijeme/Backend/backend (master)
$ git push heroku master
Enumerating objects: 124, done.
Counting objects: 100% (124/124), done.
Delta compression using up to 6 threads
Compressing objects: 100% (111/111), done.
Writing objects: 100% (124/124), 429.62 KiB | 9.14 MiB/s, done.
Total 124 (delta 34), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: ----> Building on the Heroku-20 stack
remote: ----> Determining which buildpack to use for this app
remote: ----> Java app detected
remote: ----> Installing JDK 17... done
remote: ----> Executing Maven
remote:     $ ./mvnw -DskipTests clean dependency:list install
remote:     [INFO] Scanning for projects...
remote:     [INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/2.5.6/spring-boot-starter-parent-2.5.6.pom
remote:     [INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/2.5.6/spring-boot-starter-parent-2.5.6.pom (8.6 kB at 99 kB/s)
remote:     [INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-dependencies/2.5.6/spring-boot-dependencies-2.5.6.pom
remote:     [INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-dependencies/2.5.6/spring-boot-dependencies-2.5.6.pom (109 kB at 2.9 MB/s)
remote:     [INFO] Downloading from central: https://repo.maven.apache.org/maven2/com/databricks/oss/java-driver-bom/4.11.3/java-driver-bom-4.11.3.pom
```

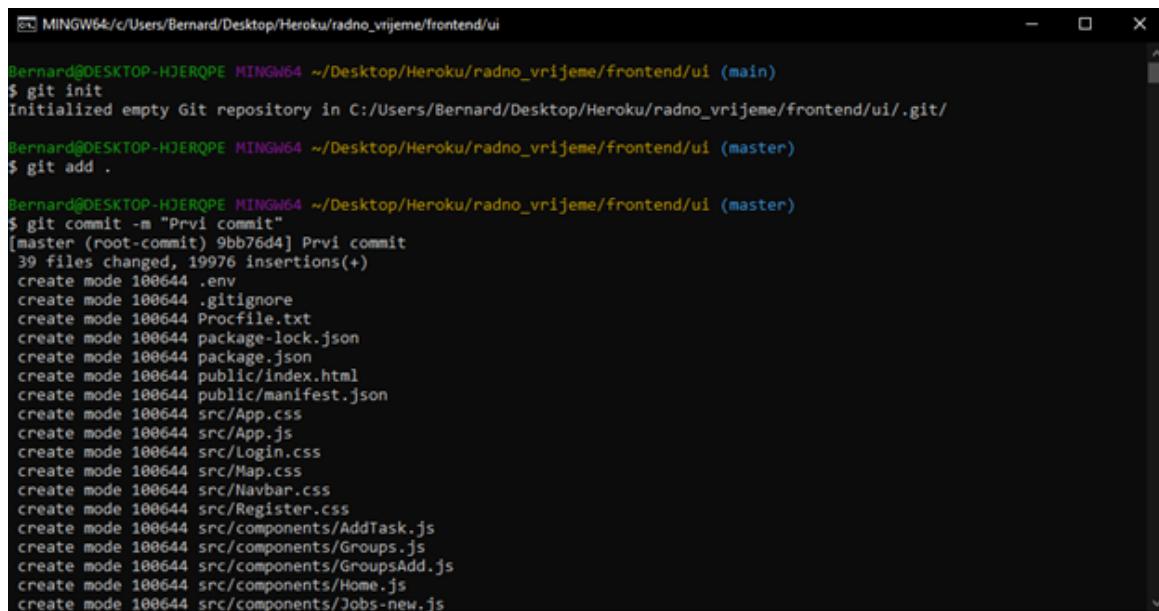
Slika 5.36: “git push heroku master” rezultati

Time smo završili sa puštanjem backend aplikacije u pogon.

Puštanje frontend aplikacije u pogon

Postupak za frontend aplikaciju je skoro identičan kao i za backend aplikaciju.

Prebacuje se u mapu .../radno_vrijeme/frontend/ui i tamo se treba otvoriti Git Bash. Inicijaliziramo novi git rezpositorij pomoću naredbe “git init”. Dodajemo i commitamo sve datoteke u tom folderu naredbama “git add .” i “git commit -m “Prvi commit””.



```
MINGW64: c:/Users/Bernard/Desktop/Heroku/radno_vrijeme/frontend/ui
```

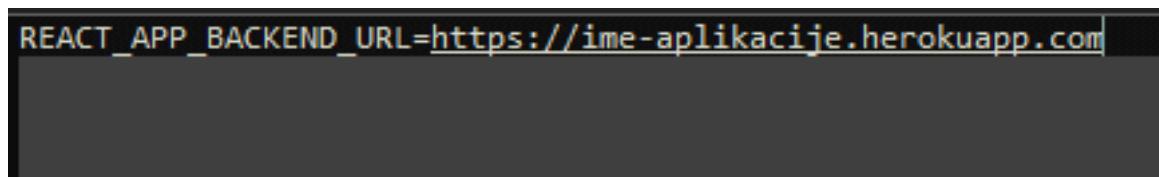
```
Bernard@DESKTOP-HJERQPE MINGW64 ~/Desktop/Heroku/radno_vrijeme/frontend/ui (main)
$ git init
Initialized empty Git repository in C:/Users/Bernard/Desktop/Heroku/radno_vrijeme/frontend/ui/.git/
Bernard@DESKTOP-HJERQPE MINGW64 ~/Desktop/Heroku/radno_vrijeme/frontend/ui (master)
$ git add .

Bernard@DESKTOP-HJERQPE MINGW64 ~/Desktop/Heroku/radno_vrijeme/frontend/ui (master)
$ git commit -m "Prvi commit"
[master (root-commit) 9bb76d4] Prvi commit
 39 files changed, 19976 insertions(+)
create mode 100644 .env
create mode 100644 .gitignore
create mode 100644 Procfile.txt
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 public/index.html
create mode 100644 public/manifest.json
create mode 100644 src/App.css
create mode 100644 src/App.js
create mode 100644 src/Login.css
create mode 100644 src/Map.css
create mode 100644 src/Navbar.css
create mode 100644 src/Register.css
create mode 100644 src/components/AddTask.js
create mode 100644 src/components/Groups.js
create mode 100644 src/components/GroupsAdd.js
create mode 100644 src/components/Home.js
create mode 100644 src/components/Jobs-new.js
```

Slika 5.37: Prvi commit za frontend

Nakon toga upisujemo komandu “heroku create”. Pošto smo već ulogirani odma će stvoriti novu aplikaciju.

Prije puštanja aplikacije u pogon moramo u .env dokumentu upisati lokaciju backend-a.



Slika 5.38: Adreasa backend aplikacije

Ovdje se ime-aplikacije odnosi na to kako nam se na Heroku zovu aplikacije.

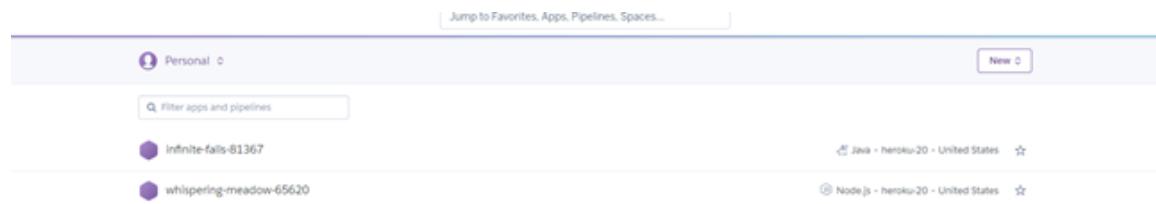
Zadnji korak je staviti izvorni kod u tu aplikaciju što se radi pomoću komande heroku “git push heroku master”.

```
Bernard@DESKTOP-HJERQPE MINGW64 ~/Desktop/Heroku/radno_vrijeme/frontend/ui (master)
$ git push heroku master
Enumerating objects: 45, done.
Counting objects: 100% (45/45), done.
Delta compression using up to 6 threads
Compressing objects: 100% (44/44), done.
Writing objects: 100% (45/45), 7.46 MiB | 4.20 MiB/s, done.
Total 45 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: ----> Building on the Heroku-20 stack
remote: ----> Determining which buildpack to use for this app
remote: ----> Node.js app detected
remote:
remote: ----> Creating runtime environment
remote:
remote:   NPM_CONFIG_LOGLEVEL=error
remote:   NODE_VERBOSE=false
remote:   NODE_ENV=production
remote:   NODE_MODULES_CACHE=true
```

Slika 5.39: "git push heroku master" za frontend

Ovime smo završili sa puštanjem frontend aplikacije u pogon.

Aplikacije možemo vidjeti na svome profilu na Heroku i otvarati ih preko linkova "<https://ime-aplikacije.herokuapp.com>"



Slika 5.40: Prikaz aplikacije na Heroku

Punjjenje baze podataka

Baza podataka se automatski stvorila sa backend aplikacijom zbog definicija u `application.properties`.

The screenshot shows the Heroku dashboard for the application 'Infinite-fails-81367'. At the top, there's a navigation bar with 'Personal' and 'Infinite-fails-81367'. Below it, tabs for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings are visible. The Overview tab is selected.

- Installed add-ons:** \$0.00/month (Heroku Postgres Hobby Dev postgresql-contoured-02285)
- Dyno formation:** \$0.00/month (This app is using free dynos)
- Activity:** Shows recent events: Deployed (529151dd), provisioning, setting DATABASE_URL, Attach DATABASE (@ref postgresql-contoured-02285), Build succeeded, Enable Logplex, and Initial release.
- Collaborator activity:** Shows 1 deploy.

Slika 5.41: Prikaz baze podataka uz backend aplikaciju

Do detalja baze dolazimo klikom na Heroku Postgres add-on i oni će nam trebati za postavljanje pristupa bazi.

The screenshot shows the Heroku Datastores page for the PostgreSQL add-on 'postgresql-contoured-02285'. It includes tabs for Overview, Durability, Settings, and Dataclips. The Settings tab is selected.

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

Please note that **these credentials are not permanent**. Heroku rotates credentials periodically and updates applications where this database is attached.

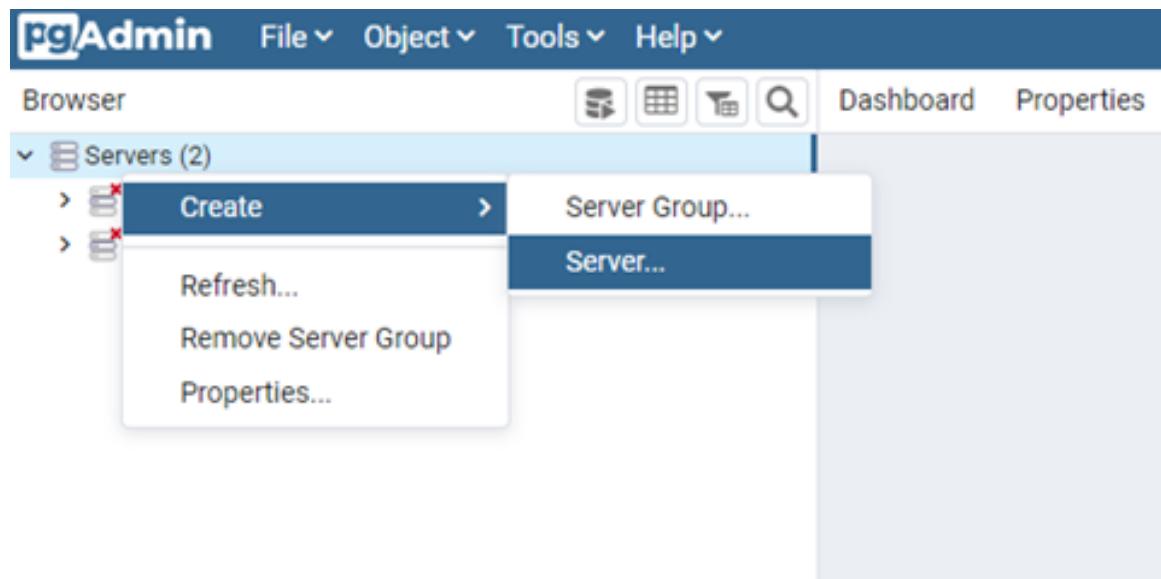
Host	ec2-3-232-22-121.compute-1.amazonaws.com
Database	d9df6hd20c70gu
User	plafnkdppclxa
Port	5432
Password	41fd0013a7990c960f97e3c06e4e3646cf3b5701d8d2cf61392204fb0dbe4f
URI	postgres://plafnkdppclxa 41fd0013a7990c960f97e3c06e4e3646cf3b5701d8d2cf61392204fb0dbe4f@ec2-3-232-22-121.compute-1.amazonaws.com:5432/d9df6hd20c70gu
Heroku CLI	heroku pg:psql postgresql-contoured-02285 --app infinite-fails-81367

Slika 5.42: Detalji baze podataka

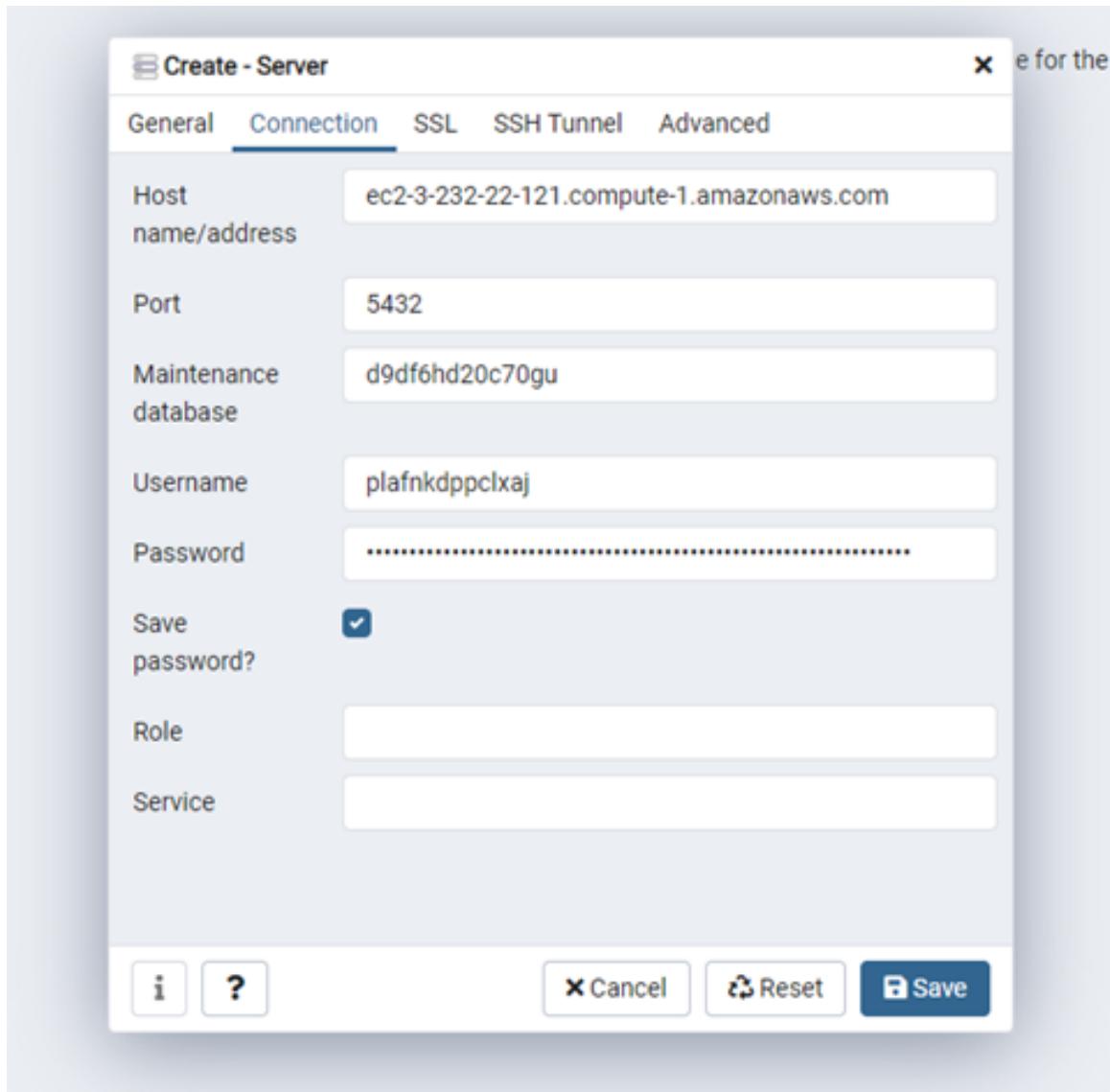
Treba se instalirati pgAdmin na računalo pomoću windows installera koji možemo naći na službenoj stranici.

Nakon što otvorimo aplikaciju prvi put potrebno je stvoriti korisničko ime i lozinku po volji. Sada se treba stvoriti nova veza na server gdje se nalazi baza.

Server se naziva po volji i u polja upisujemo vrijednosti koje se nalaze u detaljima naše baze na Heroku.

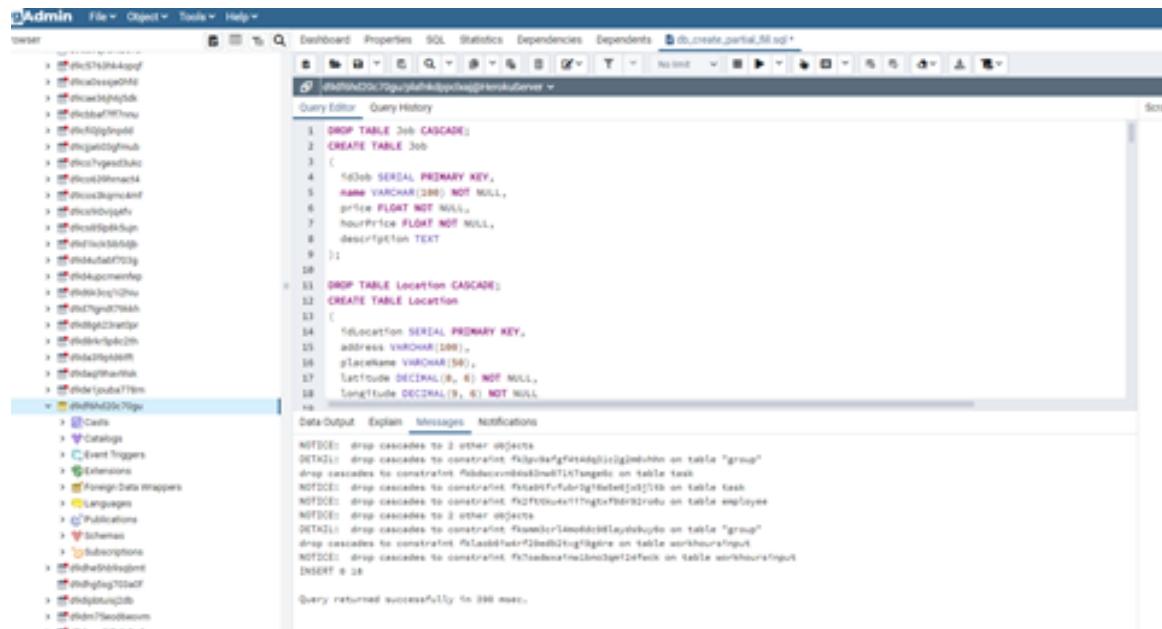


Slika 5.43: pgAdmin stvaranje poslužitelja



Slika 5.44: pgAdmin4 stvaranje poslužitelja 2. dio

Sada među bazama podataka na HerokuServer nađemo našu i otvorimo Query Tool. U njemu otvorimo dokument *partial\database\fill.sql* koji se nalazi u .../radno\vrijeme\Database i pokrenemo ga. Time smo napunili našu bazu podataka.



Slika 5.45: pgAdmin4 "query tool"

6. Zaključak i budući rad

Zadatak našega tima bio je razvoj aplikacije koja bi omogućila poduzeću „Mi puno radimo“ praćenje realizacije zadataka te raspoloživosti djelatnika kako bi se olakšala organizacija djelatnosti i zaposlenika unutar firme. Kroz 15 tjedana razvoja uspješno smo ostvarili naš cilj.

Prvi zadatak, okupljanje tima nije bio težak. Prethodna poznanstva uvelike su olakšala komunikaciju i pozitivno pridonijele funkciranju grupe. Zatim je uslijedila prva faza provedbe projekta. Zajednički smo pisali dokumentaciju što je olakšalo daljnji rad. Dokumentirani zahtjevi, izrađeni obrasci i dijagrami detaljno su prikazivali željene funkcionalnosti aplikacije. Svaki član tima, na temelju priloženog, znao je točno što treba implementirati te što je već bilo odrađeno. Konkretnu implementaciju zadataka možemo svrstati u drugu fazu razvoja.

Početak je bio težak, no zajedničkim snagama uspjeli smo se uhodati u razvoj aplikacije. Grupa se podijelila na četiri *backend* programera, dva *frontend* te jednog zaduženog za bazu podataka. Dalnjim razvojem i obavljanjem nekih od poslova, potreba za reorganiziranjem tima bila je nužna, stoga spomenute pozicije nisu bile fiksne cijelo vrijeme razvoja. Završna situacija zahtjevala je više *frontend* programera te nekoliko osoba za sređivanje dokumentacije. Ovakva reorganizacija tima pridonijela je da svaki člana nauči više. Ipak, reorganizacijom izgubili smo dosta vremena učeći nove alate i određene jezike, ali upravo se zato možemo još više ponositi postignutim ciljem.

Za sve članove ovakav projekt bio je potpuno novo iskustvo. Možemo istaknuti nova stečena znanja iz programiranja, no i usvojene vještine poput rada u timu, organizacije tima, ali i vremena. Prostor za poboljšanje uvijek postoji, no svakim sljedećim ovakvim iskušenjem uslijedit će i rast našeg iskustva, te će aplikacije biti bolje.

Popis literature

Kontinuirano osvježavanje

Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

1. Programsко 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>
7. Lucidchart : Intelligent Diagramming, <https://www.lucidchart.com/pages/>
8. Tutorial: Intro to React, <https://reactjs.org/tutorial/tutorial.html>

Indeks slika i dijagrama

4.1	ER dijagram baze podataka	28
4.2	Relacijski dijagram baze podataka	28
4.3	Dijagram razreda TaskController	30
4.4	Dijagram razreda WorkHoursController	31
4.5	Dijagram razreda GroupController	32
4.6	Dijagram razreda JobController	33
4.7	Dijagram razreda MoneyManagementController	34
4.8	Dijagram razreda MapController	35
4.9	Dijagram razreda MyDataController	36
4.10	Dijagram razreda OccupancyController	37
4.11	Dijagram razreda RegistrationController	38
4.12	Dijagram stanja - Voditelj	39
4.13	Brisanje djelatnosti	40
4.14	Brisanje grupe	41
4.15	Pregled lokacija	42
4.16	Pregled svih djelatnosti	43
4.17	Pregled svih grupa	44
4.18	Pregled vlastitih podataka	45
4.19	Pregled zadataka	46
4.20	Provjera zauzeća djelatnika	47
4.21	Registracija	48
4.22	Stvaranje djelatnosti	49
4.23	Stvaranje grupe	50
4.24	Stvaranje novog zadatka	51
4.25	Unos radnih sati	52
4.26	Dijagram komponenti	53
5.1	GroupServiceTest 1. dio	55
5.2	GroupServiceTest 2. dio	56
5.3	GroupServiceTest rezultati 1. dio	56

5.4 GroupServiceTest rezultati 2. dio	57
5.5 JobServiceTest	58
5.6 JobServiceTest rezultati 1. dio	59
5.7 JobServiceTest rezultati 2. dio	59
5.8 MyDataServiceTest	60
5.9 MyDataServiceTest rezultati	60
5.10 OccupancyServiceTest	60
5.11 OccupancyServiceTest rezultati	61
5.12 RegistrationServiceTest	62
5.13 RegistrationServiceTest rezultati 1. dio	63
5.14 RegistrationServiceTest rezultati 2. dio	63
5.15 WorkHoursServiceTest	64
5.16 WorkHoursServiceTest rezultati	64
5.17 TasksServiceTest	65
5.18 TasksServiceTest rezultati	66
5.19 MoneyManagementServiceTest	67
5.20 MoneyManagementServiceTest rezultati	68
5.21 MapServiceTest	69
5.22 MapServiceTest rezultati	70
5.23 Selenium test 1	71
5.24 Rezultat 1	72
5.25 Selenium test 2	73
5.26 Rezultat 2	74
5.27 Selenium test 3	75
5.28 Rezultat 3	75
5.29 Selenium test 4	76
5.30 Rezultat 4	77
5.31 Dijagram razmještaja	78
5.32 Prvi commit za backend	79
5.33 Pokretanje Heroku u Git-u	79
5.34 Heroku web-stranica	80
5.35 Prikaz datoteke za varijable baze podataka	80
5.36 "git push heroku master" rezultati	81
5.37 Prvi commit za frontend	82
5.38 Adreasa backend aplikacije	82

5.39 "git push heroku master" za frontend	83
5.40 Prikaz aplikacije na Heroku	83
5.41 Prikaz baze podataka uz backend aplikaciju	84
5.42 Detalji baze podataka	84
5.43 pgAdmin stvaranje poslužitelja	85
5.44 pgAdmin4 stvaranje poslužitelja 2. dio	86
5.45 pgAdmin4 "query tool"	87
6.1 GitLab dijagram promjena	99

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 13. listopada 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, L.Raspudić, P.Sušac, V.Žunar
- Teme sastanka:
 - 1. sastanak s asistentom i demonstratorom
 - raščišćavanje nejasnoća

2. sastanak

- Datum: 14. listopada 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, L.Raspudić, V.Žunar
- Teme sastanka:
 - ažuriranje LaTeX dokumentacije
 - dodjela zadatka članovima tima
 - platforme za komunikaciju

3. sastanak

- Datum: 18. listopada 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, L.Raspudić, P.Sušac, V.Žunar
- Teme sastanka:
 - raspodjela zadataka

4. sastanak

- Datum: 22. listopada 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - dodjela zadatka članovima tima

5. sastanak

- Datum: 26. listopada 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar

- Teme sastanka:
 - prijava dosadašnjih implementacija, iskustva i problema, dogovor o prijavi istih na komunikacijske platforme kako bi ovakvi sastanci bili rjeđi, a implementacija i ispravljanje grešaka efikasnija

6. sastanak

- Datum: 6. studeni 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - diskusija o dosadašnjem napretku, ažuriranje LaTeX dokumentacije

7. sastanak

- Datum: 10. studeni 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - reorganizacija tima, dodjela zadataka, organizacija vremena kako bismo izvršili što više do predaje alfa inačice aplikacije

8. sastanak

- Datum: 15. studeni 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - zaključne riječi o dovršenim generičkim funkcionalnostima i dogovor o demonstraciji (16. studenog 2021.)

9. sastanak

- Datum: 15. studeni 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - ponavljanje gradiva i učenje koda radi prvog kolokviranja

10. sastanak

- Datum: 14. prosinac 2021.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - reorganizacija tima i podjela zadataka, ispravljanje i dopunjivanje funkcionalnosti, priprema za predaju alfa inačice

11. sastanak

- Datum: 21. prosinac 2021.

- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - odgoda predaje alfa inačice na 5.1, dogovor o nastavku s radom do navedenog datuma

12. sastanak

- Datum: 4. siječanj 2022.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - sređivanje programa za predaju, diskusija o izvršenom, ažuriranje LaTeX dokumentacije

13. sastanak

- Datum: 10. siječanj 2022.
- Prisustvovali: B.Kazazić, M.Erlić, B.Pavlović, A.Pašalić, P.Sušac, V.Žunar
- Teme sastanka:
 - završni ispravci funkcionalnosti, reorganizacija tima, zadavanje završnih zadataka, vremenska organizacija – završavanje projekta

Tablica aktivnosti

Kontinuirano osvježavanje

Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti.

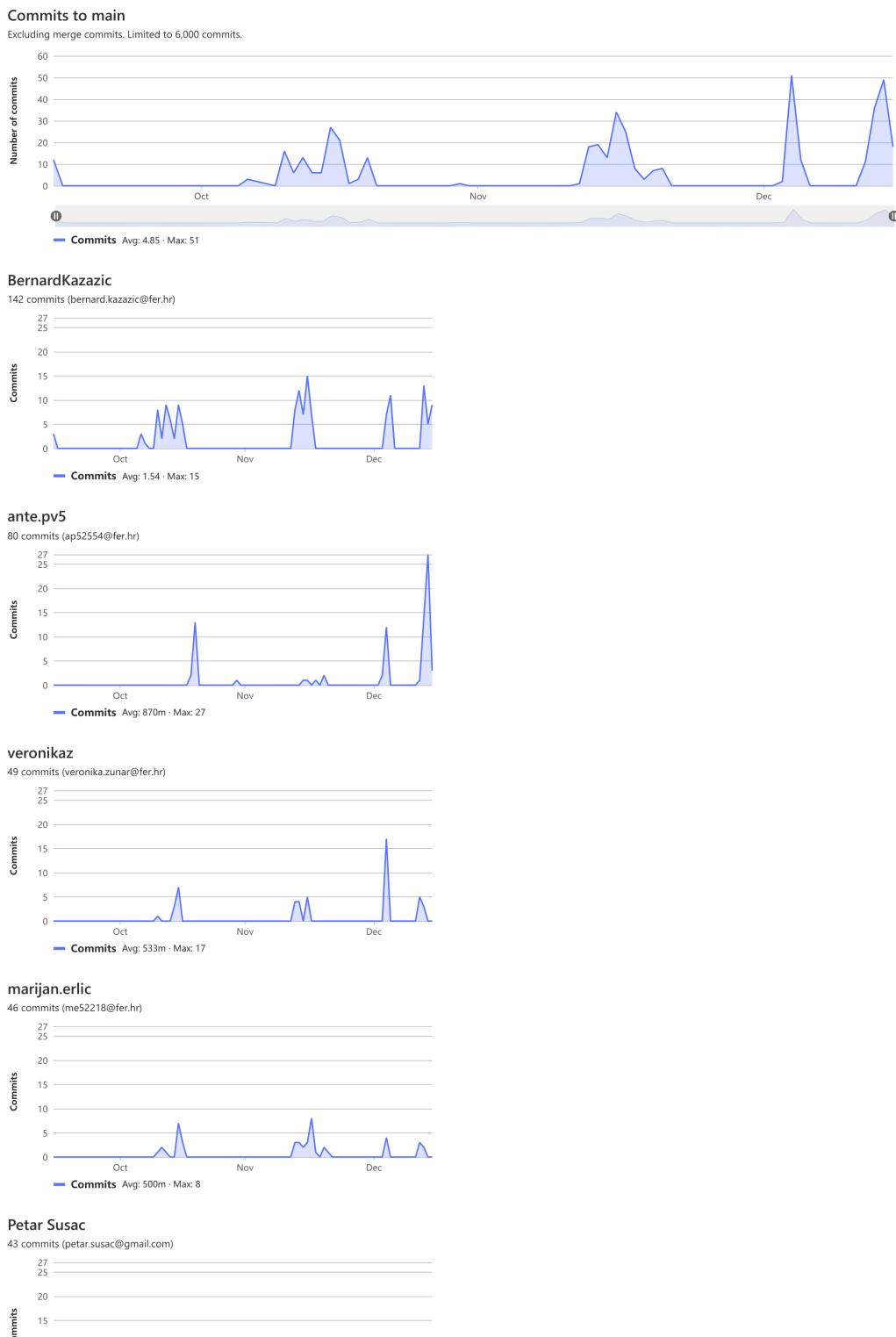
	Bernard Kazazić	Ante Pašalić	Barbara Pavlović	Luka Raspuđić	Marijan Erlić	Petar Sušac	Veronika Žunar
Upravljanje projektom	1600						
Opis projektnog zadatka		100	100		60	100	
Funkcionalni zahtjevi	30				45		
Opis pojedinih obrazaca					90		
Dijagram obrazaca	90		120		120		120
Sekvencijski dijagrami			120				120
Opis ostalih zahtjeva				60		60	
Arhitektura i dizajn sustava						60	
Baza podataka	300					480	
Dijagram razreda							60
Dijagram stanja	30		100		180		180
Dijagram aktivnosti			180				
Dijagram komponenti					90		
Korištene tehnologije i alati		45		40	30		
Ispitivanje programskog rješenja	15		120	20			
Dijagram razmještaja	200		90	90	150	120	30
Upute za puštanje u pogon	200						
Dnevnik sastajanja	100				120		
Zaključak i budući rad	5			120			60
Popis literature							60
Izrada korisničkog sučelja	60	3000		2000		2000	
Izrada backend aplikacije	3000		1800		3000		2000

	Bernard Kazazić	Ante Pašalić	Barbara Pavlović	Luka Raspudić	Marijan Erlić	Petar Sušac	Veronika Žunar
Postavljanje aplikacije na Heroku	180					120	
Pisanje LaTeX dokumentacije		240		300			

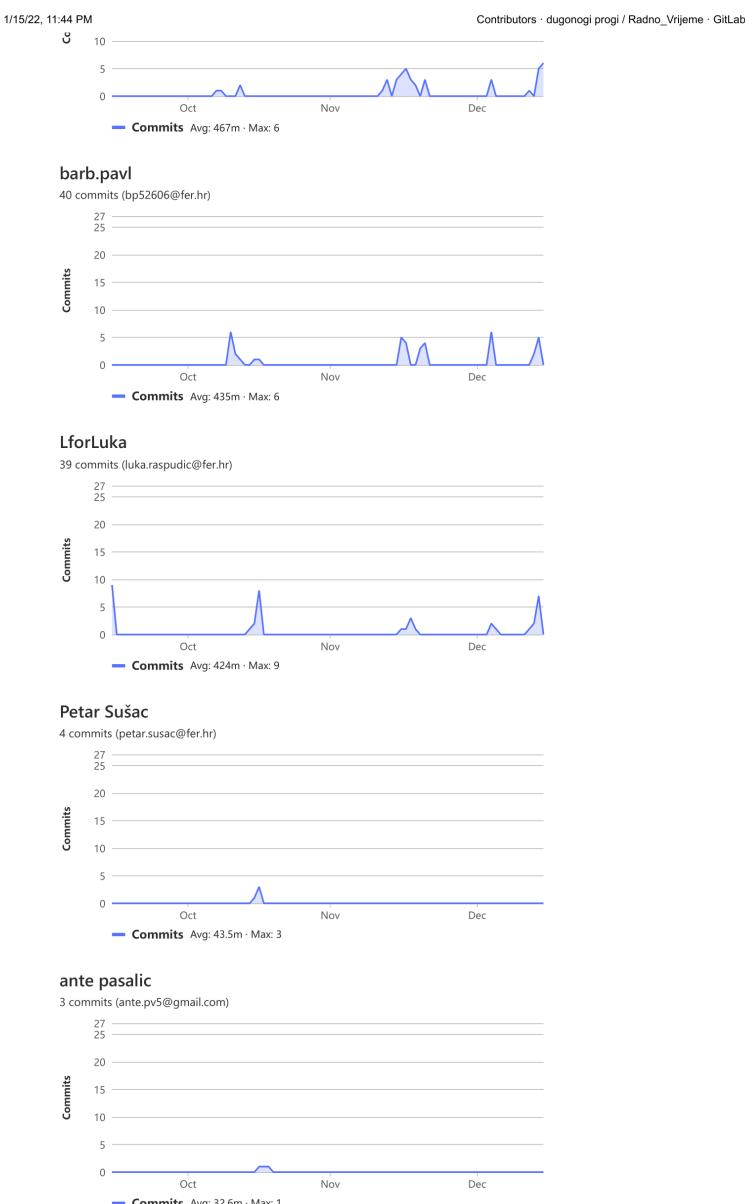
Dijagrami pregleda promjena

1/15/22, 11:44 PM

Contributors · dugonogi progi / Radno_Vrijeme · GitLab

https://gitlab.com/dugonogi-progi/radno_vrijeme/-/graphs/main

1/2



Slika 6.1: GitLab dijagram promjena