

Programsko inženjerstvo

Ak. god. 2020./2021.

Agilna platforma

Dokumentacija, Rev. 2

Grupa: *WorldPIIS*

Voditelj: *Miho Hren*

Datum predaje: *14. 01. 2021.*

Nastavnik: *Nikolina Frid*

Sadržaj

1	Dnevnik promjena dokumentacije	3
2	Opis projektnog zadatka	5
3	Specifikacija programske potpore	9
3.1	Funkcionalni zahtjevi	9
3.1.1	Obrasci uporabe	11
3.1.2	Sekvencijski dijagrami	21
3.2	Ostali zahtjevi	25
4	Arhitektura i dizajn sustava	27
4.1	Baza podataka	29
4.1.1	Opis tablica	30
4.1.2	Dijagram baze podataka	32
4.2	Dijagram razreda	33
4.3	Dijagram stanja	36
4.4	Dijagram aktivnosti	37
4.5	Dijagram komponenti	38
5	Implementacija i korisničko sučelje	39
5.1	Korištene tehnologije i alati	39
5.2	Ispitivanje programskog rješenja	40
5.2.1	Ispitivanje komponenti	40
5.2.2	Ispitivanje sustava	45
5.3	Dijagram razmještaja	52
5.4	Upute za puštanje u pogon	53
5.4.1	Operacijski sustav	53
5.4.2	Preuzimanje programskog rješenja	53
5.4.3	Baza podataka	53
5.4.4	Konfiguracija poslužitelja baze podataka	53
5.4.5	Konfiguracija baze podataka	53

5.4.6	Punjenje baze podataka	54
5.4.7	Pokretanje baze podataka	54
5.4.8	Javna vrata servera	54
5.4.9	Pokretanje web poslužitelja	54
5.4.10	Konfiguracija veze s bazom podataka	54
5.4.11	Pokretanje REST poslužitelja	55
5.4.12	Konfiguriranje adrese spajanja	55
5.4.13	Domena aplikacije	55
5.4.14	Korištenje aplikacije	55
6	Zaključak i budući rad	56
	Popis literature	58
	Indeks slika i dijagrama	59
	Dodatak: Prikaz aktivnosti grupe	60

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Preuzet predložak sa stranice FER-a	Svi	14.10.2020.
0.2	Dodan opis projektnog zadatka	Pavlic	19.10.2020.
0.3	Dodan popis dionika te aktera i njihovih funkcionalnih zahtjeva Dodani ostali zahtjevi	Hren	19.10.2020.
0.4	Dodan opis obrazaca uporabe	Fučec, Jukić, Lukenda	19.10.2020.
0.5	Dodano poglavlje Korištene tehnologije i alati	Habuš	22.10.2020.
0.6	Dodani dijagrami obrazaca uporabe i sekvencijski dijagrami	Đokić, Lukenda	25.10.2020.
0.7	Ažuriran opis projektnog zadatka - dodani opis Scrum i Kanban metodologija	Pavlic	27.10.2020.
0.8	Dodan opis arhitekture sustava, opis baze podataka i dijagram baze podataka	Hren	2.11.2020.
0.9	Dodani sekvencijski dijagrami	Đokić	7.11.2020.
0.10	Dodani dijagrami razreda	Hren	7.11.2020.
0.11	Ažurirani dijagrami razreda i sekvencijski dijagrami	Đokić, Hren	9.11.2020.
1.0	Korigiranje teksta i provjera dokumentacije	Fučec, Habuš, Lukenda	12.11.2020.
1.1	Dodana uputa za puštanje u pogon	Hren	14.1.2021.
1.2	Ažurirani dnevnik sastajanja i arhitektura	Hren	14.1.2021.
1.3	Opisano ispitivanje programskog rješenja	Lukenda	14.01.2021.
1.4	Dodan dijagram razmještaja	Fučec	14.01.2021.
1.5	Dodani dijagram aktivnosti i dijagram komponenti	Habuš	14.01.2021.

Rev.	Opis promjene/dodatka	Autori	Datum
1.6	Dodan dijagram stanja	Đokić	14.01.2021.
1.7	Dodan dijagram pregleda promjena i ažuriran dnevnik promjena	Hren	14.01.2021.
2.0	Ispravljene gramatičke pogreške	Svi	14.01.2021.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti platformu za upravljanje zadacima, koji se razvijaju agilnom metodologijom, na projektima unutar neke tvrtke. Platforma se temelji na osnovama metodologija Scrum i Kanban.

Scrum je radni okvir koji pomaže zajedničkom radu timova. Opisuje skup sastanaka, alata i uloga koji zajedno djeluju kako bi pomogli timovima da upravljaju svojim radom i kvalitetno ga strukturiraju. Potiče timove da se samoorganiziraju prilikom rada na problemu i uče kroz iskustva te da razmišljaju o svojim usponima i padovima kako bi se kontinuirano mogli usavršiti. Zbog toga što je usmjeren na kontinuirano usavršavanje često ga se smatra agilnim okvirom za upravljanje projektima, iako zapravo ne može biti agilniji jer je potrebna posvećenost cijelog tima da bi se promijenio način na koji razmišljaju o davanju vrijednosti korisnicima. Scrum je heuristički okvir jer se temelji na kontinuiranom učenju i prilagodbi. Priznaje da na početku projekta tim ne zna sve i da će se razvijati kroz radno iskustvo. Strukturiran je na način da pomogne timovima da se prirodno prilagode promjenjivim uvjetima i zahtjevima korisnika tako da se ponovno određuju prioritete i postoje kratki ciklusi objavljivanja kako bi tim mogao stalno usvajati znanja i usavršavati se. U Scrumu postoje tri artefakta. Artefakti su nešto što mi izrađujemo, poput alata za rješavanje problema. Ta tri artefakta su:

- product backlog - glavni popis poslova
- sprint backlog - popis stavki, korisničkih priča ili ispravaka pogreški
- increment - cilj sprinta, korisni krajnji proizvod sprinta

U timu Scrum-a trebaju postojati tri posebne uloge:

- vlasnik proizvoda - fokusiran na razumijevanje zahtjeva poslovanja, tržišta i kupaca, određuju prioritet poslova
- Scrum master - glavni voditelj, planira potrebne resurse, podučava timove i vlasnike proizvoda te pomaže pri optimizaciji transparentnosti i tijeka isporuke

- razvojni tim - 5 do 7 članova, upravlja planom za svaki sprint, predviđa vrijeme potrebno za posao

Kanban je radni okvir koji se koristi za razvoj agilnog softvera. Zahtjeva punu transparentnost posla. Elementi posla prikazani su vizualno na kanban ploči kako bi se članovima tima u svakom momentu omogućio uvid u stanje svakog od dijelova posla. Kanban ploča je alat koji se koristi za vizualizaciju i optimizaciju tijeka rada među timovima. Funkcije kanban ploče uz vizualizaciju rada su standardizacija tijeka rada tima te prepoznavanje i rješavanje ovisnosti i blokatora. Kanban ploča ima tijek rada u tri koraka:

- To Do
- In Progress
- Done

Kanban timovi imaju svaki element posla predstavljen kao posebnu karticu na ploči. Glavna svrha kartica je omogućiti članovima da prate napredak rada na što zorniji način. Kanban kartice sadrže ključne informacije o elementu posla kojeg predstavljaju. Kanban svim time omogućuje fleksibilnost planiranja, jednom kad tim dovrši neki element posla pređe se na sljedeći element posla s vrha product backloga. Vlasnik proizvoda ima mogućnost prioritetno odrediti redoslijed u product backlogu i sve dok on na njegovom vrhu drži najvažnije elemente posla nema straha od toga da se tvrtki ne isporučuje maksimalna vrijednost. Samim time nema potrebe za promjenama postavljene duljine rada.

Platforma omogućuje grupiranje zadataka, definiranih na projektu, u product backlog. Iz njega se u svakoj iteraciji projekta odabire skupina zadataka na kojima će se raditi i oni napreduju kroz faze te se za svaki zadatak prati napredak. Faze napredovanja su:

- oblikovanje
- implementacija
- ispitivanje
- puštanje u pogon

Također uz zadatke je moguće vezati probleme koji se pojavljuju tijekom njihovog rješavanja. Zadatak može biti preuzet od strane najviše jednog člana tima. Svaki zadatak ima:

- svoj ID
- sažeti naziv
- detaljan opis
- prioritet
- krajnji rok završetka

Projekti su definirani tako da se rade u timovima od 5 do 8 članova, gdje su članovi tima razvojni inženjeri i voditelj tima. Više timova može biti udruženo u radnu skupinu koju vodi koordinator. Koordinator i voditelj s članovima svog tima komuniciraju preko Google Calendar-a koji je integriran u platformu.

Platformi mogu pristupiti samo registrirani korisnici koji su zaposlenici tvrtke koja je vlasnik projekta. Korisnik je registriran ako je dodan u bazu podataka od strane admina baze. Registrirani korisnik prijavljuje se pomoću korisničkog imena i lozinke. Nakon uspješne prijave u sustav korisniku se prikazuje inačica Kanban ploče, koja služi za vizualizaciju zadatka i njihovom prolasku kroz faze na projektu. Što je korisniku prikazano na Kanban ploči ovisni o tome na kojoj je funkciji unutar projekta.

Uprava tvrtke je vlasnik projekta. Prilikom prijave u aplikaciju upravi tvrtke je na Kanban ploči omogućen prikaz svih projekata tvrtke tako da može pratiti njihov napredak. Od ponuđenih projekata može odabrati konkretni projekt te zatim pratiti razvoj događaja na njegovoj Kanban ploči. Također unutar projekta može odabrati određeni tim i tako dobiti uvid u njegovu Kanban ploču. Iako joj je omogućeno praćenje svih projekata, direktno ne sudjeluje u izradi niti jednog projekta.

Zaposlenik je svatko tko radi u tvrtki na bilo kojoj funkciji te je samim time registriran u bazi podataka. Može se prijaviti u aplikaciju sa svojim korisničkim imenom i lozinkom. Omogućen je prikaz njegovog profila na kojem su prikazani:

- njegovo ime
- prezime
- broj mobitela
- email

kako bi se s njim moglo stupiti u kontakt. On može mijenjati svoj broj mobitela i email te postaviti novu lozinku.

Koordinator je zaposlenik u tvrtki koji može vidjeti sve zadatke i njihovo trenutno stanje te tako pratiti napredak projekta. Od ponuđenih timova unutar projekta može odabrati konkretni i tako dobiti uvid u njegovu Kanban ploču. Na temelju viđenog može dogovarati sastanke s voditeljima timova. Kako ima uvid u sve timove na projektu, omogućeno mu je stvaranje radnih skupina koje se sastoje od više timova kojima on upravlja. Nije mu omogućeno dodavati i uređivati zadatke kao ni utjecati na njihovo izvođenje.

Voditelj tima je zaposlenik u tvrtki koji je dio jednog od timova na projektu i njega predvodi. Zadužen je za stvaranje i održavanje backloga te je jedina osoba u timu koja ima pravo mijenjati backlog. Dozvoljeno mu je preuzimanje jednog ili više zadataka s backloga i rad na njima. Ako je preuzeo neki od zadataka na sebe odgovoran je za izvještavanje o napretku zadatka kroz faze te o problemima na koje je naišao prilikom rješavanja. Ima pristup samo Kanban ploči svojeg tima te tako može pratiti razvoj. Na temelju viđenog može sazvati sastanak s članovima svog tima te zabilježiti termin sastanka u raspored sastanaka. Jedan voditelj može voditi samo jedan tim. Ima uvid u raspored sastanaka s koordinatorom. Prema dobivenim zaslugama kao voditelj može biti promaknut u koordinatora.

Razvojni inženjer je zaposlenik u tvrtki koji je dio samo jednog od timova na projektu. Kao članu tima omogućeno mu je preuzimanje jednog ili više zadataka iz backloga te je nakon preuzimanja odgovoran za izvještavanje o napredovanju zadatka kroz faze i problemima s kojima se suočava prilikom rješavanja. Ima uvid u raspored sastanaka s voditeljem tima. U skladu sa svojim zaslugama može biti promaknut u voditelja tima.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Uprava tvrtke (naručitelj)
2. Razvojni inženjer
3. Voditelj tima
4. Koordinator timova
5. Tim koji je razvio aplikaciju

Aktori i njihovi funkcionalni zahtjevi:

1. Uprava tvrtke (inicijator) može:
 - (a) se prijaviti u sustav
 - (b) pregledavati napredak bilo kojeg projekta
2. Zaposlenik (razvojni inženjer/voditelj/koordinator) (inicijator) može:
 - (a) se prijaviti u sustav
 - (b) pregledavati profil
 - (c) mijenjati email, korisničko ime i lozinku
3. Zaposlenik (razvojni inženjer) (inicijator) može:
 - (a) preuzimati zadatke
 - (b) izvještavati o stanju zadatka
 - (c) pomicati zadatke kroz faze na kanban ploči
 - (d) gledati raspored sastanaka s voditeljom
4. Zaposlenik (voditelj tima) (inicijator) može:
 - (a) dodavati zadatke na backlog
 - (b) dogovarati sastanke s razvonim inženjerima
 - (c) preuzimati zadatke

- (d) izvještavati o stanju projekta
- (e) pomicati zadatke kroz faze na kanban ploči
- (f) gledati raspored sastanaka s koordinatorom

5. Zaposlenik (koordinator) (inicijator) može:

- (a) stvarati radne skupine
- (b) dogovarati sastanke s voditeljima timova
- (c) pratiti napredak projekta
- (d) izvještavati o stanju projekta

6. Baza podataka (sudionik):

- (a) pohranjuje podatke o svim zaposlenicima i njihovim titulama
- (b) pohranjuje podatke o svim projektima, zadacima i stanjima zadataka

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Prijava u sustav

- **Glavni sudionik:** Zaposlenik
- **Cilj:** Pristupiti korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran u bazi
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime/lozinka
 1. Sustav obavještava korisnika o neuspjeloj prijavi i vraća ga na stranicu za unos korisničkog imena i lozinke

UC2 - Pregled svih projekata

- **Glavni sudionik:** Uprava tvrtke
- **Cilj:** Pregled svih aktivnih projekata
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima dodijeljene ovlasti uprave
- **Opis osnovnog tijeka:**
 1. Korisnik u aplikaciji odabire opciju "Pregled projekata"
 2. Prikaže se lista svih aktivnih projekata

UC3 - Pregled napretka projekta

- **Glavni sudionik:** Uprava tvrtke
- **Cilj:** Pregled napretka svih aktivnih projekata
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i ima dodijeljene ovlasti uprave
- **Opis osnovnog tijeka:**
 1. Korisnik u aplikaciji odabire opciju "Pregled projekata"
 2. Prikaže se lista svih aktivnih projekata
 3. Korisnik odabire željeni projekt

4. Prikazu se informacije o odabranom projektu i njegov napredak

UC4 - Pregled profila

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregled korisničkih podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za pregled profila
 2. Otvara se stranica s podacima o korisniku

UC5 - Uređivanje profila

- **Glavni sudionik:** Korisnik
- **Cilj:** Mijenjanje osobnih podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za mijenjanje osobnih podataka
 2. Otvara se stranica za mijenjanje osobnih podataka
 3. Korisnik mijenja osobne podatke
 4. Korisnik sprema promjene
 5. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 3.a Korisnik promijeni svoje osobne podatke, ali ne odabere opciju za spremanje promjena
 1. Sustav obavjestava korisnika da nije spremio podatke prije izlaska iz prozora

UC6 - Pregled zadataka

- **Glavni sudionik:** Developer
- **Cilj:** Pregled svih zadataka koji su dostupni za preuzimanje
- **Sudionici:** Baza podataka
- **Preduvjet:** Developer je prijavljen
- **Opis osnovnog tijeka:**
 1. Developer odabire opciju za prikaz backloga projekta
 2. Otvara se stranica sa zadacima odabranog projekta

3. Korisnik mijenja osobne podatke

UC7 - Preuzimanje zadatka

- **Glavni sudionik:** Developer
- **Cilj:** Preuzimanje zadatka s backloga projekta
- **Sudionici:** Baza podataka
- **Preduvjet:** Developer je prijavljen i nalazi se na stranici sa zadacima projekta
- **Opis osnovnog tijeka:**
 1. Developer odabire opciju za preuzimanje pored željenog zadatka
 2. Ažurira se baza podataka

UC8 - Mijenjanje faze zadatka

- **Glavni sudionik:** Developer
- **Cilj:** Promijeniti fazu preuzetog zadatka
- **Sudionici:** Baza podataka
- **Preduvjet:** Developer je prijavljen
- **Opis osnovnog tijeka:**
 1. Developer odabire opciju za unaprjeđenje faze pored željenog zadatka
 2. Ažurira se baza podataka
- **Opis mogućih odstupanja:**
 - 1.a Developer odabire zadatak koji nije on preuzeo
 1. Sustav obavještava korisnika da nema pravo mijenjati fazu zadatka koji nije njegov

UC9 - Izvještaj o problemima

- **Glavni sudionik:** Developer
- **Cilj:** Izvijestiti vođu tima o problemima u rješavanju zadatka
- **Sudionici:** Baza podataka
- **Preduvjet:** Developer je prijavljen
- **Opis osnovnog tijeka:**
 1. Developer odabire opciju za dodavanje problema kod željenog zadatka
 2. Otvara se forma za upis kratkog opisa problema
 3. Developer pohranjuje problem
 4. Ažurira se baza podataka
- **Opis mogućih odstupanja:**
 - 1.a Developer odabire zadatak koji nije on preuzeo

1. Sustav obavještava korisnika da nema pravo opisivati problem za zadatak koji nije njegov

UC10 - Pregled sastanaka

- **Glavni sudionik:** Developer
- **Cilj:** Pregled termina sastanaka s vođom tima
- **Sudionici:** Baza podataka
- **Preduvjet:** Developer je prijavljen
- **Opis osnovnog tijeka:**
 1. Developer odabire opciju za prikaz kalendara sastanaka
 2. Otvara se kalendarska usluga s označenim terminima sastanka

UC11 - Dodavanje zadataka

- **Glavni sudionik:** Vođa tima
- **Cilj:** Dodavanje novih zadataka u projektni backlog
- **Sudionici:** Baza podataka
- **Preduvjet:** Vođa tima je prijavljen i nalazi se na stranici sa zadatcima projekta
- **Opis osnovnog tijeka:**
 1. Vođa tima odabire opciju za dodavanje novog zadatka
 2. Otvara se obrazac na kojem mora upisati naziv zadatka
 3. Vođa tima predaje obrazac
 4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 3.a Vođa tima predaje obrazac bez ispunjenog naziva zadatka
 1. Sustav obavještava vođu tima da mora ispuniti obrazac prije predaje

UC12 - Preuzimanje zadatka

- **Glavni sudionik:** Voditelj tima ili razvojni inženjer
- **Cilj:** Preuzeti zadatak
- **Sudionici:** Baza podataka
- **Preduvjet:** Glavni sudionik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju „Kanban ploča“
 2. Aplikacija korisniku na backlogu prikazuje sve zadatke dostupne za preuzimanje
 3. Korisnik odabire željeni zadatak

4. Aplikacija korisniku prikazuje informacije o zadatku.
5. Korisnik odabire opciju „preuzmi zadatak“
6. Korisnik prima obavijest o uspješnom preuzimanju
7. Baza podataka se ažurira

UC13 - Dogovor sastanaka voditelja tima s razvojnim inženjerima

- **Glavni sudionik:** Voditelj tima
- **Cilj:** Dogovor sastanaka s razvojnim inženjerima
- **Sudionici:** Razvojni inženjeri
- **Preduvjet:** Voditelj tima je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik na početnoj stranici odabire opciju „Sastanci“
 2. Aplikacija korisniku prikazuje kalendarsku uslugu Google Calendar
 3. Korisnik unutar kalendarske usluge stvara sastanak, a kalendarska usluga automatski može poslati e-mail pozivnicu razvojnim inženjerima (inženjeri imaju opciju potvrde dolaska)

UC14 - Izvještavanje koordinatora o problemima

- **Glavni sudionik:** Voditelj tima
- **Cilj:** Izvijestiti koordinatora o problemima projekta
- **Sudionici:** Koordinator, baza podataka
- **Preduvjet:** Voditelj tima je prijavljen, postoje problemi u zadacima/projektu
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju „Kanban ploča“
 2. Aplikacija prikazuje ploču na kojoj postoje zadatci koji imaju određeni problem te su oni označeni crvenom bojom
 3. Korisnik može izvijestiti koordinatora o ovim problemima putem elektroničke pošte

UC15 - Pregled sastanaka voditelja tima

- **Glavni sudionik:** Voditelj tima
- **Cilj:** Pregled sastanaka s koordinatorom ili razvojnim inženjerima
- **Sudionici:** Koordinator, razvojni inženjeri
- **Preduvjet:** Voditelj tima je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik na početnoj stranici odabire opciju „Sastanci“

2. Aplikacija korisniku prikazuje kalendarsku uslugu Google Calendar
3. Korisnik unutar kalendarske usluge može pregledati sastanke koje je već dogovorio sa članovima svog tima. Također, može vidjeti i sastanke koje je postavio koordinator te potvrditi svoj dolazak.

UC16 - Dogovor sastanak s voditeljima timova

- **Glavni sudionik:** Koordinator
- **Cilj:** Dogovor sastanak s voditeljima timova radne skupine
- **Sudionici:** Voditelj tima
- **Preduvjet:** Koordinator je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik na početnoj stranici odabire opciju „Sastanci“
 2. Aplikacija korisniku prikazuje kalendarsku uslugu Google Calendar
 3. Korisnik unutar kalendarske usluge stvara sastanak s voditeljima timova radne skupine, a kalendarska usluga automatski može poslati e-mail pozivnicu gostima.

UC17 - Praćenje napretka projekta

- **Glavni sudionik:** Koordinator
- **Cilj:** Pratiti napredak projekta
- **Sudionici:** Baza podataka
- **Preduvjet:** Koordinator je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik na početnoj stranici odabire opciju „Popis timova unutar projekta“
 2. Korisnik odabire tim čiji napredak želi provjeriti
 3. Aplikacija prikazuje Kanban ploču odabranog tima. Korisnik može vidjeti koji je zadatak u fazi oblikovanja, implementacije, ispitivanja, puštanja u pogon...
 4. Korisnik odabire neki zadatak
 5. Aplikacija prikazuje informacije o tom zadatku

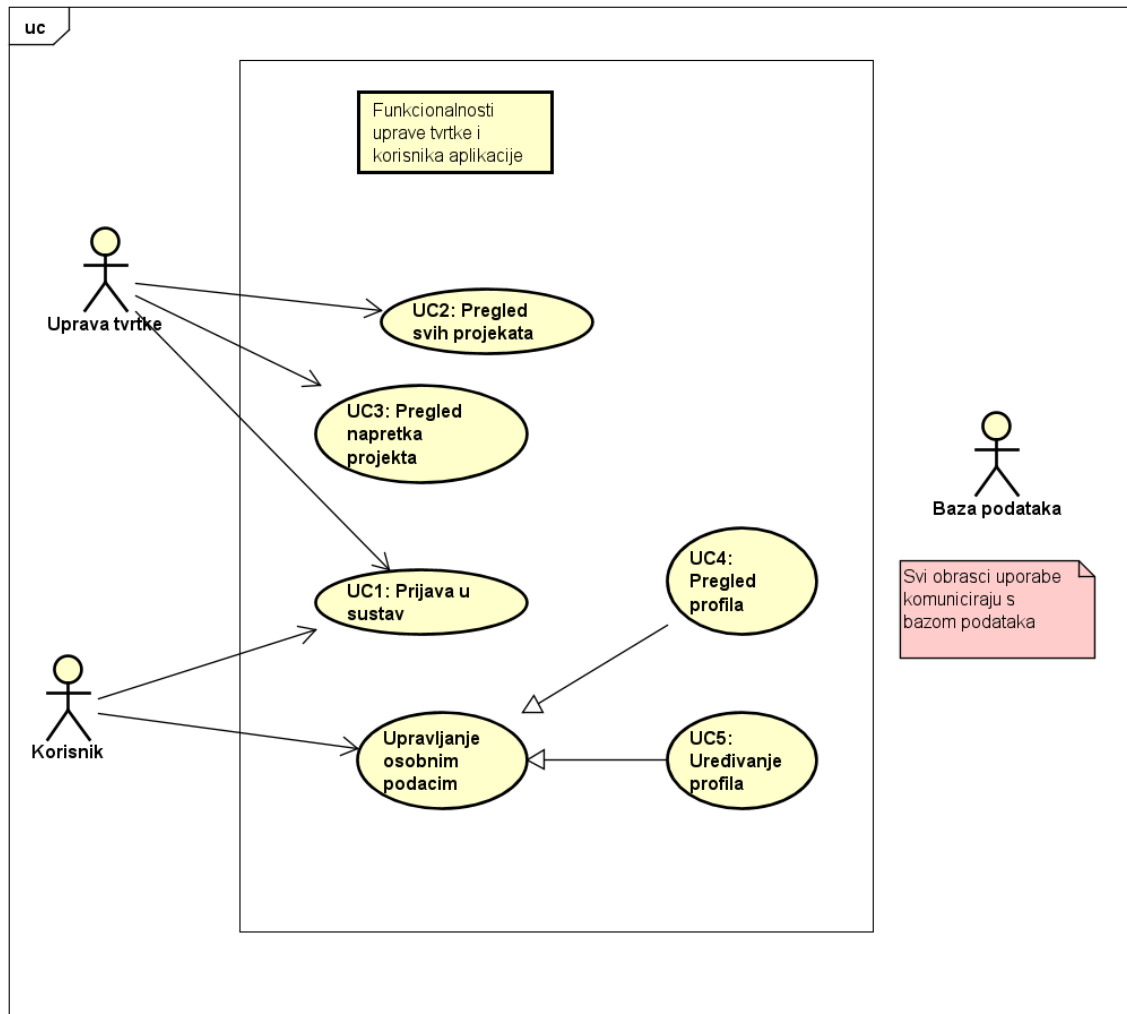
UC18 - Stvaranje radne skupine

- **Glavni sudionik:** Koordinator
- **Cilj:** Stvoriti radnu skupinu timova za projekt
- **Sudionici:** Baza podataka, Voditelj tima, Razvojni inženjer

- **Preduvjet:** Koordinator je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju „Radne skupine“
 2. Korisnik odabire opciju „Stvori novu radnu skupinu“
 3. Korisnik ispunjava informacije o radnom skupini (odabir timova koji će pripadati toj radnoj skupini, ime...)
 4. Nakon uređivanja potvrđuje stvaranje radne skupine odabirom opcije „Stvori radnu skupinu“
 5. Promjene se upisuju u bazu podataka
- **Opis mogućih odstupanja:**
 - 3.a Korisnik je odabrao tim koji je već u nekoj drugoj radnoj skupini
 1. Aplikacija obavještava korisnika o neuspjelom odabiru tima
 2. Korisnik odabire drugi tim koji nije pridružen nekoj radnoj skupini ili korisnik odustane od odabira dodatnih timova te završava unos
 - 3.b Korisnik je unio podatke radne skupine koji se već koriste
 1. Aplikacija obavještava korisnika da je ime/ID/... radne skupine već zauzeto
 - 3.c Korisnik nije odabrao niti jedan tim ili nije unio ime/ID/...
 1. Aplikacija ne dozvoljava odabir opcije „Stvori radnu skupinu“ dok ne postoji barem jedan tim u radnoj skupini/nije dodijeljeno ime/ID/....
 - 3.d Korisnik je unio podatke te želi izaći iz prozora, ali nije potvrdio stvaranje radne grupe
 1. Aplikacija obavještava korisnika da nije spremio podatke prije izlaska iz prozora

UC19 - Izvještavanje uprave o problemima

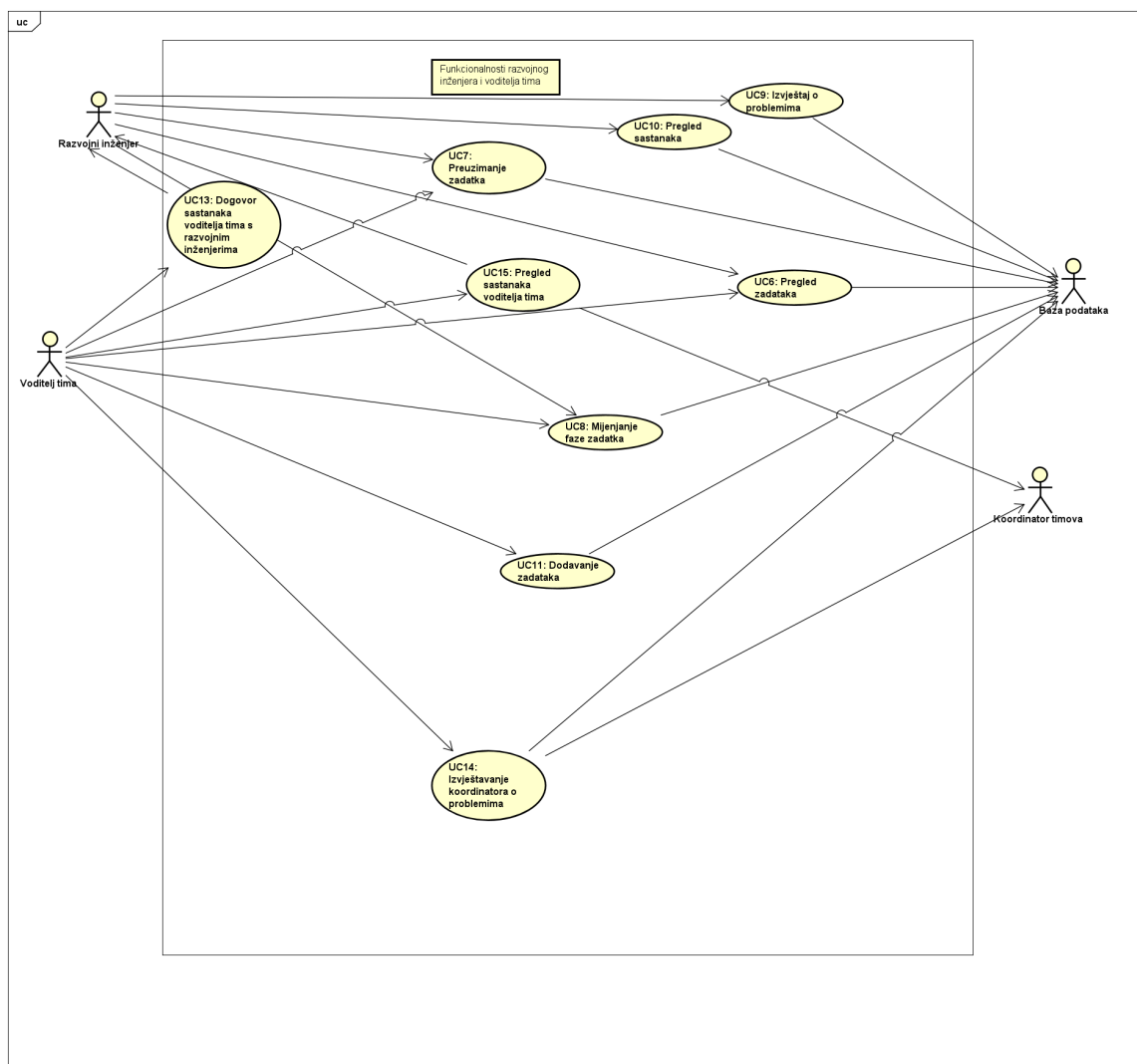
- **Glavni sudionik:** Koordinator
- **Cilj:** Izvijestiti upravu o problemima projekta
- **Sudionici:** Uprava tvrtke, baza podataka
- **Preduvjet:** Koordinator je prijavljen, postoje problemi u zadacima/projektu
- **Opis osnovnog tijeka:**
 1. Korisnik na početnoj stranici odabire opciju „Popis timova unutar projekta“
 2. Korisnik odabire tim čije probleme želi provjeriti
 3. Korisnik odabire opciju „Kanban ploča“



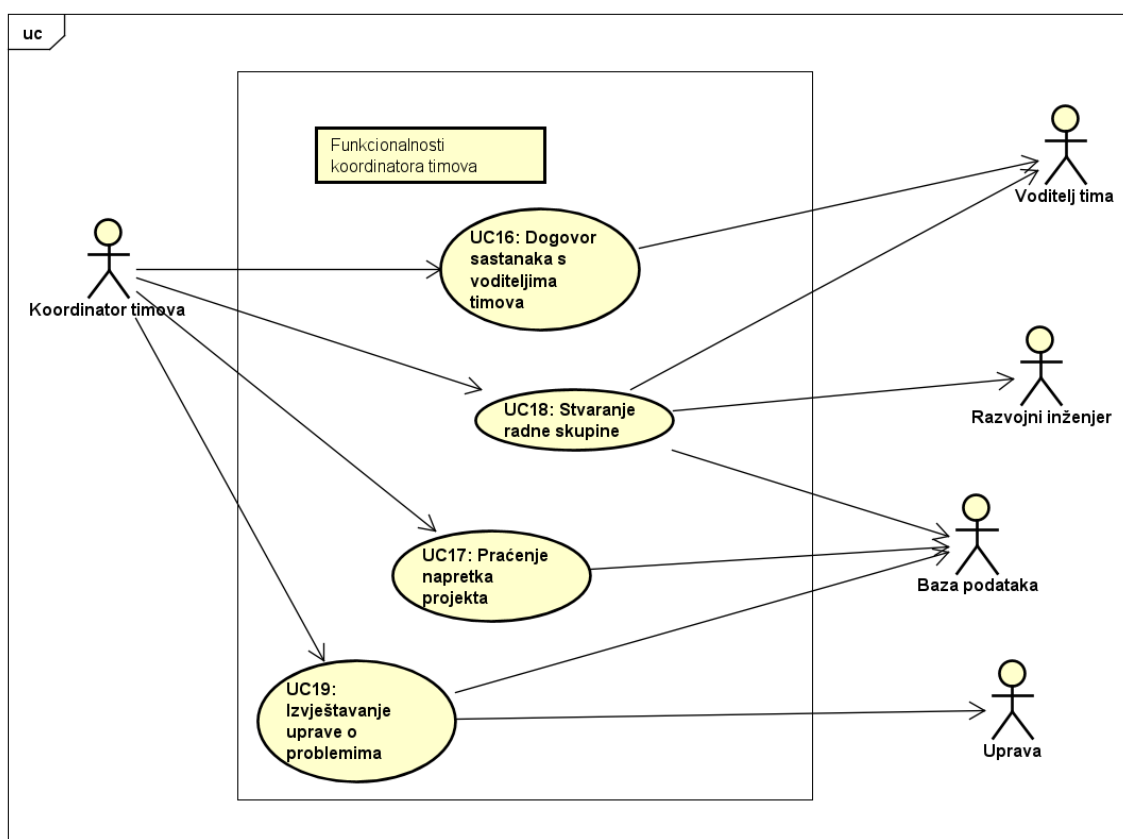
Slika 3.1: Dijagram obrasca uporabe, funkcionalnosti uprave tvrtke i korisnika

4. Aplikacija prikazuje ploču na kojoj postoje zadatci koji imaju određeni problem te su oni označeni crvenom bojom
5. Korisnik može izvijestiti upravu tvrtke o ovim problemima putem elektroničke

Dijagrami obrazaca uporabe



Slika 3.2: Dijagram obrasca uporabe, funkcionalnosti razvojnog inženjera i voditelja tima

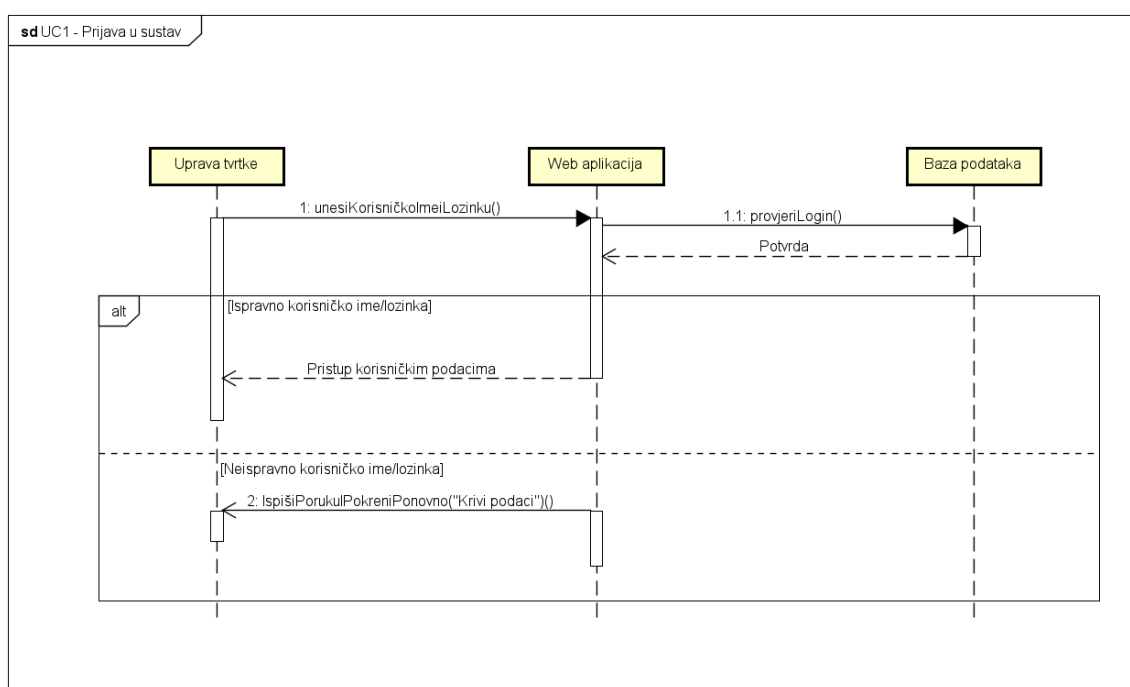


Slika 3.3: Dijagram obrasca uporabe, funkcionalnosti koordinatora timova

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC1 - Prijava u sustav

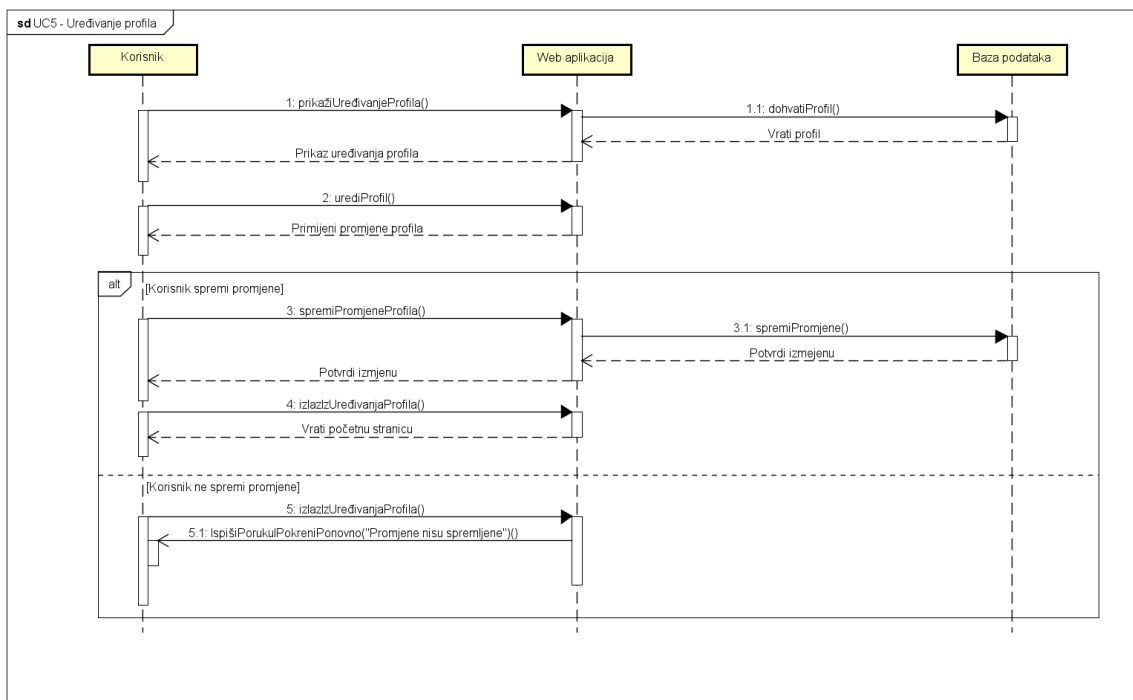
Član uprave tvrtke unosi korisničko ime i lozinku kako bi se mogao prijaviti u sustav. Poslužitelj pomoću baze podataka provjerava postoji li uneseno korisničko ime i pripada li mu navedena lozinka. Ako su podaci točni članu uprave tvrtke odobrava se ulaz u sustav. Ukoliko podaci nisu ispravni, sustav obavještava člana uprave tvrtke o tome uz poruku.



Slika 3.4: Sekvencijski dijagram za UC1

Obrazac uporabe UC5 - Uređivanje profila

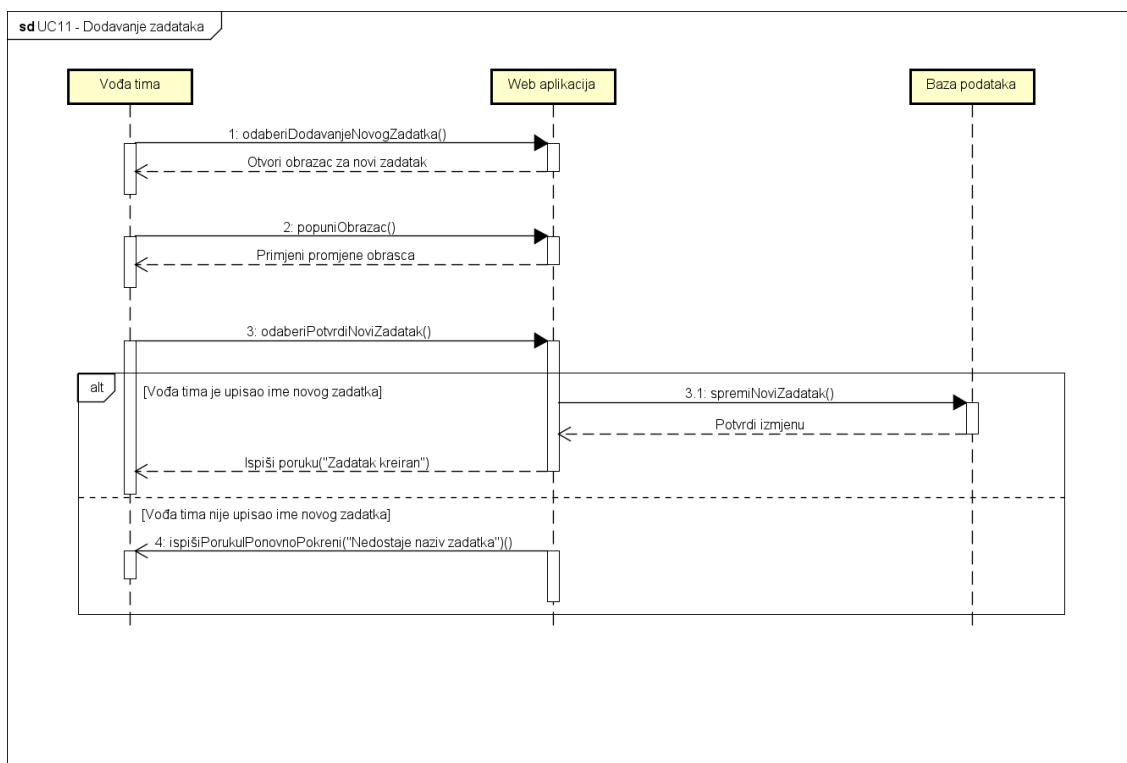
Korisnik traži pristup svojem profilu kako bi ga mogao urediti. Poslužitelj dohvaća korisnikov profil i prikazuje ga. Korisnik na poslužitelju radi promjene prije nego odabere opciju spremanja promjena na profilu. Poslužitelj promjene profila prosljeđuje bazi podataka, a korisnik od poslužitelja traži izlazak iz uređivanja profila. Ukoliko korisnik nije spremio promijenjene podatke, a želi izaći iz uređivanja profila, poslužitelj ga obavještava o tome uz poruku.



Slika 3.5: Sekvencijski dijagram za UC5

Obrazac uporabe UC11 - Dodavanje zadatka

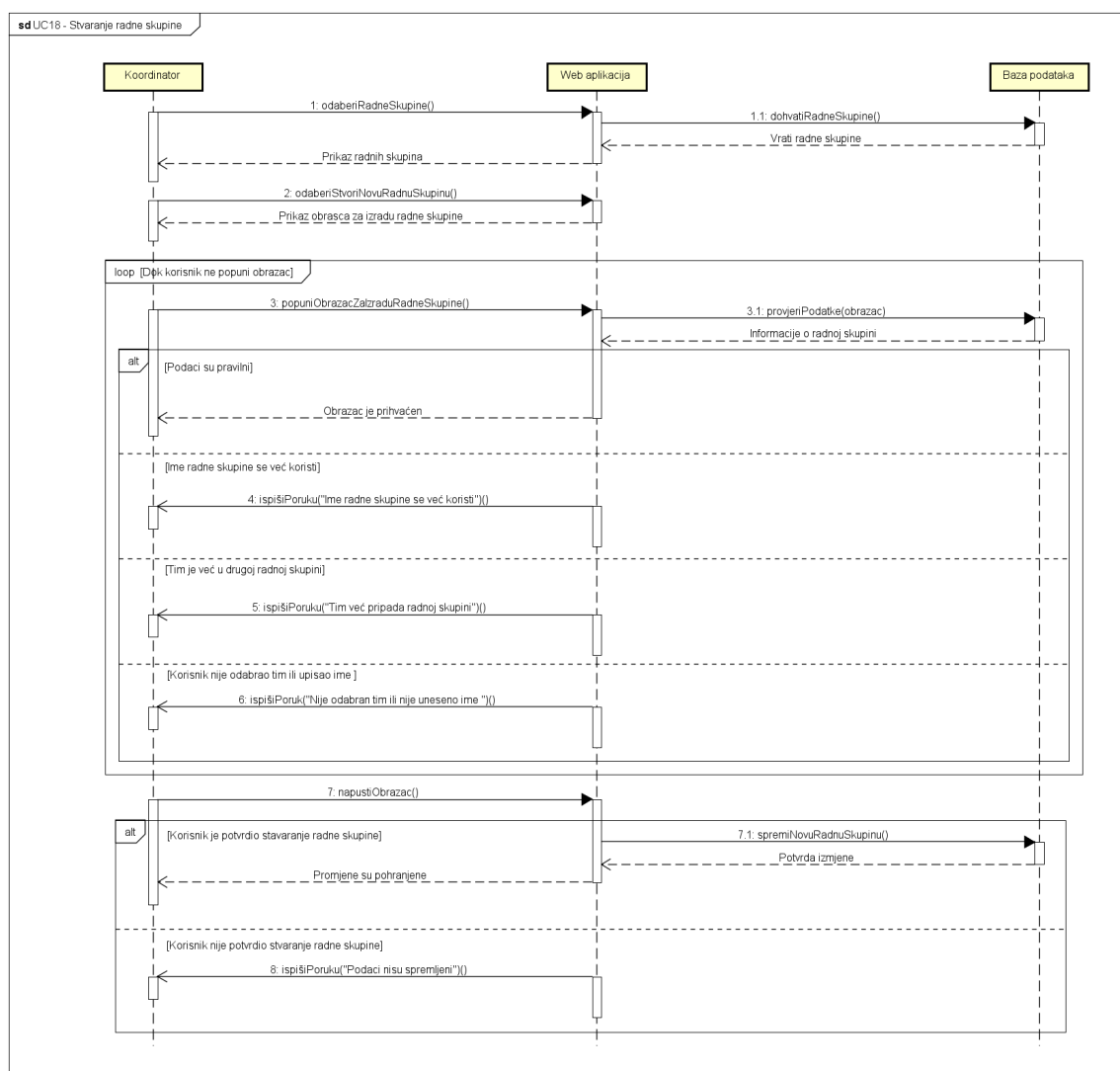
Vođa tima odabire opciju za dodavanje novog zadatka. Poslužitelj korisniku vraća obrazac koji on treba popuniti za izradu novog zadatka. Vođa tima na poslužitelju popunjava obrazac za izradu novog zadatka. Vođa tima zatim potvrđuje i sprema novi zadatak. Ukoliko vođa tima nije upisao ime novog zadatka sustav ga obavještava o tome uz poruku. U suprotnom slučaju poslužitelj prosljeđuje novi zadatak u bazu podataka.



Slika 3.6: Sekvencijski dijagram za UC11

Obrazac uporabe UC18 - Stvaranje radne skupine

Koordinator traži prikaz radnih grupa. Poslužitelj dohvaća postojeće radne grupe i prikazuje ih. Koordinator odabire opciju za izradu nove radne skupine zbog čega mu poslužitelj vraća obrazac za izradu nove radne skupine. Korisnik popuni obrazac nakon čega poslužitelj uspoređuje podatke iz obrasca s onima iz baze podataka. Ukoliko koordinator nije upisao ime radne skupine ili nije odabrao tim poslužitelj će ga obavijestiti uz pripadajuću poruku i nastaviti s ispunjavanjem obrasca. Ukoliko je koordinator odabrao tim koji već pripada radnoj skupini ili je odabrao naziv koji se već koristi, također će biti obaviješten od strane poslužitelja i nastaviti će se s popunjavanjem obrasca. Kada su podaci ispravni koordinator želi napustiti obrazac. Ako je koordinator odabrao opciju spremanja radne skupine, onda ju poslužitelj sprema na bazu podataka. U suprotnom slučaju poslužitelj obavještava koordinatora da nije spremio novu radnu skupinu.



Slika 3.7: Sekvencijski dijagram za UC18

3.2 Ostali zahtjevi

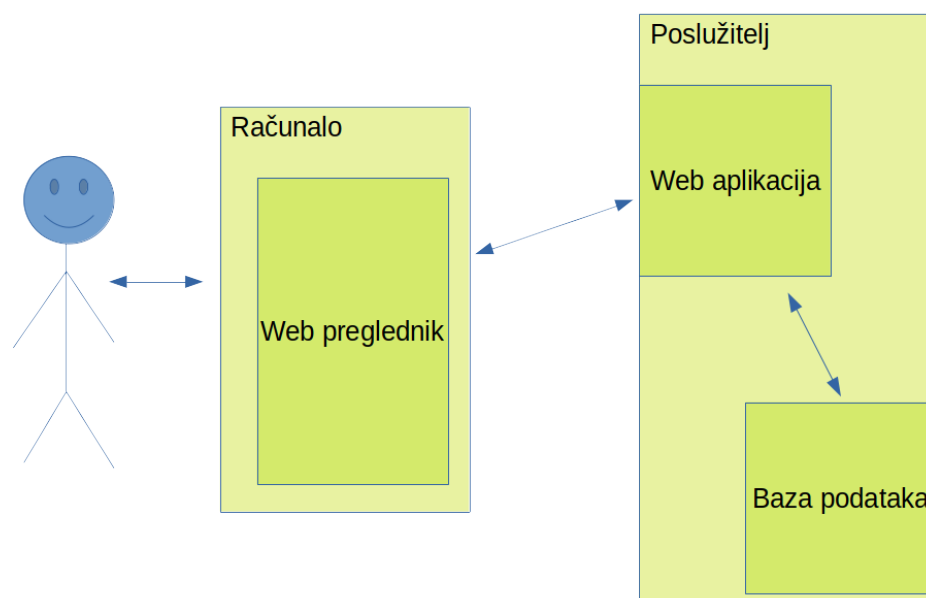
- Sustav mora podržavati rad za sve zaposlenike tvrtke istovremeno
- Korisničko sučelje i sustav moraju podržavati hrvatske znakove
- Upiti bazi podataka moraju čekati najviše 3 sekunde
- Sustav treba biti jednostavan i intuitivan za korištenje (KISS princip)
- Sustav mora poštovati načelo nadogradnje uz minimalnu promjenu
- Sustav za raspodjelu zadataka mora koristiti kanban ploču

- za komunikaciju između klijenta i sustava mora biti korišten HTTPS protokol
- veza s bazom podataka mora biti zaštićena, brza i otporna na vanjske greške
- podaci se moraju sanirati prije slanja bazi podataka

4. Arhitektura i dizajn sustava

Arhitekturu web aplikacije dijelimo na tri podsustava:

- Web aplikacija
- Web poslužitelj
- Baza podataka



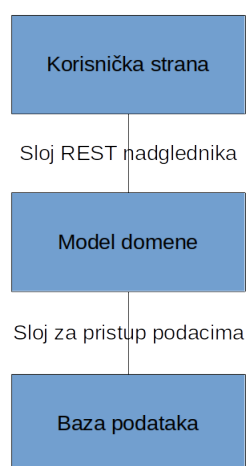
Slika 4.1: Arhitektura sustava

Web preglednik (eng. *web browser*) je korisnički program koji omogućuje pregledavanje statičkih i dinamičkih sadržaja interneta. Web preglednik dohvaća sadržaj s lokalnog ili udaljenog računala, i potom taj sadržaj interpretira i prikazuje korisniku. Neki od popularnijih web preglednika današnjice su Chrome, Safari, Firefox i Edge.

Web poslužitelj (eng. *web server*) je temelj aplikacije, a služi za komunikaciju. Korisnik i aplikacija razmjenjuju HTTP zahtjeve (eng. *HTTP request*) i HTTP odgovore (eng. *HTTP response*). Poslužitelj pokreće prednji kraj (eng. *front end*) i stražnji kraj (eng. *back end*). Radi jednostavnosti, baza podataka je također smještena na poslužitelju.

Korisnik kroz grafičko sučelje, odnosno prednji kraj, šalje zahtjeve na REST pristupne točke stražnjeg kraja. Tada stražnji kraj procesira zahtjev i ako je potrebno komunicira s bazom podataka. Nakon konstrukcije, stražnji kraj šalje odgovor prednjem kraju u obliku JSON objekta, a prednji kraj procesira odgovor i promjene prikazuje korisniku u obliku HTML stranice.

Za aplikaciju je odabrana višeslojna arhitektura temeljena na **MVC** (*model-view-controller*) arhitekturnom stilu te uslužnoj arhitekturi. Podjelu slojeva možemo napraviti na idući način:



Slika 4.2: Podjela slojeva

1. *sloj korisničke strane* - korisničko sučelje implementirano u JavaScriptu i radnom okviru AngularJS

2. *sloj nadglednika* - REST nadglednici
3. *sloj domene* - model podataka iz domene primjene
4. *sloj za pristup podacima* - posrednik između sloja domene i baze podataka
5. *sloj baze podataka* - pohrana podataka

Ovakva arhitektura odabrana je zbog poželjnih svojstava MVC arhitekturnog stila i višeslojne arhitekture: razvoj pojedinih slojeva jednostavniji je i u velikom stupnju nezavisan od razvoja drugih slojeva. Također, komunikacija prednjeg i stražnjeg kraja je ostvarena primjenom REST arhitekturnog stila. Zbog toga su prednji i stražnji kraj neovisni u smislu jezika implementacije, što potiče ponovnu uporabu.

MVC arhitekturni stil sastoji se od tri koncepta:

- **Model** - reprezentacija strukture podataka koja se koristi u rješenju, neovisna o korisničkom sučelju
- **View** - pogled na podatke, u našoj aplikaciji to je grafičko sučelje
- **Controller** - nadzornik koji koordinira zahtjeve i odgovore između modela i pogleda, sadrži svu logiku upravljanja.

REST (*representational state transfer*) arhitekturni stil kao upite koristi URI-je koji su mu poslani i tip HTTP zahtjeva, a odgovara JSON objektom. Tip HTTP zahtjeva koristi se za dohvat, ažuriranje ili umetanje podataka u bazu podataka, a dijelovi URI-ja se koriste za specificiranje zahtjeva. Tako bi na primjer REST API na GET upit usmjeren na putanju /employees odgovorio nizom koji sadrži sve zaposlenike, a GET upit na putanju /employees/user27 bi dobio odgovor JSON objekt koji predstavlja zaposlenika s korisničkim imenom user27.

4.1 Baza podataka

Odabrali smo PostgreSQL relacijsku bazu podataka za našu web aplikaciju zbog široke korisničke podrške, dokazane stabilnosti, dostupnosti sučelja s Javom, te prijašnjeg iskustva u radu s njom. Relacijska baza podataka omogućuje jednostavno modeliranje problema domene, a temeljna joj je zadaća sigurna i brza pohrana i dohvaćanje podataka. Temeljna građevna jedinica baze podataka je relacija

(tablica). Jedna relacija predstavlja jedan entitet, a naša baza sastoji se od idućih relacija:

- employees
- projects
- tasks
- teams
- workgroups

4.1.1 Opis tablica

employees - Ovaj entitet predstavlja zaposlenika. Sadrži attribute ID, korisničko ime, ime, prezime, lozinka, email, broj telefona, funkciju u firmi te ID tima kojemu pripada. U vezi je *One-to-Many* s entitetom tasks preko atributa ID i u vezi *Many-to-One* s entitetom teams preko atributa ID.

employees		
emp_id	BIGINT	jedinstveni identifikator zaposlenika
emp_name	VARCHAR	ime zaposlenika
emp_surname	VARCHAR	prezime zaposlenika
emp_username	VARCHAR	zaposlenikovo korisničko ime
emp_password	VARCHAR	zaposlenikova lozinka
emp_email	VARCHAR	zaposlenikov email
emp_phone	VARCHAR	zaposlenikov broj telefona
emp_clearance	INT	zaposlenikova funkcija - developer, team lead, koordinator ili uprava
team_id	BIGINT	identifikacijski broj tima kojemu zaposlenik pripada, (teams.team_id)

projects - Ovaj entitet predstavlja projekt. Sadrži attribute ID, opis, ime, te ID tima koji ga je preuzeo. U vezi je *One-to-One* s entitetom teams preko atributa ID tima koji ga je preuzeo.

projects		
project_id	BIGINT	jedinstveni identifikator projekta
project_desc	VARCHAR	opis projekta
project_name	VARCHAR	ime projekta
team_team_id	BIGINT	identifikator tima koji rješava projekt, (teams.team_id)

tasks - Ovaj entitet predstavlja zadatak na kanban ploči. Sadrži attribute ID, ime, opis, prioritet, status, ID zaposlenika koji ga je preuzeo, te ID tima kojemu pripada. U vezi je *Many-to-One* s entitetom teams preko atributa ID tima kojemu pripada. U istoj je vezi s entitetom employees preko atributa ID korisnika koji ga je preuzeo.

tasks		
task_id	INT	jedinstveni identifikator zadatka
task_deadline	TIMESTAMP	rok predaje zadatka
task_desc	VARCHAR	opis zadatka
task_name	VARCHAR	ime zadatka
task_prio	INT	prioritet zadatka
task_status	INT	status napretka zadatka
emp_id	BIGINT	identifikator zaposlenika koji rješava zadatak, (employee.emp_id)
team_id	BIGINT	identifikator tima kojemu ovaj zadatak pripada, (teams.team_id)

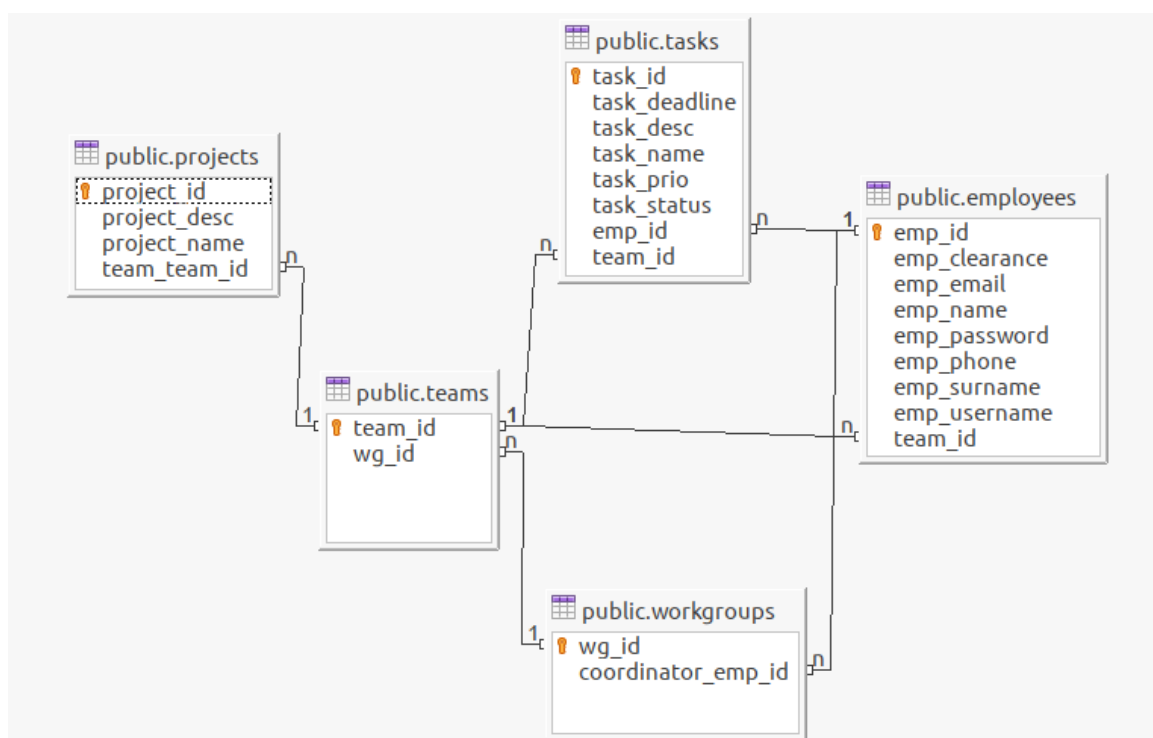
teams - Ovaj entitet predstavlja tim zaposlenika u firmi. Sadrži attribute ID tima te ID radne skupine kojoj tim pripada. U vezi je *Many-to-One* s entitetom workgroups preko atributa ID radne skupine kojoj tim pripada i u vezi *One-to-Many* s entitetom employees preko atributa ID. U istoj vezi je s entitetom tasks preko atributa ID. Također je u vezi *One-to-Many* s entitetom projects preko atributa ID.

teams		
team_id	BIGINT	jedinstveni identifikator tima
wg_id	BIGINT	identifikator radne skupine kojoj ovaj tim pripada, (workgroups.wg_id)

workgroups - Ovaj entitet predstavlja radnu skupinu. Radna skupina sastoji se od više timova i ima jednog koordinatora. Sadrži atribut ID radne skupine, te ID koordinatora tima. U vezi je *One-to-One* s entitetom employees preko atributa ID koordinatora tima, te je u vezi *One-To-Many* s entitetom teams preko atributa ID radne skupine.

workgroups		
wg_id	BIGINT	jedinstveni identifikator radne skupine
<i>coordinator_emp_id</i>	BIGINT	identifikator zaposlenika koji je koordinator ove radne skupine

4.1.2 Dijagram baze podataka

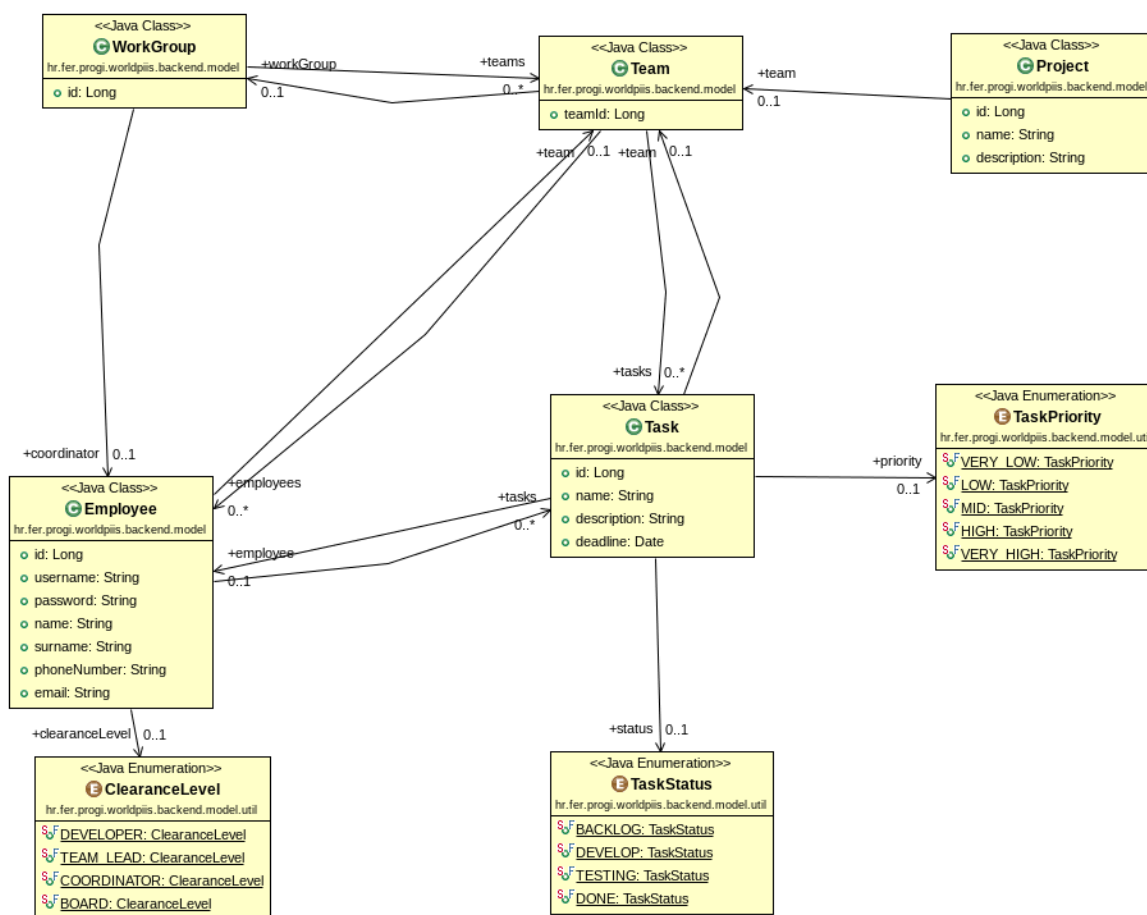


Slika 4.3: Dijagram baze podataka

4.2 Dijagram razreda

Dijagram razreda prikazuje odnose između različitih objekata, te njihove atribute i operacije kojima vladaju. Na slikama 4.3, 4.4 i 4.5 prikazani su razredi koji pripadaju *backend* dijelu naše arhitekture. Radi jednostavnosti, dijagram razreda je podijeljen u više slika, no bez obzira na to, prikazani razredi na neki način komuniciraju.

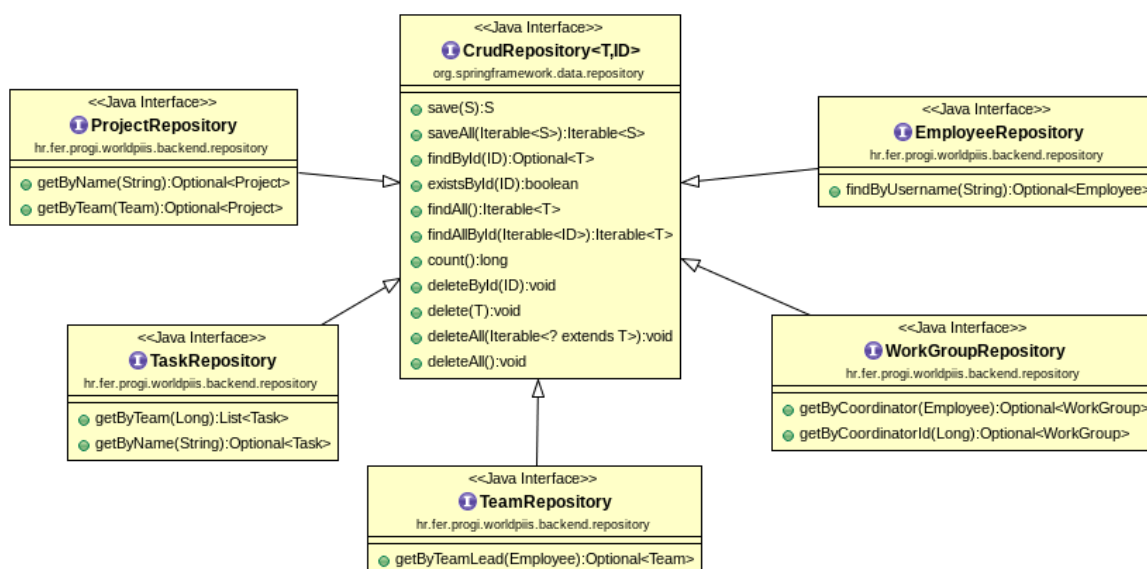
Na idućoj slici prikazan je model podataka kojim backend rukuje. Zaposlenik u firmi modeliran je razredom *Employee*. Razred *Team* modelira tim zaposlenika u firmi. Razred *Task* modelira jedan zadatak na kanban ploči. Razred *WorkGroup* modelira radnu skupinu u firmi. Razred *Project* modelira projekt na kojem tim radi.



Slika 4.4: Dijagram razreda koji opisuje model

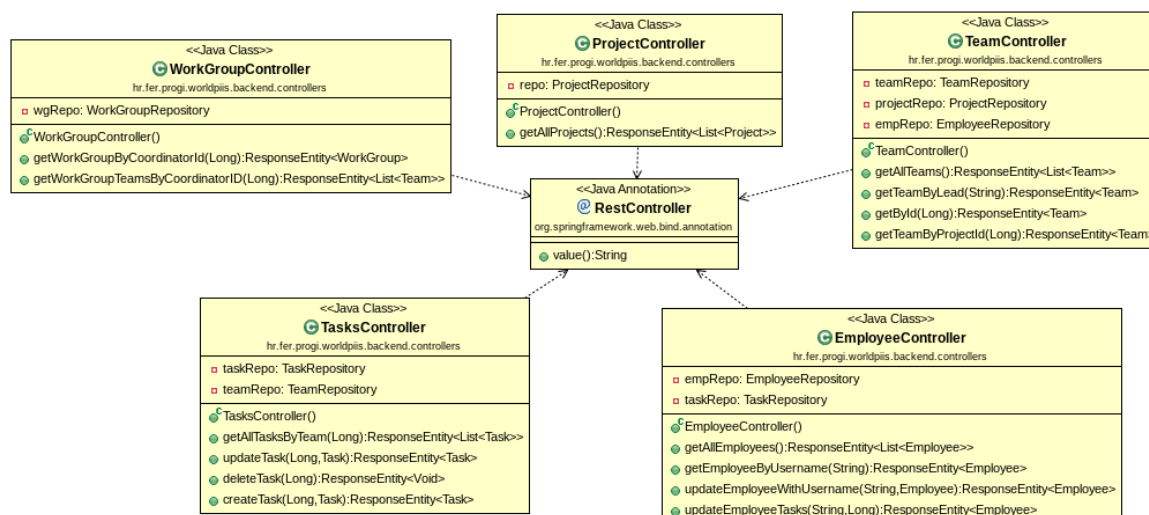
Na idućoj slici prikazana je sredina backenda. Glavni objekt ovdje je sučelje *CrudRepository*, koje predstavlja apstraktni repozitorij podataka. Iz tog sučelja,

izvedena su sučelja ProjectRepository, TaskRepository, TeamRepository, WorkGroupRepository i EmployeeRepository. Ta sučelja predstavljaju repozitorij podataka za prije navedene razrede modela, tj. oni predstavljaju poveznicu s bazom, ili DAO (eng. *Data Access Object*).



Slika 4.5: Dijagram razreda koji opisuje sredinu backenda

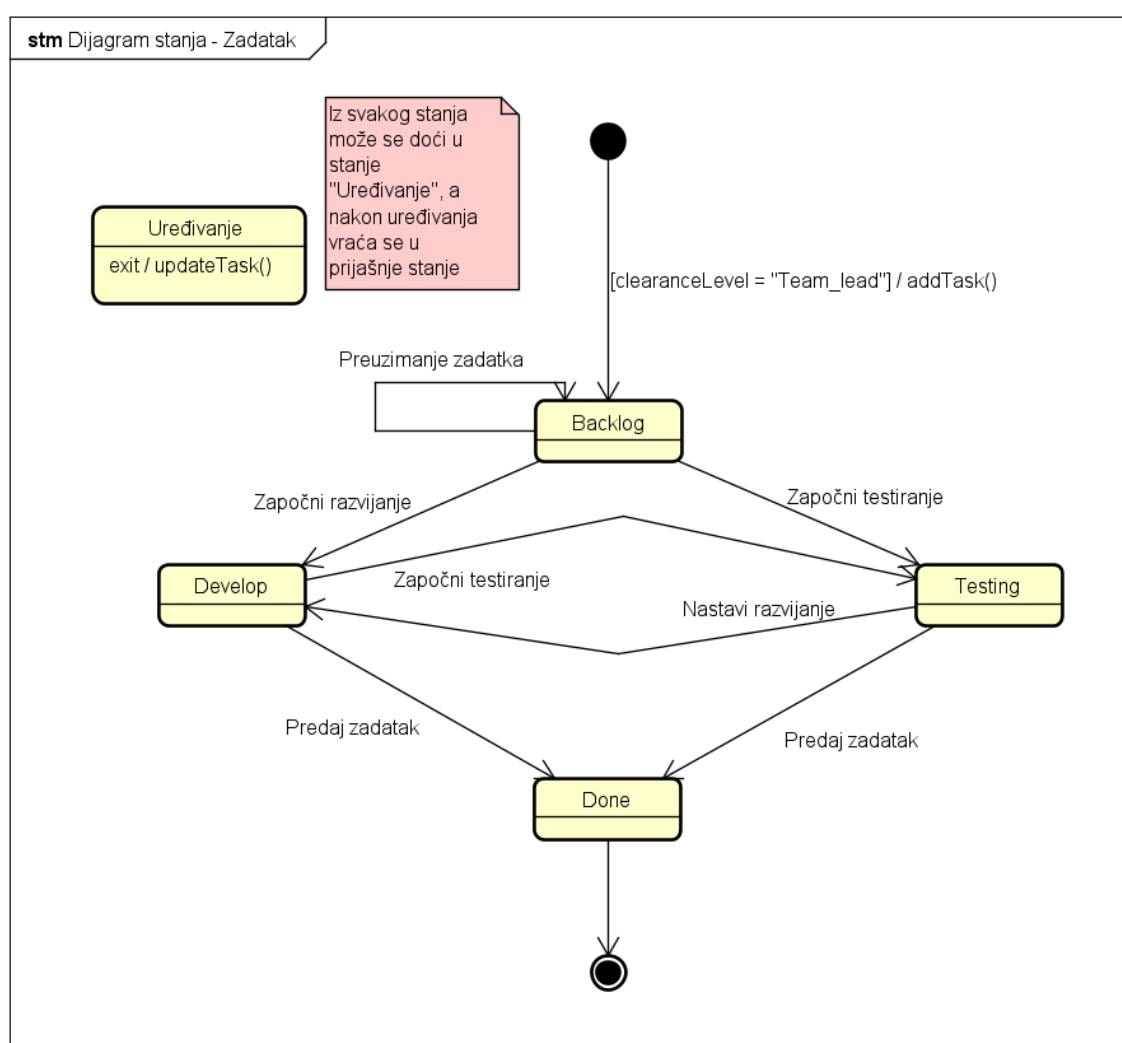
Na idućoj slici prikazan je "frontend backenda", odnosno sučelje backenda prema stvarnome svijetu. Ovdje vidimo razrede EmployeeController, TeamController, TaskController, ProjectController i WorkGroupController. Svi ti razredi implementiraju sučelje RestController koje predstavlja REST endpoint. Ti razredi su oni koji dobivaju zahtjeve iz vanjskog svijeta, a odgovaraju HTTP odgovorima i JSON objektima.



Slika 4.6: Dijagram razreda koji opisuje prednji dio backenda tj. kontrolere

4.3 Dijagram stanja

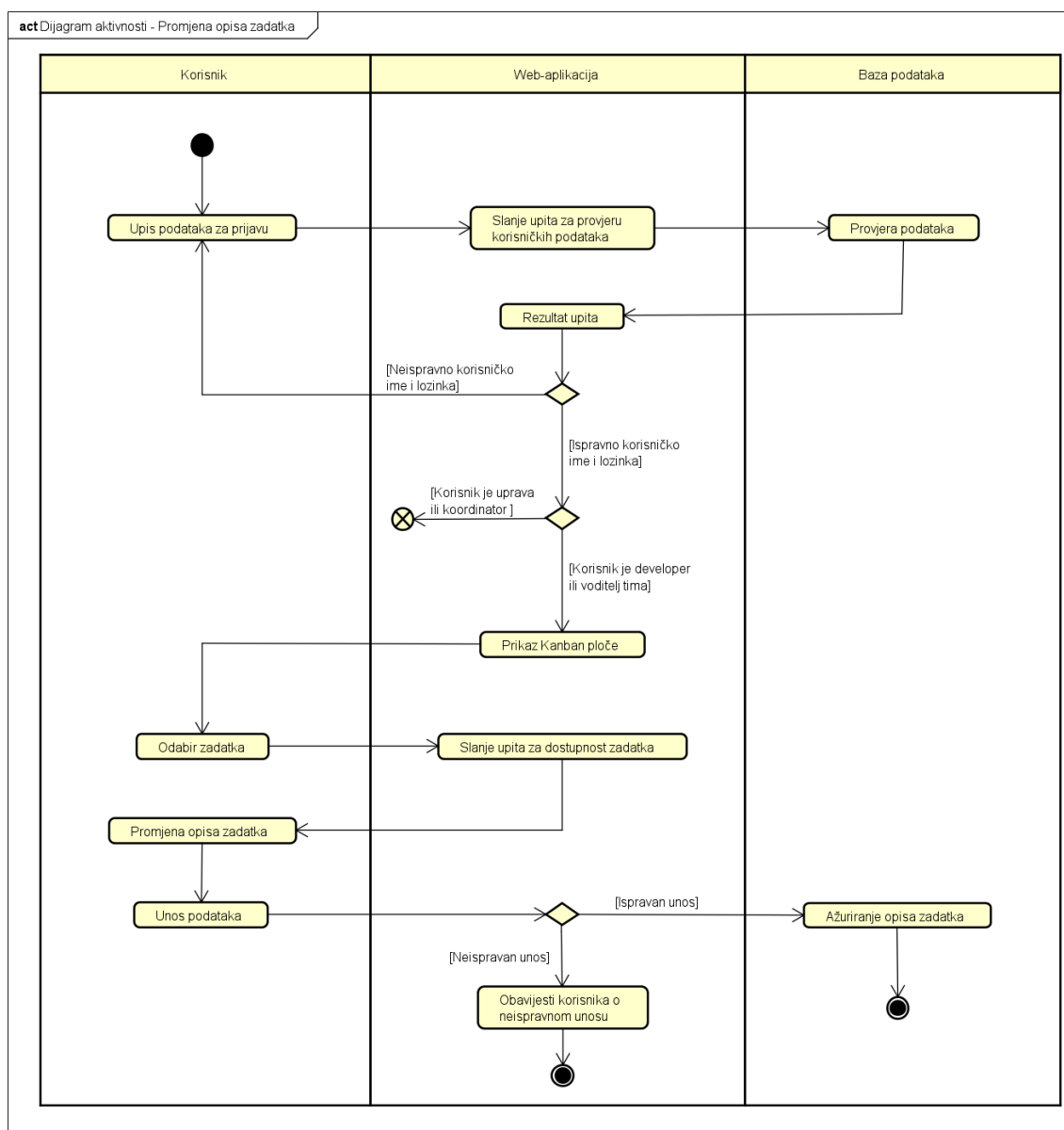
Dijagram stanja primjenjuje se za opis stanja objekta i za opisivanje prijelaza iz jednog u drugo stanje. Priložena slika prikazuje dijagram stanja objekta "Zadatak". Vođa tima kreira novi zadatak te nakon toga on prelazi u stanje "Backlog". Zadatak u tom stanju preuzima neki razvojni inženjer ili vođa tima zatim se vraća u to isto stanje. Zaposlenik može započeti razvoj ili testiranje zadataka koji time prelazi u odgovarajuće stanje. Zadatak može ući u stanje "Uređivanje" iz bilo kojeg drugog stanja. U završno stanje dolazi se nakon stanja "Done" što znači da je zadatak izvršen.



Slika 4.7: Dijagram stanja

4.4 Dijagram aktivnosti

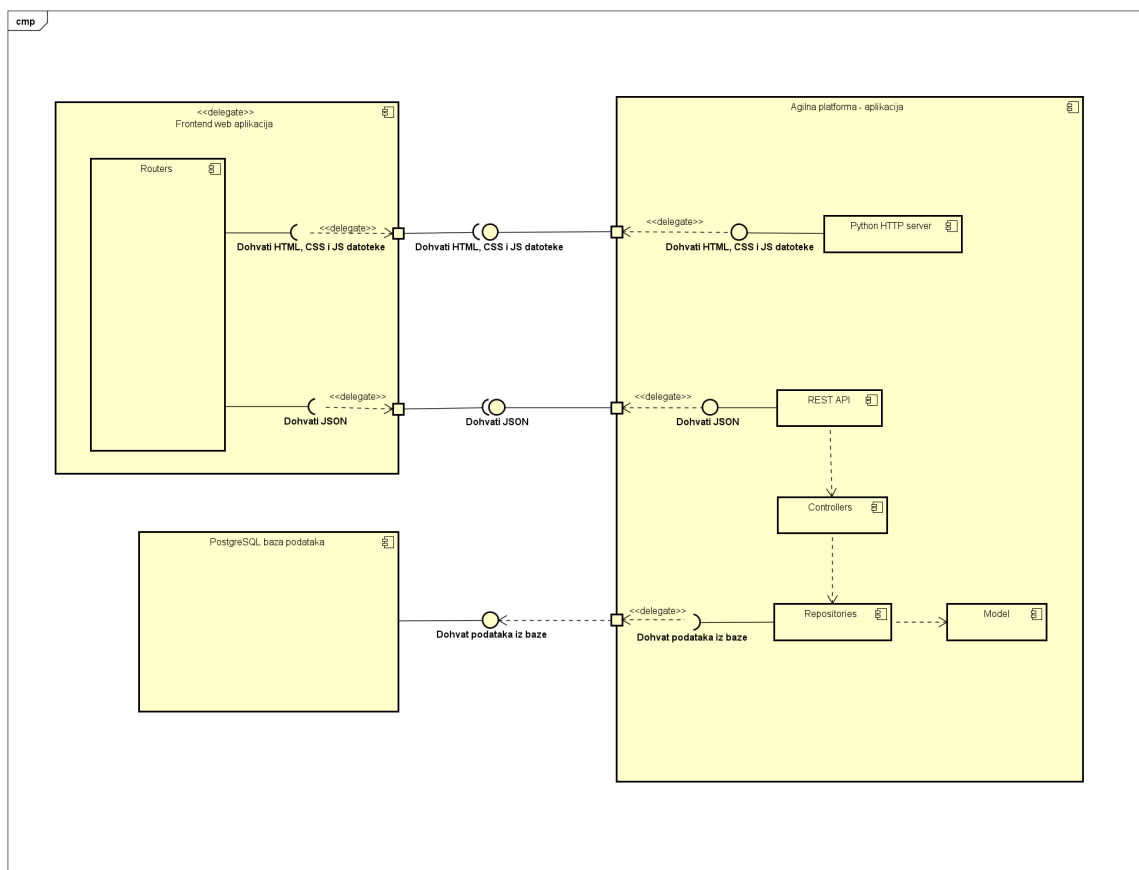
Dijagram aktivnosti prikazuje povezane aktivnosti na visokoj apstrakcijskoj razini. Dijagram aktivnosti intuitivno prikazuje kako podaci teku kroz aplikaciju i kako se kontrola nad podacima mijenja. Idući dijagram prikazuje proces preuzimanja zadatka i mijenjanja opisa zadatka. Ovaj proces može provesti neki razvojni inženjer ili voditelj tima.



Slika 4.8: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti omogućuje nam pogled na sustav s visoke razine apstrakcije. Web aplikacija komunicira sa sustavom preko dva sučelja. Prvo sučelje služi za dohvat samih stranica, dakle HTML, CSS i JS datoteka. Drugo sučelje služi za komunikaciju između aplikacije i baze podataka. Dakle, web aplikacija šalje zahtjev REST API-u, koji zauzvrat preko kontrolera komunicira s repozitorijima podataka, koji predstavljaju sloj povezanosti između baze podataka i kontrolera. Repozitoriji dohvaćaju podatke iz baze, preoblikuju ih na način kako je to zapisano u modelima, i proslijeđuju ih kontrolerima. Ovakav način obrade podataka karakterističan je za MVC obrazac.



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem platforme Microsoft Teams¹. Za izradu UML dijagrama korišten je alat Astah Professional², a kao sustav za upravljanje izvornim kodom Git³. Udaljeni repozitorij projekta je dostupan na web platformi GitLab⁴.

Kao razvojno okruženje korišten je IntelliJ IDEA⁵ - integrirano razvojno okruženje (IDE) tvrtke JetBrains. Prvenstveno se koristi za razvoj računalnih programa za operacijski sustav Windows, kao i za web-stranice, web-aplikacije, web-usluge i mobilne aplikacije.

Za razvoj korisničkog sučelja i frontenda korišten je uređivač teksta VSCode⁶. Taj uređivač teksta je danas jedan od najrasprostranjenijih i koristi se u svim sferama industrije.

Aplikacija je napisana u Javi⁷ koristeći ekosustav Java Spring⁸ za izradu backend-a te AngularJS⁹ i jezik JavaScript¹⁰ za izradu frontenda.

Baza podataka (PostgreSQL¹¹) se nalazi na poslužitelju u oblaku Digital Ocean¹².

¹<https://www.microsoft.com/hr-hr/microsoft-365/microsoft-teams/>

²<https://astah.net/products/astah-professional/>

³<https://git-scm.com/>

⁴<https://gitlab.com/>

⁵<https://www.jetbrains.com/idea/>

⁶<https://code.visualstudio.com/>

⁷<https://www.java.com/en/>

⁸<https://spring.io/>

⁹<https://angularjs.org/>

¹⁰<https://www.javascript.com/>

¹¹<https://www.postgresql.org/>

¹²<https://www.digitalocean.com/>

5.2 Ispitivanje programskog rješenja

Nakon što smo završili s izradom testirali smo rad aplikacije koristeći JUnit tehnologiju i Selenium WebDriver. Testovi su većinom dali zadovoljavajuće rezultate te možemo zaključiti da smo uspjeli implementirati zadane funkcionalnosti.

5.2.1 Ispitivanje komponenti

Za ispitivanje komponenti koristili smo JUnit tehnologiju. JUnit okvir je za testiranje komponenti sustava programiranog u Javi. Za simuliranje dohvaćanja podataka iz baze koristili smo okvir Mockito. Ispitali smo funkcionalnosti ažuriranja profila, ažuriranja zadatka i dodavanja novog zadatka. Koriste se objekti tipa `EmployeeController` i `TasksController` nad kojima se pozivaju ispitivane funkcionalnosti te objekti tipa `EmployeeRepository`, `TaskRepository` i `TeamRepository` koji su potrebni za dohvaćanje podataka korištenih u funkcijama.

Ažuriranje profila

Pozivom funkcije `updateEmployeeWithUsername` iz klase `EmployeeController` želimo zaposleniku uspješno promijeniti prezime. Na kraju uspoređujemo je li prezime ažuriranog zaposlenika jednako željenom novom prezimenu.

```
@Test
```

```
public void editProfileCorrectly() {
```

```
    Employee employee = new Employee("iva123", "Iva", "Ivić", "0910000000",  
    "iva@fer.hr", ClearanceLevel.DEVELOPER, null, null);
```

```
    Mockito.when(employeeRepository.findByUsername(  
    employee.getUsername())).thenReturn(Optional.of(employee));
```

```
    String newSurname="Ivančić";
```

```
    Employee employeeWithNewSurname = new Employee(employee.getUsername(),  
    employee.getName(), newSurname, employee.getPhoneNumber(), employee.getEmail(),  
    employee.getClearanceLevel(), employee.getTeam(), employee.getTasks());
```

```
    String[] expected = {newSurname};
```

```
String[] result = {employeeController.updateEmployeeWithUsername(  
    employee.getUsername(),employeeWithNewSurname).getBody().getSurname()};  
Assertions.assertEquals(expected,result);  
}
```

U idućem testu pozivom iste funkcije želimo neuspješno ažurirati korisnika pokušajem mijenjanja korisničkog imena, koje se se bi smjelo mijenjati. Na kraju uspoređujemo je li dobivena statusna poruka jednaka očekivanoj "404 NOT_FOUND" što bi značilo da aplikacija na pobuđenu situaciju prikladno odgovara.

```
@Test  
public void editProfileIncorrectly() {  
  
    Employee employee = new Employee("iva123","Iva","Ivić","0910000000",  
        "iva@fer.hr",ClearanceLevel.DEVELOPER,null,null);  
  
    String newUsername="iva1234";  
    Employee employeeWithNewUsername = new Employee(newUsername,  
        employee.getName(),employee.getSurname(),employee.getPhoneNumber(),  
        employee.getEmail(),employee.getClearanceLevel(),  
        employee.getTeam(),employee.getTasks());  
  
    String[] expected = {"404 NOT_FOUND"};  
    String[] result = {employeeController.updateEmployeeWithUsername(  
        employee.getUsername(),employeeWithNewUsername).getStatusCode().toString()};  
    Assertions.assertEquals(expected,result);  
}
```

Ažuriranje zadatka

Pozivom funkcije updateTask iz klase TasksController želimo zadatku uspješno promijeniti prioritet. Na kraju uspoređujemo je li prioritet ažuriranog zadatka jednak željenom novom prioritetu.

```
@Test
```

```
public void editTaskCorrectly(){

    Employee coordinator = new Employee("iva123","Iva","Ivić","0910000000",
    "iva@fer.hr",ClearanceLevel.COORDINATOR,null,null);
    WorkGroup workGroup = new WorkGroup(coordinator,null);
    Team team = new Team(null,workGroup,null);
    team.setTeamId(Long.valueOf(101));
    Task task = new Task("Testiranje sustava","Potrebno testirati sustav.",
    TaskPriority.HIGH,new Date(2020,1,2),TaskStatus.BACKLOG,null);
    task.id = Long.valueOf(22);

    Mockito.when(taskRepository.findById(
    task.getId())).thenReturn(Optional.of(task));

    TaskPriority newPriority = TaskPriority.VERY_HIGH;
    Task taskWithNewPriority = new Task(task.getName(), task.getDescription(),
    newPriority,task.getDeadline(),task.getStatus(),task.getEmployee());
    taskWithNewPriority.id = task.getId();

    TaskPriority[] expected = {newPriority};
    TaskPriority[] result = {tasksController.updateTask(task.getId(),
    taskWithNewPriority).getBody().getPriority()};
    Assertions.assertArrayEquals(expected,result);
}
```

Pozivom iste funkcije želimo neuspješno ažurirati zadatak pokušajem postavljanja praznog naziva zadatka, što ne bi smjelo prolaziti. Uspoređujemo je li dobivena statusna poruka jednaka očekivanoj "400 BAD_REQUEST" što bi značilo da aplikacija dobro reagira na naš loš zahtjev.

```
@Test
public void editTaskIncorrectly(){
```

```
    Team team = new Team(null,null,null);
    team.setTeamId(Long.valueOf(101));
```

```
Task task = new Task("Testiranje sustava", "Potrebno testirati sustav.",
TaskPriority.HIGH, new Date(2020, 1, 2), TaskStatus.BACKLOG, null);
task.id = Long.valueOf(22);

Mockito.when(taskRepository.findById(
task.getId())).thenReturn(Optional.of(task));

Task taskWithName = new Task(null, task.getDescription(),
task.getPriority(), task.getDeadline(), task.getStatus(), task.getEmployee());
taskWithName.id = task.getId();

String[] expected = {"400 BAD_REQUEST"};
String[] result = {tasksController.updateTask(task.getId(),
taskWithName).getStatusCode().toString()};
Assertions.assertArrayEquals(expected, result);
}
```

Dodavanje zadatka

Funkcijom `createTask` iz klase `TasksController` želimo uspješno dodati zadatak. Na kraju uspoređujemo jesu li atributi stvorenog zadatka jednaki željenim.

```
@Test
public void addTaskCorrectly(){

    Team team = new Team(null, null, null);
    team.setTeamId(Long.valueOf(101));
    Mockito.when(teamRepository.findById(
team.getTeamId())).thenReturn(Optional.of(team));

    Task task = new Task("Testiranje sustava", "Potrebno testirati sustav.",
TaskPriority.HIGH, new Date(2020, 1, 2), TaskStatus.BACKLOG, null);
    Mockito.when(taskRepository.save(any(Task.class))).thenReturn(task);
    Task createdTask = tasksController.createTask(team.getTeamId(), task).getBody();

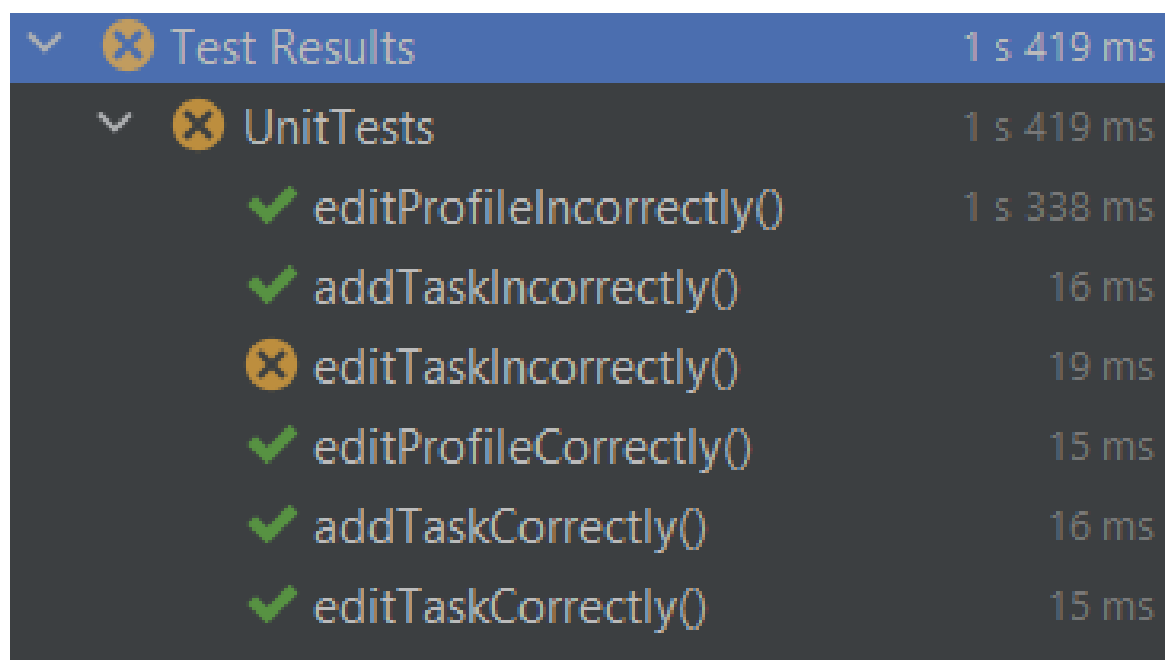
    String[] expected = {task.getName(), task.getDescription(),
```

```
task.getPriority().toString(),task.getDeadline().toString());  
String[] result = {createdTask.getName(),createdTask.getDescription(),  
createdTask.getPriority().toString(),createdTask.getDeadline().toString()};  
Assertions.assertArrayEquals(expected,result);  
}
```

Pozivom iste funkcije želimo neuspješno dodati bezimenu zadatak. Na kraju uspoređujemo je li stvoreni zadatak jednak null vrijednosti što bi značilo da aplikacija odbija stvoriti takav zadatak.

@Test

```
public void addTaskIncorrectly(){  
  
    Team team = new Team(null,null,null);  
    team.setTeamId(Long.valueOf(101));  
    Mockito.when(teamRepository.findById(  
        team.getTeamId())).thenReturn(Optional.empty());  
  
    Task taskWithoutName = new Task(null,"Potrebno testirati sustav.",  
        TaskPriority.HIGH,new Date(2020,1,2),TaskStatus.BACKLOG,null);  
  
    Task[] expected = {null};  
    Task[] result = {tasksController.createTask(team.getTeamId(),  
        taskWithoutName).getBody()};  
    Assertions.assertArrayEquals(expected,result);  
}
```



Test Results	1 s 419 ms
UnitTests	1 s 419 ms
editProfileIncorrectly()	1 s 338 ms
addTaskIncorrectly()	16 ms
editTaskIncorrectly()	19 ms
editProfileCorrectly()	15 ms
addTaskCorrectly()	16 ms
editTaskCorrectly()	15 ms

Slika 5.1: Rezultati ispitivanja komponenti

5.2.2 Ispitivanje sustava

Sustav smo ispitivali pomoću Selenium WebDriver okvira. Selenium WebDriver omogućuje ispitivanje sustava za koji je potrebna interakcija s internetskim preglednikom. Koristili smo Google Chrome preglednik. Ispitali smo funkcionalnosti uređivanja profila, uređivanja zadatka i dodavanja zadatka. Koriste se pomoćna metoda void login(WebDriver driver, String username, String password) za prijavu željenog korisnika u sustav i void logout(WebDriver driver) za odjavu korisnika. U main metodi definirane su potrebne vrijednosti za poziv svake funkcije.

Ažuriranje profila

Cilj prvog testa uspješno je ažuriranje korisnika. Funkcija editProfileCorrectly prvo prijavljuje željenog korisnika (korisničkog imena mhren), pritiskom na gumb otvara profil, odabire opciju za uređivanje, mijenja broj mobitela i mail te sprema promijenjene podatke. Test je uspješan ukoliko su broj mobitela i mail korisničkog profila postali jednaki željenima.

```
public static String editProfileCorrectly(WebDriver driver,String username,
String password,String newPhone,String newMail) {
```

```
login(driver,username,password);

driver.findElement(By.id("openProfile")).click();
driver.findElement(By.id("editProfile")).click();

WebElement element = driver.findElement(By.id("brMob"));
element.clear();
element.sendKeys(newPhone);

element = driver.findElement(By.id("eMail"));
element.clear();
element.sendKeys(newMail);

driver.findElement(By.id("spremi")).click();
boolean success=driver.findElement(By.id("mail")).getText().equals(newMail)&&
driver.findElement(By.id("mob")).getText().equals(newPhone);

return "editProfileCorrectly success: " + success;
}
```

Cilj drugog testa neuspješno je ažuriranje korisnika. Na početku funkcije editProfileIncorrectly već smo prijavljeni u sustav. Ponovno otvaramo profil i želimo ga urediti, ovaj put pokušavamo postaviti prazno korisničko ime, što se ne bi smjelo prihvatiti. Uspjehom smatramo ako u određenom vremenu iskoči prozor s upozorenjem koji označava da se podatci nisu promijenili. Ako se prozor pojavi na njemu pritišćemo "U redu". Odjavljujemo trenutnog korisnika zato što nam za idući test treba drugi korisnik.

```
public static String editProfileInorrectly(WebDriver driver){

    driver.findElement(By.id("editProfile")).click();
    WebElement element = driver.findElement(By.id("korIme"));
    element.clear();
    element.sendKeys(null);
    driver.findElement(By.id("spremi")).click();
```

```
boolean success;

try {
    WebDriverWait wait = new WebDriverWait(driver, 2);
    wait.until(ExpectedConditions.alertIsPresent());
    driver.switchTo().alert().accept();
    success = true;
}catch(TimeoutException ex){
    success = false;
}

driver.findElement(By.id("zatvori")).click();
logout(driver);
return "editProfileIncorrectly success: " + success;
}
```

Dodavanje zadatka

Želimo uspješno dodati zadatak. U fuknciji `addTaskCorrectly` prvo se prijavljujemo s korisničkim podacima koordinatora, pritišćemo gumb za dodavanje zadatka, dajemo mu ime, opis, prioritet i rok te ga spremamo. Uspješni smo ukoliko je broj zadataka na kanban ploči za jedan viši od broja zadataka na ploči prije pokušaja dodavanja zadatka.

```
public static String addTaskCorrectly(WebDriver driver,String username,
String password,String taskName,String description,
TaskPriority priority,String deadline){

    login(driver,username,password);

    int numberOfTasks = driver.findElements(By.id("task")).size();
    driver.findElement(By.id("addTask")).click();

    driver.findElement(By.id("taskName")).sendKeys(taskName);
    driver.findElement(By.id("description")).sendKeys(description);
```



```
driver.findElement(By.id("priority")).sendKeys(priority.toString());
driver.findElement(By.id("deadline")).sendKeys(deadline);
driver.findElement(By.id("add")).click();

boolean success=driver.findElements(By.id("task")).size()==numberOfTasks+1;
return "addTaskCorrectly success: " + success;
}
```

Želimo neuspješno dodati zadatak. U funkciji `addTaskIncorrectly` pritisćemo gumb za dodavanje zadatka, upisujemo naziv, opis te prioritet i rok neispravnih oblika. Spremamo podatke. Uspjehom smatramo ako u određenom vremenu iskoči prozor upozorenja koji označuje da zadatak nije dodan. Ako se prozor pojavi na njemu pritisćemo "U redu".

```
public static String addTaskIncorrectly(WebDriver driver,String taskName,
String description){

    driver.findElement(By.id("addTask")).click();

    driver.findElement(By.id("taskName")).sendKeys(taskName);
    driver.findElement(By.id("description")).sendKeys(description);
    driver.findElement(By.id("priority")).sendKeys("unknown");
    driver.findElement(By.id("deadline")).sendKeys("unknown");
    driver.findElement(By.id("add")).click();

    boolean success;

    try {
        WebDriverWait wait = new WebDriverWait(driver, 2);
        wait.until(ExpectedConditions.alertIsPresent());
        driver.switchTo().alert().accept();
        success = true;
    }catch(TimeoutException ex){
        success = false;
    }
}
```

```
    return "addTaskIncorrectly success: " + success;
}
```

Ažuriranje zadatka

Cilj ovog testa uspješno je uređivanje zadatka. Prije poziva funkcije `editTaskCorrectly` nalazimo se na kanban ploči iz prethodnog testa. Pronalazimo jedan gumb za uređivanje zadatka koji možemo pritisnuti (tj. jedan zadatak koji smo ranije preuzeli te ga imamo pravo urediti). Pritišćemo pronađeni gumb, zadatku mijenjamo opis i spremamo promjene. Uspjehom smatramo ako se nakon određenog vremena nije pojavio skočni prozor s upozorenjem koji bi signalizirao da zadatak nije uspješno promijenjen.

```
public static String editTaskCorrectly(WebDriver driver, String description){

    ArrayList<WebElement> editButtons=(ArrayList<WebElement>)driver.findElements(
        By.id("editTask"));
    for(WebElement editButton:editButtons){
        if(editButton.isEnabled()){
            editButton.click();
            break;
        }
    }

    WebElement element = driver.findElement(By.id("description"));
    element.clear();
    element.sendKeys(description);

    driver.findElement(By.id("saveTask")).click();

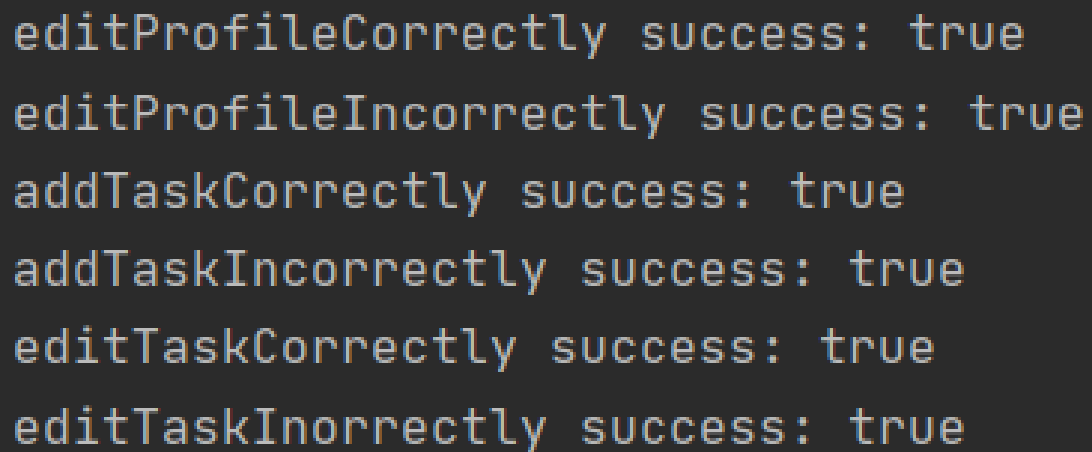
    boolean success;
    try {
        WebDriverWait wait = new WebDriverWait(driver, 2);
        wait.until(ExpectedConditions.alertIsPresent());
        driver.switchTo().alert().accept();
        success = false;
    }
```

```
}catch(TimeoutException ex){  
    success = true;  
}  
return "editTaskCorrectly success: " + success;  
}
```

Posljednjim testom želimo neuspješno ažurirati zadatak. U funkciji `editTaskIncorrectly` pritisćemo gumb za uređivanje zadatka, postavljamo prazni prioritet i spremamo promjene. Uspjehom smatramo ako se nakon nekog vremena pojavi skočni prozor koji označava da je nešto pošlo po krivu.

```
public static String editTaskIncorrectly(WebDriver driver){  
  
    ArrayList<WebElement> editButtons=  
        (ArrayList<WebElement>)driver.findElements(By.id("editTask"));  
    for(WebElement editButton:editButtons){  
        if(editButton.isEnabled()){  
            editButton.click();  
            break;  
        }  
    }  
  
    WebElement element = driver.findElement(By.id("priority"));  
    element.clear();  
  
    driver.findElement(By.id("saveTask")).click();  
  
    boolean success;  
    try {  
        WebDriverWait wait = new WebDriverWait(driver, 2);  
        wait.until(ExpectedConditions.alertIsPresent());  
        driver.switchTo().alert().accept();  
        success = true;  
    }catch(TimeoutException ex){  
        success = false;  
    }
```

```
}  
return "editTaskInorrectly success: " + success;  
}
```

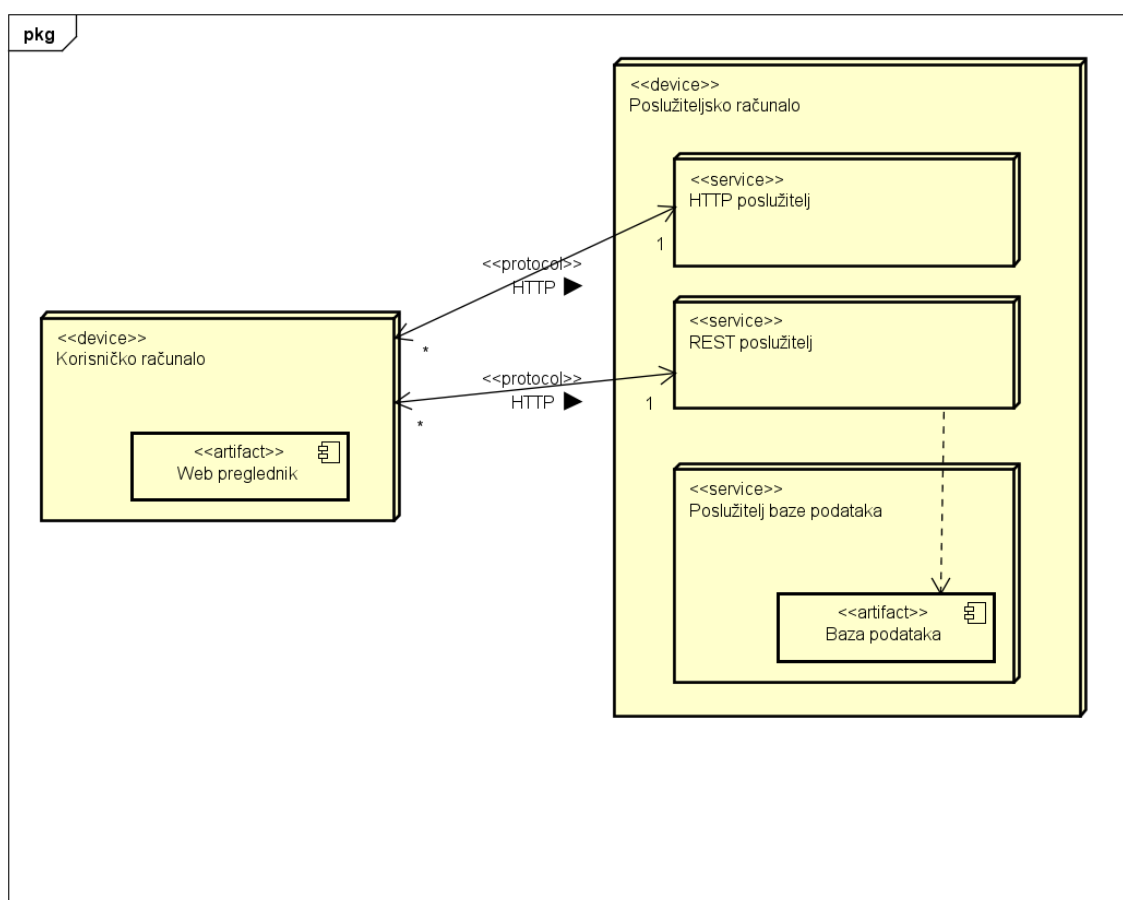


```
editProfileCorrectly success: true  
editProfileIncorrectly success: true  
addTaskCorrectly success: true  
addTaskIncorrectly success: true  
editTaskCorrectly success: true  
editTaskInorrectly success: true
```

Slika 5.2: Rezultati ispitivanja sustava

5.3 Dijagram razmještaja

Dijagrami razmještaja prikazuju topologiju sustava i odnos sklopovskih i programskih dijelova. Olakšavaju nam vizualizaciju razmještaja fizičkog dijela sustava i sklopovlja. Sustav se sastoji od korisničkog i poslužiteljskog računala. Korisnik na svojem računalu preko web preglednika pristupa aplikaciji. Korisničko računalo komunicira s poslužiteljskim računalom preko HTTP veze. Na poslužiteljskom se računalu nalaze HTTP poslužitelj, REST poslužitelj i poslužitelj baze podataka.



Slika 5.3: Dijagram razmještaja

5.4 Upute za puštanje u pogon

5.4.1 Operacijski sustav

Pretpostavlja se operacijski sustav iz Linux obitelji, na primjer Ubuntu 20.04¹³.

5.4.2 Preuzimanje programskog rješenja

Potrebno je instalirati paket `git` naredbom **`sudo apt install git`**. Zatim je potrebno pokrenuti naredbu **`git clone https://gitlab.com/henri_ohm/worldpiis.git`** kako bi se preuzelo programsko rješenje. Ovom naredbom će se stvoriti direktorij `worldpiis` (dalje `$PROJECT_ROOT`) koji sadrži sve potrebne datoteke za pokretanje aplikacije.

5.4.3 Baza podataka

Potrebno je preuzeti i instalirati najnoviju verziju paketa PostgreSQL. Pokretanjem naredbe **`sudo apt install postgresql-12`** potreban paket se preuzima i instalira, te se stvaraju daemon servisi koji pokreću bazu podataka prilikom pokretanja računala.

5.4.4 Konfiguracija poslužitelja baze podataka

Ovako instalirana baza podataka sluša zahtjeve sa adrese 127.0.0.1 na vratima 5432. Ovakva konfiguracija dovoljna je za potrebe ovog projekta. Također je potrebno osigurati da u bazi postoji korisnik **'postgres'** sa lozinkom **'postgres'**. Potrebno je osigurati i da na računalu postoji korisnik **'postgres'**, no to bi sama instalacija paketa već trebala osigurati.

5.4.5 Konfiguracija baze podataka

Sve potrebne tablice, ograničenja i veze će inicijalizirati REST poslužitelj kad se pokrene, pa je potrebno osigurati samo prije navedene uvjete kako bi baza radila.

¹³<https://ubuntu.com/>

5.4.6 Punjenje baze podataka

Za punjenje baze podataka dostupna je skripta na lokaciji `$PROJECT_ROOT/db/fill.sql`. Ova skripta može se pokrenuti tako da se prvo pokrene sučelje s bazom, naredbom **`sudo -i -u postgres psql`**, a zatim se pokrene naredba `\i db/fill.sql`. Ovdje se pretpostavlja da je korisnik postavljen u korijenskom direktoriju projekta.

5.4.7 Pokretanje baze podataka

Baza podataka pokreće se kada se računalo upali, no ukoliko postoje problemi, može se pokrenuti naredba **`sudo systemctl enable postgresql`**, koja će omogućiti pokretanje baze prilikom pokretanja računala, te naredba **`sudo systemctl start postgresql`** koja će pokrenuti bazu podataka jednom. Prva naredba osigurava pokretanje baze prilikom pokretanja računala.

5.4.8 Javna vrata servera

Potrebno je osigurati da su vrata 80 i vrata 8080 otvorena javnosti, što je moguće naredbom **`sudo ufw allow 80`** i **`sudo ufw allow 8080`**. Ovaj postupak omogućuje poslužitelju primanje zahtjeva na ta dva vrata. Vrata 80 su dobro poznata vrata za posluživanje HTTP zahtjeva, a na vrata 8080 dolaziti će zahtjevi na REST poslužitelj.

5.4.9 Pokretanje web poslužitelja

Prije svega, potrebno je osigurati prisutnost Python interpretatora, što je moguće naredbom **`sudo apt install python3`**. Nadalje, da bi se web poslužitelj pokrenuo, potrebno je pozicionirati se u direktorij `$PROJECT_ROOT/web` i pokrenuti naredbu **`sudo python3 -m http.server 80`**. Prije toga, potrebno je osigurati da su vrata 80 slobodna, odnosno da nijedan drugi proces ne sluša na tim vratima.

5.4.10 Konfiguracija veze s bazom podataka

U datoteci `$PROJECT_ROOT/src/main/resources/application.properties` postavljene su iduće konfiguracijske varijable:

- `spring.datasource.url = jdbc:postgresql://localhost:5432/postgres`
- `spring.datasource.username = postgres`

- `spring.datasource.password = postgres`

Prva varijabla govori poslužitelju na koji URI treba slati zahtjeve bazi podataka. Druga i treća varijabla govore s kojim korisničkim imenom i lozinkom se ti zahtjevi trebaju slati.

5.4.11 Pokretanje REST poslužitelja

Prije nego što se pokrene REST poslužitelj, potrebno je osigurati prisutnost Jave 13 i programskog paketa Apache Maven. U tu svrhu, potrebno je pokrenuti naredbu **`sudo apt install openjdk-13 maven`**, koja će instalirati oba paketa u jednom potezu. Nakon toga, potrebno je pozicionirati se u korijenski direktorij i pokrenuti naredbu **`mvn -N io.takari:maven:wrapper`**, koja će instalirati potreban paket za pokretanje Spring aplikacije. Konačno, u korijenskom direktoriju potrebno je pokrenuti poslužitelj naredbom **`./mvnw spring-boot:run`**. Ukoliko su potrebne debugging poruke, moguće je staviti zastavice `-e -X`.

5.4.12 Konfiguriranje adrese spajanja

Na početku datoteke `$PROJECT_ROOT/web/app.js` dostupna je varijabla **`REST_base`** koja govori na koju adresu i vrata frontend treba slati zahtjeve da bi dobio REST odgovore. Ova varijabla po potrebi se može mijenjati za lokalno testiranje ili kada se mijenja domena ili adresa aplikacije.

5.4.13 Domena aplikacije

Kako bi aplikaciji bilo lakše pristupati, potrebno je registrirati domenu na bilo kojem pružatelju domenskih imena. Za potrebe ovog projekta, korišten je **Duck-DNS**¹⁴.

5.4.14 Korištenje aplikacije

Nakon svih ovih koraka, aplikacija je upogonjena i dostupna na registriranoj domeni. Aplikacija ovog projekta dostupna je na domeni `worldpiis.duckdns.org`.

¹⁴<https://www.duckdns.org/>

6. Zaključak i budući rad

Naš zadatak bio je izraditi agilnu platformu s kanban pločom koju bi neka zamišljena tvrtka koristila.

Ideja kanban ploče za upravljanje projektima je sveprisutna u poslovnom svijetu, a dostupna je u alatima poput Trello¹ i Asana².

Projekt smo proveli u tri faze:

1. početna razrada funkcionalnih i nefunkcionalnih zahtjeva
2. implementacija zahtjeva i daljnje razrađivanje
3. testiranje i završno dokumentiranje

Na početku prve faze upoznawali smo ideju aplikacije, razrađivali smo funkcionalne zahtjeve i dogovarali se što sve aplikacija može i treba raditi. U retrospekciji, ova faza provedena je kaotično jer članovi tima još nisu bili svjesni težine projekta i količine posla, a da se provela kvalitetnije implementacija bi bila konceptualno lakša. U prvoj fazi smo se također počeli upoznavati s tehnologijama koje smo koristili. Ni jedan od članova tima nije bio u potpunosti upoznat s tim tehnologijama. Naučili smo da u stvarnosti, prije izrade samog projekta, potrebno je naučiti kako učiti alate, a zatim i naučiti sam alat.

U drugoj fazi projekta krenuli smo sa implementacijom, što je na početku teklo jako sporo jer još nismo bili upoznati s alatima, no kako je vrijeme prolazilo tako smo postajali bolji i implementirali smo zahtjeve sve brže. Također, u ovoj fazi uvidjeli smo propuste u definicijama funkcionalnih i nefunkcionalnih zahtjeva, pa smo naučili biti precizniji u svojim definicijama i uzimati više slučajeva u obzir.

U trećoj, odnosno posljednjoj fazi, nakon implementacije programskog rješenja, dovršili smo dokumentaciju projekta. Ovaj korak nam je dodatno pomogao u shvaćanju obujma našeg projekta, i ukazao nam na arhitekturne probleme koje nismo predvidjeli. Također, testiranjem smo ustanovili da sustav ima par grešaka, a najveća od tih je da REST API vraća prvi ugnježdjeni objekt u potpunosti, a sve ostale vraća samo preko njihovih identifikatora.

¹<https://trello.com/>

²<https://asana.com/>

Članovi tima su prije projekta bili upoznati s Javom i NodeJS-om, pa bi odabirom tih tehnologija umjesto AngularJS-a pomogao pri ubrzanju razvoja aplikacije.

Naučili smo važnost koordinacije i komunikacije među članovima tima. Nekad članovi nisu bili dobro koordinirani pa bi jedni implementirali funkcionalnost na ovaj, a drugi na onaj način. Naučili smo meke vještine povezane s projektima općenito, kao što je korištenje alata `git`, pisanje značajnih commit poruka, korištenje GitLab-a, te poštivanje unaprijed definiranih obrazaca programiranja.

Najviše od svega, naučili smo značenje vremenske koordiniranosti u grupnom radu. Projekti poput ovoga vremenski su zahtjevni i ponekad se može dogoditi da jedan modul programskog rješenja kasni za drugim, pa drugi ne može nastaviti svoj razvoj. U ovakvim slučajevima, iznimno je važno vremenski se uskladiti unutar tima kako bi se takve neučinkovitosti izbjegle.

Od definiranih funkcionalnih zahtjeva, uspjeli smo implementirati korisnički login, dodavanje i mijenjanje zadataka, preuzimanje zadataka, pregled kanban ploča timova od strane uprave i koordinatora, integraciju GoogleCalendar-a za sastanke i izmjenu profila zaposlenika.

Prije spomenuti bug primijetili smo tek na kraju izrade projekta i nismo imali dovoljno vremena kako bi ga popravili. Bug smo primijetili tako kasno jer smo do tada rukovali sa lijepim podacima i nismo imali priliku uvidjeti pogrešku. Iz ovoga smo naučili da temeljito testiranje sustava treba provoditi od samog početka, koliko je to god moguće.

U budućnosti bismo mogli popraviti taj bug, zatim mogli bismo omogućiti upravljanje sjednicom, čime bismo postigli mogućnost da više korisnika istovremeno koristi našu aplikaciju bez problema, i u više sjednica odjednom. Također, mogli bismo prolijeptšati korisničko sučelje, implementirati više statusa zadataka, proširiti profil korisnika da sadrži druge korisničke podatke, omogućiti upravi pregled profila i aktivnosti pojedinih zaposlenika, i slično.

Zaključno, zahvaljujući iskustvima i znanjima koja smo stekli na ovom projektu, kad bismo krenuli iz početka, napravili bismo mnogo bolji projekt mnogo brže.

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>
7. Java Spring dokumentacija, <https://spring.io/projects/spring-framework>
8. AngularJS dokumentacija, <https://docs.angularjs.org/api>

Indeks slika i dijagrama

3.1	Dijagram obrasca uporabe, funkcionalnosti uprave tvrtke i korisnika	18
3.2	Dijagram obrasca uporabe, funkcionalnosti razvojnog inženjera i vo- ditelja tima	19
3.3	Dijagram obrasca uporabe, funkcionalnosti koordinatora timova . .	20
3.4	Sekvencijski dijagram za UC1	21
3.5	Sekvencijski dijagram za UC5	22
3.6	Sekvencijski dijagram za UC11	23
3.7	Sekvencijski dijagram za UC18	25
4.1	Arhitektura sustava	27
4.2	Podjela slojeva	28
4.3	Dijagram baze podataka	32
4.4	Dijagram razreda koji opisuje model	33
4.5	Dijagram razreda koji opisuje sredinu backenda	34
4.6	Dijagram razreda koji opisuje prednji dio backenda tj. kontrolere . .	35
4.7	Dijagram stanja	36
4.8	Dijagram aktivnosti	37
4.9	Dijagram komponenti	38
5.1	Rezultati ispitivanja komponenti	45
5.2	Rezultati ispitivanja sustava	51
5.3	Dijagram razmještaja	52
6.1	Promjene na master grani	64
6.2	Aktivnosti članova tima	64
6.3	Aktivnosti članova tima	64

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 5. listopada 2020.
- Prisustvovali: Đokić, Fućec, Hren, Habuš, Jukić, Lukenda, Pavlic
- Teme sastanka:
 - sastanak s asistentom
 - analiza zadatka
 - osnovne dileme funkcionalnosti

2. sastanak

- Datum: 12. listopada 2020.
- Prisustvovali: Đokić, Habuš, Hren, Jukić, Pavlic
- Teme sastanka:
 - sastanak s asistentom
 - odabir alata i tehnologija

3. sastanak

- Datum: 14. listopada 2020.
- Prisustvovali: Đokić, Fućec, Habuš, Hren, Jukić, Lukenda, Pavlic
- Teme sastanka:
 - temeljitiji opis aplikacije
 - definiranje funkcionalnih zahtjeva i aktora
 - definiranje ostalih zahtjeva

4. sastanak

- Datum: 19. listopada 2020.
- Prisustvovali: Fućec, Habuš, Hren, Jukić, Pavlic
- Teme sastanka:
 - raspodjela poslova za razvoj aplikacije
 - idejna razrada arhitekture aplikacije

5. sastanak

- Datum: 26. listopada 2020.
- Prisustvovali: Đokić, Fućec, Habuš, Hren, Jukić
- Teme sastanka:
 - razrada baze s asistentom
 - razrada arhitekture backenda
 - definicija izgleda stranica

6. sastanak

- Datum: 2. studenog 2020.
- Prisustvovali: Fućec, Habuš, Hren, Jukić, Pavlic
- Teme sastanka:
 - REST kontroleri
 - dublja razrada frontenda
 - rasprava o idejama za v2.0

7. sastanak

- Datum: 9. studenog 2020.
- Prisustvovali: Đokić, Fućec, Habuš, Hren, Jukić, Lukenda, Pavlic
- Teme sastanka:
 - demonstracija aplikacije asistentu
 - dogovor i raspored zadataka za poliranje verzije 1.0 aplikacije
 - razumijevanje frontenda

8. sastanak

- Datum 7. prosinca 2020.
- Prisustvovali: Đokić, Fućec, Hren, Jukić, Pavlic
- Teme sastanka:
 - raspodjela poslova

9. sastanak

- Datum 30. prosinca 2020.
- Prisustvovali: Đokić, Habuš, Hren, Jukić
- Teme sastanka:
 - raspodjela poslova do α -inačice

10. sastanak

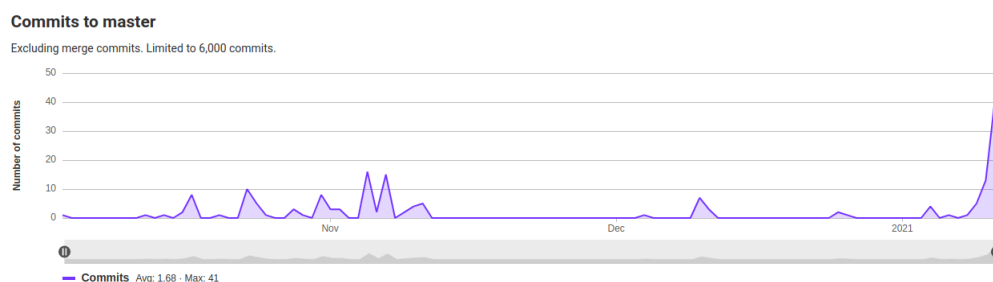
- Datum 8. siječnja 2021.
- Prisustvovali: Đokić, Fućec, Habuš, Hren, Jukić, Lukenda, Pavlic
- Teme sastanka:
 - demonstracija α -inačice

- raspodjela poslova do završetka projekta

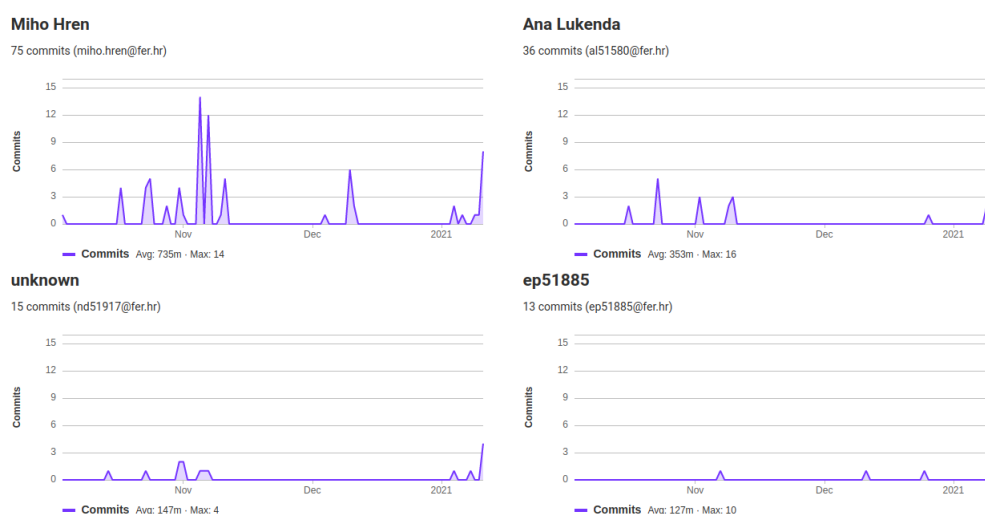
Tablica aktivnosti

	Miho Hren	Nikola Đokić	Borna Fučec	Luka Habuš	Ana-Petra Jukić	Ana Lukenda	Evelyn Pavlic
Upravljanje projektom	6						
Opis projektnog zadatka							4
Funkcionalni zahtjevi			2		2		
Opis pojedinih obrazaca						2	
Dijagram obrazaca						2	
Sekvencijski dijagrami		4					
Opis ostalih zahtjeva	1						
Arhitektura i dizajn sustava	3						
Baza podataka	5						
Dijagram razreda	2						
Dijagram stanja		3					
Dijagram aktivnosti				2			
Dijagram komponenti				2			
Korištene tehnologije i alati				1			
Ispitivanje programskog rješenja						6	
Dijagram razmještaja			2				
Upute za puštanje u pogon	2						
Dnevnik sastajanja	1						
Zaključak i budući rad				1			
Popis literature							
<i>Izrada početne stranice</i>		4	4	2	25	2	25
<i>Izrada baze podataka</i>	5						
<i>Spajanje s bazom podataka</i>	4	4	4			2	
<i>Back end</i>	15		3				2

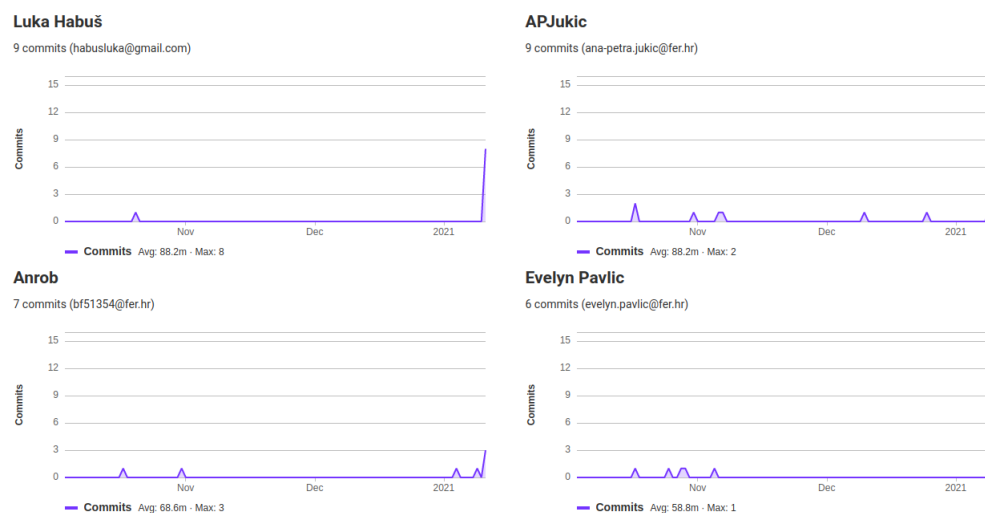
Dijagrami pregleda promjena



Slika 6.1: Promjene na master grani



Slika 6.2: Aktivnosti članova tima



Slika 6.3: Aktivnosti članova tima