

# Programsko inženjerstvo

Ak. god. 2021./2022.

## *Inventura*

Dokumentacija, Rev. 2

Grupa: DevOps404

Voditelj: Filip Bugarin

Datum predaje: 14. 1. 2022.

Nastavnik: Igor Stančin

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>5</b>
<b>3 Specifikacija programske potpore</b>	<b>8</b>
3.1 Funkcionalni zahtjevi . . . . .	8
3.1.1 Obrasci uporabe . . . . .	10
3.1.2 Sekvencijski dijagrami . . . . .	19
3.2 Ostali zahtjevi . . . . .	25
<b>4 Arhitektura i dizajn sustava</b>	<b>26</b>
4.1 Baza podataka . . . . .	26
4.1.1 Opis tablica . . . . .	27
4.1.2 Dijagram baze podataka . . . . .	32
4.2 Dijagram razreda . . . . .	33
4.3 Dijagram stanja . . . . .	36
4.4 Dijagram aktivnosti . . . . .	38
4.5 Dijagram komponenti . . . . .	40
<b>5 Implementacija i korisničko sučelje</b>	<b>41</b>
5.1 Korištene tehnologije i alati . . . . .	41
5.2 Ispitivanje programskog rješenja . . . . .	43
5.2.1 Ispitivanje komponenti . . . . .	43
5.2.2 Ispitivanje sustava . . . . .	47
5.3 Dijagram razmještaja . . . . .	49
5.4 Upute za puštanje u pogon . . . . .	50
5.4.1 Backend s bazom podataka . . . . .	50
5.4.2 Frontend - generiranje aplikacije . . . . .	53
<b>6 Zaključak i budući rad</b>	<b>56</b>
<b>Popis literature</b>	<b>58</b>

<b>Indeks slika i dijagrama</b>	<b>60</b>
<b>Dodatak: Prikaz aktivnosti grupe</b>	<b>61</b>

# 1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	D. Pipalović	13.10.2021.
0.2	Dopisan opis projektnog zadatka.	D. Pipalović	18.10.2021.
0.3	Dodan opis baze u latex oblik dokumentacije.	V. Rebernak	25.10.2021.
0.4	Dodano potpoglavlje funkcionalni zahtjevi.	D. Pipalović	30.10.2021.
0.5	Dodani zahtjevi specifikacije.	F. Bugarin	3.11.2021.
0.6	Dodan dijagram baze podataka u obliku slike.	V. Rebernak	4.11.2021.
0.7	Dodani dijagrami obrazaca uporabe, popunili tablicu aktivnosti i 4.sastanak.	F.Bugarin	6.11.2021.
0.8	Ispravljeni dijagrami obrazaca uporabe, dodani dijagrami razreda i sekvencijski dijagrami.	F.Bugarin, A.Gorup	13.11.2021.
0.9	Dodani opisi obrazaca uporabe.	Svi članovi tima	14.11.2021.
1.0	Ispravljene pogreške u dokumentaciji prije prve revizije.	V. Rebernak	15.11.2021.
1.1	Dokumentiran sastanak.	F. Bugarin	15.12.2021.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.2	Dokumentiran sastanak prije demonstracije alfa verzije.	F. Bugarin	19.12.2021.
1.3	Dodano potpoglavlje "korištene tehnologije i alati"	D. Pipalović	1.1.2022.
1.4	Dodano poglavlje "Zaključak"	D. Pipalović	7.1.2022.
1.5	Dodano poglavlje "Upute za puštanje u pogon"	D. Pipalović	9.1.2022.
1.6	Izmijenjen opis baze podataka da odgovara trenutnom stanju	D. Pipalović	11.1.2022.
1.7	Dodani dijagram razmještaja, dijagram stanja i dijagram aktivnosti	M. Hladek	11.1.2022.
1.8	Dodani testovi, osvježeni dijagrami razreda i relacijska shema baze podataka, update tablice sati	F.Bugarin	13.1.2022.
1.9	Spell check cijele dokumentacije	D. Pipalović	13.1.2022.
2.0	Konačni tekst predloška dokumentacije	*	13.1.2022.

## 2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku potporu za stvaranje mobilne aplikacije "Inventura." Mobilna aplikacija "Inventura" ima za cilj olakšati inventuru unutar skladišta uz pomoć pametnih telefona, osim samog olakšanja, cilj je povećati i efikasnost zaposlenika u skladištima te uvesti praćenje rada zaposlenika. Glavna funkcionalnost aplikacije je mogućnost očitavanja bar kodova i QR kodova artikala pomoću kamere mobitela.

Ciljana skupina korisnika aplikacije su veće i manje kompanije, trgovine te sva skladišta koja trebaju provoditi inventuru. Aplikacija pruža različite tipove korisničkih računa kao što su zaposlenik u skladištu, šef unutar skladišta te direktor kompanije. S obzirom na to da jedna kompanija može imati više od jednog skladišta na različitim lokacijama, aplikacija olakšava i proces nadgledanja zaposlenika u udaljenom skladištu omogućavanjem uvida u statistiku svakog pojedinca.



Slika 2.1: Ilustracija očitavanja QR koda

Ovisno o ulozi korisnika unutar kompanije postoje različite uloge i unutar aplikacije. Sve uloge dijele nekoliko zajedničkih funkcionalnosti. Tako je glavna funkcionalnost, skeniranje barkoda ili QR koda i izmjena stanja dostupna svim prijavljenim korisnicima aplikacije. Ova funkcionalnost radi tako da korisnik upali kameru unutar aplikacije te ako se u prostoru kamere nalazi QR ili barkod aplikacija će ga očitati i povezati s proizvodom. Od trenutka detekcije QR ili bar koda, aplikacija na zaslonu prikazuje zaslon s informacijama o očitanom proizvodu i lokaciji na kojoj je skeniran na 5 sekundi. Korisniku se tada nudi opcija da poveća ili

smanji broj takvih artikala u skladištu ili da odbaci očitani proizvod bez izmjena stanja (bez čekanja 5 sekundi). Promjenom broja očitanih artikala resetira se brojač na 5 sekundi. Istekom vremena brojača zapis se automatski pohranjuje u bazu.

Osim zajedničkih funkcionalnosti tako postoje i funkcionalnosti specifične za ulogu pojedinog korisnika:

- Zaposlenik u skladištu Osim zajedničkih funkcionalnosti zaposleniku u skladištu se omogućava uvid u pregled povijesti svojih izmjena kao i mogućnost dojava šefu skladišta da određenog artikla nema na skladištu
- Šef skladišta Osim zajedničkih funkcionalnosti šefu skladišta se omogućava uvid u sve artikle u njihovom skladištu; s obzirom na to da radnik u skladištu obavijest o tome da nekog artikla nema na skladištu šalje šefu skladišta, tako šef skladišta ima mogućnost prosljeđivanje dojava skladištara direktoru da nekog artikla nema u skladištu ako to smatra opravdanim. Osim toga, šef skladišta ima mogućnost kontrole rada skladištara tako da ako šef skladišta prijavi drugačije stanje nego skladištar automatski se generira obavijest direktoru kompanije.
- Direktor kompanije Osim zajedničkih funkcionalnosti direktoru se omogućava uvid u statistiku pogrešaka svakog skladištara. Kako šef ima mogućnost uvida u stanje artikla u svom skladištu, tako i direktor ima uvid u stanje artikala u svim skladištima. Direktoru se također omogućava kontrola pristiglih obavijesti. Pristiglu obavijest o niskom stanju direktor može ocijeniti kao valjanu ili ju može odbaciti. Pruža mu se i uvid u povijest svih obavijesti. Direktor je također zadužen za dodavanje skupina za proizvode pa tako i dodavanje samih proizvoda.

Svaki artikl nalazi se u jednoj skupini proizvoda. Skupine proizvoda su organizirane u stablo. Veličina stabla skupina nije ograničena. Primjer stabla je proizvod – prehrambeni proizvod – mliječni proizvod – jogurt – voćni jogurt, te se svi artikli koji su voćni jogurti svrstavaju u grupu voćni jogurt.

Iako ova aplikacija kao ciljanu skupinu ima mnogobrojne kompanije, inicijalno je zamišljena tako da svaka kompanija ima svoju verziju aplikacije, odnosno svoju bazu podataka i slično. Moguća nadogradnja ovog projektnog zadatka bila bi napraviti aplikaciju koja bi bila javno dostupna svima. Aplikaciju koja bi imala zajedničku bazu podataka. Odnosno aplikaciju koja bi omogućavala registraciju, pravljenje instance kompanije, uređivanje položaja unutar kompanije, dodavanje skladišta i razne druge modifikacije. Ova nadogradnja značajno bi povećala komplek-

snost samog projektnog zadatka, pristupačnost ljudima, no i troškove održavanja.

[illegible]

Slika 2.2: Provođenje inventure na papiru

Trenutno na tržištu ne postoji dostupna aplikacija sličnog tipa, već razna skladišta koriste papir i olovku za provođenje inventure. Poneke kompanije pak imaju sustave za inventuru za koje su platili dosta velike svote novca. S obzirom na navedeno, ova aplikacija bi bila inovacija na tržištu uz dobru pristupačnost.



## 3. Specifikacija programske potpore

### 3.1 Funkcionalni zahtjevi

#### Dionici:

1. Direktor (naručitelj)
2. Zaposlenici u skladištu
  - (a) Skladištar
  - (b) Šef skladišta
3. Razvojni tim

#### Aktori i njihovi funkcionalni zahtjevi:

1. Direktor (naručitelj) (inicijator) može:
  - (a) očitati proizvod
    - i. vidjeti podatke o proizvodu
    - ii. izmijeniti stanje na skladištu očitano proizvoda
  - (b) vidjeti sve artikle i njihovo stanje u skladištu
  - (c) kontrolirati rad skladištara
  - (d) vidjeti statistiku pogrešaka svakog skladištara
  - (e) kontrolirati pristigle obavijesti
  - (f) vidjeti povijest obavijesti
  - (g) dodavati skupine proizvoda
  - (h) uređivati artikle
  - (i) postavljati artikle u skupine
2. Zaposlenik (skladištar) (inicijator) može:
  - (a) očitati proizvod
    - i. vidjeti podatke o proizvodu
    - ii. izmijeniti stanje na skladištu očitano proizvoda
  - (b) dojaviti šefu skladišta da određenog artikla nema na skladištu

(c) pregledavati povijest svojih izmjena

3. Zaposlenik (šef skladišta) (inicijator) može:

(a) očitati proizvod

i. vidjeti podatke o proizvodu

ii. izmijeniti stanje na skladištu očitano proizvoda

(b) vidjeti sve artikle i njihovo stanje u skladištu

(c) kontrolirati rad skladištara

4. Baza podataka (sudionik):

(a) pohranjuje sve podatke o korisnicima i njihovim ulogama

(b) pohranjuje sve podatke o artiklima i skupinama kojima artikli pripadaju

(c) pohranjuje sve podatke o skladištima i artiklima koji se u njima nalaze

(d) pohranjuje sve podatke o obavijestima, pogreškama radnika, izmjenama stanja

### 3.1.1 Obrasci uporabe

#### Opis obrazaca uporabe

##### UC1 - Prijava u sustav

- **Glavni sudionik:** Korisnik
- **Cilj:** Prijaviti se u sustav
- **Sudionici:** Baza podataka
- **Preduvjet:** Osoba mora biti zaposlena u kompaniji, odnosno imati pravo pristupa aplikaciji
- **Opis osnovnog tijeka:**
  1. Otvaranje aplikacije na mobilnom uređaju
  2. Korisnik upisuje svoj nickname i password na za to predviđena mjesta
  3. Korisnik odabire opciju 'Log in'
  4. Ispravnost upisanih podataka provjerava se u bazi podataka
  5. Korisniku je dozvoljen pristup aplikaciji i učitava se početna stranica
- **Opis mogućih odstupanja:**
  - 5.a Korisniku nije dozvoljen pristup aplikaciji
    1. Sustav obavještava korisnika da upisani podaci nisu ispravni

##### UC2 - Skeniranje QR i bar koda

- **Glavni sudionik:** Korisnik
- **Cilj:** Skenirati QR ili bar kod
- **Sudionici:** Baza podataka, kamera na mobitelu
- **Preduvjet:** Korisnik je prijavljen u mobilnoj aplikaciji
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za skeniranje
  2. Kamera na mobitelu očitava kod
  3. Korisniku se otvara prozor s informacijama o proizvodu i mogućnosti izmjene količine
- **Opis mogućih odstupanja:**
  - 2.a Korisnik skenira nevažeći kod
    1. Sustav javlja da je nevažeći kod

##### UC3 - Izmjena stanja artikla u skladištu

- **Glavni sudionik:** Korisnik

- **Cilj:** Izmijeniti stanje artikla u skladištu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen, skeniran je QR kod
- **Opis osnovnog tijeka:**
  1. Korisnik ima mogućnost mijenjanja stanja artikla
  2. Ako korisnik ne napravi nikakvu izmjenu u aplikaciji u periodu od 5 sekundi prozor se zatvara
  3. Ako korisnik napravi izmjenu brojač od 5 sekundi se resetira
  4. Nakon izmjene stanja artikla korisnik sprema izmjenu

#### UC4 - Pristup povijesti svojih unosa

- **Glavni sudionik:** Skladištar
- **Cilj:** Pregledati povijest svojih prošlih unosa o izmjenama stanja artikala na skladištu
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljeni korisnik je skladištar
- **Opis osnovnog tijeka:**
  1. Skladištar odabire opciju za pregled povijesti o izmjenama stanja artikala
  2. Podatci se čitaju iz baze podataka
  3. Dohvaćeni podatci prikazuju se na ekranu mobilnog uređaja

#### UC5 - Pošalji dojavu o niskom stanju šefu skladišta

- **Glavni sudionik:** Skladištar
- **Cilj:** Poslati dojavu šefu skladišta o nedostatku određenog artikla na skladištu
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljeni korisnik je skladištar
- **Opis osnovnog tijeka:**
  1. Skladištar odabire opciju za slanje dojave o niskom stanju
  2. Obavijest se šalje šefu skladišta

#### UC6 - Pregled dojava o niskom stanju

- **Glavni sudionik:** Šef skladišta
- **Cilj:** Pregled i upravljanje dojavama o niskom stanju
- **Sudionici:** Baza podataka

- **Preduvjet:** Prijavljeni korisnik je šef skladišta
- **Opis osnovnog tijeka:**
  1. Šef skladišta odabire opciju za pregled dojava o niskom stanju
  2. Dojave o niskom stanju se učitaju iz baze podataka i pokažu na ekranu
  3. Šef skladišta upravlja dojavama tako da dojavu prosljeđuje ili odbacuje.
- **Opis mogućih odstupanja:**
  - 2.a Šef skladišta pritišće tipku odbaci umjesto proslijedi (ili suprotno)
    1. Aplikacija traži potvrdu učinjenog

#### UC7 - Uvid u sve artikle u skladištu

- **Glavni sudionik:** Šef skladišta
- **Cilj:** Pregled artikala u skladištu
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljeni korisnik je šef skladišta
- **Opis osnovnog tijeka:**
  1. Šef skladišta odabire opciju za uvid u artikle skladišta
  2. Podatci se čitaju iz baze podataka i prikazuju na ekranu

#### UC8 - Uvid u statistiku pogrešaka

- **Glavni sudionik:** Direktor
- **Cilj:** Uvid u statistiku pogrešaka svih skladištara
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljeni korisnik je direktor
- **Opis osnovnog tijeka:**
  1. Direktor odabire opciju za uvid u statistiku
  2. Iz baze se učitava lista svih skladištara i pokazuje na ekranu
  3. Direktor odabire skladištara
  4. Iz baze se učitava broj uspješnih i neuspješnih očitavanja za skladištara te se statistika pokaže na ekranu

#### UC9 - Uvid u stanje artikala u svim skladištima

- **Glavni sudionik:** Direktor
- **Cilj:** Pregled artikala u svim skladištima
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljeni korisnik je direktor
- **Opis osnovnog tijeka:**

1. Direktor odabire opciju za uvid u stanje artikala u skladištima
2. Iz baze se učitava lista skladišta i pokazuje na ekranu
3. Direktor odabire skladište
4. Iz baze se učitava list artikala u odabranom skladištu i pokazuje na ekranu

#### UC10 -Pristup obavijestima

- **Glavni sudionik:** Direktor
- **Cilj:** Pregledati sve obavijesti
- **Sudionici:** Baza podataka
- **Preduvjet:** Direktor je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Direktor odabire opciju koja vodi do prikaza obavijesti
  2. Aplikacija dohvaća obavijesti od baze podataka
  3. Obavijesti se prikazuju na ekranu mobilnog uređaja
  4. Direktor odabire jednu po jednu obavijest
  5. Direktor određuje je li obavijest riješena ili odbačena
  6. Direktorova odluka pohranjuje se u bazu podataka

#### UC11 - Dodavanje proizvoda

- **Glavni sudionik:** Direktor
- **Cilj:** Dodati određen proizvod
- **Sudionici:** Baza podataka
- **Preduvjet:** Direktor je prijavljen
- **Opis osnovnog tijeka:**
  1. Direktor odabire opciju dodavanja novog proizvoda
  2. Direktor unosi podatke za novi proizvod
  3. Podatci se šalju serveru
- **Opis mogućih odstupanja:**
  - 2.a Direktor doda novi proizvod ali ne pritisne opciju Spremi promjene
    1. Sustav upozorava direktora da nije spremio promjene

#### UC12 - Brisanje proizvoda

- **Glavni sudionik:** Direktor
- **Cilj:** Obrisati proizvod
- **Sudionici:** Baza podataka

- **Preduvjet:** Direktor je prijavljen
- **Opis osnovnog tijeka:**
  1. Direktor odabire opciju brisanja proizvoda
  2. Direktor odabire proizvod koji želi obrisati
  3. Podaci se šalju serveru
- **Opis mogućih odstupanja:**
  - 2.a Direktor obriše neželjeni proizvod ali ne pritisne opciju Spremi promjene
    1. Sustav upozorava direktora da nije spremio promjene

### UC13 - Uređivanje proizvoda

- **Glavni sudionik:** Direktor
- **Cilj:** Urediti određeni proizvod
- **Sudionici:** Baza podataka
- **Preduvjet:** Direktor je prijavljen
- **Opis osnovnog tijeka:**
  1. Direktor odabire opciju uređivanja proizvoda
  2. Direktor odabire proizvod koji želi urediti
  3. Direktor preuređuje proizvod
  4. Podaci se šalju serveru
- **Opis mogućih odstupanja:**
  - 2.a Direktor preuredi proizvod ali ne pritisne opciju Spremi promjene
    1. Sustav upozorava direktora da nije spremio promjene

### UC14 - Dodavanje grupe proizvoda

- **Glavni sudionik:** Direktor
- **Cilj:** Dodati određenu grupu proizvoda
- **Sudionici:** Baza podataka
- **Preduvjet:** Direktor je prijavljen
- **Opis osnovnog tijeka:**
  1. Direktor odabire opciju dodavanja nove grupe proizvoda
  2. Direktor unosi podatke za novu grupu
  3. Podaci se šalju serveru
- **Opis mogućih odstupanja:**
  - 2.a Direktor doda novu grupu ali ne pritisne opciju Spremi promjene
    1. Sustav upozorava direktora da nije spremio promjene

**UC15 -Brisanje grupe proizvoda**

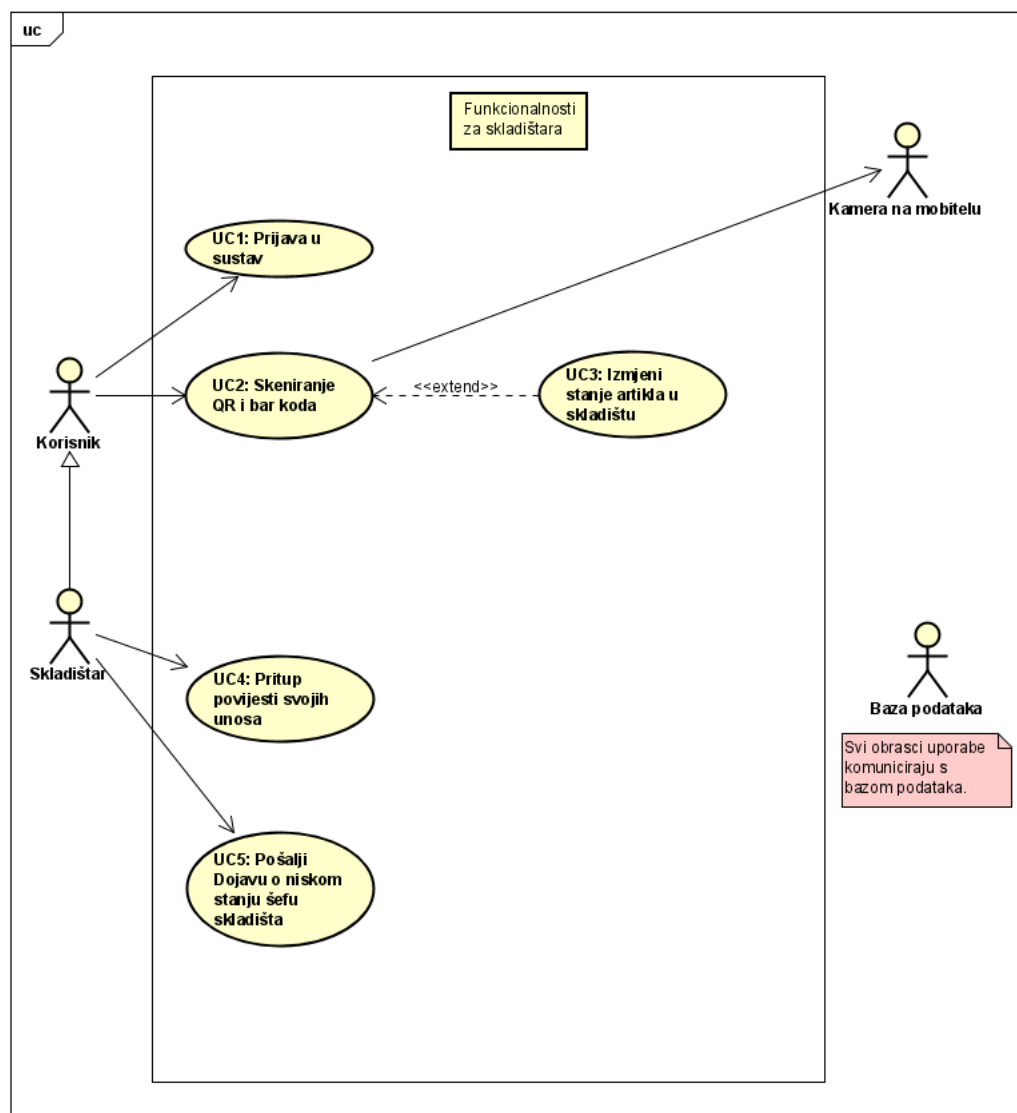
- **Glavni sudionik:** Direktor
- **Cilj:** Obrisati neželjenu grupu proizvoda
- **Sudionici:** Baza podataka
- **Preduvjet:** Direktor je prijavljen
- **Opis osnovnog tijeka:**
  1. Direktor odabire opciju brisanja grupa proizvoda
  2. Direktor odabire neželjenu grupu proizvoda
  3. Podaci se šalju serveru
- **Opis mogućih odstupanja:**
  - 2.a Direktor obriše neželjenu grupu ali ne pritisne opciju Spremi promjene
    1. Sustav upozorava direktora da nije spremio promjene

**UC16 - Uređivanje grupe proizvoda**

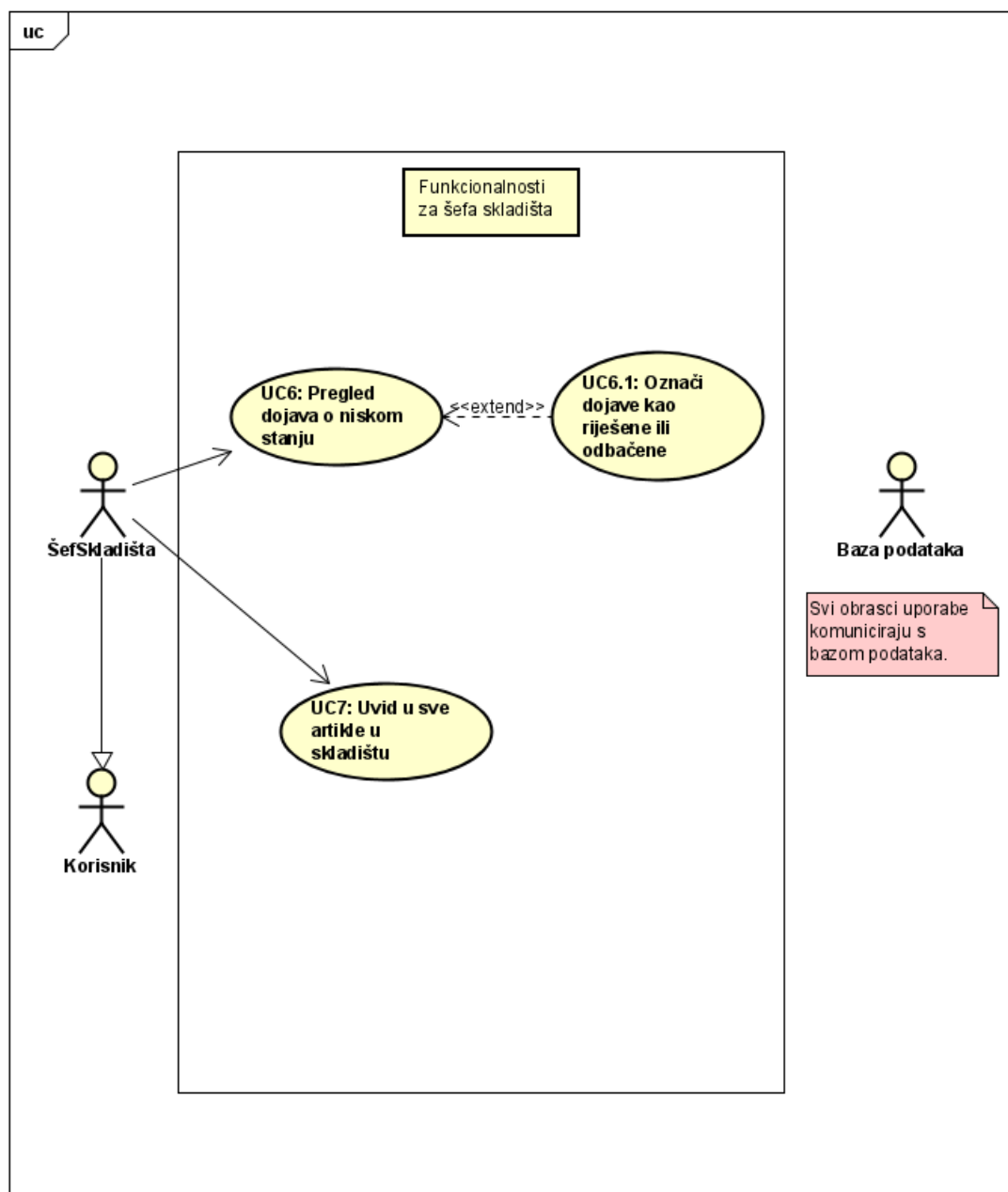
- **Glavni sudionik:** Direktor
- **Cilj:** Urediti određenu grupu proizvoda
- **Sudionici:** Baza podataka
- **Preduvjet:** Direktor je prijavljen
- **Opis osnovnog tijeka:**
  1. Direktor odabire opciju uređivanja grupe proizvoda
  2. Direktor odabire grupu proizvoda za uređivanje
  3. Direktor preuređuje grupu proizvoda
  4. Podaci se šalju serveru
- **Opis mogućih odstupanja:**
  - 2.a Direktor preuredi grupu ali ne pritisne opciju Spremi promjene
    1. Sustav upozorava direktora da nije spremio promjene



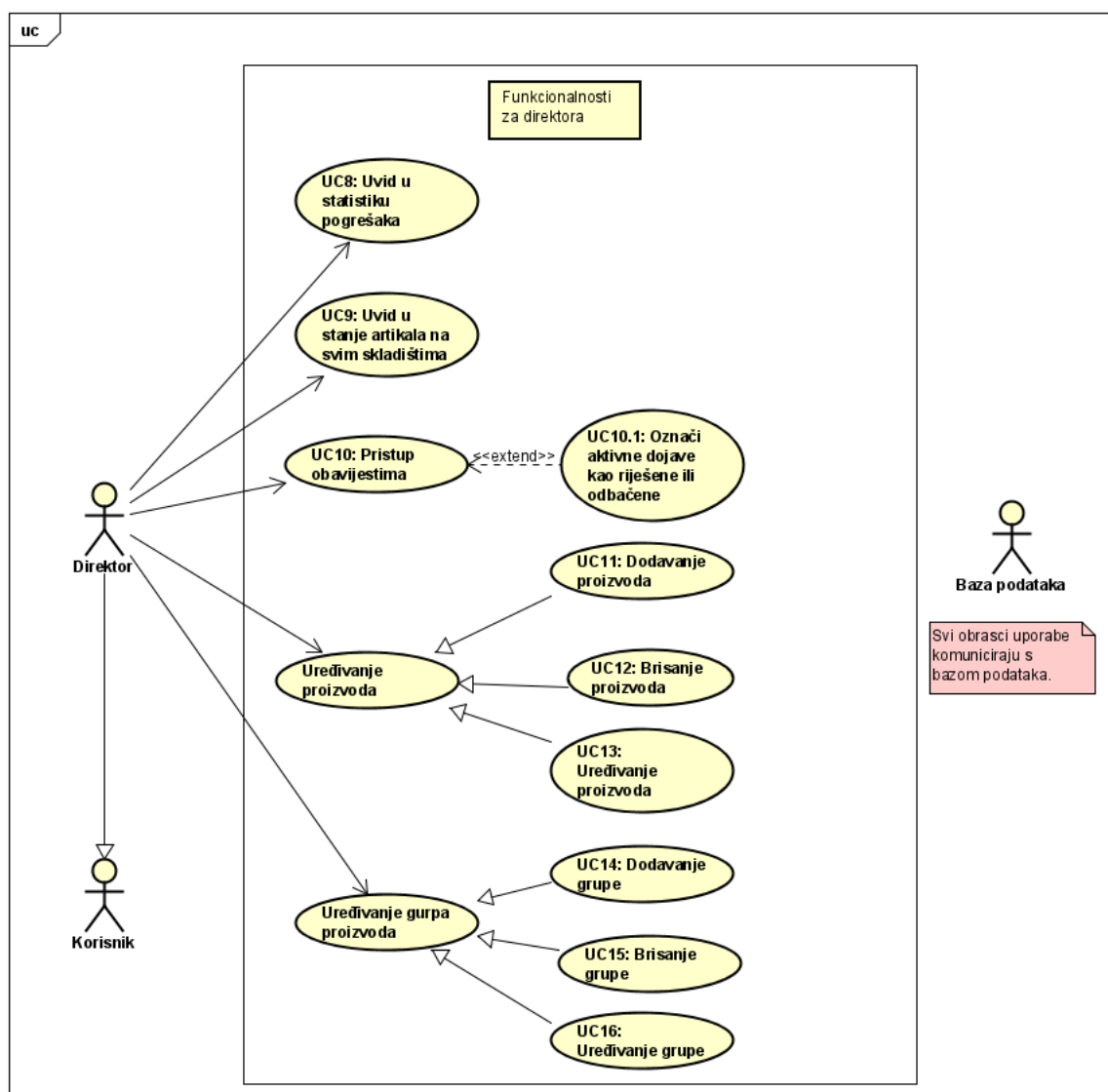
## Dijagrami obrazaca uporabe



Slika 3.1: Use case za korisnika i skladištara



Slika 3.2: Use case za šefa skladišta

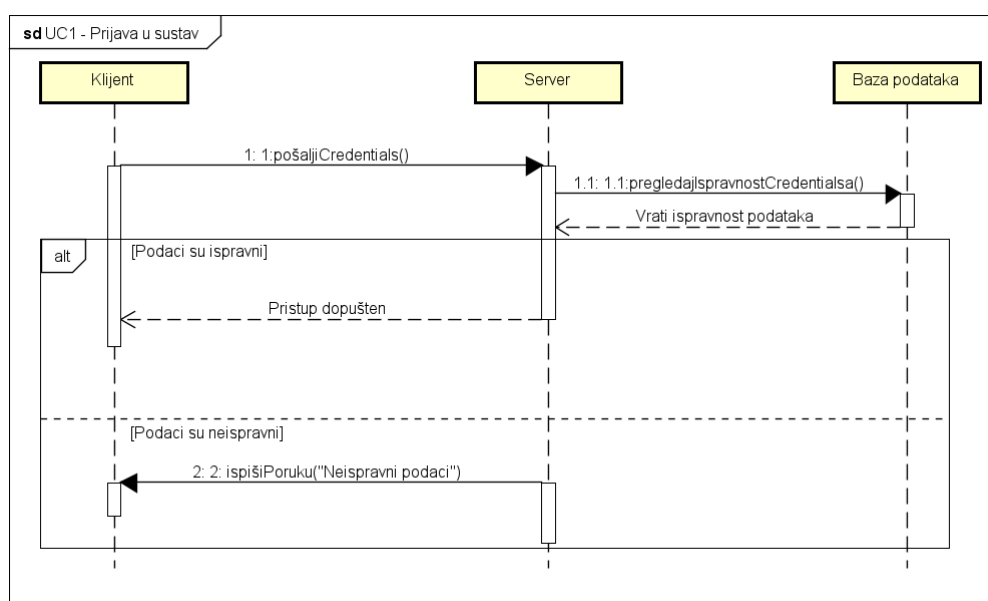


Slika 3.3: Use case za direktora

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe UC1 - Prijava u sustav

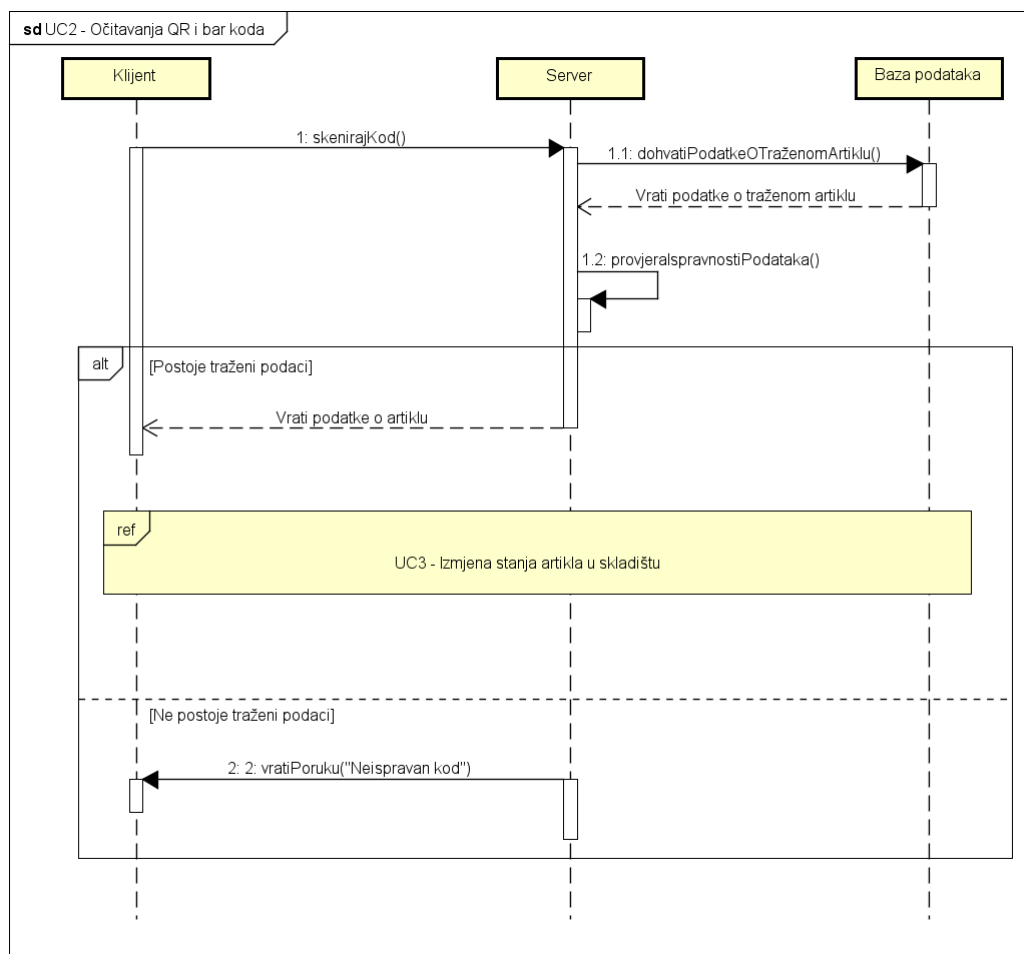
Korisnik unosi svoje korisničko ime i lozinku te upisane podatke predaje serveru na validaciju. Ako su upisani podaci ispravni, server mu vraća token koji će Korisnik koristiti za daljnju validaciju te mu je dopušten pristup aplikaciji. Uz token, vraća se i korisnikova uloga. Uloga može biti: Skladištar, Šef skladišta i Direktor. Ovisno o vraćenoj ulozi, korisniku se nadalje nude drukčije opcije. Ako su podaci koje je korisnik upisao neispravni, server će mu vratiti poruku o neispravnim podacima te korisniku nije dopuštena prijava u aplikaciju.



Slika 3.4: Sekvencijski dijagram za prijavu u sustav

**Obrazac uporabe UC2 - Skeniranje QR ili bar koda**

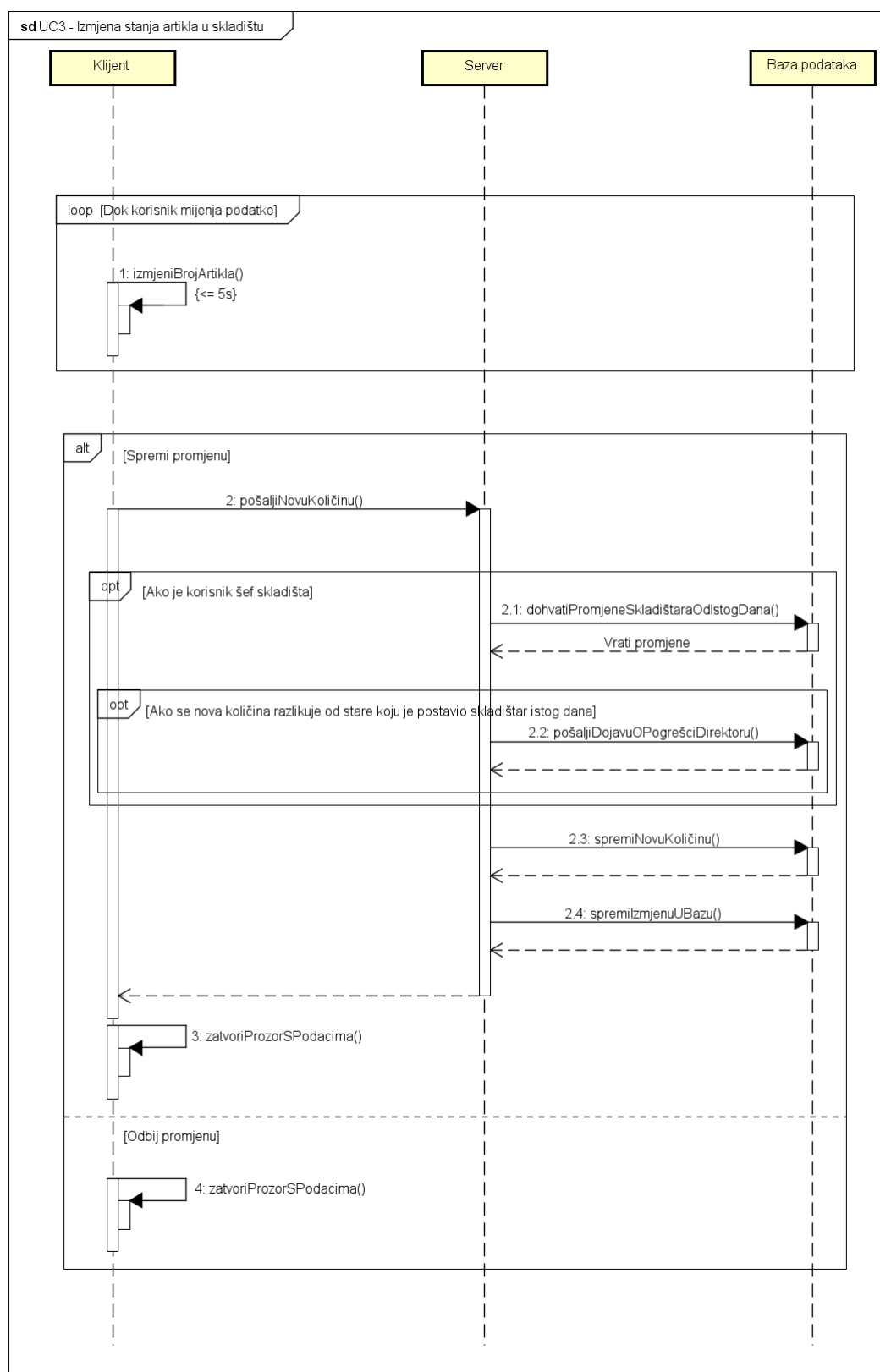
Korisnik odabire opciju za skeniranje koda te mu se potom otvara prozor za skeniranje i aktivira kamera na mobitelu. Korisnik potom može skenirati QR ili bar kod na određenom artiklu. Povodom skeniranja koda, server iz baze podataka dohvaća podatke o traženom artiklu te se korisniku na zaslonu ispišu traženi podaci i lokacija na kojoj je artikl skeniran. Korisnik potom može mijenjati broj traženog artikla u trenutnom skladištu, što je detaljnije opisano u idućem obrascu uporabe.



Slika 3.5: Sekvencijski dijagram za očitavanje QR koda ili bar koda

**Obrazac uporabe UC3 - Izmjena stanja artikla u skladištu**

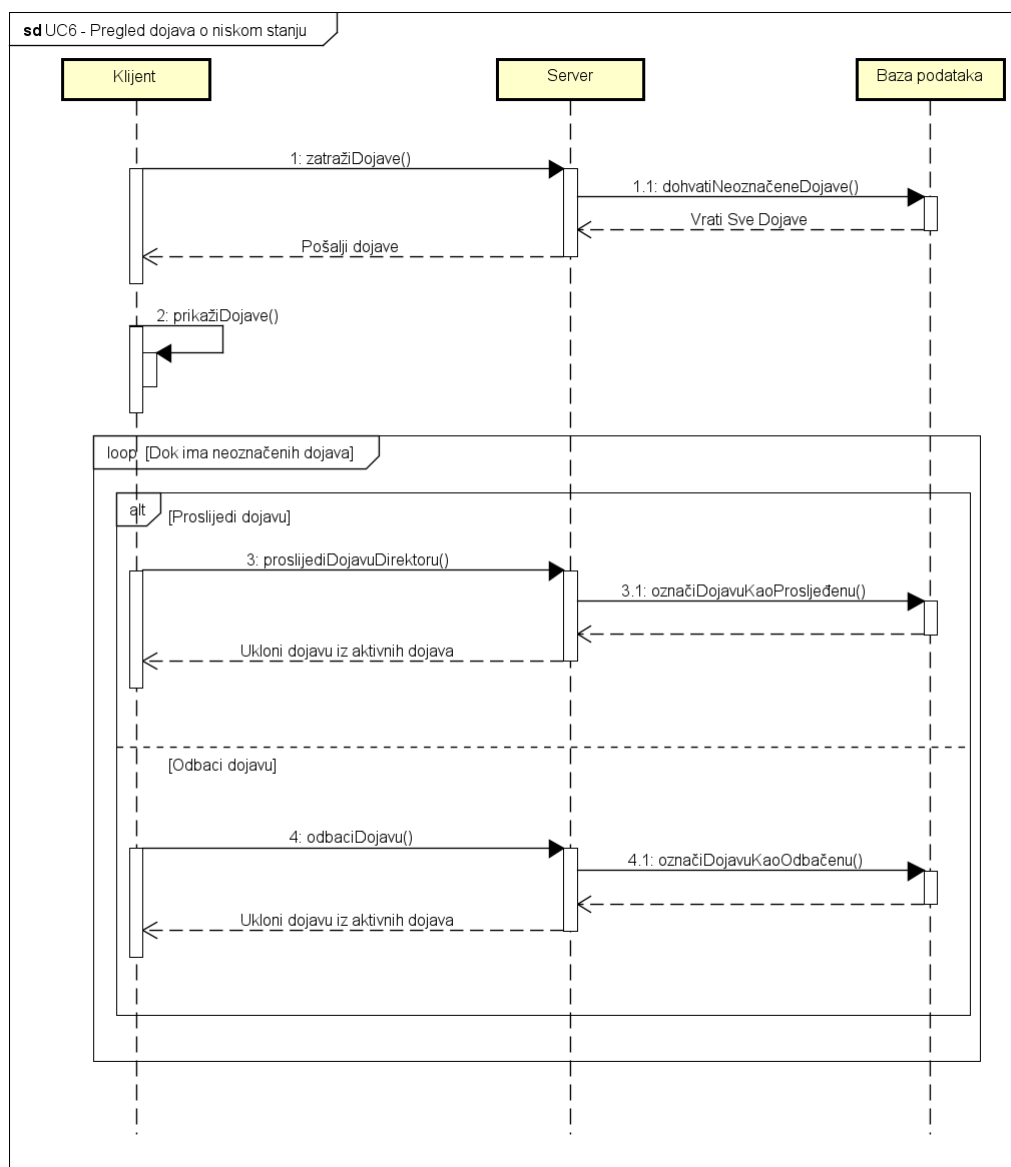
Korisnik može, unutar 5 sekundi, mijenjati broj traženog artikla u trenutnom skladištu. Ako korisnik promjeni broj artikla, brojač vremena se resetira na 5 sekundi. Korisnik također ima opciju pohraniti podatke prije isteka navedenih 5 sekundi ili odbaciti bilo kakve promjene. Ako korisnik nije odbacio promjene, nova navedena količina se sprema u bazu, kao i zapis o novoj izmjeni. Ako je korisnik koji unosi promjene prijavljen kao šef skladišta, na serveru se vrši dodatna provjera ispravnosti podataka. Ako je posljednje skeniranje istog artikla koje je proveo skladištar tijekom iste inventure upisalo različitu količinu od količine koju je upisao šef skladišta, direktoru se automatski šalje obavijest o pogrešnom skeniranju.



Slika 3.6: Sekvencijski dijagram za izmjenu stanja artikla u skladištu

**Obrazac uporabe UC5 - Pregled dojava o niskom stanju**

Šef skladišta odabire opciju za prikaz aktivnih dojava o niskom stanju artikala u skladištu te mu se dojave ispišu na zaslону. Korisnik potom može dojave odbaciti ili proslijediti direktoru. Ako korisnik označi dojavu kao odbačenu ili proslijeđenu, ta dojava mu se miče iz aktivnih dojava.

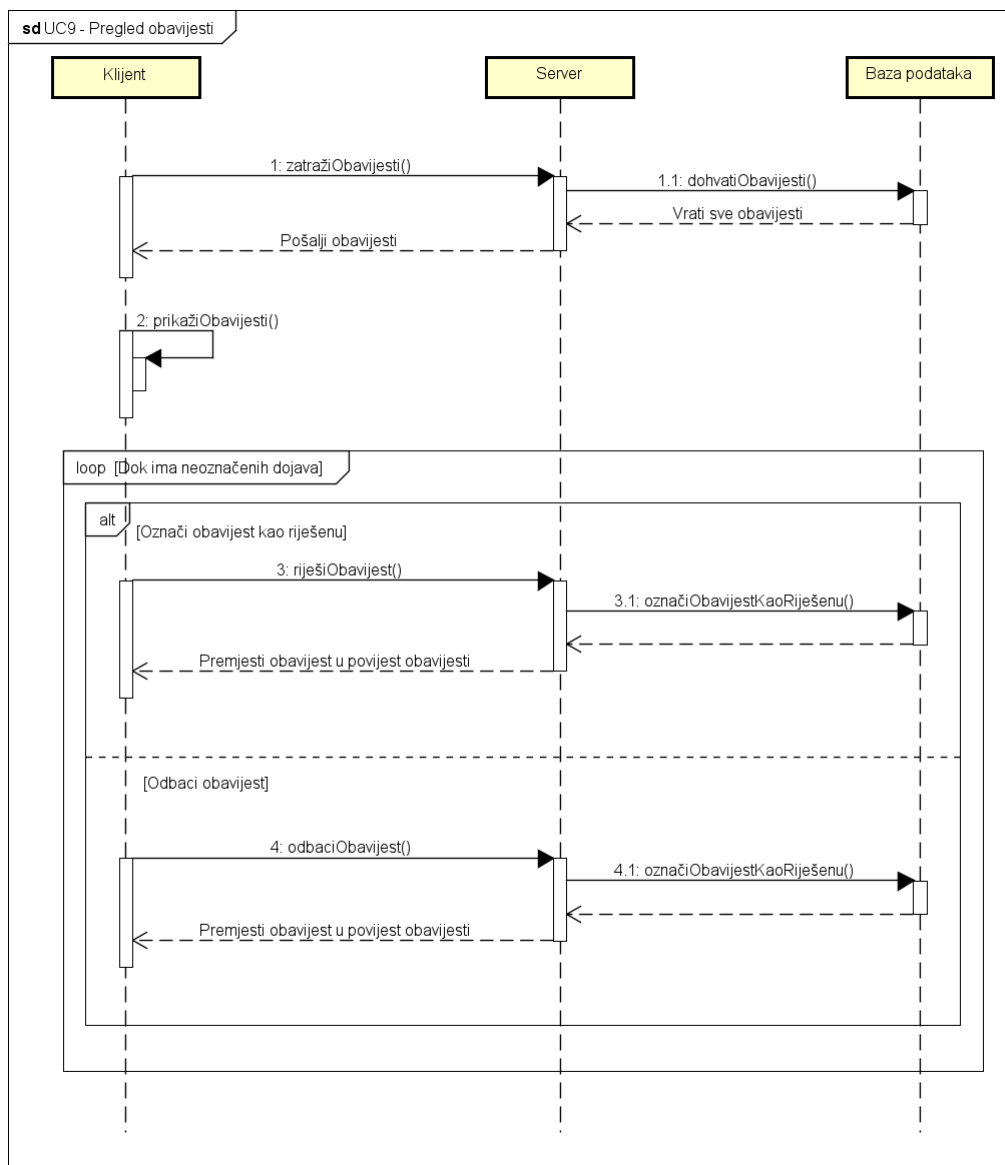


Slika 3.7: Sekvencijski dijagram za pregled dojava o niskom stanju



### Obrazac uporabe UC10 - Pristup obavijestima

Direktor odabire opciju za prikaz aktivnih obavijesti ili povijesti obavijesti. Ako odabere aktivne obavijesti, prikažu mu se sve trenutno neoznačene obavijesti te ih on ima opciju označiti kao riješene ili odbačene. Nakon što ih korisnik označi, aktivne obavijesti se premještaju u povijest obavijesti.



Slika 3.8: Sekvencijski dijagram za pregled obavjesti

## 3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu pri unosu i prikazu tekstualnog sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Sustav mora biti implementiran kao mobilna aplikacija koristeći objektno-orijentirane jezike
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Sustav treba biti jednostavan za korištenje, korisnici ne moraju znati koristiti sučelje bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen preko mobilnog uređaja
- Artikli u skladištima moraju biti pod hrvatskim nazivima

## 4. Arhitektura i dizajn sustava

- Arhitekturu možemo podijeliti u tri podsustava:
  - Baza podataka
  - Server
  - Mobilna aplikacija
- **Baza podataka** je pisana u jeziku PostgreSQL zbog znanja rada s njom iz prijašnjih kolegija. Baza se sastoji od svih potrebnih tablica kao što su tablice za validaciju korisnika aplikacije, tablice za manipulaciju artiklima i tablice za obavijesti i promjene.
- **Server** je pisan objektno orijentiranim jezikom Java 17. Pri izradi servera korišten je framework Spring Boot zbog prijašnjeg iskustva članova tima u Spring-u. Na serveru je napravljena raspodjela u obliku: controllers, services i entities zbog bolje preglednosti koda i lakšeg budućeg razvoja aplikacije. Odabrano razvojno okruženje je Eclipse i java project je napisan u arhitekturi maven project-a. Za pristup i manipulaciju bazom podataka smo koristili Spring JPA sistem.
- **Mobilna aplikacija** je pisana u react native-u. Primarni razlog odabira react native-a je zbog kompatibilnosti s Android i IOS uređajima u isto vrijeme. Odabrano razvojno okruženje je VS Code.
- Komunikaciju između podsustava smo omogućili preko besplatne platforme Heroku. Jedini problem s trenutnim načinom deploy-a je što Heroku stavlja podsustave u sleep mode ako nisu korišteni neko vrijeme. Razmatraju se bolje alternative.

### 4.1 Baza podataka

Za potrebe sustava koristit ćemo relacijsku bazu podataka koja svojom strukturom olakšava modeliranje stvarnog svijeta. Gradivna jedinka baze je relacija,

odnosno tablica koja je definirana svojim imenom i skupom atributa. Osnovna zadaća baze podataka brza je i jednostavna pohrana, izmjena i dohvat podataka za daljnju obradu. Baza podataka izrađene aplikacije sastoji se od navedenih entiteta:

- Uloga
- Skladište
- Korisnik
- Artikl
- Skupina

#### 4.1.1 Opis tablica

**Uloga** Ovaj entitet označava pojedinu ulogu korisnika koji se prijavljuje u aplikaciju. Uloga korisnika može biti: skladištar, šef skladišta, direktor. Sadrži attribute: identifikacijski broj, naziv. Ovaj entitet u vezi je *One-To-Many* s entitetom Korisnik preko identifikatora uloge.

Uloga		
id-uloga	INT	jedinstveni identifikator uloge
naziv-uloga	VARCHAR	ime uloge

**Skladište** Ovaj entitet označava skladište u kojemu se nalazi korisnik prijavljen u aplikaciji. Sadrži attribute: identifikacijski broj skladišta, naziv skladišta, adresu, identifikacijski broj šefa skladišta. Ovaj entitet u vezi je *One-To-One* s entitetom Korisnik preko identifikacijskog broja šefa skladišta, u vezi *One-To-Many* s entitetom Korisnik preko identifikacijskog broja skladišta, u vezi *Many-To-Many* s entitetom Artikl preko identifikacijskog broja skladišta.

Skladište		
id-skladište	INT	jedinstveni identifikator skladišta

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Skladište		
naziv-skladište	VARCHAR	naziv skladišta
adresa	VARCHAR	lokacija skladišta
id-šef-skladište	INT	identifikacijski broj šefa skladišta

**Korisnik** Ovaj entitet sadrži sve informacije o korisniku koji je prijavljen u aplikaciji. Sadrži sljedeće atribute: identifikacijski broj korisnika, naziv, lozinku pomoću koje ovjerava svoj identitet prilikom prijave u aplikaciju, atribut enabled koji označava ima li taj korisnik pristup aplikaciji ili više nije zaposlen u kompaniji, identifikacijski broj uloge kojoj pripada, identifikacijski broj skladišta u kojemu radi. Ovaj entitet u vezi je *One-To-One* s entitetom Skladište preko identifikacijskog broja korisnika, u vezi *Many-To-One* s entitetom Skladište preko identifikacijskog broja skladišta, u vezi *Many-To-One* s entitetom Uloga preko identifikatora uloge.

Korisnik		
id-korisnik	INT	jedinstveni identifikator korisnika
naziv-korisnik	VARCHAR	ime korisnika
lozinka	VARCHAR	lozinka pomoću koje korisnik ovjerava svoj identitet
id-uloga	INT	identifikacijski broj uloge korisnika
id-skladište	INT	identifikacijski broj skladišta u kojemu korisnik radi
enabled	INT	atribut koji označava ima li prijavljeni korisnik pristup aplikaciji ili više nije zaposlen u kompaniji

**Artikl** Ovaj entitet sadrži informacije o proizvodu čiji se kod skenira unutar mobilne aplikacije. Sadrži sljedeće atribute: identifikacijski broj artikla, naziv,

opis, identifikacijski broj skupine kojoj artikl pripada (grupe artikala su organizirane u stablo s korijenom 'artikl'). Ovaj entitet u vezi je *Many-To-Many* s entitetom Skladište preko identifikacijskog broja artikla, u vezi *Many-To-One* s entitetom Skupina preko identifikacijskog broja skupine.

Artikl		
id-artikl	INT	jedinstveni identifikator artikla
naziv-artikl	VARCHAR	ime artikla
opis-artikl	VARCHAR	tekstualni opis artikla
id-skupina	INT	jedinstveni identifikator skupine

**Skupina** Ovaj entitet sadrži sve potrebne informacije o grupi proizvoda kojoj skenirani artikl pripada (primjer stabla: 'proizvod - prehrambeni proizvod - mliječni proizvod - jogurt - voćni jogurt'). Sadrži sljedeće attribute: identifikacijski broj skupine, naziv, identifikacijski broj nadređene skupine kojoj pripada u hijerarhijskom stablu (osim ako nije korijen stabla). Ovaj entitet u vezi je *One-To-Many* s entitetom Artikl preko identifikacijskog broja skupine, u vezi *Many-To-One* sa samim sobom preko identifikacijskog broja nadređene skupine.

Skupina		
id-skupina	INT	jedinstveni identifikator skupine
naziv-skupina	VARCHAR	ime skupine
id-nadSkupina	INT	jedinstveni identifikator nadređene skupine proizvoda u hijerarhijskom stablu

**Skladište-artikl** Ova tablica nastala je kao posljedica veze između skeniranog artikla i skladišta u kojemu se on nalazi. Sadrži attribute: količina, identifikacijski broj skladišta, identifikacijski broj artikla.

Skladište-artikl		
id-skladište	INT	jedinstveni identifikator skladišta
id-artikl	INT	jedinstveni identifikator artikla
količina	INT	atribut koji označava količinu skeniranog artikla

**Izmjena-stanja** Ova tablica označava promjenu količine artikla u pojedinom skladištu. Svaki korisnik aplikacije ima mogućnost povećavanja ili smanjivanja broja pojedinih artikala u skladištu te se ta promjena upisuje u ovu tablicu. Sadrži sljedeće atribute: identifikacijski broj unosa, vrijeme unosa, lokaciju na kojoj se nalazi, identifikacijski broj skladišta, identifikacijski broj artikla, identifikacijski broj korisnika koji unosi izmjenu.

Izmjena-stanja		
id-unos	INT	jedinstveni identifikator unosa
id-skladište	INT	jedinstveni identifikator skladišta
id-artikl	INT	jedinstveni identifikator artikla
id-korisnik	INT	jedinstveni identifikator korisnika
lokacija	VARCHAR	atribut koji označava lokaciju skladišta u kojemu je zabilježena izmjena stanja
vrijeme	DATETIME	atribut koji označava vrijeme unosa izmjene stanja u skladištu
kolicina	VARCHAR	atribut koji označava novu očitanu količinu

**Dojava-nisko-stanje** Ova tablica označava dojavu da pojedinog artikla nema na skladištu. Svaki skladištar ima mogućnost dojave te informacije šefu skladišta. Tu informaciju šef skladišta može proslijediti direktoru kompanije ako to smatra opravdanim (Status se postavlja na 1). Šefovi skladišta imaju uvid u sve artikle u njihovom skladištu. Sadrži sljedeće atribute: identifikacijski broj dojave, vrijeme dojave, status (0 - šef skladišta još uvijek nije odlučio, -1 - šef skladišta je odbacio zahtjev, 1 - šef skladišta je zahtjev prosljedio direktoru, -2 direktor je odbacio zahtjev, 2 direktor je prihvatio zahtjev), identifikacijski broj skladišta, identifikacijski

broj artikla o kojemu je riječ.

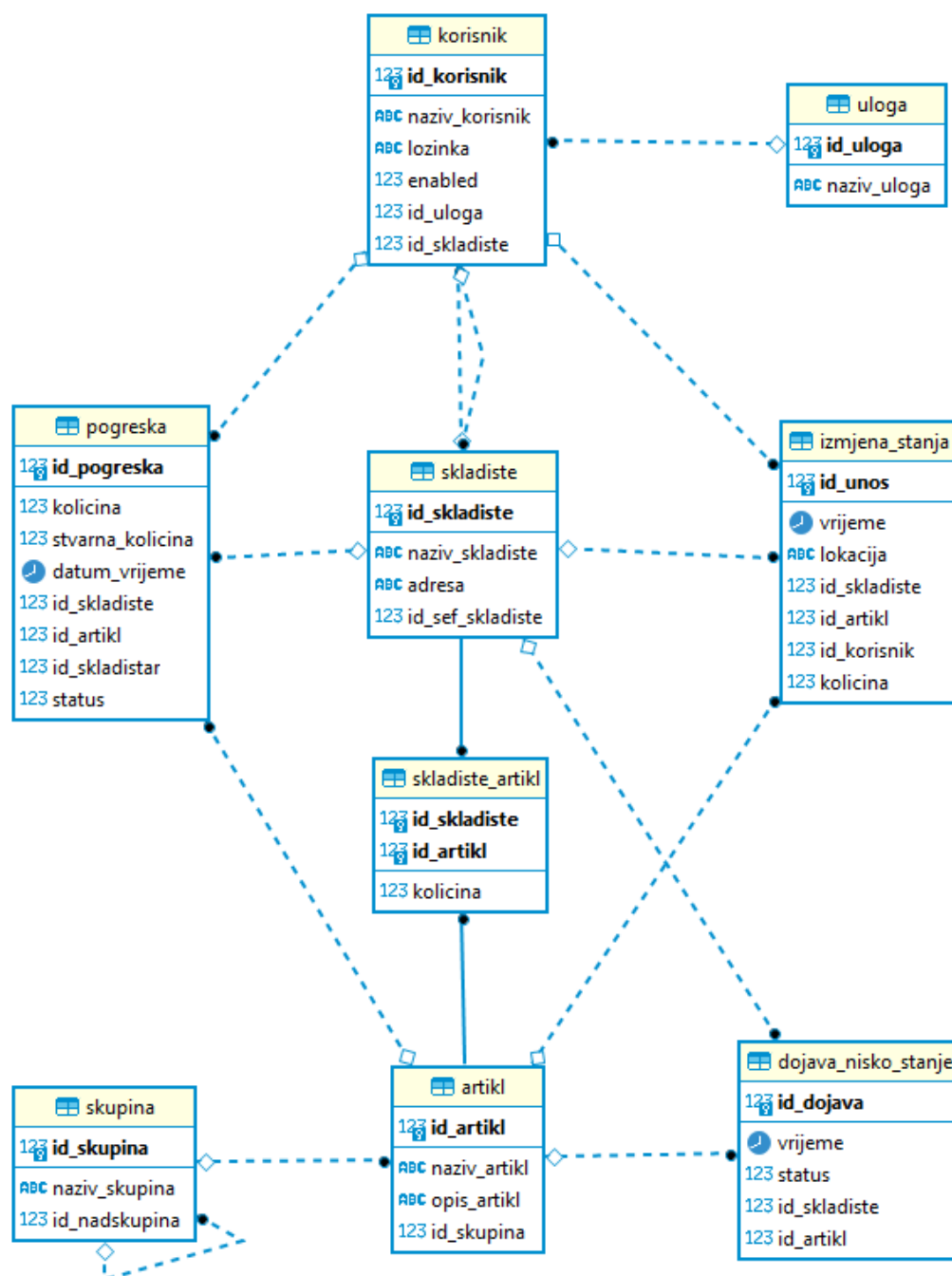
Dojava-nisko-stanje		
id-dojava	INT	jedinstveni identifikator dojava niskog stanja
id-skladište	INT	jedinstveni identifikator skladišta
id-artikl	INT	jedinstveni identifikator artikla
vrijeme	DATETIME	atribut koji označava vrijeme dojava niskog stanja
status	INT	stanje dojava

**Pogreška** Ova tablica označava pogrešku koja se dogodila kao posljedica netočne prijave pojedinih artikala na razini skladištara. Ta se pogreška onda šalje putem automatske obavijesti direktoru kompanije. Sadrži sljedeće attribute: identifikacijski broj pogreške, količina artikla koja je prijavljena, stvarna količina artikla, datum i vrijeme pogreške, identifikacijski broj skladišta u kojemu se dogodila pogreška, identifikacijski broj artikla o kojemu se radi, identifikacijski broj skladištara koji je uzrokovao pogrešku te status pogreške.

Pogreška		
id-pogreška	INT	jedinstveni identifikator pogreške
id-skladište	INT	jedinstveni identifikator skladišta
id-artikl	INT	jedinstveni identifikator artikla
id-skladištar	INT	jedinstveni identifikator skladištara
količina	INT	atribut koji označava količinu artikla koja je prijavljena
stvarna-količina	INT	atribut koji označava stvarnu količinu artikla na skladištu
datum-vrijeme	DATETIME	datum i vrijeme pogreške
status	INT	stanje pogreške



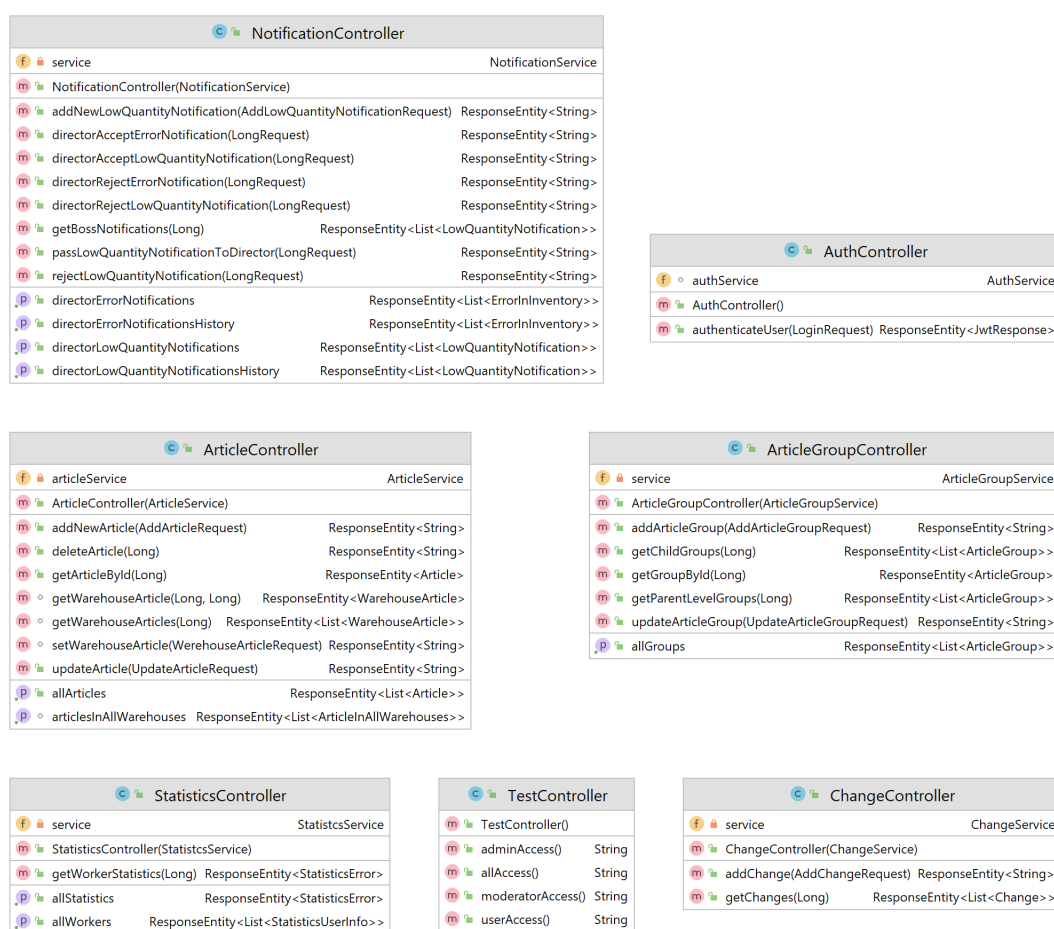
### 4.1.2 Dijagram baze podataka



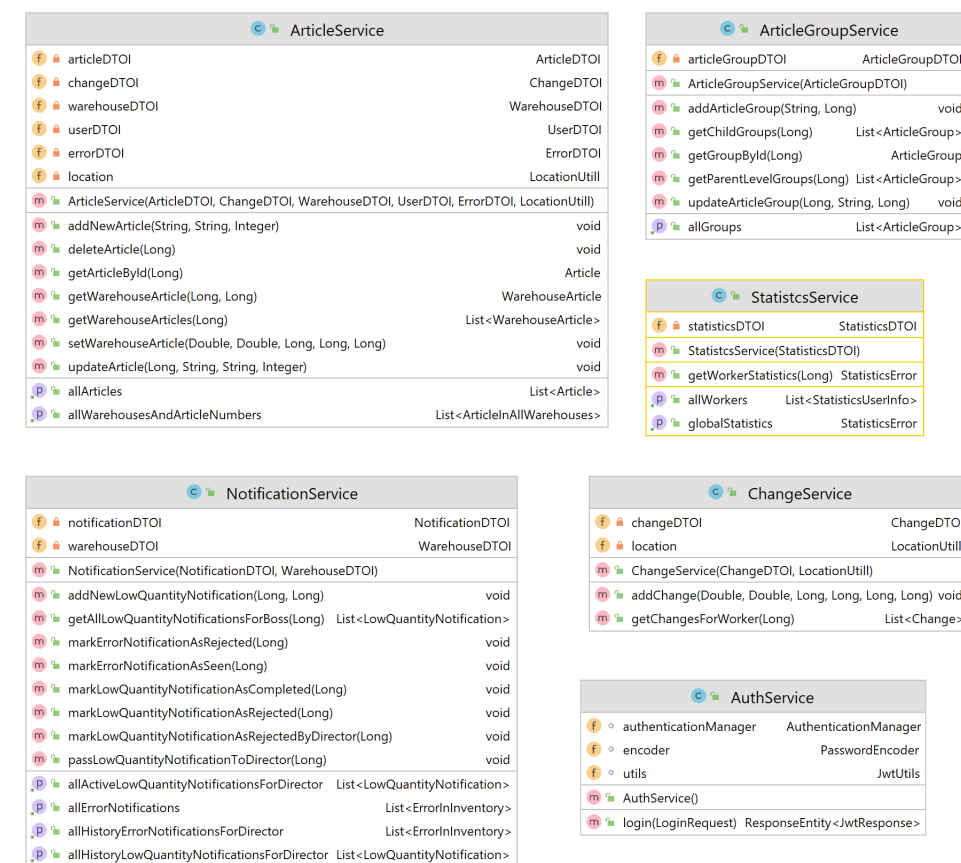
Slika 4.1: relacijski model baze podataka

## 4.2 Dijagram razreda

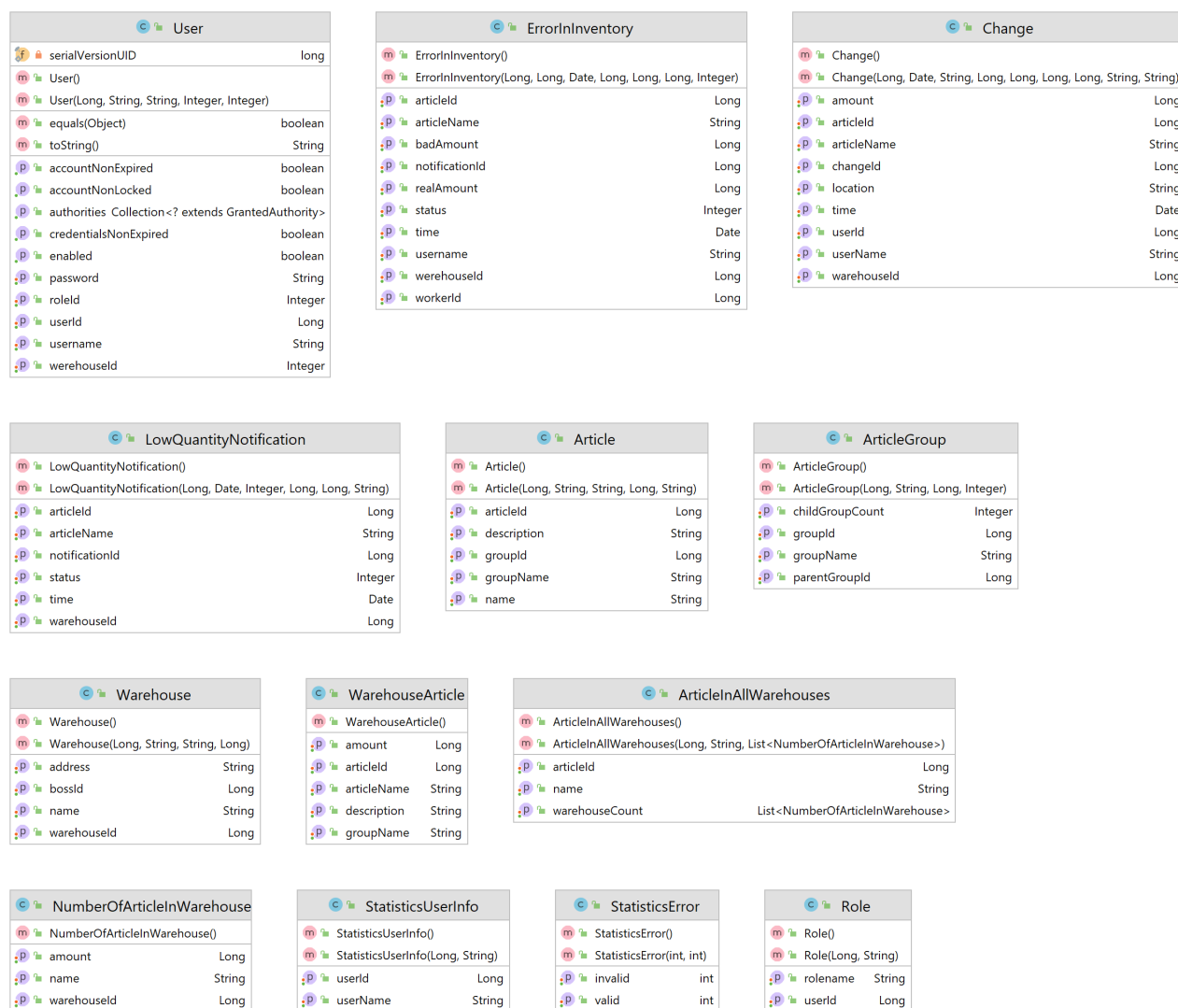
Na slikama 4.2, 4.3 i 4.4 su prikazani razredi servera pisanog u Javi. Klase koje implementiraju sučelje controller prikazane su na dijagramu 4.2. Grupirani su u taj dijagram zato što nam te klase služe kao prva linija pristupa našem serveru. Dijagram 4.3 sadrži sve klase koje imaju ulogu servisa. Servise koristimo kao glavne klase za računanje određenih kompleksnijih zahtjeva. Na servise se može gledati kao na "mozak" servera. Na svaki servise se mora vezati određeni repository koji prikazuje 1 na 1 reprezentaciju baze podataka. Dijagram 4.4 prikazuje klase koje koristimo kao vlastite modele za pomoć pri manipulaciji određenim entitetima.



Slika 4.2: Dijagram razreda za controllere



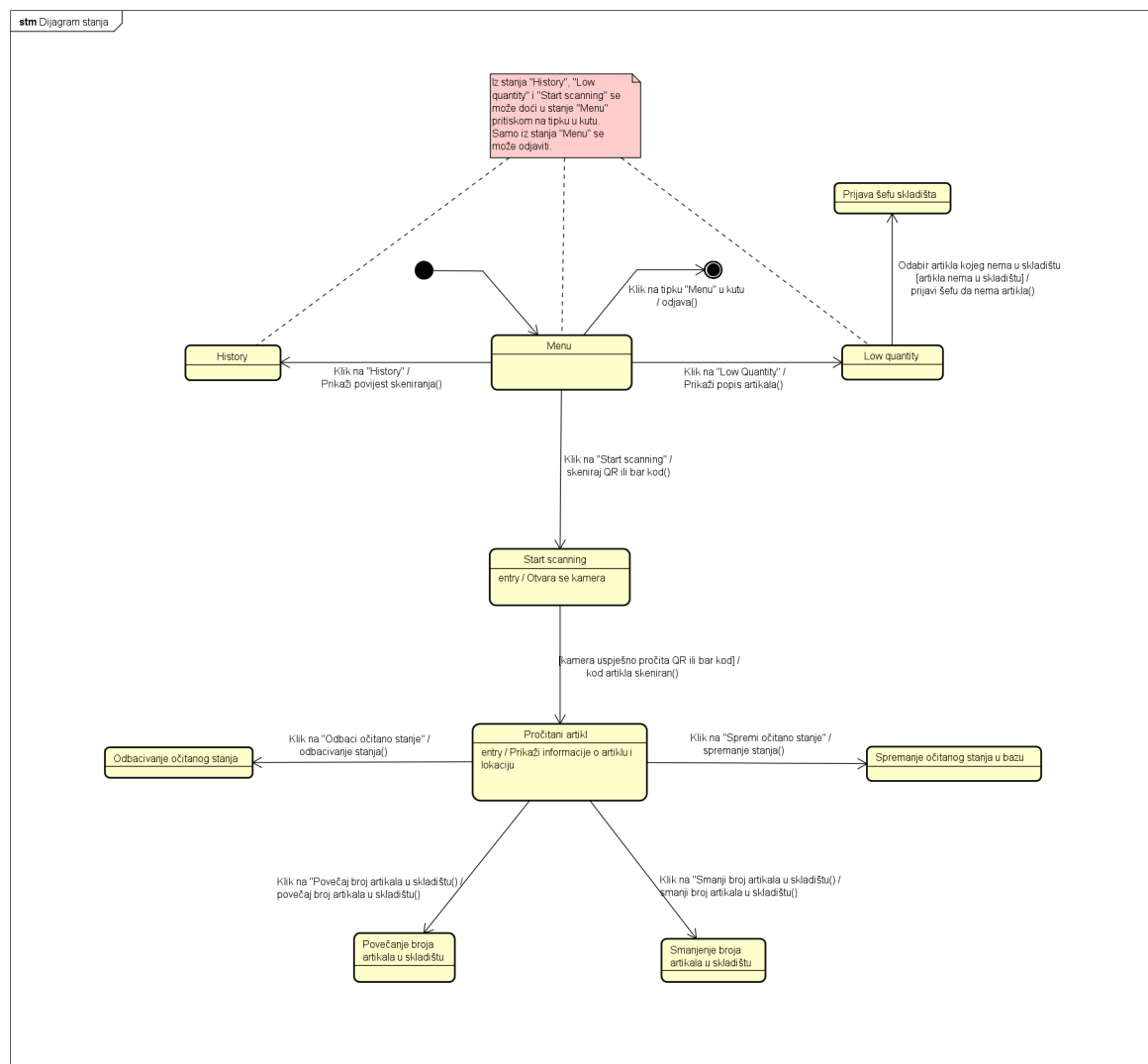
Slika 4.3: Dijagram razreda za servise



Slika 4.4: Dijagram razreda za entitete

## 4.3 Dijagram stanja

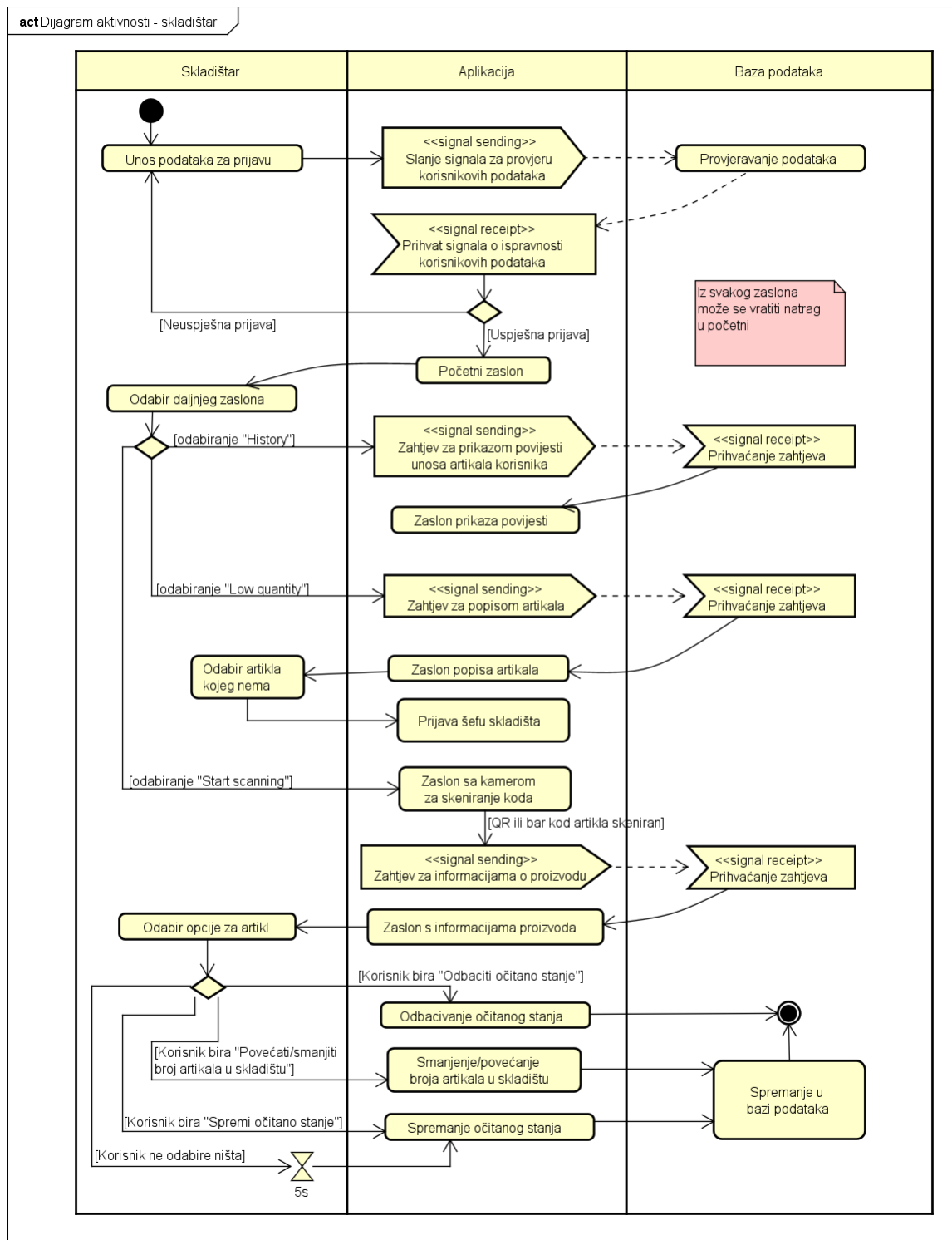
Dijagram stanja prikazuje prijelaze iz jednog stanja u drugo prouzrokovane nekim događajima. Ovdje je prezentiran dijagram stanja za skladištara, prvenstveno jer se njegove aktivnosti na ovoj aplikaciji svrstavaju kao najznačajnije. Na slici je prikazan tijek aplikacije nakon što se skladištar uspješno prijavi. Na početku se nalazi na glavnom zaslonu („Menu“) iz kojega se može doći do zaslona s poviješću skeniranih artikala, zaslona s novim zahtjevom za skeniranje te zaslona za prijavljivanje da određenoga artikla nema u skladištu. Prijavljivanjem izostanka određenog artikla automatski se obavijesti šef skladišta. Za određeni skenirani artikl skladištar dobije informacije o artiklu, također ima na izbor zatražiti „Povećanje broja artikala u skladištu“ ili „Smanjenje broja artikala u skladištu“, odnosno spremiti novo očitano stanje ili ga odbaciti.



Slika 4.5: Dijagram stanja

## 4.4 Dijagram aktivnosti

Dijagram aktivnosti opisuje model toka upravljanja, odnosno toka podataka. Pokazuje kako se odvija neka aktivnost. Na ovom dijagramu pokazuje se tok korištenja aplikacije kroz ulogu zaposlenika skladišta. Nakon što se prijavi, skladištar se nalazi na početnom zaslonu. Ako želi pogledati povijest svojih skeniranih artikala ili prijaviti da nekog artikla nema u skladištu onda bira tipku „History“ ili tipku „Low quantity“. Ako želi pročitati informacije o nekom artiklu onda mora skenirati njegov QR ili bar kod. Nakon što to napravi skladištar može zatražiti da se poveća/smanji broj tog artikla u skladištu, spremiti novo očitano stanje ili odbaciti novo očitano stanje.

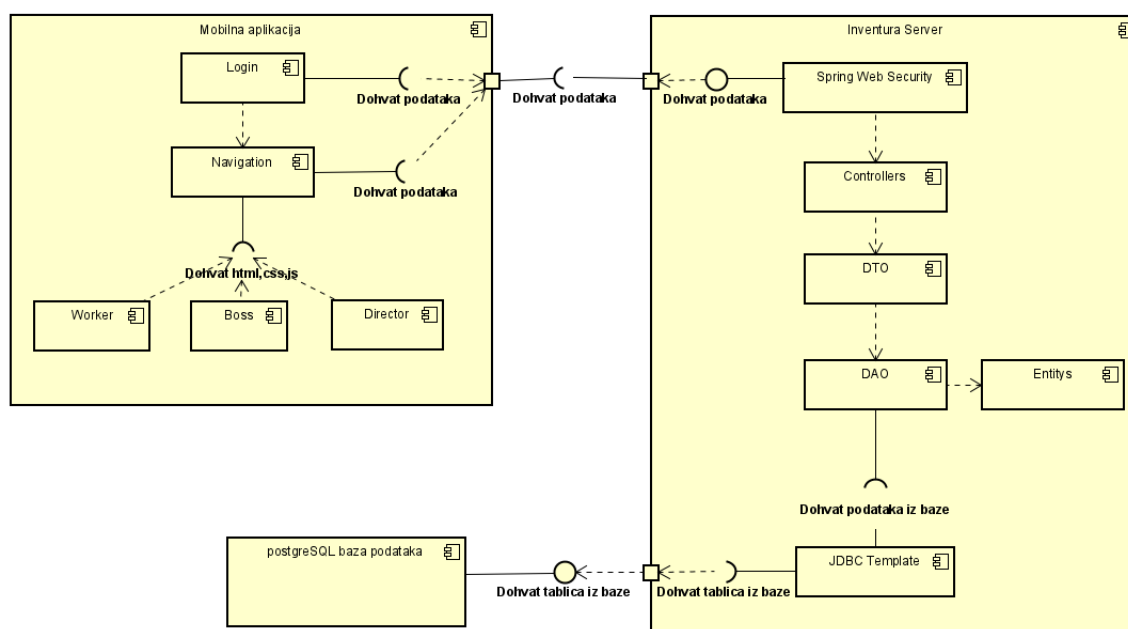


Slika 4.6: Dijagram aktivnosti



## 4.5 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.7 opisuje organizaciju i komunikaciju između komponenti. Aplikaciji se pristupa preko mobilne aplikacije. Na mobilnoj aplikaciji se nalaze sve html, css i js datoteke potrebne za rad aplikacije. Mobilna aplikacija traži sve podatke sa server-a. Jedini zahtjevi koji nisu zaštićeni JWT tokenom su zahtjevi za autorizaciju. Svi ostali zahtjevi u sebi moraju sadržavati JWT token. Server sve zahtjeve koje prima provodi kroz konfigurirani spring security, nakon čega dolazi na kontrolere i ovisno o endpointu pristupa za to delegiranom servisu. U servisima se nalazi sva logika i funkcije koje pozivaju DAO u kojima su sql upiti prema bazi podataka.



Slika 4.7: Dijagram komponenti

## 5. Implementacija i korisničko sučelje

### 5.1 Korištene tehnologije i alati

Komunikacija unutar tima odvijala se putem (internet) aplikacije Discord<sup>1</sup>, dok je kao sustav za upravljanje inačicama izvornog koda korišten GitLab<sup>2</sup>. Izrada UML dijagrama realizirana je korištenjem programa AstahUML<sup>3</sup> čija nam je licenca besplatno dostupna kao studentima. Astah UML je alat za modeliranje koji podržava UML dijagrame i mentalne mape, jednostavan je za učenje i jednostavan za korištenje i stvaranje UML dijagrama.

Za pomoć pri izradi baze podataka korišten je alat ERDPlus<sup>4</sup>. ERDPlus je web-baziran alat za modeliranje baze podataka koji omogućuje brzo i jednostavno stvaranje:

1. Dijagrama odnosna entiteta (ER dijagram)
2. Relacijske sheme (Relacijski dijagrami)
3. Zvezdane sheme (dimenzionalni modeli)

Osim toga ERD plus također nudi sljedeće mogućnosti

1. automatsko pretvaranje ER dijagrama u relacijsku shemu
2. automatsko pretvaranje relacijske sheme u SQL kod
3. export dijagrama u .PNG obliku
4. spremanje učinjenog online

Za upravljanje bazom podataka korišten je alat pgAdmin<sup>5</sup> te malo napredniji alati DBeaver<sup>6</sup> i DataGrip<sup>7</sup>. Svi ovi alati nude mogućnost izvršavanja upita koja nam je

---

<sup>1</sup><https://discord.com/>

<sup>2</sup><https://about.gitlab.com/>

<sup>3</sup><https://astah.net/>

<sup>4</sup><https://erdplus.com/>

<sup>5</sup><https://www.pgadmin.org/>

<sup>6</sup><https://dbeaver.io/>

<sup>7</sup><https://www.jetbrains.com/datagrip/>

bila najbitnija, dok napredniji alati nude i mogućnost generiranja relacijske sheme, pomoći pri pisanju upita i sličnog.

Kao razvojno okruženje za korišteni su IntelliJ IDEA<sup>8</sup> - IDE (integrirano je razvojno okruženje) tvrtke JetBrains, Microsoft Visual Studio Code (VSC)<sup>9</sup> - IDE tvrtke Microsoft te Eclipse<sup>10</sup> - IDE tvrtke Eclipse Foundation. IntelliJ IDEA te VSC koriste se za razvoj programske potpore za operacijske sustave, web stranice i aplikacije, mobilne aplikacije i drugo, dok je IDE Eclipse najpoznatiji kao razvojno okruženje za pisanje koda u Javi.

Mobilnu aplikacija (frontend) napisana je koristeći React Native<sup>11</sup>. React Native je open-source radni okvir nastao 2015. godine od strane Facebooka (Meta Platforms). React Native baziran je na JavaScriptu te omogućuje izradu nativnih mobilnih aplikacija za Android i iOS korištenjem iste kodne baze; odnosno iz istog koda možemo generirati nativne aplikacije za oba operacijska sustava! Osim samog React Nativea korištene su razne dodatne biblioteke poput biblioteke korištene za očitavanje QR i 2D kodova. Mobilna aplikacija je generirana uz pomoć Expo<sup>12</sup>.

Web server je napisan koristeći Java Spring Boot<sup>13</sup>. Java Spring Boot (Spring Boot) je alat koji čini razvoj web aplikacije i mikroservisa sa Spring Frameworkom bržim i lakšim kroz tri osnovne mogućnosti: Autokonfiguraciju. Odlučan pristup konfiguraciji. Mogućnost izrade samostalnih aplikacija

Backend (web server i baza podataka) deployani su na Heroku<sup>14</sup>. Heroku je cloud-platforma koja podržava nekolicinu jezika i omogućava jednostavno puštanje aplikacija u pogon.

---

<sup>8</sup><https://www.jetbrains.com/idea/>

<sup>9</sup><https://code.visualstudio.com/>

<sup>10</sup><https://www.eclipse.org/>

<sup>11</sup><https://reactnative.dev/>

<sup>12</sup><https://expo.dev/>

<sup>13</sup><https://spring.io/projects/spring-boot>

<sup>14</sup><https://dashboard.heroku.com/>

## 5.2 Ispitivanje programskog rješenja

Ispitivanje je provedeno na simulatoru mobilne aplikacije i serveru zasebno, na simulatoru se testira cjelokupni sustav s alatom po nazivu Appium, a server je testiran s JUnit testovima nad Mock serverom. Tijekom razvoja aplikacije, inkrementalno smo testirali sve funkcionalnosti. Ovo je provedeno redovnim pregledavanjem napisanog koda, kao i korištenjem usluga aplikacije Postman za ispitivanje ispravnosti slanja upita na server i dobivanja ispravnih podataka od servera.

### 5.2.1 Ispitivanje komponenti

Test zabranjenog pristupa U ovom testu ispituje se reagiranje servera ako korisnik pokuša pristupiti podacima za koje nije ovlašten. Važno je napomenuti da samo direktor ima pristup statistici, dok je za ispitne slučajeve „korisnik“ ulogiran kao skladištar.

```
@BeforeEach
void signIn() throws Exception {
    MvcResult result = mockMvc.perform(post("/signin")
        .content("{\"username\":\"Worker1\",\"password\":\"worker\"}")
        .contentType(MediaType.APPLICATION_JSON)
        .characterEncoding("UTF-8"))
        .andExpect(status().isOk())
        .andReturn();

    this.token = JsonPath.read(result.getResponse().getContentAsString(), "$.token");
    this.userId = JsonPath.read(result.getResponse().getContentAsString(), "$.id");
}
```

Slika 5.1: beforeEach setup

```
@Test
@Order(1)
void testUnauthorized() throws Exception {
    mockMvc.perform(get("/statistics/all")
        .header("Authorization", "Bearer " + this.token)
        .contentType(MediaType.APPLICATION_JSON)
        .characterEncoding("UTF-8"))
        .andDo(print())
        .andExpect(status().isForbidden());
}
```

Slika 5.2: Test 1

Test dohvaćanja i izmjenjivanja podataka o artiklu U ovom testu ispituje se ispravnost dohvaćanja, kao i izmjene podataka o određenom artiklu. Na početku se dohvate podaci o traženom artiklu te se spremaju u pomoćni objekt.

```
@Test
@Order(2)
void getArticleWithId1() throws Exception {
    MvcResult result = mockMvc.perform(get("/articles/byId")
        .param("id", "2")
        .header("Authorization", "Bearer " + this.token)
        .contentType(MediaType.APPLICATION_JSON)
        .characterEncoding("UTF-8"))
        .andDo(print())
        .andExpect(status().isOk())
        .andReturn();

    JSONObject tmp = new JSONObject(result.getResponse().getContentAsString(StandardCharsets.UTF_8));
```

Slika 5.3: Test 2

Potom se tom istom artiklu izmjenjuje podaci u bazi (naziv i opis se postavljaju na „test“) te se ponovno dohvaćaju kako bismo bili sigurni da su dohvaćeni ispravni podaci.

```
this.mockMvc.perform(put("/articles/updateArticle")
    .content("{\"articleId\":\"2\",\"articleName\":\"test\",\"articleDescription\":\"test\",\"groupId\":\"1\"}")
    .header("Authorization", "Bearer " + this.token)
    .contentType(MediaType.APPLICATION_JSON)
    .characterEncoding("UTF-8"))
    .andDo(print())
    .andExpect(status().isOk());

mockMvc.perform(get("/articles/byId")
    .param("id", "2")
    .header("Authorization", "Bearer " + this.token)
    .contentType(MediaType.APPLICATION_JSON)
    .characterEncoding("UTF-8"))
    .andDo(print())
    .andExpect(status().isOk())
    .andExpect(content().json(
        "{\"articleId\":\"2\", \"name\":\"test\", \"description\":\"test\", \"groupId\":\"1\", \"groupName\":\"proizvod\"}"));
```

Slika 5.4: Test 2

Konačno, zadanom artiklu se u bazi vraćaju stari podaci koji su pohranjeni u pomoćnom objektu.

```

String name = tmp.getString("name");
String desc = tmp.getString("description");

this.mockMvc.perform(put("/articles/updateArticle")
    .content("{\"articleId\": " + tmp.getInt("articleId")
        + ", \"articleName\": \"" + name + "\""
        + ", \"articleDescription\": \"" + desc + "\""
        + ", \"groupId\": " + tmp.getInt("groupId") + "}")
    .header("Authorization", "Bearer " + this.token)
    .contentType(MediaType.APPLICATION_JSON)
    .characterEncoding("UTF-8"))
    .andDo(print())
    .andExpect(status().isOk());
}

```

Slika 5.5: Test 2

Test stvaranja dojava o niskom stanju U ovom testu se ispituje stvaranje nove dojava o niskom stanju.

```

@Test
@Order(3)
void postLowQuantityNotification() throws Exception {
    this.mockMvc.perform(post("/notifications/addNotification")
        .content("{\"userId\": \"" + this.userId + "\", \"articleId\": \"1\"}")
        .header("Authorization", "Bearer " + this.token)
        .contentType(MediaType.APPLICATION_JSON)
        .characterEncoding("UTF-8"))
        .andDo(print())
        .andExpect(status().isOk());
}

```

Slika 5.6: Test 3

Test dohvaćanja dojava o niskom stanju U ovom testu ispituje se dohvaćanje dojava za određenog šefa skladišta. Test započinje tako što se „korisnik“ ulogira kao šef skladišta kako bi imao pristup dojavama.

```

@Test
@Order(4)
void getLowQuantityNotification() throws Exception {
    MvcResult result = mockMvc.perform(post("/signin")
        .content("{\"username\": \"Boss1\", \"password\": \"boss\"}")
        .contentType(MediaType.APPLICATION_JSON)
        .characterEncoding("UTF-8"))
        .andExpect(status().isOk())
        .andReturn();
}

```

Slika 5.7: Test 4

Nakon toga, dohvaćaju se sve dojave za ulogiranog šefa skladišta i spremaju u pomoćni JSONArray.

```
this.token = JsonPath.read(result.getResponse().getContentAsString(), "$.token");
this.userId = JsonPath.read(result.getResponse().getContentAsString(), "$.id");

result = this.mockMvc.perform(get("/notifications/bossNotifications")
    .param("bossId", this.userId.toString())
    .header("Authorization", "Bearer " + this.token)
    .contentType(MediaType.APPLICATION_JSON)
    .characterEncoding("UTF-8"))
    .andDo(print())
    .andExpect(status().isOk())
    .andReturn();

JSONArray obj = new JSONArray(result.getResponse().getContentAsString(StandardCharsets.UTF_8));
```

Slika 5.8: Test 4

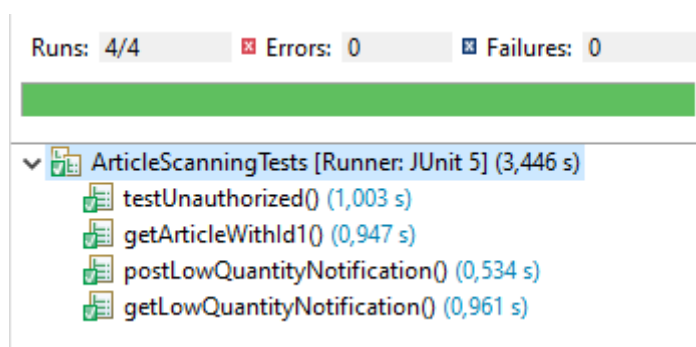
Konačno, provjerava se ispravnost dojava stvorene u prošlom testu te se ona odbija (status joj se postavlja na -1) kako se ne bi pojavljivala niti šefu skladišta niti direktoru u aktivnim ili riješenim dojavama.

```
Integer notificationId = obj.getJSONObject(obj.length()-1).getInt("notificationId");

this.mockMvc.perform(post("/notifications/rejectNotification")
    .content("{\"value\":"+notificationId+"}")
    .header("Authorization", "Bearer " + this.token)
    .contentType(MediaType.APPLICATION_JSON)
    .characterEncoding("UTF-8"))
    .andDo(print())
    .andExpect(status().isOk());
}
```

Slika 5.9: Test 4

Svi testovi uspješno prolaze.



Slika 5.10: Rezultat

## 5.2.2 Ispitivanje sustava

Ispitivanje cijelog sustava je rađeno na mobilnoj aplikaciji tako da se kroz front naše aplikacije ispituje i ispravnost servera i baze. Testovi su pisani s alatom Ap-pium.

```
it('should login as Worker ', async function () {  
  // Mount Login screen  
  const loginScreen = await client.$('~login-view');  
  await loginScreen.waitForDisplayed({ timeout: 10000 });  
  
  // Type credentials  
  const username = await client.$('~username');  
  await username.addValue('Worker1');  
  
  const password = await client.$('~password');  
  await password.addValue('worker');  
  
  // Click on Login button  
  const loginButton = await client.$('~login-button');  
  await loginButton.click();  
  
  // Wait for login  
  const menuScreen = await client.$('~menu');  
  await menuScreen.waitForDisplayed({ timeout: 10000 });  
});
```

Slika 5.11: Testiranje logiranja workera

Kod testiranja prijave korisnika odlazi se na login view i u polja username i password se upisuju točni podaci, nakon čega se pritisće login gumb i očekuje se da nas aplikacija odvede na menu stranicu.

```
it('should send low quantity notification', async function () {  
  // Click on Low quantity button  
  const lowQuantityButton = await client.$('~low-quantity-button');  
  await lowQuantityButton.waitForDisplayed({ timeout: 5000 });  
  await lowQuantityButton.click();  
  
  // Wait for Low quantity screen to login  
  const lowQuantityScreen = await client.$('~low-quantity');  
  await lowQuantityScreen.waitForDisplayed({ timeout: 3000 });  
  
  // Click on first article  
  const lowQuantityArticle = await client.$('~low-quantity-article');  
  await lowQuantityArticle.click();  
  
  // Wait for Menu screen to mount  
  const menuScreen = await client.$('~menu');  
  await menuScreen.waitForDisplayed({ timeout: 3000 });  
});
```

Slika 5.12: Testiranje slanja low quantity notificationa



Kod testiranja slanja notifikacije šefu skladišta, nalazimo se na početku na menu stranici i s nje stišćemo gumb za low quantity, nakon čega nas aplikacija odvede na low quantity screen i tamo se odabire article za koji se šalje obavijest i vraća nas nazad na menu.

```
it('should logout and login as Boss', async function () {  
  // Logout  
  const logoutButton = await client.$('~logout-button');  
  await logoutButton.click();  
  
  // Wait for Login screen to mount  
  const loginScreen = await client.$('~login-view');  
  await loginScreen.waitForDisplayed({ timeout: 3000 });  
  
  // Login as Boss1  
  const username = await client.$('~username');  
  await username.addValue('Boss1');  
  
  const password = await client.$('~password');  
  await password.addValue('boss');  
  
  const loginButton = await client.$('~login-button');  
  await loginButton.click();  
  
  // Wait for Menu screen to mount  
  const menuScreen = await client.$('~menu');  
  await menuScreen.waitForDisplayed({ timeout: 3000 });  
});
```

Slika 5.13: Testiranje logiranja kao boss

Testiranje prijave šefa je isto kao i testiranje za workera samo s drugačijim podacima.

```
it('should dismiss notification', async function () {  
  // Click on notification button  
  const notificationButton = await client.$('~notifications-button');  
  await notificationButton.click();  
  
  // Check if notification exists  
  const notification = await client.$('~notification');  
  
  await client.execute('mobile: scroll', { direction: 'down' });  
  await notification.waitForDisplayed({ timeout: 5000 });  
  await notification.click();  
  await client.pause(2000);  
});
```

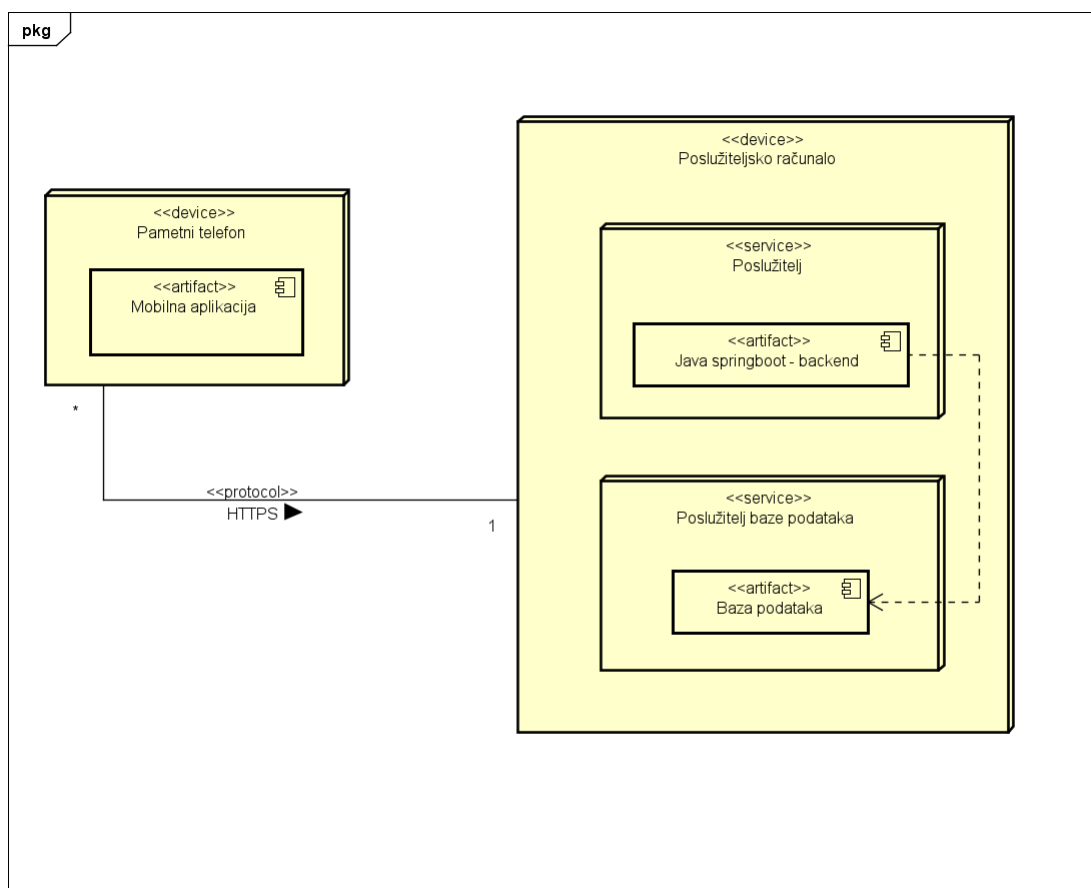
Slika 5.14: Testiranje odbacivanja notifikacije kao boss

Testiranje odbijanja notifikacije kao šef kreće s menu stranice, nakon čega se

odlazi na obavijesti, i među najnovijim obavijestima se odbija zadnje poslana obavijest.

## 5.3 Dijagram razmještaja

Dijagram razmještaja opisuje topologiju sustava i usredotočen je na odnos sklopovskih i programskih dijelova. Također sadrži prikaz sklopovskih komponenti, komunikacijskih puteva te smještaja i izvođenja programskih artefakata. Korisnici koriste pametne telefone kako bi pristupili aplikaciji. Komunikacija korisnikovog uređaja odvija se putem HTTPS veze s poslužiteljskim računalom koje sadrži komponentu baze podataka i komponentu samog poslužitelja.

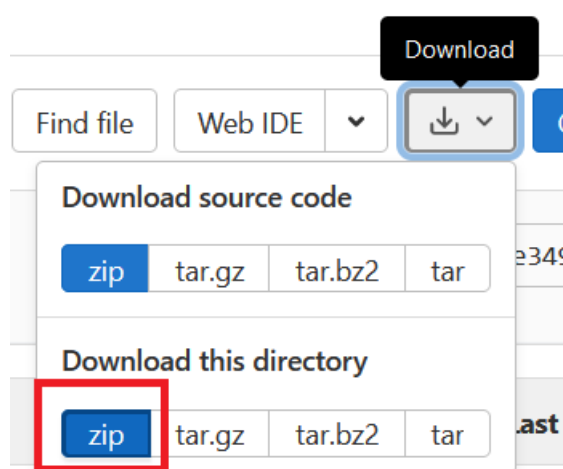


Slika 5.15: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

### 5.4.1 Backend s bazom podataka

Preporučuje se postavljanje backenda na <https://www.heroku.com/>. Heroku je cloud-platforma koja podržava nekolicinu jezika i omogućava jednostavno puštanje aplikacija u pogon. Potrebno je preuzeti izvorni kod backenda sa GitLab repozitorija koji se nalazi na grani master u mapi IzvorniKod potom backend. To se može učiniti pozicioniranjem u navedeni direktorij i pritiskom na tipku za preuzimanje trenutnog direktorija.



Slika 5.16: Preuzimanje GitLab repozitorija backenda

Najjednostavniji način puštanja koda u pogon preko stranice *Heroku* je namještanje da se aplikacija pokreće s postojećeg GitHub repozitorija. Stoga je potrebno napraviti novi GitHub repozitorij i na njega postaviti preuzeti backend iz prošlog koraka. To možemo učiniti na jednostavan način. Nakon logiranja na <https://github.com/> pritiskom na tipku "New" u gornjem lijevom ćošku zaslona potrebno je izraditi novi repozitorij proizvoljnog imena. Nakon izrade repozitorija dobivamo upute na stranici kako preko komandne linije postaviti ranije preuzeti kod na ovaj novoizrađeni repozitorij.

```
...or create a new repository on the command line

echo "# deployUpate" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ /deployUpate.git
git push -u origin main
```

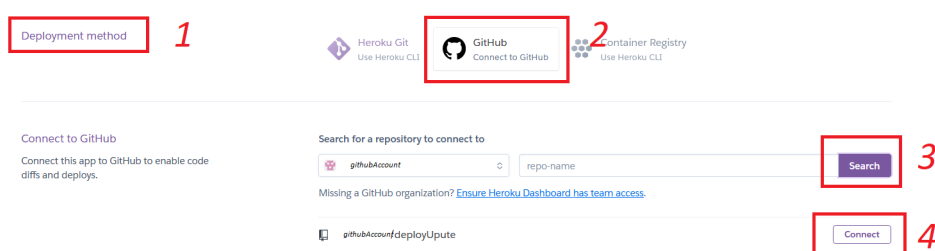
Slika 5.17: Postavljanje backenda na GitHub

Ukratko, potrebno se pozicionirati s CMD unutar ranije preuzetog repozitorija te napisati slijed naredbi:

1. git init
2. git remote add origin LINK IZ UPUTA
3. git add .
4. git commit -m "first commit"
5. git push --set-upstream origin master

nakon čega je naš gihub repozitorij spreman za deploy na Heroku.

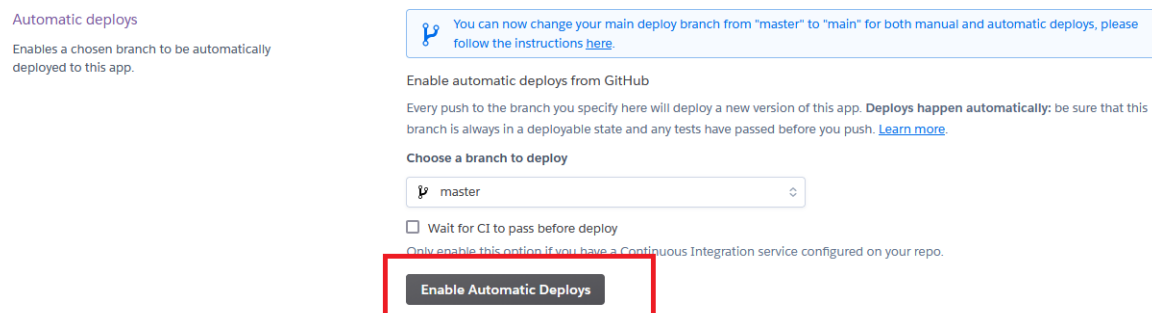
Na stranici <https://dashboard.heroku.com/apps> potrebno je u gornjem desnom dijelu ekranu pritisnuti "New" te u padajućem izborniku "Create new app". Potom odabrati proizvoljno ime aplikacije i regiju te završiti izradu nove aplikacije pritiskom na "Create app". Heroku će nas onda preusmjeriti u postavke novoizrađene aplikacije gdje odabiremo "GitHub" u izborniku "Deployment method". Prvi put potrebno je povezati vaš GitHub i Heroku profil sljedeći jednostavne korake (sve se svodi na par pritisaka tipke "Accept". Nakon povezivanja GitHuba s Heroku u izborniku "Connect to GitHub" pritiskom na "Search" možemo izlistati sve naše GitHub repozitorije te uz onaj pravi možemo odabrati "Connect".



Slika 5.18: Povezivanje GitHub repozitorija i Heroku

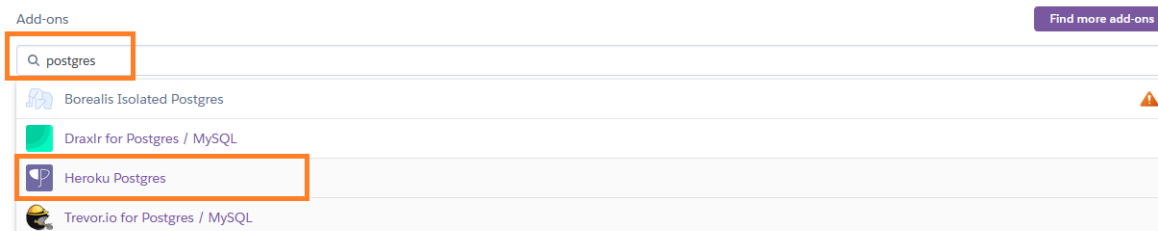
Nakon toga potrebno je još samo u novootvorenom izborniku "Automatic deploys" pritisnuti na tipku "Enable Automatic Deploys". Ova opcija omogućuje da

se poslužitelj svaki puta automatski osvježi kada učinimo promjenu na GitHub repozitoriju.



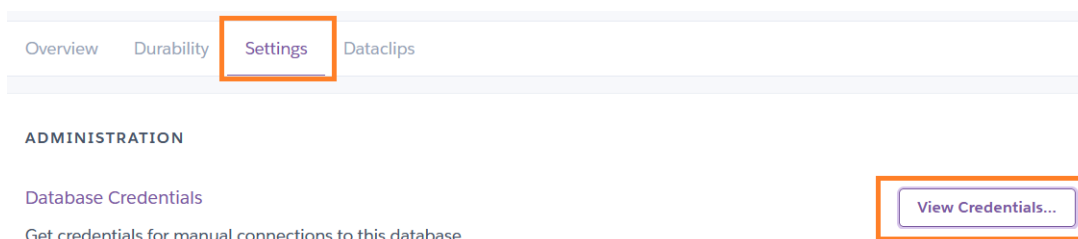
Slika 5.19: Automatski deploy na Heroku

S obzirom na to da je u pozadini našeg backenda kao metoda za pohranu podataka odabrana baza podataka, potrebno je postaviti i bazu podataka na heroku kako bi joj naš backend mogao pristupiti. Bazu podataka možemo jednostavno stvoriti klikom na "Resources" i pretraživanjem "Add-ons" gdje odabiremo "Heroku Postgres".



Slika 5.20: Dodavanje Postgres baze podataka

Kako bi mogli povezati novo stvorenu bazu podataka s našim backend poslužiteljom trebamo vjerodajnice baze podataka postaviti u src main resources application.properties. Navedene vjerodajnice baze podataka pronalazimo odabirom novostovrene baze podataka u tabu "Settings" potom "View Credentials".



Slika 5.21: Vjerodajnice baze podataka

Završno za postavljanje baze podataka, a tako i cijelog backend poslužitelja je pokretanje skripte za bazu podataka, koja stvara sve tablice i ograničenja. Navedena skripta može se pronaći na GitLab repozitoriju ovog projekta na grani master u mapi "Dokumentacija" pod imenom "databaseReset.sql".

### 5.4.2 Frontend - generiranje aplikacije

Kao i za deploy backend poslužitelja, prvi korak u izradi same mobilne aplikacije upravo je preuzimanje izvornog koda sa GitLaba. Opet, potrebno se pozicionirati na granu master u mapu "IzvorniKod" potom frontend i preuzeti trenutni direktorij. Kako bi mobilna aplikaciju uspješno slala zahtjeve "na pravu adresu" potrebno je promjeni "baseURL" u AxiosInstance.tsx datoteci koja se može pronaći u direktoriju auth. Novi baseURL je upravo onaj URL na koji nas web preglednik preusmjeri kada na Heroku pritisnemo "Open app". Nakon promjene ovog URLa, aplikacija je spremna za izradu.

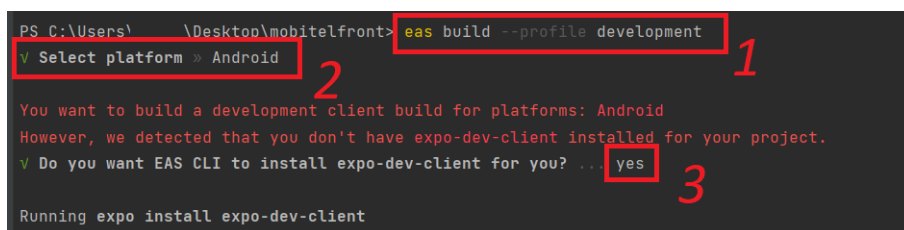
Aplikaciju "generiramo" preko alata `https://expo.dev/` gdje se prvo trebamo ulogirati/napraviti korisnički račun. Za daljnji proces potrebno je na računalu imati instaliran eas-cli. Njega možemo instalirati naredbom

1. `npm install -g eas-cli`

Nakon izvođenja ove naredbe potrebno je otvoriti terminal u mapi u kojoj se nalazi izvorni kod te napisati sljedeći slijed naredbi

1. `eas login`
2. `eas build --profile production`

Potom odabiremo platformu za koju želimo generirati aplikaciju (Android/iOS ili obje) te pričekamo. Postoji mogućnost da će biti potrebno izvršiti dodatne korake, no eas-cli nam uvelike olakšava ovaj proces te je potrebno samo prihvaćati sve što nam nudi. Na sljedećoj slici moguće je vidjeti jednostavnost procesa.



```
PS C:\Users\...\Desktop\mobitelfront> eas build --profile development
√ Select platform » Android
You want to build a development client build for platforms: Android
However, we detected that you don't have expo-dev-client installed for your project.
√ Do you want EAS CLI to install expo-dev-client for you? .. yes
Running expo install expo-dev-client
```

Slika 5.22: Početak procesa izrade mobilne aplikacije

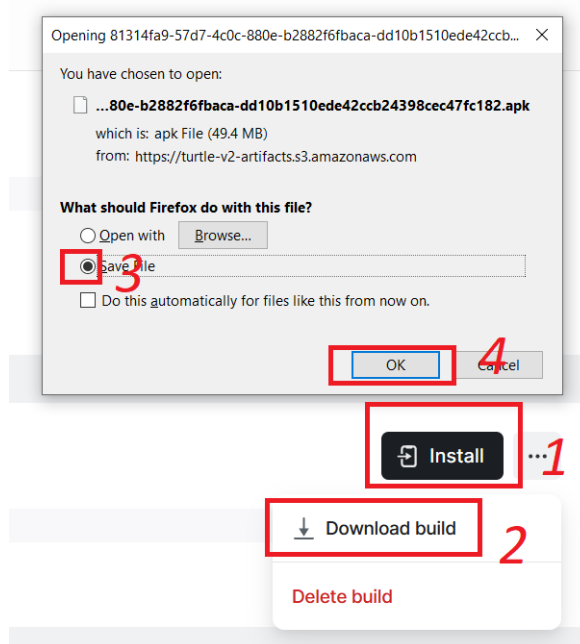
U trenutku jedan upisujemo naredbu, u trenutku dva odabiremo platformu (u ovom slučaju odabrali smo Android) pomoću strelica na tipkovnici i u trenutku 3 EAS CLI uočava da u našem sustavu nedostaje potrebni paket "expo-dev-client" te nam nudi mogućnost instalacije istog. Njega instaliramo pritiskom tipke "y" na tastaturi. Nakon uspješnog izvršavanja ovog niza naredbi sada je potrebno čekati nekoliko minuta (oko 10min) dok se aplikacija "builda".

Nakon što je "build" gotov dobivamo QR kod i URL na kojem možemo preuzeti našu aplikaciju.



Slika 5.23: Završetak builda

Potrebno je URL otvoriti u web pregledniku. Na stranici odabiremo gumb install te potom download build s čim uspješno završavamo ovaj proces!



Slika 5.24: Preuzimanje aplikacije



## 6. Zaključak i budući rad

Projektni zadatak ove grupe bio je razvoj mobilne aplikacije za *Inventuru*. Osim same provedbe inventure ova aplikacija pruža i dodatne mogućnosti kao što je kontrola rada zaposlenika u skladištu, statistika pogrešaka zaposlenika u skladištu i slično. Vrijeme potrebno za razvijanje ove aplikacije bilo je u skladu sa zadanim vremenom od 18 tjedana te smo uspjeli implementirati sve tražene funkcionalnosti. U ovih 18 tjedana imali smo dva ciklusa rada.

Početak prvog ciklusa svakako je obilježilo formiranje timova i saznavanje da je naš zadatak mobilna aplikacija; što je ujedno bio prvi izazov za naš tim jer smo očekivali da će naš zadatak biti u obliku web stranice. Ovo je bio veliki tehnički izazov za sve nas jer nitko nije imao prijašnja iskustva s izradom mobilnih aplikacija, niti s potrebnim tehnologijama i alatima. Već u jako ranoj fazi projekta napravili smo bazu podataka koja je zapravo od samog početka približno jasno definirala i strukturu same aplikacije. Početni entuzijazam u timu nestao je kroz drugi tjedan rada na aplikaciji te je proces razvoja bio veoma spor. Dva tjedna prije prve predaje projekta voditelj je sazvao sastanak i podijeljene su detaljne uloge za preostali dio posla koji smo stigli obaviti na vrijeme.

Drugi ciklus izrade ove aplikacije odvijao se mnogo brže nego prvi ciklus te je u kratkom roku implementirana većina funkcionalnosti koja je od nas tražena. To možemo pripisati tome što je većina tima stekla nova znanja potrebna za razvoj ove aplikacije u prvom ciklusu te ih je sada počela upotrebljavati. Također rad tima je uvelike potaknuo voditelj tima sa sazivanjem tjednih sastanaka. Sve manje pitanja je išlo u smjeru kako nešto implementirati, već kako bi bilo najbolje da se to implementira. Za prezentiranje alfa inačice aplikacija je bila većinom spremna, jedino što nam je nedostajalo bila je statistika pogrešaka radnika, no ne zato što nismo znali kako da ju implementiramo, već nismo znali što sve točno trebamo prikazati. Nakon konzultacije s asistentom dobili smo ideju te uspješno implementirali i taj dio.

U budućnosti rada na ovom projektu u cilju bi bilo usavršiti dizajn te izraditi web stranicu. Web stranica kao takva bila bi odlična ideja, jer bi korisnicima s ulogama šefa skladišta i direktora omogućila lakši pregled svih artikala u skladi-

štima i slične funkcionalnosti koje su na razne načine limitirane veličinom zaslona mobitela.

U konačnici ovaj projekt svim članovima tima dao je razna iskustva, dok je nekima ovo bio čak prvi put rada u timu. Osim stečenog iskustva svi smo stekli i nova znanja i vještine. Naš prvi izazov, izrada mobilne aplikacije; u konačnici je urodio znanjem React Nativea za pola članova tima, dok je drugi dio tima naučio Java Spring Boot te osnove LaTeXa.

# Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. The Unified Modeling Language, <https://www.uml-diagrams.org/>
3. Astah Community, <http://astah.net/editions/uml-new>
4. MiKTeX, <https://miktex.org/>
5. ERDPlus, <https://erdplus.com/>
6. Baeldung, <https://www.baeldung.com>
7. Adobe Stock, [https://stock.adobe.com/be\\_en/](https://stock.adobe.com/be_en/)

# Indeks slika i dijagrama

2.1	Ilustracija očitavanja QR koda . . . . .	5
2.2	Provođenje inventure na papiru . . . . .	7
3.1	Use case za korisnika i skladištara . . . . .	16
3.2	Use case za šefa skladišta . . . . .	17
3.3	Use case za direktora . . . . .	18
3.4	Sekvencijski dijagram za prijavu u sustav . . . . .	19
3.5	Sekvencijski dijagram za očitavanje QR koda ili bar koda . . . . .	20
3.6	Sekvencijski dijagram za izmjenu stanja artikla u skladištu . . . . .	22
3.7	Sekvencijski dijagram za pregled dojava o niskom stanju . . . . .	23
3.8	Sekvencijski dijagram za pregled obavjesti . . . . .	24
4.1	relacijski model baze podataka . . . . .	32
4.2	Dijagram razreda za controllere . . . . .	33
4.3	Dijagram razreda za servise . . . . .	34
4.4	Dijagram razreda za entitete . . . . .	35
4.5	Dijagram stanja . . . . .	37
4.6	Dijagram aktivnosti . . . . .	39
4.7	Dijagram komponenti . . . . .	40
5.1	beforeEach setup . . . . .	43
5.2	Test 1 . . . . .	43
5.3	Test 2 . . . . .	44
5.4	Test 2 . . . . .	44
5.5	Test 2 . . . . .	45
5.6	Test 3 . . . . .	45
5.7	Test 4 . . . . .	45
5.8	Test 4 . . . . .	46
5.9	Test 4 . . . . .	46
5.10	Rezultat . . . . .	46
5.11	Testiranje logiranja workera . . . . .	47

5.12 Testiranje slanja low quantity notificationa . . . . .	47
5.13 Testiranje logiranja kao boss . . . . .	48
5.14 Testiranje odbacivanja notifikacije kao boss . . . . .	48
5.15 Dijagram razmještaja . . . . .	49
5.16 Preuzimanje GitLab repozitorija backenda . . . . .	50
5.17 Postavljanje backenda na GitHub . . . . .	51
5.18 Povezivanje GitHub repozitorija i Heroku . . . . .	51
5.19 Automatski deploy na Heroku . . . . .	52
5.20 Dodavanje Postgres baze podataka . . . . .	52
5.21 Vjerodajnice baze podataka . . . . .	53
5.22 Početak procesa izrade mobilne aplikacije . . . . .	54
5.23 Završetak builda . . . . .	54
5.24 Preuzimanje aplikacije . . . . .	55
6.1 Commitovi na projektu . . . . .	66
6.2 Commitovi na projektu . . . . .	67

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- Datum: 13. listopada 2021.
- Prisustvovali: D. Pipalović, A. Gorup, F. Bugarin, L. Škarpa, M. Hladek, P.Vulić
- Teme sastanka:
  - Odabiranje tehnologija:
    - \* Frontend: react native
    - \* Backend: java - Spring boot
    - \* Baza: postgresql
  - Dodjeljivanje uloga unutar tima:
    - \* Frontend: Paula Vulić, Leo Škarpa, Mario Hladek
    - \* Backend: Andrija Gorup, Filip Bugarin
    - \* Baze: Denis Pipalović, Vid Rebernak
- Postavljanja trello board za lakši team management
- Postavljanje GitLab repozitorija i dodavanje članova
- Kontaktiran demonstrator grupe za inicijalne prijedloge i savjete
- Dodijeljeni inicijalni zadaci - istražiti tehnologije, kako spojiti sve zajedno, upoznat ostatak tima s Git servisom

### 2. sastanak

- Datum: 17. listopada 2021.
- Prisustvovali: D. Pipalović, V. Rebernak, A. Gorup, F. Bugarin
- Teme sastanka:
  - Kolega Denis je osmislio V1.0.0 baze i prezentirao nam strukturu baze
  - Na skicama baze smo demonstrirali kolegama alat Git Extensions za commit/push na DevDoc granu
  - Dogovoreno s kolegom Andrijom oko tutoriala za Spring Boot arhi-

tekturu

- Dogovorena dokumentacija baze s kolegom Vidom

### 3. sastanak

- Datum: 28. listopada 2021.
- Prisustvovali: D. Pipalović, V. Rebernak, A. Gorup, F. Bugarin, L. Škarpa, M. Hladek
- Teme sastanka:
  - Dogovor oko dokumentacije s kolegama Denisom i Vidom
  - Dogovor oko komunikacije frontenda s backendom i autentifikacije
  - Istraživanje frontend autentifikacije za mobilnu aplikaciju
  - Demonstracija rada sa latex-om i alatom za izradu UML dijagrama

### 4. sastanak

- Datum: 6. Studenog 2021.
- Prisustvovali: D. Pipalović, V. Rebernak, A. Gorup, F. Bugarin, L. Škarpa, M. Hladek, P.Vulić
- Teme sastanka:
  - Izradili dijagrame obrazaca uporabe
  - Dodijelili izradu obrazaca uporabe svakom članu tima
  - Zaključili kako dalje nastaviti sa razvojem i što nam fali

### 5. sastanak

- Datum: 14. Studenog 2021.
- Prisustvovali: D. Pipalović, V. Rebernak, A. Gorup, F. Bugarin, L. Škarpa, M. Hladek, P.Vulić
- Teme sastanka:
  - Prolazak kroz dokumentaciju, potvrda od svih članova da se slažu s trenutnom dokumentacijom za 1. reviziju
  - Prepravke tipfelera i nesmislenih rečenica u dokumentaciji
  - Upoznavanje frontend članova tima s bazom
  - Prezentiranje strukture java projekta u Spring Boot-u
  - Prezentiranje mobilne aplikacije svim članovima tima

## 6. sastanak

- Datum: 4. Prosinca 2021.
- Prisustvovali: D. Pipalović, V. Rebernak, A. Gorup, F. Bugarin, L. Škarpa, M. Hladek, P.Vulić
- Teme sastanka:
  - Organizacija nakon međuispita o nastavku rada na projektu
  - Podjela zadataka

## 7. sastanak

- Datum: 14. Prosinca 2021.
- Prisustvovali: D. Pipalović, V. Rebernak, A. Gorup, F. Bugarin, L. Škarpa
- Teme sastanka:
  - Review prvih zadataka
  - Manje izmjene u kodu na frontu
  - Podjela novih zadataka

## 8. sastanak

- Datum: 19. Prosinca 2021.
- Prisustvovali: D. Pipalović, V. Rebernak, A. Gorup, F. Bugarin, L. Škarpa, M. Hladek
- Teme sastanka:
  - Review drugih zadataka
  - Rješavanje bugova na frontu i backu



## Tablica aktivnosti

	Fili Bugarin (voditelj)	Denis Pipalović	Paula Vulić	Andrija Gorup	Leo Škarp	Mario Hladek	Vid Rebernak
Upravljanje projektom	15	8	6	8	8	9	7
Opis projektnog zadatka		4					
Funkcionalni zahtjevi		2					
Opis pojedinih obrazaca	1	1	1	1	1	1	1
Dijagram obrazaca	3	3	3	4	3	3	3
Sekvencijski dijagrami	6			6			
Opis ostalih zahtjeva	1						
Arhitektura i dizajn sustava	1						
Baza podataka							8
Dijagram razreda	7			4			
Dijagram stanja						2	
Dijagram aktivnosti						2	
Dijagram komponenti	3						
Korištene tehnologije i alati		3					
Ispitivanje programskog rješenja	5			4	4		
Dijagram razmještaja						2	
Upute za puštanje u pogon	1	2	2				
Dnevnik sastajanja	2						

Nastavljeno na idućoj stranici

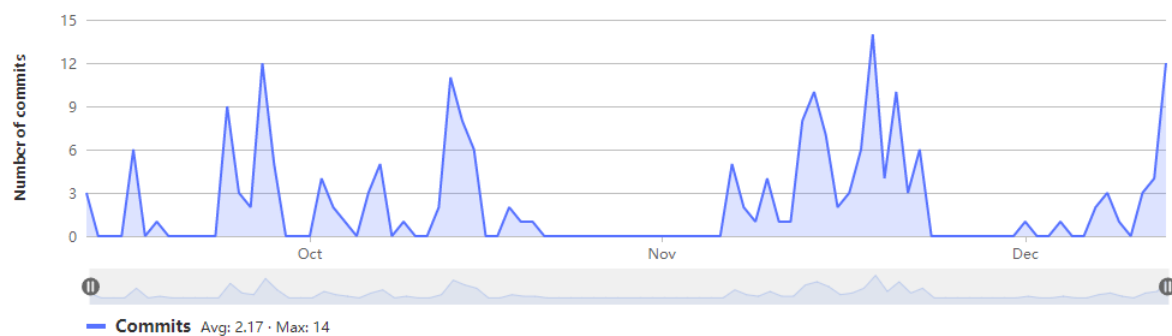
Nastavljeno od prethodne stranice

	Fili Bugarin (voditelj)	Denis Pipalović	Paula Vulić	Andrija Gorup	Leo Škarpa	Mario Hladek	Vid Rebernak
Zaključak i budući rad		3					
Popis literature		1					1
<i>Dodatne stavke</i>							
<i>izrada početne stranice</i>			30		35	11	15
<i>izrada baze podataka</i>		11					7
<i>spajanje s bazom podataka</i>	3			3			
<i>back end</i>	25	15		20			
<i>dizajn</i>	1		4				
<i>proučivanje tehnologija</i>	1	2	2	4	5	14	1
<i>ispravljanje dokumentacije</i>		1					3

## Dijagrami pregleda promjena

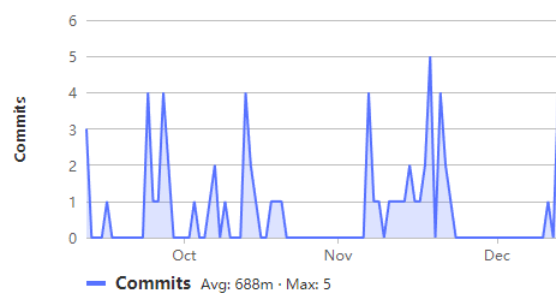
### Commits to master

Excluding merge commits. Limited to 6,000 commits.



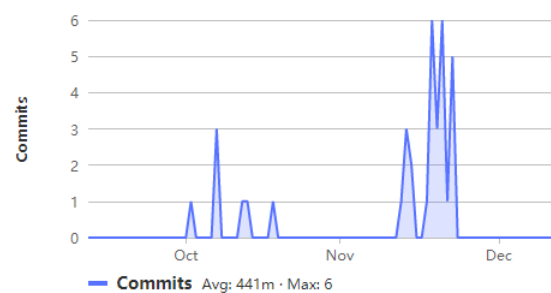
### FilipBugarin

64 commits (filip.bugarin@fer.hr)



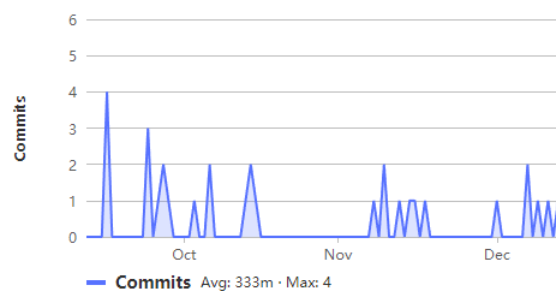
### Leo Skarpa

41 commits (ls52270@fer.hr)



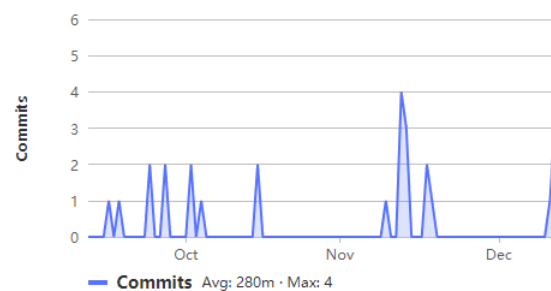
### Denis Pipalovic

31 commits (denis.pipalovic@fer.hr)



### vidre

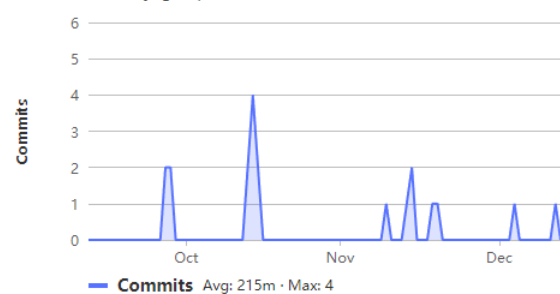
26 commits (vr119124546@fer.hr)



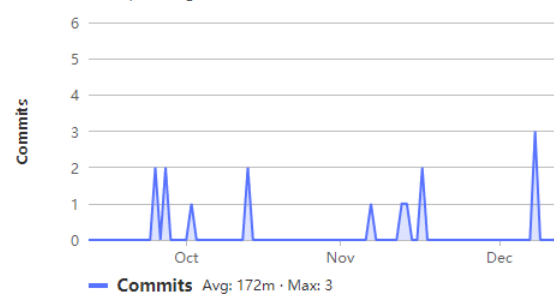
Slika 6.1: Commitovi na projektu

**AndrijaGorup**

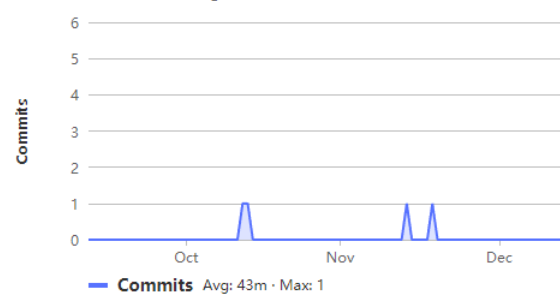
20 commits (andrija.gorup@fer.hr)

**emojapaula**

16 commits (vulpaula@gmail.com)

**MarioHla**

4 commits (mario.hladek@gmail.com)



Slika 6.2: Commitovi na projektu