

Oblikovanje programske potpore

Ak. god. 2015/2016

Sustav za arhivu i reprodukciju tonskih zapisa

Dokumentacija, revizija 2.0

Grupa: *BananaBlade*
Voditelj: *Zvonimir Jurelinac*

Datum predaje: *20. studeni 2015.*

Asistent: *Miljenko Krhen*
Nastavnik: *Vlado Sruk*

Sadržaj

| | |
|---|-----------|
| 1. Dnevnik promjene dokumentacije..... | 3 |
| 2. Opis projektnog zadatka..... | 5 |
| 3. Rječnik pojmova..... | 9 |
| 4. Funkcionalni zahtjevi..... | 10 |
| 4.1 Opis obrazaca uporabe..... | 12 |
| 4.2 Sekvencijski dijagrami..... | 27 |
| 5. Ostali zahtjevi..... | 48 |
| Nefunkcionalni zahtjevi..... | 48 |
| Zahtjevi domene primjene..... | 48 |
| 6. Arhitektura i dizajn sustava..... | 49 |
| 6.1. Svrha, opći prioriteti i skica sustava..... | 49 |
| 6.2. Dijagram razreda s opisom..... | 60 |
| 6.3. Dijagram objekata..... | 63 |
| 6.4. Ostali UML dijagrami..... | 64 |
| 7. Implementacija i korisničko sučelje..... | 68 |
| 7.1. Dijagram razmještaja..... | 68 |
| 7.2. Korištene tehnologije i alati..... | 69 |
| 7.3. Isječak programskog koda vezan za temeljnu funkcionalnost sustava..... | 70 |
| 7.4. Ispitivanje programskog rješenja..... | 74 |
| 7.5. Upute za instalaciju..... | 77 |
| 7.6. Korisničke upute..... | 81 |
| 8. Zaključak i budući rad..... | 83 |

| | |
|---|-----------|
| 9. Popis literature..... | 84 |
| Dodatak A: Dnevnik sastajanja..... | 85 |

1. Dnevnik promjene dokumentacije

| Rev. | Opis promjena / dodataka | Autor(i) | Datum |
|-------|--|--------------------------------------|---------------|
| 0.1 | Stvoren predložak za dokumentaciju | Jurelinac, Škalec | 25. 10. 2015 |
| 0.1.1 | Napisan dio opisa Dodan rječnik pojmova | Jurelinac | 25. 10. 2015 |
| 0.2 | Proširen opis zadatka | Škalec | 02. 11. 2015 |
| 0.3 | Proširen pojmovnik Započeti funkcionalni zahtjevi | Škalec | 05. 11. 2015 |
| 0.4 | Dovršeni funkcionalni zahtjevi Manje izmjene | Škalec | 10. 11. 2015 |
| 0.5 | Izmjene dijela opisa, izmijenjeni neki obrasci upotrebe, dodani neki novi, izmijenjen rječnik pojmova | Jurelinac | 14. 11. 2015 |
| 0.7 | Dodani dijagrami obrazaca uporabe, započeto opisivanje arhitekture sustava, stavljen ER model i opis relacija u bazi podataka | Jurelinac | 16. 11. 2015. |
| 0.8 | Dodan opis klijentskog dijela i upravitelja | Ivošević, Jurelinac | 17. 11. 2015. |
| 0.9 | Dodani dijagrami razreda | Jurelinac | 18. 11. 2015. |
| 0.9.1 | Dodani neki od sekvencijskih dijagrama | Mašić, Jerković, Jurelinac | 19. 11. 2015. |
| 0.10 | Dodan dijagram objekata | Škalec | 20. 11. 2015. |
| 0.11 | Dodani svi sekvencijski dijagrami i njihovi opisi | Peroš, Mašić, Jerković, Jurelinac | 20. 11. 2015. |
| 1.0 | Provjera prije predaje prve revizije | Svi | 20. 11. 2015. |
| 1.1 | Sitni popravci nakon dobivene povratne informacije | Jurelinac | 01. 01. 2016. |
| 1.2 | Dodani dijagram stanja i aktivnosti | Škalec, Peroš | 12. 01. 2016. |
| 1.3 | Dodan dijagram razmještaja, korištene tehnologije i upute za instalaciju | Jurelinac | 12. 01. 2016. |

| | | | |
|------------|----------------------------|--------------------------------|---------------|
| 1.4 | Dodan popis literature | Jurelinac | 18. 01. 2016. |
| 1.5 | Dodani ispitni slučajevi | Peroš | 18. 01. 2016. |
| 1.6 | Dodane upute za korištenje | Čupić, Škalec, Peroš, Mašić | 19. 01. 2016. |

2. Opis projektnog zadatka

Cilj ovog projekta jest razviti informacijski sustav u obliku web aplikacije čija je namjena upravljanje tonskim zapisima internetske radio postaje. Aplikacija bi korisnicima – vlasniku, administratorima, glazbenim urednicima i registriranim korisnicima – trebala omogućiti brzo, jednostavno i lako dostupno ispunjavanje svojih zaduženja i sudjelovanje u radu radio postaje. Također, posjetiteljima web stranice na kojoj se nalazi aplikacija trebalo bi biti omogućeno slušanje trenutno sviranog zvučnog zapisa na toj radio postaji.

Kako bi aplikacija bila prikladna što širem spektru korisnika, poželjno je da ona bude pristupačna, pregledna i dovoljno jednostavna za korištenje kako bi se njome mogli služiti i korisnici bez velikog informatičkog znanja. Također, bilo bi poželjno da dizajn aplikacije bude privlačan i moderan.

Detaljniji rad ove aplikacije je sljedeći: Za svaki dan unaprijed će se stvarati nova glazbena lista radio postaje, i to na način da će svaki glazbeni urednik stvarati liste za njemu dodijeljene termine unutar toga dana (jedan dodijeljeni termin traje sat vremena). Svi zapisi za reprodukciju moraju biti poznati najmanje 24 sata prije vremena njihove reprodukcije. Registrirani će korisnici stvaranjem lista želja, u koje će urednici imati uvid, moći i sami sudjelovati u odlučivanju o programu radio postaje. Administratori radio postaje moći će upravljati zvučnim zapisima kojima postaja raspolaže, kao i drugim korisnicima – moći će postavljati glazbene urednike, odlučivati o njima dodijeljenim terminima te uređivati podatke svih ostalih korisnika (izuzev vlasnika i drugih administratora). Administratore postavlja vlasnik radio postaje, koji je određen prilikom izrade informacijskog sustava.

Korisnike sustava možemo podijeliti u pet grupa: vlasnik sustava, administrator, glazbeni urednik, registrirani korisnik i neregistrirani korisnik (posjetitelj).

Vlasnik je sustava odgovoran za definiranje administratora, te upisivanje kontakt podataka i podataka o radio postaji.

Administrator sustava, kao što je već rečeno, određuje glazbene urednike, upravlja zvučnim zapisima, te uređuje podatke o urednicima i registriranim korisnicima postaje.

Registrirani korisnici mogu sastavljati liste glazbenih želja.

Neregistriranim su korisnicima dostupne mogućnosti slušanja glazbe, besplatne registracije i kratak pregled informacija o postaji.

Sustav može imati jednog vlasnika, najviše deset administratora, te neograničen broj registriranih korisnika. Svi korisnici mogu istovremeno koristiti sustav.

Na početnoj se stranici web aplikacije nalaze osnovni podaci o radio postaji, područje za prijavu korisnika u sustav kao i za registraciju novih korisnika, te na najistaknutijem mjestu, glazbeni player uz koji se nalaze i podaci o trenutno sviranom glazbenom zapisu.

Prijavom u sustav, korisniku će biti dostupna i upravljačka stranica sa svim njemu dostupnim mogućnostima, ovisno o vrsti korisničkog računa. Svi korisnici kao ponuđenu mogućnost imaju upravljanje vlastitim računom: pregled i izmjena osobnih podataka, promjena lozinke te brisanje korisničkog računa. Obični korisnici imaju mogućnost stvaranja i pregleda svoje liste želja. Glazbeni urednik ima prikazanu mogućnost slanja zahtjeva za dodjelom termina za uređivanje, te stvaranje i uređivanje lista za reprodukciju u dodijeljenim mu terminima. Administratoru su na raspolaganju mogućnosti pregledavanja i uređivanja podataka o drugim korisnicima, upravljanja zvučnim zapisima, upravljanje glazbenim urednicima i njihovim terminima, te pregledavanje statistika korisnika i zapisa.

Liste korisničkih glazbenih želja sastoje se od maksimalno deset zapisa. Korisnik može po volji uređivati svoju listu želja, no ona je tada vidljiva samo njemu. Da bi se želje s nje učinile globalno dostupnima (urednicima i administratorima), korisnik mora svoju listu potvrditi, što može učiniti jednom svaka 24 sata (nakon jedne potvrde mora proći najmanje toliko vremena do iduće). Glazbeni urednici uvidom u globalnu listu želja dobivaju povratnu informaciju od korisnika o traženosti pojedinih zapisa, što im omogućava da se bolje prilagode interesima slušatelja.

Za uspjeh ovog projekta ključno je da glazbeni urednici redovito koriste sustav i kreiraju nove glazbene liste za reprodukciju. Ako neki glazbeni urednik ne kreira svoju listu na vrijeme, ponovit će se reprodukcija liste od njegovog prethodnog termina. Problem nastaje ako se to događa prečesto, ili više dana za redom; korisnici ne žele slušati iste pjesme iz dana u dan, te bi u tom slučaju sustav trebao reagirati na odgovarajući način. Također je

moguće da, propustom administratora, neki termin ostane neodijeljen, te će i tada sustav reagirati na odgovarajući način kako bi se spriječio privremeni prestanak emitiranja sadržaja.

Zbog mogućnosti da korisnik koristi slabiju internetsku vezu, a aplikacija uključuje prijenos i reprodukciju zvučnih zapisa preko iste, potrebno je da aplikacija bude što manja u pogledu količine podataka, kako bi se poboljšala brzina i kvaliteta usluge korisniku.

Osobni podaci svakog korisnika koji su pohranjeni u sustavu uključuju:

- ime
- prezime
- e-mail adresu
- godinu rođenja
- lozinku
- zanimanje

Svi se podaci naknadno mogu promijeniti. Ispravnost email adrese je bitna jer će se putem nje korisnici obavještavati o svim bitnim događajima i promjenama.

Postaja ima arhivu tonskih zapisa koji su dostupni za reprodukciju. Za svaki su zvučni zapis poznati sljedeći podaci:

- ime glazbenog zapisa
- ime izvođača
- putanja do datoteke zvučnog zapisa
- album
- nakladnik
- glazbeni žanr
- godina izdanja
- tip nosača

- trajanje zapisa
- frekvencija uzorkovanja
- format zapisa
- broj bitova kvantizacije

Svi ovi podaci bit će pohranjeni u bazi podataka na poslužitelju.

Web aplikacija bit će napisana u nekoliko trenutno popularnih web tehnologija, redom *Python Flask* za poslužiteljski dio aplikacije i komunikaciju s bazom podataka, *AngularJS* za klijentski dio aplikacije koji se izvršava u web pregledniku, te *JADE* i *SASS* za dizajn i strukturu web stranice.

Detalji sustava i njegove implementacije navedeni su u nastavku ovog dokumenta.

3. Rječnik pojmova

Flask – Framework za izradu web aplikacija u programskom jeziku Python, popularan zbog svoje jednostavnosti i lakoće korištenja, kao i male veličine

Peewee ORM – Python biblioteka koja olakšava dizajn i korištenje baze podataka

AngularJS – JavaScript framework za izradu web aplikacija, omogućuje njihov brz i intuitivan razvoj

REST – *Representational State Transfer* – stil arhitekture mrežnih aplikacija koja komunikaciju između klijenta i servera ostvaruje putem HTTP zahtjeva

SASS – *Syntactically Awesome Style Sheets* – proširenje CSS jezika koje dodaje brojne mogućnosti i bitno olakšava pisanje stilskih datoteka, kao i snalaženje u njima

JADE – strukturirani predlošci koji olakšavaju pisanje i održavanje HTML koda

MVC – *Model/View/Controller* – obrazac arhitekture programske podrške, razdvaja sustav na **modele** koji opisuju podatke i njihove operacije, **poglede** koji vrše interakciju s krajnjim korisnicima (prikaz sučelja i podataka), te **upravitelje** koje povezuju modele s pogledima

TypeScript – Nadgradnja programskog jezika JavaScript koja podržava statičke tipove podataka i potpuni OO model programiranja te olakšava samo pisanje koda

AJAX – *Asynchronous Javascript And XML* – vrsta komunikacije između web preglednika i poslužitelja koja omogućava asinkrone web aplikacije (promjena sadržaja bez potrebe za ponovnim učitavanjem stranice)

HTTP – *Hypertext Transfer Protocol* – internetski protokol koji se koristi na World Wide Webu za komunikaciju između web poslužitelja i web preglednika

JSON – *Javascript Object Notation* – format za prijenos podataka prikladan za korištenje pri prijenosu preko HTTP protokola

4. Funkcionalni zahtjevi

Dionici našeg sustava, odnosno osobe koje u njemu imaju interes, su:

- vlasnik sustava
- administrator
- glazbeni urednik
- registrirani korisnik
- posjetitelj (neregistrirani korisnik)
- korisnik (zajednički naziv za sve registrirane korisnike, urednike, administratore i vlasnika)

Aktorima se nazivaju oni koje vrše direktnu komunikaciju sa sustavom. To mogu biti inicijatori, koji pokreću procese u sustavu, ili sudionici, koji obavljaju zadane poslove.

Aktori i njihovi funkcionalni zahtjevi su:

- **vlasnik sustava**, inicijator:
 - može uređivati podatke o postaji
 - može postavljati administratore
 - može uređivati osobne podatke
 - može slušati program radio stanice
- **administrator**, inicijator:
 - može upravljati glazbenim urednicima
 - može upravljati zvučnim zapisima
 - može uređivati osobne podatke
 - može upravljati podacima drugih korisnika
 - može slušati program radio stanice

- **glazbeni urednik**, inicijator:
 - može slagati liste za izvođenje
 - može tražiti termine za reprodukciju
 - može uređivati osobne podatke
 - može slušati program radio stanice
- **registrirani korisnik**, inicijator:
 - može slagati listu želja
 - može uređivati osobne podatke
 - može slušati program radio stanice
- **korisnik**, inicijator
 - *apstraktni aktor, stvoren radi nasljeđivanja*
 - može uređivati osobne podatke
 - može slušati program radio stanice
- **posjetitelj**, inicijator:
 - može se registrirati
 - može slušati program radio stanice
- **baza podataka**, sudionik:
 - pohranjuje podatke o zvučnim zapisima
 - pohranjuje podatke o korisnicima
 - pohranjuje podatke o terminima izvođenja i zahtjevima za iste

4.1 Opis obrazaca uporabe

Napomena: Kako se radi o web aplikaciji, za sve obrasce uporabe nužan je preduvjet pristup Internetu

UC1 – RegistrirajNovogKorisnika

- **Glavni sudionik:** posjetitelj (neregistrirani korisnik)
- **Cilj:** stvoriti novi korisnički račun
- **Sudionici:** baza podataka
- **Preduvjeti:** nema ih
- **Rezultat:** stvoren je novi korisnički račun
- **Željeni scenarij:**
 1. Korisnik u odgovarajuća polja unosi svoje osobne podatke i email adresu te izabire lozinku
 2. Sustav provjerava točnost unesenih podataka, te koristi li se već odabrana email adresa
 3. Ako ne postoji, stvara se novi korisnički račun i na uneseni email se šalje pozdravna poruka s aktivacijskim linkom
 4. Klikom na taj link korisnik aktivira svoj račun te se sada može prijaviti u sustav
- **Mogući drugi scenariji:**
 1. Unesena email adresa se već koristi
 - Korisniku se dojavljuje greška i od njega se zahtjeva da odabere drugu email adresu

UC2 – PrijaviKorisnikaUSustav

- **Glavni sudionik:** korisnik
- **Cilj:** prijava u sustav
- **Sudionici:** baza podataka
- **Preduvjeti:** korisnik je registriran
- **Rezultat:** korisnik je prijavljen u sustav i sada su mu dostupne sve njegove mogućnosti
- **Željeni scenarij:**
 1. Korisnik unosi svoju email adresu i lozinku
 2. Sustav provjerava ispravnost unesenih podataka
 3. Ako uneseni podaci odgovaraju podacima korisničkog računa, korisnik se prijavljuje u sustav
- **Mogući drugi scenariji:**
 1. Uneseni su neispravni podaci
 - Korisniku se prikazuje odgovarajuća poruka o grešci i vraća ga se na prijavni obrazac

UC3 – UrediOsobnePodatke

- **Glavni sudionik:** korisnik
- **Cilj:** urediti osobne podatke
- **Sudionici:** baza podataka
- **Preduvjeti:** korisnik je prijavljen u sustav
- **Rezultat:** korisnik je izmijenio svoje osobne podatke
- **Željeni scenarij:**
 1. Korisniku se prikažu njegovi osobni podaci s mogućnošću promjene
 2. Korisnik mijenja neke od podataka te inicira pohranjivanje promjena
 3. Sustav vrši provjeru ispravnosti unesenih podataka

4. Ako su uneseni podaci ispravni, pohranjuju se u sustav i korisniku se prikazuje poruka o uspjehu

- **Mogući drugi scenariji:**

1. Neki od unesenih podataka su neispravni
 - Korisniku se prikazuje poruka o grešci i od njega se traži da unese ispravne podatke

UC4 – PromijeniLozinku

- **Glavni sudionik:** korisnik
- **Cilj:** promijeniti lozinku
- **Sudionici:** baza podataka
- **Preduvjeti:** korisnik je prijavljen u sustav
- **Rezultat:** korisnik je promijenio svoju lozinku
- **Željeni scenarij:**
 1. Korisnik unosi redom svoju staru lozinku, novoizabranu lozinku, te još jednom novoizabranu lozinku kako bi potvrdio promjenu
 2. Sustav provjerava ispravnost stare lozinke, kao i jednakost dviju unesenih novih lozinki
 3. Ako su svi uneseni podaci ispravni, sustav pohranjuje promijenjenu lozinku
- **Mogući drugi scenariji:**
 1. Nisu uneseni ispravni podaci
 - Korisniku se prikazuje odgovarajuća poruka o grešci i od njega se traži da unese valjane podatke

UC5 – IzbrišiKorisničkiRačun

- **Glavni sudionik:** korisnik
- **Cilj:** izbrisati korisnički račun

- **Sudionici:** baza podataka
- **Preduvjeti:** korisnik je prijavljen u sustav, korisnik nije vlasnik sustava
- **Rezultat:** korisnički račun više ne postoji
- **Željeni scenarij:**
 1. Korisnika unosi svoju lozinku (sigurnosna mjera)
 2. Korisnik inicira brisanje korisničkog računa
 3. Ako je lozinka ispravna, korisnički se račun zauvijek briše
- **Mogući drugi scenariji:**
 1. Korisnik nije unio ispravnu lozinku
 - U tom slučaju prikazuje mu se odgovarajuća poruka o grešci i od njega se traži da unese ispravnu lozinku

UC6 – SastaviListuŽelja

- **Glavni sudionik:** registrirani korisnik
- **Cilj:** izrada liste glazbenih želja
- **Sudionici:** baza podataka
- **Preduvjeti:** korisnik je prijavljen u sustav
- **Rezultat:** korisnik je sastavio i pohranio listu želja
- **Željeni scenarij:**
 1. Korisnik pregledava svoju listu želja (ako već postoji, ako ne, onda je prazna) te u nju unosi izmjene (dodaje ili briše pjesme, pri čemu se može služiti pretraživanjem pjesama)
 2. Kada je napravio sve planirane izmjene, inicira pohranjivanje
 3. Sustav pohranjuje korisnikovu listu želja

UC7 – PotvrdiListuŽelja

- **Glavni sudionik:** registrirani korisnik

- **Cilj:** potvrditi listu želja i time ju učiniti globalno dostupnom
- **Sudionici:** baza podataka
- **Preduvjeti:** korisnik je prijavljen u sustav, korisnik u protekla 24 sata već nije potvrđivao listu želja
- **Rezultat:** lista želja je potvrđena, želje za pjesmama na listi su sada vidljive glazbenim urednicima i administratorima
- **Željeni scenarij:**
 1. Korisnik pregledava svoju listu želja
 2. Ako je zadovoljan s njome, potvrđuje ju
 3. Sustav pohranjuje korisnikove želje u globalnu listu

UC8 – Zatraži Termin Za Reprodukciju

- **Glavni sudionik:** glazbeni urednik
- **Cilj:** zatražiti dodjelu termina (jednog ili više) za reprodukciju od administratora
- **Sudionici:** baza podataka
- **Preduvjeti:** glazbeni urednik je prijavljen u sustav
- **Rezultat:** zahtjev za dodjelom termina je uspješno pohranjen u sustav
- **Željeni scenarij:**
 1. Glazbeni urednik pregledava kalendar sa označenim slobodnim terminima
 2. Urednik odabire termin(e) koji mu odgovara(ju) i šalje zahtjev za njima
 3. Sustav pohranjuje urednikov zahtjev

UC9 – Sastavi Listu Za Reprodukciju

- **Glavni sudionik:** glazbeni urednik
- **Cilj:** sastaviti listu pjesama za reprodukciju za dani termin (trajanje 1 sat)
- **Sudionici:** baza podataka

- **Preduvjeti:** glazbeni urednik je prijavljen u sustav, dodijeljen mu je termin, do trenutka emitiranja ima više od 24 sata vremena
- **Rezultat:** lista zapisa za dani termin je sastavljena
- **Željeni scenarij:**
 1. Urednik pregledava i pretražuje pjesme, uzimajući u obzir korisničke želje
 2. Urednik odabire pjesme koje će se reproducirati u danom terminu
 3. Urednik inicira pohranu liste
 4. Sustav ispituje ispravnost sastavljene liste (trajanje najmanje 1 sat, zadnja pjesma ne počinje unutar 15 sekundi od kraja termina)
 5. Ako su uvjeti zadovoljeni, lista se pohranjuje u sustav
- **Mogući drugi scenariji:**
 1. Nisu zadovoljeni uvjeti za listu
 - Korisniku se prikazuje odgovarajuća poruka o pogrešci i od njega se traži da sastavi ispravnu listu

UC10 – DodajZvučniZapis

- **Glavni sudionik:** administrator
- **Cilj:** u sustav dodati novi zvučni zapis
- **Sudionici:** baza podataka
- **Preduvjeti:** administrator je prijavljen u sustav
- **Rezultat:** u sustav je dodan novi glazbeni zapis sa svim potrebnim podacima
- **Željeni scenarij:**
 1. Administrator unosi sve bitne podatke o zvučnom zapisu
 2. Administrator prilaže datoteku zvučnog zapisa
 3. Sustav provjerava ispravnost unesenih podataka
 4. Ako su podaci ispravni, pohranjuju se u sustav zajedno sa samom datotekom zapisa

- **Mogući drugi scenariji:**

1. Uneseni su neispravni podaci

- Prikazuje se odgovarajuća poruka o grešci i od administratora se traži da unese ispravne podatke

1. Slanje datoteke zvučnog zapisa na sustav nije uspjelo

- Prikazuje se odgovarajuća poruka o pogrešci i traži se ponovno obavljanje akcije

UC11 – UrediZvučniZapis

- **Glavni sudionik:** administrator

- **Cilj:** urediti podatke o zvučnom zapisu

- **Sudionici:** baza podataka

- **Preduvjeti:** administrator je prijavljen u sustav

- **Rezultat:** podaci o zvučnom zapisu su izmijenjeni

- **Željeni scenarij:**

1. Administrator odabire zvučni zapis kojeg želi izmijeniti
2. Administrator pregledava podatke o zvučnom zapisu i po želji ih mijenja
3. Administrator inicira pohranu podataka
4. Sustav ispituje ispravnost unesenih podataka
5. Ako su podaci ispravni, pohranjuju se u sustav

- **Mogući drugi scenariji:**

1. Uneseni su neispravni podaci

- Prikazuje se odgovarajuća poruka o grešci, te se od administratora traži unos ispravnih podataka

UC12 – ObrišiZvučniZapis

- **Glavni sudionik:** administrator
- **Cilj:** obrisati zvučni zapis
- **Sudionici:** baza podataka
- **Preduvjeti:** administrator je prijavljen u sustav
- **Rezultat:** zvučni zapis je zauvijek izbrisan
- **Željeni scenarij:**
 1. Administrator odabire zvučni zapis kojeg želi izbrisati
 2. Administrator inicira brisanje zvučnog zapisa
 3. Sustav briše zvučni zapis

UC13 – UrediPodatkeKorisnika

- **Glavni sudionik:** administrator
- **Cilj:** urediti podatke korisnika
- **Sudionici:** baza podataka
- **Preduvjeti:** administrator je prijavljen u sustav, korisnik čiji se podaci uređuju nije vlasnik niti administrator
- **Rezultat:** korisnikovi podaci su uređeni
- **Željeni scenarij:**
 1. Administrator odabire korisnika kojem želi izmijeniti podatke
 2. Administrator pregledava podatke korisnika i po želji unosi promjene
 3. Administrator inicira spremanje promjena
 4. Sustav pohranjuje promjene

UC14 – PostaviGlazbenogUrednika

- **Glavni sudionik:** administrator
- **Cilj:** postaviti novog urednika

- **Sudionici:** baza podataka
- **Preduvjeti:** administrator mora biti prijavljen u sustav
- **Rezultat:** postavljen je novi urednik
- **Željeni scenarij:**
 1. Administrator pregledava popis korisnika
 2. Odabire jednog od korisnika i postavlja ga za glazbenog urednika
 3. Sustav bilježi promjene i mijenja tip korisničkog računa odabranog korisnika

UC15 – UkloniGlazbenogUrednika

- **Glavni sudionik:** administrator
- **Cilj:** ukloniti glazbenog urednika
- **Sudionici:** baza podataka
- **Preduvjeti:** administrator mora biti prijavljen u sustav
- **Rezultat:** glazbenom uredniku oduzima se urednički status
- **Željeni scenarij:**
 1. Administrator određuje urednika kojem želi oduzeti uredničke ovlasti
 2. Sustav bilježi promjene i mijenja tip korisničkog računa odabranog urednika

UC16 – OdlučiOZahtjevuZaTerminom

- **Glavni sudionik:** administrator
- **Cilj:** odlučiti o uredničkom zahtjevu za terminom
- **Sudionici:** baza podataka
- **Preduvjeti:** administrator mora biti prijavljen u sustav
- **Rezultat:** urednički zahtjev je ili prihvaćen i time je taj termin dodijeljen tom uredniku, ili je odbijen
- **Željeni scenarij:**
 1. Administrator odlučuje o prihvaćanju ili odbijanju zahtjeva

2. Ako je zahtjev prihvaćen, taj se termin dodjeljuje uredniku, što se bilježi u sustavu

UC17 – PrikažiStatistiku

- **Glavni sudionik:** administrator
- **Cilj:** pregledati statistike o radu postaje
- **Sudionici:** baza podataka
- **Preduvjeti:** administrator je prijavljen u sustav
- **Rezultat:** administrator je dobio uvid u statistike radio postaje
- **Željeni scenarij:**
 1. Administrator odabire jednu od ponuđenih statistika
 2. Administratoru se prikazuje odabrana statistika

UC18 –PostaviAdministratora

- **Glavni sudionik:** vlasnik postaje
- **Cilj:** postaviti novog administratora
- **Sudionici:** baza podataka
- **Preduvjeti:** ne smije biti postavljeno više od deset administratora, vlasnik je prijavljen u sustav
- **Rezultat:** postavljen je novi administrator
- **Željeni scenarij:**
 1. Vlasnik odabire jednog od korisnika i dodjeljuje mu administratorske ovlasti
 2. Sustav bilježi promjene i mijenja tip korisničkog računa odabranog korisnika
- **Mogući drugi scenariji:**
 1. U sustavu već postoji 10 administratora
 - Akcija se ne dozvoljava, ispisuje se odgovarajuća poruka o grešci

UC19 – UkloniAdministratora

- **Glavni sudionik:** vlasnik postaje
- **Cilj:** ukloniti administratora
- **Sudionici:** baza podataka
- **Preduvjeti:** vlasnik postaje je prijavljen u sustav
- **Rezultat:** odabranom korisniku ukinute su administratorske ovlasti
- **Željeni scenarij:**
 1. Vlasnik postaje odabire jednog od administratora i uklanja mu administratorske ovlasti
 2. Sustav bilježi promjene i mijenja tip korisničkog računa odabranog korisnika

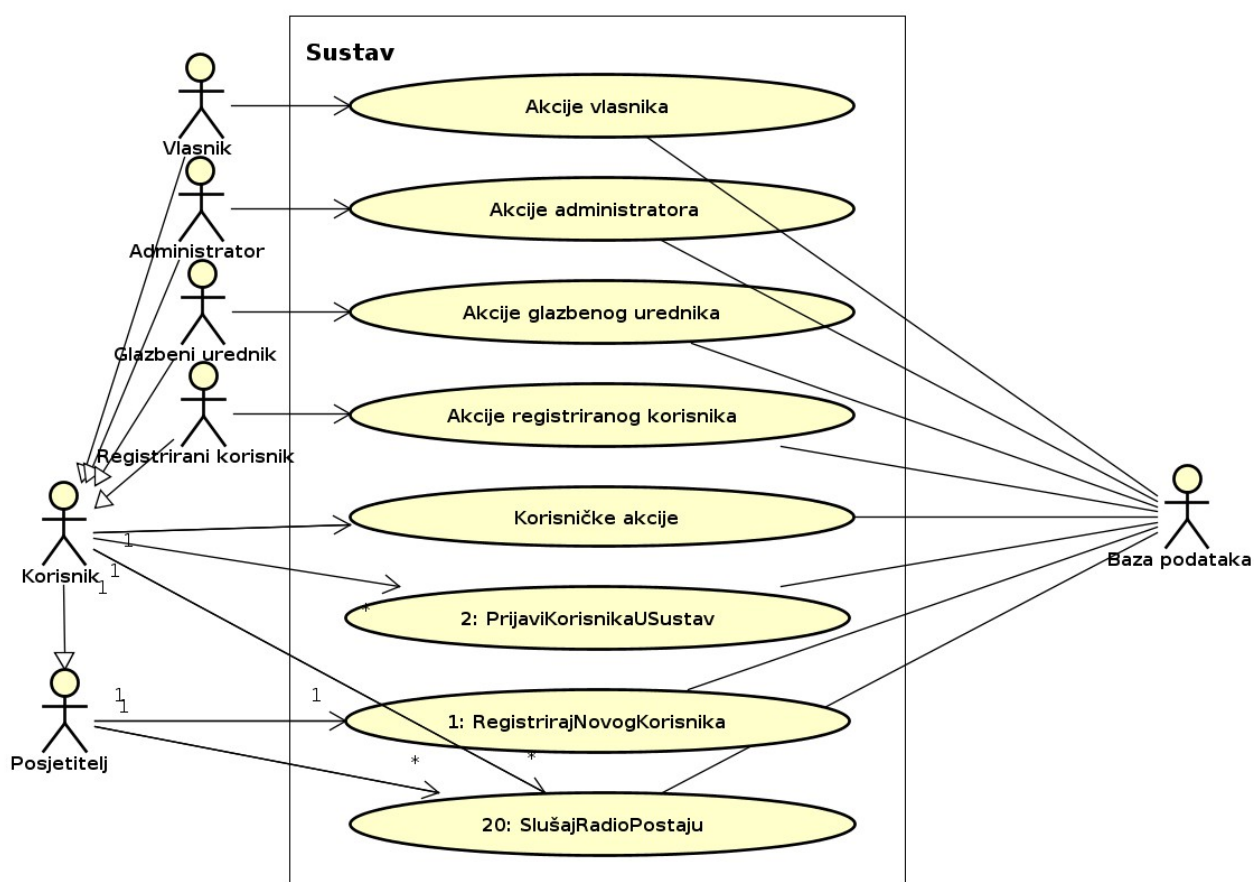
UC20 – UrediPodatkeOPostaji

- **Glavni sudionik:** vlasnik postaje
- **Cilj:** unjeti ili urediti podatke
- **Sudionici:** baza podataka
- **Preduvjeti:** vlasnik je prijavljen u sustav
- **Rezultat:** uneseni su novi podaci o postaji
- **Željeni scenarij:**
 1. Vlasniku se prikazuju podaci o radio postaji s mogućnošću promjena
 2. Vlasnik unosi izmjene i inicira spremanje
 3. Baza podataka pohranjuje i čuva nove podatke

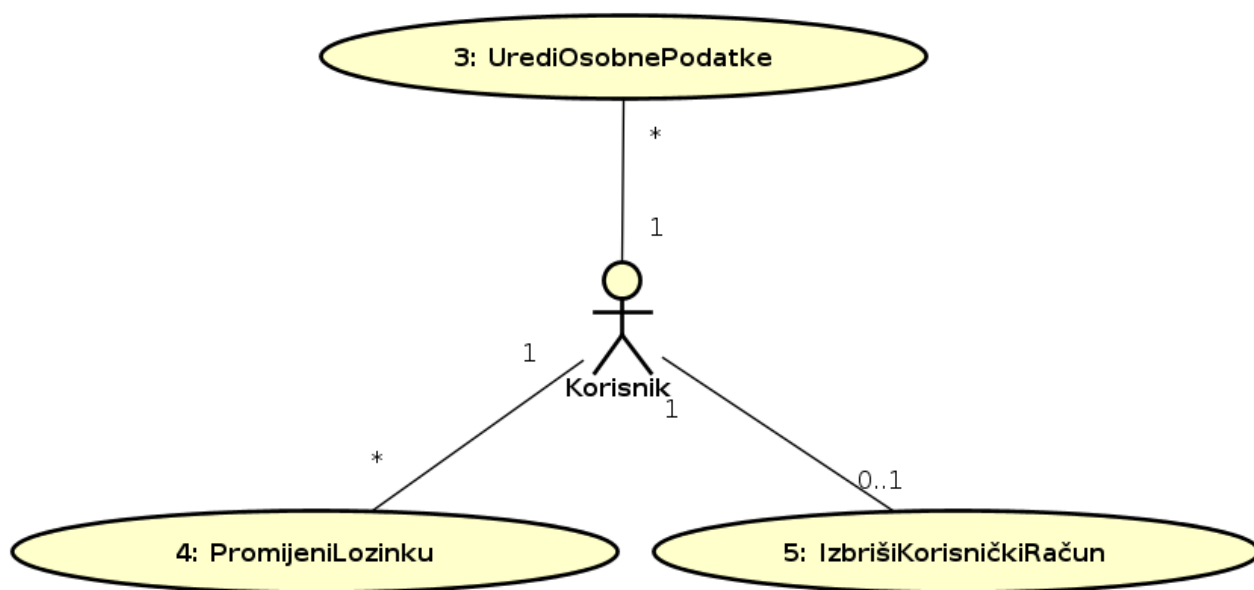
UC21 – SlušajRadioPostaju

- **Glavni sudionik:** korisnik
- **Cilj:** slušati program radio postaje
- **Sudionici:** baza podataka
- **Preduvjeti:** nema

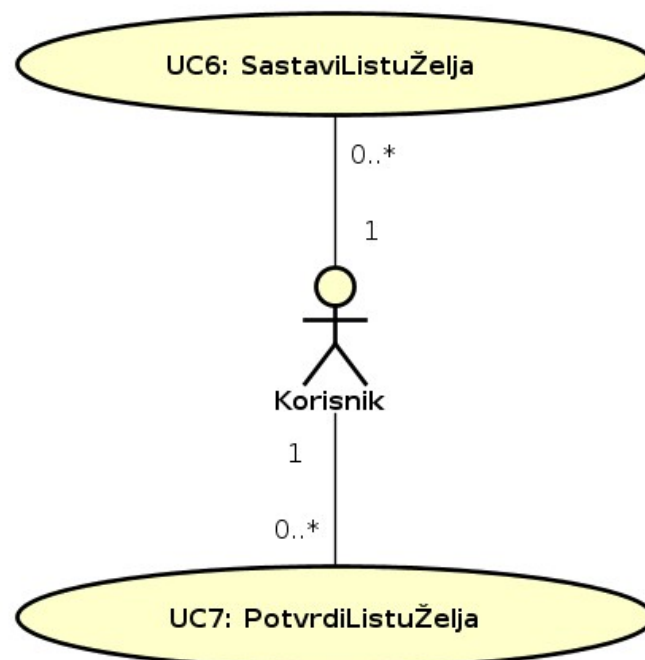
- **Rezultat:** korisnik uživa u programu radio postaje
- **Željeni scenarij:**
 1. Korisnik otvara web stranicu radio postaje
 2. Pokreće glazbeni player na početnoj stranici i započinje slušati program radio postaje



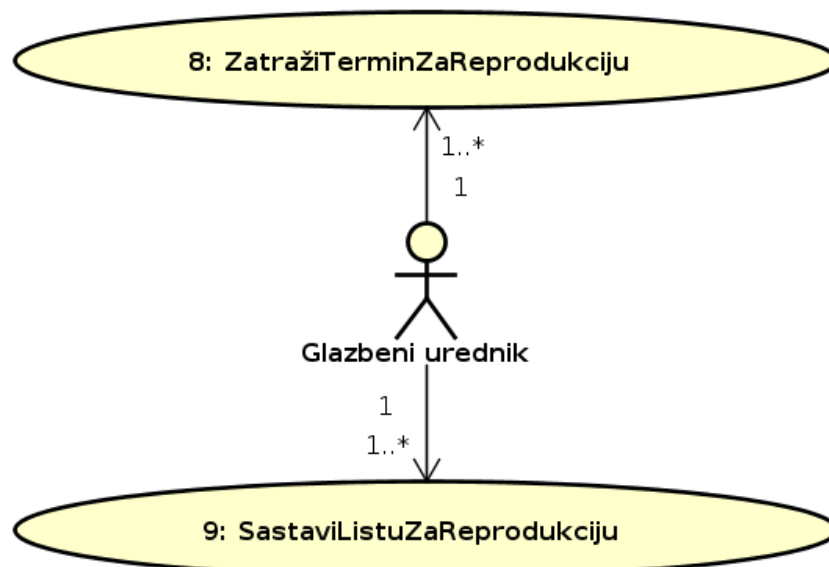
Slika 1: Dijagram obrazaca uporabe



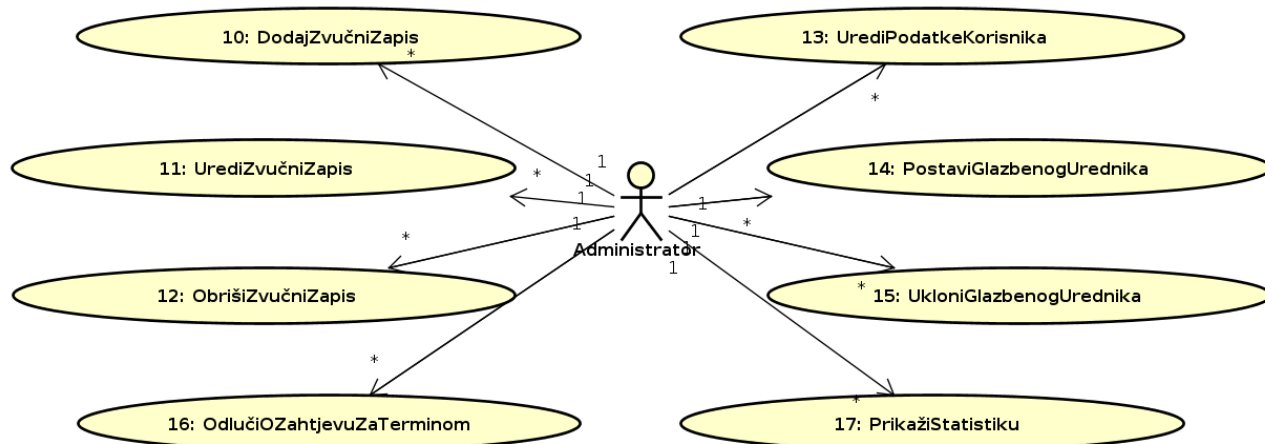
Slika 2: Dijagram obrazaca korisničkih akcija



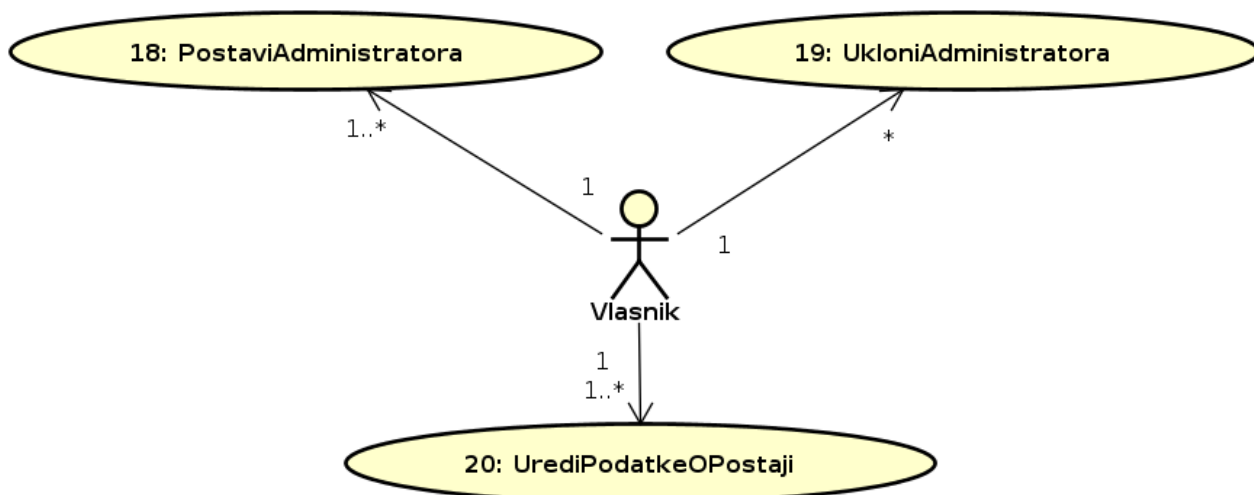
Slika 3: Dijagram obrazaca akcija registriranog korisnika



Slika 4: Dijagram obrazaca akcija glazbenog urednika



Slika 5: Dijagram obrazaca akcija administratora

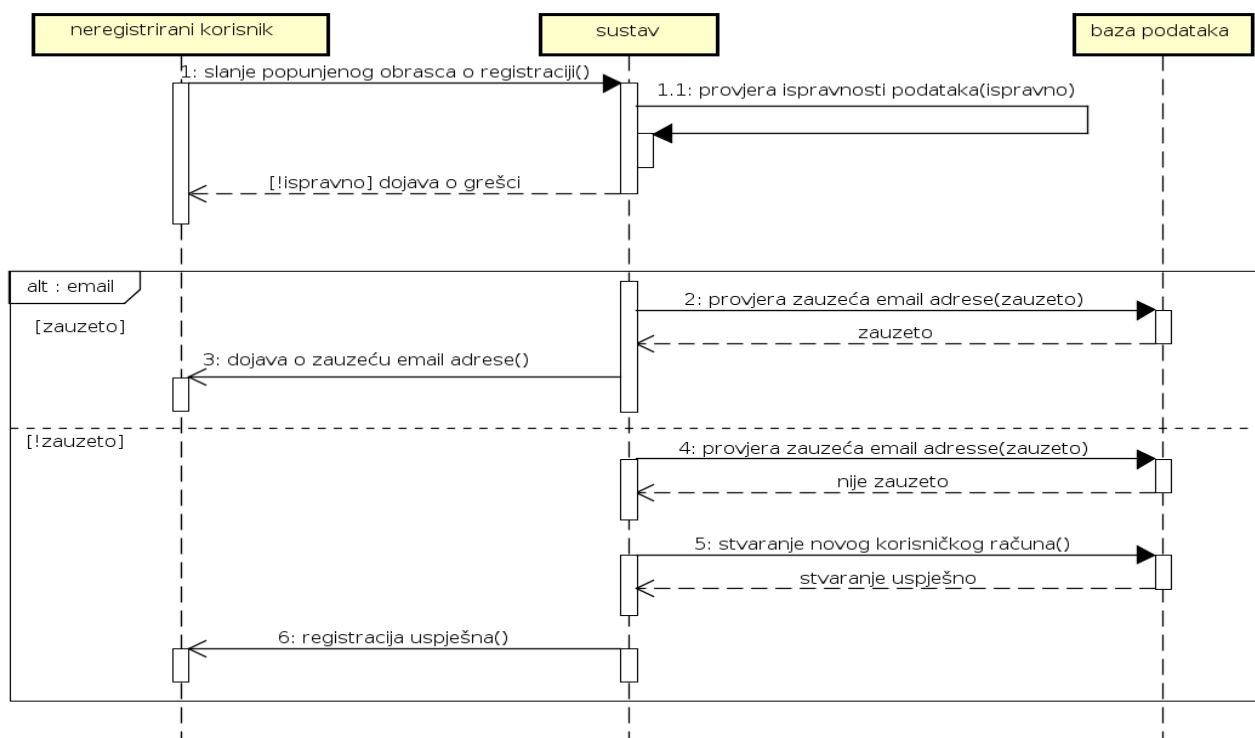


Slika 6: Dijagram obrazaca akcija vlasnika

4.2 Sekvencijski dijagrami

UC1: RegistrirajNovogKorisnika

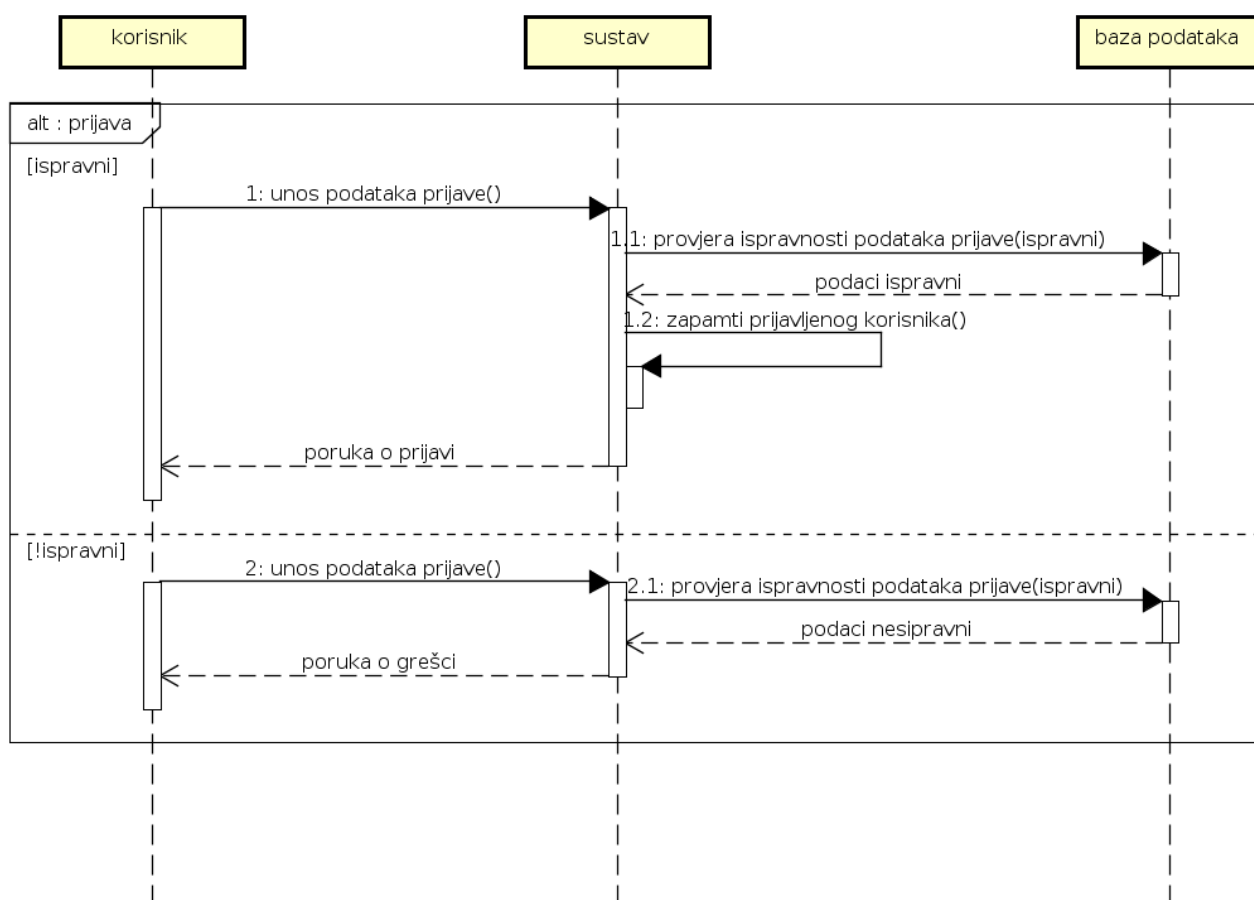
Neregistrirani korisnik želi se registrirati. Ispunjava obrazac za registraciju i šalje ga sustavu. Sustav provjerava ispravnost unesenih podataka. Sustav potom provjerava dostupnost odabrane email adrese. Ako bilo koja od ove dvije provjere ne uspije, korisniku se prikazuje poruka o pogrešci i on mora ponovno ispuniti obrazac. Inače se stvara novi korisnički račun i podaci o njemu pohranjuju se u bazu podataka. Korisniku se na kraju dojavljuje poruka o uspješnoj registraciji.



Slika 7: Sekvencijski dijagram za **UC1: RegistrirajNovogKorisnika**

UC2: PrijaviKorisnikaUSustav

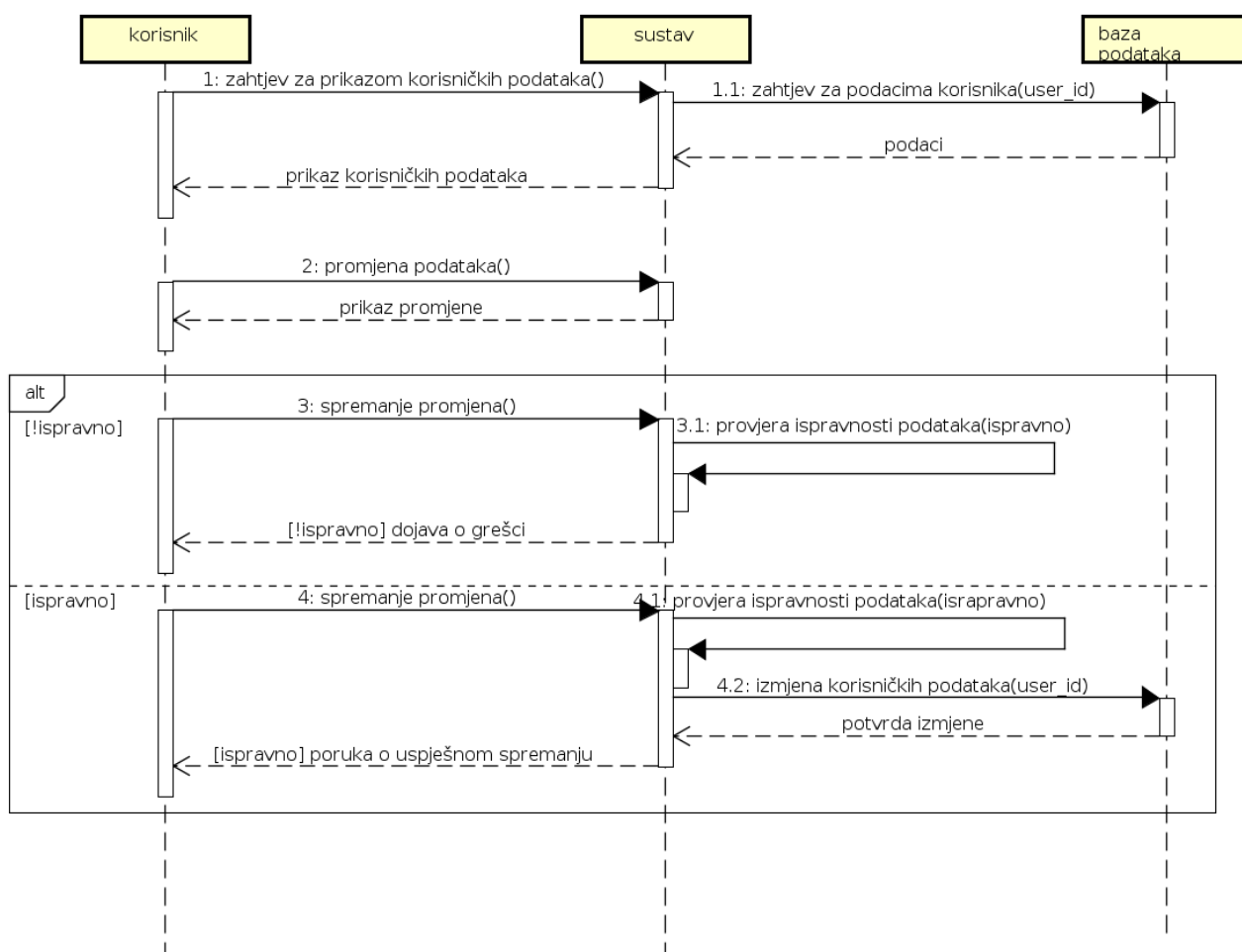
Korisnik se želi prijaviti u sustav. Ispunjava obrazac za prijavu i šalje ga sustavu. Sustav ispituje ispravnost unesenih podataka (postojanje korisnika s tom email adresom i lozinkom) upitom u bazu podataka, te ako su prijavni podaci ispravni, sustav bilježi prijavu korisnika te vraća poruku o uspješnoj prijavi. Ako podaci nisu ispravni, korisniku se prikazuje odgovarajuća poruka o pogrešci.



Slika 8: Sekvencijski dijagram za UC2: PrijavaKorisnikaUSustav

UC3: UrediOsobnePodatke

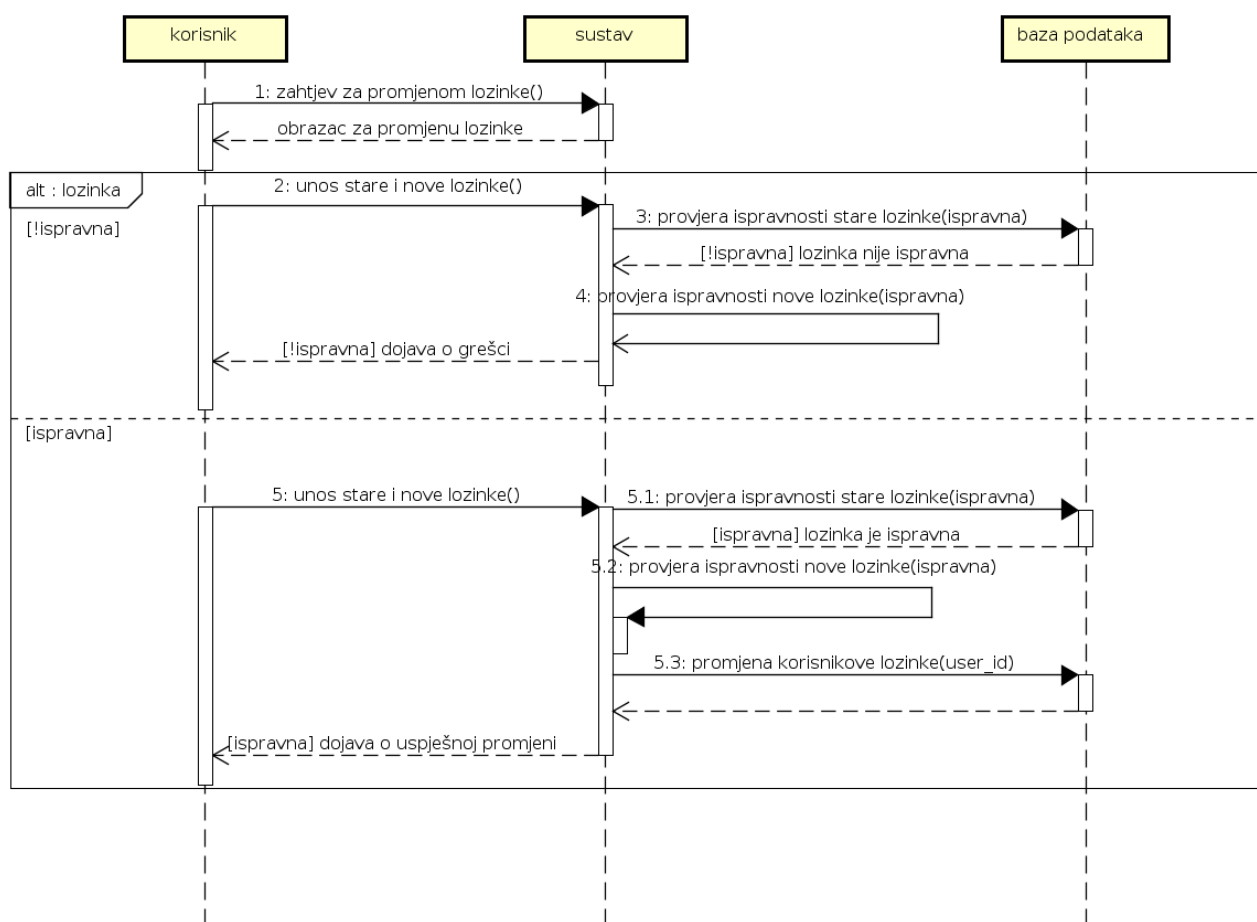
Korisnik želi urediti svoje osobne podatke. Korisnik odabere tu mogućnost iz popisa. Sustav mu prikaže njegove korisničke podatke koje je dohvatio iz baze podataka. Korisnik unosi promjene u te podatke dok god nije zadovoljan s njima. Potom inicira spremanje podataka, koji se šalju sustavu. Sustav provjerava ispravnost unesenih podataka, i ako su oni ispravni, pohranjuje promjene u bazu podataka te dojavljuje korisniku da su promjene uspješno obavljene. Inače mu prikazuje poruku o pogrešci.



Slika 9: Sekvencijski dijagram za **UC3: UrediOsobnePodatke**

UC4: PromijeniLozinku

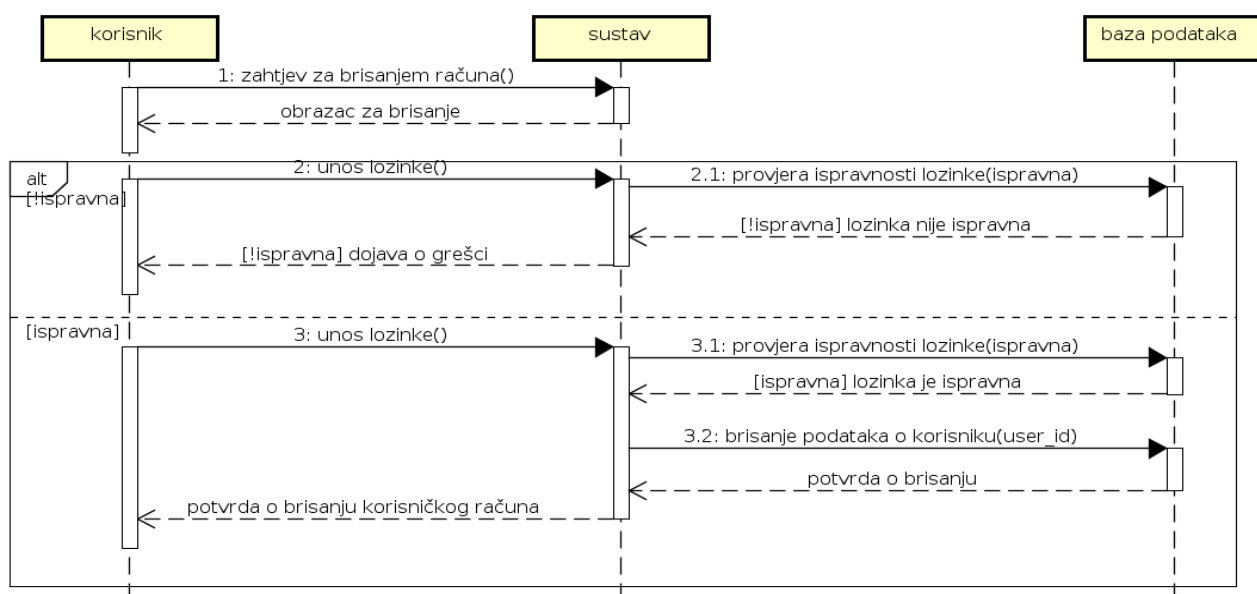
Korisnik želi promijeniti svoju lozinku. Odabire tu mogućnost iz popisa. Sustav mu prikazuje obrazac za promjenom korisničke lozinke kojeg korisnik ispunjava. Kada ga je ispunio, šalje ga na sustav koji potom vrši provjere ispravnosti unesenih podataka, stare i nove lozinke. Ako su podaci ispravni, promjene se pohranjuju u bazu podataka i korisniku se prikazuje poruka o uspjehu, a ako nisu, dojavljuje mu se pogreška.



Slika 10: Sekvencijski dijagram za UC4: PromijeniLozinku

UC5: IzbrišiKorisničkiRačun

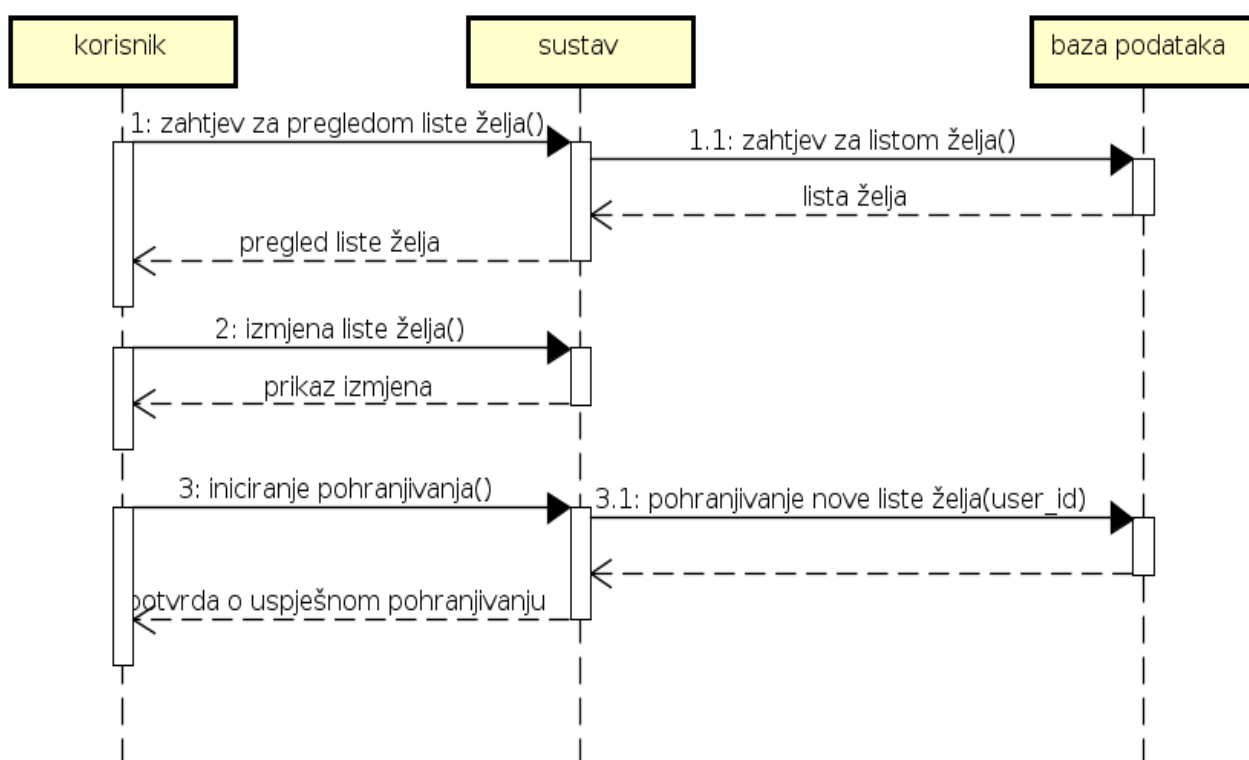
Korisnik želi izbrisati svoj korisnički račun. Odabire tu opciju s popisa ponuđenih, na što mu sustav prikazuje obrazac za brisanje računa. Korisnik unosi svoju lozinku u obrazac i potvrđuje da želi obrisati svoj račun. Sustav ispituje ispravnost unesene lozinke i ako je ona ispravna, iz baze podataka brišu se podaci o korisniku, te mu se prikazuje odgovarajuća poruka. Ako lozinka nije ispravna, korisniku se prikazuje poruka o pogrešci.



Slika 11: Sekvencijski dijagram za **UC5: IzbrišiKorisničkiRačun**

UC6: SastaviListuŽelja

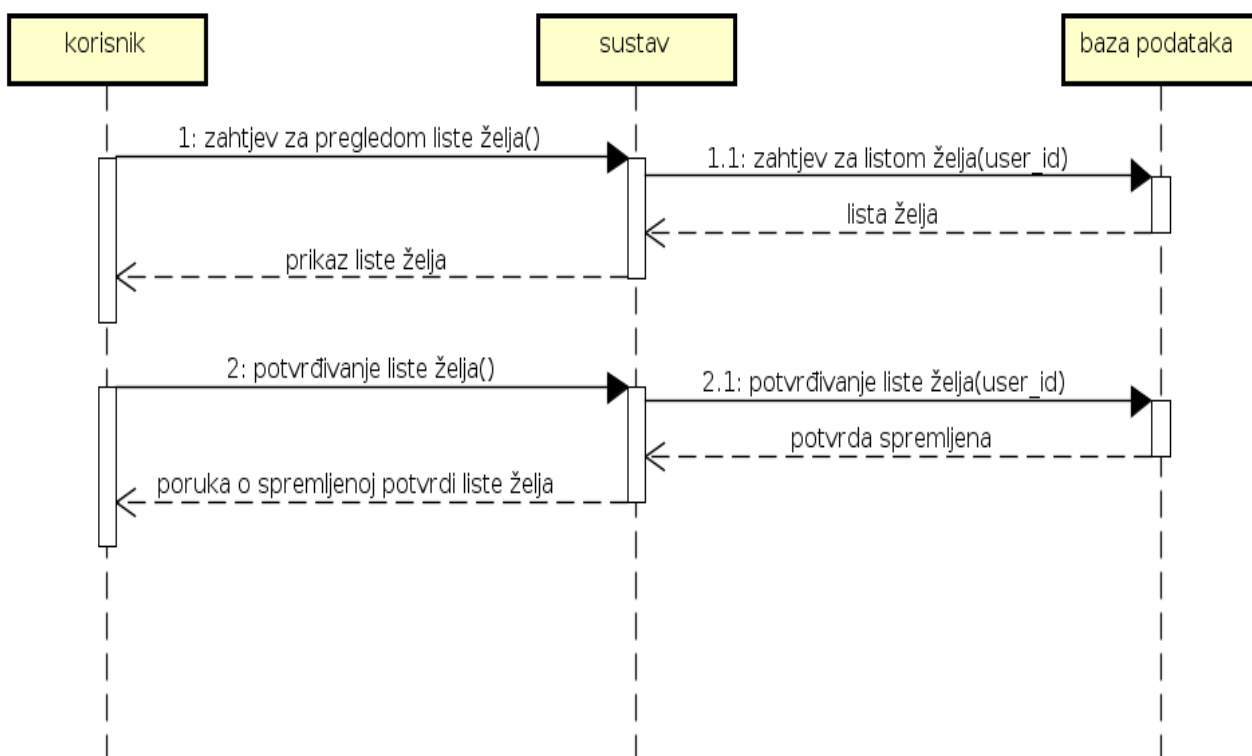
Korisnik želi sastaviti listu želja. Odabire tu mogućnost iz popisa ponuđenih, na što mu sustav prikazuje njegovu dosadašnju listu želja (ako postoji, ako ne, prazna je), s mogućnostima izmjena. Korisnik unosi izmjene i nakon što s njima završi, inicira pohranjivanje liste. Sustav prima novu listu od korisnika, pohranjuje ju u bazu podataka te korisniku prikazuje poruku o uspjehu.



Slika 12: Sekvencijski dijagram za **UC6: SastaviListuŽelja**

UC7: PotvrdiListuŽelja

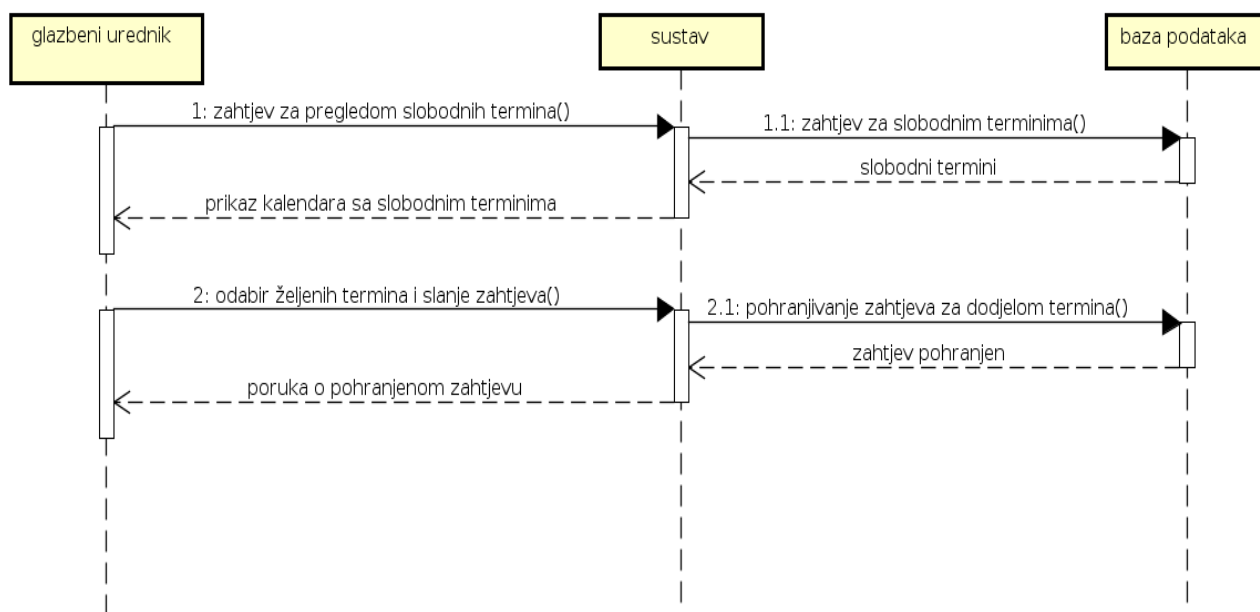
Korisnik želi potvrditi svoju listu želja, čime će želje s nje učiniti globalno dostupnom. Odabire tu mogućnost iz popisa opcija, na što mu sustav prikazuje njegovu spremljenu listu želja. Ako je korisnik zadovoljan njome, inicira njeno potvrđivanje. Sustav tada u bazu podataka sprema informaciju o potvrđivanju, te korisniku dojavljuje poruku o uspjehu.



Slika 13: Sekvencijski dijagram za **UC7: PotvrdiListuŽelja**

UC8: ZatražiTerminZaReprodukciju

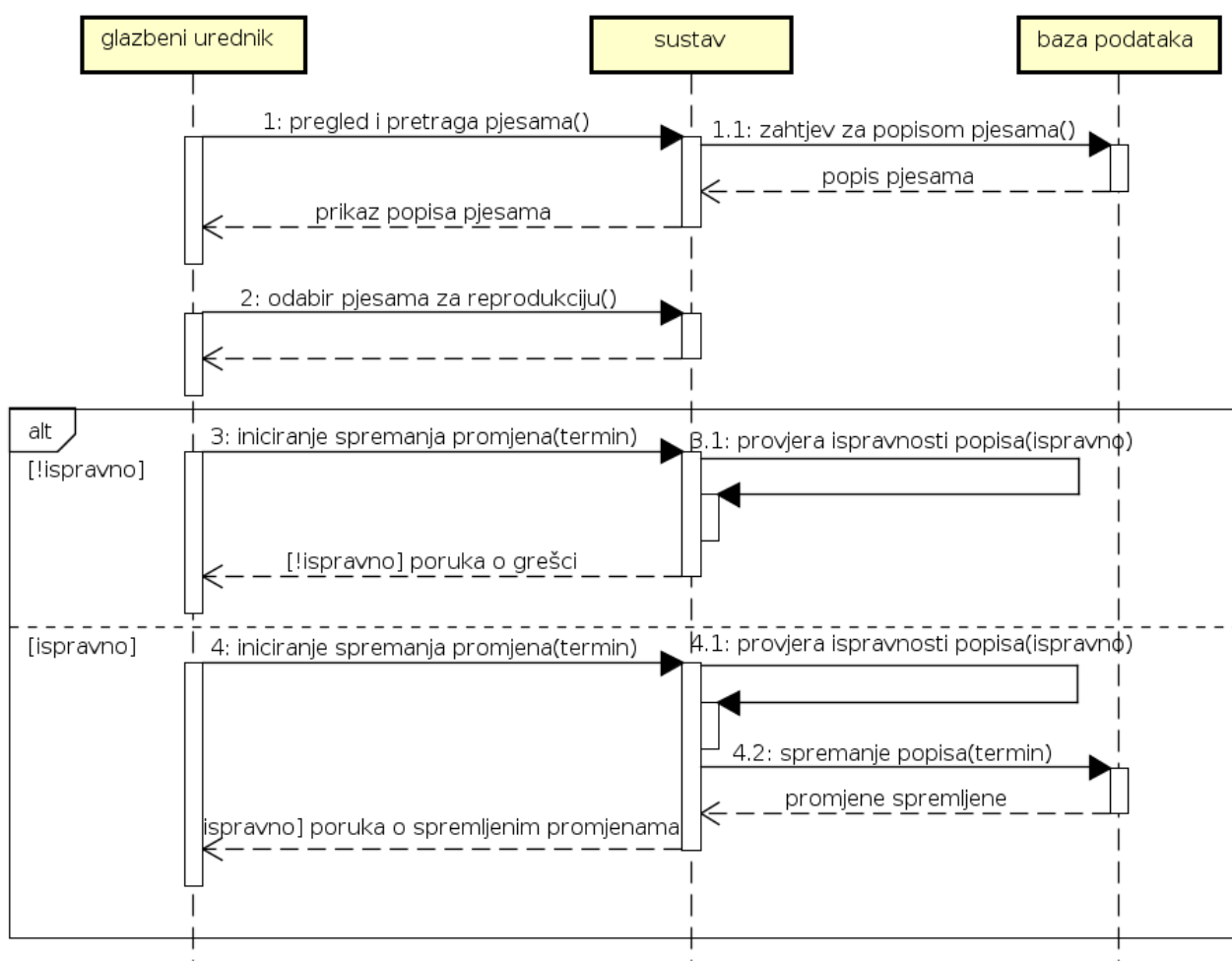
Glazbeni urednik želi zatražiti dodjelu novog/ih termina. Odabire tu mogućnost iz popisa ponuđenih, na što mu sustav prikazuje kalendar na kojemu su označeni slobodni termini. Korisnik izabire neke od ponuđenih termina i sustavu šalje zahtjev za njima. Sustav prima zahtjev, pohranjuje ga u bazu podataka te obavještava korisnika o uspješnoj pohrani.



Slika 14: Sekvencijski dijagram za **UC8: ZatražiTerminZaReprodukciju**

UC9: SastaviListuZaReprodukciju

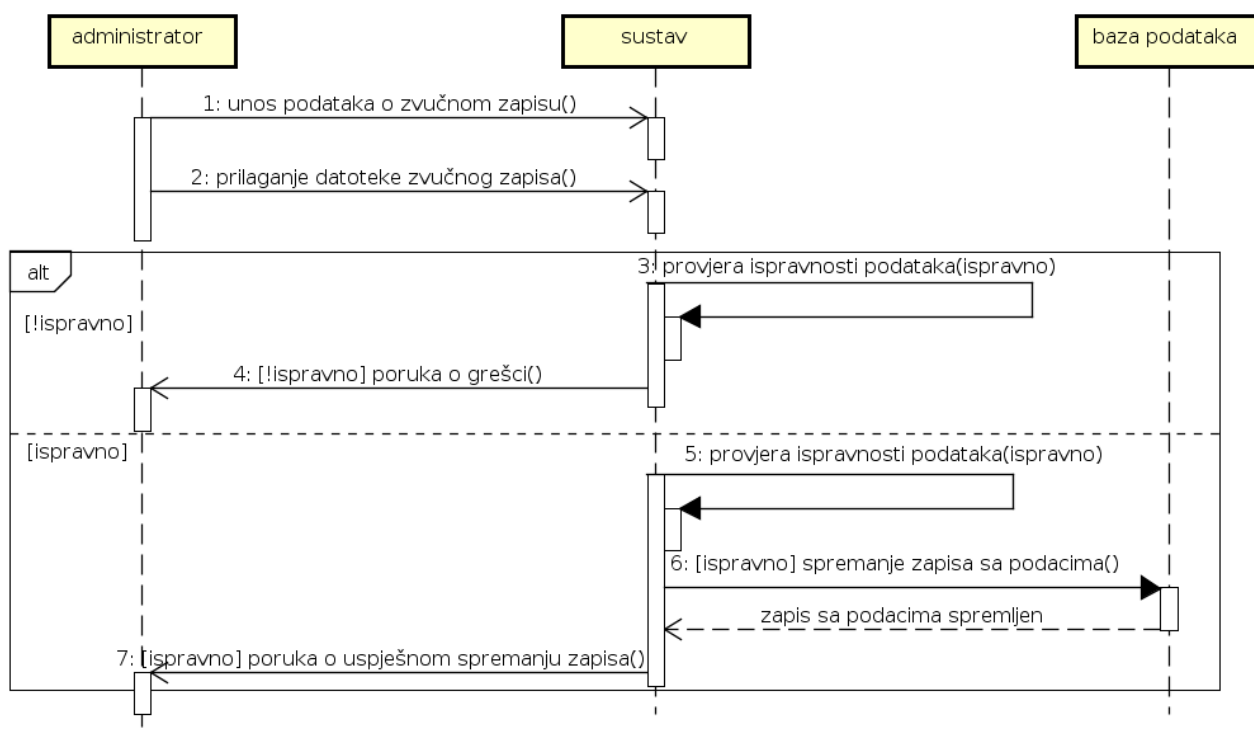
Glazbeni urednik želi sastaviti listu za reprodukciju za neki od svojih termina. Urednik odabere tu mogućnost iz popisa. Sustav mu omogućuje pregledavanje i pretraživanje zapisa, pomoću kojih urednik sastavlja svoju listu, uzimajući u obzir korisničke želje. Kada je zadovoljan listom urednik inicira pohranu liste. Sustav ispituje ispravnost sastavljene liste, i ako su uvjeti zadovoljeni lista se pohranjuje u sustav. Inače se uredniku prikazuje odgovarajuća poruka o pogrešci i od njega se traži da sastavi ispravnu listu.



Slika 15: Sekvencijski dijagram za **UC9: SastaviListuZaReprodukciju**

UC10: DodajZvučniZapis

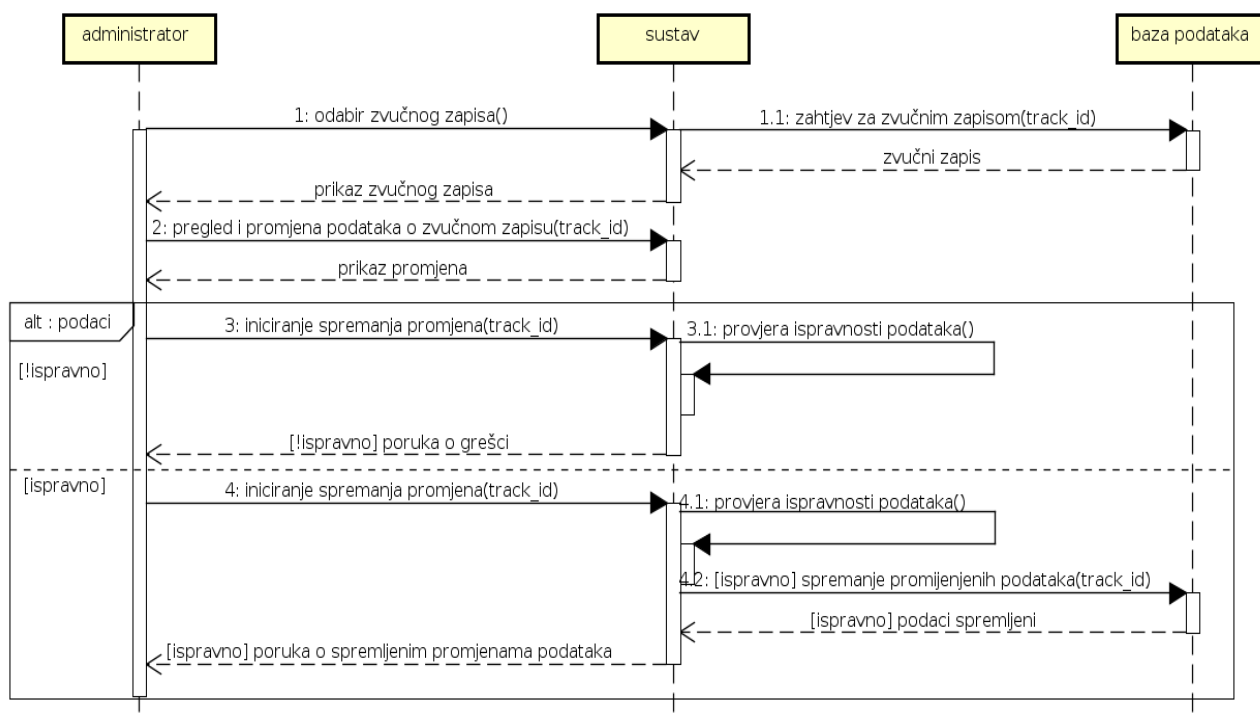
Administrator želi dodati zvučni zapis. Administrator unosi sve bitne podatke o zvučnom zapisu i prilaže datoteku zvučnog zapisa. Sustav provjerava ispravnost unesenih podataka i ako su podaci ispravni pohranjuje ih u bazu podataka, zajedno sa samom datotekom zapisa. Inače sustav prikazuje administratoru odgovarajuću poruku o grešci, te od administratora traži unos ispravnih podataka.



Slika 16: Sekvencijski dijagram za UC10: DodajZvučniZapis

UC11: UrediZvučniZapis

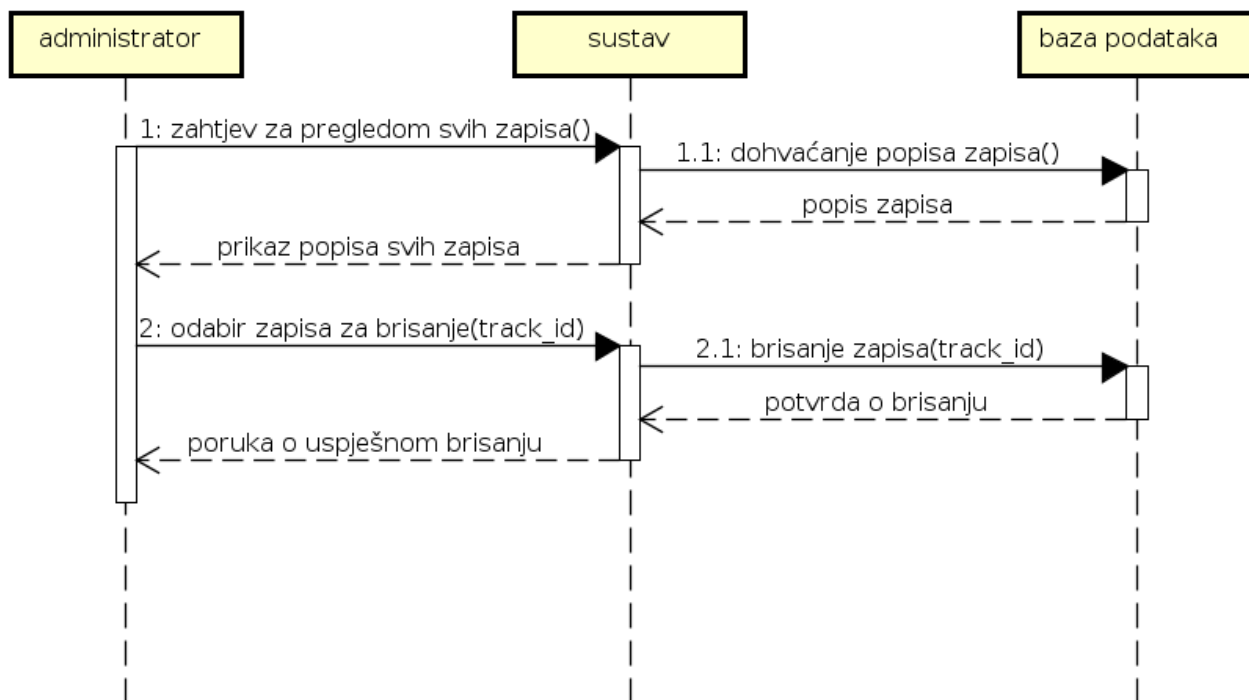
Administrator želi urediti zvučni zapis. Administrator odabere tu mogućnost iz popisa. Sustav mu prikaže listu svih zvučnih zapisa koje je dohvatio iz baze podataka. Administrator odabire zvučni zapis kojeg želi izmijeniti. Administrator pregledava podatke o zvučnom zapisu i po želji ih mijenja. Kada je zadovoljan promjenama administrator inicira promjenu podataka, koji se šalju u sustav. Sustav ispituje ispravnost unesenih podataka, i ako su podaci ispravni pohranjuje ih u bazu podataka. Inače sustav prikazuje administratoru odgovarajuću poruku o grešci, te od administratora traži unos ispravnih podataka.



Slika 17: Sekvencijski dijagram za UC11: UrediZvučniZapis

UC12: ObrišiZvučniZapis

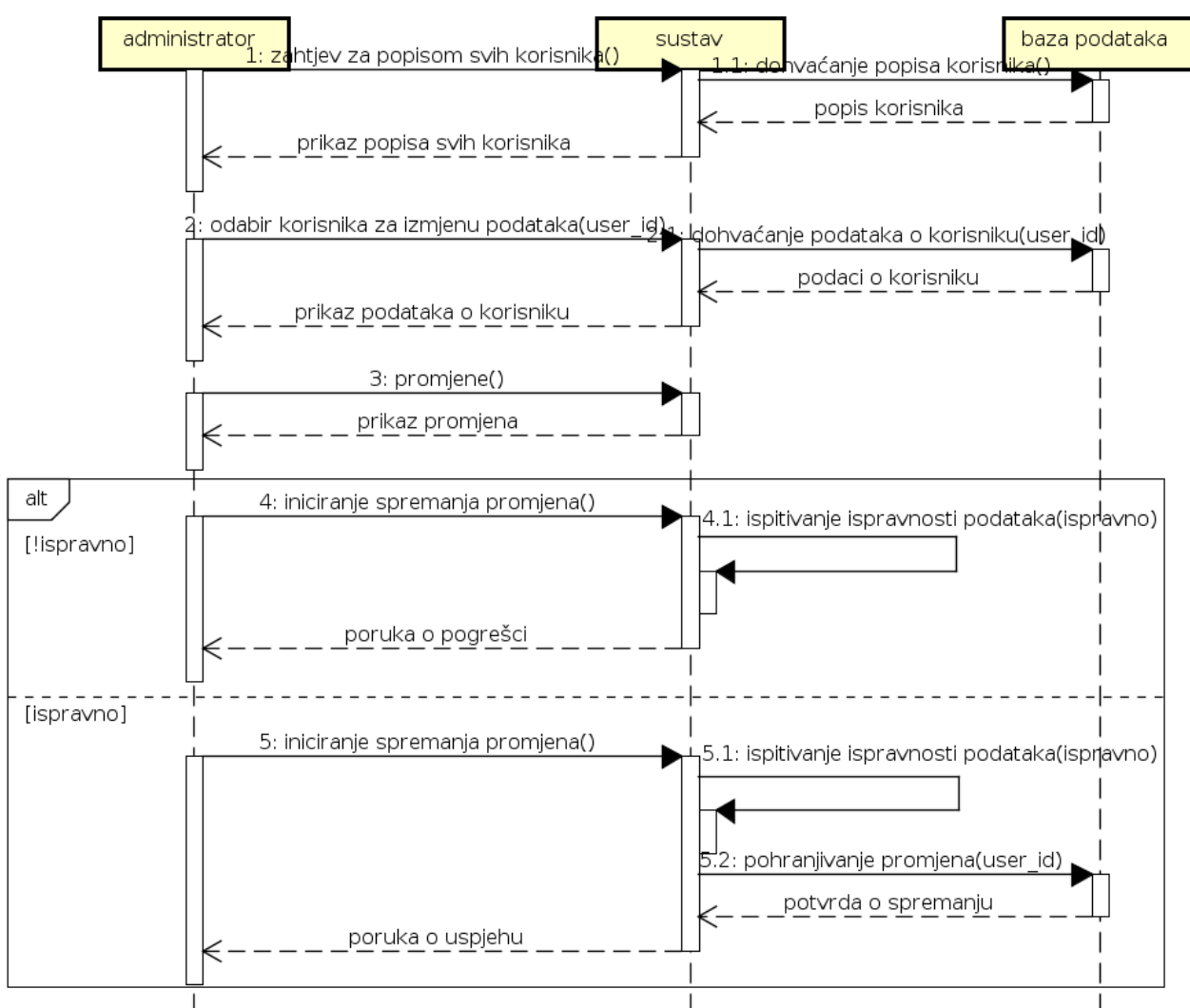
Administrator želi obrisati zvučni zapis. Administrator odabere tu mogućnost iz popisa. Sustav mu prikaže listu svih zvučnih zapisa koje je dohvatio iz baze podataka. Administrator odabire zvučni zapis kojeg želi izbrisati i inicira brisanje zvučnog zapisa. Sustav briše zvučni zapis iz baze podataka.



Slika 18: Sekvencijski dijagram za **UC12: ObrišiZvučniZapis**

UC13: UrediPodatkeKorisnika

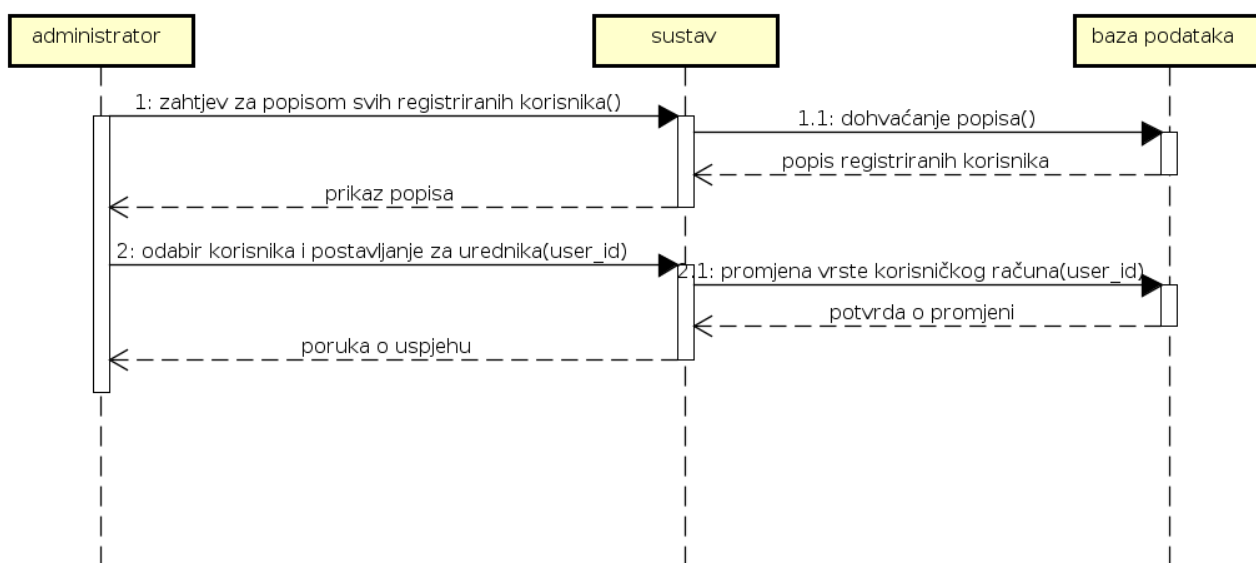
Administrator želi urediti podatke o nekog korisnika. Administrator odabere tu mogućnost iz popisa. Sustav mu prikaže listu svih korisnika koji nisu administratori ili vlasnik, koju je dohvatio iz baze podataka. Administrator odabire korisnika kojem želi izmijeniti podatke, pregledava podatke korisnika i po želji unosi promjene, te kada je zadovoljan izmjenama inicira spremanje promjena. Sustav ispituje ispravnost podataka, potom pohranjuje promjene u bazu podataka, te dojavljuje administratoru da su promjene uspješno obavljenje.



Slika 19: Sekvencijski dijagram za UC13: UrediPodatkeKorisnika

UC14: PostaviGlazbenogUrednika

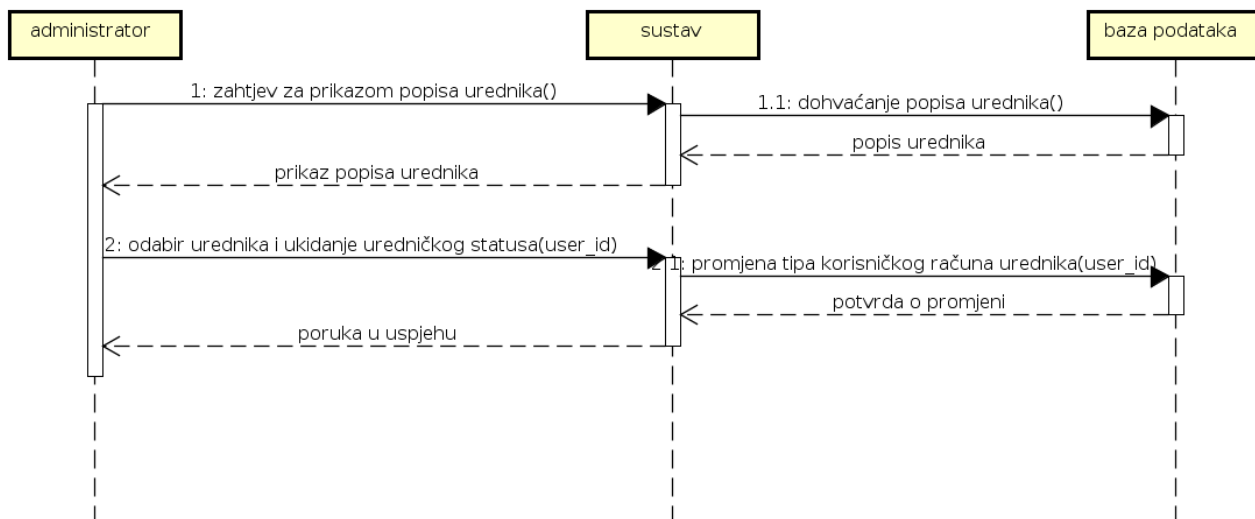
Administrator želi postaviti glazbenog urednika. Administrator odabere tu mogućnost iz popisa. Sustav mu prikaže listu svih korisnika koje je dohvatio iz baze podataka. Administrator pregledava popis korisnika i odabire jednog od korisnika i postavlja ga za glazbenog urednika. Sustav bilježi promjene i mijenja tip korisničkog računa odabranog korisnika u urednika.



Slika 20: Sekvencijski dijagram za **UC14: PostaviGlazbenogUrednika**

UC15: UkloniGlazbenogUrednika

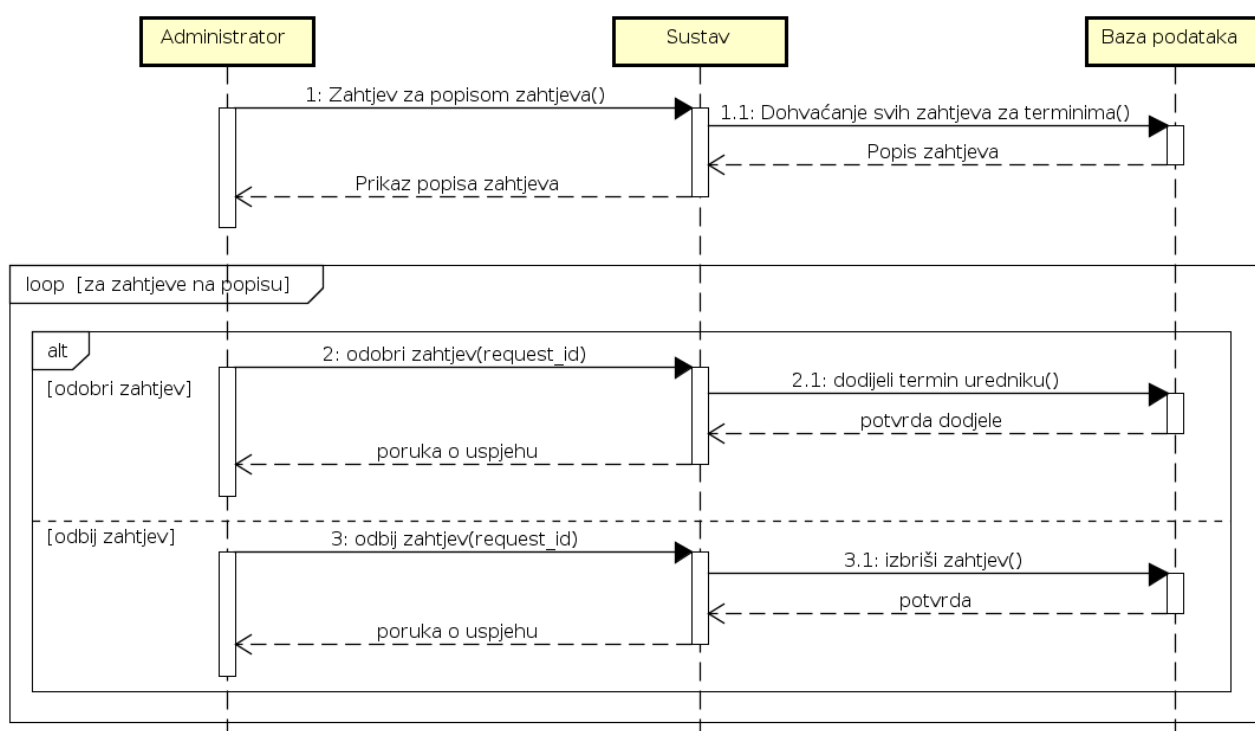
Administrator želi ukloniti glazbenog urednika. Administrator odabere tu mogućnost iz popisa. Sustav mu prikaže listu svih urednika koje je dohvatio iz baze podataka. Administrator određuje urednika kojem želi oduzeti uredničke ovlasti i potvrđuje odluku. Sustav bilježi promjene i mijenja tip korisničkog računa odabranog urednika u korisnika.



Slika 21: Sekvencijski dijagram za **UC15: UkloniGlazbenogUrednika**

UC16: OdlučiOZahtjevuZaTerminom

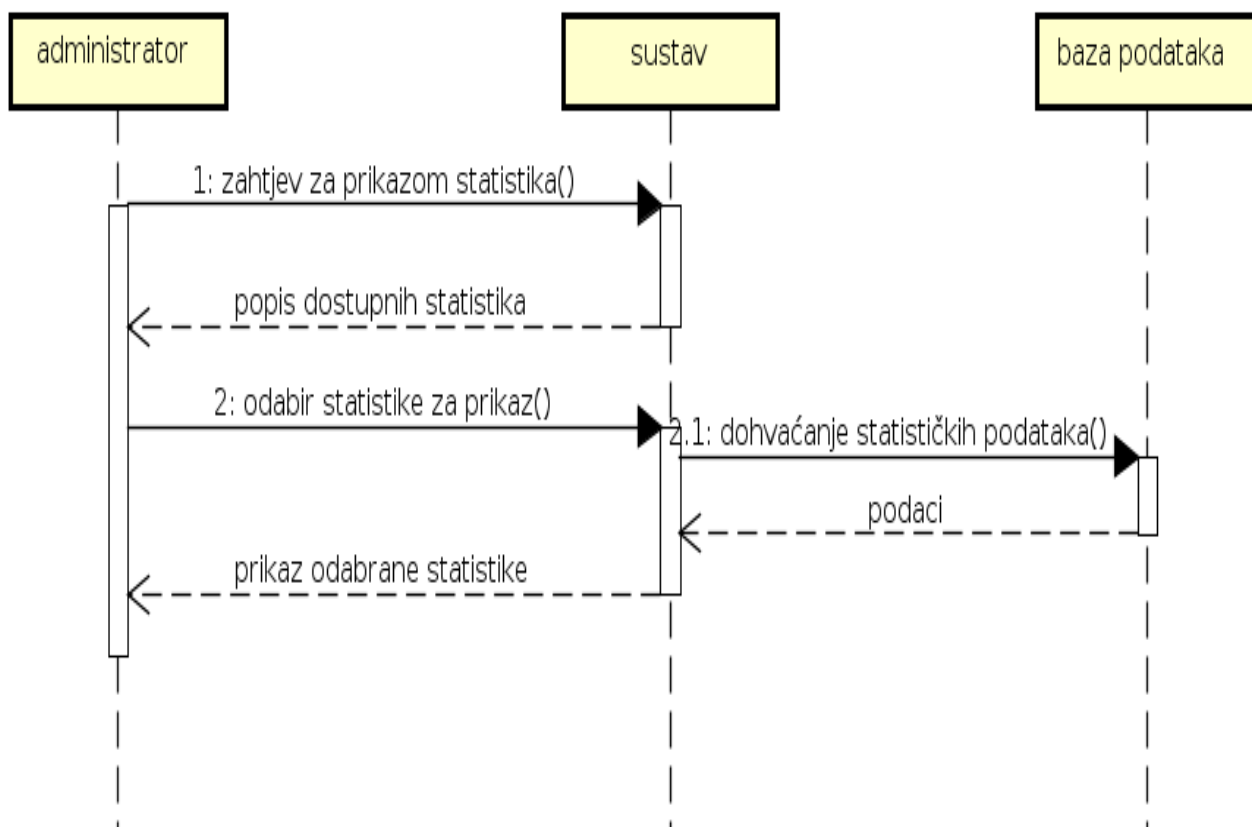
Administrator želi odlučiti o uredničkom zahtjevu za terminom. Administrator odabere tu mogućnost iz popisa. Sustav mu prikaže sve zahtjeve za terminom od urednika koje je dohvatio iz baze podataka. Administrator odlučuje o prihvaćanju ili odbijanju zahtjeva. U slučaju prihvaćanja zahtjeva, sustav dodjeljuje termin uredniku, šalje obavijest uredniku i bilježi promjenu. U slučaju odbijanja zahtjeva sustav šalje obavijest uredniku.



Slika 22: Sekvencijski dijagram za UC16: OdlučiOZahtjevuZaTerminom

UC17: PrikažiStatistiku

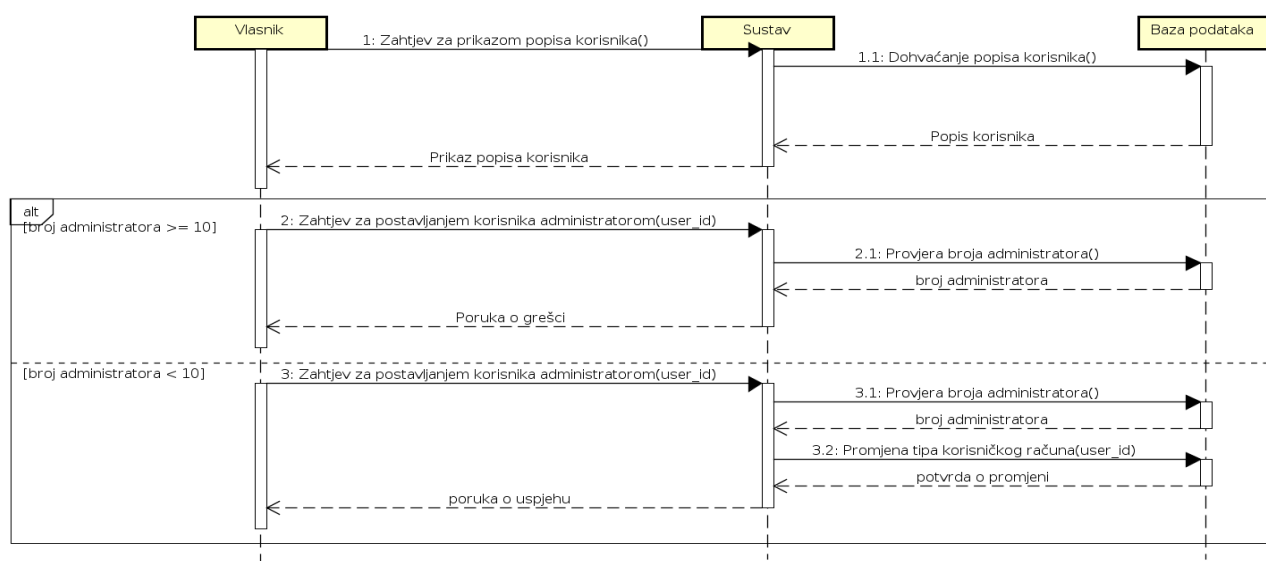
Administrator želi pregledati statistike o radu postaje. Administrator odabere tu mogućnost iz popisa. Sustav mu prikaže sve ponuđene statistike koje je dohvatio iz baze podataka. Administrator odabire jednu od ponuđenih statistika. Sustav prikazuje odabranu statistiku administratoru.



Slika 23: Sekvencijski dijagram za UC17: PrikažiStatistiku

UC18: PostaviAdministratora

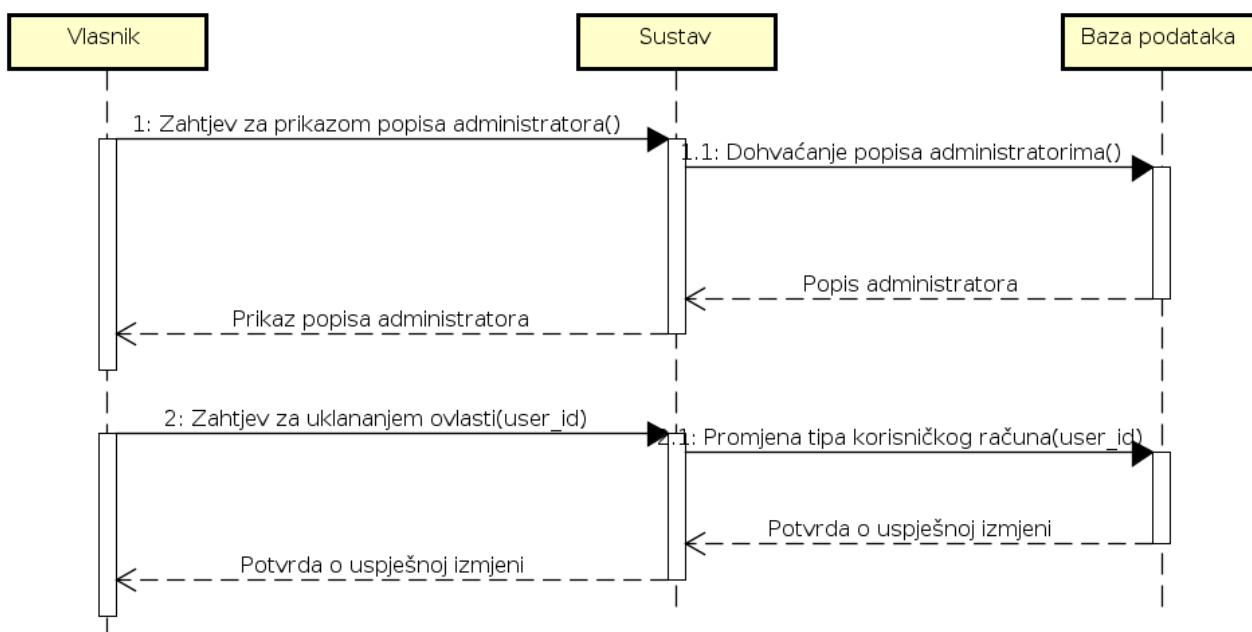
Vlasnik postaje želi postaviti novog administratora. Vlasnik odabere tu mogućnost iz popisa. Sustav mu prikaže sve trenutačne korisnike koje je dohvatio iz baze podataka ili u slučaju da je broj administratora 10 prekida akciju i ispisuje poruku o grešci. Vlasnik odabire jednog od korisnika i dodjeljuje mu administratorske ovlasti. Sustav bilježi promjene i mijenja tip korisničkog računa odabranog korisnika u administratora.



Slika 24: Sekvencijski dijagram za UC18: PostaviAdministratora

UC19: UkloniAdministratora

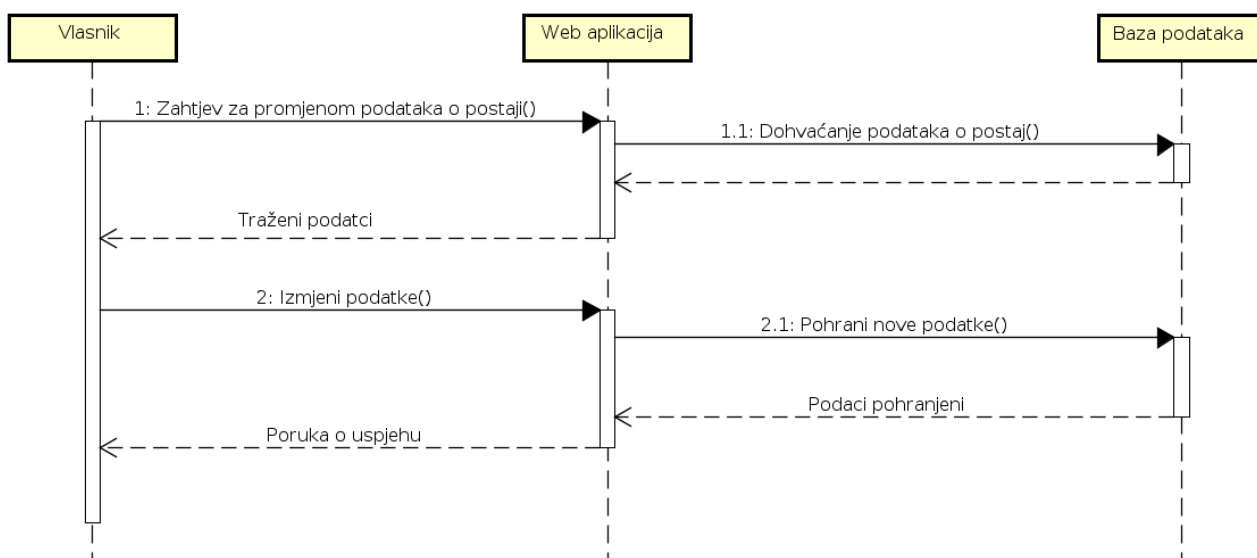
Vlasnik postaje želi ukloniti administratora. Vlasnik odabere tu mogućnost iz popisa. Sustav mu prikaže sve trenutne administratore koje je dohvatio iz baze podataka. Vlasnik odabire administratora kojeg želi ukloniti i inicira zahtjev za uklanjanjem koji se šalje sustavu. Sustav bilježi promjene i mijenja tip korisničkog računa odabranog administratora u korisnika.



Slika 25: Sekvencijski dijagram za **UC19: UkloniAdministratora**

UC20: UrediPodatkeOPostaji

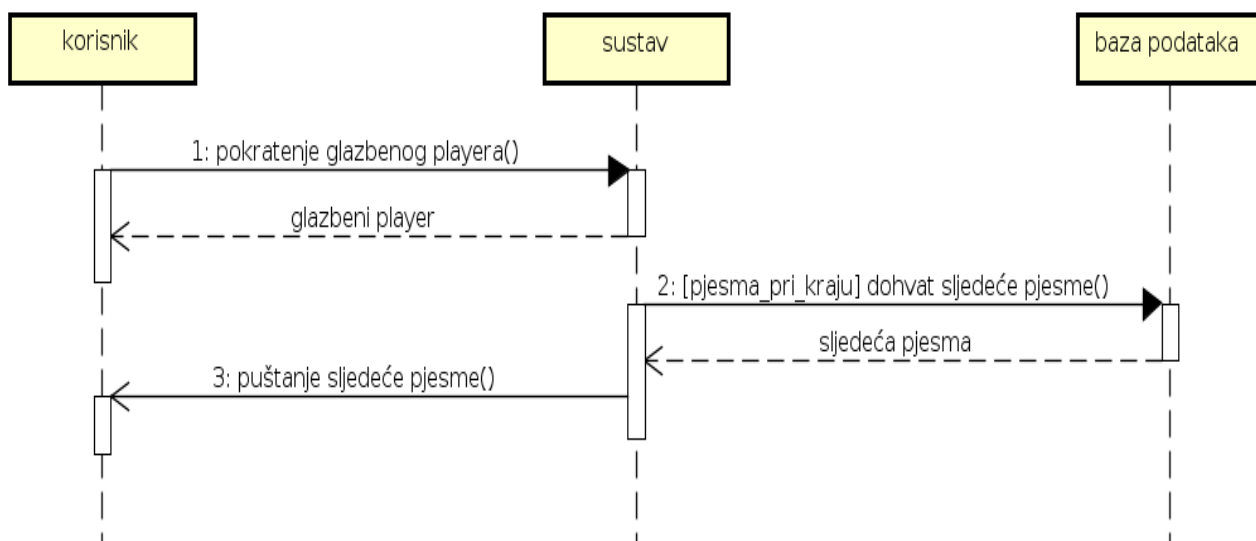
Vlasnik želi urediti podatke o postaji. Vlasnik odabere tu mogućnost iz popisa. Sustav mu prikaže podatke o postaji koje je dohvatio iz baze podataka. Vlasnik unosi promjene u te podatke dok god nije zadovoljan s njima. Potom inicira spremanje podataka, koji se šalju sustavu. Sustav provjerava ispravnost unesenih podataka, i ako su oni ispravni, pohranjuje promjene u bazu podataka te dojavljuje korisniku da su promjene uspješno obavljene. Inače mu prikazuje poruku o pogrešci.



Slika 26: Sekvencijski dijagram za UC20: UrediPodatkeOPostaji

UC21: SlušajRadioPostaju

Korisnik želi slušati radio postaju. Korisnik otvara web stranicu radio postaje. Sustav mu prikazuje koja pjesma trenutno svira. Korisnik zatim odabere opciju slušanja pjesme. Sustav nalazi pjesmu u bazi podataka te je pušta korisniku. Kada trenutna pjesma završi, sustav korisniku pušta iduću.



Slika 27: Sekvencijski dijagram za **UC21: SlušajRadioPostaju**

5. Ostali zahtjevi

Nefunkcionalni zahtjevi

1. Sustav mora podržavati neograničen broj registriranih korisnika.
2. Veoma je poželjna zastupljenost što više glazbenih žanrova, kako bi postaja privukla što više slušatelja.
3. Podaci o glazbenim zapisima moraju biti točni i pravilno uneseni.
4. Sustav mora biti jasan i pregledan, kako bi se korisnici lako snašli i mogli iskoristiti sve funkcionalnosti sustava bez obzira na razinu informatičke pismenosti.
5. Sustav mora svim aktorima omogućavati istovremeno korištenje svih funkcionalnosti; korisničko iskustvo ne smije biti ometeno zbog održavanja baze ili unošenja promjena u nju.

Zahtjevi domene primjene

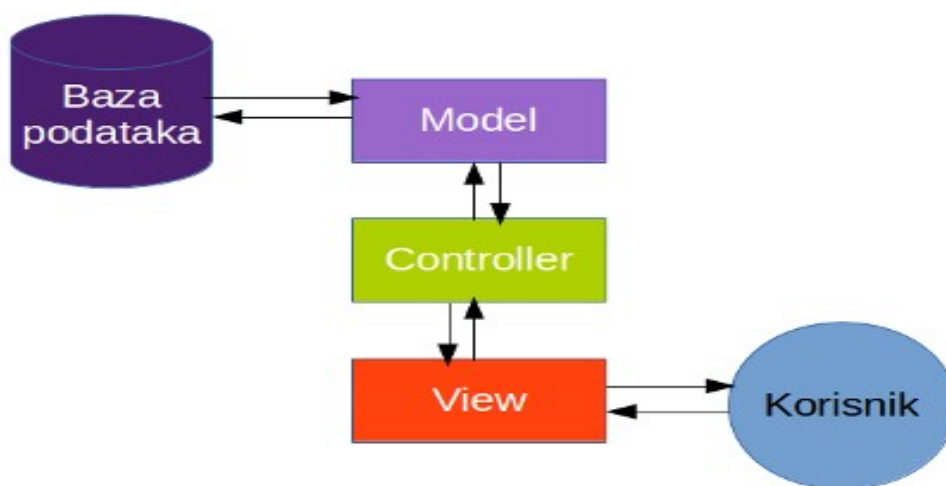
1. Postaja mora poštovati autorska prava i emitirati samo zapise koji su legalno dostupni.

6. Arhitektura i dizajn sustava

6.1. Svrha, opći prioriteti i skica sustava

Kako je cilj ovog projekta napraviti informacijski sustav za internetsku radio postaju, prirodno se nameće ideja da se isti izradi u obliku **web aplikacije**. Prednosti takve arhitekture sustava nad primjerice arhitekturom **desktop klijent-poslužitelj** su prenosivost (svaku računalo danas ima web preglednik), jednostavnost korištenja (korisnici su naviknuti na rad u web pregledniku), kao i jednostavnost izrade te održavanja (laka izrada sučelja u HTML-u i CSS-u, mnogobrojni resursi za pomoć i podršku). Također, ako dođe do daljnjeg razvoja aplikacije, zbog centraliziranosti neće doći do problema s fragmentacijom – svi će korisnici koristiti istu, najnoviju inačicu aplikacije.

Aplikacija će biti podijeljena u dva dijela, jedan koji će se pokretati unutar web preglednika te drugi koji će se pokretati na poslužitelju i komunicirati s bazom podataka. Komunikacija između ta dva dijela aplikacije vršit će se putem AJAX zahtjeva, prema modelu REST sučelja.



Slika 1: Dijagram MVC obrasca

Oba dijela aplikacije bit će oblikovana prema *Model-View-Controller* oblikovnom obrascu, koji odvađa pojedine dijelove aplikacije ovisno o namjerna na modele koji opisuju podatke i operacije nad njima, poglede (*views*) koji su zaduženi za prikaz podataka korisnicima, te

upravitelje (*controllers*) koji upravljaju korisničkim zahtjevima.

Klijentski dio

Klijentski dio aplikacije će se bazirati na *AngularJS* frameworku kao središnjem dijelu koji će obavljati dvosmjernu komunikaciju između modela i pogleda te će se brinuti za integritet i funkcionalnost same aplikacije. On će s poslužiteljskim dijelom komunicirati preko *REST* sučelja kojem će slati zahtjeve za raznim entitetima iz modela.

Angular2 karakterizira razdjeljivanje aplikacije u module koji se mogu smatrati zasebnim cjelinama te se uklapaju jedni u druge. Svaki modul ima svoj djelomični pogled koji se umeće u vanjski predložak na odgovarajuće mjesto i može imati svoj zaseban dizajn i logiku. Za interoperabilnost modula se koristi *dependency injection* mehanizam.

Za dizajniranje korisničkog sučelja koristit ćemo *SASS* koji je nadgradnja *CSS*-a i nudi brojne napredne funkcionalnosti, poput varijabli, hijerarhija elemenata, petlji i funkcija. Za strukturiranje korisničkog sučelja koristit će se *JADE*, a to je jezik koji se kompilira u *HTML*. On također predstavlja dodatan sloj funkcionalnosti te omogućava petlje, funkcije jednostavniju sintaksu, i još mnogo toga.

Za upravljanjem korisničkim sučeljem ćemo koristiti *Typescript* koji je nadjezik *JavaScripta* i pruža dodatne funkcionalnosti koje su u nacrtu *ECMAScript 6*, a još nisu implementirane nativno u preglednicima. Također je mnogo veća podrška dokumentacijom za *Angular2* i *TypeScript* nego li je za *JavaScript* ili *Dart*.

Komponente

Komponente koje ćemo implementirati se poklapaju s funkcijskim zahtjevima od same aplikacije budući da se na taj način modularizira klijentska strana. Komponente će biti ugniježdene u nadkomponente koje odgovaraju ulogama koje postoje u našoj aplikaciji (Za detaljan popis svih komponenti pogledajte u popis funkcijskih zahtjeva). Nadkomponente će biti *User*, *Administrator*, *Owner* ... , a komponente će biti *Listen*, *Settings*, *Wishlist* od *User*, i tako dalje će se poklapati s funkcijskim zahtjevima i za ostale nadkomponente.

Poslužiteljski dio

Poslužiteljski dio aplikacije bit će oblikovan kao *REST* sučelje koje će primati zahtjeve od klijentskog dijela, obaviti odgovarajuće akcije i potom vratiti rezultate klijentskom dijelu koji će ih prikazati korisniku. Ta komunikacija obavljat će se putem *HTTP* zahtjeva i odgovora, unutar kojih će podaci biti zapisani u *JSON* formatu.

Ovaj dio aplikacije bit će također razdijeljen prema MVC obrascu, no kako u općenitom slučaju on neće biti zadužen za prikaz podataka korisniku, uglavnom će sadržavati samo modele i upravitelje. Ulogu korisnika u interakciji s njime imat će klijentski dio aplikacije.

Poslužiteljski dio aplikacije vršit će i komunikaciju s bazom podataka, u koju će se pohranjivati svi podaci o korisnicima, zvučnim zapisima, listama želja i listama za reprodukciju, i svemu ostalom što je potrebno za rad aplikacije. No ta komunikacija neće biti izravna, već će se za nju pobrinuti *Peewee ORM*, biblioteka koja će iz entiteta definiranih kao razredi u programskom jeziku *Python* generirati odgovarajuće *SQL* tablice, kao i metode kojima će se implicitno vršiti spremanje, dohvaćanje i izmjena podataka u tablici.

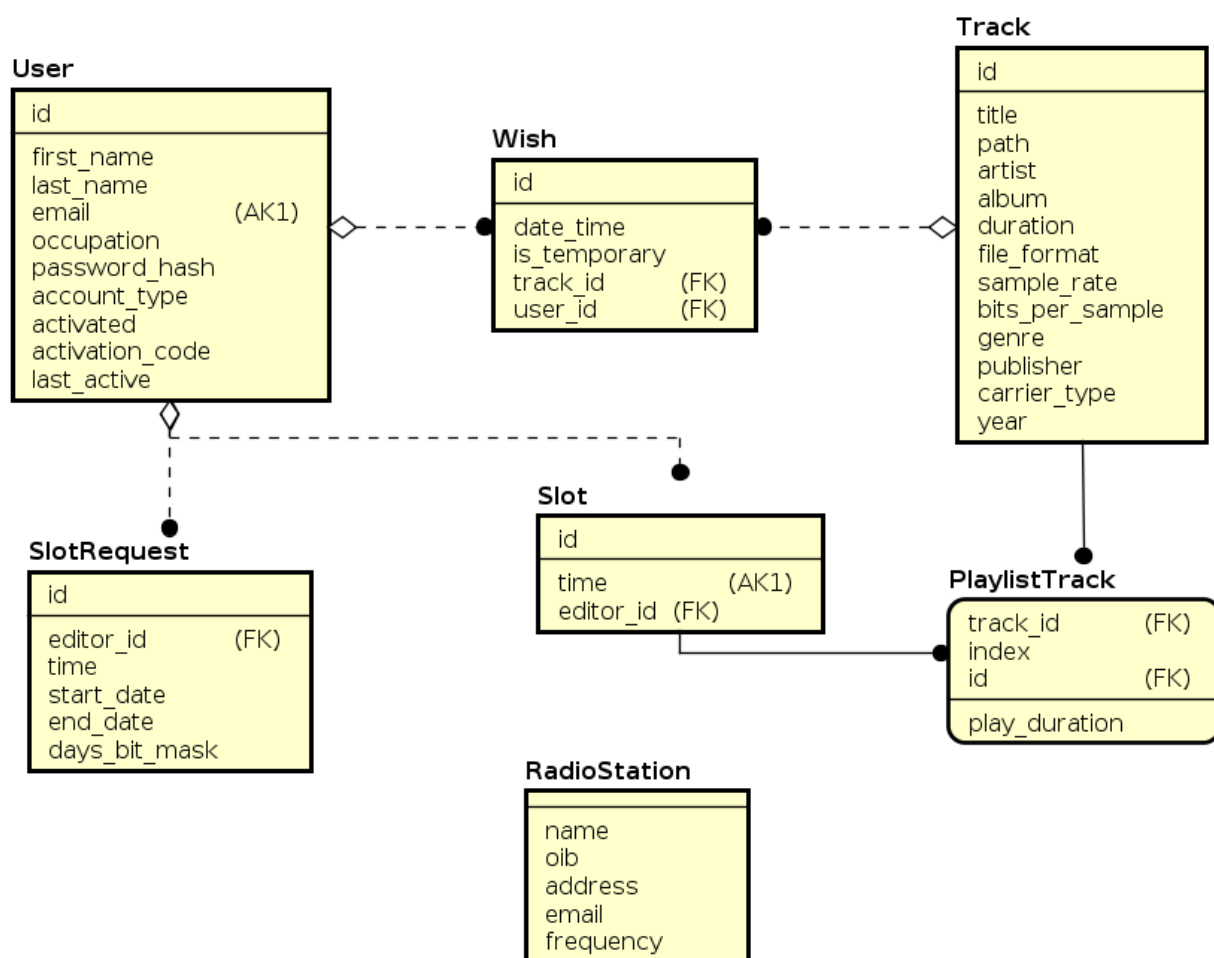
Za izradu poslužiteljskog dijela odabran je programski jezik *Python* te njegov framework *Flask*, koji se odlikuje velikom jednostavnošću, brzinom i lakoćom učenja, a za komunikaciju s bazom podataka odabran je već spomenuti *Peewee*.

Modeli

U MVC obrascu, modeli predstavljaju podatke kojima aplikacija upravlja te akcije koje nad njima može izvršiti. Oni se definiraju kao razredi sa svojim atributima i metodama. Ova aplikacija sadržavat će sljedeće modele: *Track*, *User*, *Slot*, *SlotRequest*, *PlaylistTrack*, *Wish* te *RadioStation*. Značenje njihovih atributa opisano je prilikom definiranja strukture baze podataka, dok su sve moguće akcije odgovarajućeg modela opisane u odjeljku 6.2, prilikom opisa dijagrama razreda.

Baza podataka

Iz prethodno navedenih modela automatski će se generirati SQL tablice u bazi podataka, kao i izvršavati sve potrebne SQL operacije. Izgled nastale sheme baze podataka dan je na slici:



Slika 2: ER model baza podataka

Detaljniji opis pojedinih relacija i značenja njihovih atributa dan je u sljedećim tablicama:

| User |
|---------------------------------|
| <u>Id</u> , INTEGER |
| first_name, VARCHAR(255) |
| last_name, VARCHAR(255) |
| email, VARCHAR(255) |

Korisnik aplikacije

ID korisnika, **primarni ključ**

Ime korisnika

Prezime korisnika

Adresa e-pošte, **alternativni ključ**

| |
|--------------------------------------|
| Occupation, VARCHAR(255) |
| password_hash, VARCHAR(255) |
| account_type, INTEGER |
| activated, BOOLEAN |
| activation_code, VARCHAR(255) |
| last_active, DATETIME |

*Zanimanje korisnika**Hash vrijednost lozinke**Tip korisničkog računa, (1 – reg. korisnik, 2 – urednik, 3 – administrator, 4 – vlasnik)**Je li korisnički račun aktiviran****Jedinstveni** kod kojim se vrši aktivacija korisničkog računa**Vrijeme zadnje aktivnosti, potrebno za statistiku*

| Track |
|-----------------------------------|
| <u>Id</u> , INTEGER |
| title, VARCHAR(255) |
| path, VARCHAR(255) |
| artist, VARCHAR(255) |
| album, VARCHAR(255) |
| duration, INTEGER |
| file_format, VARCHAR(255) |
| sample_rate, FLOAT |
| bits_per_sample, INTEGER |
| genre, VARCHAR(255) |
| publisher, VARCHAR(255) |
| carrier_type, VARCHAR(255) |
| year, INTEGER |

Zvučni zapis

*ID zapisa, **primarni ključ****Naziv zvučnog zapisa**Putanja do datoteke na poslužitelju**Umjetnik – autor zapisa**Album na kojemu je zvučni zapis**Trajanje zvučnog zapisa u sekundama**Format u kojemu je pohranjen zapis (MP3, WAV, OGG...)**Učestalost uzorkovanja**Broj bitova po uzorku**Žanr kojem pripada glazbeni zapis**Izdavač zapisa**Vrsta nosača zvuka**Godina izdavanja*

| Slot |
|----------------------------|
| <u>Id</u> , INTEGER |
| time, TIME |
| editor_id, INTEGER |

Urednički termin

*ID termina, **primarni ključ****Datum i vrijeme početka termina, **alternativni ključ****ID urednika kojemu je dodijeljen taj termin, **strani ključ***

| SlotRequest |
|----------------------------|
| <u>Id</u> , INTEGER |
| time, TIME |
| editor_id, INTEGER |

Zahtjev za terminom

*ID zahtjeva, **primarni ključ****Vrijeme termina u danu, obavezno počinje na puni sat**ID urednika koji je zatražio taj termin, **strani ključ***

| |
|-------------------------------|
| start_date, DATE |
| end_date, DATE |
| days_bit_mask, INTEGER |

Početni datum od kojega bi započinjao dodijeljeni termin

Konačni datum do kojega bi termin bio dodijeljen

Za koje sve dane u tjednu je zatražen termin (enkodirano kao bit-mask, spremljena kao cijeli broj, npr. 19 = 0010011₂ = pon., uto. i pet.)

| Wish |
|------------------------------|
| <u>Id</u> , INTEGER |
| track_id, INTEGER |
| user_id, INTEGER |
| date_time, DATETIME |
| is_temporary, BOOLEAN |

Korisnička želja

ID želje, **primarni ključ**

ID zapisa koji je željen, **strani ključ**

ID korisnika koji je izrazio želju, **strani ključ**

Datum i vrijeme kada je želja izražena

Je li ta želja privremena (vidljiva samo korisniku i podložna promjenama), ili je već potvrđena (konačna, vidljiva administratorima i urednicima)

| PlaylistTrack |
|--------------------------------------|
| track_id, INTEGER |
| time, TIME |
| index, INTEGER |
| play_duration, INTEGER |
| K = { track_id, time, index } |

Zapis na listi za reprodukciju

ID zapisa, **strani ključ**

Vrijeme početka termina, **strani ključ** relacije **Slot**

Redni broj pjesme u tom terminu

Koliko će se dugo svirati ovaj zapis

Složeni primarni ključ

| RadioStation |
|------------------------------|
| name, VARCHAR(255) |
| description, TEXT |
| oib, VARCHAR(255) |
| address, VARCHAR(255) |
| email, VARCHAR(255) |
| frequency, FLOAT |

Podaci o radio postaji

Naziv radio postaje

Opis radio postaje

OIB radio postaje kao pravne osobe

Adresa

Adresa e-pošte

Frekvencija na kojoj se odašilje program radio postaje

Napomene:

1. U ovoj se tablici smije nalaziti samo jedan redak
2. Tablica stoga nema primarni ključ

Za potrebe ove web aplikacije konkretno ćemo koristiti *SQLite* bazu podataka, koja je prikladna za manje aplikacije s ograničenom količinom prometa. No zbog korištenja *Peewee* biblioteke, ako se u budućnosti pojavi potreba za podržavanjem mnogo većeg prometa, moguć je vrlo jednostavan prijelaz na neki od moćnijih sustava kao što je *PostgreSQL*, bez ikakvih promjena koda same aplikacije.

Upravitelji

Za razliku od nekih drugih jezika i frameworka, u Pythonu i Flasku upravitelji su jednostavne funkcije, te su stoga opisani ovdje, a ne u odjeljku o dijagramima razreda.

Kako je poslužiteljski dio ove aplikacije modeliran kao *REST* sučelje, tako svaki pojedini *URL* koji predstavlja neku akciju ima svog upravitelja, koji prihvaća i obrađuje zahtjeve koje mu klijent pošalje, te na njih odgovara u dogovorenom obliku (JSON format).

Osim *REST* upravitelja, postoji i nekoliko njih drugačijeg tipa, koji ne odgovaraju na zahtjeve klijenta, već služe prikazivanju same web stranice, tj. njena dva dijela, početne stranice te stranice s postavkama i korisničkim mogućnostima.

Upravitelji koje sadrži poslužiteljski dio aplikacije, grupirani prema zajedničkim objektima djelovanja, su (uz naziv upravitelja nalazi se i njegov relativni *URL*):

Općenito, prikaz stranica i slušanje programa radio postaje

- **show_index()** *"/"*
 - Prikazuje početnu stranicu, na kojoj su glazbeni player za slušanje radio postaje, obrazac za prijavu i registraciju te osnovni podaci o radio postaji
- **show_settings()** *"/settings"*
 - *Za registrirane korisnike.* Prikazuje glavnu stranicu web aplikaciju, na kojoj su sve postavke i mogućnosti korisnika.
- **get_currently_playing_track()** *"/player/get"*
 - Služi za dohvaćanje zvučnog zapisa koji se trenutno pušta na radio postaji
- **get_currently_playing_track_info()** *"/player/info"*
 - Služi za dohvaćanje informacija o trenutno reproduciranom zapisu

- **get_next_on_schedule()** *"/player/schedule"*
 - Služi za dohvaćanje idućih uredničkih emisija na rasporedu

Prijava, registracija i aktivacija korisničkih računa

- **process_login()** *"/user/auth/login"*
 - Obraduje korisnikov pokušaj prijave u sustav, ako je pokušaj uspješan, prijavljuje korisnika i preusmjerava ga natrag na početnu stranicu
- **process_registration()** *"/user/auth/register"*
 - Obraduje korisnikov zahtjev za registracijom, ako je uspješan, stvara novog korisnika i šalje email s aktivacijskim linkom
- **process_activation()** *"/user/auth/activate"*
 - Aktivira korisnički račun putem aktivacijskog linka, čime omogućuje korisniku da se ubuduće prijavljuje u sustav
- **process_signout()** *"/user/auth/signout"*
 - Odjavljuje korisnika iz sustava

Upravljanje vlastitim korisničkim računom

- **get_account_data()** *"/user/account/get"*
 - Vraća sve podatke o trenutnom korisniku
- **modify_account_data()** *"/user/account/modify"*
 - Mijenja korisničke podatke trenutnog korisnika
- **delete_account()** *"/user/account/delete"*
 - Zauvijek briše korisnički račun trenutnog korisnika iz sustava
- **change_account_password()** *"/user/account/change_password"*
 - Mijenja lozinku trenutnog korisnika, uz sigurnosnu provjeru (ispitivanje stare lozinke)
- **get_account_type()** *"/user/account/type"*
 - Vraća vrstu korisničkog računa trenutno prijavljenog korisnika

Korisničko upravljanje listom želja

- **get_wishlist()** *"/user/wishlist/get"*
 - Vraća trenutnu listu želja korisnika

- **set_wishlist()** *"/user/wishlist/set"*
 - Sastavlja novu listu želja korisnika
- **confirm_wishlist()** *"/user/wishlist/confirm"*
 - Potvrđuje korisnikovu listu želja

Administratorsko upravljanje zvučnim zapisima

- **add_track()** *"/admin/tracks/add"*
 - Dodaje zvučni zapis u sustav, postavlja datoteku na poslužitelj i unosi sve bitne podatke o zapisu u sustav
- **get_track()** *"/admin/tracks/<id>/get"*
 - Vraća podatke o odabranom zvučnom zapisu
- **edit_track(id)** *"/admin/tracks/<id>/edit"*
 - Uređuje podatke o zvučnom zapisu s danim ID-om
- **delete_track(id)** *"/admin/tracks/<id>/delete"*
 - Briše zvučni zapis s danim ID-om iz sustava

Administratorsko upravljanje urednicima

- **list_editors()** *"/admin/editors/list"*
 - Vraća popisa svih glazbenih urednika u sustavu
- **add_editor(id)** *"/admin/editors/add/<id>"*
 - Postavlja jednog od registriranih korisnika (onog s danim ID-om) za novog glazbenog urednika
- **remove_editor(id)** *"/admin/editors/<id>/remove"*
 - Ukida urednički status korisniku s danim ID-om

Administratorsko upravljanje zahtjevima za terminima

- **list_requests()** *"/admin/requests/list"*
 - Vraća popis svih trenutno aktivnih zahtjeva za dodjelom termina urednicima
- **allow_request(id)** *"/admin/requests/<id>/allow"*
 - Potvrđuje zahtjev urednika za dodjelom nekog termina, taj se termin u sustavu bilježi kao zauzet
- **deny_request(id)** *"/admin/requests/<id>/deny"*

- Odbija zahtjev urednika za dodjelom termina

Administratorsko upravljanje korisnicima

- **list_users()** *"/admin/users/list"*
 - Vraća popis svih korisnika u sustavu
- **get_user_data(id)** *"/admin/users/<id>/get"*
 - Vraća podatke korisničkog računa s danim ID-om
- **modify_user_data(id)** *"/admin/users/<id>/modify"*
 - Izmjenjuje podatke korisničkog računa s danim ID-om
- **delete_user(id)** *"/admin/users/<id>/delete"*
 - Briše korisnika s danim ID-om iz sustava

Uredničko upravljanje terminima za reprodukciju

- **list_editor_slots()** *"/editor/slots/list"*
 - Vraća popis svih termina dodjeljenih trenutnom korisniku (koji je urednik), kao i trenutno aktivnih zahtjeva koje je poslao
- **request_slot()** *"/editor/slots/request"*
 - Stvara zahtjev za dodjelom određenog termina uredniku
- **get_list(id)** *"/editor/slots/<id>/get_list"*
 - Vraća trenutno spremljenu listu za reprodukciju u danom terminu
- **set_list(id)** *"/editor/slots/<id>/set_list"*
 - Sastavlja novu listu za reprodukciju za dani termin

Vlasničko upravljanje administratorima

- **list_admins()** *"/owner/admins/list"*
 - Vraća popis svih administratora u sustavu
- **add_admin(id)** *"/owner/admins/add/<id>"*
 - Postavlja korisnika s danim ID-om za administratora sustava (ako ih je u sustavu manje od 10)
- **remove_admin(id)** *"/owner/admins/<id>/remove"*
 - Uklanja administratorski status korisniku s danim ID-om

Vlasničko upravljanje radio stanicom

- **modify_station_data()** *"/owner/station/modify"*
 - Mijenja podatke o radio stanici

Pregled podataka o postaji

- **get_station_data()** *"/station/get"*
 - Vraća sve podatke o radiopostaji

Pregled i pretraživanje zvučnih zapisa

- **list_tracks()** *"/tracks/list"*
 - Vraća popis svih zvučnih zapisa u sustavu
- **search_tracks()** *"/tracks/search"*
 - Vraća popis svih zvučnih zapisa koji zadovoljavaju kriterije pretrage
- **get_global_wishlist()** *"/tracks/wishlist"*
 - Vraća globalnu listu želja svih korisnika
- **get_popular()** *"/tracks/popular"*
 - Vraća popis najpopularnijih zvučnih zapisa na radiopostaji

Pretraživanje korisnika

- **search_users()** *"/users/search"*
 - Vraća popis svih korisnika koji zadovoljavaju kriterije pretrage

Pregled rasporeda termina

- **get_global_schedule()** *"/slots/schedule"*
 - Vraća raspored svih dodjeljenih termina

Pregled raznih statistika

- **get_track_play_stat()** *"/stats/tracks/<id>/play_count"*
 - Vraća koliko je puta zvučni zapis bio reproduciran u programu postaje
- **get_global_wishlist_stat()** *"/stats/tracks/wishlist"*
 - Vraća popis svih zapisa koji su na listama želja korisnika, zajedno s brojačem pojavljivanja na listi

- **get_most_wished_track()** `"/stats/tracks/most_wanted"`
 - Vraća podatke o najtraženijem zvučnom zapisu (traženom putem lista želja)
- **get_most_wished_track_stat()** `"/stats/tracks/most_wanted/wish_count/<start_date>/<end_date>"`
 - Vraća koliko je puta najtraženiji zapis bio tražen u nekom vremenskom intervalu
- **get_active_users_count()** `"/stats/active_users/count"`
 - Vraća broj trenutno aktivnih korisnika u sustavu
- **get_active_admins_list()** `"/stats/active_admins/list"`
 - Vraća popis svih trenutno aktivnih administratora u sustavu
- **get_editor_preferred_tracks(id)** `"/stats/editors/<id>/preferred_tracks"`
 - Vraća popis zvučnih zapisa koje urednik s danim ID-om najčešće stavlja na liste za reprodukciju

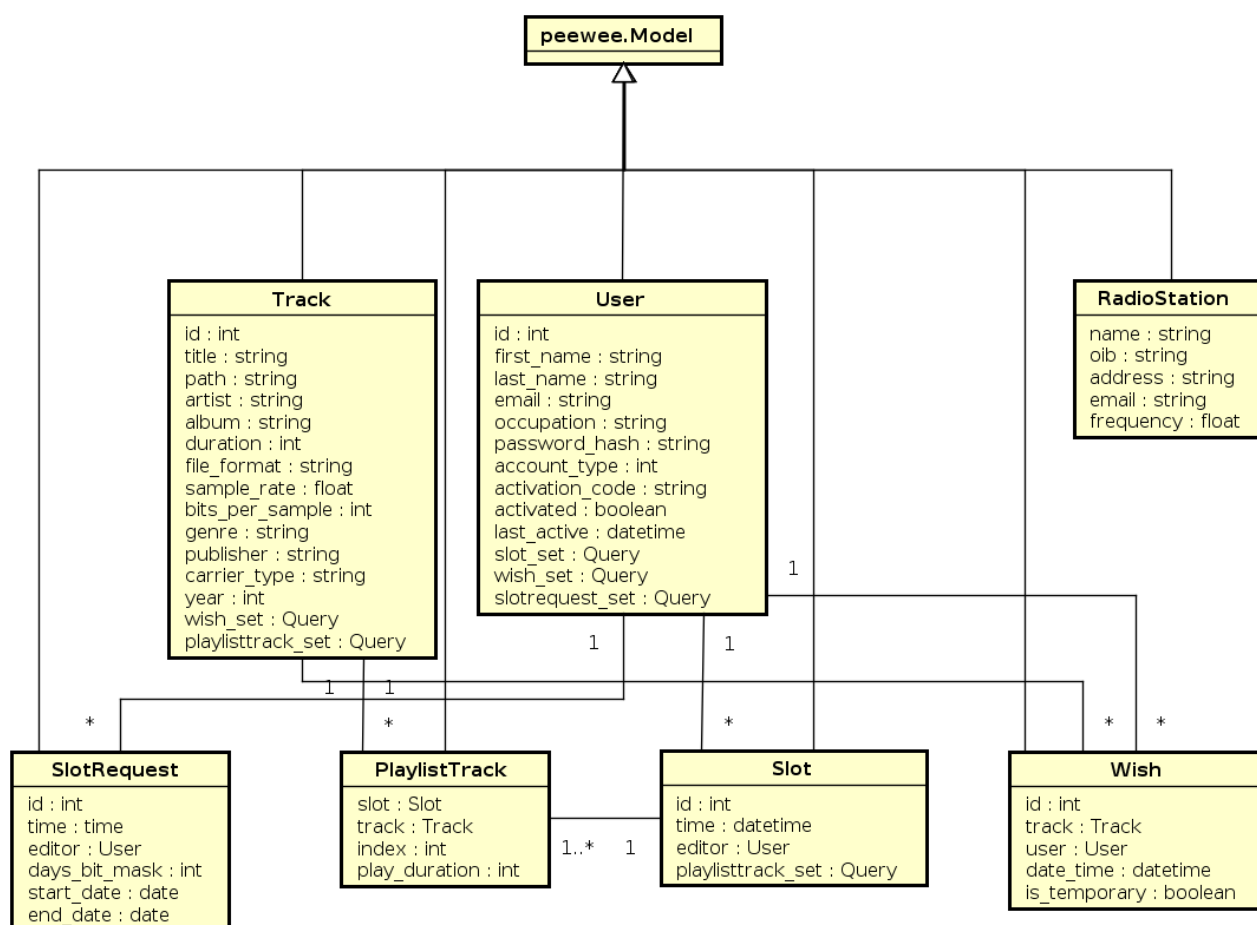
6.2. Dijagram razreda s opisom

Napomena: U korištenom programskom jeziku, Pythonu, objektno orijentirana paradigma se upotrebljava na malo drugačiji način nego li primjerice u C#-u ili Javi, pa se tako ne koriste getter i setter metode, ne postoje eksplicitni modifikatori pristupa itd., što znači da će razredi možda biti drugačije strukturirani od "standardnog načina" na koji su mnogi naviknuti.

U oblikovanju ovog sustava koristi se objektno usmjerena paradigma, ponajprije pri izgradnji modela *MVC* obrasca. Svaki model oblikovan je kao poseban razred koji nasljeđuje razred *peewee.Model* (on pruža sve funkcionalnosti komunikacije s bazom podataka, neke implicitno, a neke putem definiranih metoda koje njegova djeca nasljeđuju, kao što su *save()* i *delete_instance()*), i predstavlja određeni tip objekata s kojima upravlja ova aplikacija.

Budući da se modeli odnose jedan-na-jedan prema relacijama u bazi podataka, značenje pojedinih modela i njihovih atributa opisano je prilikom opisa strukture baze podataka. Jedini dodatak tome su povratne reference stranih ključeva koje *Peewee* biblioteka sama nadodaje modelima (primjer: relacija *Slot* ima strani ključ na relaciju *User*, pa model *User* u sebi sadrži atribut *slot_set* koji omogućuje jednostavniji pristup s njime povezanim *Slot* objektima. Taj se pristup implicitno ostvaruje SQL upitima, najčešće u samom trenutku

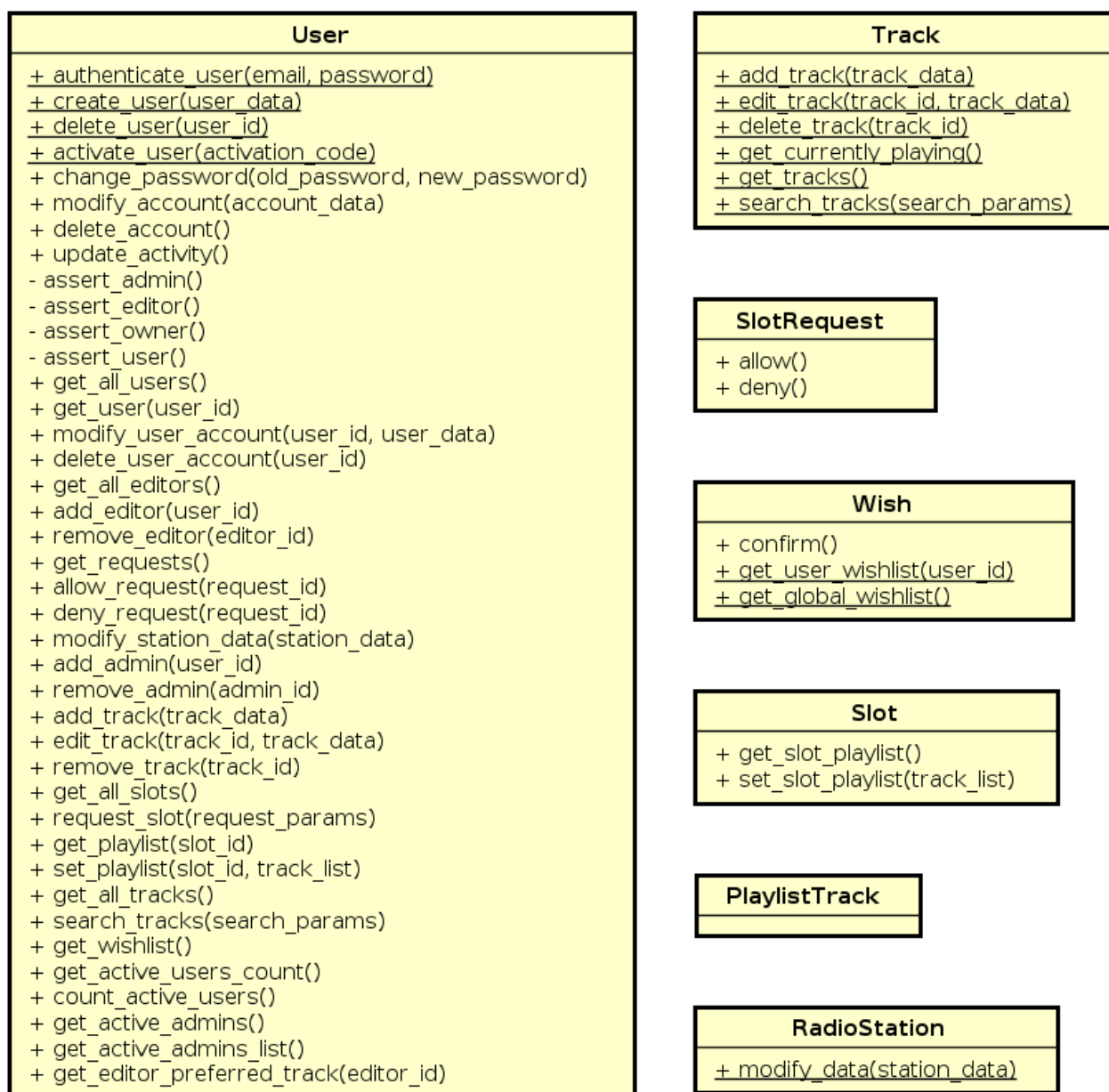
pristupa njegovoj vrijednosti). Prikaz nasljeđivanja i atributa svih modela dan je na slici 3.



Slika 3: Dijagram razreda – modeli – dio 1

Svaki model ima definirane osnovne CRUD (create, read, update, delete) operacije koje nasljeđuje od *peewee.Model* razreda, a dodatno i neke operacije specifične za taj model. Uz to, ključni dio logike cijele aplikacije sadržan je unutar modela *User*, koji sadrži operacije stvarnih korisnika, poput *create_wishlist()* ili *change_password()*. Aplikacija tako najčešće radi na sljedeći način: klijentski dio prosljeđuje korisničku akciju kao zahtjev nekom od upravitelja na poslužiteljskom dijelu, koji ispituje ispravnost parametara zahtjeva, i ako su ispravni, poziva odgovarajuću metodu instance razreda *User* koja predstavlja trenutno prijavljenog korisnika, te čeka njen rezultat. Kada ga primi, pretvara ga u dogovoreni format i vraća klijentskom dijelu, koji ga potom prikazuje korisniku.

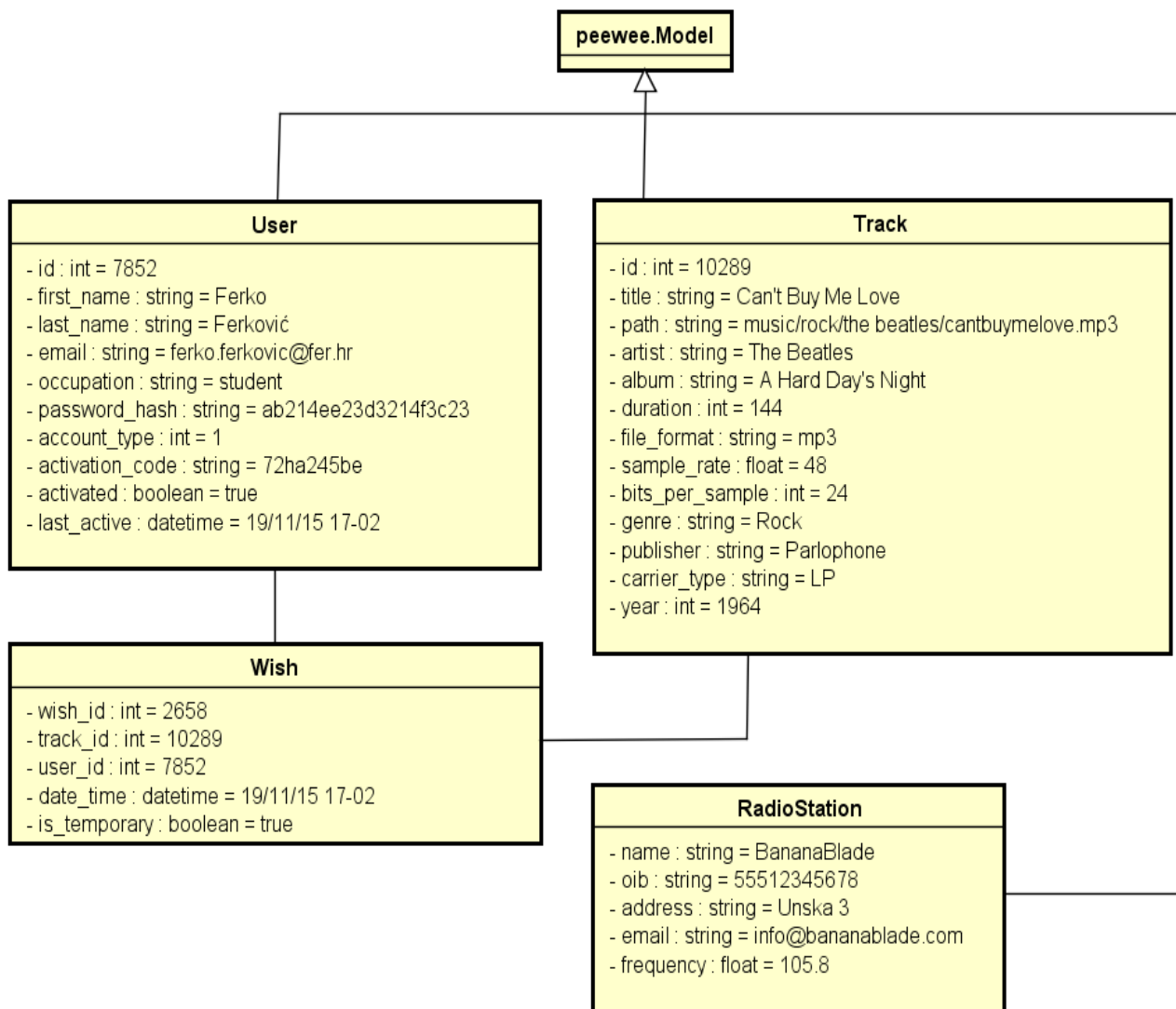
Popis svih operacija pojedinih modela dan je na slici 4.



Slika 4: Dijagram razreda – modeli – dio 2

6.3. Dijagram objekata

Na slici je dan prikaz dijagrama objekata u trenutku dodavanja nove želje na korisnikovu listu želja.

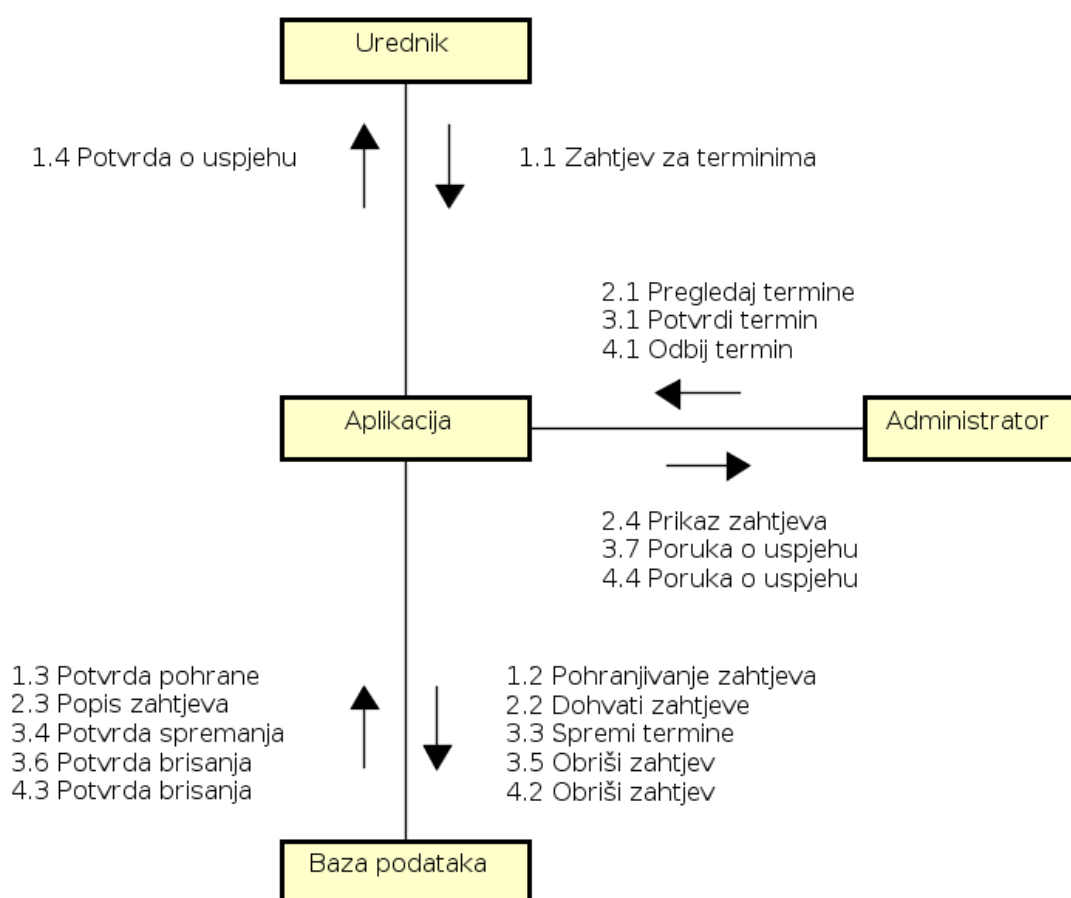


Slika 5: Dijagram objekata u trenutku dodavanja nove želje na korisnikovu listu želja

6.4. Ostali UML dijagrami

Komunikacijski dijagram

Glazbeni urednik želi zatražiti dozvolu novog termina. Urednik sustavu šalje zahtjev za određenim terminima. Sustav prima zahtjev, pohranjuje ga u bazu podataka i obavještava urednika o uspješnoj pohrani.



Slika 6: Komunikacijski dijagram za **UC8: ZatražiTerminZaReprodukciju** i **UC16: OdlučiOZahtjevuZaTerminom**

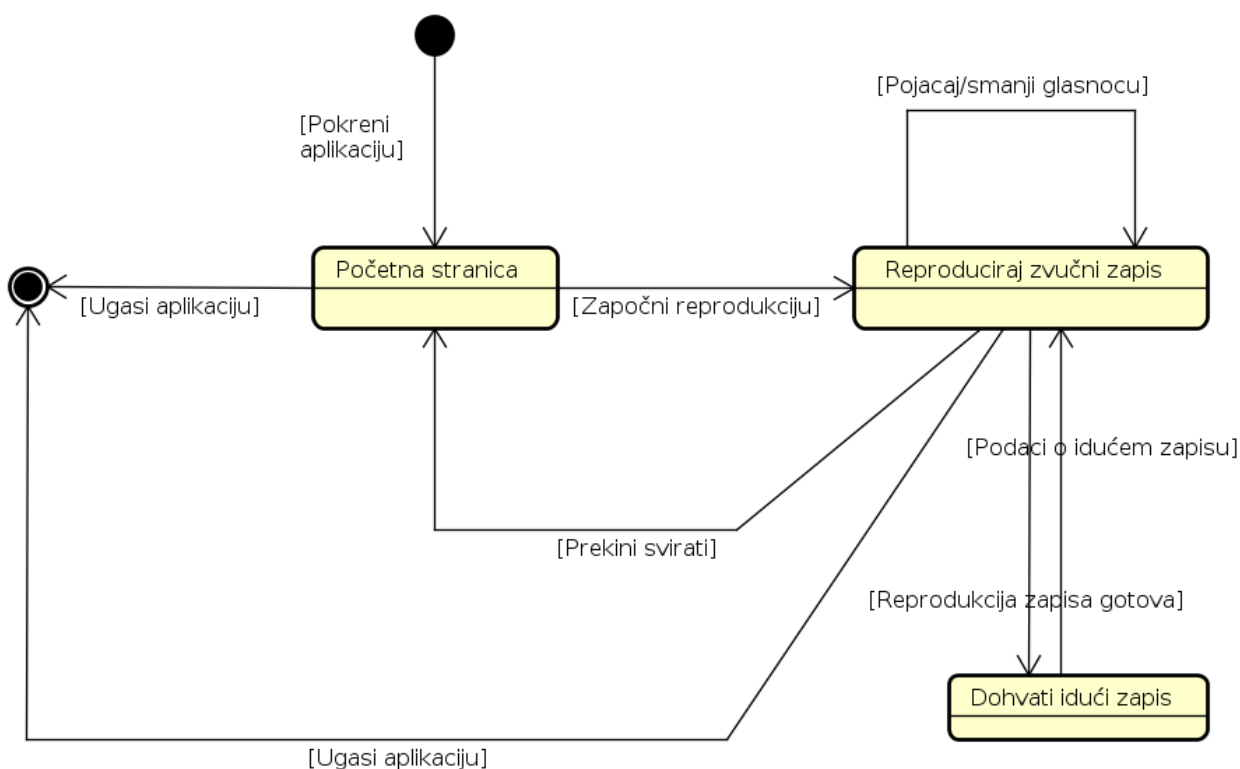
Administrator želi pregledati zahtjeve za terminima. Šalje zahtjev sustavu, koji iz baze podataka dohvaća sve postojeće zahtjeve za terminima te ih prikazuje administratoru. Administrator odlučuje o prihvatanju ili odbijanju pojedinih zahtjeva. U slučaju

prihvaćanja zahtjeva, sustav zahtjevine termine dodjeljuje uredniku pohranjujući informacije o njima u bazu podataka, briše odobreni zahtjev iz baze te dojavljuje administratoru poruku o uspjehu. U slučaju odbijanja zahtjeva sustav briše zahtjev iz baze podataka i šalje obavijest o uspjehu administratoru.

Dijagram stanja

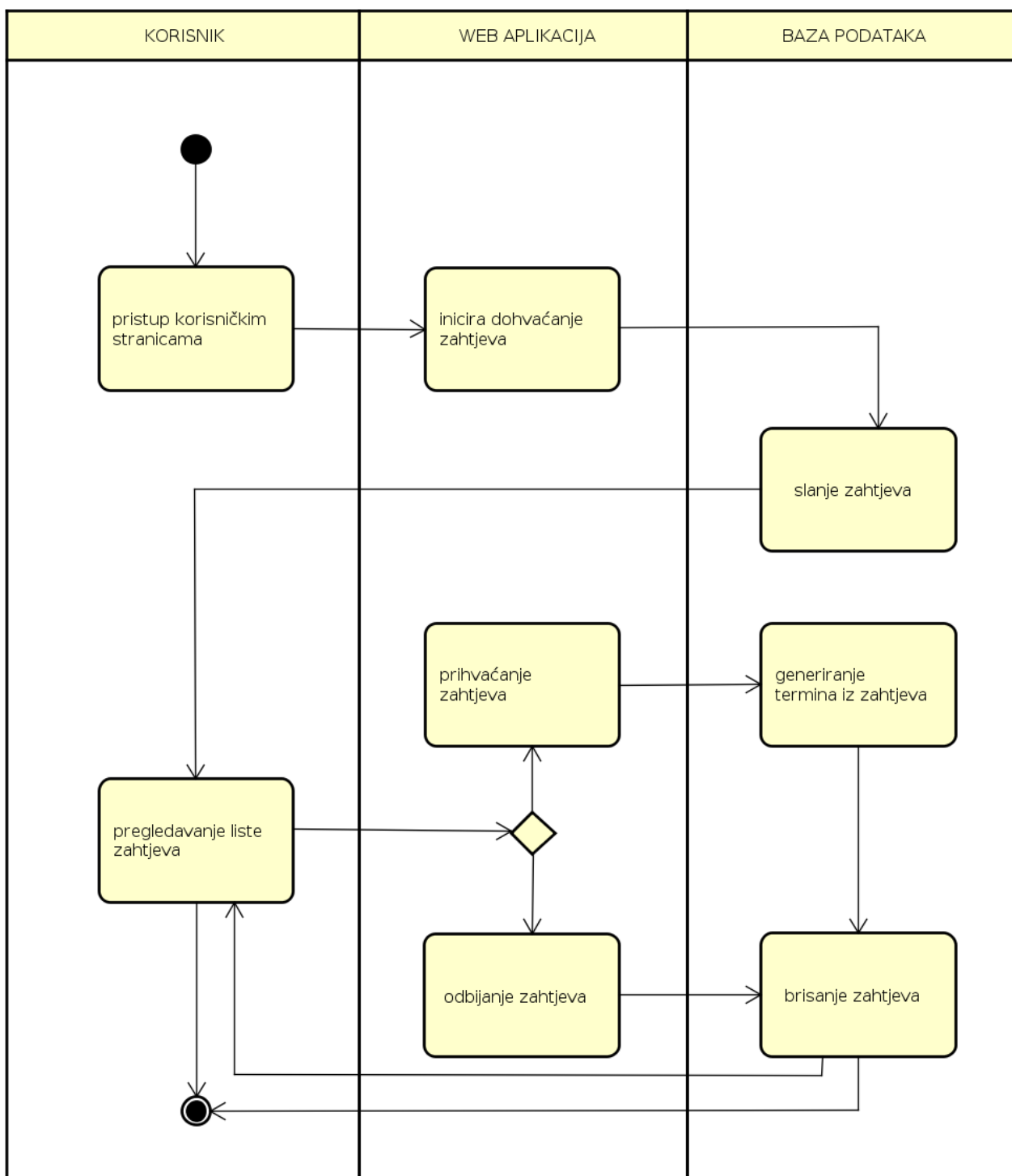
Dijagram prikazuje proces reprodukcije programa radiopostaje na početnoj stranici aplikacije na zahtjev korisnika.

Dolaskom na početnu stranicu radiopostaje korisnik, neovisno o tome je li registriran ili nije, ima mogućnost slušanja programa radiopostaje, točnije zvučnog zapisa koji se trenutno emitira. Aplikacija mu prikazuje informacije o tom zapisu, i odabirom opcije slušanja programa postaje, aplikacija dohvaća trenutni zapis sa poslužitelja i reproducira ga korisniku u njegovom pregledniku. Za vrijeme slušanja zapisa korisniku je dostupna mogućnost mijenjanja glasnoće, kao i prekidanja cijele reprodukcije. Kada reproduciranje trenutnog zapisa završi, aplikacija pronalazi idući zapis na listi i nastavlja reprodukciju puštanjem tog novog zapisa. Korisnik u bilo kojem trenutku može odabrati opciju izlaska iz aplikacije prilikom koje će se ona zatvoriti, i reprodukcija programa će naravno završiti.



Slika 7: Dijagram stanja za UC21:SlušajRadioPostaju

Dijagram aktivnosti



Slika 8: Dijagram aktivnosti za UC16: OdlučiOZahtjevuZaTerminom

Dijagram aktivnosti koristi se za modeliranje dinamičkih aspekata sustava. Na ovom je dijagramu prikazan slijed interakcija korisnika administratora, web aplikacije, i njene baze podataka tokom uređivanja liste zahtjeva za terminima, koju su prethodno generirali glazbeni urednici. Lista zahtjeva pohranjena je u bazi i web aplikacija je dohvaća na zahtjev administratora. Administrator pregledava listu, te može prihvatiti ili odbiti pojedine zahtjeve.

Prihvaćanjem zahtjeva u bazu se sprema informacija o odobrenim terminima i uredniku zaduženom za te termine. Nakon prihvaćanja ili odbijanja zahtjev se briše s liste. Administrator u bilo kojem trenutku može završiti rad.

Dijagram komponenti

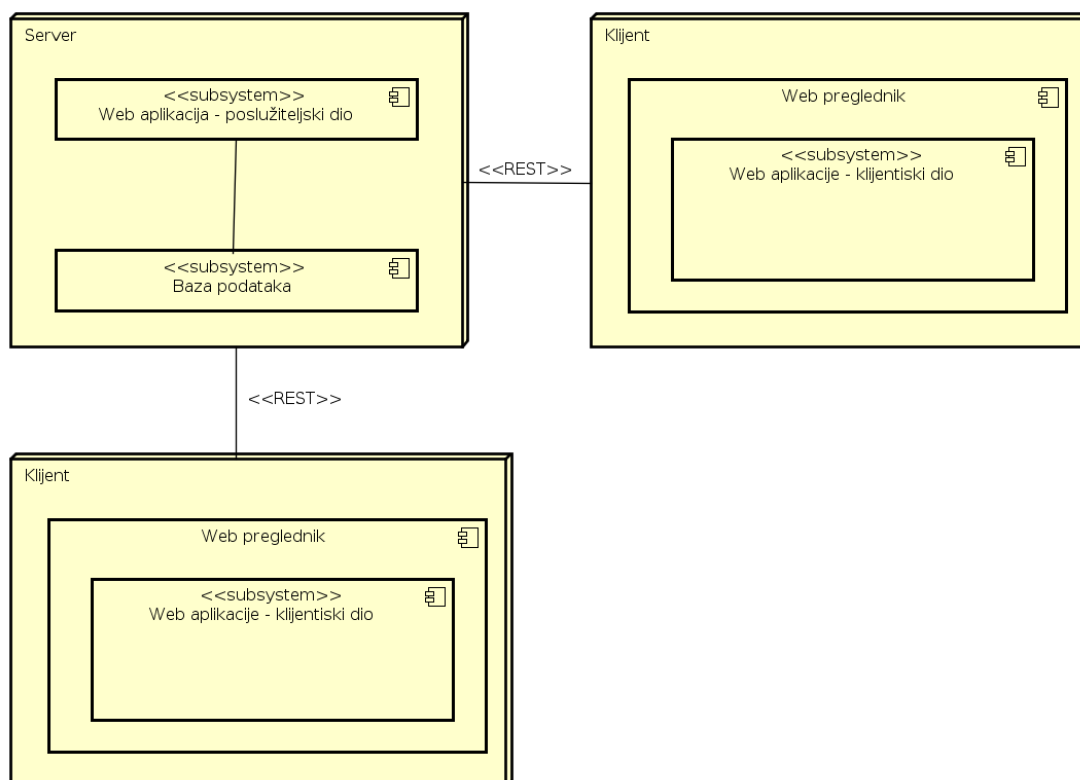
- tena/edi

7. Implementacija i korisničko sučelje

7.1. Dijagram razmještaja

Ova je web aplikacija u osnovi razdvojena na dva dijela, klijentski i poslužiteljski. Na poslužitelju se izvršava veći dio aplikacije, prvenstveno aplikacijska logika koja ostvaruje propisane funkcionalnosti sustava, a na njemu su pohranjeni i svi podaci aplikacije – zvučni zapisi kao datoteke na disku, a ostali podaci u SQLite bazi podataka. Poslužiteljski dio aplikacije izvršava se kao samostalan program, koji u sebi sadrži osnovni web server (u sklopu Flask frameworka)

Klijentski dio aplikacije izvršava se u web pregledniku svakog od korisnika aplikacije i uključuje prvenstveno prezentaciju podataka i interakciju s korisnicima. Komunikacija između jednog poslužitelja i proizvoljnog broja klijenata odvija se putem HTTP protokola, korištenjem REST zahtjeva.



Slika 1: Dijagram razmještaja aplikacije

7.2. Korištene tehnologije i alati

Za izradu cjelokupne aplikacije korišteno je više programskih jezika i različitih tehnologija.

Dizajn

Za izradu dizajna web aplikacije korišteni su osnovni jezici *HTML* i *CSS*, kao i njihova moderna proširenja *Jade* i *SASS*, koja se prije pokretanja aplikacije prevode u *HTML* odnosno *CSS*.

Klijentski dio

Klijentski dio aplikacije, izuzev dizajna, napravljen je u programskom jeziku *TypeScript*, koji se također prije pokretanja aplikacije prevodi, u *JavaScript* kojeg podržavaju svi moderni preglednici. Za izradu aplikacije korišten je *Angular2* framework te nekoliko pomoćnih biblioteka poput *SystemJS*, *RxJS* i *ES6-shim* za razne stvari. U izradi dizajna korišteni su *SASS* i *Jade*, jezici koji dizajniranje web stranica dosta pojednostavljaju, a koji se prije pokretanja aplikacije prevode u *CSS* i *HTML*.

Poslužiteljski dio

Poslužiteljski dio aplikacije izrađen je u programskom jeziku *Python* i njegovu web frameworku *Flask*. Za komunikaciju s i upravljanje bazom podataka koristi se biblioteka *Peewee*, a za slanje email poruka *Flask* ekstenzija *Flask-Mail*. Korištena baza podataka je *SQLite*, koja je ugrađena u programski jezik *Python*.

Alati

Kao pomoć u izradi ove aplikacija korišteni su alati *pip* i *npm* za instalaciju potrebnih dodataka, te alat *grunt* za automatizaciju prevođenja *SASS*, *Jade* i *TypeScript* datoteka. Nije korištena niti jedna posebna razvojna okolina, već samo napredniji urednici teksta poput *Sublime Texta* i *Atoma*.

7.3. Isječak programskog koda vezan za temeljnu funkcionalnost sustava

Modeli

Programskim kodom na *slici 2* definira se model zvučnog zapisa koji se pohranjuje u bazu podataka: podaci koji ga sačinjavaju, njihovi tipovi i svojstva (npr. nužnost). Uz to, u razredu modela defininiraju se i neke od ključnih operacija koje se nad instancom tog modela mogu izvršavati.

```
17 class Track( BaseModel ):
18     """Track model
19
20     This model is a track representation, containing it's various
21     metadata. The track itself is stored on the server as a music
22     file, whose location is stored in the `path` field.
23     `name`, `path`, `artist` and `duration` fields are mandatory.
24     """
25     title          = CharField()
26     path           = CharField()
27     artist         = CharField()
28     album          = CharField( null = True )
29     duration       = IntegerField()
30     file_format    = CharField( null = True )
31     sample_rate    = FloatField( null = True )
32     bits_per_sample = IntegerField( null = True )
33     genre          = CharField( null = True )
34     publisher      = CharField( null = True )
35     carrier_type   = CharField( null = True )
36     year           = IntegerField( null = True )
```

Slika 2: Isječak koda koji definira model zvučnog zapisa

Primjer jedne takve operacije dan je na *slici 3*. Programski kod na toj slici služi za dohvaćanje trenutno reproduciranog zvučnog zapisa u programu radiopostaje. To čini tako što prvo dohvaća trenutni termin (koji uvijek počinje na puni sat), i potom prolazi po

svim zvučnim zapisima koji se nalaze na listi za reprodukciju za taj termin i iz dostupnih podataka računa u kojem trenutku njihova reprodukcija završava. Prvi zapis koji završava u nekom trenutku kasnijem od trenutnog (`datetime.now()`) jest onaj koji se upravo reproducira, te ga metoda vraća pozivatelju, zajedno s informacijom o trenutnoj poziciji reprodukcije unutar zapisa te urednikom čiji je to termin.

```
62     @classmethod
63     def get_currently_playing( cls ):
64         """Returns the currently playing track and the editor who selected it
65
66         Raises DoesNotExist, IndexError
67         """
68         current_time = datetime.now()
69         start_time = current_time.replace( minute = 0, second = 0, microsecond = 0 )
70         slot = Slot.get( Slot.time == start_time )
71         playlist = iter( slot.get_playlist().order_by( PlaylistTrack.index ) )
72         ptrack = next( playlist )
73         try:
74             while True:
75                 if start_time > current_time - timedelta( seconds = ptrack.duration ):
76                     return ptrack, ( current_time - start_time ).total_seconds(), slot.editor
77                 ptrack = next( playlist )
78                 start_time += timedelta( seconds = ptrack.duration )
79         except StopIteration:
80             raise IndexError
```

Slika 3: Isječak programskog koda za dohvaćanje trenutno reproduciranog zapisa

Upravitelji

Isječak koda na slici 4 odgovara na *REST* upit za dohvaćanjem popisa svih korisnika i urednika registriranih u sustavu. Za taj upit definirana je posebna putanja na serveru, `/admin/users/list`, i njoj se pristupa isključivo **HTTP GET** zahtjevima. Da bi dobio odgovor, korisnik mora biti prijavljen, što provjerava dekorator `@login_required`. Podaci o ID-ju trenutno prijavljenog korisnika pohranjeni su u session cookie-ju, i prije obrade svakog zahtjeva podaci o tom korisniku dohvaćaju se iz baze podataka.

Dohvaćanje popisa korisnika vrši se pomoću metode definirane unutar modela **User**, koja prije dohvaćanja traženih podataka vrši provjeru razine ovlasti trenutnog korisnika – ova

je operacija ograničena na administratore sustava. Nakon provjere i dohvaćanja, podaci se strukturiraju u rječnik i vraćaju kao *JSON* odgovor. Ako provjera ovlasti nije zadovoljena, bit će uhvaćena iznimka **AuthorizationError**, te će se umjesto traženih podataka kao odgovor vratiti poruka o pogrešci, zajedno s odgovarajućim *HTTP* statusnim kodom. Isto će se dogoditi i u slučaju neke druge, nedefinirane iznimke.

```
681 @app.route( '/admin/users/list', methods = [ 'GET' ] )
682 @login_required
683 def list_users():
684     """Returns a list of all the users (excluding admins and owner)
685
686     Returns a list of dicts { id, first_name, last_name, occupation, year_of_birth, email, account_type }
687     representing users.
688     No request params.
689     """
690     try:
691         users = g.user.get_all_users()
692         data = [{
693             'id'           : user.id,
694             'first_name'    : user.first_name,
695             'last_name'     : user.last_name,
696             'occupation'    : user.occupation,
697             'year_of_birth' : user.year_of_birth,
698             'email'         : user.email,
699             'account_type'  : user.account_type
700         } for user in users ]
701         return data_response( data )
702     except AuthorizationError:
703         return error_response( 'Neuspješno dohvaćanje popisa korisnika: Nedovoljne ovlasti.', 403 )
704     except:
705         return error_response( 'Neuspješno dohvaćanje popisa korisnika: Nevaljan zahtjev.' )
```

Slika 4: Isječak koda koji vraća popis korisnika sustava

Na slici 5 prikazan je isječak koda koji vrši odobravanje uredničkog zahtjeva za terminima kao odgovor na *REST* zahtjev. Za tu je mogućnost također definirana posebna putanja na serveru, `/admin/requests/<int:request_id>/allow`, unutar koje se nalazi i parametar koji određuje koji se zahtjev treba odobriti. Dozvoljena metoda pristupa je isključivo **HTTP POST**, jer se kao rezultat ovog zahtjeva očekuje promjena stanja podataka na poslužitelju. Ovaj zahtjev može podnijeti samo prijavljeni korisnik, što provjerava `@login_required`, i to samo administrator sustava. Provjera administratorskih ovlasti vrši se unutar `allow_request()` metode *User* modela, koja zapravo ujedno i izvršava samo odobravanje zahtjeva. Ako je odobravanje uspješno, kao odgovor se vraća poruka o uspjehu, dok se u suprotnom, ovisno o uočenoj pogrešci – iznimci – vraća odgovarajuća

poruka o istoj.

```
643 @app.route( '/admin/requests/<int:request_id>/allow', methods = [ 'POST' ] )
644 @login_required
645 def allow_request( request_id ):
646     """Allows a given request for slots
647
648     No request params.
649     """
650     try:
651         g.user.allow_request( request_id )
652         return success_response( 'Zahtjev uspješno odobren.' )
653     except AuthorizationError:
654         return error_response( 'Neuspješno odobravanje zahtjeva: Nedovoljne ovlasti.', 403 )
655     except DoesNotExist:
656         return error_response( 'Neuspješno odobravanje zahtjeva: Ne postoji zahtjev s danim ID-om.', 404 )
657     except peewee.IntegrityError:
658         return error_response( 'Neuspješno odobravanje zahtjeva: Preklapanje s već postojećim terminom.', 409 )
659     except Exception as e:
660         return error_response( 'Neuspješno odobravanje zahtjeva: Nevaljan zahtjev.' )
```

Slika 5: Isječak koda koji obavlja potvrđivanje zahtjeva za terminom

Klijentski dio

Klijentski je dio ove aplikacije napisan u jeziku *TypeScript*, proširenju *JavaScripta*, i frameworku *Angular2*. Sastoji se od više komponenti, svaka sa svojom specifičnom ulogom, koje se povezuju u cjelinu i čine jedinstvenu aplikaciju. Svaka komponenta ima svoj način prikaza, koji je opisan *HTML*-om, i karakteristične operacije koje se za nju definiraju.

Na slici 6 prikazana je jedna takva komponenta, *ManageUsers*, koja administratorima omogućuje pregled i upravljanje korisnicima radiopostaje. Način njenog prikaza definiran je u datoteci `./dest/views/manageUsers/manageUsers.html`, što je navedeno pomoću atributa `templateUrl` u opisniku komponente, dekoratoru `@Component`. Za izvršavanje operacija dohvaćanja popisa korisnika, brisanja i uređivanja koristi se *AJAX* način slanja zahtjeva na poslužitelj. To omogućava razred *Http*, koji pruža mogućnost slanja **GET** i **POST** zahtjeva na odabrani *URL*, i naknadnu obradu dobivenih rezultata. Kako je ova aplikacija *Single-Page App* – ni u jednom trenutku u radu stranica se ne učitava ponovno u cijelosti, navigaciju između njenih dijelova potrebno je riješiti na poseban način, i to je omogućeno od strane razreda *Router*.

```
1  import {View, Component} from 'angular2/core';
2  import {Location, RouteConfig, RouterLink, Router, CanActivate} from 'angular2/router';
3  import { Http } from 'angular2/http';
4
5  @Component({
6    selector: 'ManageUsers',
7    templateUrl: './dest/views/manageUsers/manageUsers.html'
8  })
9  export class ManageUsers {
10    http: Http;
11    router: Router;
12    users: any[];
13    editable: boolean = false;
14
15    toggleEditable() { this.editable = !this.editable; }
16    constructor(http: Http, router: Router) {
17      this.http = http;
18      this.router = router;
19      http.get('/admin/users/list').map((res) => res.json()).subscribe((res) => {
20        console.log(res);
21        this.users = new Array();
22        for (let i in res.data) this.users.push(res.data[i]);
23      })
24    }
25    editUser(userId) { this.router.navigate(['/Settings', 'EditUser', { userId: userId }]); }
26    deleteUser(userId) {
27      for (let i in this.users) {
28        if (this.users[i].id.toString() === userId.toString()) {
29          this.users.splice(i, 1);
```

Slika 6: Isječak koda koji omogućuje upravljanje korisnicima

7.4. Ispitivanje programskog rješenja

Ispit 1: Registracija korisnika

Očekivanje:

Neregistrirani korisnik se želi registrirati. Upisuje svoje podatke na početnoj stranici zajedno s lozinkom i mailom. Ako je mail ispravan i neiskorišten sustav šalje mail korisniku za potvrdu.

TIJEK IZVOĐENJA:

1. Upisivanje korisničkih podataka, uključujući email adresu i lozinku
2. Klik na gumb *Registriraj se*
3. Potvrđivanje prijave preko linka primljenog u email poruci

Rezultat:

Ispitivanje je uspješno provedeno, ostvareni su očekivani rezultati. U slučaju neispravno unesenih podataka, sustav prepoznaje problem i reagira na odgovarajuć način.

Ispit 2.: Postavljanje urednika za administratora

Očekivanje:

Vlasnik za administratora može postaviti samo registrirane obične korisnike. U slučaju da za administratora želi postaviti urednika, sustav mu to ne bi smio ponuditi niti dozvoliti.

TIJEK IZVOĐENJA:

1. Odabir mogućnosti uređivanja administratora
2. Odabir mogućnosti *Uredi popis administratora*
3. Upisivanje imena urednika u tražilicu

Rezultat:

Među ponuđenim korisnicima za postavljanje za administratora nisu urednici sustava, kako i treba biti.

Ispit 3: Pokušaj zaobilaska ograničenja ovlasti**Očekivanje:**

Korisnik želi pristupiti mogućnostima koje mu prema razini ovlasti nisu dostupne i stoga nisu ni prikazane na sučelju aplikacije. Znajući rad sustava, korisnik ručno stvara HTTP zahtjev na proslužitelj kako bi izveo operacije koje mu prema razini prava nisu dozvoljene – dohvaćanje podataka o pojedinom korisniku, dohvaćanje popisa administratora i sl. Sustav bi trebao prepoznati pokušaj zaobilaženja ograničenja i odgovoriti na primjeren način.

TIJEK IZVOĐENJA:

1. Korisnik kreira zahtjev koji pokušava zaobići ograničenja sustava
2. Korisnik šalje zahtjev i pregledava odgovor

Rezultat:

Sustav prepoznaje da korisnik nema ovlasti pristupiti toj funkcionalnosti i odgovara porukom: *"Neuspješno XYZ: Nedovoljne ovlasti"* (XYZ je pokušana akcija).

Ispit 4: Provjera 24-satnog ograničenja na potvrđivanje liste želja**Očekivanje:**

Između dva potvrđivanje liste želja pojedinog korisnika mora proći najmanje 24 sata. Ako korisnik ipak pokuša potvrditi listu želja unutar 24 sata od prethodnog potvrđivanja, sustav mu to ne smije dopustiti.

TIJEK IZVOĐENJA:

1. Otvaranje stranice postavki
2. Odabir mogućnosti *Lista želja*
3. Odabir akcije *Potvrdi listu želja*
4. Obavljanje nekih drugih akcija ili čekanje
5. Nakon nekoliko minuta, ponavljanje koraka 1-3

Rezultat:

Akcija *Potvrdi listu želja* u drugom pokušaju nije omogućena.

Ispit 5: Stvaranje liste za reprodukciju za određeni termin**Očekivanje:**

Urednik radi listu zvučnih zapisa za reprodukciju u svom terminu. Prilikom odabira zapisa sustav bilježi ukupno trajanje liste za reprodukciju. Ako trajanje liste za reprodukciju premaši jedan sat, dodavanje daljnjih zapisa ne smije biti dozvoljeno. Isto tako, ne smije se dozvoliti da zadnji zapis na listi započne unutar 15 sekundi od kraja termina.

TIJEK IZVOĐENJA:

1. Otvaranje stranice postavki
2. Odabir mogućnosti *Termini za reprodukciju*
3. Odabir termina za reprodukciju
4. Sastavljanje liste za reprodukciju

Rezultat:

???

7.5. Upute za instalaciju

Kratke upute u ovoj dokumentaciji odnose se na instalaciju web aplikacije na vlastito računalo ili server i korištenje iste uz interni Flask web server. Korištenje drugih poslužitelja, poput *Apachea* ili *Nginx-a*, također je moguće, no nije pokriveno u ovim uputama.

Preduvjeti

Za uspješnu instalaciju i korištenje aplikacije **nužno** je imati instaliran *Python* programski jezik i okolinu, najmanje verziju 3.4. Upute za instalaciju Pythona mogu se pronaći na stranici: <https://www.python.org/> U njega je uključena i SQLite baza podataka, pa nju nije potrebno zasebno instalirati.

Dohvaćanje i instalacija

Instalaciju ove aplikacije najlakše je provesti preuzimanjem s GitHub repozitorija na web lokaciji: <https://github.com/zjurelinac/FM-Radio>. To je moguće učiniti pozicioniranjem u odabrani direktorij za instalaciju i kloniranjem repozitorija naredbama:

```
mkdir aplikacija
cd aplikacija
git clone https://github.com/zjurelinac/FM-Radio
```

Ovime je stvoren direktorij **FM-Radio**, unutar kojeg se nalazi kod aplikacije. Sada je potrebno pozicionirati se u taj direktorij i instalirati sve potrebne biblioteke:

```
cd FM-Radio
pip install -r requirements.txt
```

Moguće je, ovisno o okolnostima, da će za drugu naredbu biti potrebne administratorske ovlasti. Također, ako imate instalirane i Python 2 i Python 3, pripazite da koristite **pip** za Python 3.

Postavljanje

Da bi se web aplikacija učinila sposobnom za rad, potrebno je prvo urediti odgovarajuće postavke. One se nalaze u datoteci **config.py**, i to su redom:

```
# Temeljna adresa aplikacije, postaviti na vlastitu mrežnu lokaciju
APPLICATION_ROOT = 'localhost/'

# Postavke koje definiraju razvojno stanje aplikacije, za
# produkcijsku varijantu oboje postaviti na False
DEBUG = False
TESTING = False

# Sigurnosne postavke aplikacije - popis email adresa osoba
# odgovornih za sigurnost aplikacije, te tajni ključ koji se koristi
# za šifriranje session cookie-a
ADMINS = frozenset(['admin@mail.com'])
SECRET_KEY = '$j#e&7+*2w2y)0if$c-gvlf$^%@)q)7$(gv@6xk%*^o9r^1u1n'

# Postavke spremanja podataka - lokacija baze podataka i direktorija
# u kojeg se pohranjuju zvučni zapisi
DATABASE = 'radio.db'
UPLOAD_FOLDER = "/dest/static/audio"

# Postavke mail servera - koristi se za slanje aktivacijskih linkova
# i ostalih informacija korisnicima sustava
MAIL_SERVER = "mail.server.com"
MAIL_PORT = 465
MAIL_USE_SSL = True
MAIL_USERNAME = "user_name"
MAIL_PASSWORD = "password_plain_text"
MAIL_DEFAULT_SENDER = "email@server.com"
```

Ove postavke promijenite prema vlastitim željama i potrebama.

U ovom trenutku sama aplikacija je spremna za pokretanje, no u bazi podataka još nisu stvorene odgovarajuće tablice niti su u nju pohranjeni ikakvi podaci. Stoga je prvo potrebno kreirati tablice sljedećom naredbom:

```
python init.py
```

Da bi aplikacija ispravno funkcionirala, u njoj mora postojati korisnik-vlasnik, kojeg je potrebno kreirati ručno. To se može učiniti pokretanjem modificirane *Python* konzole naredbom `./shell.py` (za Linux operacijski sustav) ili `python shell.py` za Windows. U otvorenu konzolu potrebno je upisati sljedeće:

```
# Umjesto 'Ime', 'Prezime' i ostalog unesite prave podatke vlasnika
>>> user = User.create_user( 'Ime', 'Prezime', 'zanimanje',
godina_rođenja, 'email@adresa.com', 'lozinka' )

>>> user.activated = True

>>> user.account_type = AccountType.OWNER
```

Time je stvoren novi korisnik koji je ujedno i postavljen za vlasnika postaje. No, u bazi još uvijek ne postoje nikakvi podaci o postaji, i njih je također potrebno inicijalno ručno unijeti (poslije ih je moguće mijenjati unutar aplikacije). To se može učiniti na sljedeći način:

```
>>> RadioStation.insert( name = 'Ime', description = 'Opis', oib =
'oib', email = 'email@address.com', address = 'Adresa', frequency =
100 ).execute()
```

I ovim korakom aplikacija je uspješno postavljena, te ju je sada moguće pokrenuti

naredbom:

```
python run.py          # Za ovo su potrebne administratorske ovlasti
```

čime je aplikacija pokrenuta i spremna za usluživanje korisnike.

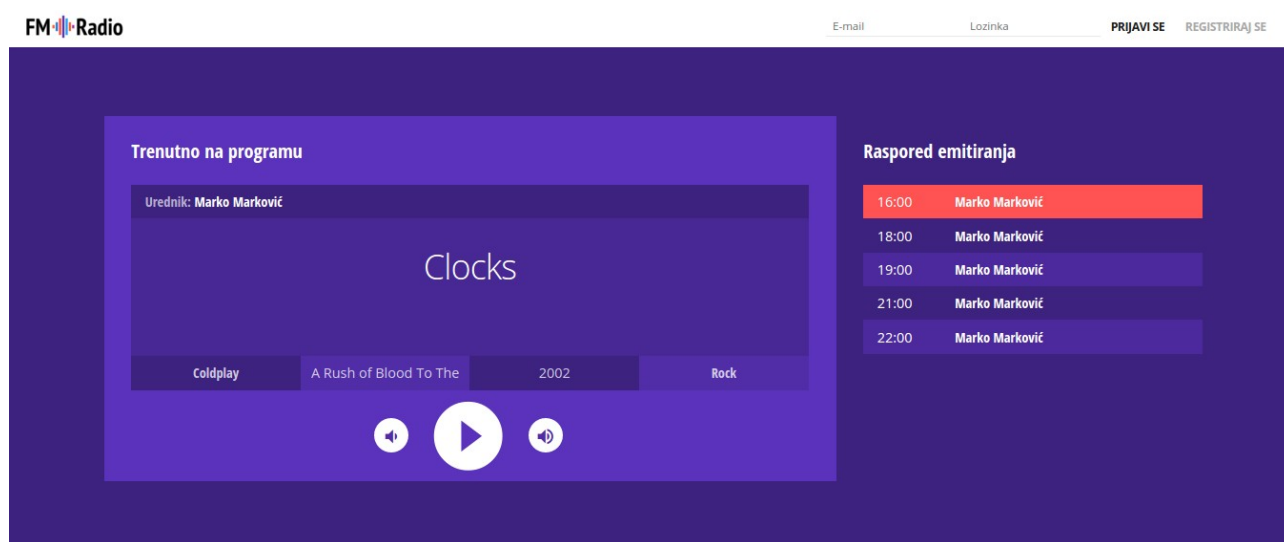
7.6. Korisničke upute

- tena, edi, peros, tonko

Općenito

Početna stranica

Slušanje programa radiopostaje



Slika 7: Prikaz odjeljka za slušanje programa na početnoj stranici radiopostaje

Registracija i prijava


FM Radio

E-mailLozinkaPRIJAVI SEREGISTRIRAJ SE

Postani dio FM Radija


Registriraj se, postani korisnik **FM Radija** i sudjeluj u njegovom radu. Tvoje ideje, želje i prijedlozi mogu pomoći kako bi tvoj omiljeni radio postao još i bolji!

Kao korisnik, možeš biti:




Registrirani korisnik

Reci nam što želiš slušati i pomoz nam da program učinimo još boljim!



Glazbeni urednik

Pomoz nam obogatiti program sastavljanjem vlastite glazbene emisije!



Administrator

Pomoz nam upravljati pjesmama, urednicima, korisnicima i terminima!

Registriraj se

Ime

Prezime

E-mail adresa

Lozinka

Ponovljena lozinka

Godina rođenja

Zanimanje

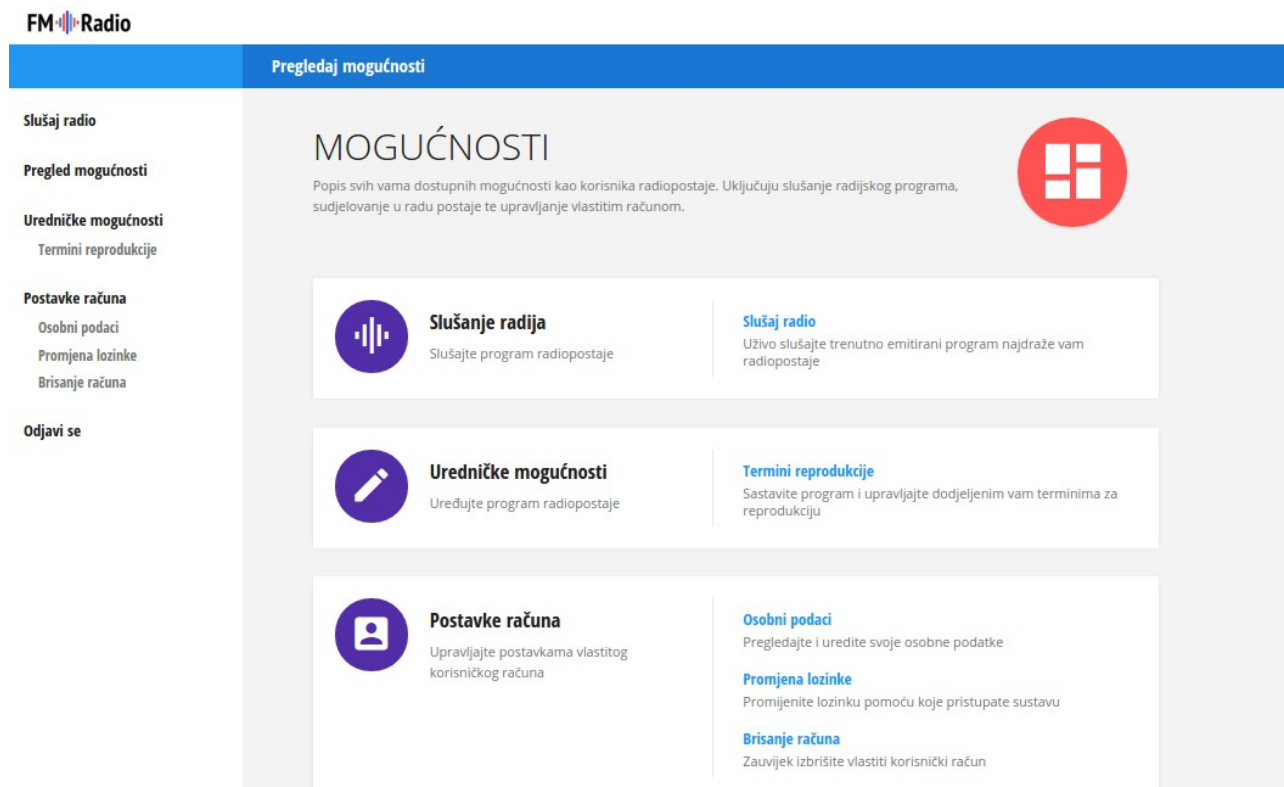
Klikom na **REGISTRIRAJ SE!** prihvaćate **uvjete korištenja** ove aplikacije.

REGISTRIRAJ SE!

Slika 8: Prikaz obrazaca za prijavu i registraciju u sustav

Svi korisnici

Pregled mogućnosti



Slika 9: Prikaz stranice pregleda mogućnosti

Uređivanje osobnih podataka

Promjena lozinke

Brisanje korisničkog računa

Urednik

Pregled termina za reprodukciju

Sastavljanje liste za reprodukciju

Administrator

Upravljanje zvučnim zapisima

Upravljanje urednicima

Odlučivanje o zahtjevima za terminima

Upravljanje korisnicima

Pregled statistika

Vlasnik

Upravljanje administratorima

Uređivanje podataka o postaji

Pregled statistika

8. Zaključak i budući rad

- svi

9. Popis literature

- **Oblikovanje programske potpore, FER**
<http://www.fer.hr/predmet/opp>
- **Zbirka UML** – *Alan Jović, Marko Horvat, Igor Grudenić*, Zagreb 2012.
- **Flask – Python web framework**
<http://flask.pocoo.org/docs/0.10/>
- **Peewee – Python ORM**
<http://docs.peewee-orm.com/en/latest/>
- **Angular2 – Client-side framework (JS & TS)**
<https://angular.io/docs/ts/latest/>
- **SASS – Syntactically Awesome Style Sheets**
<http://sass-lang.com/guide>
- **MDN – Mozilla Developer Network**
<https://developer.mozilla.org/en-US/docs/Web>

Dodatak A: Dnevnik sastajanja

| | |
|--------------------|--|
| 16.10.2015. | Prvi sastanak, rasprava o zadatku |
| 20.10.2015. | Drugi sastanak, nakon konzultacija, okvirni razgovor o podjeli posla i planu rada |
| 01.11.2015. | Treći sastanak |
| 15.11.2015. | Četvrti sastanak, razgovor o tijeku projekta i konkretnim problemima u izradi i njihovom rješavanju |
| 16.11.2015. | Peti sastanak, isto kao i 4. |
| 07.12.2015. | Sastanak prije usmenog ispitivanja, upoznavanje svih članova tima sa svim dijelovima dosad obavljenog posla. |
| 08.12.2015. | Sastanak nakon usmenog ispitivanja, kratki dogovor oko budućeg rada |
| 14.12.2015. | Razrađivanje preostalog posla, rasprava o dizajnu aplikacije – izgledu i funkcionalnosti |
| 22.12.2015. | Dodatni razgovor o preostalom poslu, raspodijela zaduženja na pojedince i dogovor oko rokova |
| 07.01.2015. | Razrada dijagrama i razgovor o preostalim dijelovima dokumentacije |
| 12.01.2015. | Izrada skica dijagrama, komentiranje istih, konačni dogovor oko izvedbe aplikacije |