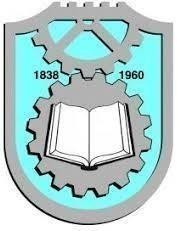
### Univerzitet u Kragujevcu

### Fakultet inženjerskih nauka



Projektovanje informacionih sistema i baza podataka

Tourist Agency

***INSTABOOK***

|  |  |  |
| --- | --- | --- |
| Studenti: | Predmetni profesor: | |
| Mihailo Pantić 604/2019 Prof. dr Milan Erić | |
| Jovan Ivošević 617/2019 Prof. Nenad Pantelić | |
| Sonja Arsić 629/2019 Doc dr Aleksandar Đorđević | |
| Stevan Popović 631/2019 | |

**Sadržaj**

[**1.** **Uvod** 3](#_Toc64216122)

[1.1. Razvoj aplikacije 3](#_Toc64216123)

[1.1.1. MERN stack 3](#_Toc64216123)

[1.1.1. MongoDB 4](#_Toc64216123)

[1.1.1. Express.js 4](#_Toc64216123)

[1.1.1. React 5](#_Toc64216123)

[1.1.1. Node.js 6](#_Toc64216123)

[**2.** **Opis koriščenja aplikacije** 6](#_Toc64216124)

[**3.** **Strukturne napomene, problemi i unapredjenja sajta** 18](#_Toc64216124)

[**4.** **IDEF0 Projektovanje** 19](#_Toc64216124)

[2.1. Dijagram konteksta 19](#_Toc64216125)

[2.2. Stablo aktivnosti 20](#_Toc64216126)

[2.3. Dijagrami dekompozicije 21](#_Toc64216127)

[**5.** **ER dijagram** 25](#_Toc64216128)

[5.1. Entiteti 26](#_Toc64216129)

[5.2. Veze 27](#_Toc64216130)

[5.3. Kompletan ER dijagram 28](#_Toc64216131)

[**6.** **Logička šema relacione baze podataka** 29](#_Toc64216132)

[6.1. Jaki entiteti 29](#_Toc64216133)

[6.2. Gerundi 29](#_Toc64216136)

[6.3. Kompletna logička šema 30](#_Toc64216138)

[**7.** **UML dijagrami** 30](#_Toc64216139)

[7.1. Dijagrami sekvenci 30](#_Toc64216133)

[7.2. Dijagram klasa 36](#_Toc64216136)

1. [**Literatura** 37](#_Toc64216140)
2. ***Uvod***

Danas se računari koriste više nego ikad, informacioni sistem je u stalnom razvoju, a baze podataka čine njegov značajan deo. Digitalno skladištenje ima mnoštvo pogodnosti u pogledu lakog menjanja strukture i sadržaja podataka, kao i brzo umnožavanje podataka tj. kopiranje na druge lokacije.

Pojavom sistema za upravljanje bazama podataka je ceo ovaj proces učinio lakšim za sprovođenje a u isto vreme jako efikasnim u pogledu brzine, sigurnosti, skalabilnosti i drugog.

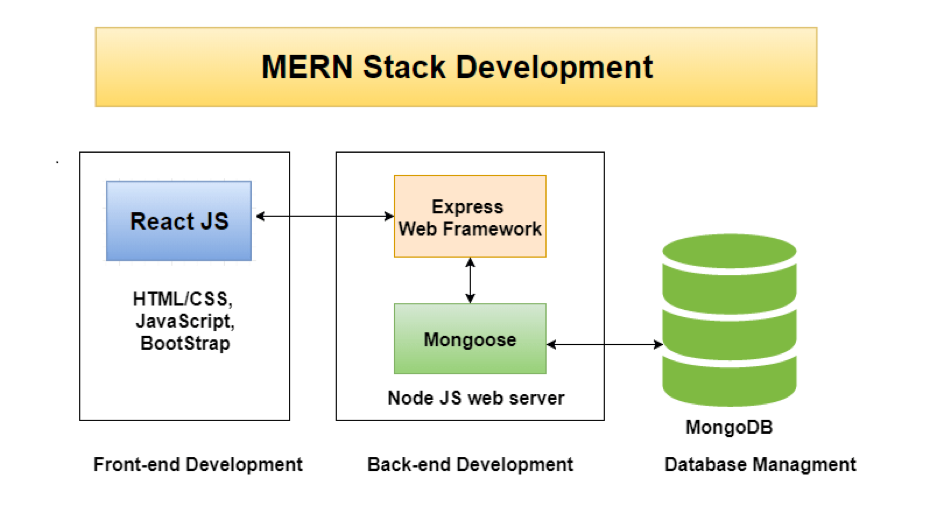
Retko da cemo danas videti neki sistem koji ne koristi neki oblik informacionog sistema za sladištenje i manipulisanje podataka. Poznavanje veštine modelovanja i implementacija baza podataka je krucijalno za efikasan sistem.

U poslednjih desetak godina razvoj i primena tehnologije doneo je različite pogodnosti i nametnuo potrebu korišćenja interneta i web aplikacija u različitim domenima života. Znatno uvećano korišćenje interneta i internet zasnovanih aplikacija uticalo je, kao povratna sprega, i na proces razvoja aplikacija. Kao jedan od najbitnijih zadataka koje programer ima prilikom izrade web aplikacija jeste odabir odgovarajućih tehnologija i alata za razvoj kako bi se ispunili glavni zahtevi web aplikacija kao što su brzina, stabilnost, bezbednost, dinamičnost i dizajn koji je prilagoden korisnicima. Jedan skup tehnologija koji se može koristiti za pravljenje takvih web aplikacija je i MERN (MongoDB , Express , React , Node.js ) stek. Njegov naziv je akronim koji označava skup od četiri tehnologije čija je osnova JavaScript, a koje se razlikuju po nameni. Svaka od ovih tehnologija se može koristiti odvojeno ili u kombinaciji sa nekim drugim tehnologijama, ali zajedno one čine osnovu novog pravca u pravljenju web aplikacija. Node.js ima široku primenu, ali se uglavnom koristi za implementaciju web servera, uz pomoć modula express. MongoDB je NoSQL baza podataka, a React služi za implementaciju vizuelnog dela aplikacije. U radu je dat prikaz tehnika razvoja web aplikacije korišćenjem MERN stek–a. Cilj ovog projekta jeste da objasni modelovanje baze podataka koja zadovoljava zahteve realnog sistema kakav je Agencija za putovanja kao i samu implementaciju baze podataka na MongoDB Atlas serveru i aplikativnog softvera u Node.js okruženju.

# *1.1. Razvoj aplikacije*

*1.1.1. MERN stack*

Rečeno je da u tehnologije MERN steka spadaju Node.js, Express, MongoDB i React ali se pored njih često koristi i neki od drajvera koji služi da omogući upotrebu baze podataka MongoDB u Node.js aplikaciji. Većina ovih tehnologija se može koristiti samostalno. Web aplikacija se može napraviti samo korišćenjem Node.js-a, koji bi imao ulogu i servera i koji bi služio za prikaz HTML stranica. Medutim, Node.js u kontekstu MERN steka predstavlja osnovnu platformu za razvoj koju koriste ostale tehnologije steka. U Node.js platformi se inicijalizuje server pomoću modula express, uz korišćenje drajvera Mongoose za povezivanje sa bazom podataka. Server se kreira tako da prima HTTP zahteve, obrađuje ih i vraća HTTP odgovor. HTTP zahteve šalje React aplikacija, u zavisnosti od akcije korisnika u njoj. Baza podataka MongoDB služi da čuva informacije koje se koriste u web aplikaciji čime se obezbeđuje dinamičnost upotrebe. Kako su ovo zapravo četiri različite tehnologije organizovane u jednu celinu, neophodno je prvo upoznati se sa svakom tehnologijom zasebno.



Slika 1. MERN stack

## *1.1.2. MongoDB*

MongoDB je NoSQL baza podataka. NoSQL je zapravo akronim nastao od izraza Not Only SQL i označava skup baza podataka koje ne prate principe relacionih baza podataka. Među NoSQL bazama postoji podela prema tipu, odnosu načinu na koji se čuvaju podaci u bazi, gde MongoDB spada u baze koje se zasnivaju na modelu dokumenta. To bi značilo da se objekti podataka posmatraju i skladište kao dokumenti. Dokumenti imaju strukturu sličnu JSON formatu, a skup dokumenata, koji imaju istu namenu, naziva se kolekcija. Kolekcija se može posmatrati kao tabela u relacionom modelu baze podataka, samo što dokumenti u okviru kolekcije, mogu ali i ne moraju da imaju iste atribute. Svakom dokumentu se dodeljuje jedinstveni identifikator koji se smešta u polje \_id.

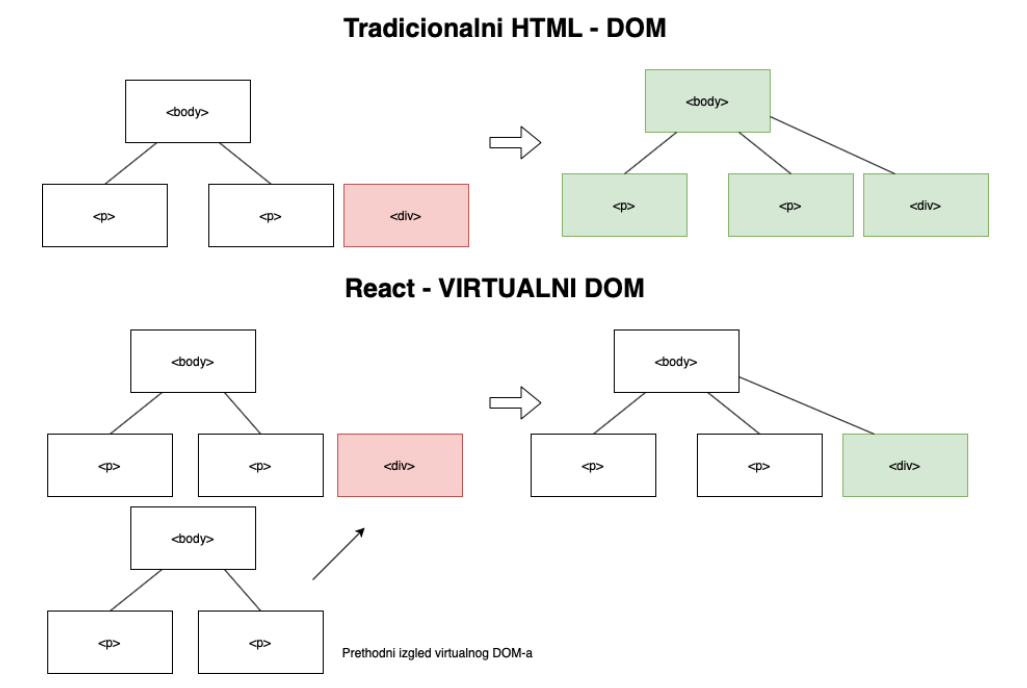
***1.1.3. Express.js***

Iako se celokupna funkcionalnost potrebna za izvođenje koda može implementirati u „čistom“ JavaScript-u pomoću node.js, takav pristup je relativno komplikovan iz razloga što uključeni moduli node.js-a nisu toliko prilagođeni za brzi razvoj koda. Tu u igru dolazi Express.js koji je svojevrsna „nadogradnja“ tog pristupa i uveliko olakšava pisanje koda. Prema odgovoru na pitanje postavljeno na poznatom programerskom portalu Stack Overflow, neke od mogućnosti koje Express pruža su: napredno usmeravanje (eng. routing), moguće je definisati posebne upravljače događaja (eng. event handlers) zavisno o HTTP metodi (GET, POST, PUT, DELETE itd.), omogućuje serviranje statičkih datoteka (HTML, CSS, JavaScript), mnoštvo metoda i svojstava za objekte zahteva (request objekt) i odgovora (response objekt), a i drugo. Neke od navedenih mogućnosti moguće je implementirati u „sirovom“ node.js-u, ali je način komplikovaniji nego da se jednostavno koristi Express.

## *React*

React.js (ili React, ReactJS) je deklarativna biblioteka za izgradnju korisničkih interfejsa. Stičući popularnost od izdavanja 2015. godine pa sve do danas, zahvaljujući svojoj jednostavnosti i deklarativnoj paradigmi, React se sve češće koristi na webu za opisivanje korisničkih interfejsa, osim toga koristi se i u izgradnji interfejsa za mobilne aplikacije (kao zasebna biblioteka po imenu React Native). Deklarativnost biblioteke označava osobinu da je bitan izgled i finalni rezultat gradivnih elemenata interfejsa, a ne i njihova implementacija „ispod haube“, tj. način na koji ćemo doći do prikazanih elemenata nije toliko bitan. Ovakav pristup omogućava brže razvijanje interfejsa, ali i lakše rešavanje grešaka. Osnovni gradivni elementi interfejsa kod React-a su tzv. komponente (eng. components). Komponente predstavljaju male, kompaktne i ponovno iskoristive elemente koji enkapsuliraju svu svoju funkcionalnost unutar sebe. Takođe, upravljaju svojim stanjem (eng. state), a logika (interaktivnost) je pisana u JavaScriptu.

S druge strane, React koristi virtualni DOM što znači da prilikom prvog renderovanja stranice, React u predmemoriji stvara početni izgled celokupnog DOM-a. Svaki element DOM–a ima svoj korespondirajući element u virtualnom DOM-u. Kada se bilo koji element virtualnog DOM-a ažurira ili promeni stanje, React upoređuje novo stanje virtualnog DOM-a sa onim pre samog ažuriranja, te na temelju toga identifikuje element koji se promenio i ponovno prikaže/osveži samo taj element u regularnom DOM-u, štedeći resurse i vreme. Na slici 2. ilustrirana je ova razlika. S leve strane prikazano je početno stanje DOM-a (element s dvoje dece elemenata). Crvenom bojom je naznačen element koji dodajemo u DOM-u, a na desnoj strani slike naznačeni su elementi koji se ponovno renderuju nakon dodavanja elementa u DOM.



Slika 2. Tradicionalni i virtualni DOM

## *1.1.5. Node.js*

Dok se MongoDB brine za perzistenciju podataka, programiranje će biti realizovano koristeći Node.js, JavaScript okolinu u vreme izvršavanja (eng. runtime environment) u kojoj se kod izvršava izvan pretraživača, kao i kod drugih serverskih programskih jezika (PHP, Ruby, Python, Java itd.). Iako je prva verzija Node.js-a puštena 2009. godine, ova okolina još ne uživa toliku upotrebu kao konsolidirani jezici poput PHP-a, kako vreme prolazi, node.js postaje sve zreliji za široku, skalabilnu upotrebu. Jedan od glavnih aduta node.js-a je taj što omogućuje pisanje koda u JavaScript-u, a pošto se JavaScript koristi i za programiranje interaktivnog sloja aplikacije na strani klijenta, nekom programeru je ovo prilično velika „olakšana okolnost“ budući da može programirati u istom jeziku i na strani klijenta i na strani pretraživača. „Ispod haube“ node.js je implementiran na Google-ovom V8 JavaScript engine-u otvorenog koda (koji pokreće i Google Chrome) pa je samim time osigurana pouzdanost, stabilnost i brzina izvođenja koda na strani pretraživača.[[2]](#NPM)

## *2. Opis korišćenja aplikacije*

## *Inicijalizacija projekta*

Inicijalizacija aplikacije obavlja se u koracima koji slede: klonirajte sa GitHub linka <https://github.com/stevislav/tourist-agency> vezu u lokalnu fasciklu na računaru. Otvorite fasciklu koristeći VSCode ili bilo koji uređivač teksta po vašem izboru  i ukucajte sledeću liniju koda u terminalu.

git clone --single-branch -b "main" https://github.com/stevislav/tourist-agency.git .

*Docker Compose*

Compose je alatka za definisanje i pokretanje Docker aplikacija sa više kontejnera. Sa Compose, koristite YAML datoteku da biste konfigurisali usluge vaše aplikacije. Zatim, sa jednom komandom, kreirate i pokrećete sve usluge iz vaše konfiguracije.

Da bismo pokrenuli celu našu aplikaciju zajedno, tj. paralelno pokrenuli sve kontejnere, moramo da konfigurišemo datoteku docker-compose. [*[3]*](#DOCKER)[*[4]*](#fin)

*Kreiranje i pokretanje aplikacije*

Da bismo napravili build za celu aplikaciju, potrebno je da pokrenemo sledeću komandu: *docker-compose build*

Sistem sa više kontejnera možemo pokrenuti pomoću sledeće komande: *docker-compose up*

Konačno, možemo da otvorimo <http://localhost:3000> da vidimo naš React Frontend.

Pozadinski server je aktivan na <http://localhost:8800>

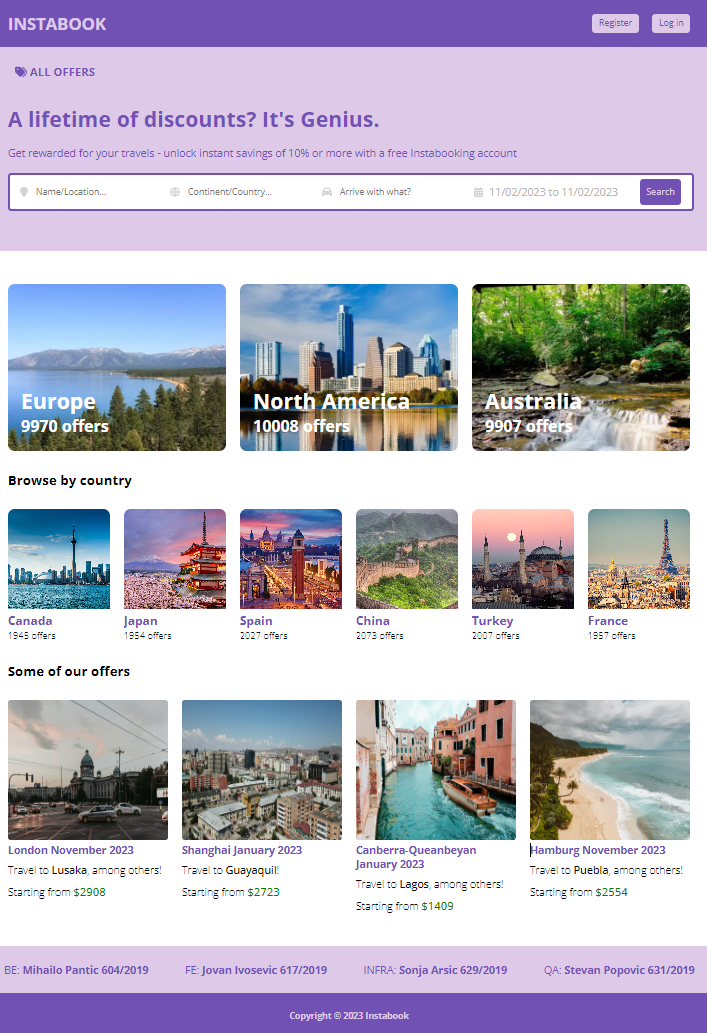
A MongoDB radi na <http://localhost:27017>

U nastavku teksta dat je detaljan opis i dizajn aplikacije.

Prilikom pokretanja aplikacije, na ekranu se pojavljuje početna stranica sajta(slika 3.).

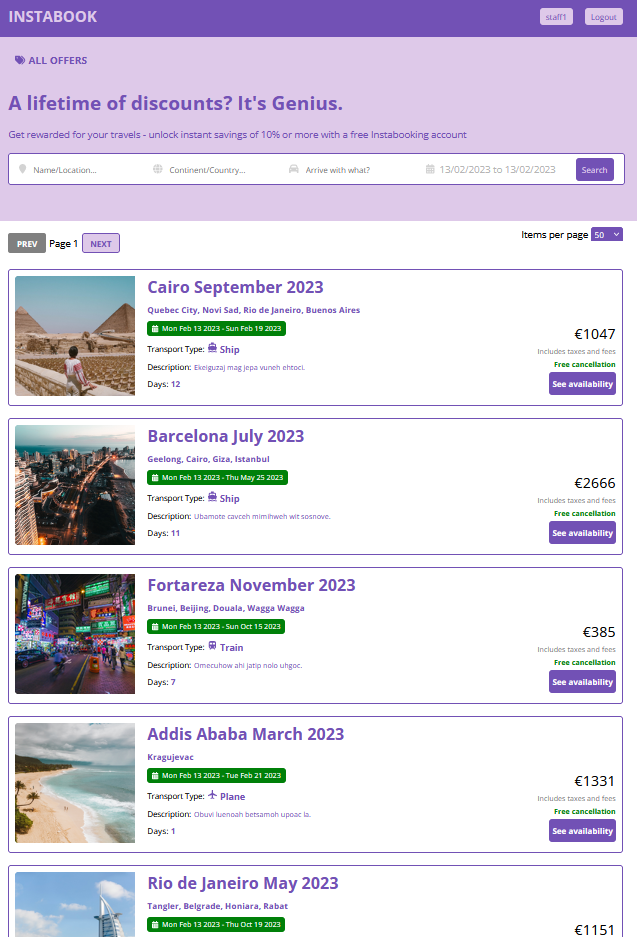
Na početnoj stranici se, pored pojedinog sadržaja koji je izdvojen, daje mogućnost pretrage unošenjem parametara:

* lokacije
* kontinenta ili grada
* načinu prevoza
* datumu odlaska i povratka

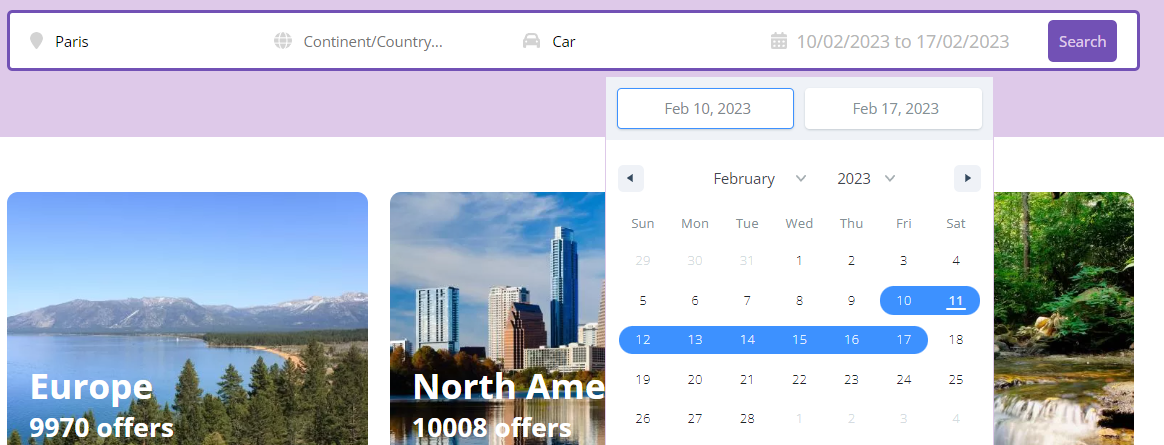


Slika 3. Početna stranica aplikacije

Korisnik klikom na ALL OFFERS može videte celokupnu ponudu aranžmana dostupnih na sajtu(slika 4.).



Slika 4. Lista svih aranžmana

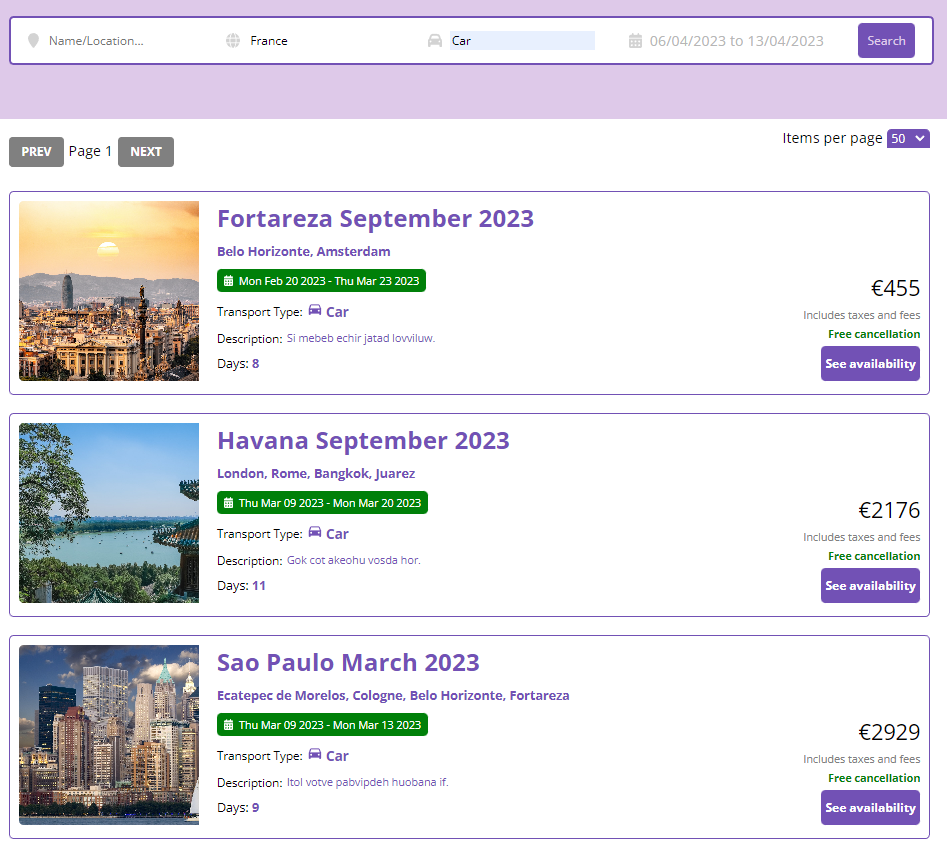


Slika 5. Kriterijumi pretrage

Nakon uspešne pretrage po zadatom kriterijumu, korisniku se prikazuje lista aranžmana po želji. Korisnik može na istoj stranici zatražiti neke druge kriterijume, pregledati aranžmane po želji od datih parametara(slika 5., slika 6. i slika 7).

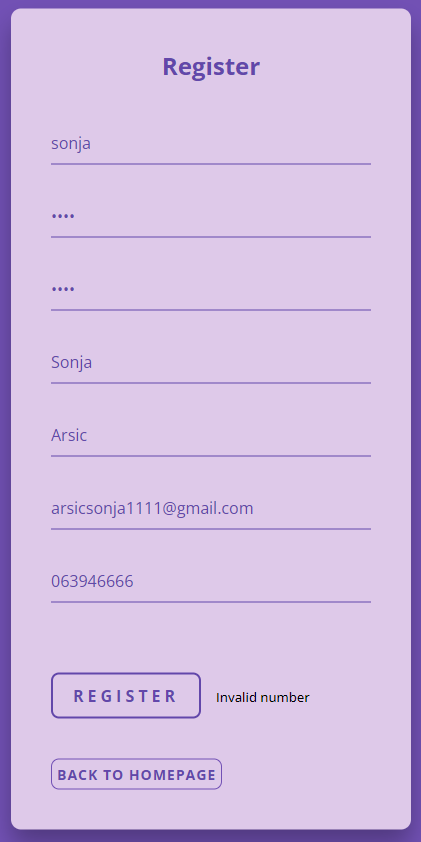


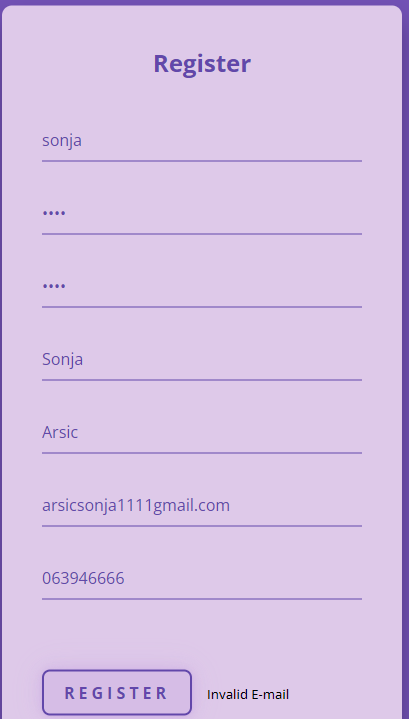
Slika 6. Pretraga aranžmana



Slika 7. Uspesna pretraga aranžmana

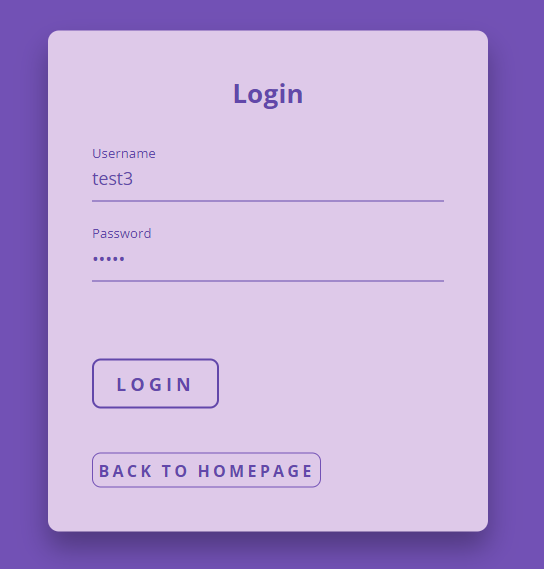
Ukoliko korisnik želi da rezerviše aranžman, moraće da prvo izvrši registraciju i prijavu (slika 8.). Na prikazanoj slici se nalazi forma za registrovanje korisnika. Potrebno je uneti podatke ukoliko korisnik želi da se prijavi kao user ili član staff-a. Ukoliko korisnik unese netačan e-mail ili mobilni telefon, na ekranu se pojavljuje poruka obaveštenja o netačnom podatku.





Slika 8. Forma za registraciju korisnika

Nakon uspešne registracije, korisnik se može prijaviti(slika 9.). Ukoliko prijava uspešno prođe, korisnik će biti u mogućnosti da vidi formu jedinstvenu svojoj ulozi. Različite su forme date ukoliko je korisnik bio admin, user ili član staff-a.

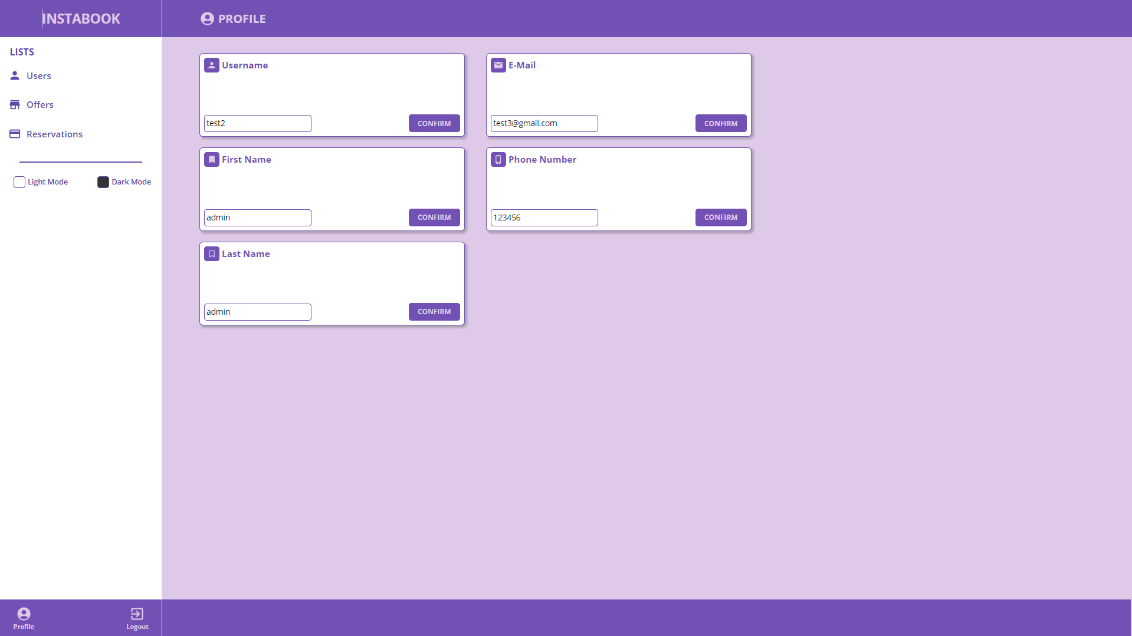


Slika 9. Forma za prijavu korisnika

U aplikaciji postoje tri načina prijavljivanja i to kao:

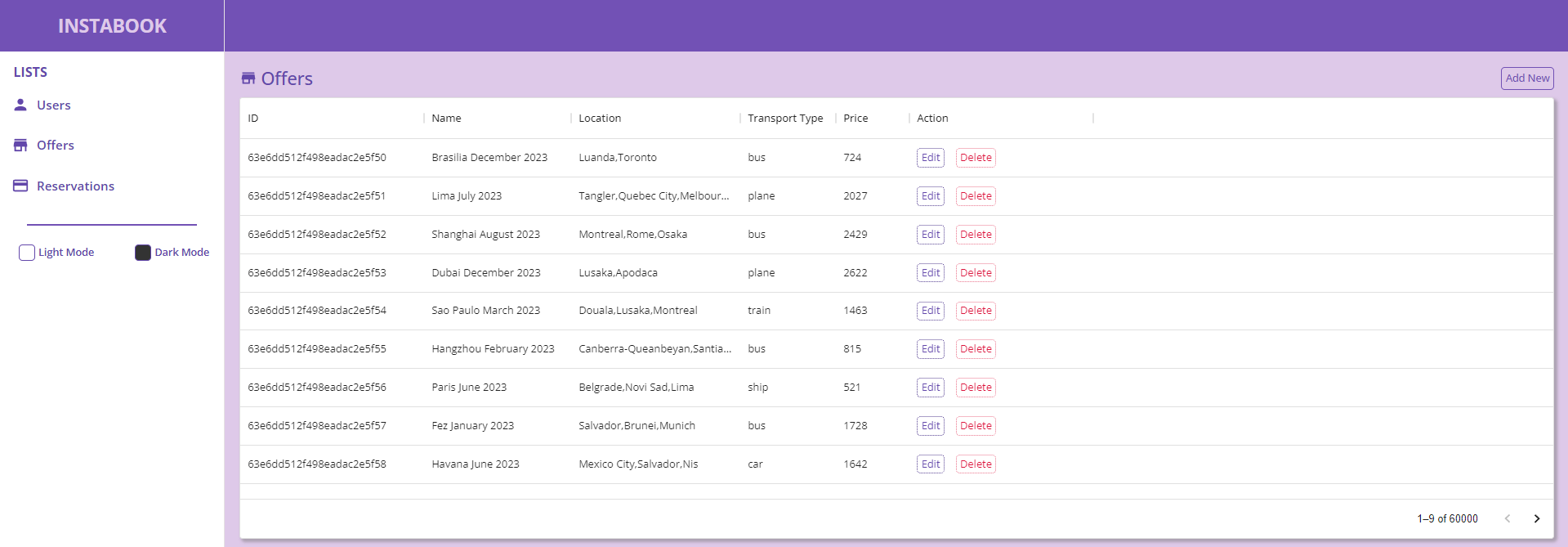
* admin
* user
* staff

Ukoliko se korisnik uspešno prijavi kao admin, na ekranu se pojavljuje forma sa slike 10.

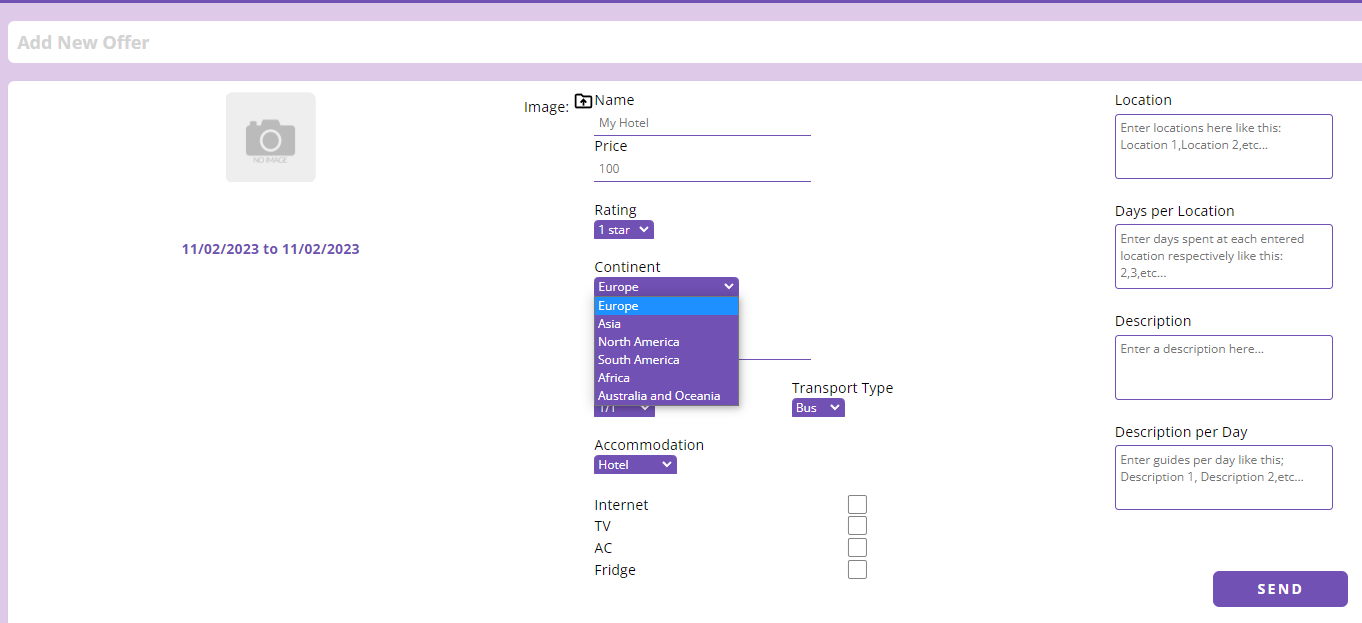


Slika 10. – Kartica “Profile”-ADMIN

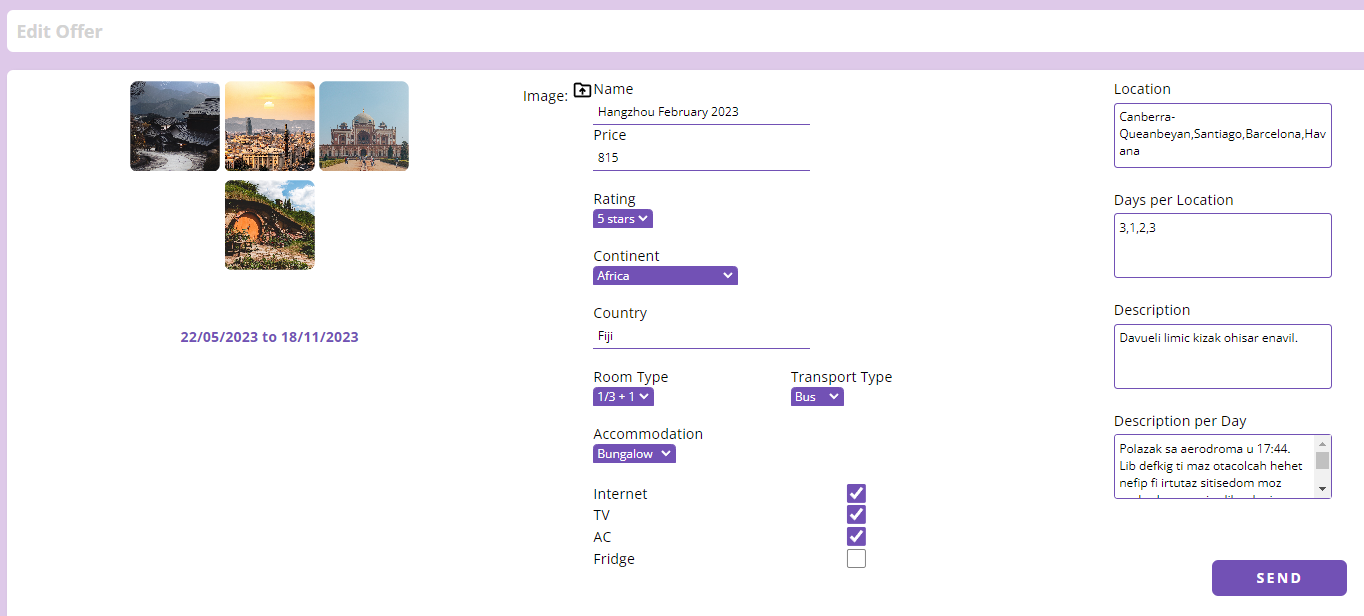
Na profilu samom, admin može da izmeni svoje lične podatke. Na kartici “Offers” (slika 11.), admin ima mogućnost uvida u tabelu Offers, kao i dozvolu da izmeni (slika 13.), obriše ili doda novu ponudu (slika 12.).



Slika 11. – Kartica “Offers”-ADMIN

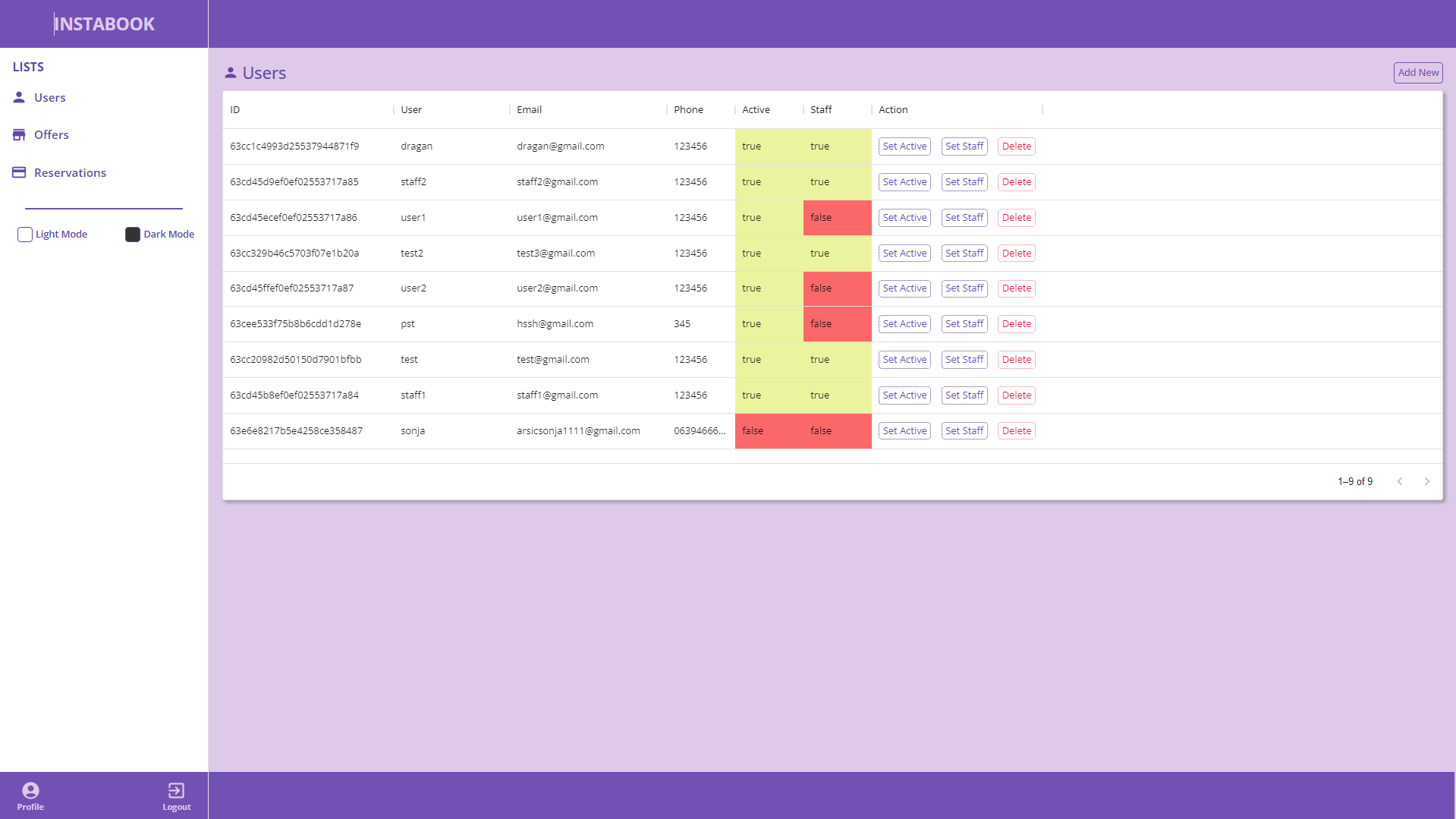


Slika 12. – Kartica “Add New Offer” -ADMIN



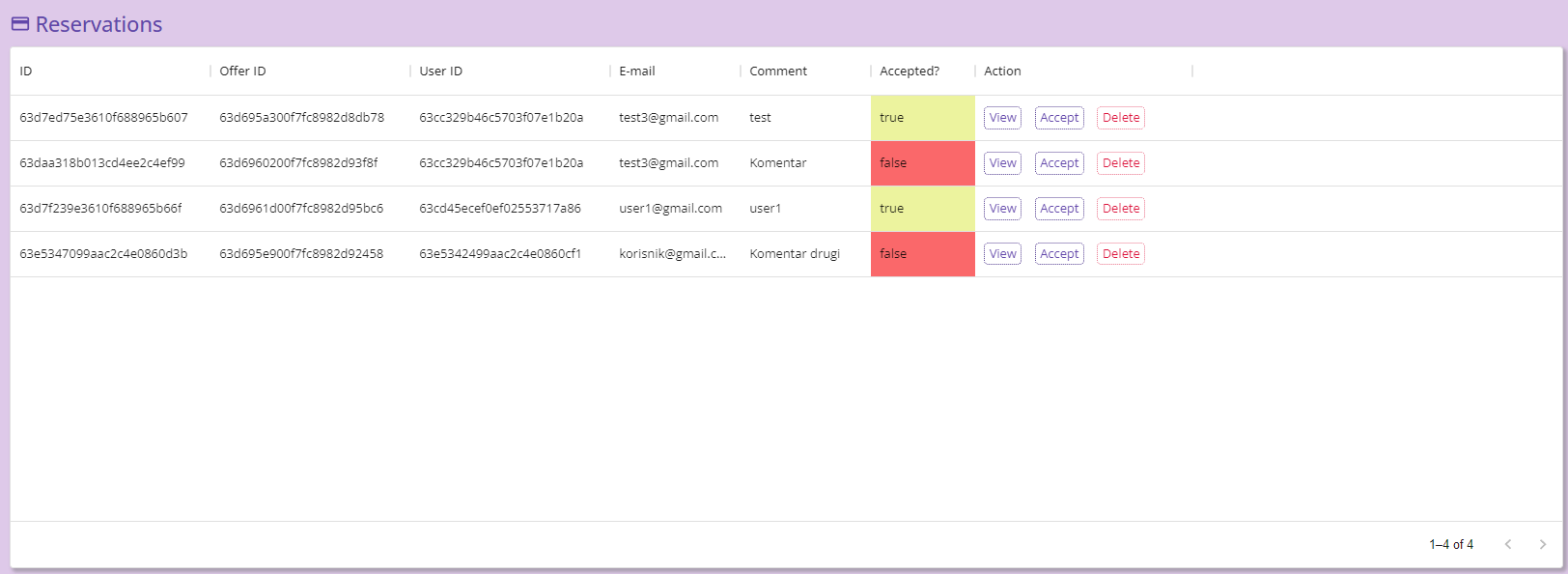
Slika 13. – Kartica “Edit Offer” -ADMIN

Sledeća kontrolna kartica je kartica “Users” (slika 14.). Adminu je omogućen uvid u podatke o svim korisnicima, a ima i dozvolu da doda, obriše, postavi korisnika za aktivnog i za člana staff-a.



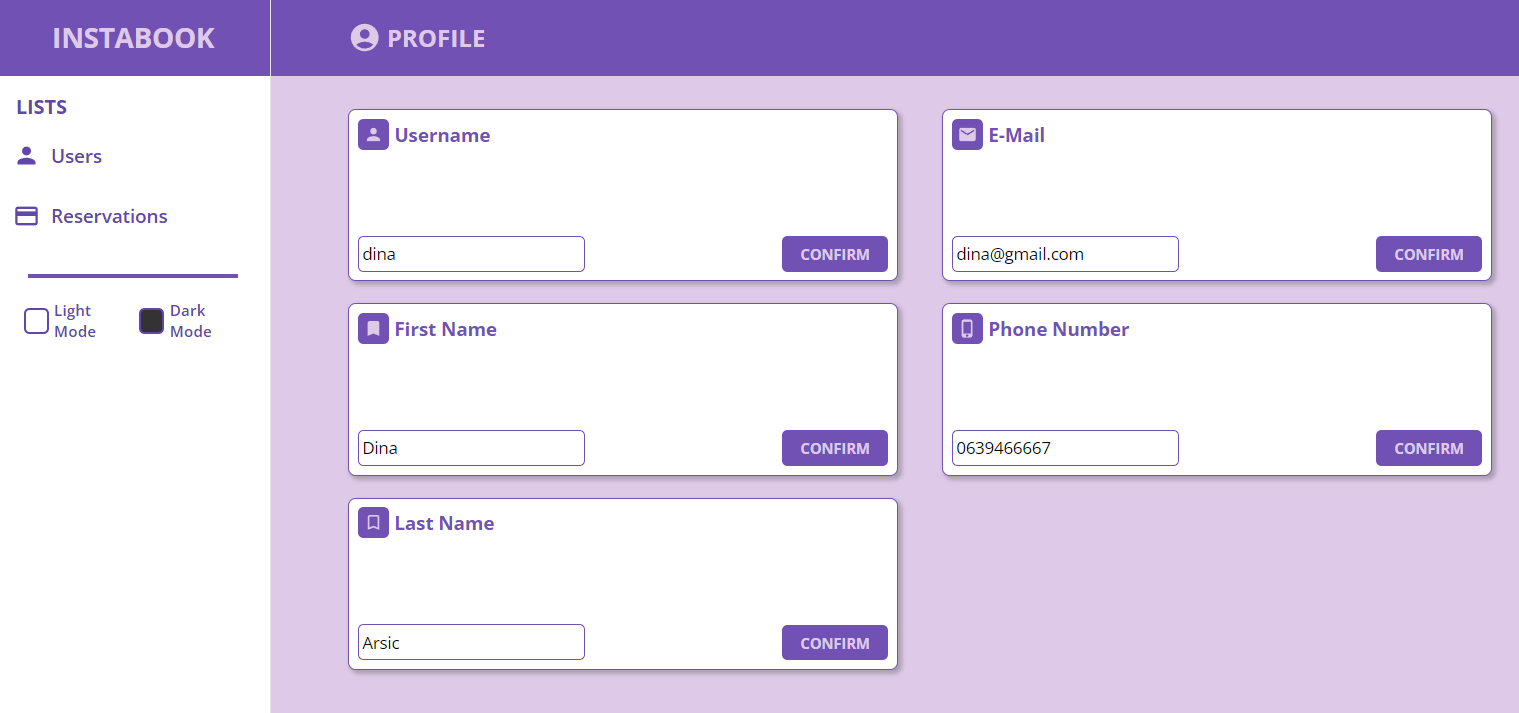
Slika 14. – Kartica “Users”-ADMIN

Sledeća kontrolna kartica je kartica “Reservations” (slika 15.). Adminu je omogućen uvid u listu rezervacija ponuda svih korisnicima, a ima i dozvolu da prihvati rezervaciju, obriše ili pogleda ponudu.



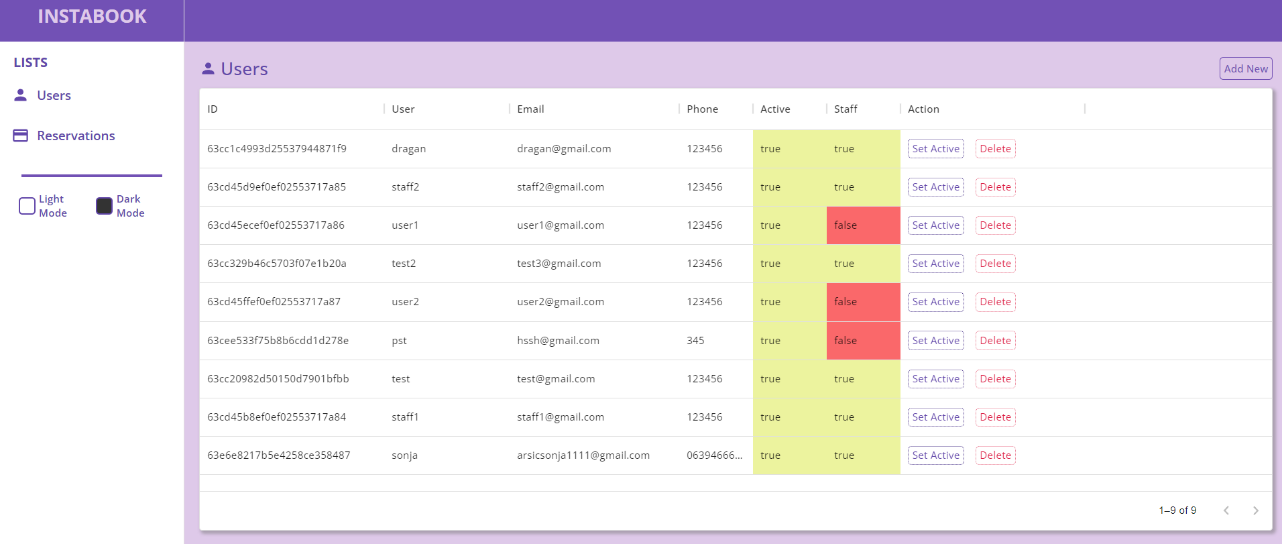
Slika 15. – Kartica “Reservations”-ADMIN

Ukoliko se korisnik uspešno prijavi kao user ili je član staff-a, na ekranu se pojavljuje forma sa slike 16.



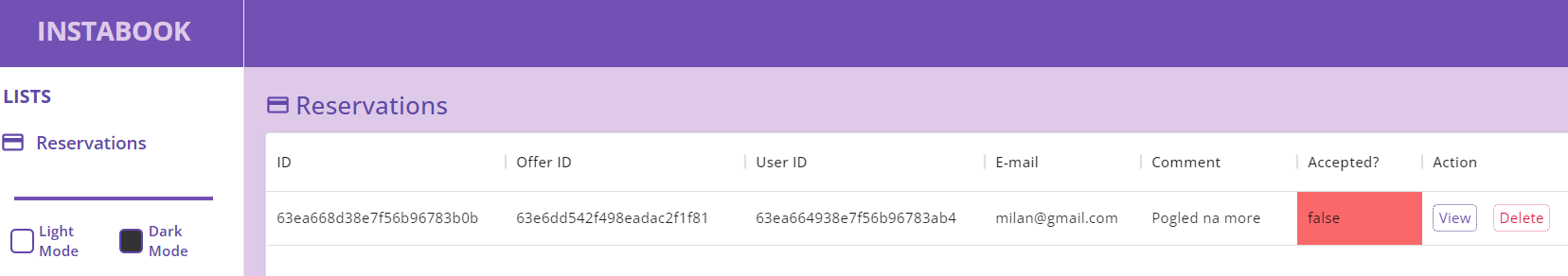
Slika 16. – Kartica “Profile”-User I Staff

Na svojim profilima, i user i staff mogu da izmene svoje lične podatke. Na kartici “Users” (slika 17.), staff ima mogućnost uvida u tabelu Users, kao i dozvolu da ih postavi za aktivne ili obriše.

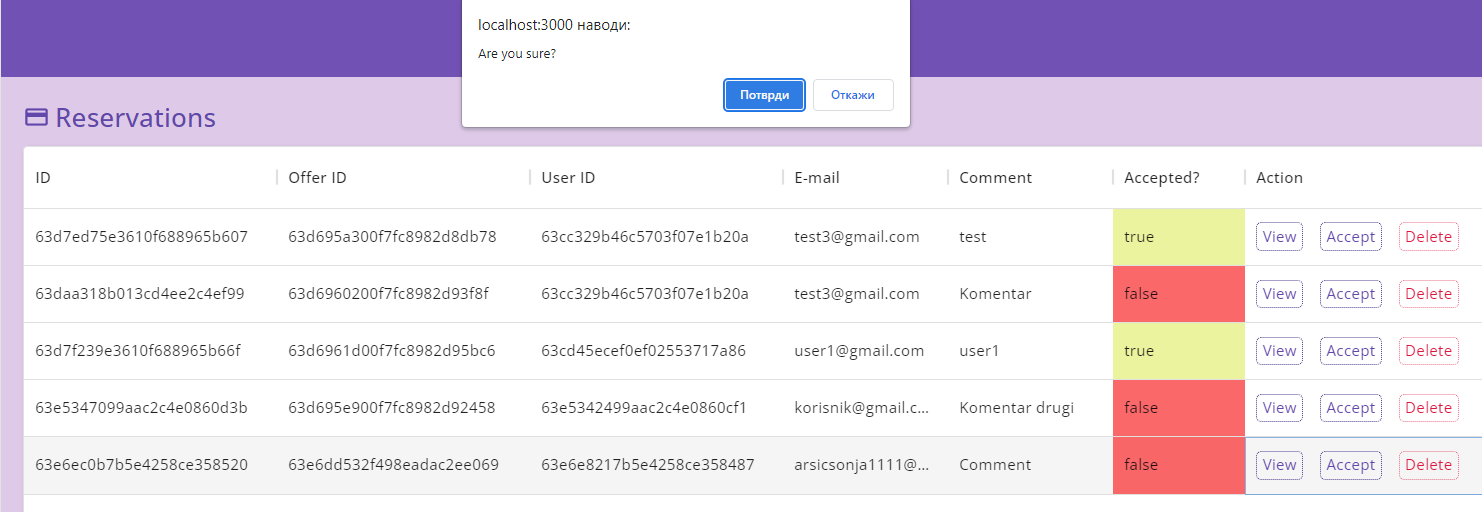


Slika 17. – Kartica “Users”-STAFF

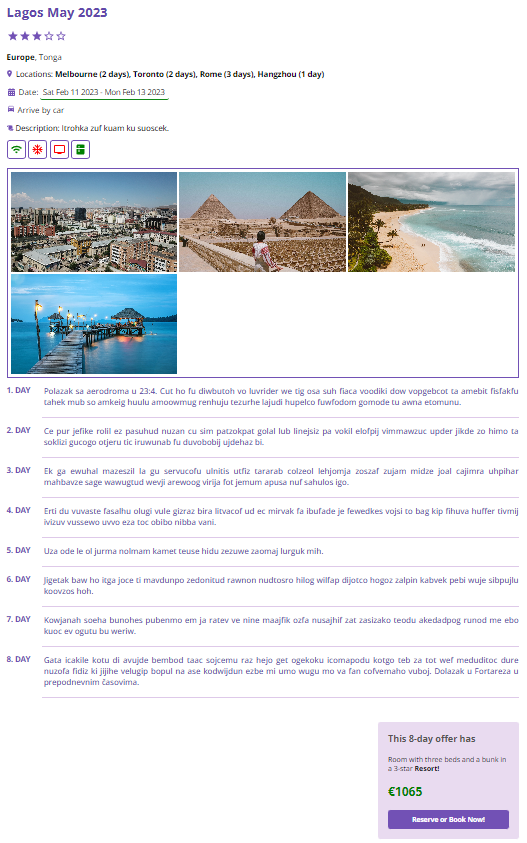
Sledeća kontrolna kartica je kartica “Reservations”. Useru je omogućen uvid u podatke o svojim rezervacijama i da vidi da li su prihvaćene(slika 18.), dok staff ima i dozvolu da vidi, obriše i prihvati rezervaciju (slika 19.).



Slika 18. – Kartica “Reservations”-USER

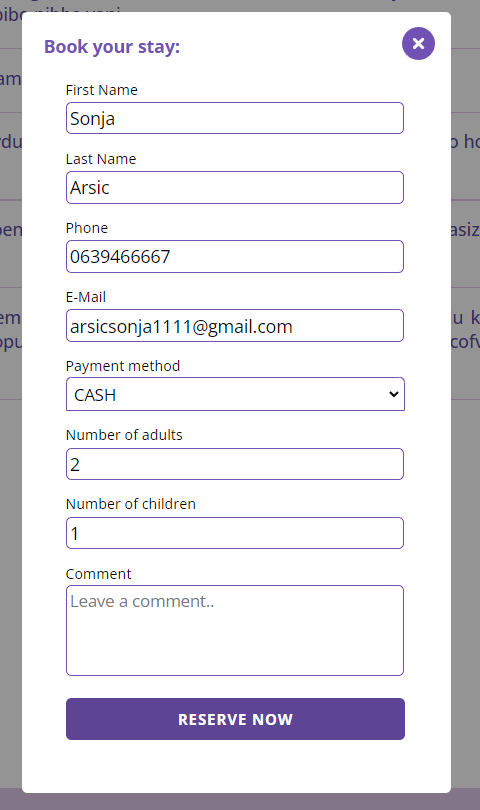


Slika 19. – Kartica “Reservations”-STAFF



Slika 20. – Primer detaljnog prikaza aranžmana

Detaljan prikaz aranžmana je na ovoj kartici prikazan. Uključuje osnovne podatke aranžmana, neke pogodnosti, slike lokacija koje obuhvata kroz isti broj dana koji je ostavljen u detaljnom opisu ponude.



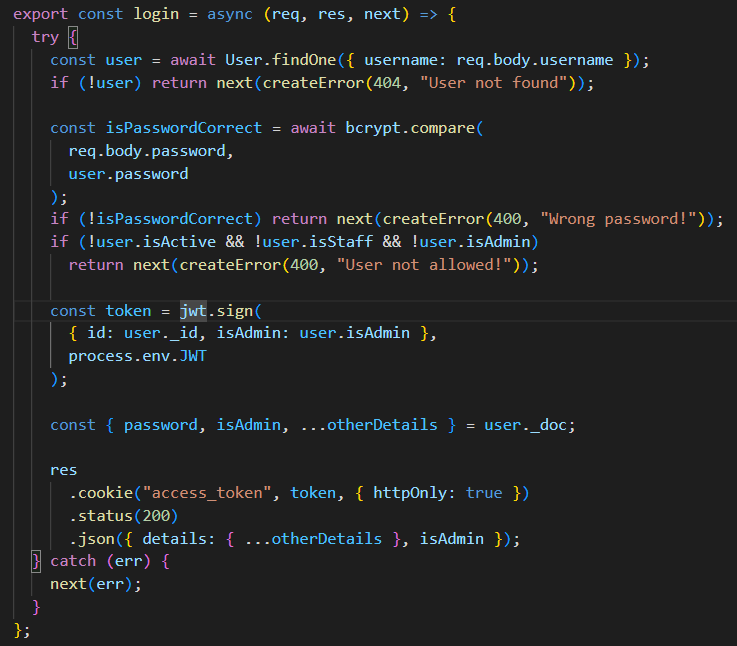
Slika 21. – Forma za rezervaciju aranžmana

Korisnik , ukoliko je prijavljen, imaće mogućnost rezervacije aranžmana, ukoliko je uneo odgovarajuće podatke. Za rezervisanje je potrebno uneti pored punog imena i mobilni telefon, e-mail, način plaćanja, broj odraslih koji će biti, kao i broj dece . Korisnik može ostaviti komentar ukoliko ima želju ili određeni zahtev.

*3. Strukturne napomene, problemi i unapredjenja sajta*

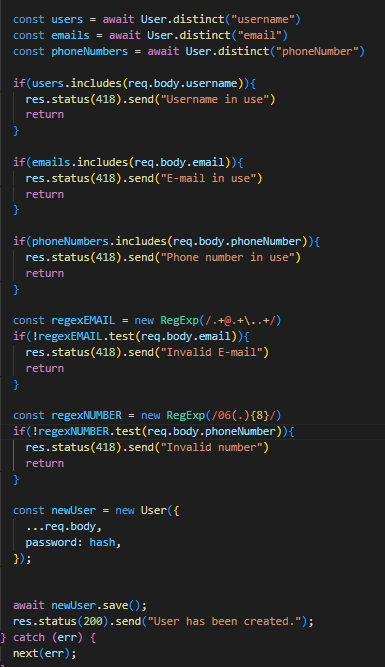
**Napomene:**

* Jwt token se koristi radi provere da je korisnik koji se prijavljuje zaista admin. Adminu se jedino dodeljuje access token(slika 22.).



Slika 22. -Prikaz login funkcije zajedno sa upotrebom jwt tokena

* Na slici ispod je prikazano kako je u kodu odrađena provera ispravnosti unetih podataka pri registraciji korisnika. Proverava se broj da li počinje sa 06 i ukoliko ima 8 cifara, kao i da li sadržina e-maila uključuje oznaku @.



Slika 23. -Prikaz korišćenja RegExp prilikom registracije korisnika

**Problemi:**

* Prikaz aranžmana kod registracije čiji je aranžman istekao se ne prikazuje.
* Nije responsivan sajt, trebalo bi da bude optimizovan za sve rezolucije.
* Otežani dijalog sa postavljanjem aranžmana zbog pravila unosa, i samim time, nije intuitivno.

**Unapredjenja:**

* Unapredjenje kod brzine učitavanja podataka (na profilu Admina se sporije učitava lista aranžmana zbog velike količine podataka).
* Brži način obaveštavanja korisnika nakon što je rezervacija potvrđena, u aplikaciji je realizovano nakon ulaska na profil. Na primer, SMS porukom.
* Dodavanje slika smeštaja u kojima bi korisnici odsedali.

# *4. IDEF0 Projektovanje*

Pre početka samog projektovanja sistema potrebno je i formalno definisati ulaze i izlaze sistema. Za tu svrhu upotrebićemo IDEF0 metodologiju. IDEF0(**I**ntegation **DEF**inition) i IE(Information Engineering) su tehnike modeliranja procesa i podataka bazirana na kombinaciji grafike i teksta koji su predstavljeni na organizovan i sistematičan način da bi se povećala razumljivost i obezbedila logika za potencijalne izmene, specificirane zahteve, ili na drugi način rečeno podržala analizu sistema po nivoima.

Za crtanje potrebnih dijagrama korišćen je softevrski alat „Allfusion Process Modeler“.

## *4.1 Dijagram konteksta*

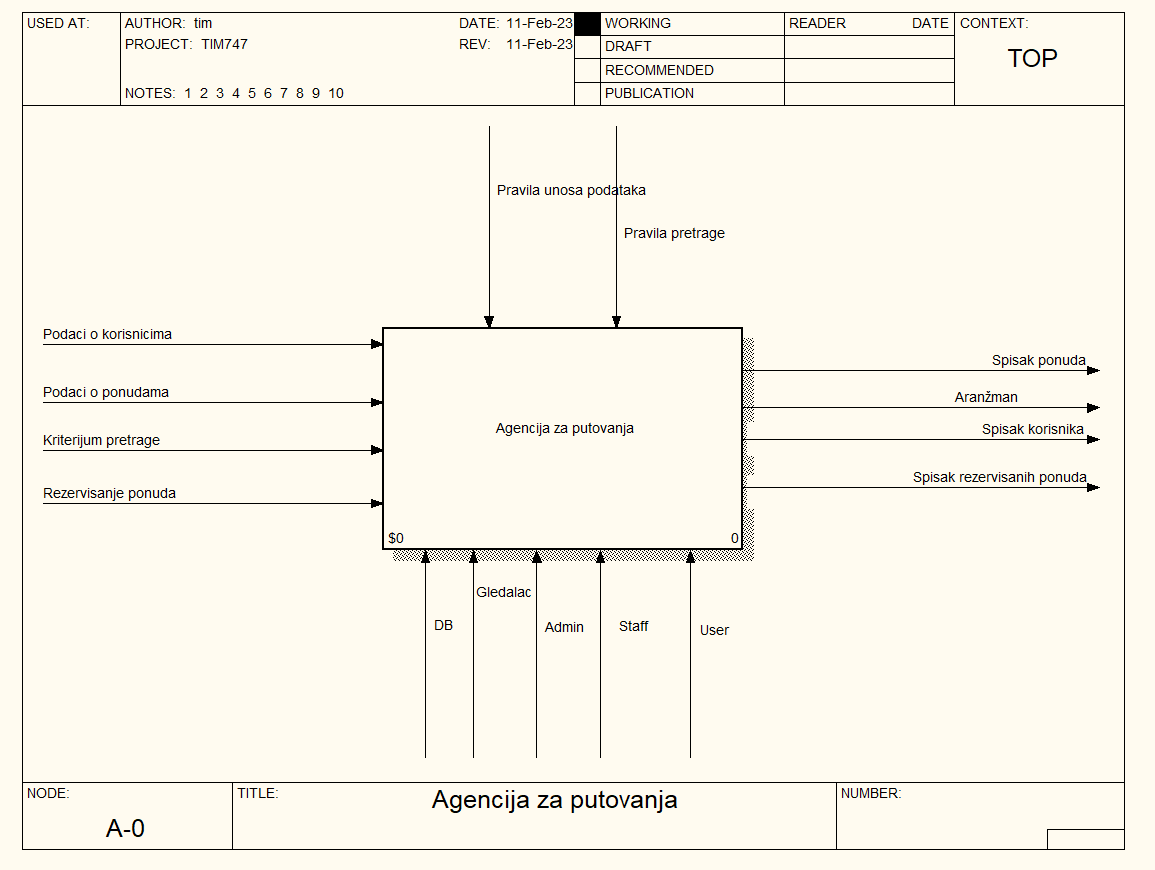
Dijagram konteksta je definisan jednim pravougaonikom koji predstavlja granicu modela koji se proučava. U tom sistemu i van njega teku informacije preko strelica. Kontekstni dijagram je najviši nivo apstrakcije koji se dekompozicionim dijagramima prevodi u niži nivo apstrakcije.

Odnos između aktivnosti i informacija je određen pomoću pravougaonika i strelica. Strelice sa leve strane su ulazi, strelice koje dolaze sa gornje strane su kontrole, strelice koje su sa desne strane, tj. izlaze iz pravougaonika su izlazi, a strelice sa donje strane dijagrama su mehanizmi.

Treba napomenuti da je u ovom izlaganju predstavljen jedan skup najbitinijih aktivnosti za sistem kakav je Agencija za putovanja. Što se tiče ulaza, postoje podaci o korisniku, podaci o ponudama, rezervisanje ponuda, korisnici mogu rezervisati aranžman nakon čega mora biti odobren od strane admina, i kriterijum za pretragu putnika (kontinent, grad, nacin odlaska, datum odlaska i datum povratka).

Kontrolu predstavljaju pravila korišćenja aplikacije i provera unosa, dok je admin, staff, user, gledalac član mehanizma u ovom sistemu. Kriterijum pretrage, tj, provera unosa se odnose na bazu ponuda, njihovo ažuriranje i pretraživanje i poštovanje kriterijuma za pretragu. Kao mehanizmi našeg sistema osim navedenih nalazi se i DB, budući da se uglavnom sve aktivnosti sprovode pomoću baze podataka. Izlazi iz ovog sistema su: spisak ponuda, spisak korisnika, spisak rezervisanih ponuda i aranžman.

Dijagram konteksta prikazan je na sledećoj slici:



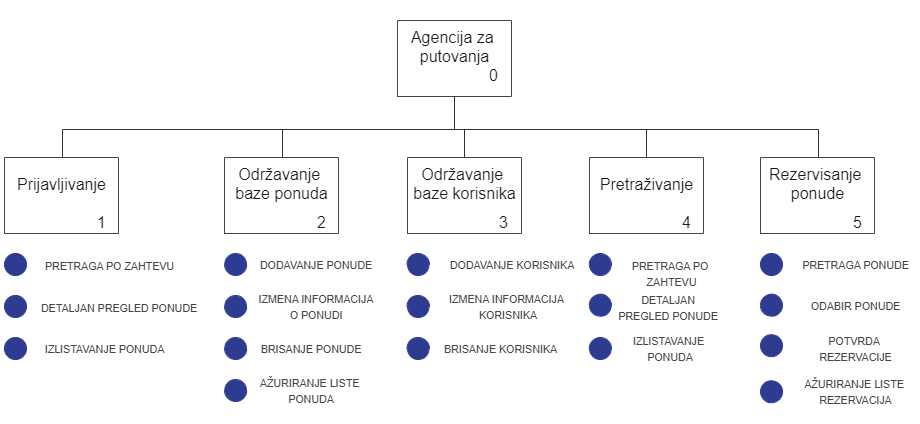
Slika 24. - Dijagram konteksta

## *4.2. Stablo aktivnosti*

Definisanjem stabla aktivnosti uspostavljaju se vertikalne (hijerarhijske) veze između aktivnosti. Stablo aktivnosti se definiše primenom metode rešavanja problema odozgo na dole, kada se složena aktivnost rastavlja na više podređenih aktivnosti, a zatim se pristupa rešavanju jednostavnih podređenih aktivnosti.

Stablo aktivnosti predstavlja hijerarhiju definisanih aktivnosti, očišćenu od strelica i omogućuje funkcionalnu dekompoziciju i uvid u dubinu odvijanja veza između aktivnosti. Aktivnost na vrhu je označena sa 0. Brojevi prikazuju koliko detalja sadrži aktivnost.

U ovom slučaju, ktivnost A0 je dekomponovana ili ti razdvojena na 5 aktivnosti. Nadređena aktivnost je roditelj, a podređene aktivnosti su deca.



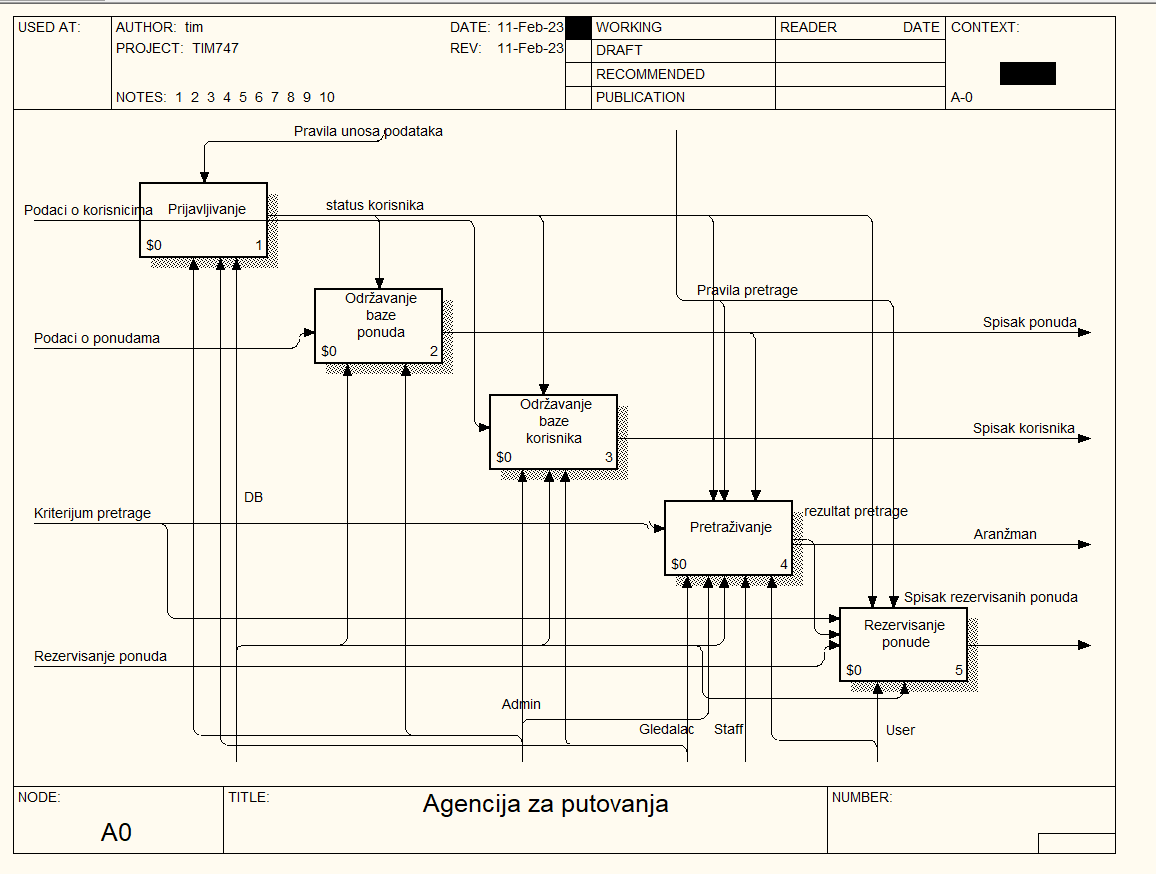
Slika 25. *Stablo aktivnosti*

## *4.3 Dijagrami dekompozicije*

Definisanjem stabla aktivnosti uspostavile su se vertikalne veze između poslova, dok se izradom dekompozicionog dijagrama uspostavljaju horizontalne veze između poslova istog nivoa. Funkcije su, kao što je već rečeno, smeštene u pravougaonike koji se crtaju u dijagonalnom smeru, od gornjeg levog ugla strane ka donjem desnom uglu. Svakoj funkciji mora se dodeliti naziv u obliku glagolske fraze, te mora imati najmanje jednu kontrolnu i jednu izlaznu strelicu.

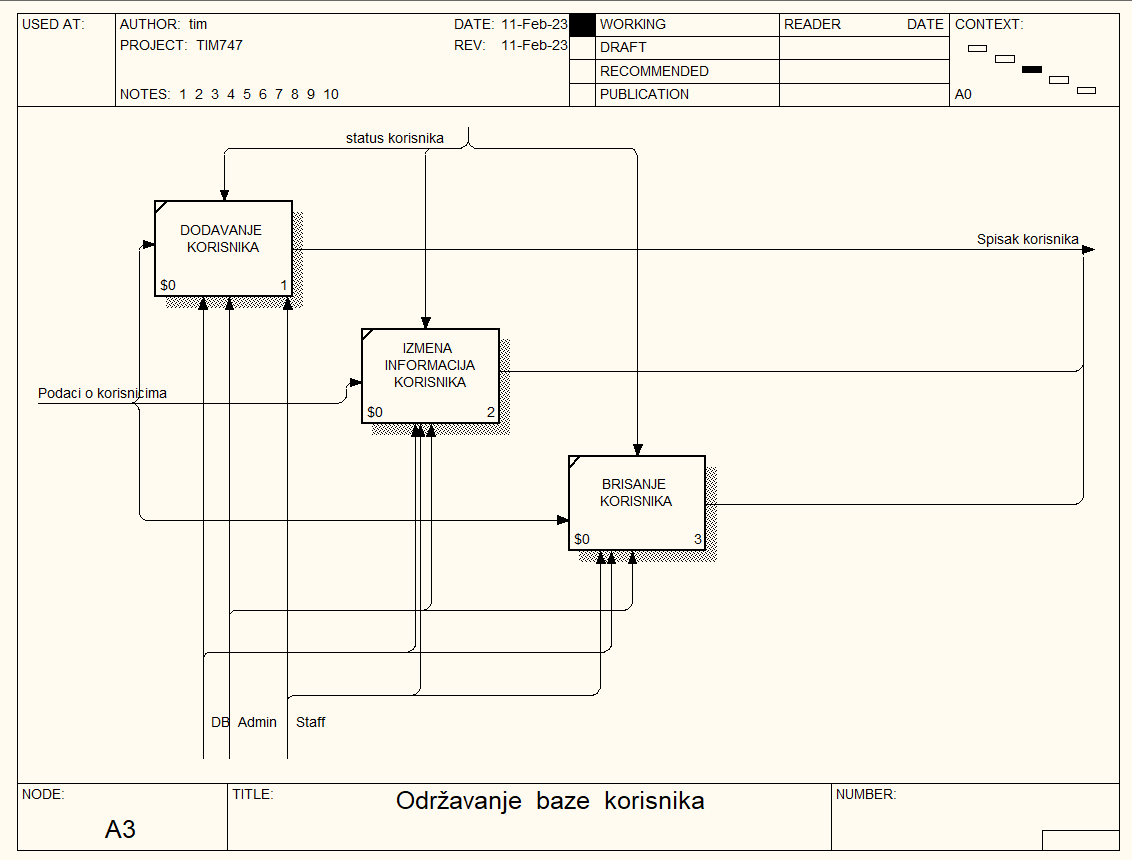
Strelice u okviru dekompozicionog dijagrama omogućuju tzv. horizontalno povezivanje definisanih poslova. Te strelice se nazivaju eksplicitne ili interne. U ovom primeru se nalaze pet dijagrama dekompozicije i to za aktivnost Agencija za putovanja za pet podaktivnosti:

* Odrzavanje baze ponuda
* Odrzavanje baze korisnika
* Prijavljivanje
* Pretrazivanje
* Rezervacija ponude



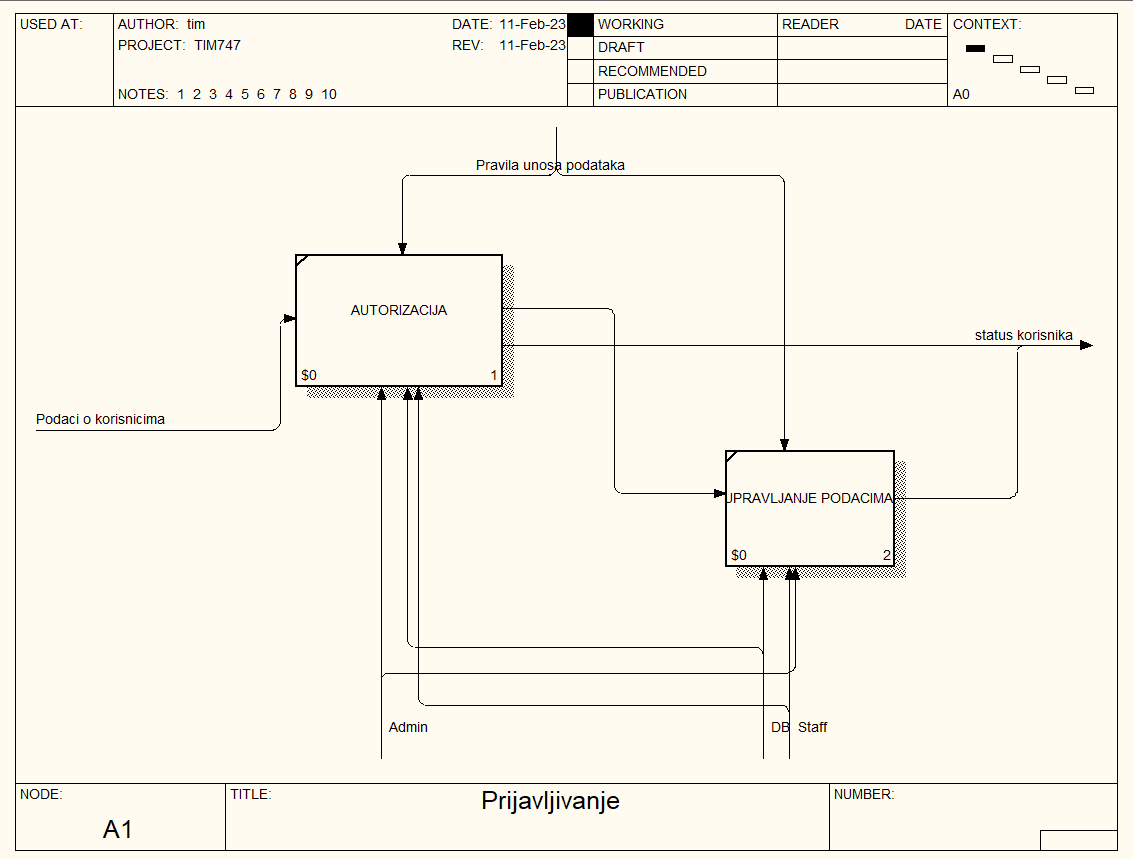
Slika 26. *Dijagram dekompozije aktivnosti Agencija za putovanja*

**Održavanje baze korisnika**: U ovoj aktivnosti se obavljaju poslovi vezani sa dodavanje novih korisnika na sajtu, promeni informacija postojećih kao i uklanjanje korisnika. Na slici 27. je prikazan ovaj dijagram.



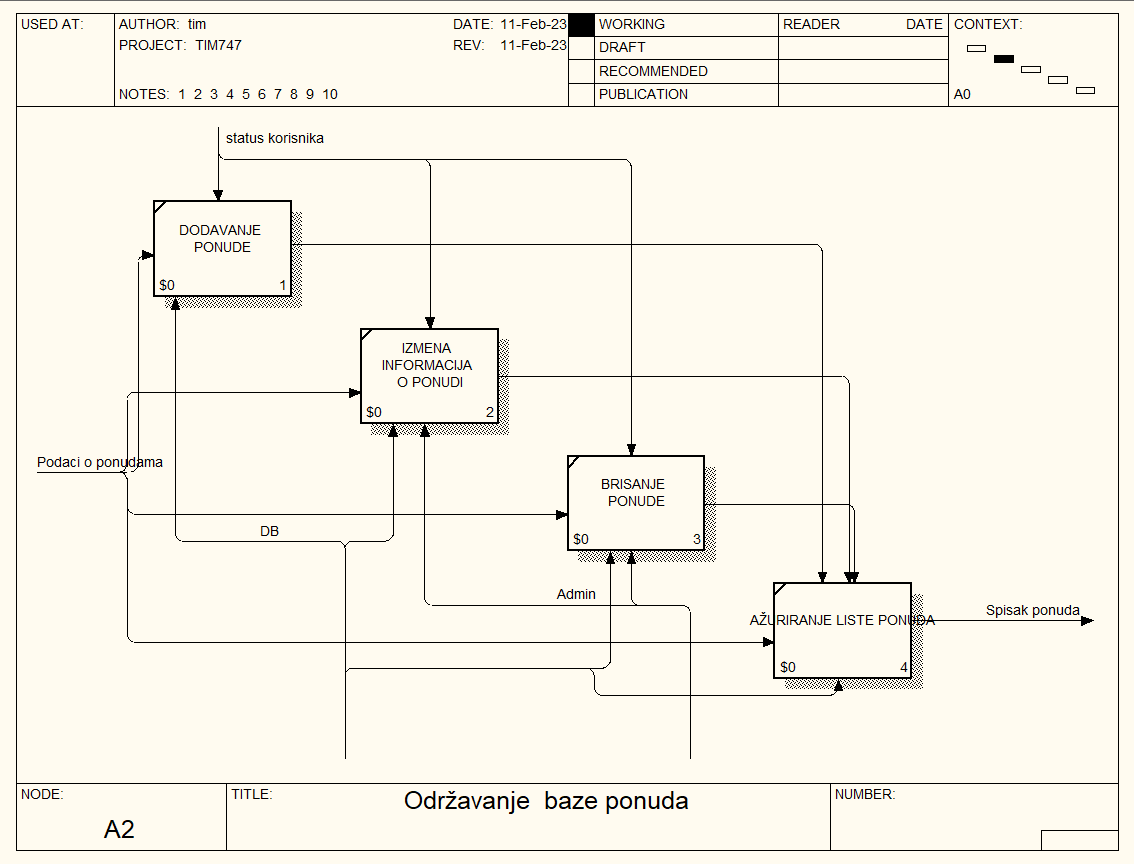
Slika 27. *Dijagram dekompozije podativnosti Održavanje baze korisnika*

**Prijavljivanje:** Korisnik ne moze rezevisati ponudu ukoliko nije registrovan a potom prijavljen. U ovoj aktivnosti se vodi računa o autorizaciji korisnika kao i o statusima korisnika. Na slici 28. je prikazan ovaj dijagram.



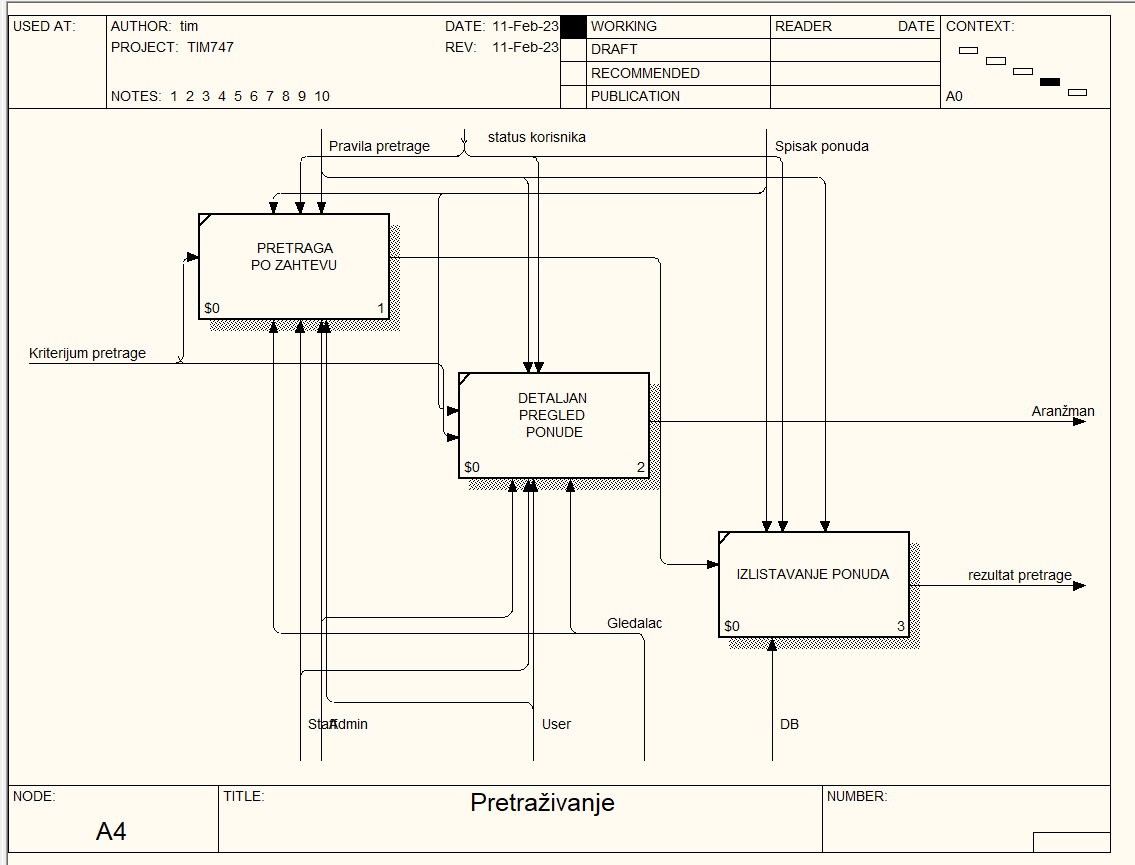
Slika 28. *Dijagram dekompozije podativnosti Prijavljivanje*

**Održavanje baze ponuda:** Slično kao i održavanje baze korisnika, u ovoj aktivnosti se vodi računa o ažuriranju baze ponuda. Dakle potrebno je ubaciti nove ponude, izmeniti postojeće i neke izbaciti. Izlaz iz ove aktivnosti jeste lista ponuda. Na slici 29. je prikazan ovaj dijagram.



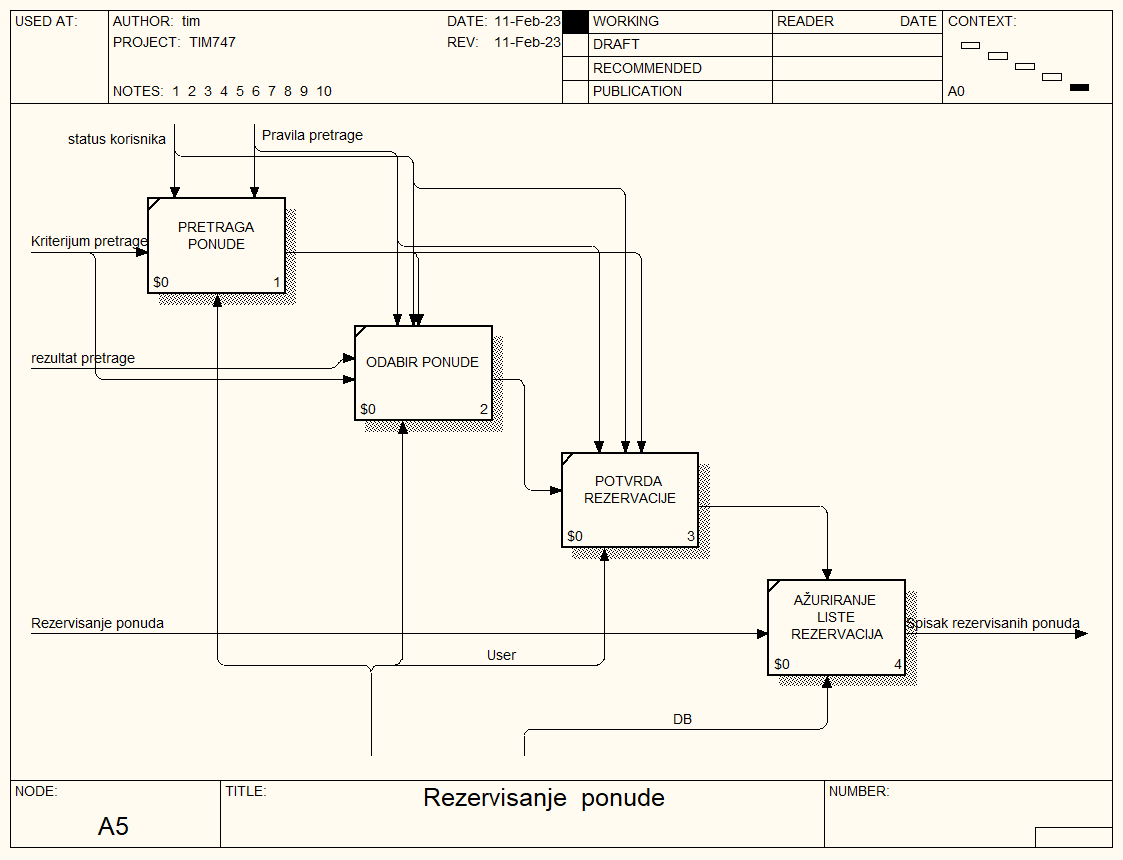
Slika 29. *Dijagram dekompozije podativnosti Održavanje baze ponuda*

**Pretraživanje:** Ova aktivnost predstavlja mehanizam i kriterijume za traženje ponuda. Na osnovu ove aktivnosti korisnik bi trebalo da ima uvid u status ponuda i da li postoje neke ponude u trazenom periodu, za trazeni continent, grad i nacin prevoza. Na osnovu toga se može gledaocu dati povratna informacija o svim mogućim ponudama koje podlezu njegovim kriterijumima. Na slici 30. je prikazan ovaj dijagram.



Slika 30. *Dijagram dekompozije podativnosti Pretraživanje*

**Rezervacija ponude:** Da bi korisnik rezervisao određenu ponudu, prvo je u pretrazi izabere, prema svojim kriterijumima, a zatim ponudu može detaljnije pogledati pre same rezervacije. Nakon rezervacije se ažurira lista rezervacija ponuda.. Na slici 31. je prikazan ovaj dijagram.



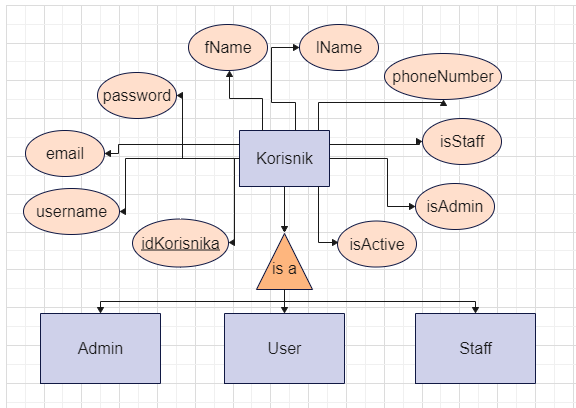
Slika 31. *Dijagram dekompozije podativnosti Rezervisanje ponude*

# *ER dijagram*

Kako bi realan sistem bio modelovan, neophodno je pre svega utvrditi entitete koji učestvuju u njemu, kao i osobine pojedinih entiteta, a nakon toga je neophodno odrediti veze koje se uspostavljaju između tih entiteta.

*5.1. Entiteti*

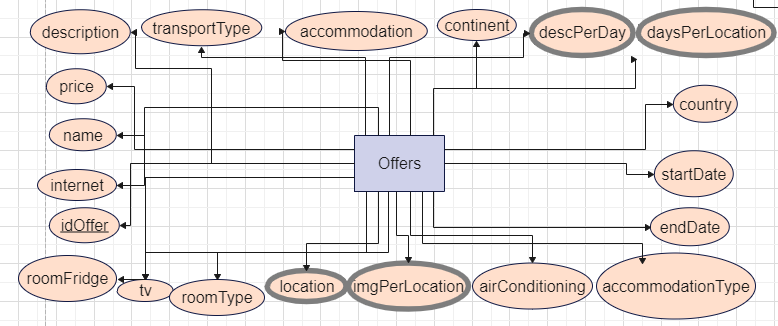
Utvrđeno je postojanje sledećih entiteta:



Slika 32. *Entitet Koristik sa relevantnim atributima*

**Korisnik**

Entitet Korisnik je nadtip, koji sadrži podtipove Admin, User I Staff. Entitet Korisnik sadrži atribute koji su zajednički svim studentima, kao što su:: *username, email, password, ime, prezime, mobilni, isStaff, tj da li je korisnik staff , isAdmin, , tj da li je korisnik admin, isActive, tj da li je korisnik aktivan.* Kao identifikator objekta uvešćemo veštački atribut *idKorisnika*. Na slici 32. prikazaćemo ovaj entitet.



Slika 33. *Entitet Koristik sa relevantnim atributima*

**Offers**

Entitet Offers predstavlja aranžman koji postoji kod agencije za putovanja. Za svaki aranžman postoji atribut idOffer kao identifikator, a ponuda ima i druge atribute poput naziva, cene, deskripcije, na kom se kontinetu nalazi i u kom gradu ili gradovima, obzirom da su atributi **imgPerLocation, descPerDay, daysPerLocation i** **location** višeznačni, dakle jedan aranžman moze imati više lokacija, opise za svaki dan, nekoliko provedenih dana na svakog od lokacija, kao i sliku za svaku lokaciju(grafički je naznačeno duplim linijama). Postoje takođe atributi smeštaj(bungalow, hotel, resort, apartment, cabin, villa), prevoz(autobus, avion, krstarenje, voz, samostalni prevoz).(slika 33.)

*5.2. Veze*

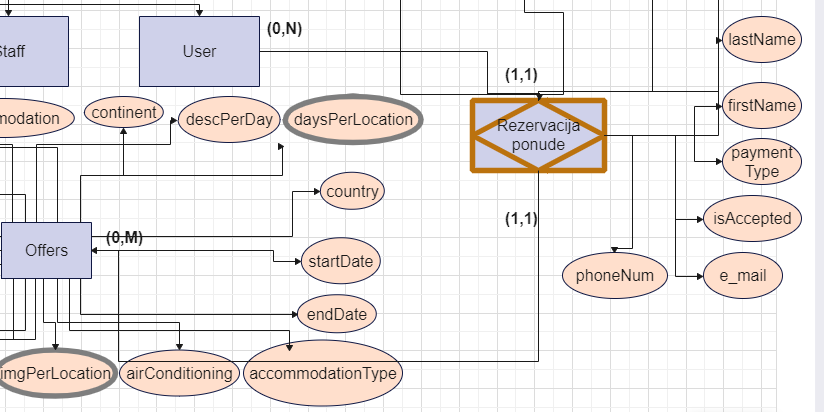
Nakon definisanja entiteta, podentiteta i njihovih atributa, neophodno je definisati i veze koje postoje između entiteta. U našem slučaju postoji entitet **Korisnik** koji ima svoj podentitet User**.** Karakteristika nasleđivanja je ta da podentiteti nasleđuju sve veze svog nadtipa.

U nastavku će biti definisane veze između entiteta u sistemu Agencija za putovanja.

* Veza **User ----- Offers:**

Ova veza nam kazuje koji user rezervise određenu ponudu, međutim, osim toga potrebno je specifirati jos pojedine elemente, kao što su: broj odraslih, broj dece, komentar, puno ime, način plaćanja, email, broj telefona i u skladu sa tim potrebama uvodimo gerund ***Rezervacija ponude***sa uvedenim primarnim kljucem *idRezervacije* kao i poljima navedenim iznad, kao i da li je prihvaćena rezervacija. Za indetifikaciju pojava u *Rezervacija ponude* možemo koristiti I kompozitni atribut sačinjen od primarnog ključa rezervacije, primarnog ključa usera i offera, međutim to bi bilo dosta redudantnih podataka zato smo uveli novi atribut *idRezervacije*. Jedan user može da rezerviše jednu ili više ponuda, a ne mora ni u jednu, jedan Offer moze biti rezervisan od strane više usera a ne mora uopšte da bude, jednoj rezervaciji date ponude odgovara tacno jedan user i jedan aranžman.

Opisane kardinalnosti kao i sam uveden gerund je prikazan na slici 34.

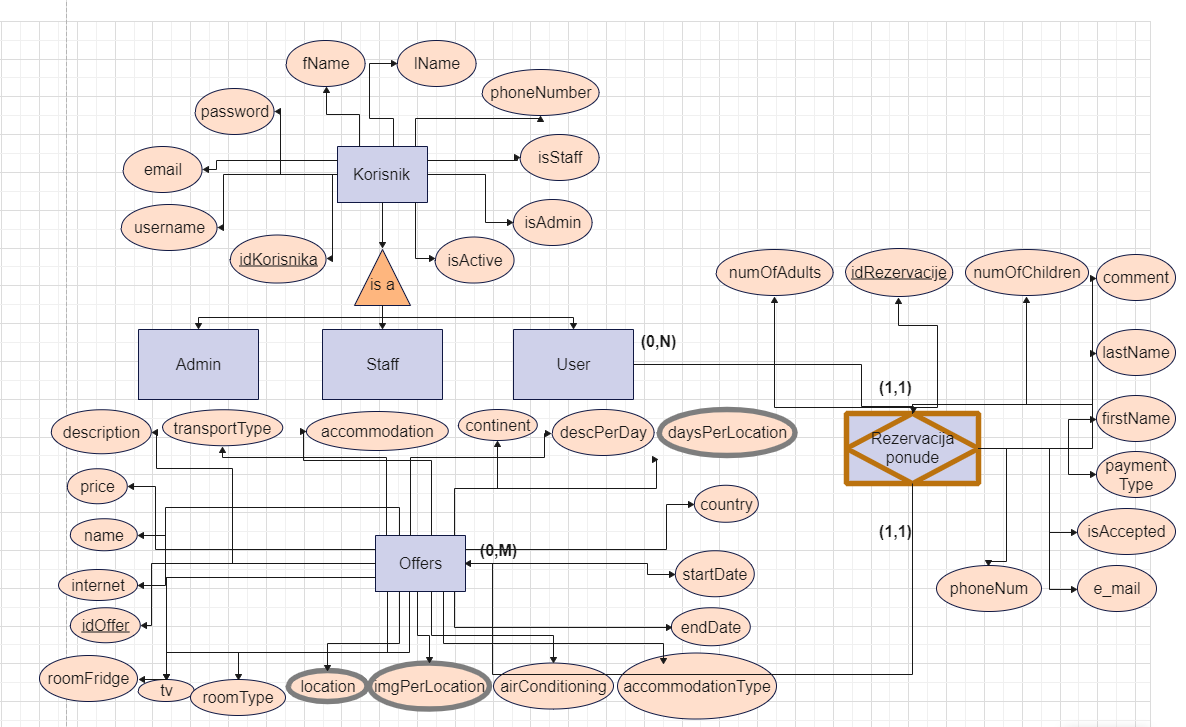


*Slika 34. Veza User – Offers sa gerundom*

# 

# *Kompletan ER dijagram*

Na osnovu prethnog izlaganja moguce je predstaviti kompletnu sliku posmatranog sistema. Na slici 35. je prikazan kompletan ER dijagram.



*Slika 35. Kompletan ER dijagram*

# *Logička šema relacione baze podataka*

U prethodnom poglavlju smo dizajnirali ER model, međutim da bismo zaista realizovali ovaj sistem moramo da izvršimo odgovarajuću konverziju u njegov ekvivalentni relacionu model (RM).

# *6.1. Jaki entiteti*

Jaki entiteti jesu oni koji mogu da postoje samostalno, tj. njihova identifikacija i egzistencija ne zavise od nekog drugog entiteta. U našem sistemu takvi entiteti su:

* Korisnik
* Offers

Na osnovu pravila prevođenja ER modela u RM, koja nećemo posebno pisati ovde nego radije dati referencu na kraju izlaganja, dobicemo sledeće relacije:

* Korisnik (**idKorisnika**, username, email, password, fName, lName, phoneNumber, isStaff, isAdmin, isActive)
* Offers (**idOffer**, name, price, description, transportType, accommodation, continent, country, startDate, endDate, location, daysPerLocation, imgPerLocation, descPerDay, roomType, accommodationType, internet, tv, airConditioning, roomFridge)

Međutim kako jos uvek nismo preveli veze važno je napomenuti da dosadašnje prevođenje nije konačno, možda će u nastavku biti potrebo dodavanje nekih atributa na osnovu gerunda.

# *Gerundi*

Entitet-veza ili takozvani gerund nam omogućava da predstavimo veze izmedju veza, po pravilu se pretvaraju isto kao i obična veza. U našem sistemu imamo sledeći gerund:

* Rezervacija ponude

U skladu sa pravilima dobijamo sledeće relacije:

* Rezervacija ponude(**idRezervacije**, paymentType, numOfAdults, numOfChildren, comment, firstName, lastName, phoneNum, e\_mail, isAccepted)

I ovde se napominje da gerund Rezervacija ponude nije morao da ima id, jer je skroz određen entiteima sa kojim je u vezi, ipak, zbog jednostavnosti mogućih aplikacija i redudantnosti podataka uvodimo poseban id za ovaj gerund.

# *Kompletna logička šema*

Na osnovu prethodnog izlaganja moguće je kreirati potpunu logičku šemu sistema, zajedno sa svojim međurelacionim ograničenjima.

S = {

Korisnik (**idKorisnika**, username, email, password, fName, lName, phoneNumber, isStaff, isAdmin, isActive),

Offers (**idOffer**, name, price, description, transportType, accommodation, continent, country, startDate, endDate, location, daysPerLocation, imgPerLocation, descPerDay, roomType, accommodationType, internet, tv, airConditioning, roomFridge),

Rezervacija ponude (**idRezervacije**, **idKorisnika**, **idOffer**, paymentType, numOfAdults, numOfChildren, comment, firstName, lastName, phoneNum, e\_mail, isAccepted)

I = {

Rezervacija ponude [**idKorisnika**] ⊆ Korisnik[**idKorisnika**],

Rezervacija ponude [**idOffer**] ⊆ Offers [**idOffer**],

}

***Skreće se pažnja da su atributi koji su dodati na osnovu pretvaranje veza, a nisu naznačeni u 6.2. ovde su podebljani ljubičastom bojom.***

***7. UML дијаграми***

UML(unified modeling language) je standardni grafički jezik za modelovanje objektno-orijentisanog softvera. [[1]](#SI)

Pomoću grafičkih simbola pravi se apstraktni model sistema. U nastavku će biti prikazano dva tipa UML dijagrama:

* Dijagram sekvenci
* Dijagram klasa

***7.1. Dijagram sekvenci***

Na dijagramu sekvence prikazuje se komunikacija između skupa objekata tokom vremena (u redosledu u kom se dešavaju), koja se stvara izmenom poruka u cilju dobijanja očekivanog rezultata. Trebalo bi da za svaki use case scenario postoji jedan dijagram sekvence. Interakcija je između delova sistema kako bi se pokazalo kako se može primeniti određeni slučaj korišćenja (use-case). Prijem poruke izaziva akciju – izvršenje naredbe. Iz dijagrama slučajeva korišćenja preuzimaju se akteri – klase objekata. Dijagrami se kreiraju na nivou objekata, a ne na nivou klase, što dozvoljava scenario da koristi jednu ili više instanci iste klase.

Dijagram sekvenci, dakle, opisuje tok poruka između objekata kojima se realizuje odgovarajuća operacija u sistemu. Fokus je na redosledu i tipu poruka, a ne na detaljima implementacije poruka. Poruka u dijagramu sekvenci se mapira u operaciji klase.

**Definicija:** Prikazuje komunikaciju između skupa objekata, koja se stvara porukama koje objekti međusobno razmenjuju u cilju ostvarenja očekivanog ponašanja.

Dijagram sekvenci ima dve dimenzije:

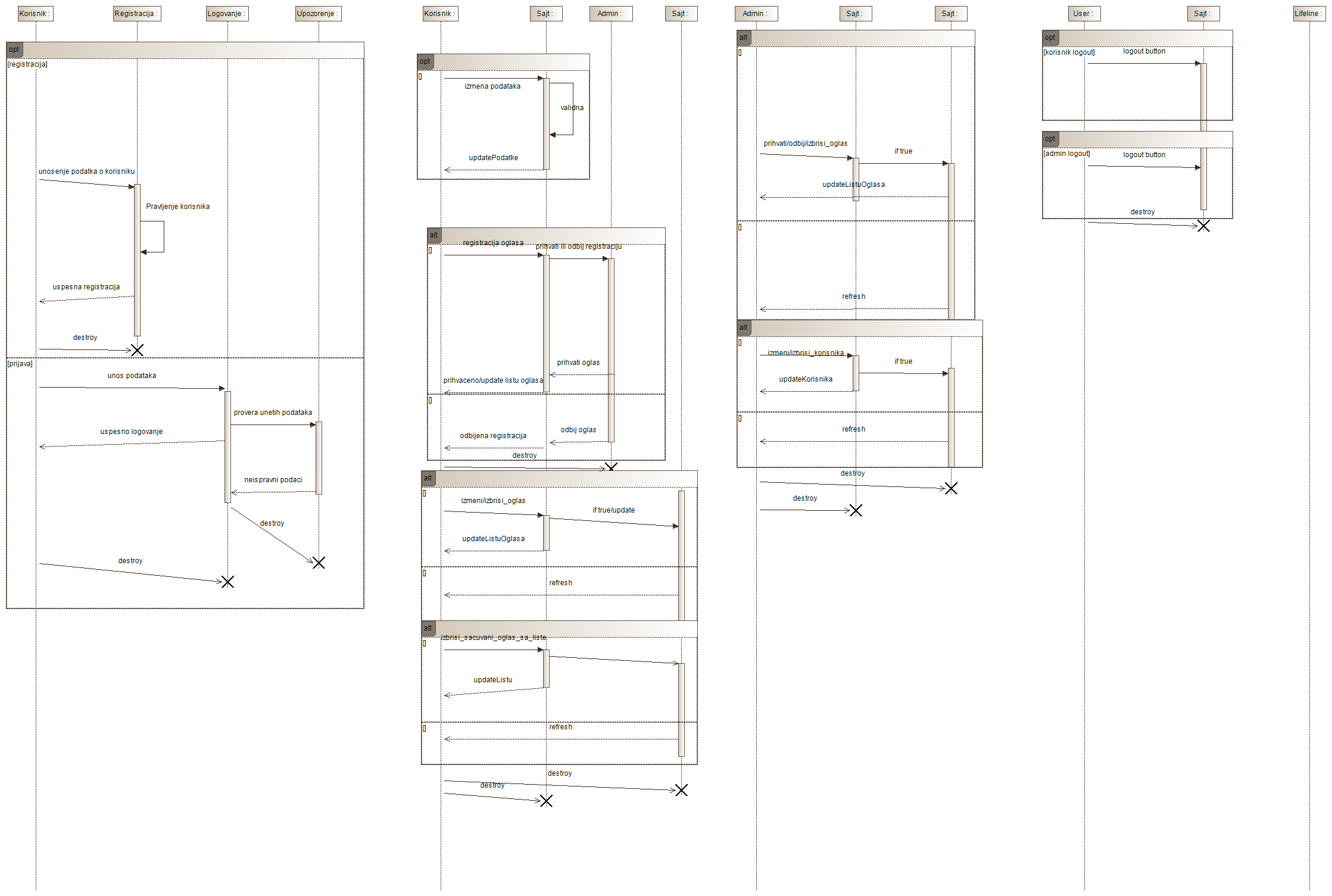
* + - Vertikalna (vremenska) - prikazuje životni vek objekata (hronološki red)
    - Horizontalna - kolekcija objekata i aktera (interakcija)

Kod dijagrama sekvenci vreme teče odozgo na dole; događaj iznad se desio pre. Može da sadrži: **aktere, objekte** i **poruke**.

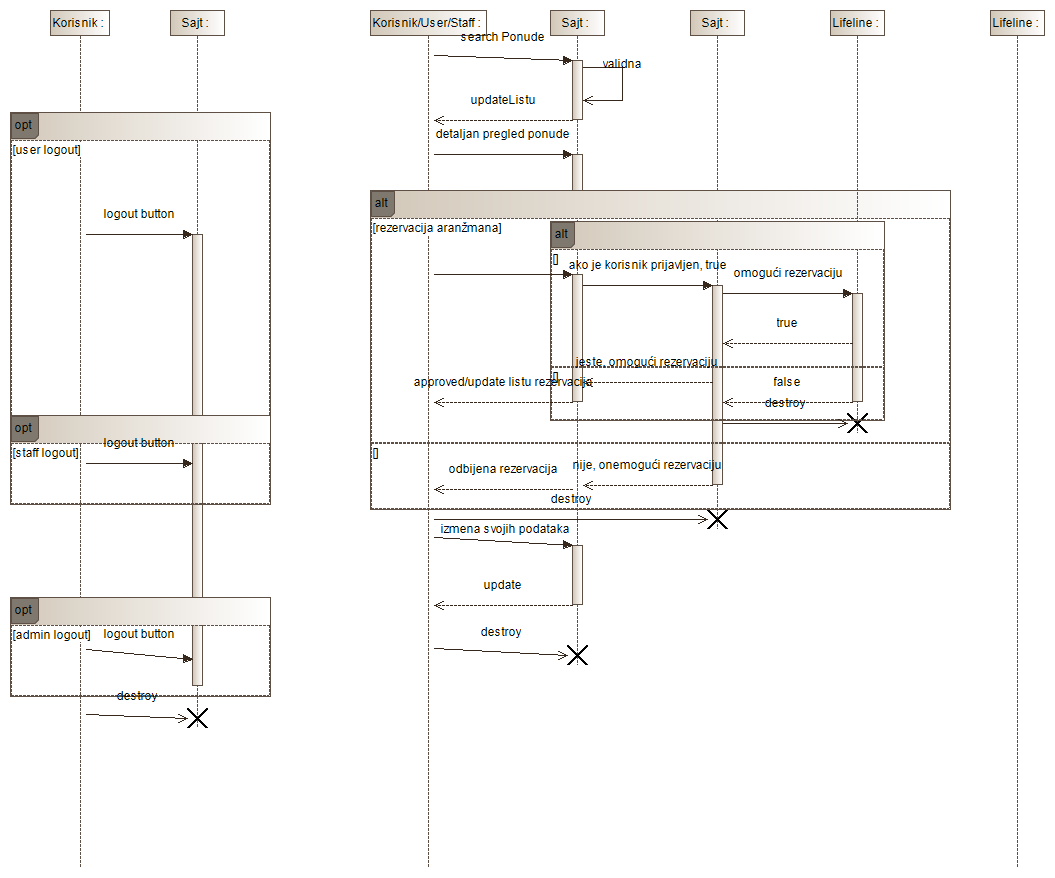
U nastavku su prikazani dijagrami sekvenci posmatranog sistema.

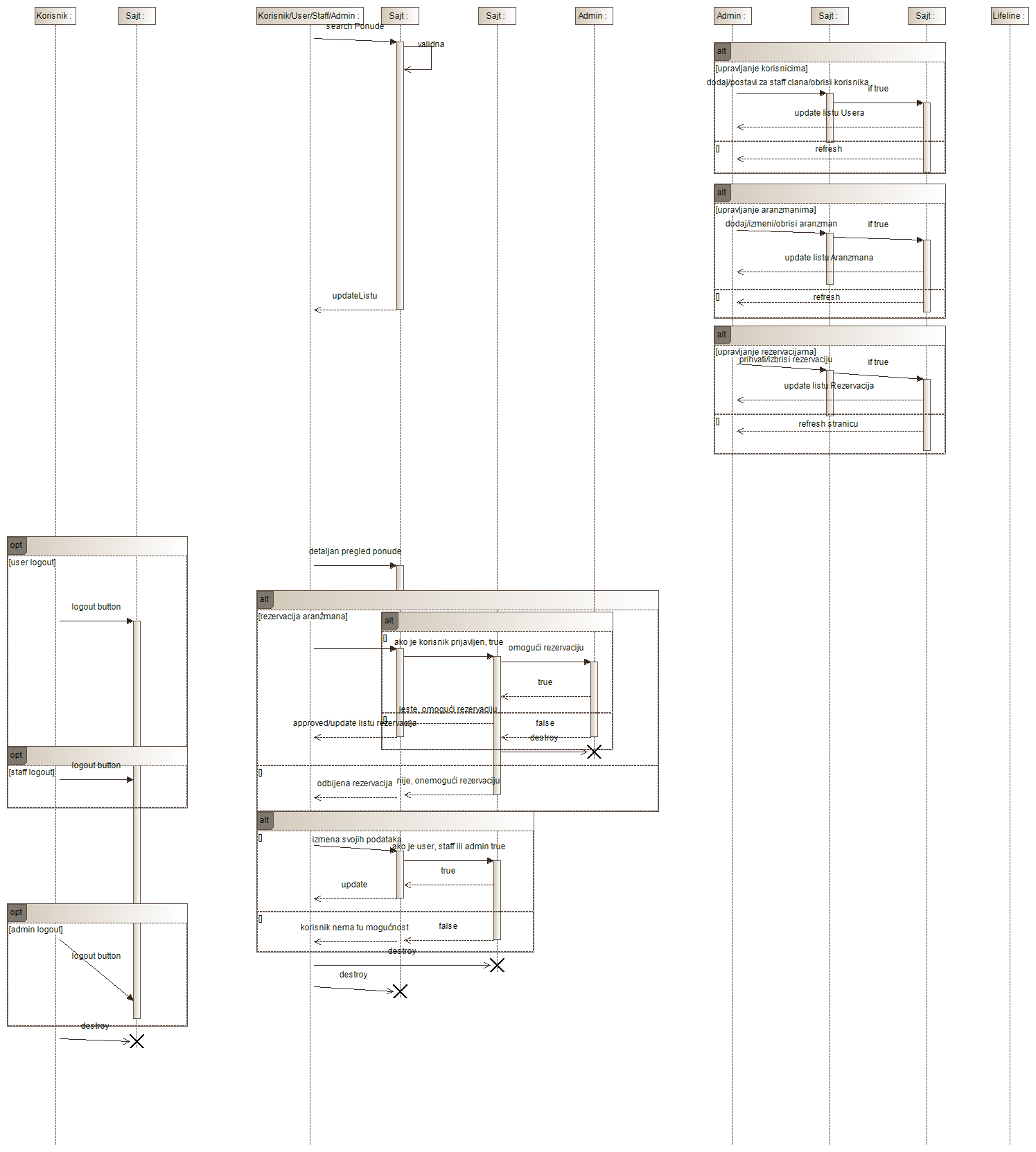
Na slici 36. prikazan je dijagram sekvence za korisnika. Korisnik pokušava da se uloguje na informacioni sistem i uneti podaci se proveravaju. Ukoliko su podaci validni, korisnik je prijavljen, inače, javlja se upozorenje.

Nakon uspešnog logovanja (slika 36.), adminu, useru i staffu je omogućen pristup profilu. Omogućena je rezervacija ponuda useru.



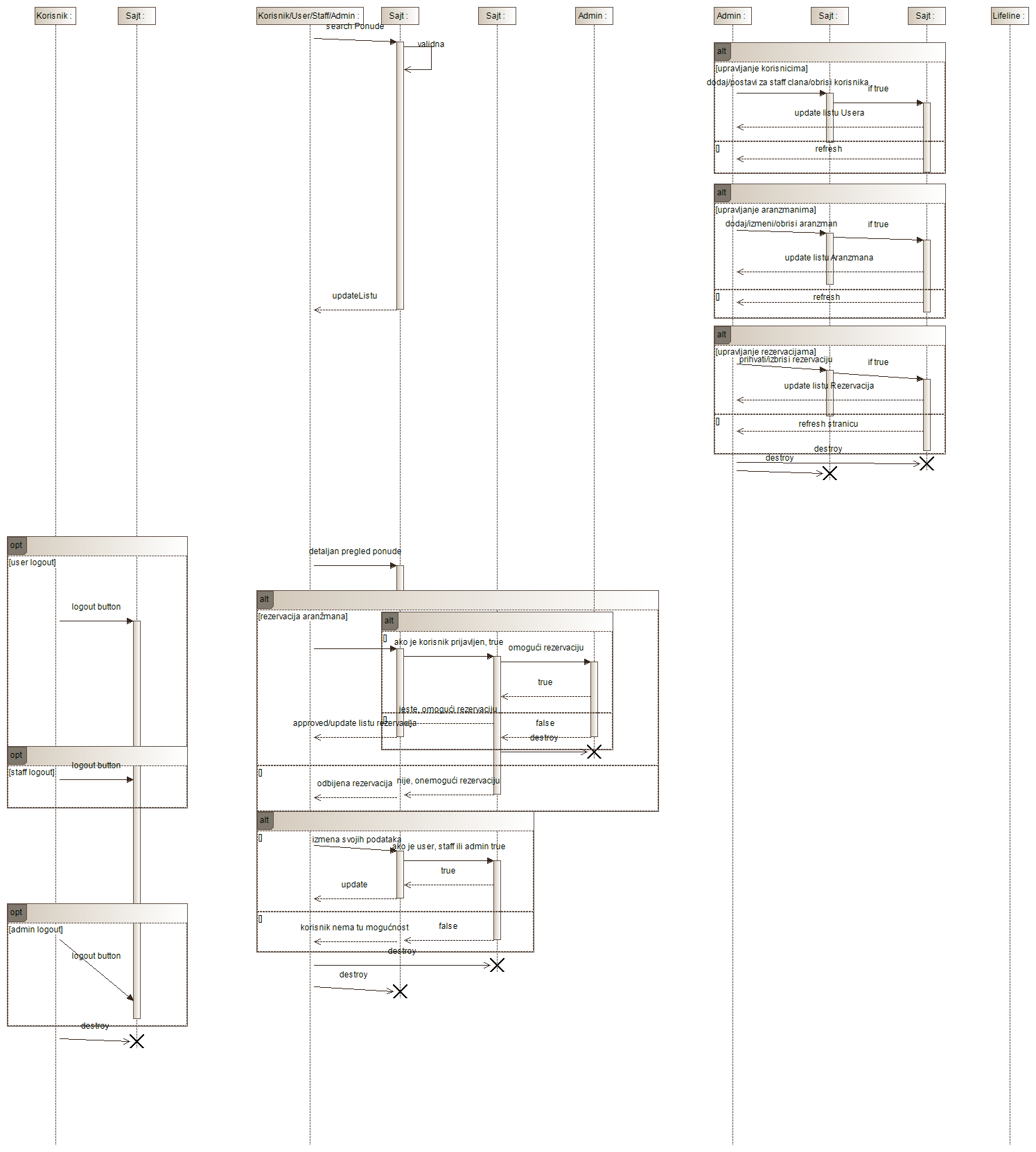
Slika 36. – Dijagram sekvenci za pravljenje korisnika





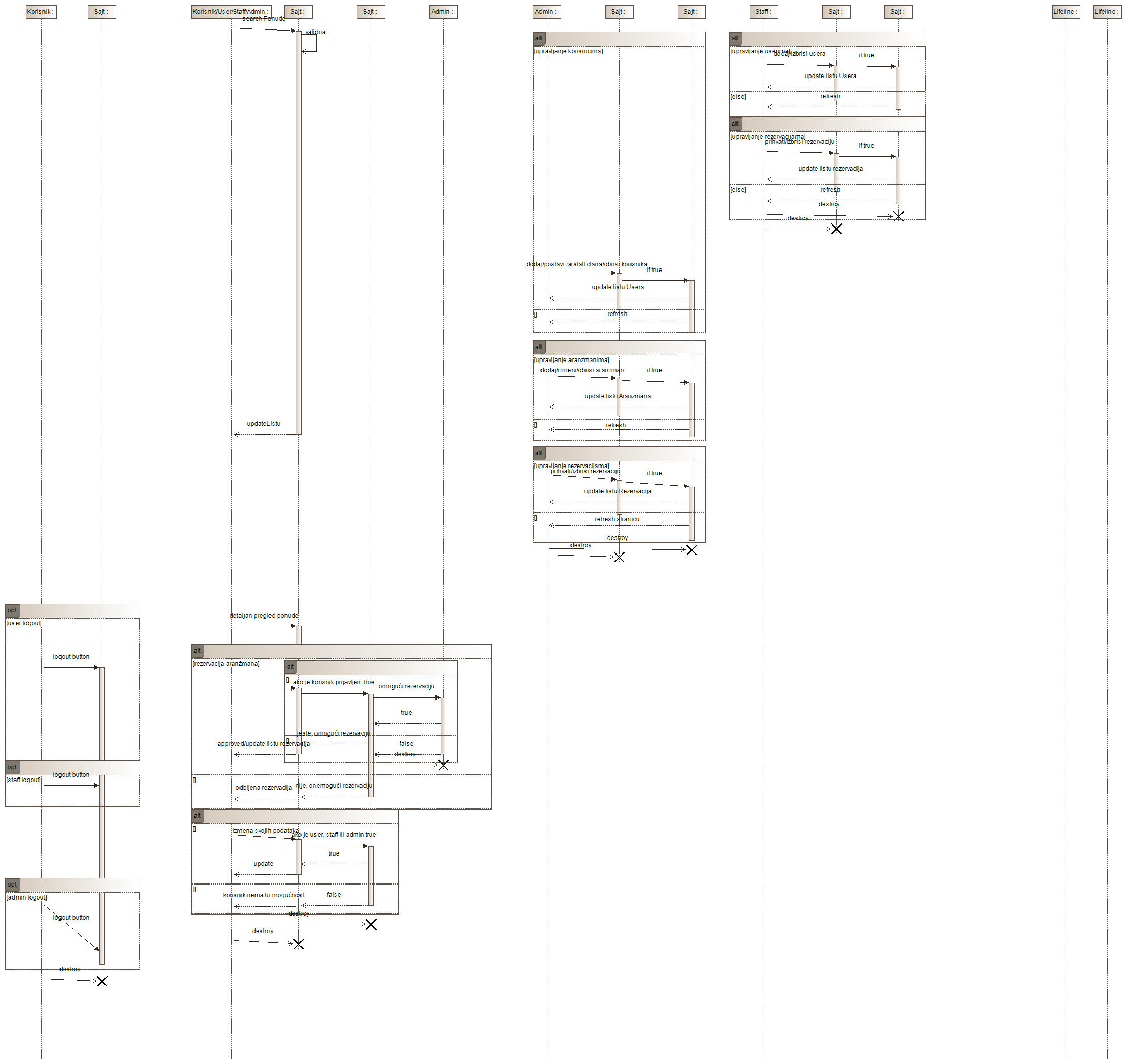
Slika 37. – Dijagram sekvenci za pretragu i rezervaciju aranžmana

Svakom korisniku je omogućena pretraga aranžmana kao i detaljniji pregled ponuda. Korisnik će se morati prijaviti za više mogućnosti, koje spadaju u gore navedenog korisnika. Admin postavlja ulogu korisniku da bi postao član staff-a. Svaki prijavljeni korisnik ima mogućnost izmene svojih ličnih podataka(slika 37.).



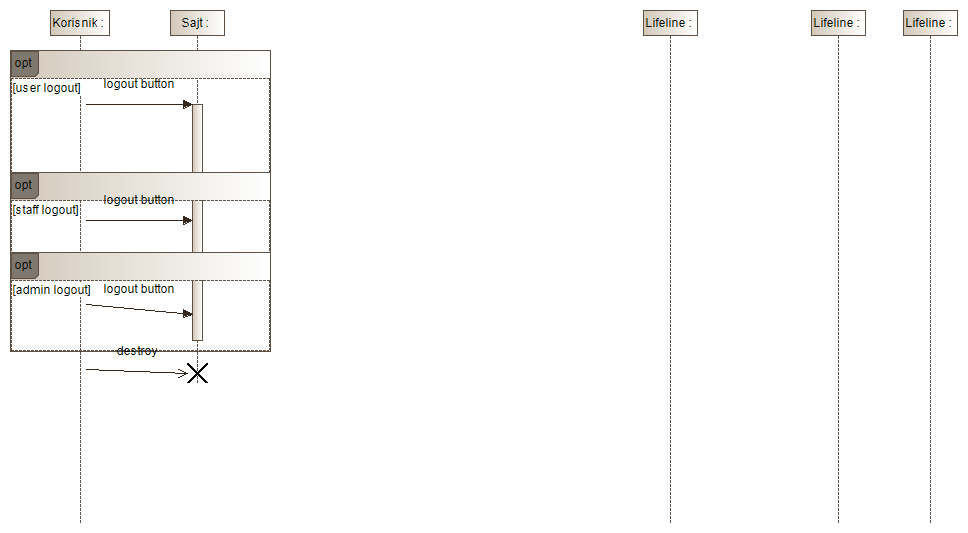
Slika 38. - Dijagram sekvenci za upravljanje podacima-Admin

Adminu je omogućen pristup profilu gde može upravljati podacima korisnika i aranžmana. On može da doda, postavi za člana staff-a usera ili da ga obriše. Dozvoljeno mu je da izmeni, doda i obriše aranžmane, kao i da prihvati i izbriše rezervacije(slika 38.). Ako su promene uspešno prošle, biće promenjene i u bazi.



Slika 39. – Dijagram sekvenci za upravljanje podacima-Staff

Adminu je omogućen pristup profilu gde može upravljati podacima korisnika i aranžmana. On može da doda, postavi za člana staff-a usera ili da ga obriše. Dozvoljeno mu je da izmeni, doda i obriše aranžmane, kao i da prihvati i izbriše rezervacije(slika 39). Ako su promene uspešno prošle, biće promenjene i u bazi.

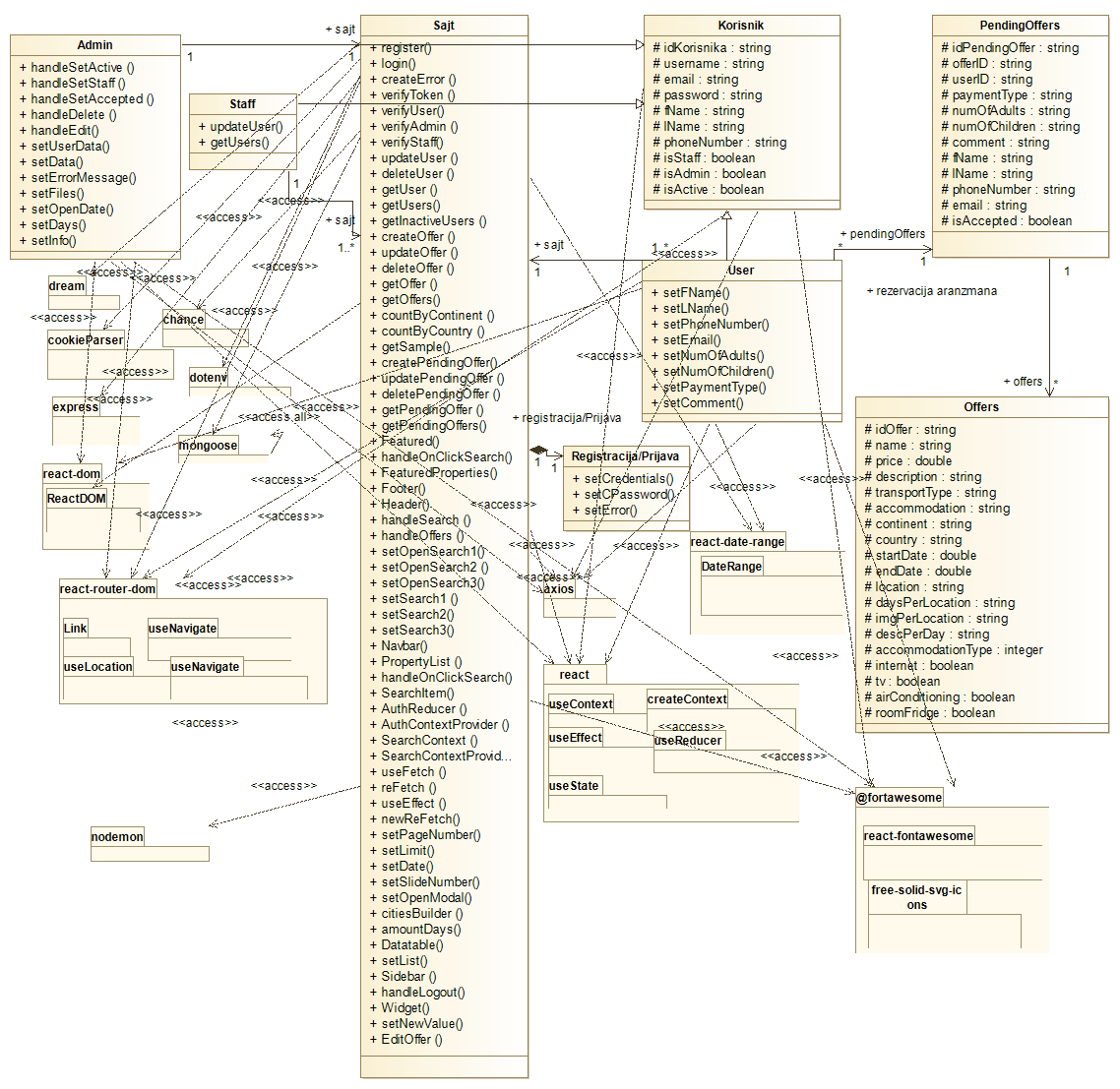


Slika 40. – logout dijagram sekvence

U zavisnosti da li user, admin ili staff želi da se odjavi sa profila, poslaće se poruke preko logout dugmeta.

***7.2. Dijagram klasa***

Dijagram klasa prikazuje skup klasa, interfejsa, saradnji i drugih stvari strukture, povezanim relacijama.



Slika 41. – Dijagram klasa

U našem slučajum klasa korisnik je klasa koju nasleđuju klase admin, user i staff. Ove tri klasa su u vezi sa klasom Sajt i to svaki user, admin i staff može da se prijavi na jedan i samo jedan sajt, dok svaki sajt može da ima jednog admina i jednog ili više prijavljenih usera i članova staff-a.

User može da rezerviše aranžman jedan ili više, mada ne mora ni jedan. Aranžman može biti ali i ne mora, od strane jednog ili više usera.

Sve ovo su bile veze asocijacije.

Klasa Sajt je sa klasom Registracija/Prijava povezana jakom agregacijom(kompozijija), to jest, klasa Registracija/Prijava ne bi postojala bez klase Sajt. Jedan sajt ima jedan i samo jedan login sistem, dok login sistem postoji za jedan i samo jedan sajt.

Klase Sajt, Admin, User i Staff imaju pristup gore navedenim bibliotekama.

# *8. Literatura*

1. Materijal sa moodle portala Fakulteta inženjerskih nauka: Softverski inženjering

<http://moodle.fink.rs/course/view.php?id=978>

[2] <https://www.npmjs.com/>

[3] <https://docs.docker.com/reference/>

[4] Materijal sa moodle portala : Projektovanje informacionih sistema i baza podataka

<http://moodle.fink.rs/course/view.php?id=986>

5. <https://stackoverflow.com/>

6. [https://www.infoworld.com/article/3388036/what-is-jdbc-introduction-to-java-database- connectivity.html](https://www.infoworld.com/article/3388036/what-is-jdbc-introduction-to-java-database-%20%20connectivity.html)

7. <https://www.geeksforgeeks.org/mern-stack/>

8. https://hub.docker.com/

9. <https://www.w3schools.com/>

10. <https://www.selenium.dev/>

11. <https://github.com/adleroliveira/dreamjs>

12. <https://chancejs.com/>

13. <https://www.mongodb.com/docs/database-tools/mongoimport/>

14. <https://mui.com/material-ui/material-icons/>

15. <https://fontawesome.com/icons>

16. <https://www.npmjs.com/package/react-date-range>

17. <https://axios-http.com/docs/intro>

18. <https://reactjs.org/docs/react-dom.html>

19. <https://www.npmjs.com/package/react-spinners>

20. <https://www.npmjs.com/package/node-modules>

21. <https://mongoosejs.com/>

22. <https://www.npmjs.com/package/dotenv>

23. <https://www.npmjs.com/package/react-router-dom>

24. <https://www.npmjs.com/package/cookie-parser>