

Programsko inženjerstvo

Ak. god. 2023./2024.

SpotPicker

Dokumentacija, Rev. 1

Grupa: Šargarepoljupci

Voditelj: Matko Krnić

Datum predaje: 17. studenog. 2023.

Nastavnik: Hrvoje Nuić

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	22
3.2 Ostali zahtjevi	26
4 Arhitektura i dizajn sustava	27
4.1 Baza podataka	29
4.1.1 Vrsta i implementacija	29
4.1.2 Glavne komponente baze podataka	29
4.1.3 Opis tablica	30
4.1.4 Dijagram baze podataka	33
4.2 Dijagram razreda	34
4.3 Dijagram stanja	41
4.4 Dijagram aktivnosti	42
4.5 Dijagram komponenti	44
5 Implementacija i korisničko sučelje	46
5.1 Korištene tehnologije i alati	46
5.2 Ispitivanje programskog rješenja	48
5.2.1 Ispitivanje komponenti	48
5.2.2 Ispitivanje sustava	55
5.3 Dijagram razmještaja	57
5.4 Upute za puštanje u pogon	58
6 Zaključak i budući rad	59
Popis literature	60

Indeks slika i dijagrama	63
Dodatak: Prikaz aktivnosti grupe	64

1. Dnevnik promjena dokumentacije

Kontinuirano osvježavanje

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	matkokrnic	16.10.2023.
0.2	Dodan opis projektnog zadatka.	JureRoncevic	29.10.2023.
0.3	Dodani opis obrazaca uporabe u specifikaciji programske potpore.	matkokrnic	30.10.2023.
0.4	Dodane neke osnovne funkcionalosti za backend	piilip	11.11.2023.
0.4.1	Popravljene greške za backend	piilip	11.11.2023
0.5	Dodani dijagrami obrazaca uporabe	MegaNoris	12.11.2023.
0.5.1	Ispravak pravopisnih pogrešaka	MegaNoris	12.11.2023.
0.6	Napravljen backend login	matkokrnic	13.11.2023.
0.6.1	Ostvarena registracija za backend	matkokrnic	13.11.2023.
0.7	Napravljene izmjene u dijagramima obrazaca uporabe	MegaNoris	14.11.2023.
0.8	Dodane slike za ER dijagram i relacijsku shemu	piilip	14.11.2023.
0.8.1	Napravljene tablice i opis baze podataka	piilip	14.11.2023.
0.9	Dodana mapa .idea pod mapom Izvorni-Kod sa svim potrebnim datotekama	piilip	14.11.2023.
0.10	Dodani sekvencijski dijagrami u mapu slike	romgko	15.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.11	Dodana prva verzija frontenda	domajenjic	16.11.2023.
0.11.1	Dodani komentari za frontend	domajenjic	16.11.2023.
0.12	Dodana arhitektura i dizajn sustava	JureRoncevic	17.11.2023.
0.12.1	Dodani obrasci uporabe i sekvencijski dijagrami u projektnu dokumentaciju	JureRoncevic	17.11.2023.
0.13	Dodan backend API i pomoćne klase	piilip	17.11.2023.
0.14	Dodani opisi uz sekvencijske dijagrame	JureRoncevic	17.11.2023.
0.15	Ažuriran i popravljen dnevnik promjena dokumentacije	JureRoncevic	17.11.2023.
0.16	Popravljena arhitekutra, ažuriran dnevnik sastajanja	JureRoncevic	17.11.2023.
0.17	Dodane reference i popravljen opis projektnog zadatka	JureRoncevic	17.11.2023.
0.18	Ažurirana tablica aktivnosti i dodani dijagrami razreda	JureRoncevic	17.11.2023.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	JureRoncevic	17.11.2023.

Moraju postojati glavne revizije dokumenata 1.0 i 2.0 na kraju prvog i drugog ciklusa. Između tih revizija mogu postojati manje revizije već prema tome kako se dokument bude nadopunjavao. Očekuje se da nakon svake značajnije promjene (dodataka, izmjene, uklanjanja dijelova teksta i popratnih grafičkih sadržaja) dokumenta se to zabilježi kao revizija. Npr., revizije unutar prvog ciklusa će imati oznake 0.1, 0.2, ..., 0.9, 0.10, 0.11.. sve do konačne revizije prvog ciklusa 1.0. U drugom ciklusu se nastavlja s revizijama 1.1, 1.2, itd.

2. Opis projektnog zadatka

Ideja našeg projekta je ostvariti aplikaciju koja će omogućiti korisniku da pregleđava, rezervira i naplaćuje parkirna mjesta za automobile i bicikle. Aplikacija to ostvaruje na način da prilikom njezinog otvaranja prikazuje korisniku kartu nad kojom korisnik može birati željeno odredište, tip vozila koje koristi te trajanje parkiranja. Nakon svih odabira, na karti mu se iscrtava najbrža ruta do slobodnog parkirališnog mesta te ako je mjesto slobodno, ono se rezervira. Parkirališta za bicikle nemaju ucrtana pojedinačna mjesta te se ne rezerviraju ni naplaćuju, već aplikacija jedino prati ukupan broj slobodnih mesta. Za računanje najbližeg dostupnog parkirališnog mesta aplikacija koristi OSRM, što je kratica za Open Source Routing Machine. Riječ je o modernom C++ algoritmu za usmjeravanje koji se koristi za izračunavanje najkraćeg puta u cestovnom prometu.

Aplikacija koristi API za više funkcija:

- implementacija novčanika u koji korisnik uplaćuje sredstva
- osvježavanje parkirališnog mesta radi provjere zauzetosti
- dohvaćanje početnih informacija o stanju parkirališnih mesta pri čemu se koristi overpass API

Aplikacija se može pokrenuti bez registracije, no za pristup svim alatima aplikacije, potrebna je registracija. Korisnici koji nisu registrirani mogu na karti vidjeti sva dostupna parkirališta i parkirališna mjesta, ali nemaju informacije u njihovom stanju u stvarnom vremenu. Neregistrirani korisnik koji se želi registrirati šalje zahtjev za registraciju pri čemu ima opciju birati između dvije uloge (voditelj parkirališta i klijent). Za registraciju su potrebni:

- korisničko ime
- ime
- prezime
- slika osobne
- IBAN račun
- email adresa

Za registraciju kao klijent dovoljna je samo potvrda preko email adrese. Ako je riječ o voditelju parkirališta tada je potrebna i potvrda administratora.

Klijent otvaranjem aplikacije dobiva pregled nad kartom te odabire lokaciju odredišta, tip vlastitog vozila, i trajanje parkiranja. Klijent može rezervirati parkirališno mjesto na dva načina. Može prvo označiti parkirališna mjesta koja mu odgovaraju te mu tada aplikacija nudi moguće termine u kojima će ta mjesta biti slobodna. Druga opcija je da prvo definira odgovarajući termin nakon čega će mu aplikacija prikazati sva parkirališna mjesta koja su slobodna u tom terminu. Aplikacija razlikuje parkirališna mjesta čije su razine različite te korisnik ima mogućnost odabira. Kada je klijent zadovoljan s parkirališnim mjestom, rezervira mjesto na proizvoljno vrijeme ako je slobodno te pri tome ima opciju da napravi rezervaciju ponavlјajućom. Klijent jedino može rezervirati mjesto u budućnosti, odnosno, mjesto ne može biti rezervirano u istom datumu u kojem je korisnik otvorio aplikaciju. Plaćanje parkiranja korisnik obavlja tijekom rezervacije ili po dolasku na parkirališno mjesto. Plaćanje se obavlja preko novčanika koji je implementiran unutar aplikacije u koji korisnik može proizvoljno ubaciti novac.

Voditelj parkirališta ima veće ovlasti od klijenta. On ima dozvolu unijeti podatke o vlastitom parkiralištu koje uključuju:

- naziv
- opis
- fotografija
- cjenik i sl.

Voditelj nad svojim parkiralištem može individualno ucrtati svako parkirališno mjesto prema vlastitim željama. Nakon što ucrtava mjesta, on odlučuje je li mjesto dostupno za rezervaciju. Dodatno, voditelj postavlja i senzor koji služi za dohvatanje informacije o zauzetosti mjesta. Za svoje parkiralište voditelj određuje cijenu rezervacije ovisnu o vremenu. Voditelj također ima i pristup statističkim podacima o svom parkiralištu. Moguće je promatrati zauzetost cijelog parkirališta te pojedinačnih parkirališnih mjesta kroz vrijeme uz pomoć grafa.

Administrator aplikacije ima najveće ovlasti i sposoban je vidjeti popis svih registriranih korisnika i njihovih osobnih podataka te im po potrebi može mijenjati osobne podatke.

Motivacija za razvoj ovog projekta dolazi od mnogih prednosti koje aplikacija za parkiranje može donijeti u odnosu na klasičan način rezerviranja i plaćanja parkiranja. Za početak, korištenjem ove aplikacije korisniku se značajno smanjuje

vrijeme potrebno za pronalaženje parkirališnog mjesta. Dodatno, plaćanje parkirališnog mjesta može biti izvor frustracije zbog korištenja kovanica i gomilanja sitniša ili zbog potrebe za vađenjem kartice. Aplikacija za taj problem nudi praktično rješenje implementacijom novčanika što omogućuje korisniku da sva svoja plaćanja obavi unaprijed preko mobilnog telefona. Jednostavnost i elegantnost ove aplikacije pomaže i pri smanjenju stresa kod korisnika. Pitanja poput: "Gdje ćemo parkirati?", "Smijem li ja parkirati na ovom mjestu?" ili "Ima li uopće slobodnih mjesta?" više neće biti razlog za brigu jer aplikacija obavlja sve razmišljanje za vas. Problem traženja parkirališnog mjesta u nepoznatom gradu će se svesti na samo par klikova na aplikaciji te će uštedjeti korisnicima i novac budući da neće gubiti vrijeme vozajući se po gradu, tražeći slobodno mjesto. Konačno, aplikacija pridonosi i okolišu jer manje vremena potrošeno na potražnju mjesta za parkiranje znači manje vremena koje trošimo na sagorijevanje goriva.

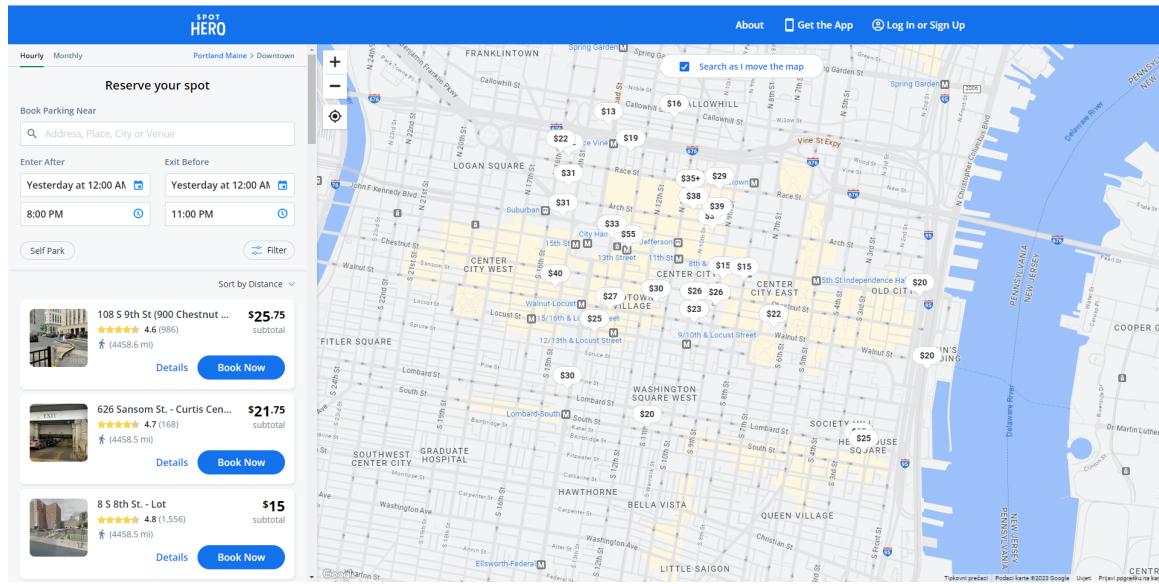
Neke od aplikacija koje pružaju slična rješenja su:

- SpotHero
- Best in Parking
- ParkWhiz
- Parkopedia
- Way

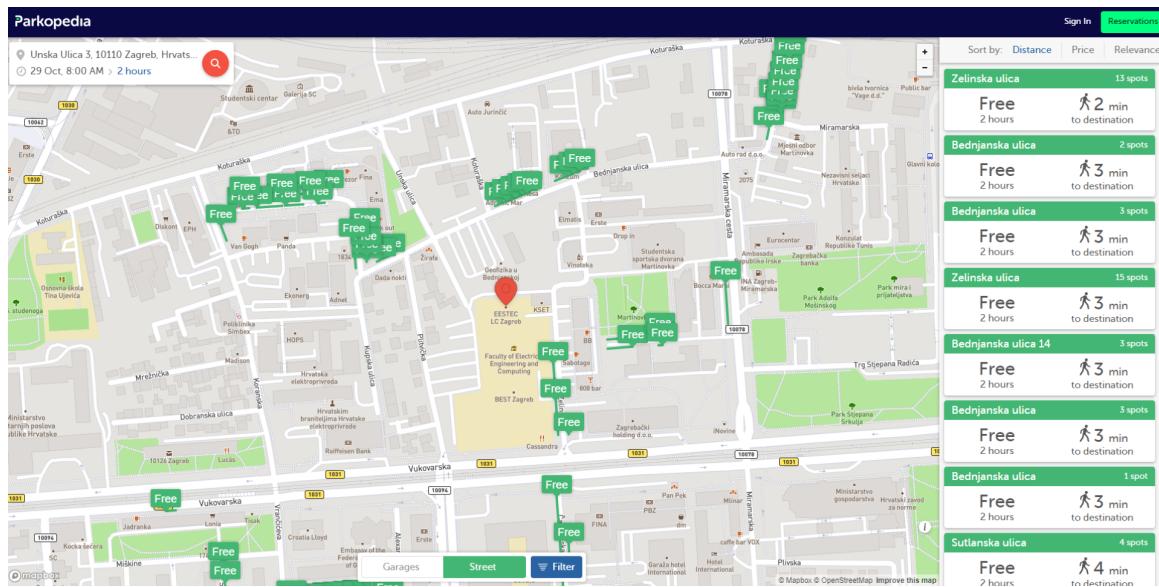
Kako bi bolje ilustrirali prednosti koje naša aplikacija ima u odnosu na neke od postojećih rješenja, uzet ćemo kao primjer *SpotHero*. *SpotHero* korisniku prikazuje dostupna parkirališna mjesta u blizini nakon što korisnik upiše adresu na kojoj se nalazi. Korisnik tada može kartu rezervirati i tada dobiva propusnicu koju skenira na parkirališnom mjestu. *SpotHero* omogućuje korisniku mjesecno parkiranje te parkiranje u zračnoj luci. Bitna razlika između naše aplikacije i *SpotHero* je to da kod korištenja *SpotHero-a* korisnik mora odmah definirati termin i vrijeme parkiranja dok naša aplikacija dozvoljava korisniku da prvo označi parkirališna mjesta, a nakon toga mu se nude dostupni termini. Aplikacije se razlikuju i u tome što *SpotHero* ne daje opciju za bicikle, već samo za osobna vozila. Dodatno, *SpotHero* je jedino dostupan u SAD-u i Kanadi.

Parkopedia je još jedna takva slična aplikacija. *Parkopedia* koristi Google Maps kako bi iscrtala najbližu rutu korisniku. Razlika između našeg projekta i ove aplikacije je što *Parkopedia* zahtijeva da se prvo uneše adresa odredišta nakon čega nudi popis dostupnih parkirališta. *Parkopedia* na prikazu karte daje opciju između prikazivanja parkirališnih mjesta na ulici i u garaži. Kao i kod *SpotHero-a*, *Parkopedia*

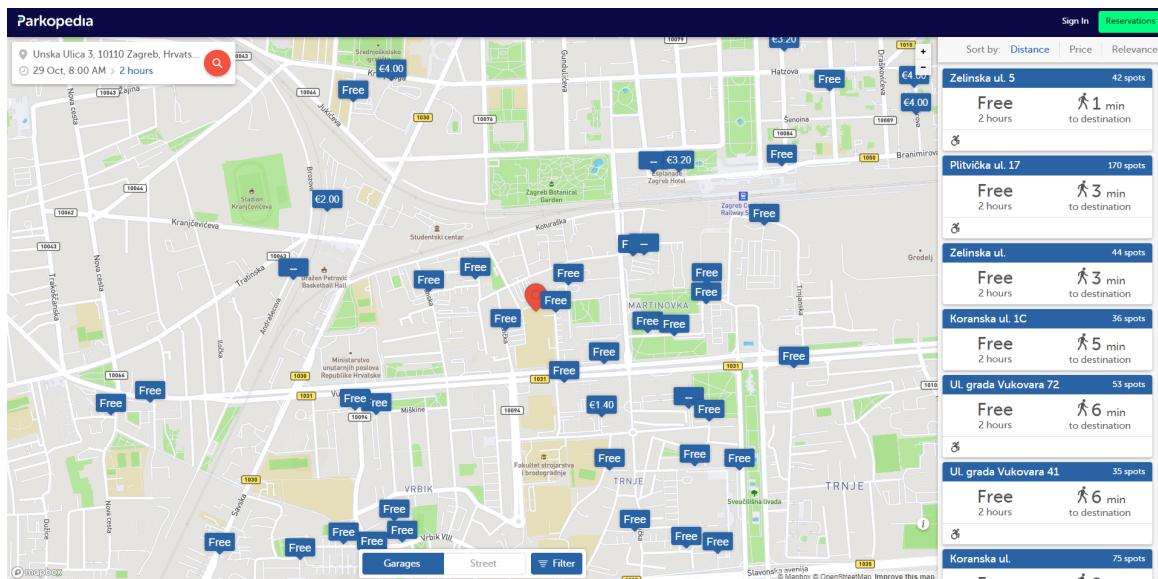
ne nudi opciju za bicikle te pretraga prvo po terminu parkiranja nije moguća.



Slika 2.1: Prikaz aplikacije SpotHero



Slika 2.2: Prikaz aplikacije Parkopedia s označenim parkiralištima na ulici



Slika 2.3: Prikaz aplikacije Parkopedia s označenim parkiralištima na garaži

Vjerujemo da našom aplikacijom možemo zainteresirati ljude koji su već upoznati s parking aplikacijama, ali smatraju da postoje područja u kojima trenutačno najpopularnije aplikacije nisu dovoljno prilagodljive korisničkim željama. Konačan plan i opseg ovog projekta je učiniti ovu aplikaciju dostupnu cijeloj Hrvatskoj. Ovisno o potrebi i potražnji korisnika, aplikaciju je moguće u budućnosti i nadograditi implementiranjem brojača slobodnih mesta za bicikle.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Voditelj parkirališta
2. Klijent
3. Administrator
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) pregledati na karti pozicije svih parkirališta i dostupnim mjestima na parkiralištima
 - (b) poslati zahtjev za registraciju, a potrebni su: korisničko ime, lozinka, ime, prezime, slika osobne, IBAN račun i email adresa
 - i. moguće se registrirati kao klijent
 - ii. moguće se registrirati kao voditelj parkirališta
2. Klijent (inicijator) može:
 - (a) uz pregled parkirališta te dostupnih mesta istih ima uvid u aktualnu zauzetost parkirnih mesta
 - (b) unijeti odredište, tipa vozila i procijenjenu duljinu trajanja parkiranja čime aplikacija generira rutu do najbližeg slobodnog parkirališta te rezervira parkirno mjesto ako postoji slobodno mjesto za rezervaciju
 - (c) rezervirati parkiranje:
 - i. prema slobodnim terminima tražene lokacije
 - ii. prema slobodnim lokacijama u traženom terminu
 - (d) nadopuniti novčanik sredstvima za plaćanje parkiranja
 - (e) pregledavati i mijenjati osobne podatke
 - (f) obrisati račun

3. Voditelj parkirališta(inicijator) može:

- (a) unijeti informacije o svom parkiralištu (naziv, opis, fotografija, cjenik i sl.)
 - i. u kartu ucrtati svako dostupno parkirališno mjesto za to parkiralište
- (b) definirati je li moguće rezervirati parkirališno mjesto
- (c) definirati cijenu ovisno o trajanju rezervacije
- (d) pristupiti statistici zauzetosti parkirališta i parkirališnih mjesta kroz vrijeme
- (e) obrisati parkirno mjesto
- (f) mijenjati informacije o svom parkiralištu
- (g) obrisati račun

4. Administrator (inicijator) može:

- (a) potvrditi neregistriranog korisnika kao voditelja parkirališta
- (b) pristupiti statistici zauzetosti parkirališta i parkirališnih mjesta kroz vrijeme
- (c) dodati ili obrisati parkiralište i parkirna mjesta
- (d) pristupiti i izmijeniti podatke o parkiralištu
- (e) korisnike brisati i mijenjati im razinu pristupa aplikaciji
- (f) obrisati parkiralište

5. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o parkiralištima i stanja zauzetosti parkirališnih mjesta

3.1.1 Obrasci uporabe

UC1 -Pregled parkirališta i parkirališnih mjesta na karti

- **Glavni sudionik:** Klijent, Neregistrirani korisnik
- **Cilj:** Pregledati dostupna parkirna mjesta
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Učitavanjem aplikacije prikazuje se karta s ucrtanim parkiralištima
 2. Korisnik odabire parkiralište
 3. Prikazuju se dostupna parkirališna mjesta za odabranu parkiralište te informacije o parkiralištu
- **Opis mogućih odstupanja:**
 - 3.a Glavni sudionik je registrirani Klijent
 1. Dodatno se prikazuje zauzetost parkirališnih mjesta u stvarnom vremenu

UC2 -Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Korisnik unosi potrebne korisničke podatke
 3. Korisnik odabire vrstu registracije
 4. Korisnik registraciju potvrđuje preko maila
- **Opis mogućih odstupanja:**
 - 2.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnoga e-maila
 1. Sustav upozorava korisnika na probleme u registraciji i vraća ga na stranicu registracije
 - 3.a Odabrana je registracija računa Voditelja parkirališta
 1. Administrator mora dodatno potvrditi takvu registraciju

UC3 -Prijava u sustav

- **Glavni sudionik:** Klijent
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Potvrda o ispravnosti unesenih podataka
 3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime/lozinka
 1. Sustav obaveštava korisnika o neuspjeloj prijavi i vraća ga na stranicu za prijavu

UC4 -Pregled osobnih podataka

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Osobni podatci"
 2. Otvara se stranica s osobnim podacima korisnika

UC5 -Promjena osobnih podataka

- **Glavni sudionik:** Klijent
- **Cilj:** Promijeniti osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Osobni podatci"
 2. Otvara se stranica s osobnim podacima korisnika
 3. Korisnik odabere opciju za promjenu podataka
 4. Korisnik mijenja svoje osobne podatke
 5. Korisnik sprema promjene
 6. Baza podataka se ažurira
- **Opis mogućih odstupanja:**

- 2.a Korisnik promijeni svoje osobne podatke, ali ne odabere opciju "Spremi promjene"
 1. Sustav obavještava korisnika da nije spremio podatke prije izlaska iz prozora
- 4.a Korisnik promijeni svoje osobne podatke, ali unese neispravne podatke
 1. Sustav obavještava korisnika da podatci nisu ispravni te traži od korisnika ponovni unos podataka.
- 5.a Korisnik promijeni svoje osobne podatke, ali odustane od izmjena prije spremanja promjena
 1. Sustav obavještava korisnika da će se napravljene izmjene u slučaju izlaska odbaciti.

UC6 -Brisanje korisničkog računa

- **Glavni sudionik:** Klijent
- **Cilj:** Obrisati korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju "Osobni podatci"
 2. Otvara se stranica s osobnim podacima korisnika
 3. Korisnik odabere opciju brisanja računa
 4. Korisnički račun se izbriše iz baze podataka
 5. Otvara se stranica za registraciju

UC7 -Rezervacija parkirnog mjesta

- **Glavni sudionik:** Klijent
- **Cilj:** Rezervirati parkirališno mjesto
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen, Klijent (ili aplikacija) je odabrao mjesto, datum i vrijeme parkiranja
- **Opis osnovnog tijeka:**
 1. Baza podataka zapiše podatke o rezervaciji
 2. Parkiranje se naplati pri rezervaciji ili po dolasku na parkirališno mjesto
- **Opis mogućih odstupanja:**
 - 1.a Klijent rezervaciju definira kao ponavljaču
 1. Rezervacija se spremi u bazu podataka u posebnom obliku

UC8 -Odabir rezervacije parkirnog mesta

- **Glavni sudionik:** Klijent
- **Cilj:** Rezervirati parkirališno mjesto
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Klijent bira termin ili parkirališno mjesto te upisuje predviđeno trajanje rezervacije
 2. Ako je Klijent odabrao termin nude mu se odgovarajuća slobodna parkirališna mjesta, a ako je Klijent odabrao parkirališno mjesto nude mu se dostupni termini za dotično mjesto
 3. Klijent odabire mjesto, datum i trajanje rezervacije
 4. Klijent odabire je li rezervacija ponavljača
- **Opis mogućih odstupanja:**
 - 1.a Klijent odabire današnji datum ili datum u prošlosti
 1. Sustav upozori korisnika da se ne može rezervirati mjesto na datum odabira
 - 3.a Klijent odabire današnji datum ili datum u prošlosti
 1. Sustav upozori korisnika da se ne može rezervirati mjesto na datum odabira

UC9 -Dohvat parkirališta

- **Glavni sudionik:** Klijent
- **Cilj:** Pronaći put do najbližeg parkirališta i rezervirati ako je moguće
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Klijent odabire odredište na karti, tip vozila i procjenu trajanja parkiranja
 2. Aplikacija iscrta rutu na mapi do najbližeg slobodnog parkirališnog mesta
 3. Mjesto se rezervira ako je slobodno za rezervaciju

UC10 -Pregled rezervacija

- **Glavni sudionik:** Klijent
- **Cilj:** Pregled aktivnih rezervacija

- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen i napravio je barem jednu rezervaciju
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju "Moje rezervacije"
 2. Prikaže se lista rezervacija korisnika

UC11 -Uređivanje rezervacije

- **Glavni sudionik:** Klijent
- **Cilj:** Uređivanje aktivne rezervacija
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen i napravio je barem jednu rezervaciju
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju "Moje rezervacije"
 2. Prikaže se lista rezervacija korisnika
 3. Klijent odabire rezervaciju koju želi urediti
 4. Prikaže se lista dostupnih parkirnih mjesta i slobodnih termina za iste
 5. Korisnik čini promjene te ih sprema
- **Opis mogućih odstupanja:**
 - 3.a Korisnik odabire rezervacije koja će se uskoro ostvariti (npr. u sljedećih 2 sata)
 1. Sustav obavještava korisnika da se rezervacija ne može mijenjati

UC12 -Brisanje rezervacije

- **Glavni sudionik:** Klijent
- **Cilj:** Otkazati zakazanu rezervaciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen i postoji barem jedna aktualna rezervacija
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju "Moje rezervacije"
 2. Prikaže se lista rezervacija korisnika
 3. Klijent bira opciju za brisanje rezervacije
 4. Rezervacija se uklanja iz Baze podataka

UC13 -Plaćanje

- **Glavni sudionik:** Klijent
- **Cilj:** Platiti parkiranje

- **Sudionici:** Baza podataka, Voditelj parkirališta
- **Preduvjet:** Klijent je prijavljen, parkiralište se plača aplikacijom
- **Opis osnovnog tijeka:**
 1. Uzme se određena količina iz računa Klijenta
 2. Stavlja se na račun Voditelja parkirališta koje se rezervira
 3. Rezervacija se označi plaćenom
- **Opis mogućih odstupanja:**
 - 1.a Klijent nema dovoljno novca na računu
 1. Sustav upozori korisnika da nema dovoljno novca
 2. Sustav prekine sa tijekom izvođenja

UC14 -Uplata

- **Glavni sudionik:** Klijent
- **Cilj:** Uplatiti novac na račun
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Klijent odabere opciju uplate novca
 2. Klijent upiše podatke računa/kartica s koje se uzima novac i koliko novca želi prenijeti
 3. Potvrđuje se transakcija
 4. Dodaju se novi novci na račun
- **Opis mogućih odstupanja:**
 - 2.a Krivo upisane informacije računa/kartice
 1. Sustav upozori korisnika da je krivo upisao podatke
 2. Sustav vrati korisnika na stranicu upisa podataka
 - 3.a Nedovoljno novca na računu
 1. Sustav ipozori korisnika da nema dovoljno novca na računu
 2. Sustav vrati korisnika na stranicu upisa podataka

UC15 -Pregled aktivnih rezervacija parkirališta

- **Glavni sudionik:** Voditelj parkirališta
- **Cilj:** Pregledati sve aktualne rezervacije
- **Sudionici:** Baza podataka
- **Preduvjet:** Rezervacije su zaprimljene i plaćene te voditelj je prijavljen
- **Opis osnovnog tijeka:**

1. Voditelj odabere opciju pregleda aktivnih rezervacija
2. Prikaže se lista aktivnih rezervacija

UC16 -Ucrtavanje novog parkirnog mjesta

- **Glavni sudionik:** Voditelj parkirališta
- **Cilj:** Ucrtati novo parkirno mjesto u sklopu postojećeg parkirališta
- **Sudionici:** Baza podataka
- **Preduvjet:** Voditelj je prijavljen
- **Opis osnovnog tijeka:**
 1. Voditelj odabire opciju prikaza tlocrta parkirališta
 2. Prikazuje mu se tlocrt parkirališta
 3. Voditelj odabire opciju za ucrtavanje novog parkirnog mjesta
 4. Voditelj ucrtava na tlocrt novo parkirno mjesto
 5. promjene se upisuju u Bazu podataka

UC17 -Uređivanje podacima o parkiralištu

- **Glavni sudionik:** Voditelj parkirališta
- **Cilj:** Izmijeniti informacije o parkiralištu
- **Sudionici:** Baza podataka
- **Preduvjet:** Voditelj parkirališta je prijavljen
- **Opis osnovnog tijeka:**
 1. Voditelj parkirališta bira prikaz opciju prikaza informacija o parkiralištu
 2. Prikazuju mu se ranije unesene informacije
 3. Voditelj bira opciju izmjene dotičnih podataka
 4. Izmijenjeni podatci se spremaju u Bazu podataka

UC18 -Uklanjanje parkirnog mjesta

- **Glavni sudionik:** Voditelj parkirališta
- **Cilj:** Ukloniti željena parkirna mjesta
- **Sudionici:** Baza podataka
- **Preduvjet:** Voditelj parkirališta je prijavljen, postoji ucrtana parkirna mjesto
- **Opis osnovnog tijeka:**
 1. Voditelj bira opciju prikaza tlocrta parkirališta
 2. Voditelj potom bira opciju uklanjanja parkirnog mjesta
 3. Odabrana uklonjena parkirna mjesto se uklanjuju iz Baze podataka

UC19 -Potvrđivanje voditelja parkirališta

- **Glavni sudionik:** Administrator
- **Cilj:** Potvrditi registraciju računa Voditelja parkirališta
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljen je administrator i postoji nepotvrđena registracija za račun Voditelja parkirališta
- **Opis osnovnog tijeka:**
 1. Administrator otvara listu registraciju na čekanju
 2. Administrator potvrđuje određene registracije
 3. U bazu podataka se spremaju promijene

UC20 -Pregled Korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati registrirane korisnike
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljeni korisnik ima administratorska prava
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregledavanja korisnika
 2. Prikaže se lista svih ispravno registriranih korisnika s osobnim podacima

UC21 -Brisanje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Ukloniti korisnika iz Baze podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljeni korisnik ima administratorska prava
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju uklanjanja korisnika
 2. Administrator pronalazi željenog korisnika
 3. Administrator uklanja željenog korisnika i njegove podatke iz baze podataka

UC22 -Promjena prava pristupa

- **Glavni sudionik:** Administrator
- **Cilj:** Promijeniti prava korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Prijavljeni korisnik ima administratorska prava

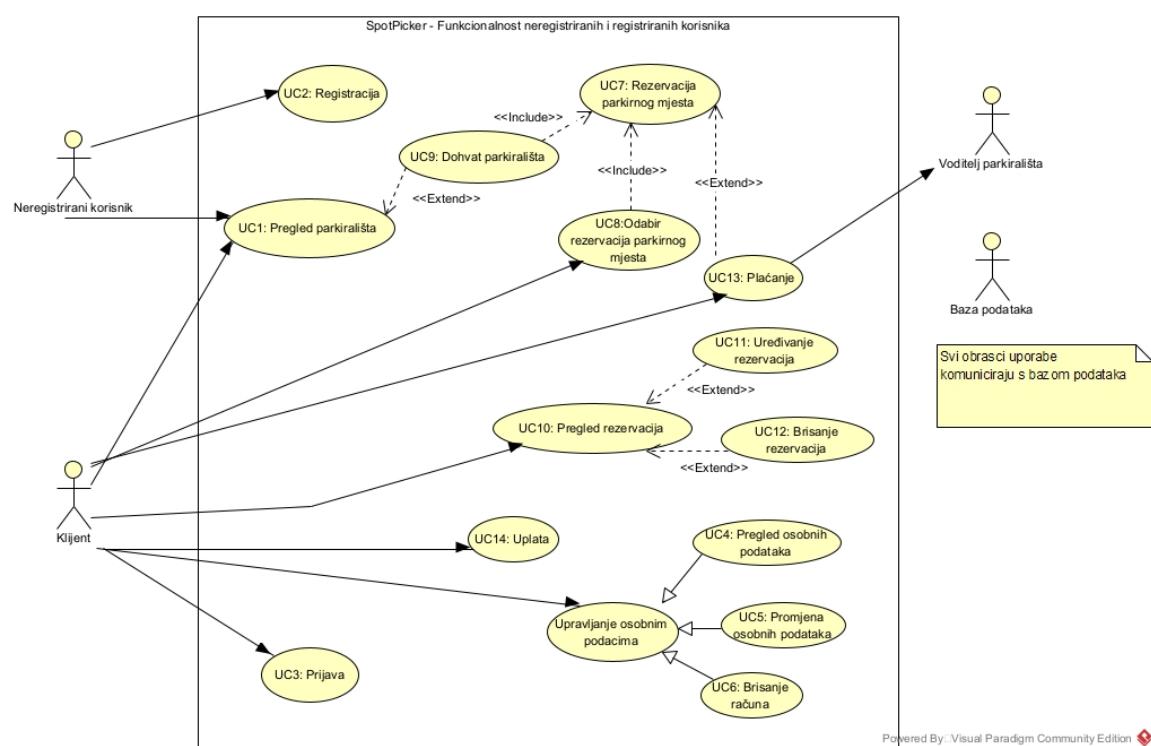
- **Opis osnovnog tijeka:**

1. Administrator pronalazi željenog korisnika
2. Administrator mijenja razinu pristupa željenom korisniku

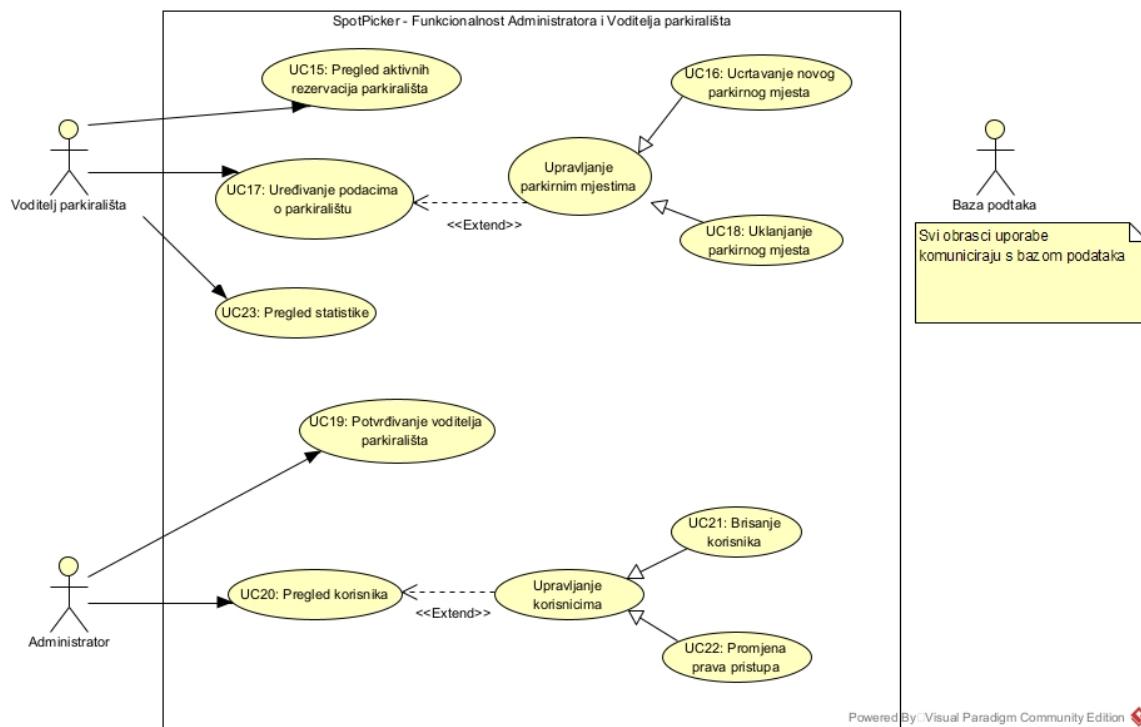
UC23 -Pregled statistike

- **Glavni sudionik:** Voditelj parkirališta
- **Cilj:** Pregledati statistiku popunjenošti parkirališnih mesta
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen kao Voditelj parkirališta
- **Opis osnovnog tijeka:**
 1. Izabere se opcija prikaza statistike za parkiralište
 2. Prikazuje se statistika zauzetosti mesta dotičnog parkirališta

Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnost neregistriranih i registriranih korisnika

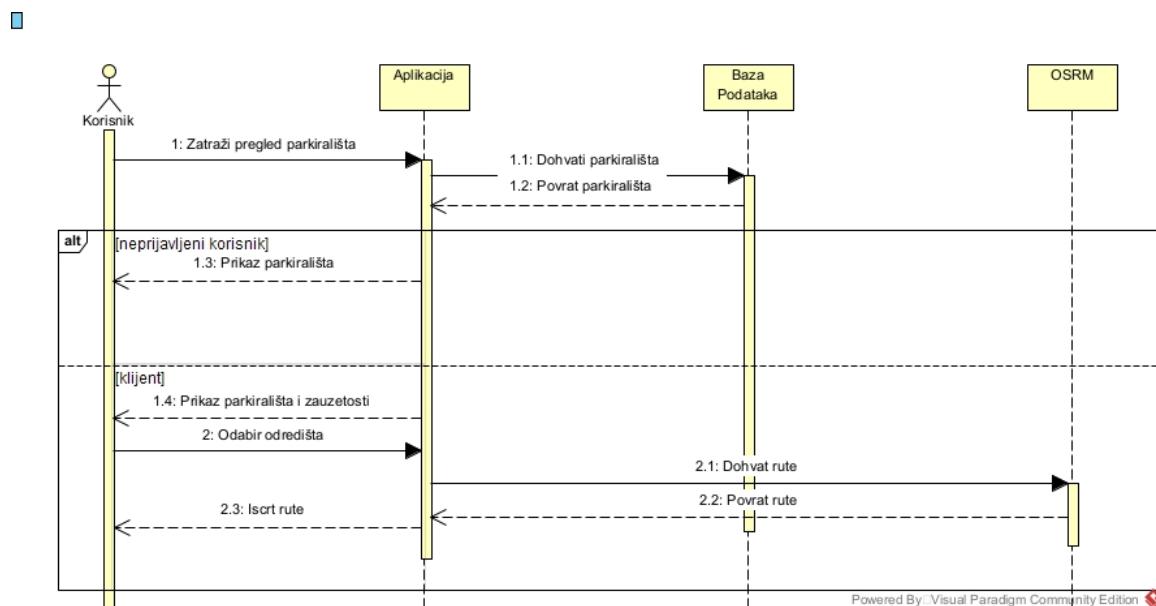


Slika 3.2: Dijagram obrasca uporabe, funkcionalnost administratora i voditelja parkirališta

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC1 - Pregled parkirališta

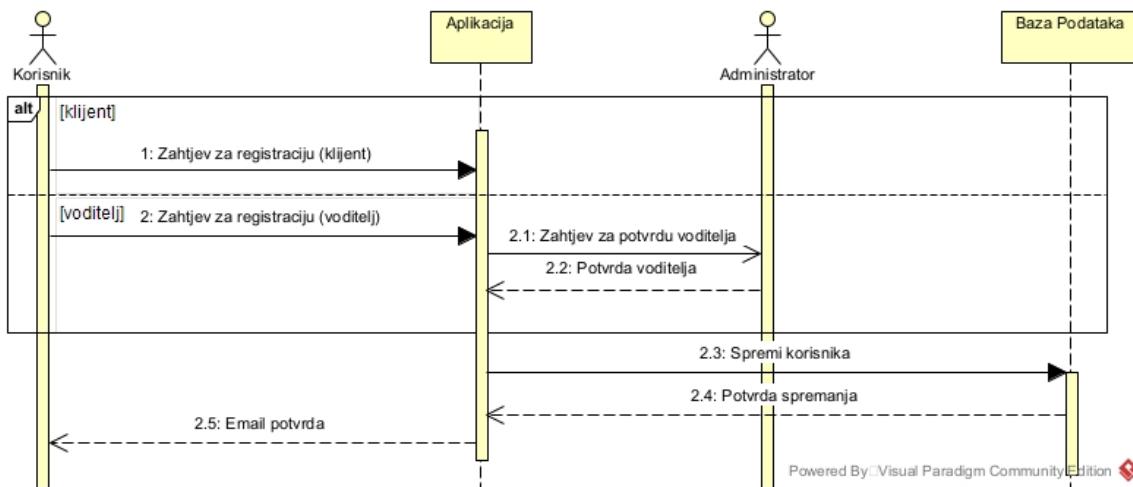
Korisnik šalje zahtjev za pregledom parkirališta kako bi mogao odabrati parkirališno mjesto. Poslužitelj šalje zahtjev prema bazi podataka i dohvaća iz nje parkirališta. Ako je korisnik neprijavljen tada mu poslužitelj samo prikazuje parkirališta. Ako je riječ o prijavljenom korisniku, odnosno klijentu tada mu poslužitelj prikazuje parkirališta s informacijom o zauzetosti. Klijent zatim odabire odredište pri čemu poslužitelj šalje zahtjev prema OSRM (Open Source Routing Machine) za dohvrat rute. OSRM vraća rutu poslužitelju koji ju iscrtava klijentu.



Slika 3.3: Sekvencijski dijagram za UC1

Obrazac uporabe UC2 - Registracija

Korisnik šalje zahtjev za registraciju sa željenom ulogom za koju se prijavljuje prema poslužitelju. Ako je željena uloga voditelj parkirališta, tada poslužitelj prima zahtjev za registraciju od korisnika i šalje prema administratoru zahtjev za potvrdu voditelja. Ako administrator potvrdi korisnika, potvrda se šalje poslužitelju. Ako je željena uloga klijent samo se šalje zahtjev za registraciju prema poslužitelju te nema potvrde administratora. Neovisno o ulozi koju je korisnik izabrao proces registracije se nastavlja. Poslužitelj šalje zahtjev prema bazi podataka za spremanjem korisnika. Baza podataka sprema korisnika i šalje nazad prema poslužitelju potvrdu spremanja. Konačno, poslužitelj šalje email potvrdu korisniku čime proces registracije završava.

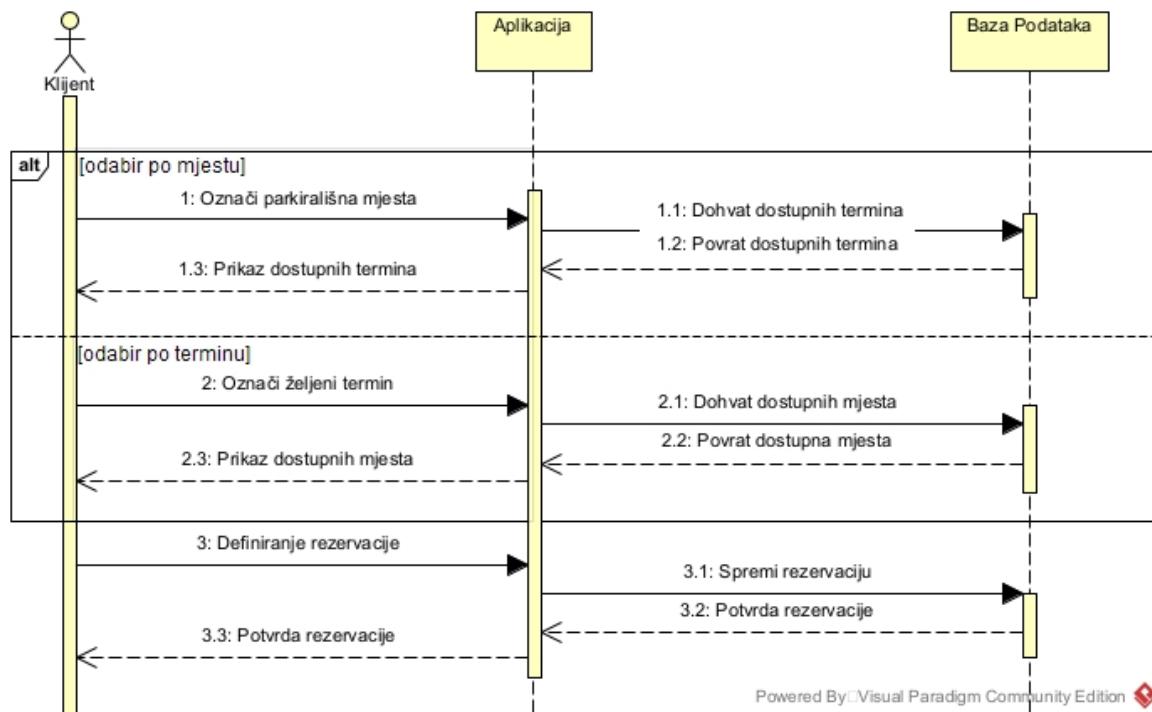


Slika 3.4: Sekvenčni dijagram za UC2

Obrazac uporabe UC7 - Rezervacija parkirališnog mesta

Klijent može rezervirati parkirališna mjesta na dva načina. Prvi način je odbir po mjestu, gdje korisnik označuje željena parkirališna mjesta te šalje zahtjev

prema poslužitelju. Poslužitelj zatim šalje zahtjev prema bazi podataka za dohvata dostupnih termina. Baza podataka vraća dostupne termine poslužitelju nakon čega ih poslužitelj prikazuje klijentu. Drugi način je odabir po terminu, gdje korisnik odabire željeni termin i šalje zahtjev prema poslužitelju. Poslužitelj šalje zahtjev za dohvata dostupnih mesta prema bazi podataka. Baza podataka vraća dostupna mesta poslužitelju koji ih prikazuje klijentu. Nakon odabira načina rezervacije klijent šalje zahtjev za definiranje rezervacije prema poslužitelju. Poslužitelj šalje zahtjev za spremanjem rezervacije prema bazi podataka te baza podataka vraća potvrdu rezervacije. Poslužitelj potvrdu rezervacije prosljeđuje nazad prema klijentu čime rezervacija parkirališnog mesta završava.

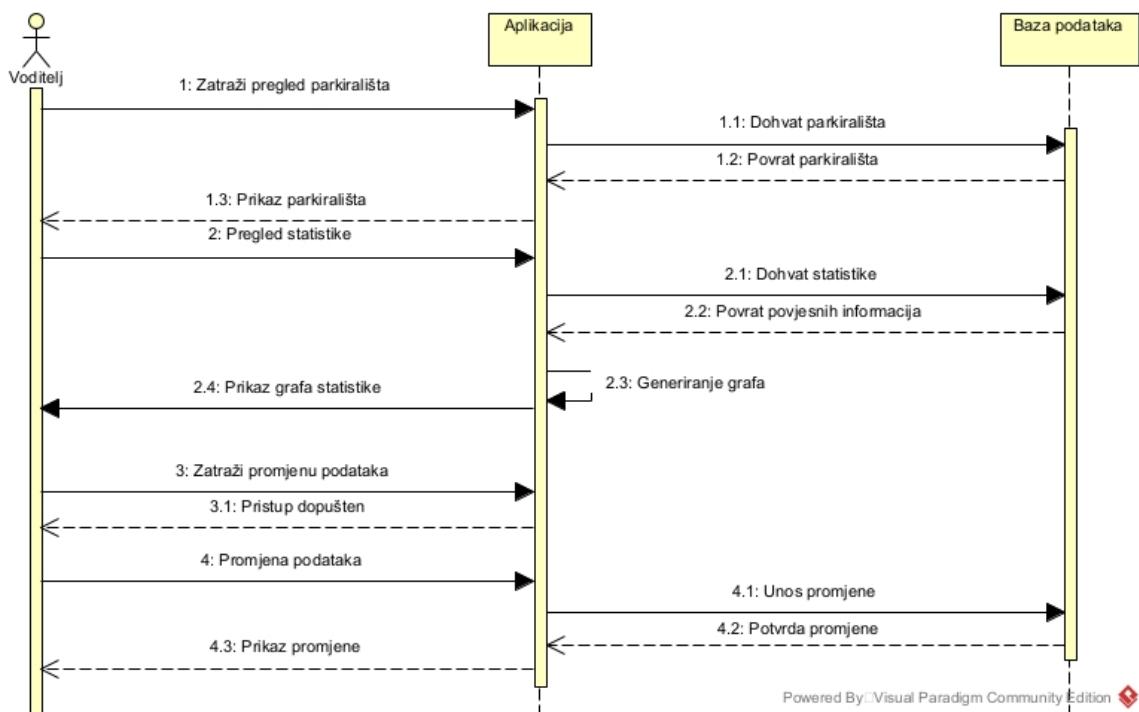


Slika 3.5: Sekvenčni dijagram za UC7

Obrazac uporabe UC17 - Uređivanje podataka o parkiralištu

Vlasnik šalje zahtjev za pregledom parkirališta. Poslužitelj šalje zahtjev za dohvata parkirališta prema bazi podataka. Baza podataka vraća parkirališta poslužitelju

koji ih prikazuje voditelju. Voditelj traži pregled statistike zauzetosti parkirališta i parkirališnih mesta kroz vrijeme. Poslužitelj šalje zahtjev za dohvaćanje statistike prema bazi podataka. Baza podataka vraća povijesne informacije o zauzetosti parkirališta poslužitelju. Poslužitelj na temelju dobivenih povijesnih informacija generira statistiku u obliku grafa. Dobiveni graf statistike prikazuje se vlasniku. Vlasnik šalje zahtjev za promjenu podataka poslužitelju te mu poslužitelj vraća dozvolu za pristup. Vlasnik mijenja podatke i šalje promjene poslužitelju koji promjene proslijedi bazi podataka. Baza podataka potvrđuje promjene i vraća potvrdu poslužitelju koji prikazuje promjene vlasniku.



Slika 3.6: Sekvencijski dijagram za UC17

3.2 Ostali zahtjevi

1. Koristiti OSRM za dohvati rute na mapi
2. Koristiti Overpass API za dohvati početne informacije o parkirališnim mjestima
3. Sustav mora biti implementiran kao web aplikacija koristeći isključivo objektno-orientirane jezike
4. Statistika parkirališta se prikazuje u obliku grafa

4. Arhitektura i dizajn sustava

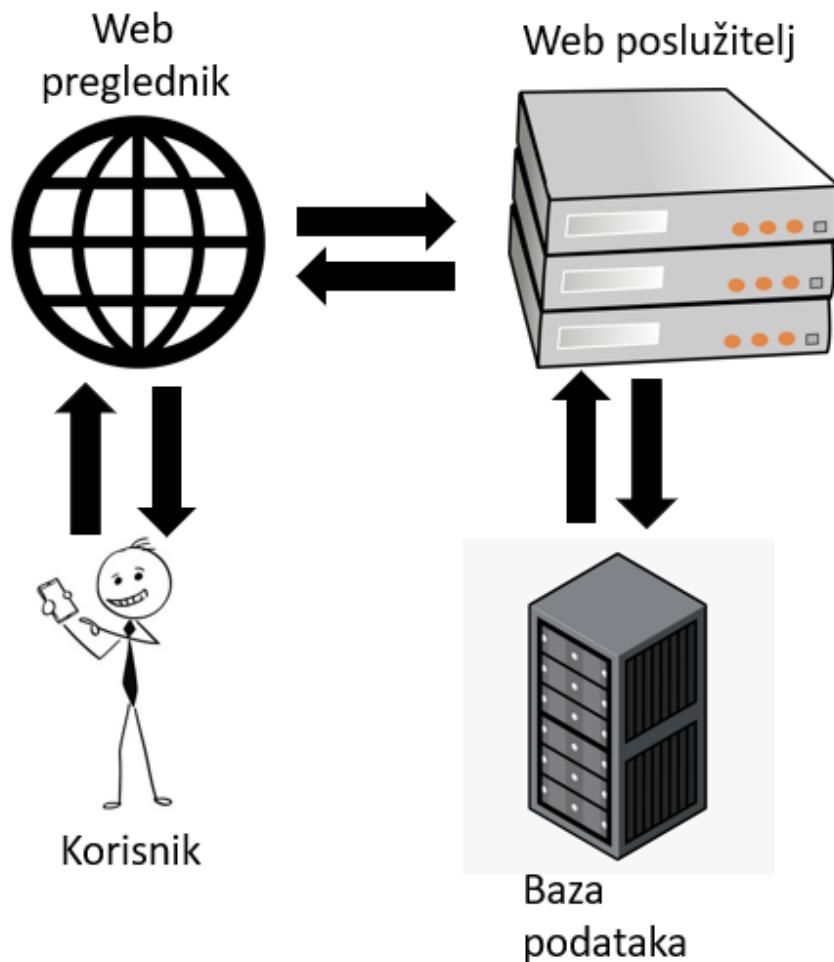
Arhitektura sustava je struktura sustava koja sadrži: elemente programa, njihove izvana vidljive karakteristike te odnose među njima. Arhitektura sustava se dijeli na tri manja sustava:

- Web preglednik
- Web aplikacija
- Baza podataka

Web preglednik je program preko kojega korisnik može pregledavati web-stranice te multimedijalne sadržaje od tih web-stranica. Web-stranica je napisana u kodu koji web preglednik prevodi tako da svatko razumije web-stranicu. Web preglednik također služi kao kanal između korisnika i web poslužitelja.

Web poslužitelj služi za komunikaciju između klijenta i aplikacije. Ta komunikacija se odvija preko HTTP-a. HTTP je protokol koji se koristi za prijenos podataka na webu. Web poslužitelj također pokreće web aplikaciju i prosljeđuje joj zahtjeve.

Web aplikacija obrađuje zahtjeve od korisnika, a za pojedine zahtjeve ona pristupa bazi podataka nakon čega se odgovor šalje korisniku putem web poslužitelja te web preglednika u HTML obliku.



Slika 4.1: Arhitektura sustava

U našemu sustavu za razvoj radnog okvira na poslužiteljskoj strani (backend-u) odlučili smo koristiti *Spring Boot*, a na klijentskoj strani (frontend-u) smo odlučili koristiti *React*. Programske jezike za koje smo se odlučili su *Java* i *JavaScript*.

Korišteni razvojni okvir Spring se služi tzv. MVC arhitekturom. Takva arhitektura podrazumijeva sljedeću podjelu:

- **Model:** Središnja komponenta sustava, dohvata te manipulacija podataka
- **View:** dostupni razni prikazi podataka (tablično, grafom)
- **Controller:** upravlja zahtjevima korisnika te na temelju njih izvodi daljnju interakciju s ostalim komponentama

Naša web aplikacija koristi višeslojni stil arhitekture. Prednosti višeslojnog stila arhitekture su brojne:

- olakšava razvoj programa
- timovi se mogu raspodijeliti na razvoj zasebnih slojeva arhitekture
- podjela briga odnosno svaki sloj se mora brinuti za samo svoju zadaću
- moguće je jednostavno povećanje i poboljšanje sustava

4.1 Baza podataka

4.1.1 Vrsta i implementacija

Za modeliranje sustava je korištena relacijska baza podataka, a nju smo implementirati pomoću open-source sustava za upravljanje bazama podataka PostgreSQL. Za izradu dijagrama ER i generiranje relacijske sheme je korišten besplatan online alat ERDplus (<https://erdplus.com/>) koji je korišten i u sklopu predmeta Baze podataka.

4.1.2 Glavne komponente baze podataka

Baza podataka sastoji se od sljedećih entiteta:

- **User** (*id, email username, password, name, lastName, IBAN, photo, role, walletBalance, approved, verified, verificationToken*)
- **Parking** (*picture, parkingName, parkingDescription, costPerHour, parkingId, voditeljId*)
- **ParkingSpot** (*parkingSpotId, latitude, longitude, polygon, label, reserveable, parkingId*)
- **Voditelj** (*voditeljId*)
- **Reservation** (*reservationId, duration, reservationStart, reservationEnd, korisnikId, parkingSpotId*)
- **BicycleParking** (*bicycleId, latitude, longitude, numberOfSpots, polygon, parkingId*)

4.1.3 Opis tablica

User: sadržava sve bitne informacije o registriranim korisnicima u sustavu. Sadrži atribute: id, email, username, password, name, lastName, IBAN, photo, role, walletBalance, approved, verified, verificationToken. Također sadrži dodatne atribute putem kojih pratimo je li korisnik verificirao e-mail (verified) te, u slučaju role=voditelj, je li potvrđen od strane administratora (approved). Ovaj entitet je preko korisničkog identifikatora u One-to-Many vezi s entitetom Rezervacija i u One-to-One vezi s entitetom Voditelj.

User		
id	INT	Jedinstveno korisničko ime
email	VARCHAR	Email korisnika
username	VARCHAR	Korisničko ime
password	VARCHAR	Hash lozinke dobiven s Bcrypt encoderom
name	VARCHAR	Ime korisnika
lastName	VARCHAR	Prezime korisnika
IBAN	VARCHAR	IBAN korisnika
photo	BYTEA	Byte array slike osobne iskaznice
role	INT	Enum koji označava ulogu korisnika ('klijent', 'voditelj', 'administrator')
walletBalance	NUMERIC	Decimalni broj koji označava koliko novaca se nalazi u novčaniku
approved	BOOLEAN	Označava je li voditelj potvrđen od strane admina.
verified	BOOLEAN	Je li korisnik verificiran preko emaila?
verificationToken	BOOLEAN	Token verifikacije

Parking: Parking: sadrži ključne informacije o parkiralištima. Sadrži atribute koje postavlja voditelj parkirališta: fotografija, naziv, opis i cijena po satu. Uz to sadrži i generiran ID parkirališta. Ovaj entitet je u vezi Many-to-One s entitetom voditelj putem ID-a voditelja i u One-to-Many vezi s ParkingSpotom preko ID-a parkirališta.

Parking		
parkingId	INT	Jedinstveni identifikator parkirališta
picture	BYTEA	Slika parkirališta koju voditelj može priložiti
parkingName	VARCHAR	Naziv parkirališta
parkingDescription	VARCHAR	Opis parkirališta
costPerHour	NUMERIC	Cijena po satu koju definira voditelj parkirališta
voditeljId	INT	Jedinstveni identifikator voditelja parkirališta

ParkingSpot: ParkingSpot sadrži informacije o pojedinačnim parkirnim mjestima unutar parkirališta. U ovu tablicu ćemo unijeti početne informacije o mjestima za automobile koje smo dobili od Overpass API-ja. Ti atributi su id, centralne koordinate parkirališta (dužina, širina) i poligon. Također sadrži oznaku parkirališnog mjesta, informaciju o dostupnosti i mogućnosti rezervacije mjesta koje postavlja voditelj, te ID parkirališta kojemu pripada. U vezi je Many-to-One s entitetom "Parking" putem ID-a parkirališta i One-to-Many s entitetom Rezervacija putem identifikacije parkirališnog mjesta.

ParkingSpot		
parkingSpotId	VARCHAR	Identifikacija mjesta koje generira overpassAPI koji generira overpassAPI. Npr. "node/11310562209"
label	VARCHAR	Oznaka ili broj parkirališnog mjesta
longitude	NUMERICAL	Označuje duljinu
latitude	NUMERICAL	Označuje širinu
reserveable	BOOLEAN	Je li mjesto slobodno ili nije
polygon	VARCHAR	Poligon parkirališnog mjesta
parkingId	INT	Jedinstveni identifikator parkirališta (Parking.parkingId)

Manager: sadrži informacije o voditeljima parkirališta. Povezuje voditelja sa parkiralištem. Ima vezu One-to-One s entitetom Korisnik i One-to-Many s Parkingom.

Manager		
voditeljId	INT	Jedinstveni identifikator voditelja parkirališta

Reservation: sadrži sve rezervacije koje su napravili korisnici. Ovaj entitet ima vezu Many-to-One s entitetom Korisnik putem korisničkog identifikatora. Također, ima vezu Many-to-One s entitetom "ParkingSpot" putem lokacije parkirnog mesta.

Reservation		
reservationId	INT	Jedinstven identifikator rezervacije
duration	INT	Trajanje rezervacije
reservationStart	TIMESTAMP	Početak rezervacije
reservationEnd	TIMESTAMP	Kraj rezervacije
korisnikId	INT	Jedinstveno korisničko ime
parkingSpotId	INT	Jedinstven identifikator parkirališnog mesta

BicycleParking: sadrži atribute koje dobivamo pozivom overpassAPI-a: (id, koordinate, broj dostupnih mesta) te identifikator parkirališta kojem pripada. U Many-to-One vezi je s Parkingom preko identifikatora parkirališta.

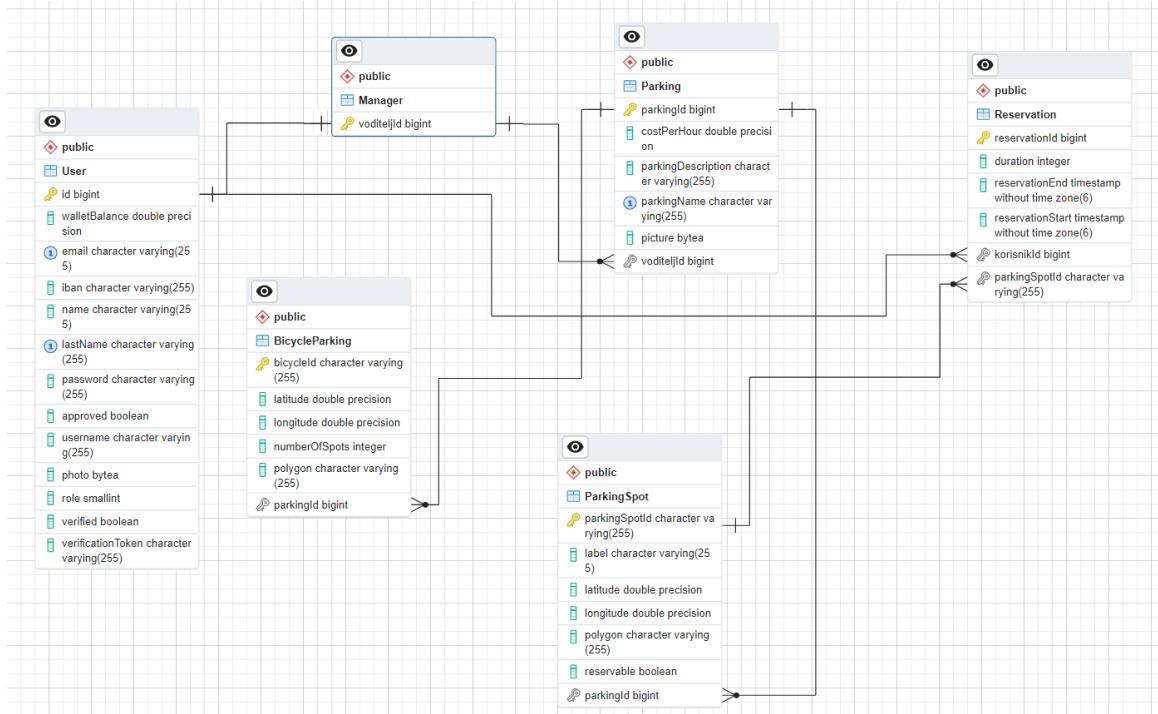
BicycleParking		
bicycleId	INT	Jedinstven identifikator parkirališnog mesta za bicikle
numberOfSpots	INT	Broj dostupnih mesta na parkingu za bicikle
longitude	NUMERICAL	Označuje duljinu
latitude	NUMERICAL	Označuje širinu

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

BicycleParking		
polygon	VARCHAR	Poligon parkirališnog mesta
parkingId	INT	Jedinstven identifikator parkirališta

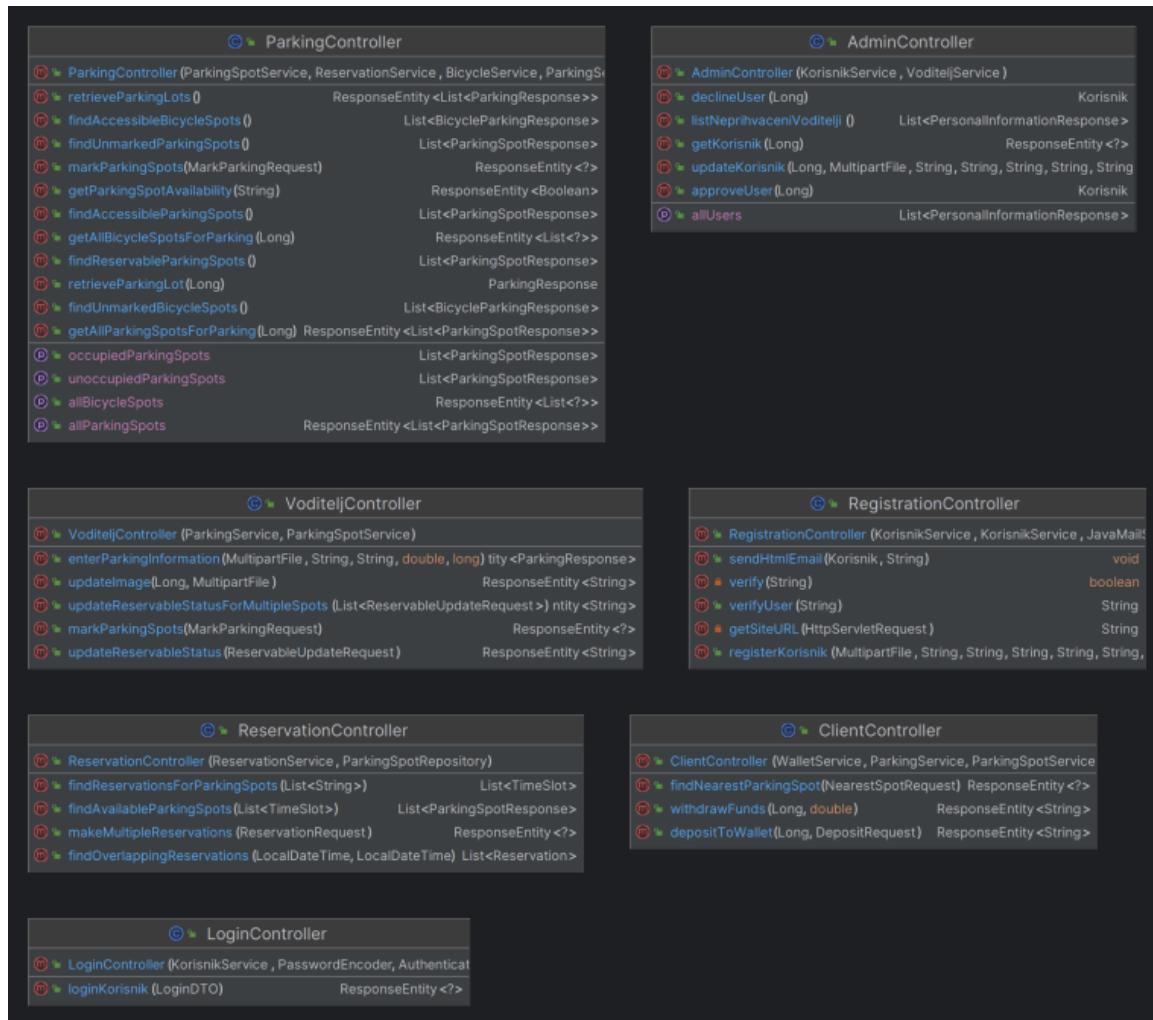
4.1.4 Dijagram baze podataka



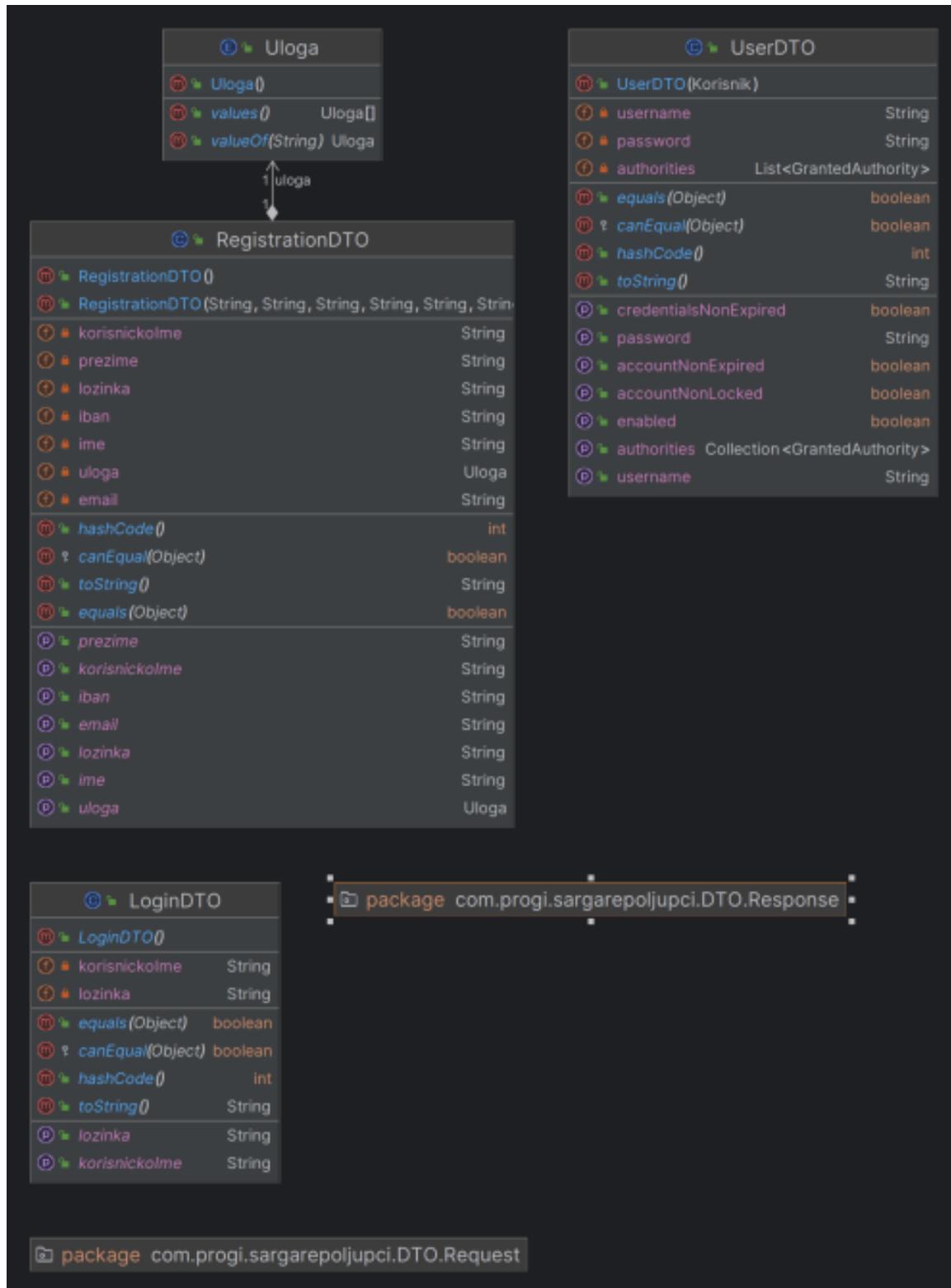
Slika 4.2: E-R dijagram baze podataka

4.2 Dijagram razreda

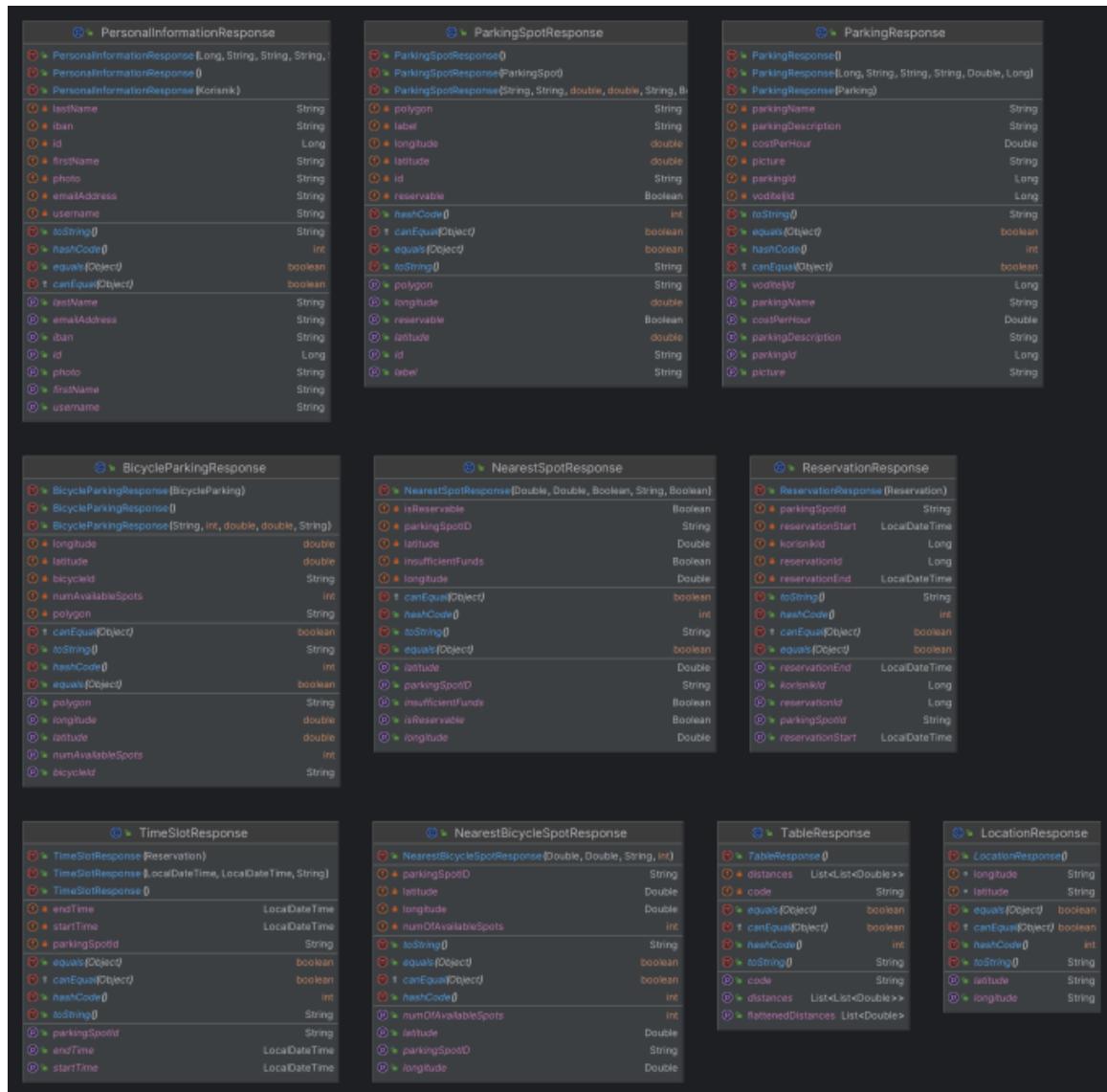
Na slikama 4.3, 4.4, 4.5, 4.6, 4.7, i 4.8 su prikazani razredi koji pripadaju backend dijelu MVC arhitekture. Servisi obično predstavljaju logiku poslovanja (business logic) vaše aplikacije. To su komponente koje obavljaju specifične zadatke ili operacije, a ne direktno vezane za obradu HTTP zahtjeva. Na primjer, servis za korisnike može sadržavati logiku za registraciju, prijavu, promjenu lozinke i slično. Servisi se često koriste kako bi se izdvojila poslovna logika iz kontrolera i omogućila ponovna upotreba koda. Modeli predstavljaju podatke vaše aplikacije. Ovi podaci se često mapiraju na bazu podataka. Na primjer, ako imate entitet "Korisnik", model bi predstavljao strukturu podataka koja opisuje korisnika, uključujući atributе kao što su ime, prezime, e-mail adresa, itd. DTO-ovi su objekti koji se koriste za prijenos podataka između različitih slojeva aplikacije. Na primjer, kada želite prenijeti podatke iz kontrolera do servisa ili obrnuto, koristiti ćete DTO-ove kako biste definirali strukturu podataka koja će se prenositi. Ovo može pomoći u smanjenju količine podataka koja se prenosi, poboljšava čitljivost koda i omogućava bolju kontrolu nad tim koji se podaci šalju. Kontroleri su odgovorni za obradu HTTP zahtjeva i komunikaciju s korisničkim sučeljem (frontend). Kontroleri često koriste servise kako bi dobili podatke i obavili određene akcije.



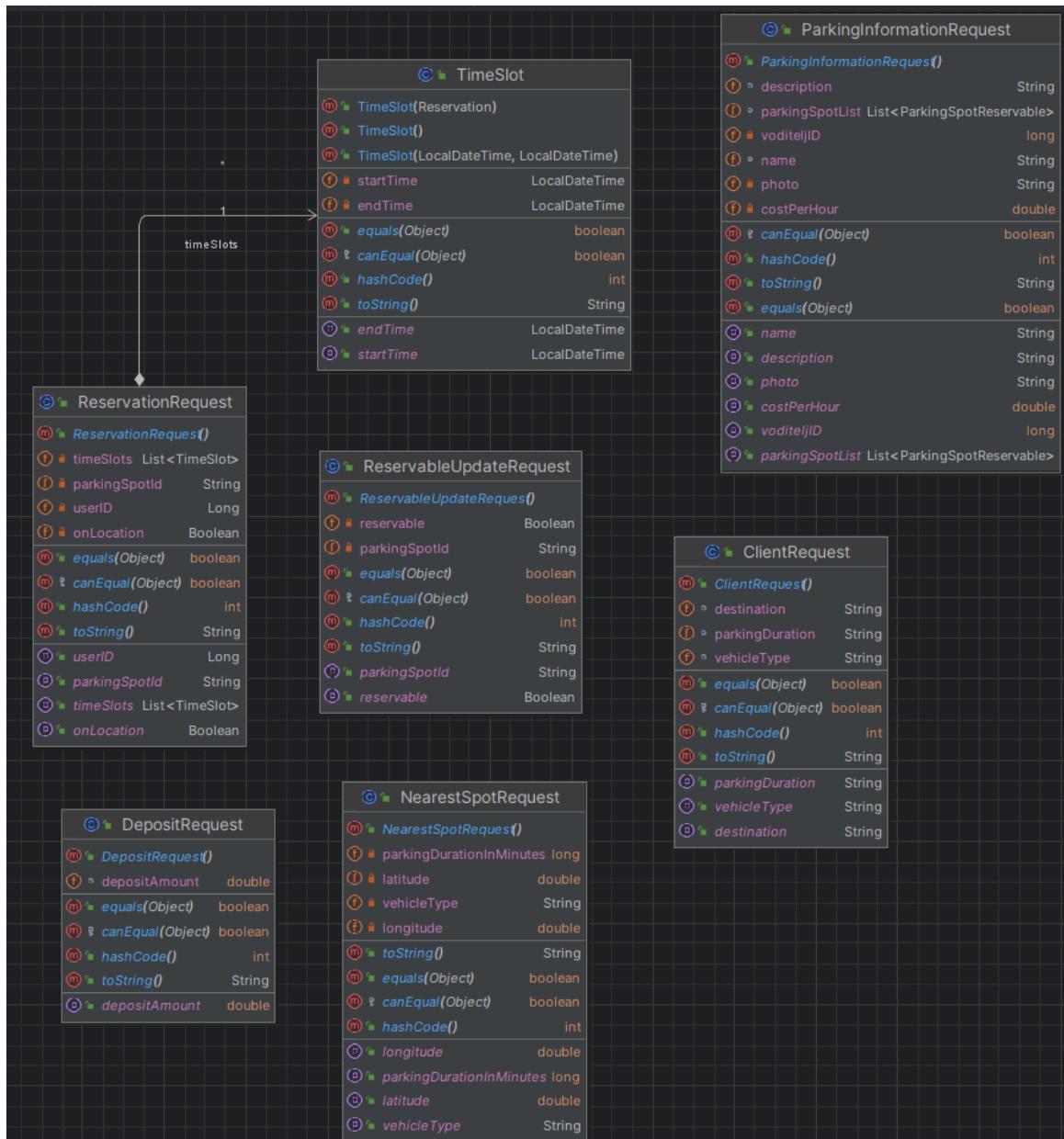
Slika 4.3: Dijagram razreda - Controllers



Slika 4.4: Dijagram razreda - Data transfer objects



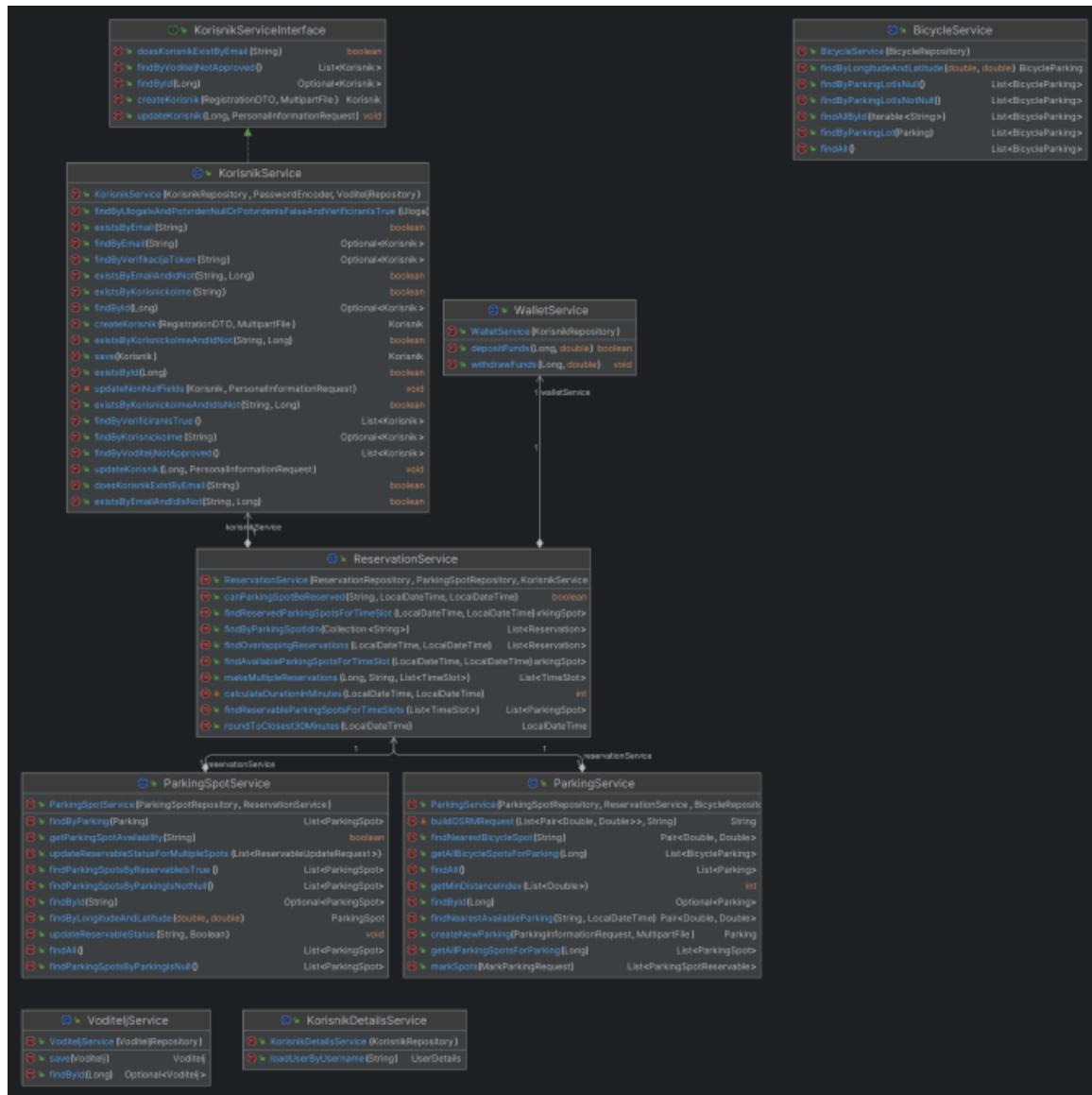
Slika 4.5: Dijagram razreda - Response



Slika 4.6: Dijagram razreda - Request



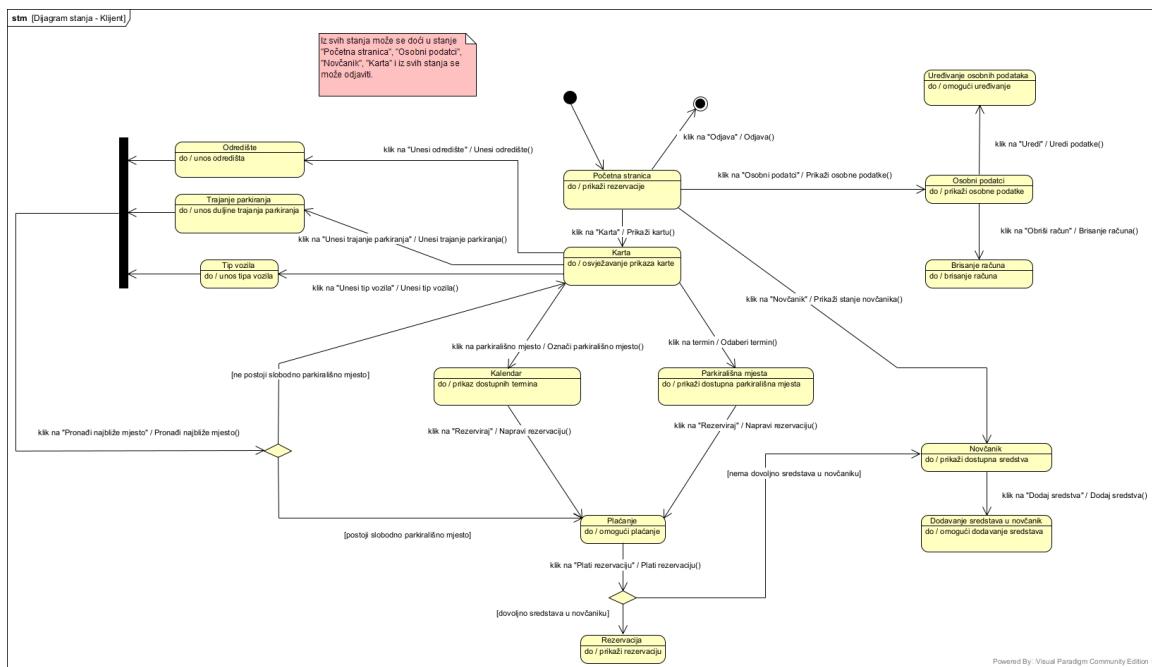
Slika 4.7: Dijagram razreda - Models



Slika 4.8: Dijagram razreda - Service

4.3 Dijagram stanja

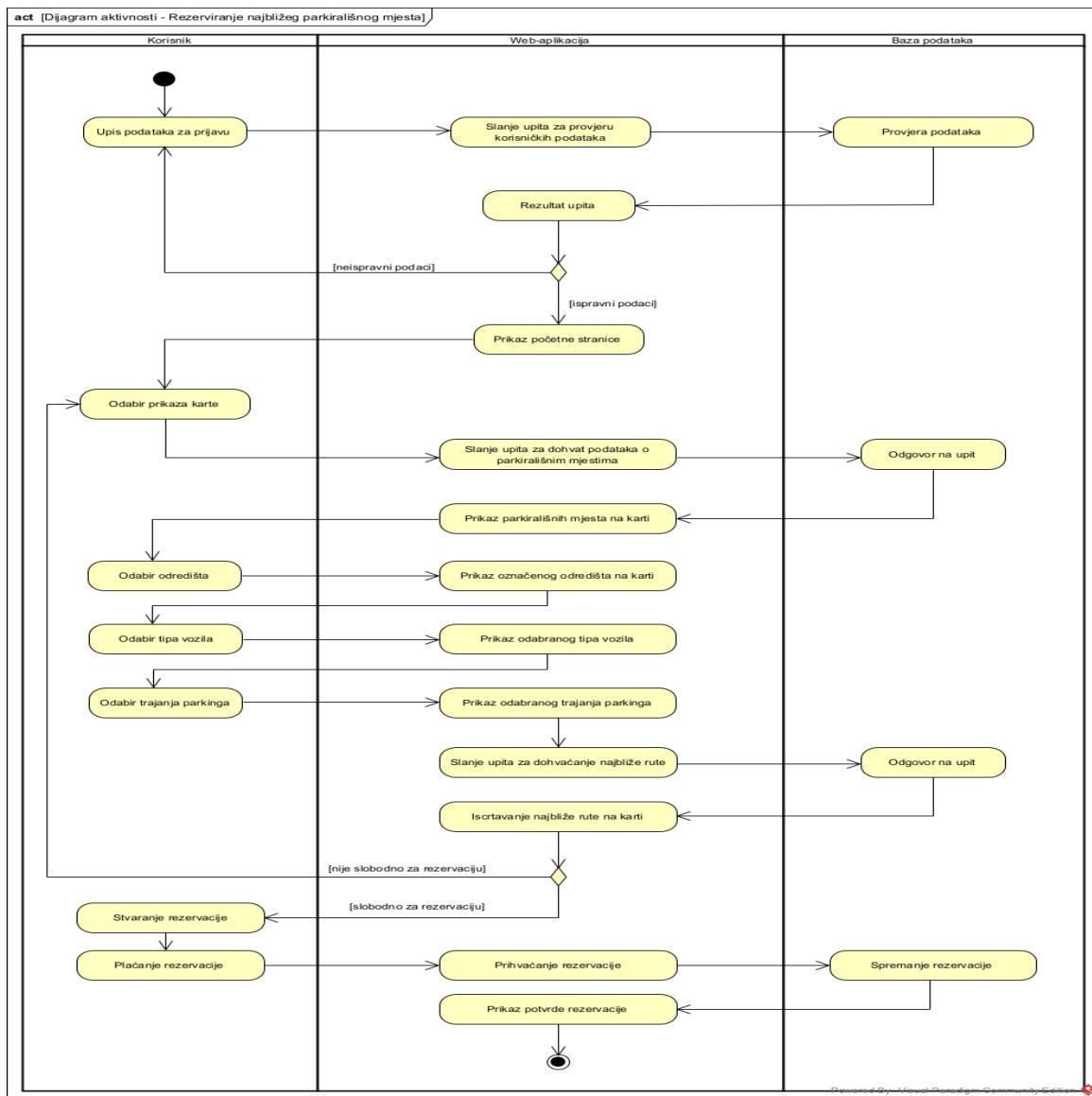
UML-dijagram stanja (engl. state machine diagram) je ponašajni UML-dijagram kojim se prikazuje diskretno ponašanje objekta ili sustava putem prelazaka između konačnog broja stanja. Ova vrsta dijagrama se koristi za modeliranje ponašanja entiteta tijekom vremena, naglašavajući odgovor na događaje i okidače. Na slici 4.6 prikazan je dijagram stanja za klijenta, odnosno registriranog korisnika. Nakon prijave, korisniku se prikazuje početna stranica na kojoj može vidjeti vlastite rezervacije. Klijent može pregledavati kartu u stvarnom vremenu te klikom označiti parkirališna mjesta koja ga zanimaju te zatim vidjeti prikaz dostupnih termina za ta parkirališna mjesta. Za klijenta postoji i opcija da prvo odabere termin klikom na odgovarajući termin te zatim bira željena parkirališna mjesta. U slučaju da želi pronaći najbliže parkirališno mjesto, klijent može unijeti odredište, tip vozila i procijenjeno vrijeme trajanja parkinga te će mu aplikacija na karti iscrtati rutu do najbližeg parkirališnog mjeseta. Nakon uspješnog plaćanja klijent može vidjeti svoju rezervaciju te ostale rezervacije na početnoj stranici. Klijent u svakom trenutku ima mogućnost pregleda stanja novčanika klikom na "Novčanik" te može po potrebi dodavati sredstva. Klikom na "Osobni podaci" klijent može pregledavati i uređivati osobne podatke ili odabrati opciju brisanja računa.



Slika 4.9: Dijagram stanja

4.4 Dijagram aktivnosti

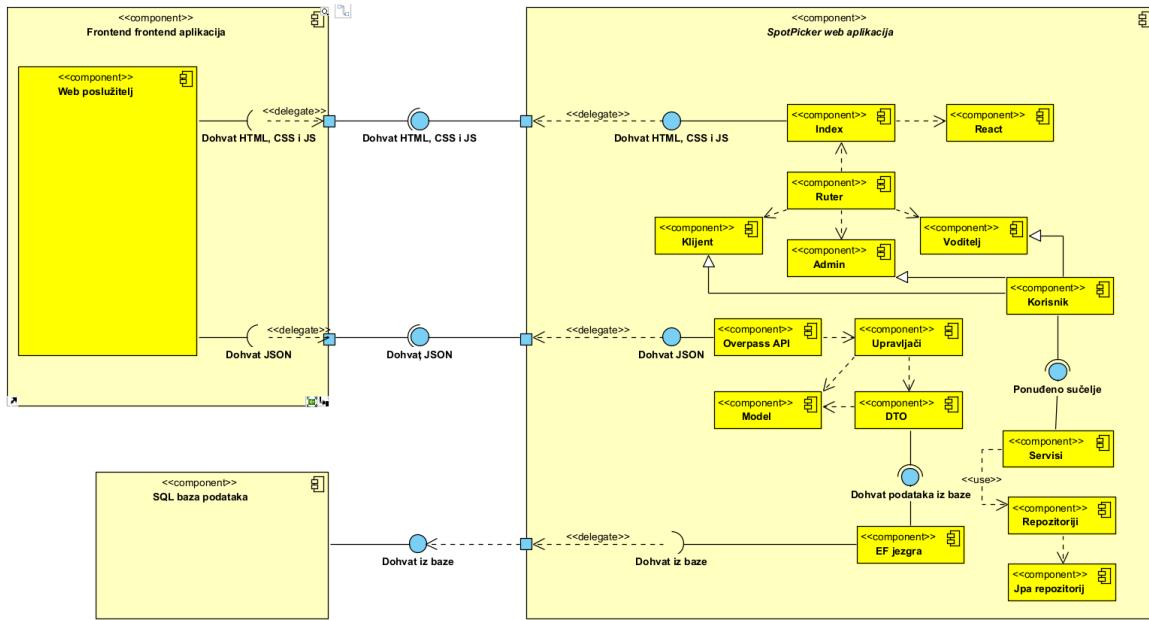
UML-dijagrami aktivnosti (engl. activity diagrams) su ponašajni UML-dijagrami koji se u programskom inženjerstvu upotrebljavaju za modeliranje i grafički prikaz dinamičkog ponašanja sustava. Na njima je prikazano izvođenje aktivnosti kroz niz akcija koje čine upravljačke tokove i tokove objekata. Pokretanje neke akcije uvjetovano je završetkom jedne ili više prethodnih akcija ili dostupnošću objekata i podataka. Registrirani korisnik se prijavljuje u sustav te otvara kartu koja mu prikazuje parkirališna mjesta te informacije o njihovoj zauzetosti. Klijent odabire odredište, tip vozila i trajanje parkinga nakon čega mu aplikacija na karti iscrtava rutu do najbližeg parkirališnog mesta. Ako je zadovoljan mjestom, klijent stvara i plaća rezervaciju čime ona postaje aktivnom.



Slika 4.10: Dijagram aktivnosti

4.5 Dijagram komponenti

UML-dijagrami komponenti (engl. component diagrams) su vrsta strukturnih UML-dijagrama koji prikazuju organizaciju i odnose komponenti koje čine programsku potporu. Pružaju vizualni prikaz arhitekture sustava, naglašavajući modularnu strukturu i interakcije između komponenti. Sustavu se pristupa preko dva različita sučelja. Preko sučelja za dohvrat HTML, CSS i JS datoteka poslužuju se datoteke koje pripadaju *frontend* dijelu aplikacije. Komponenta *Ruter* na upit s url određuje koja datoteka će se poslužiti na sučelje, odnosno prikazati korisniku. *Frontend* dio se sastoji od niza JavaScript datoteka koje su raspoređene u logičke cjeline nazvane po tipovima dijelova aplikacije koje prikazuju, odnosno po aktorima koji tim dijelovima aplikacije pristupaju. Sve JavaScript datoteke ovise o React biblioteci iz koje dohvaćaju gotove komponente kao što su gumbi, forme, stil, obavijesti, tablice, itd. Preko sučelja za dohvrat JSON podataka pristupa se Overpass API komponenti. Overpass API poslužuje podatke koji pripadaju backend dijelu aplikacije. EF Jezgra zadužena je za dohvaćanje tablica iz baze podataka pomoću SQL upita. Podaci koji su pristigli iz baze se šalju dalje MVC arhitekturi u obliku DTO (Data transfer object). Web poslužitelj komponenta preko dostupnih sučelja komunicira sa SpotPicker web aplikacijom te ovisno o korisnikovim akcijama osvježava prikaz i dohvaća nove podatke ili datoteke.



Slika 4.11: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu odvijala se putem aplikacija WhatsApp¹ i Discord². Kao sustav za upravljanje izvornim kodom korišten je Git³, distribuirani sustav kontrole verzija koji prati promjene u bilo kojem skupu datoteka. Udaljeni rezervorij projekta je dostupan na web platformi GitLab⁴.

UML dijagrami su napravljeni u Visual Paradigm Online⁵, besplatnom alatu za crtanje.

Pri oblikovanju aplikacije za *backend* je korišten radni okvir Spring Boot⁶ i jezik Java⁷. Spring Boot je otvoreni Java radni okvir namijenjen pojednostavljenoj izgradnji samostalnih, produkcijski orijentiranih Spring aplikacija. Pruža optimiziranu platformu za razvoj Java aplikacija s naglaskom na konvenciju umjesto konfiguracije, omogućavajući programerima brzo postavljanje i razvoj robusnih, skalabilnih i efikasnih aplikacija. Spring Boot dolazi s ugrađenim značajkama poput automatske konfiguracije, podrške za ugrađeni web poslužitelj i postavkama spremnim za produkcijsko okruženje. Java je objektno orijentirani i platformski neovisan programski jezik. Kod napisan u Javi može izvoditi na bilo kojem uređaju ili platformi s Java Virtual Machine (JVM). Java je poznata po svojoj prenosivosti, snažnoj podršci zajednice i obilnim bibliotekama, što je čini popularnim izborom za izgradnju različitih vrsta aplikacija.

Za implementaciju *frontenda* koristili smo razvojno okruženje React⁸ i jezik JavaScript⁹. React je otvorena JavaScript biblioteka za izgradnju korisničkih sučelja. Omogućuje stvaranje ponovno upotrebljivih komponenti korisničkog sučelja, upravljanje njihovim vlastitim stanjem i učinkovito ažuriranje korisničkog sučelja putem

¹<https://www.whatsapp.com/>

²<https://discord.com/>

³<https://git-scm.com/>

⁴<https://about.gitlab.com/>

⁵<https://www.visual-paradigm.com/>

⁶<https://spring.io/projects/spring-boot/>

⁷<https://www.java.com/en/>

⁸<https://react.dev/>

⁹<https://www.javascript.com/>

virtualnog DOM-a. React slijedi jednosmjerni tok podataka, koristi JSX za sintaksu komponenata i široko se koristi zbog modularnosti i optimizacija performansi. JavaScript je svestrani i objektno orijentirani jezik, uglavnom korišten za stvaranje dinamičnog i interaktivnog sadržaja na web stranicama. Podržava asinkrono programiranje i radi na različitim platformama, što ga čini ključnom tehnologijom u web razvoju. Osim u preglednicima, proširuje se i na druge domene, uključujući i izradu mobilnih aplikacija.

LaTeX¹⁰ je sustav za stvaranje visokokvalitetnih dokumenata, posebno u znanstvenim i akademskim kontekstima. Koristi označavanje teksta pomoću naredbi kako bi se definirala struktura dokumenta i formatiranje. Korisnici pišu dokumente u običnom tekstu s ekstenzijom .tex i koriste LaTeX compiler za generiranje formatiranih izlaza, često u PDF formatu.

¹⁰<https://www.latex-project.org/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Ispitivanje komponenti provjerava ispravnost djelovanja pojedinih dijelova programa koji su mogući za odvojeno ispitivanje, poput pojedinačnih funkcija ili metoda unutar objekta. U ovom projektu koristimo JUnit i Mockito za implementaciju i izvođenje automatskih testova. JUnit je okvir za testiranje koji omogućuje definiranje, organiziranje i izvođenje testova kako bismo provjerili ispravnost funkcionalnosti naše aplikacije. Mockito je biblioteka koja nam omogućuje stvaranje lažnih objekata kako bismo simulirati njihovu funkcionalnost, omogućujući izolaciju jedinica koda i neovisno testiranje. Pomoću Mockito-a smo stvorili mock objekte za servise i rezervorije, što nam omogućuje precizno kontroliranje ponašanja tih objekata tijekom testiranja.

```
5 usages
@Mock
private ParkingRepository parkingRepository;

7 usages
@Mock
private ParkingSpotRepository parkingSpotRepository;

3 usages
@InjectMocks
private ParkingService parkingService;
```

Slika 5.1: Stvaranje mock objekata

Ispitivanje komponenti za CreateKorisnikTest fokusira se na verifikaciju funkcionalnosti klase KorisnikService prilikom stvaranja novog korisnika. Testiraju se različiti scenariji, uključujući prepoznavanje već postojećeg emaila ili korisničkog imena te obrada null vrijednosti slike.

```

    @Test
    public void UsernameAlreadyExists() {
        registrationDTO = new RegistrationDTO();
        registrationDTO.setKorisnickoIme("john_doe");
        registrationDTO.setLozinka("password123");
        registrationDTO.setEmail("john.doe@example.com");
        registrationDTO.setIban("HR1234567890123456789");
        registrationDTO.setIme("John");
        registrationDTO.setPrezime("Doe");
        registrationDTO.setUloga(Uloga.KLIENT);
        photo = new MockMultipartFile( name: "photo", originalFilename: "test.jpg", contentType: "image/jpeg", "test image".getBytes());

        when(userRepository.existsByEmail(anyString())).thenReturn( false );
        when(userRepository.existsByKorisnickoIme(anyString())).thenReturn( true );

        RequestDeniedException exception = assertThrows(RequestDeniedException.class,
            () -> korisnikService.createKorisnik(registrationDTO, photo));

        verify(userRepository, times( wantedNumberOfInvocations: 1 )).existsByEmail(anyString());
        verify(userRepository, times( wantedNumberOfInvocations: 1 )).existsByKorisnickoIme(anyString());

        assertEquals( message: "Korisnik s tim usernameom vec postoji u bazi podataka", exception.getMessage(),
            actual: "Korisnik s tim usernameom vec postoji u bazi podataka" );
    }
}

```

Slika 5.2: 1. ispitni slučaj za CreateKorisnikTest

```

    @Test
    public void createKorisnikEmptyPhoto() {
        registrationDTO = new RegistrationDTO();
        registrationDTO.setKorisnickoIme("john_doe");
        registrationDTO.setLozinka("password123");
        registrationDTO.setEmail("john.doe@example.com");
        registrationDTO.setIban("HR1234567890123456789");
        registrationDTO.setIme("John");
        registrationDTO.setPrezime("Doe");
        registrationDTO.setUloga(Uloga.KLIENT);
        photo = new MockMultipartFile( name: "photo", originalFilename: "test.jpg", contentType: "image/jpeg", "test image".getBytes());
        MockMultipartFile photoNull = new MockMultipartFile( name: "photo", (byte[] ) null);
        MockMultipartFile photoEmpty = new MockMultipartFile( name: "photo", new byte[0]);
        RequestDeniedException exception = assertThrows(RequestDeniedException.class,
            () -> korisnikService.createKorisnik(registrationDTO, photoEmpty));

        assertEquals("Picture field can't be empty", exception.getMessage(),
            ("Picture field can't be empty"));
        verify(userRepository, never()).save(any());
    }
}

```

Slika 5.3: 2. ispitni slučaj za CreateKorisnikTest

```
± piliip
@Test
public void EmailAlreadyExists(){
    registrationDTO = new RegistrationDTO();
    registrationDTO.setKorisnickoIme("john_doe");
    registrationDTO.setLozinka("password123");
    registrationDTO.setEmail("john.doe@example.com");
    registrationDTO.setIban("HR1234567890123456789");
    registrationDTO.setIme("John");
    registrationDTO.setPrezime("Doe");
    registrationDTO.setUloga(Uloga.KLIENT);
    photo = new MockMultipartFile( name: "photo", originalFilename: "test.jpg", contentType: "image/jpeg", "test image".getBytes());

    when(userRepository.existsByEmail(anyString())).thenReturn( t true);

    RequestDeniedException exception = assertThrows(RequestDeniedException.class,
        () -> korisnikService.createKorisnik(registrationDTO, photo));

    verify(userRepository, times( wantedNumberOfInvocations: 1)).existsByEmail(anyString());

    assertEquals( message: "Korisnik s tim emailom vec postoji u bazi podataka", exception.getMessage(),
        actual: "Korisnik s tim emailom vec postoji u bazi podataka");
}
```

Slika 5.4: 3. ispitni slučaj za CreateKorisnikTest

Ispitivanje komponenti za UpdateKorisnikTest usmjeren je na verifikaciju funkcionalnosti metode updateKorisnik unutar klase KorisnikService. Testiraju se scenariji ažuriranja korisničkih podataka, uključujući promjenu korisničkog imena, imena, lozinke i slike.

```

@ Phillip *
@Test
public void testUpdateKorisnik() throws SQLException, IOException {
    Long userId = 1L;
    PersonalInformationRequest userRequest = new PersonalInformationRequest();
    userRequest.setUsername("newUsername");
    userRequest.setFirstName("newFirstName");
    userRequest.setPhoto(new MockMultipartFile( name: "test.jpg", originalFilename: "test.jpg", contentType: "image/jpeg", "test".getBytes()));
    userRequest.setPassword("password123");

    Korisnik existingUser = new Korisnik();
    existingUser.setId(userId);
    existingUser.setKorisnickoIme("staraKorisnicko");
    existingUser.setIme("stanoIme");
    userRequest.setPassword("stara");

    when(userRepository.findById(userId)).thenReturn(Optional.of(existingUser));
    when(passwordEncoder.encode(anyString())).thenReturn( t: "encodedPassword");

    korisnikService.updateKorisnik(userId, userRequest);

    verify(userRepository).save(argThat(updatedUser ->
        updatedUser.getId().equals(userId) &&
        updatedUser.getKorisnickoIme().equals(userRequest.getUsername()) &&
        updatedUser.getIme().equals(userRequest.getFirstName())
    ));

    verify(passwordEncoder).encode(eq(userRequest.getPassword()));

    if (existingUser.getUloga() == Uloga.VODITELJ) {
        verify(voditeljRepository).save(any(Voditelj.class));
    } else {
        verify(voditeljRepository, never()).save(any(Voditelj.class));
    }
}

```

Slika 5.5: 1. ispitni slučaj za UpdateKorisnikTest

Ispitivanje komponenti za ParkingServiceTest usmjeren je na verifikaciju funkcionalnosti klase ParkingService. Fokusirano je na ispravno označavanje parkirnih mesta kao rezervirana i ponašanje sustava u različitim scenarijima (verifikacija ispravnosti označavanja parkirnih mesta kao rezervirana, provjera ponašanja sustava u slučaju nepostojećeg parkirališta, verifikacija ponašanja sustava kada je parkirno mjesto već označeno i testiranje ispravnosti dobivanja indeksa najmanje udaljenosti). Provjeravamo metodu markSpots koja je dio ParkingServicea. Ta metoda će pridružiti dana parkirališna mjesta s parkiralištem, osim u slučaju kad parkirno već pripada nekom drugom parkiralištu. U slučaju kad su sva mjesta uspješno rezervirana ta metoda vraća null vrijednost, a u suprotnom vraća listu onih mesta koja nismo uspjeli rezervirati. U testu markSpotsSuccess metode želimo provjeriti hoće li metoda vratiti null vrijednost, ako smo rezervirali sva mjesta. Testiramo samo s jednim parkingSpotom koji nema označen Parking, odnosno parkingSpot.getParking() == null i zbog toga očekujemo da će povratna vrijednost poziva markSpots biti null.

```
± philip *
@Test
public void markSpotsSuccess() {

    Long parkingId = 1L;
    String parkingSpotId = "2";
    ParkingSpotReservable reservable = new ParkingSpotReservable();
    reservable.setReservable(true);
    reservable.setSpotId(parkingSpotId);

    Parking parking = new Parking();
    parking.setParkingId(parkingId);
    ParkingSpot parkingSpot = new ParkingSpot();
    parkingSpot.setId(parkingSpotId);

    MarkParkingRequest request = new MarkParkingRequest();
    request.setParkingId(parkingId);
    request.setParkingSpotReservableList(List.of(reservable));

    when(parkingRepository.findById(parkingId)).thenReturn(Optional.of(parking));
    when(parkingSpotRepository.findById(parkingSpotId)).thenReturn(Optional.of(parkingSpot));

    List<ParkingSpotReservable> unmarkableParkingSpots = parkingService.markSpots(request);

    verify(parkingRepository, times( wantedNumberOfInvocations: 1 )).findById(parkingId);
    verify(parkingSpotRepository, times( wantedNumberOfInvocations: 1 )).findById(parkingSpotId);
    verify(parkingSpotRepository, times( wantedNumberOfInvocations: 1 )).save(any(ParkingSpot.class));

    assertNull(unmarkableParkingSpots, message: "No parking spots should be unmarkable");
}
```

Slika 5.6: 1. ispitni slučaj za ParkingServiceTest

S testom markSpotsSomeFoundSomeNot testiramo slučaj kad imamo jedno ispravno i jedno neispravno mjesto. Odnosno kad imamo jedno mjesto koje već pripada nekom parkiralištu i jedno koje ne pripada. U tom slučaju bi trebali spremiti ispravno parkiralište (broj poziva parkingSpotRepository.save() bi trebao biti jedan) i vratiti neispravna mjesta.

```
public void markSpotsSomeFoundSomeNot() {
    Long parkingId = 1L;
    String parkingSpotId = "2";
    String parkingSpotId2 = "3";

    ParkingSpotReservable reservable = new ParkingSpotReservable();
    reservable.setReservable(true);
    reservable.setSpotId(parkingSpotId);

    ParkingSpotReservable reservable2 = new ParkingSpotReservable();
    reservable2.setReservable(true);
    reservable2.setSpotId(parkingSpotId2);

    Parking parking = new Parking();
    parking.setParkingId(parkingId);

    ParkingSpot parkingSpot = new ParkingSpot();
    parkingSpot.setId(parkingSpotId);

    ParkingSpot parkingSpot2 = new ParkingSpot();
    parkingSpot2.setId(parkingSpotId2);
    parkingSpot2.setParking(parking);

    MarkParkingRequest request = new MarkParkingRequest();
    request.setParkingId(parkingId);
    request.setParkingSpotReservableList(Arrays.asList(reservable,reservable2));

    when(parkingRepository.findById(parkingId)).thenReturn(Optional.of(parking));
    when(parkingSpotRepository.findById(parkingSpotId)).thenReturn(Optional.of(parkingSpot));
    when(parkingSpotRepository.findById(parkingSpotId2)).thenReturn(Optional.of(parkingSpot2));

    List<ParkingSpotReservable> unmarkableParkingSpots = parkingService.markSpots(request);

    verify(parkingRepository, times( wantedNumberOfInvocations: 1 )).findById(parkingId);
    verify(parkingSpotRepository, times( wantedNumberOfInvocations: 1 )).findById(parkingSpotId);
    verify(parkingSpotRepository, times( wantedNumberOfInvocations: 1 )).save(any(ParkingSpot.class));

    assertNotNull(unmarkableParkingSpots, message: "Return value should be unmarkedParking spots");
}
```

Slika 5.7: 2. ispitni slučaj za ParkingServiceTest

```

± philip
@Test
public void testGetMinDistanceIndex() {
    // Test case 1: Normal case with positive distances
    List<Double> distances1 = Arrays.asList(3.0, 1.0, 4.0, 2.0);
    long result1 = ParkingService.getMinDistanceIndex(distances1);
    assertEquals(message: "Minimum distance index should be 1", expected: 1L, result1);

    // Test case 2: Normal case with negative distances
    List<Double> distances2 = Arrays.asList(-3.0, -1.0, -4.0, -2.0);
    int result2 = ParkingService.getMinDistanceIndex(distances2);
    assertEquals(message: "Minimum distance index should be 1", expected: 1L, result1);

    // Test case 3: Case with one element
    List<Double> distances3 = List.of( e1: 5.0 );
    int result3 = ParkingService.getMinDistanceIndex(distances3);
    assertEquals(message: "Minimum distance index should be 0 for a single element list", expected: 0, result3);

    // Test case 4: Case with duplicate minimum distances
    List<Double> distances4 = Arrays.asList(1.0, 2.0, 1.0, 3.0);
    int result4 = ParkingService.getMinDistanceIndex(distances4);
    assertEquals(message: "Minimum distance index should be 0 for duplicate minimum distances", expected: 0, result4);
}

```

Slika 5.8: 3. ispitni slučaj za ParkingServiceTest

```

± philip
@Test
public void markSpotsParkingNotFound() {
    // Arrange
    Long parkingId = 1L;
    ParkingSpotReservable reservable = new ParkingSpotReservable();
    reservable.setReservable(true);
    reservable.setSpotId("2");
    MarkParkingRequest request = new MarkParkingRequest();

    request.setParkingId(parkingId);
    request.setParkingSpotReservableList(List.of(reservable));
    when(parkingRepository.findById(parkingId)).thenReturn( t: Optional.empty());

    RequestDeniedException exception = assertThrows(RequestDeniedException.class, () -> {
        | parkingService.markSpots(request);
    });

    verify(parkingRepository, times( wantedNumberOfInvocations: 1 )).findById(parkingId);
    verify(parkingSpotRepository, never()).findById(any());
    verify(parkingSpotRepository, never()).save(any(ParkingSpot.class));

    assertEquals(expected: "Parking with that id doesn't exist", exception.getMessage());
}

```

Slika 5.9: 4. ispitni slučaj za ParkingServiceTest

✓ ✓ sargarepoljupci (com.progi)	584 ms
✓ ✓ CreateKorisnikTest	351 ms
✓ EmailAlreadyExists	290 ms
✓ createKorisnikSuccess	57 ms
✓ createKorisnikEmptyPhoto	2 ms
✓ UsernameAlreadyExists	2 ms
✓ ✓ UpdateKorisnikTest	41 ms
✓ testUpdateKorisnik	41 ms
✓ ✓ ParkingServiceTest	160 ms
✓ markSpotsSuccess	74 ms
✓ markSpotAlreadyMarked	81 ms
✓ markSpotsSomeFoundSomeNot	3 ms
✓ markSpotsParkingNotFound	2 ms
✓ testGetMinDistanceIndex	
✓ ✓ SargarepoljupciApplicationTests	32 ms
✓ contextLoads()	32 ms

Slika 5.10: Rezultati svih ispitnih slučajeva

5.2.2 Ispitivanje sustava

Potrebno je provesti i opisati ispitivanje sustava koristeći radni okvir Selenium¹¹. Razraditi **minimalno 4 ispitna slučaja** u kojima će se ispitati redovni slučajevi, rubni uvjeti te poziv funkcionalnosti koja nije implementirana/izaziva pogrešku kako bi se vidjelo na

¹¹<https://www.seleniumhq.org/>

koji način sustav reagira kada nešto nije u potpunosti ostvareno. Ispitni slučaj se treba sastojati od ulaza (npr. korisničko ime i lozinka), očekivanog izlaza ili rezultata, koraka ispitivanja i dobivenog izlaza ili rezultata.

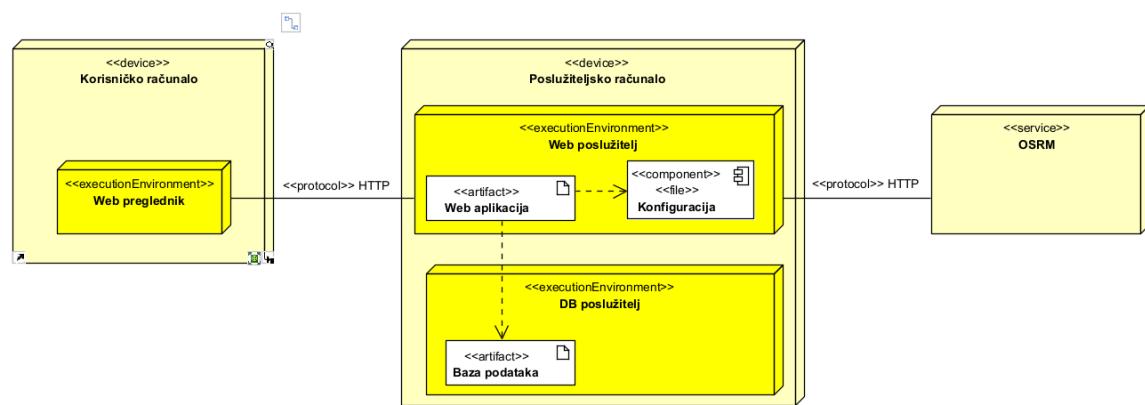
Izradu ispitnih slučajeva pomoću radnog okvira Selenium moguće je provesti pomoću jednog od sljedeća dva alata:

- dodatak za preglednik **Selenium IDE** - snimanje korisnikovih akcija radi automatskog ponavljanja ispita
- **Selenium WebDriver** - podrška za pisanje ispita u jezicima Java, C#, PHP koristeći posebno programsко sučelje.

Detalji o korištenju alata Selenium bit će prikazani na posebnom predavanju tijekom semestra.

5.3 Dijagram razmještaja

UML-dijagrami razmještaja (engl. deployment diagrams) su također vrsta strukturnih UML-dijagrama koji prikazuju fizičku arhitekturu i konfiguraciju razmještaja programskog sustava. Oni ilustriraju raspodjelu programskih komponenti, izvršnih datoteka i knjižnica na sklopovske čvorove ili virtualna izvršna okruženja. Na poslužiteljskom računalu nalaze se Web poslužitelj i poslužitelj baze podataka. Korisnik do web aplikacije dolazi preko web preglednika pri čemu mobilna aplikacija komunicira s poslužiteljem pomoću protokola HTTP. Osim s korisnikom, web poslužitelj komunicira i s OSRM pomoću protokola HTTP.



Slika 5.11: Dijagram razmještaja

5.4 Upute za puštanje u pogon

dio 2. revizije

U ovom poglavlju potrebno je dati upute za puštanje u pogon (engl. deployment) ostvarene aplikacije. Na primjer, za web aplikacije, opisati postupak kojim se od izvornog kôda dolazi do potpuno postavljene baze podataka i poslužitelja koji odgovara na upite korisnika. Za mobilnu aplikaciju, postupak kojim se aplikacija izgradi, te postavi na neku od trgovina. Za stolnu (engl. desktop) aplikaciju, postupak kojim se aplikacija instalira na računalo. Ukoliko mobilne i stolne aplikacije komuniciraju s poslužiteljem i/ili bazom podataka, opisati i postupak njihovog postavljanja. Pri izradi uputa preporučuje se naglasiti korake instalacije uporabom natuknica te koristiti što je više moguće slike ekrana (engl. screenshots) kako bi upute bile jasne i jednostavne za slijediti.

Dovršenu aplikaciju potrebno je pokrenuti na javno dostupnom poslužitelju. Studentima se preporuča korištenje neke od sljedećih besplatnih usluga: Amazon AWS, Microsoft Azure ili Heroku. Mobilne aplikacije trebaju biti objavljene na F-Droid, Google Play ili Amazon App trgovini.

6. Zaključak i budući rad

Zadatak naše ekipe bio je razvoj web aplikacije za spašavanje nestalih osoba iz područja pogodjenih nepogodama. Projekt je bio podijeljen u dvije ključne faze, s ciljem ostvarenja postavljenih zadataka.

U prvoj fazi, koja je trajala od 16. listopada 2023. do 17. studenog 2023., naš tim se okupio kroz PROGl app sustav. Prvi sastanak poslužio je za upoznavanje s projektnim zadatkom, nakon čega je uslijedila precizna podjela zaduženja među članovima tima. Grupiranje u podtimove za frontend, backend i dokumentaciju pokazalo se kao ključno kako bi se efikasno obuhvatile različite aspekte projekta. Fokus prve faze bio je na temeljitom dokumentiranju zahtjeva aplikacije, uz izradu opisa i dijagrama obrazaca uporabe, sekvensijskih dijagrama, opisa, modela i dijagrama baze podataka, te dijagrama razreda. Ovi koraci bili su od neprocjenjive pomoći u drugoj fazi projekta prilikom implementacije funkcionalnosti, pružajući jasnu smjernicu i olakšavajući suradnju između podtimova.

Druga faza projekta, od 4. prosinca 2023. do 19. siječnja 2024., bila je usmjerenja na izradu implementacijskog rješenja. Unatoč tehničkim izazovima koji su proizašli iz nedostatka iskustva u izradi web aplikacija, tim se pokazao odlučnim i spretnim u rješavanju problema. Dodatni fokus bio je na završetku dokumentacije, uključujući dijagrame stanja, aktivnosti, komponenti i razmještaja, te evaluaciju programskog rješenja. Aktivno sudjelovanje na projektu donijelo je vrijedno iskustvo svim članovima tima, pruživši priliku za stjecanje novih vještina u izradi web aplikacija, upoznavanje s novim tehnologijama i alatima, te unaprijeđenje komunikacijskih i organizacijskih vještina.

S obzirom na ostvarene rezultate, grupa je uglavnom zadovoljna krajnjim proizvodom, koji je većinom implementiran u skladu s planiranim funkcionalnostima. Otvorenost za buduće nadogradnje i unaprijeđenja aplikacije ostavlja prostor za kontinuirani razvoj, dok je cijeli projekt pridonio razvoju timskog duha i stvaranju temelja za uspješno suočavanje s izazovima sličnih projekata u budućnosti.

Popis literature

Kontinuirano osvježavanje

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

1. Programsko inženjerstvo, <https://www.fer.unizg.hr/predmet/proinz/projekt>
2. Visual Paradigm, <https://www.visual-paradigm.com/>
3. ERDPlus, <https://erdplus.com/>
4. CROZ tehničke radionice, Prezentacije: frontend, backend (1. i 2. dio) i deployment, <https://moodle.fer.hr/mod/folder/view.php?id=15928>
5. Repozitorij s primjerom deploymenta za potrebe predmeta Programsko inženjerstvo, <https://github.com/progi-devops>
6. J. Tomić, Demo aplikacija React frontenda za potrebe predmeta Programsko inženjerstvo provided by CROZ, <https://gitlab.com/jtomic/opp-project-teams-fronter>
7. H. Šimić, Izrada REST backenda u Spring Bootu, <https://gitlab.com/hrvojesimic/progi-project-teams-backend/-/blob/master/education.md>
8. H. Šimić, Demo aplikacija Spring Boot backenda za potrebe kolegija Programsko inženjerstvo, provided by CROZ, <https://gitlab.com/hrvojesimic/progi-project-teams-backend>
9. A. Rathore, User Registration with Email verification in Java and Spring Boot, <https://rathoreaparna678.medium.com/user-registration-with-email-verification>
10. D. Kontorskyy, Sending Emails with Spring Boot, <https://mailtrap.io/blog/spring-send-email/>
11. Spring Docs, Spring Security, <https://docs.spring.io/spring-security/reference/index.html>
12. SuperSimpleDev, Backend web development - a complete overview, <https://www.youtube.com/watch?v=XBu54nfzxAQ>

13. SuperSimpleDev, Frontend web development - a complete overview, <https://www.youtube.com/watch?v=WG5ikvJ2TKA>
14. H. Šimić, PROGI project backend - a complete overview, https://gitlab.com/hrvojesimic/progi-project-teams-backend/-/tree/p2023/src/main/java/opp?ref_type=heads
15. Baze podataka, Predavanja iz predmeta Baze podataka (11. ER model baze podataka 1. dio, 12. ER model baze podataka 2. dio)
16. Programsко inženjerstvo, Primjer projektne dokumentacije, https://www.fer.unizg.hr/_download/repository/Primjer_projektne_dokumentacije.pdf
17. N. Frid, A. Jović, Modeliranje programske potpore UML-dijagramima, E-skripta (radna inačica), [https://www.fer.unizg.hr/_download/repository/UML_zadaci_za_vjezbu_-_nadopuna_sveucilisnog_prirucnika\[1\].pdf](https://www.fer.unizg.hr/_download/repository/UML_zadaci_za_vjezbu_-_nadopuna_sveucilisnog_prirucnika[1].pdf)
18. E. Paraschiv, Registration with Spring Security – Password Encoding, <https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt>
19. Generiranje tokena za verifikaciju, https://en.wikipedia.org/wiki/Universally_unique_identifier
20. Spring Docs, SQL Databases, <https://docs.spring.io/spring-boot/docs/3.1.5/reference/htmlsingle/index.html#data.sql>
21. Amigoscode, Java Tutorial - Complete User Login and Registration Backend + Email Verification <https://www.youtube.com/watch?v=QwQuro7ekvc>
22. L. Gupta, Spring Boot Security Role-based Authorization <https://howtodoinjava.com/spring-security/spring-boot-role-based-authorization/>
23. Spring Docs, ResponseEntity, <https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-controller/ann-methods/responseentity.html>

Indeks slika i dijagrama

2.1 Prikaz aplikacije SpotHero	8
2.2 Prikaz aplikacije Parkopedia s označenim parkiralištima na ulici	8
2.3 Prikaz aplikacije Parkopedia s označenim parkiralištima na garaži	9
3.1 Dijagram obrasca uporabe, funkcionalnost neregistriranih i registriranih korisnika	20
3.2 Dijagram obrasca uporabe, funkcionalnost administratora i voditelja parkirališta	21
3.3 Sekvencijski dijagram za UC1	22
3.4 Sekvencijski dijagram za UC2	23
3.5 Sekvencijski dijagram za UC7	24
3.6 Sekvencijski dijagram za UC17	25
4.1 Arhitektura sustava	28
4.2 E-R dijagram baze podataka	33
4.3 Dijagram razreda - Controllers	35
4.4 Dijagram razreda - Data transfer objects	36
4.5 Dijagram razreda - Response	37
4.6 Dijagram razreda - Request	38
4.7 Dijagram razreda - Models	39
4.8 Dijagram razreda - Service	40
4.9 Dijagram stanja	41
4.10 Dijagram aktivnosti	43
4.11 Dijagram komponenti	45
5.1 Stvaranje mock objekata	48
5.2 1. ispitni slučaj za CreateKorisnikTest	49
5.3 2. ispitni slučaj za CreateKorisnikTest	49
5.4 3. ispitni slučaj za CreateKorisnikTest	50
5.5 1. ispitni slučaj za UpdateKorisnikTest	51
5.6 1. ispitni slučaj za ParkingServiceTest	52

5.7 2. ispitni slučaj za ParkingServiceTest	53
5.8 3. ispitni slučaj za ParkingServiceTest	54
5.9 4. ispitni slučaj za ParkingServiceTest	54
5.10 Rezultati svih ispitnih slučajeva	55
5.11 Dijagram razmještaja	57

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

Kontinuirano osvježavanje

U ovom dijelu potrebno je redovito osvježavati dnevnik sastanja prema predlošku.

1. sastanak

- Datum: u ovom formatu: 16. listopada 2023.
- Prisustvovali: D.Jenjić, M.Krnić, F.Martinović, L. Matić, J. Rončević, L. Varga, R. Vojinović
- Teme sastanka:
 - međusobno upoznavanje
 - prvi sastanak s asistentom i demonstratorom
 - rasprava o temi i osnovnim funkcionalnostima

2. sastanak

- Datum: u ovom formatu: 25. listopada 2023.
- Prisustvovali: D.Jenjić, M.Krnić, F.Martinović, L. Matić, J. Rončević, L. Varga, R. Vojinović
- Teme sastanka:
 - prva podjela poslova
 - rasprava o korištenim tehnologijama

3. sastanak

- Datum: u ovom formatu: 29. listopada 2023.
- Prisustvovali: D.Jenjić, M.Krnić, F.Martinović, L. Matić, J. Rončević, L. Varga, R. Vojinović
- Teme sastanka:
 - konačna raspodjela dužnosti vezano uz frontend, backend i dokumentaciju
 - rasprava o napravljenoj dokumentaciji

4. sastanak

- Datum: u ovom formatu: 30. listopada 2023.
- Prisustvovali: D.Jenjić, M.Krnić, F.Martinović, L. Matić, J. Rončević, L. Varga, R. Vojinović
- Teme sastanka:
 - uživo termin s asistentom i demonstratorom
 - detaljna rasprava o pojedinim funkcionalnostima aplikacije
 - komentiranje napravljenih obrazaca uporabe i sekveničkih dijagrama

5. sastanak

- Datum: u ovom formatu: 14. studenog 2023.
- Prisustvovali: D.Jenjić, M.Krnić, F.Martinović, L. Matić, J. Rončević, L. Varga, R. Vojinović
- Teme sastanka:
 - rasprava o dosad napravljenom backendu i frontendu
 - rasprava o dijelu dokumentacije koji se tiče baze podataka i ER dijagrama
 - dodatna podjela posla unutar frontenda, backenda i dokumentacije

6. sastanak

- Datum: u ovom formatu: 16. studenog 2023.
- Prisustvovali: D.Jenjić, M.Krnić, F.Martinović, L. Matić, J. Rončević, L. Varga, R. Vojinović
- Teme sastanka:
 - rasprava o povezivanju frontenda i backenda
 - podjela posla oko preostale dokumentacije
 - rasprava o deploymentu aplikacije

Tablica aktivnosti

Kontinuirano osvježavanje

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

	Matko Krnić	Domagoj Jenjić	Filip Martinović	Lovro Matić	Jure Rončević	Luka Varga	Roko Vojinović
Upravljanje projektom	8						
Opis projektnog zadatka					9		
Funkcionalni zahtjevi	3						
Opis pojedinih obrazaca					3		
Dijagram obrazaca	4					7	
Sekvencijski dijagrami					1		4
Opis ostalih zahtjeva		1					
Arhitektura i dizajn sustava				8	2		
Baza podataka			10				
Dijagram razreda	5				2		
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Korištene tehnologije i alati							
Ispitivanje programskog rješenja							
Dijagram razmještaja							
Upute za puštanje u pogon							

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Matko Krnić	Domagoj Jenjić	Filip Martinović	Lovro Matić	Jure Rončević	Luka Varga	Roko Vojinović
Dnevnik sastajanja					2		
Zaključak i budući rad							
Popis literature					2		
Implementacija frontenda		14			7		
Implementacija backenda	9		20				
Deployment	8	8					
Izrada početne stranice		5					
Izrada baze podataka			10				

Dijagrami pregleda promjena

dio 2. revizije

Prenijeti dijagram pregleda promjena nad datotekama projekta. Potrebno je na kraju projekta generirane grafove s gitlaba prenijeti u ovo poglavlje dokumentacije. Dijagrami za vlastiti projekt se mogu preuzeti s gitlab.com stranice, u izborniku Repository, pritiskom na stavku Contributors.