

Programsko inženjerstvo
Ak. god. 2020./2021.

Parkiraj Me
Dokumentacija, Rev. 2

Grupa: *seVen*
Voditelj: *Marin Ćubela*

Datum predaje: 14. 1. 2021

Nastavnik: *Nikolina Frid*

Sadržaj

| | |
|--|-----------|
| 1 Dnevnik promjena dokumentacije | 3 |
| 2 Opis projektnog zadatka | 5 |
| 3 Specifikacija programske potpore | 8 |
| 3.1 Funkcionalni zahtjevi | 8 |
| 3.1.1 Obrasci uporabe | 10 |
| 3.1.2 Sekvencijski dijagrami | 24 |
| 3.2 Ostali zahtjevi | 28 |
| 4 Arhitektura i dizajn sustava | 29 |
| 4.1 Baza podataka | 31 |
| 4.1.1 Opis tablica | 31 |
| 4.1.2 Dijagram baze podataka | 36 |
| 4.2 Dijagram razreda | 37 |
| 4.3 Dijagram stanja | 48 |
| 4.4 Dijagram aktivnosti | 49 |
| 4.5 Dijagram komponenti | 51 |
| 5 Implementacija i korisničko sučelje | 52 |
| 5.1 Korištene tehnologije i alati | 52 |
| 5.2 Ispitivanje programskog rješenja | 54 |
| 5.2.1 Ispitivanje komponenti | 54 |
| 5.2.2 Ispitivanje sustava | 58 |
| 5.3 Dijagram razmještaja | 64 |
| 5.4 Upute za puštanje u pogon | 65 |
| 5.4.1 Frontend aplikacija | 66 |
| 5.4.2 Backend aplikacija | 69 |
| 5.4.3 Baza podataka | 71 |
| 6 Zaključak i budući rad | 74 |

| | |
|---|-----------|
| Popis literature | 76 |
| Indeks slika i dijagrama | 78 |
| Dodatak: Prikaz aktivnosti grupe | 79 |

1. Dnevnik promjena dokumentacije

| Rev. | Opis promjene/dodatka | Autori | Datum |
|------|--|------------------------------------|-------------|
| 0.1 | Preuzet predložak. | Ćubela | 20.10.2020. |
| 0.2 | Dodan opis projektnog zadatka. | Matković, Strejček | 22.10.2020. |
| 0.3 | Dodani obrasci uporabe. | Petković | 21.10.2020. |
| 0.4 | Dodani obrasci uporabe. | Bakić, Djaković | 25.10.2020. |
| 0.5 | Dodani dijagrami obrazaca uporabe. | Krišto, Ćubela | 29.10.2020. |
| 0.6 | Dodana dokumentacija za bazu podataka. | Krišto, Strejček | 5.11.2020. |
| 0.7 | Dodani sekvencijski dijagrami. | Djaković, Matković, Strejček | 10.11.2020. |
| 0.8 | Popravljeni obrasci uporabe | Bakić, Matković, Strejček | 12.11.2020. |
| 0.9 | Popravljeni dijagrami | Krišto, Strejček | 13.11.2020. |
| 0.10 | Dodan opis arhitekture sustava | Petković | 13.11.2020. |
| 0.11 | Dodani dijagrami razreda | Bakić, Ćubela, Krišto | 13.11.2020. |
| 1.0 | Gotova prva verzija | | 13.11.2020. |
| 1.1 | Ispravke s prve predaje | Strejček | 9.12.2020. |
| 1.2 | Dodane upute za puštanje u pogon i korištene tehnologije i alati | Djaković | 12.01.2021. |
| 1.3 | Dodani dijagrami(act, stm, cmp) i popravljeni obrasci uporabe | Matković, Strejček | 13.01.2021. |
| 1.4 | Dodan zaključak i budući rad | Ćubela | 14.1.2021. |

| Rev. | Opis promjene/dodataka | Autori | Datum |
|------|--|---------------------|-------------|
| 1.5 | Dopunjeni dijagrami razreda | Bakić, Krišto | 14.01.2021. |
| 1.6 | Ispravljena arhitektura i dizajn sustava | Bakić, Petković | 14.01.2021. |
| 1.7 | Ispravljeni dijagrami obrazaca uporabe | Krišto | 14.01.2021. |
| 1.8 | Dodano ispitivanje programskog rješenja | Ćubela, Petković | 14.01.2021. |
| 1.9 | Dodani dijagrami razmještaja | Petković | 14.01.2021. |
| 1.10 | Dodana tablica aktivnosti grupe | Ćubela, Strejček | 14.01.2021. |
| 1.11 | Dodan dnevnik sastajanja | Bakić | 14.01.2021. |
| 1.12 | Dodan dijagram pregleda promjena | Ćubela | 14.01.2021. |
| 2.0 | Gotova finalna verzija | | 14.1.2021. |

2. Opis projektnog zadatka

Cilj projekta je razviti programsku podršku za stvaranje web aplikacije *Parkiraj me*. Aplikacija će korisniku omogućiti praćenje stanja slobodnih parkirališnih mesta na svim lokacijama diljem Zagreba u stvarnom vremenu te jednokratnu, ponavljajuću ili trajnu rezervaciju parkirališnog mesta. Putem aplikacije sve zainteresirane tvrtke moći će ponuditi svoja parkirališta na kojima će korisnici kasnije moći rezervirati svoja mjesta.

Pokretanjem aplikacije prikazuje se karta s prikazanim dostupnim parkiralištima i brojem slobodnih parkirališnih mesta te opcija za registraciju ili prijavu u sustav. Neprijavljeni korisnik može se prijaviti na postojeći račun (mora upisati e-mail adresu i lozinku). Neregistrirani korisnik ima mogućnost registracije kao: *Klijent* (rezervira parkirališna mjesta) ili *Tvrtka* (ovlašteni zaposlenik tvrtke).

Za klijenta potrebni podaci za registraciju su:

- OIB
- Ime i prezime
- E-mail adresa
- Lozinka
- Broj kreditne kartice

Za tvrtku potrebni podaci za registraciju su:

- OIB
- Naziv tvrtke
- E-mail adresa
- Lozinka
- Adresa sjedišta

Klijent je registrirani korisnik stariji od 18 godina, koji ima pravo rezervirati parkirno mjesto. Klijent na karti odabire parkiralište na kojem želi rezervirati mjesto i dobiva informaciju o broju slobodnih mjesta. Druga mogućnost je da na temelju trenutne lokacije aplikacija daje prijedlog najbližih parkirališta sa slobodnim parkirnim mjestima (prednost imaju parkirališta s brojem slobodnih mjesta većim od 10). Ukoliko odabrano parkiralište ima slobodnih mjesta odabire mogućnost rezervacije. Nakon toga klijent bira tip rezervacije koju želi i odabire vozilo za koje želi izvršiti rezervaciju. Rezervacija se naplaćuje izravnim terećenjem njegove kreditne kartice. Potvrdom plaćanja aktivira se rezervacija u sustavu.

Klijent istovremeno može imati više različitih tipova rezervacija, na različitim parkiralištima s različitim vozilima.

Tvrtka je korisnik kojeg prijavljuje ovlašteni zaposlenik tvrtke. Ovlašteni zaposlenik nakon registracije svoje tvrtke unosi podatke o parkiralištima koje nudi ta tvrtka. Odabirom opcije za dodavanje parkirališta unosi se:

- ime parkirališta
- lokacija parkirališta
- tip parkirališta (otvoreno ili zatvoreno)
- ukupni broj mjesta
- broj invalidskih mjesta (potrebno za određivanje broja slobodnih mjesta)
- cijena parkinga

Preduvjet za prijavu parkirališta je opremljenost pametnim kamerama na svakom parkirališnom mjestu kako bi se prepoznala zauzeta mjesta te kako bi se mogla očitati registracija vozila na parkirališnom mjestu. Nakon uspješne prijave parkirališta, to parkiralište postaje vidljivo svim korisnicima aplikacije na karti te su na njemu moguće rezervacije.

Administrator sustava upravlja svim korisnicima aplikacije. On ima pristup popisu svih registriranih korisnika i njihovih osnovnih podataka (ime, prezime, e-mail adresa, popis vozila). Ima pristup popisu rezervacija, također može potvrditi dodavanje ili brisanje parkirališta.

Upravljanje korisničkim računom

Korisnici imaju mogućnost upravljanja svojim računima, pregledavanjem, uređivanjem i brisanjem svojeg računa.

Klijent ima pregled svih svojih prijavljenih vozila. Svako vozilo ima svoju registraciju, ime i boju. Vozilo klijent može uređivati (promjenom registracije, imena ili boje vozila). Može dodavati nova vozila ili postojeća obrisati.

Klijent ima prijavljenu kreditnu karticu. Karticu može uređivati u slučaju pogrešnog unosa i postojeću zamijeniti novom, dakle izbrisati staru i stvoriti novu. Klijent uvijek treba imati prijavljenu jednu karticu.

Tvrtka ima pregled vlastitih parkirališta. Svako parkiralište može uređivati, može dodavati nova parkirališta ili brisati postojeća.

Rezervacija

Parkirno mjesto se može rezervirati jednokratno, kao ponavljača rezervacija ili kao trajna rezervacija. Svaka rezervacija ima svoj jedinstveni id. Cijena rezervacije ovisi o tipu rezervacije i cjeniku pojedinog parkirališta.

Jednokratna rezervacija se rezervira za vremenski period kraći od 24 sata i rezervacija se mora obaviti barem 6 sati unaprijed. Plaćanje se vrši odmah u trenutku rezervacije.

Ponavljača rezervacija mora trajati najmanje 1 sat i mora se ponavljati barem jednom tjedno tijekom mjesec dana. Plaćanje se vrši svakih 30 dana.

Trajna rezervacija odnosi se na rezervaciju parkirališnog mjesta 0-24 sata svaki dan na neodređeni period. Plaćanje se vrši svakih 30 dana.

Dodatno:

Ukoliko se izbriše korisnički račun neke tvrtke, brišu se sva njena parkirališta i ukidaju se rezervacije na tim parkiralištima. Ukoliko se izbriše korisnički račun klijenta, brišu se i sva njegova vozila i sve njegove rezervacije. Ukoliko tvrtka izbriše neko parkiralište, brišu se i sve rezervacije napravljene na tom parkiralištu. Ukoliko klijent izbriše vozilo, brišu se i sve rezervacije napravljene s tim vozilom.

Sustav treba podržavati rad više korisnika u stvarnom vremenu.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Ovlašteni zaposlenik tvrtke
2. Klijent
3. Administrator
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) vidjeti kartu s parkiralištima (adresa, broj ukupnih i slobodnih mesta, cijena)
 - (b) registrirati se i/ili prijaviti se (kao klijent ili tvrtka)
2. Klijent (inicijator) može:
 - (a) vidjeti kartu s parkiralištima (adresa, broj ukupnih i slobodnih mesta, cijena)
 - (b) pregledavati, uređivati i izbrisati korisnički račun
 - (c) pregledavati, uređivati i izbrisati vozila
 - (d) pregledavati, uređivati i izbrisati karticu
 - (e) rezervirati parkirališno mjesto
 - (f) pregledavati, uređivati i izbrisati rezervacije
 - (g) tražiti upute do parkirališta
3. Ovlašteni zaposlenik tvrtke (inicijator) može:
 - (a) vidjeti kartu s parkiralištima
 - (b) pregledavati, uređivati, dodavati i brisati svoja parkirališta
 - (c) pregledavati, uređivati i izbrisati korisnički račun
 - (d) pregledavati rezervacije na vlastitim parkiralištima

4. Administrator (inicijator) može:

- (a) vidjeti kartu s parkiralištima
- (b) pregledavati, uređivati i izbrisati korisnički račun
- (c) vidjeti popis svih registriranih korisnika i osnovnih podataka
- (d) brisati i mijenjati razinu pristupa aplikaciji drugim korisnicima

5. Baza podataka (sudionik) može:

- (a) pohranjivati podatke o korisnicima (osobni podaci, kartica, vozila)
- (b) pohranjivati podatke o parkiralištima (njihovim kapacitetima i ponudama)
- (c) pohranjivati podatke o rezervacijama
- (d) pohranjivati podatke o tvrtkama

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled karte

- **Glavni sudionik:** Korisnik/Klijent/Tvrtka/Administrator
- **Cilj:** Pogledati kartu s označenim parkiralištima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik/Klijent/Tvrtka/Administrator je u aplikaciji
- **Opis osnovnog tijeka:**
 1. Korisnik/Klijent/Tvrtka/Administrator otvara početnu stranicu
 2. Prikazuje se karta s označenim parkiralištima

UC2 - Pregled podataka o parkiralištu na karti

- **Glavni sudionik:** Korisnik/Klijent/Tvrtka/Administrator
- **Cilj:** Pregled podataka o parkiralištu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC1
- **Opis osnovnog tijeka:**
 1. Korisnik/Klijent/Tvrtka/Administrator pritišće na oznaku parkirališta na karti
 2. Prikazuju se osnovni podaci o parkiralištu

UC3 - Prijava klijenta/tvrtke/administratora

- **Glavni sudionik:** Klijent/Tvrtka/Administrator
- **Cilj:** Prijaviti klijenta/tvrtku/administratora
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent/Tvrtka/Administrator nije prijavljen
- **Opis osnovnog tijeka:**
 1. Klijent/Tvrtka/Administrator odabire opciju za prijavu
 2. Otvara se forma za popunjavanje korisničkih podataka potrebnih za prijavu
 3. Klijent/Tvrtka/Administrator je prijavljen u aplikaciju i preusmјeren na početni zaslon aplikacije
- **Opis mogućih odstupanja:**
 - 2.a Nevažeći podaci prilikom prijave klijenta/tvrtke/administratora
 1. Klijentu/Tvrtki/Administratoru se prikazuje poruka o nevažećim podacima
 2. Klijent/Tvrtka/Administrator ispravlja nevažeće podatke za prijavu ili odustaje od prijave

UC4 - Registracija korisnika

- **Glavni sudionik:** Korisnik
- **Cilj:** Registrirati korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik nije prijavljen niti registriran
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Otvara se forma za popunjavanje korisničkih podataka
 3. Korisnik otvara novi račun
 4. Korisnik je automatski prijavljen u sustav i preusmјeren na početni zaslon aplikacije
- **Opis mogućih odstupanja:**
 - 2.a Nevažeći podaci prilikom registracije korisnika
 1. Korisnik se obavještava o neuspjeloj registraciji
 2. Korisnik ispravlja nevažeće podatke te pokušava ponovno ili odustaje od registracije

UC5 - Pregled podataka o korisničkom računu

- **Glavni sudionik:** Klijent/Tvrtka/Administrator
- **Cilj:** Pregledati podatke o korisničkom računu
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent/Tvrtka/Administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. Klijent/Tvrtka/Administrator otvara podatke o korisničkom računu
 2. Prikaže se podaci o korisničkom računu

UC6 - Uređivanje podataka o korisničkom računu

- **Glavni sudionik:** Klijent/Tvrtka/Administrator
- **Cilj:** Urediti podatke o korisničkom računu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC5
- **Opis osnovnog tijeka:**
 1. Klijent/Tvrtka/Administrator odabire akciju za uređivanje
 2. Prikaže se forma za uređivanje podataka o korisničkom računu
 3. Klijent/Tvrtka/Administrator uređuje podatke
 4. Klijent/Tvrtka/Administrator sprema promjene
- **Opis mogućih odstupanja:**
 - 3.a Nevažeći podaci prilikom spremanja promjena
 1. Klijentu/Tvrtki/Administratoru se prikazuje poruka o nevažećim podacima
 2. Klijent/Tvrtka/Administrator ispravlja podatke
 3. Klijent/Tvrtka/Administrator ponovno sprema podatke

UC7 - Brisanje korisničkog računa

- **Glavni sudionik:** Klijent/Tvrtka/Administrator
- **Cilj:** Izbrisati korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** UC5
- **Opis osnovnog tijeka:**
 1. Klijent/Tvrtka/Administrator odabire akciju za brisanje računa
 2. Briše se korisnikov račun
 3. Klijent/Tvrtka/Administrator se preusmjerava na početni zaslon

UC8- Rezervacija parkirališnog mjesta

- **Glavni sudionik:** Klijent
- **Cilj:** Rezervirati parkirališno mjesto na parkiralištu
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je odabrao parkiralište
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju rezervacije parkirališnog mjesata
 2. Klijent odabira tip rezervacije
 3. Otvara se forma za popunjavanje podataka o rezervaciji
 4. Klijenta se preusmjerava na plaćanje
 5. Klijent potvrđuje plaćanje
 6. Klijenta se tereti za navedeni iznos
 7. Prikazuje se poruka o uspješnom plaćanju
 8. Klijenta se preusmjerava na kartu
- **Opis mogućih odstupanja:**
 - 3.a Nema slobodnih parkirališnih mjesta
 1. Klijenta se obavještava porukom o zauzetosti parkirališta
 2. Klijent mijenja podatke o rezervaciji ili odustaje od rezervacije
 - 3.b Nevažeći podaci prilikom rezervacije
 1. Klijent se obavještava o neuspjeloj rezervaciji
 2. Klijent ispravlja nevažeće podatke te pokušava ponovno ili odustaje od rezervacije
 - 5.a Nedovoljno sredstava na kartici
 1. Klijenta se obavještava porukom o neuspjelom plaćanju

UC9- Upute do parkirališta

- **Glavni sudionik:** Klijent
- **Cilj:** Navođenje do parkirališta
- **Sudionici:** Baza podataka
- **Preduvjet:** UC2
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju za navigaciju
 2. Otvara se navigacija na korisnikovom uređaju

UC10 - Pregled popisa rezervacija

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati popis vlastitih rezervacija
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju pregleda rezervacija
 2. Prikaže se popis svih aktivnih i neaktivnih rezervacija s osnovnim informacijama

UC11- Uređivanje podataka o rezervaciji

- **Glavni sudionik:** Klijent
- **Cilj:** Urediti podatke rezervacije
- **Sudionici:** Baza podataka
- **Preduvjet:** UC10
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju za uređivanje rezervacije
 2. Otvara se forma za uređivanje podataka o rezervaciji popunjena s podacima odabrane rezervacije
 3. Klijent uređuje podatke
 4. Klijent spremi promjene
 5. Klijenta se preusmjerava na popis ažurirani popis rezervacija
- **Opis mogućih odstupanja:**
 - 3.a Nevažeći podaci prilikom spremanja promjena
 1. Klijentu se prikazuje poruka o nevažećim podacima
 2. Klijent ispravlja podatke
 3. Klijent ponovno spremi podatke

UC12- Uklanjanje rezervacije

- **Glavni sudionik:** Klijent
- **Cilj:** Ukloniti rezervaciju
- **Sudionici:** Baza podataka
- **Preduvjet:** UC10
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju za uklanjanje rezervacije
 2. Rezervacija se uklanja
 3. Klijenta se preusmjerava na popis ažurirani popis rezervacija

UC13 - Pregled podataka o vozilima

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati podatke o vozilima
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen
- **Opis osnovnog tijeka:**
 1. Klijent otvara popis vozila
 2. Prikažu se podaci o vozilima

UC14 - Dodavanje vozila

- **Glavni sudionik:** Klijent
- **Cilj:** Dodati vozilo
- **Sudionici:** Baza podataka
- **Preduvjet:** UC13
- **Opis osnovnog tijeka:**
 1. Klijent otvara formu za dodavanje vozila
 2. Unose se svi potrebni podaci
 3. Klijent odabire akciju za dodavanje vozila
 4. Vozilo se dodaje
 5. Klijenta se preusmjerava na popis ažurirani popis vozila
- **Opis mogućih odstupanja:**
 - 2.a Nevažeći podaci prilikom dodavanja vozila
 1. Klijentu se prikazuje poruka o nevažećim podacima
 2. Klijent ispravlja podatke
 3. Klijent ponovno odabire akciju za dodavanje vozila

UC15 - Uređivanje vozila

- **Glavni sudionik:** Klijent
- **Cilj:** Urediti podatke o vozilu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC13
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju za uređivanje vozila
 2. Otvara se forma za uređivanje vozila popunjena s podacima odabranoga vozila
 3. Klijent uređuje podatke
 4. Klijent spremi promjene
 5. Klijenta se preusmjerava na popis ažurirani popis vozila
- **Opis mogućih odstupanja:**
 - 3.a Nevažeći podaci prilikom spremanja promjena
 1. Klijentu se prikazuje poruka o nevažećim podacima
 2. Klijent ispravlja podatke
 3. Klijent ponovno spremi podatke

UC16 - Uklanjanje vozila

- **Glavni sudionik:** Klijent
- **Cilj:** Ukloniti vozilo
- **Sudionici:** Baza podataka
- **Preduvjet:** UC13
- **Opis osnovnog tijeka:**
 1. Klijent odabire akciju uklanjanja vozila
 2. Vozilo se uklanja zajedno sa svim rezervacijama napravljenim s tim vozilom
 3. Klijenta se preusmjerava na ažurirani popis vozila

UC17 - Pregled podataka o parkiralištu

- **Glavni sudionik:** Tvrta
- **Cilj:** Pregledati podatke o parkiralištu
- **Sudionici:** Baza podataka
- **Preduvjet:** Tvrta je prijavljena
- **Opis osnovnog tijeka:**
 1. Tvrta otvara popis parkirališta
 2. Prikažu se podaci o parkiralištima

UC18 - Dodavanje parkirališta

- **Glavni sudionik:** Tvrta
- **Cilj:** Dodati parkiralište
- **Sudionici:** Baza podataka
- **Preduvjet:** UC17
- **Opis osnovnog tijeka:**
 1. Tvrta otvara formu za dodavanje parkirališta
 2. Unose se svi potrebni podaci
 3. Tvrta odabire akciju za dodavanje parkirališta
 4. Parkiralište postaje javno vidljivo na karti i u popisu parkirališta svim korisnicima aplikacije
- **Opis mogućih odstupanja:**
 - 2.a Nevažeći podaci prilikom dodavanja parkirališta
 1. Tvrtki se prikazuje poruka o nevažećim podacima
 2. Tvrta ispravlja podatke
 3. Tvrta ponovno odabire akciju za dodavanje parkirališta

UC19 - Uređivanje parkirališta

- **Glavni sudionik:** Tvrta
- **Cilj:** Urediti podatke o parkiralištu
- **Sudionici:** Baza podataka
- **Preduvjet:** UC17, UC2
- **Opis osnovnog tijeka:**
 1. Tvrta odabire opciju za uređivanje parkirališta
 2. Tvrta uređuje podatke
 3. Tvrta sprema promjene
 4. Tvrta se preusmjerava na ažurirani popis parkirališta
- **Opis mogućih odstupanja:**
 - 2.a Nevažeći podaci prilikom spremanja promjena
 1. Tvrta prikazuje poruka o nevažećim podacima
 2. Tvrta ispravlja podatke
 3. Tvrta ponovno sprema podatke

UC20 - Uklanjanje parkirališta

- **Glavni sudionik:** Tvrta
- **Cilj:** Ukloniti parkiralište
- **Sudionici:** Baza podataka
- **Preduvjet:** UC17
- **Opis osnovnog tijeka:**
 1. Tvrta odabire opciju za uklanjanja parkirališta
 2. Parkiralište se uklanja zajedno sa svim rezervacijama napravljenim na tom parkiralištu
 3. Tvrta se preusmjerava na ažurirani popis parkirališta

UC21 - Pregled registriranih korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregled podataka o registriranim korisnicima
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. Administrator na administratorskoj ploči odabire opciju pregledavanja svih registriranih korisnika
 2. Prikaže se popis svih korisnika koji su registrirani u aplikaciju

UC22 - Pregled registriranog korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati podatke o registriranom korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** UC21
- **Opis osnovnog tijeka:**
 1. Na popisu registriranih korisnika odabire jednog korisnika
 2. Prikažu se osnovni podaci o odabranom korisniku

UC23 - Uklanjanje registriranog korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Izbrisati račun registriranog korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** UC21
- **Opis osnovnog tijeka:**
 1. Administrator odabire akciju za uklanjanje korisnika
 2. Korisnik se uklanja
 3. Administrator se preusmjerava na ažurirani popis korisnika

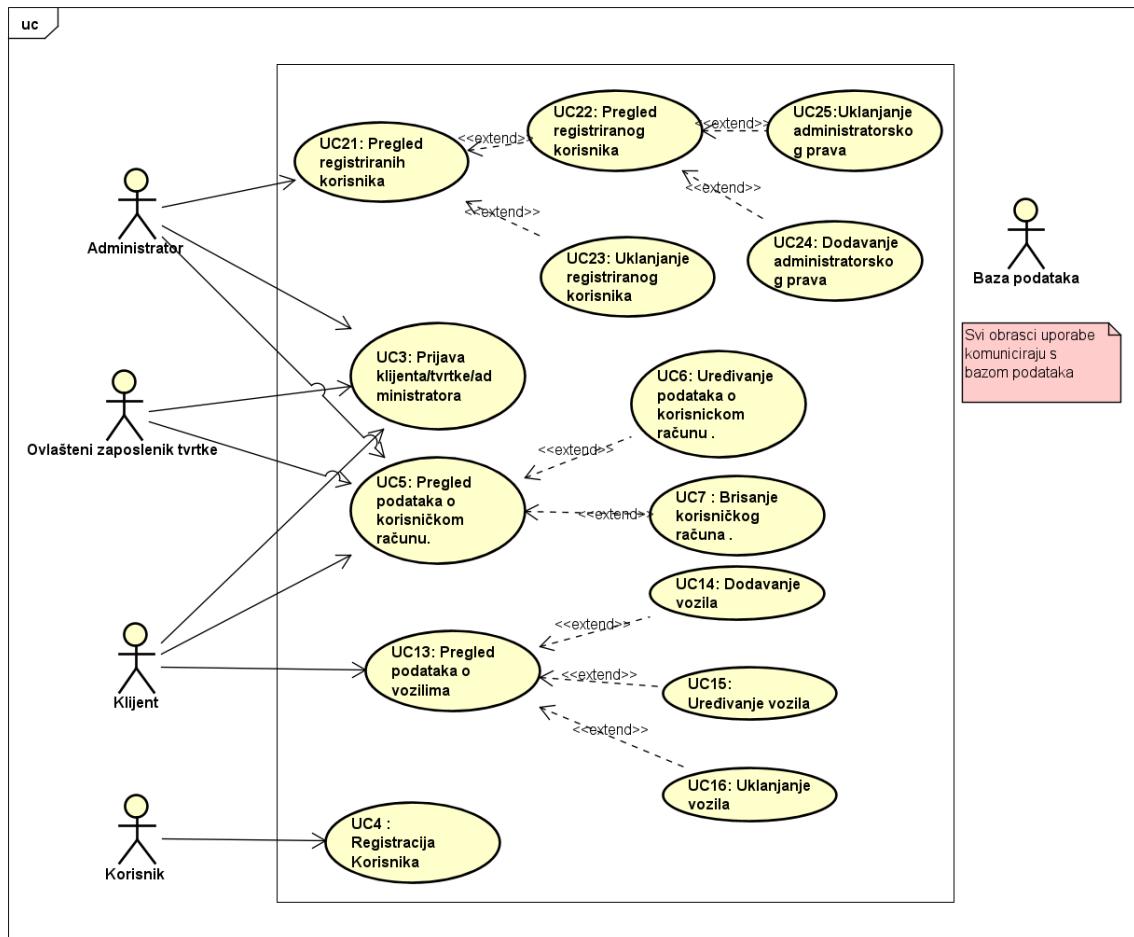
UC24 - Dodavanje administratorskog prava

- **Glavni sudionik:** Administrator
- **Cilj:** Korisniku dodjeliti status administratora
- **Sudionici:** Baza podataka
- **Preduvjet:** UC22
- **Opis osnovnog tijeka:**
 1. Administrator odabire akciju za postavljanje administratorskog prava korisnika
 2. Korisniku se dodjeluje status administratora
 3. Administrator se preusmjerava na popis korisnika

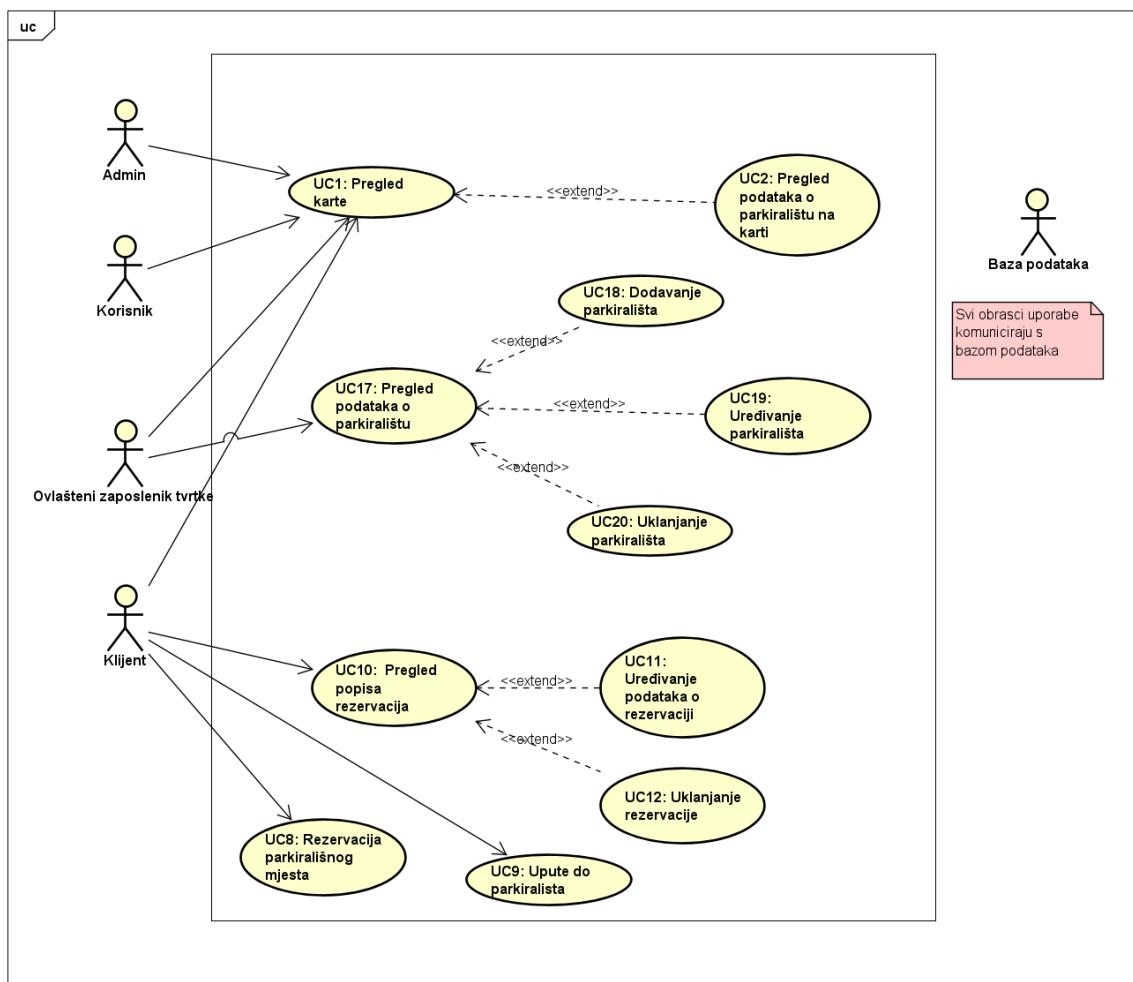
UC25 - Uklanjanje administratorskog prava

- **Glavni sudionik:** Administrator
- **Cilj:** Korisniku oduzeti status administratora
- **Sudionici:** Baza podataka
- **Preduvjet:** UC22
- **Opis osnovnog tijeka:**
 1. Administrator odabire akciju za uklanjanje administratorskog prava korisnika
 2. Korisniku se oduzima status administratora
 3. Administrator se preusmjerava na popis korisnika

Dijagrami obrazaca uporabe



Slika 3.1: Prikaz funkcionalnosti vezanih za korisničke podatke

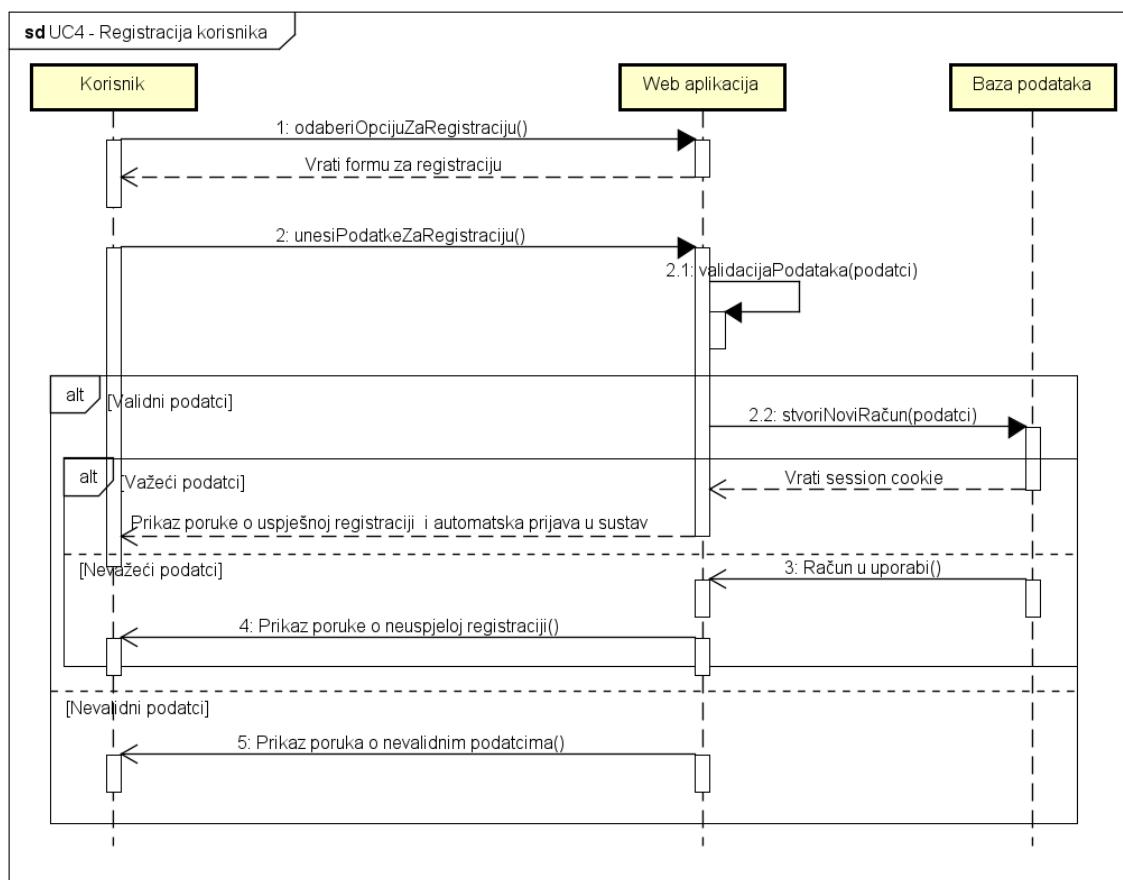


Slika 3.2: Pregled ostalih funkcionalnosti

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC4 - Registracija korisnika

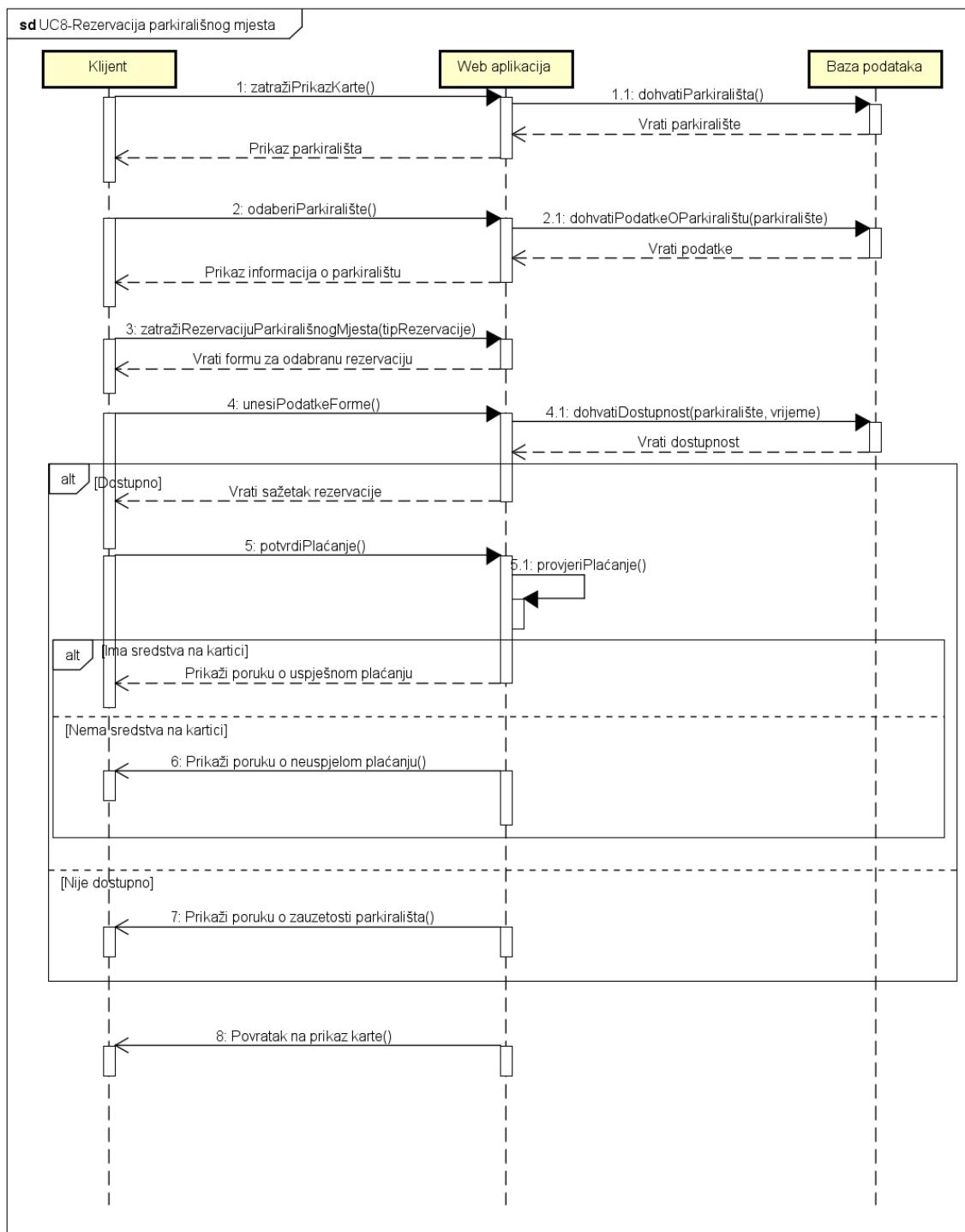
Korisnik šalje zahtjev za registraciju. Poslužitelj prikazuje formu za popunjavanje korisničkih podataka. Korisnik unosi tražene podatke. U slučaju da neki od traženih podataka nisu validni (podaci nisu uneseni ili su uneseni neispravno) poslužitelj prikazuje odgovarajuću poruku za pojedini podatak korisnika. Ako su podaci validni, a nisu važeći (već postoji korisnik s istom e-mail adresom ili OIB-om ili brojem kartice) poslužitelj prikazuje korisniku poruku o neuspjeloj registraciji. Ako su podaci važeći poslužitelj prikazuje poruku o uspješnoj registraciji te ga automatski prijavljuje u aplikaciju.



Slika 3.3: Sekvencijski dijagram za UC4

Obrazac uporabe UC8 - Rezervacija parkirališnog mjesta

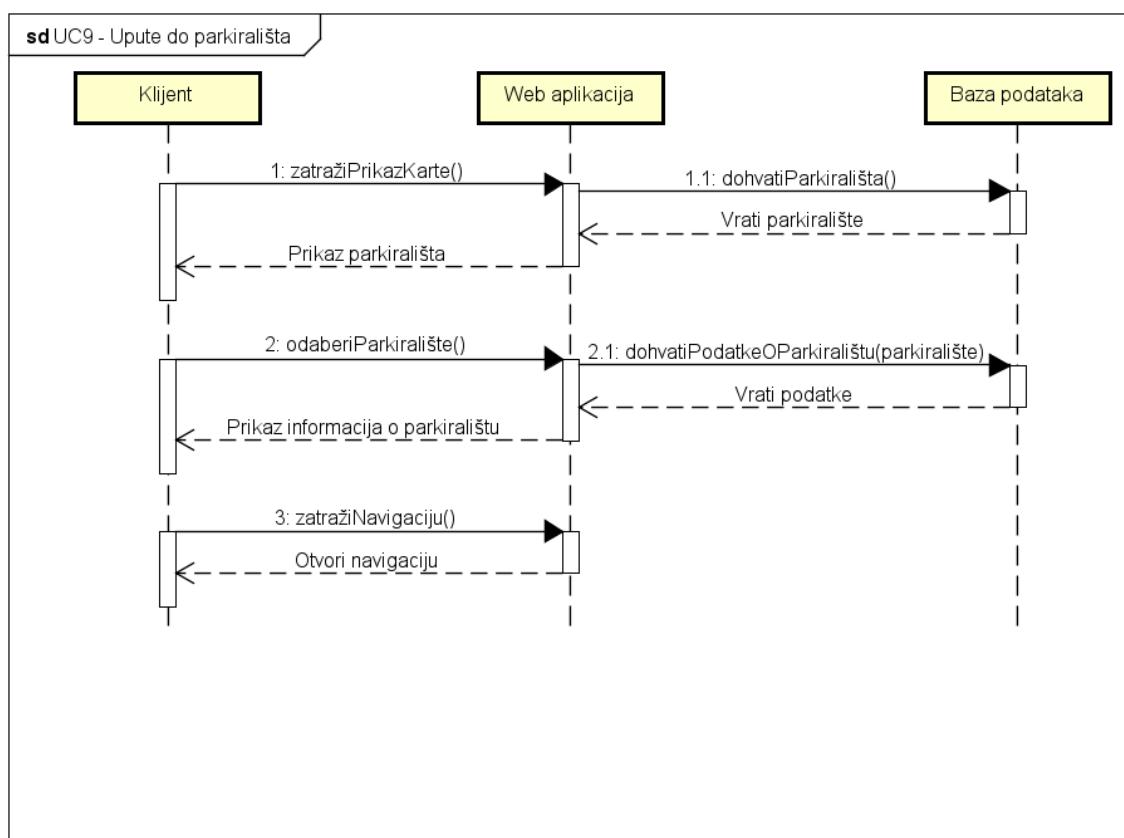
Klijent šalje zahtjev za prikaz karte s parkiralištima. Poslužitelj dohvaća i prikazuje prijavljena parkirališta. Odabirom parkirališta, poslužitelj iz baze podataka dohvaća osnovne podatke o parkiralištu i prikazuje ih korisniku. Kako bi obavio rezervaciju parkirališnog mjesta, klijent šalje zahtjev s tipom rezervacije i potrebne informacije vezane za rezervaciju. Poslužitelj provjerava ispravnost primljenih podatka o odabranoj rezervaciji te iz baze podataka dohvaća i provjera postoji li slobodno mjesto za traženi period na parkiralištu. Ukoliko nema slobodnih parkirališnih mjesta, sustav obavještava o tome klijenta. Ako postoji slobodno mjesto, proces rezervacije se nastavlja te se klijentu prikazuje sažetak rezervacije. Klijent tada potvrđuje plaćanje, ako kartica nema dovoljno sredstva na računu poslužitelj obavještava klijenta o neuspjelom plaćanju i vraća ga na prikaz karte. U slučaju da kartica ima dovoljno sredstva, sredstva se skidaju izravnim terećenjem. Rezervacija je završena i poslužitelj informaciju o rezervaciji prosljeđuje bazi koja sprema promjenu.



Slika 3.4: Sekvencijski dijagram za UC8

Obrazac uporabe UC9 - Upute do parkirališta

Klijent šalje zahtjev za prikaz karte s parkiralištima. Poslužitelj dohvaća i prikazuje prijavljena parkirališta. Klijent odabire parkiralište do kojeg želi doći. Odbirom parkirališta, poslužitelj iz baze podataka dohvaća osnovne podatke o parkiralištu i prikazuje ih korisniku. Klijent šalje zahtjev za navigaciju do parkirališta. Poslužitelj otvara navigaciju te klijent dobiva upute kako doći do željenog parkirališta.



Slika 3.5: Sekvencijski dijagram za UC9

3.2 Ostali zahtjevi

- Sustav treba imati programsku potporu za web platformu s javnim sučeljem te prikazom karte s parkiralištima
- Sustav treba omogućavati istovremeni pristup za više korisnika
- Programsko sučelje treba podržavati hrvatski jezik i dijakritičke znakove njegove abecede
- Sustav treba podržavati plaćanje u kunama
- Dohvat podataka iz baza mora se obaviti unutar nekoliko sekundi
- Sustav mora imati podršku za senzore koji očitavaju parkirna mjesta te komuniciraju s bazom podataka i redovito ju osvježavaju prilikom promjene stanja
- Sustav treba podržavati 3 vrste rezervacija:
 1. Jednokratne rezervacije koje moraju biti napravljene barem 6 sati unaprijed te traju do 24 sata
 2. Ponavljaće rezervacije koje moraju trajati barem 1 sat tjedno u razdoblju od minimalno mjesec dana
 3. Trajne rezervacije (0-24)
- Sustav mora imati implementiranu podršku za oporavak u slučaju neispravnog korištenja korisničkog sučelja
- Sustav mora zabraniti pristup privatnim rutama. Ako korisnik nije bio prijavljen, preusmjerava se na formu za prijavu i ako nakon prijave ima ovlasti za tu rutu preusmjerava ga se na nju. Ako korisnik i dalje nema ovlasti ili nije bio prijavljen, preusmjerava ga se na početni zaslon aplikacije.
- Korisničko sučelje mora biti user friendly i jednostavno za korištenje
- Podaci koji se od korisnika prikupljaju i pohranjuju u bazu moraju biti zaštićeni i ograničenog pristupa

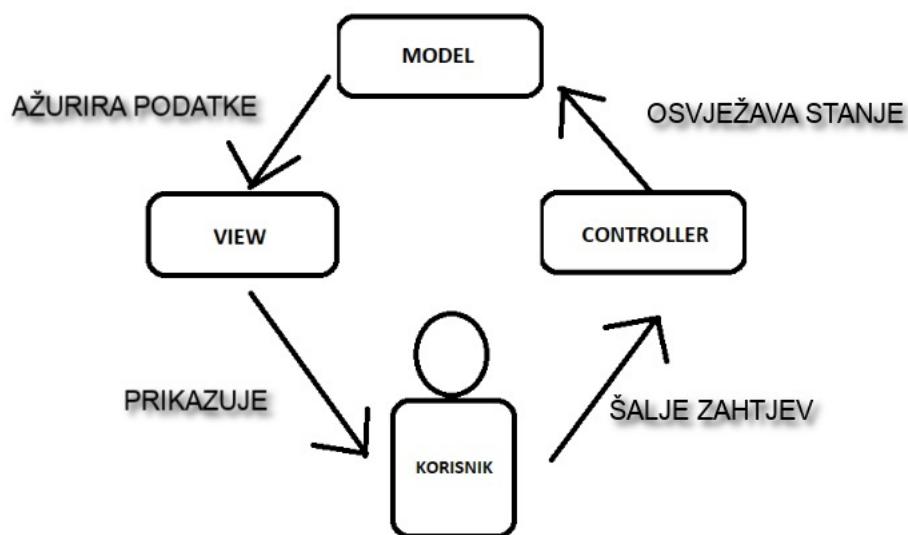
4. Arhitektura i dizajn sustava

Dijelovi arhitekture našeg sustava su web aplikacija, web poslužitelj i baza podataka.

Web preglednik služi kao posrednik između klijenta i internetske stranice kojoj se pristupa. On dohvata podatke te ih prevodi u korisniku razumljiv jezik. Web poslužitelj obrađuje zahtjeve klijenta i šalje odgovore. Komunikacija se odvija putem HTTP protokola. On pohranjuje i omogućuje pristup web stranicama. Baza podataka pohranjuje sve potrebne informacije o registriranim korisnicima te interne podatke o modelima aplikacije.

Sama aplikacija sastoji se od *frontend* i *backend* dijela koji se izvršavaju u *Node.js* okolini na cloud platformi.

Arhitektura sustava zasniva se na MVC (Model-View-Controller) načelu. Temeljno obilježje ovog obrasca jest podjela aplikacije u 3 međusobno povezane komponente:



Slika 4.1: MVC načelo

- **Model** je centralna komponenta sustava, sadrži podatke te razrede čijim se objektima modelira, .
- **View** predstavlja prikaz obrađenih podataka.
- **Controller** upravlja korisničkim zahtjevima te ih prosljeđuje modelu.

Na ovaj je način korisničko sučelje izdvojeno, čime je automatski smanjena njegova međuvisnost s ostatkom sustava. Upravitelj modelu šalje naredbe i osvježava njegovo stanje dok pogled od modela prikuplja informacije koje potom predstavlja korisniku.

Za razvoj web aplikacije, koristitimo programski jezik JavaScript na *frontendu*, te TypeScript na *backendu*. Za implementaciju *backenda* koristimo Node.js okruženjem, a u *frontend* dijelu JavaScript knjižnicom React. Za pohranu podataka u bazu, odlučili smo se za PostgreSQL te koristimo tehniku ORM (*Object-relational mapping*). Kao platformu uspostave aplikacije na udaljenom serveru, koristitimo cloud platformu [Heroku](#) koja podržava jezik Node.js.

Komunikacija aplikacije i klijenta na aplikacijskom sloju odvija se protokolom HTTP. Web aplikacija potom obrađuje njegov zahtjev te posreduje između njega i baze podataka služeći se [ORM sequelize](#) Node.js tehnologijom.

4.1 Baza podataka

Za naš sustav koristit ćemo relacijsku bazu podataka koja svojom strukturom olakšava modeliranje stvarnog svijeta. Gradivna jedinka baze je relacija, odnosno tablica koja je definirana svojim imenom i skupom atributa. Zadaća baze podataka je brza i jednostavna pohrana, izmjena i dohvata podataka za daljnju obradu.

Baza podataka ove aplikacije sastoji se od sljedećih entiteta:

- Račun
- Klijent
- Tvrta
- Vozilo
- Parkiralište
- Rezervacija
- Jednokratna
- Trajna
- Ponavljajuća

4.1.1 Opis tablica

Račun

Entitet sadrži informacije o napravljenom računu u aplikaciji. Sadrži atribute: idRacun, email, OIB, admin, lozinka. U vezi je *One-To-One* s entitetima Klijent i Tvrta.

| Račun | | |
|----------------|----------|----------------------------------|
| idRacun | INT | jedinstveni identifikator računa |
| email | VARCHAR | e-mail adresa računa |
| OIB | CHAR(11) | oib osobe čiji je račun |
| admin | BOOLEAN | true ako je osoba administrator |
| lozinka | VARCHAR | hash lozinke |

Klijent

Entitet sadrži informacije o klijentu koji koristi aplikaciju. Sadrži atribute: idKlijent, ime, prezime, broj kartice, idRacun. U vezi je *One-To-Many* s entitetima Vozilo i Rezervacija, a s entitetom Račun je u vezi *Many-To-One*.

| Klijent | | |
|------------------|---------|---|
| idKlijent | INT | jedinstveni identifikator klijenta |
| ime | VARCHAR | ime klijenta |
| prezime | VARCHAR | prezime klijenta |
| broj kartice | VARCHAR | broj kartice klijenta |
| <i>idRacun</i> | INT | jedinstveni identifikator računa (račun.ID) |

Tvrtka

Entitet sadrži informacije o tvrtki koja želi prijaviti svoje parkiralište u aplikaciju. Sadrži atribute: idTvrtka, naziv, adresa, idRacun. U vezi je *One-To-Many* s entitetom Parkiralište, a s entitetom Račun je u vezi *Many-To-One*.

| Tvrtka | | |
|-----------------|---------|---|
| idTvrtka | INT | jedinstveni identifikator tvrtke |
| naziv | VARCHAR | naziv tvrtke |
| adresa | VARCHAR | adresa sjedišta tvrtke |
| <i>idRacun</i> | INT | jedinstveni identifikator računa (račun.ID) |

Vozilo

Entitet sadrži informacije o vozilo kojeg je klijent prijavio. Sadrži atribute: idVozilo, registracija, naziv vozila, boja, i idKlijent. U vezi je *One-To-Many* s entitetom Rezervacija, a s entitetom Klijent je u vezi *Many-To-One*.

| Vozilo | | |
|------------------|---------|---|
| idVozilo | INT | jedinstveni identifikator vozila |
| registracija | VARCHAR | broj registracije vozila |
| naziv vozila | VARCHAR | naziv vozila koje dodjeljuje klijent |
| boja | VARCHAR | boja vozila koju dodjeljuje klijent |
| <i>idKlijent</i> | INT | jedinstveni identifikator klijenta (klijent.ID) |

Parkiralište

Entitet sadrži informacije o parkiralištu neke tvrtke koje se nudi klijentima. Sadrži atribute: idParkiraliste, naziv, broj mesta, broj invalidskih mesta, tip parkirališta, koordinate, cijena jednokratne, cijena ponavljuće, cijena trajne i idTvrтka. U vezi je *One-To-Many* s entitetom Rezervacija, a s entitetom Tvrтka je u vezi *Many-To-One*.

| Parkiralište | | |
|------------------------|---------|---|
| idParkiraliste | INT | jedinstveni identifikator parkirališta |
| naziv | VARCHAR | naziv parkirališta |
| broj mesta | INT | broj mesta koje parkiralište nudi |
| broj invalidskih mesta | INT | broj invalidskih mesta koje ima parkiralište |
| tip parkirališta | VARCHAR | tip parkirališta (otvoreno, zatvoren) |
| koordinate | VARCHAR | zemljopisna dužina i širina parkirališta |
| cijena jednokratne | DECIMAL | cijena sata za jednokratnu rezervaciju parkirališta |
| cijena ponavljuće | DECIMAL | cijena sata za ponavljuće rezervaciju parkirališta |
| cijena trajne | DECIMAL | cijena trajne rezervacije parkirališta |
| <i>idTvrтka</i> | INT | jedinstveni identifikator tvrtke (tvrтka.ID) |

Rezervacija

Entitet sadrži informacije o stvorenoj rezervaciji u aplikaciji. Ima atribute: idRezervacija, idKlijent, idParkiraliste, idVozilo. U vezi je *One-To-Many* s entitetima Jednokratna, Ponavljuća i Trajna. S entitetima Klijent, Parkiralište i Vozilo je u vezi *Many-To-One*.

| Rezervacija | | |
|-----------------------|-----|--|
| idRezervacija | INT | jedinstveni identifikator rezervacije |
| <i>idKlijent</i> | INT | jedinstveni identifikator klijenta (klijent.ID) |
| <i>idParkiraliste</i> | INT | jedinstveni identifikator parkirališta (parkiralište.ID) |
| <i>idVozilo</i> | INT | jedinstveni identifikator vozila (vozilo.ID) |

Jednokratna

Entitet sadrži informacije o jednokratnoj rezervaciji stvorenoj u aplikaciji. Ima atribute: idJednokratna, vrijeme početak, vrijeme kraj, idRezervacija. U vezi je *Many-To-One* s entitetom Rezervacija.

| Jednokratna | | |
|----------------------|-----------|--|
| idJednokratna | INT | jedinstveni identifikator jednokratne rezervacije |
| vrijeme početak | TIMESTAMP | vrijeme početka rezervacije |
| vrijeme kraj | TIMESTAMP | vrijeme kraja rezervacije |
| <i>idRezervacija</i> | INT | jedinstveni identifikator rezervacije (rezervacija.ID) |

Trajna

Entitet sadrži informacije o trajnoj rezervaciji stvorenoj u aplikaciji. Ima atribute: idTrajna, vrijeme početak, vrijeme kraj, idRezervacija. U vezi je *Many-To-One* s entitetom Rezervacija.

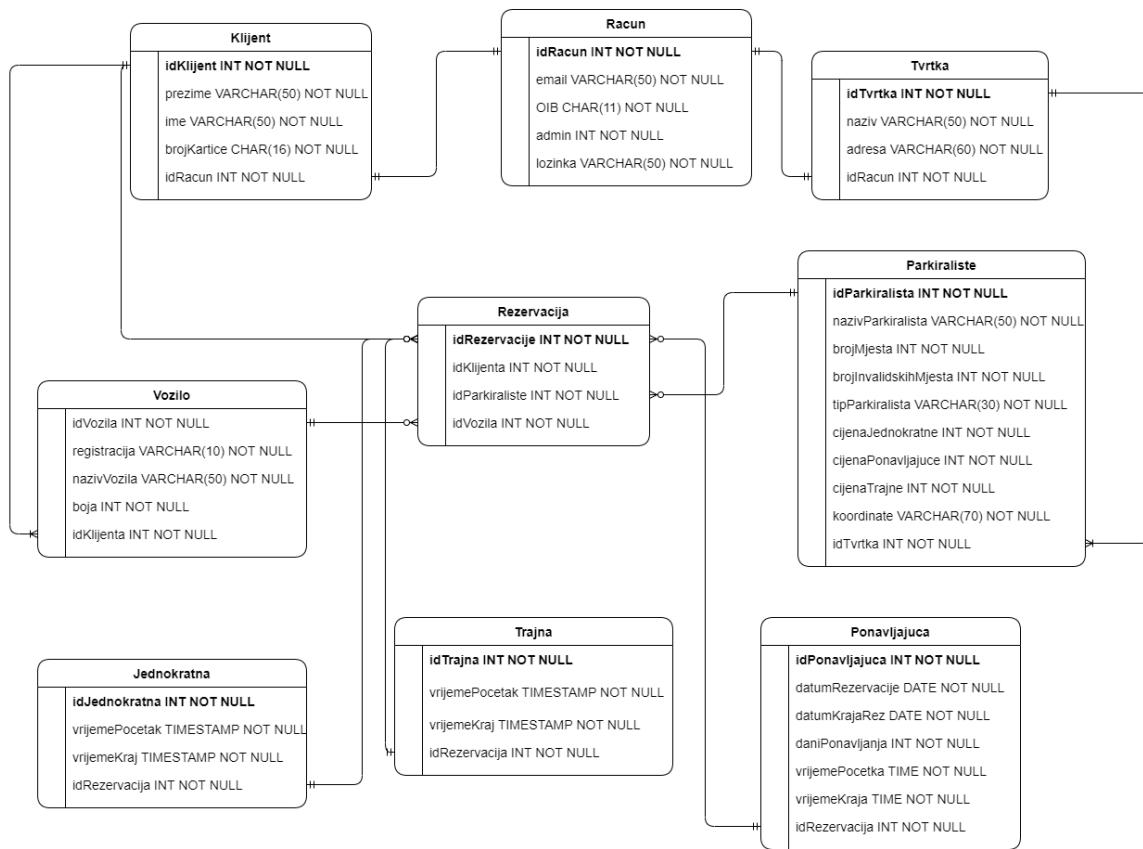
| Trajna | | |
|----------------------|-----------|--|
| idTrajna | INT | jedinstveni identifikator trajne rezervacije |
| vrijeme početak | TIMESTAMP | vrijeme početka rezervacije |
| vrijeme kraj | TIMESTAMP | vrijeme kraja rezervacije |
| <i>idRezervacija</i> | INT | jedinstveni identifikator rezervacije (rezervacija.ID) |

Ponavljača

Entitet sadrži informacije o ponavljajućoj rezervaciji stvorenoj u aplikaciji. Ima atribute: idPonavljača, datum rezervacije, datum kraja rezervacije, dani ponavljanja, vrijeme početak, vrijeme kraj i idRezervacija. U vezi je *Many-To-One* s entitetom Rezervacija.

| Ponavljača | | |
|-------------------------|------|--|
| idPonavljača | INT | jedinstveni identifikator ponavljajuće rezervacije |
| datum rezervacije | DATE | datum početka rezervacije |
| datum kraja rezervacije | DATE | datum kraja rezervacije |
| dani ponavljanja | INT | dani ponavljanje rezervacije (pon=1, uto=2, ...) |
| vrijeme početak | TIME | vrijeme početka rezervacije |
| vrijeme kraj | TIME | vrijeme kraja rezervacije |
| <i>idRezervacija</i> | INT | jedinstveni identifikator rezervacije (rezervacija.ID) |

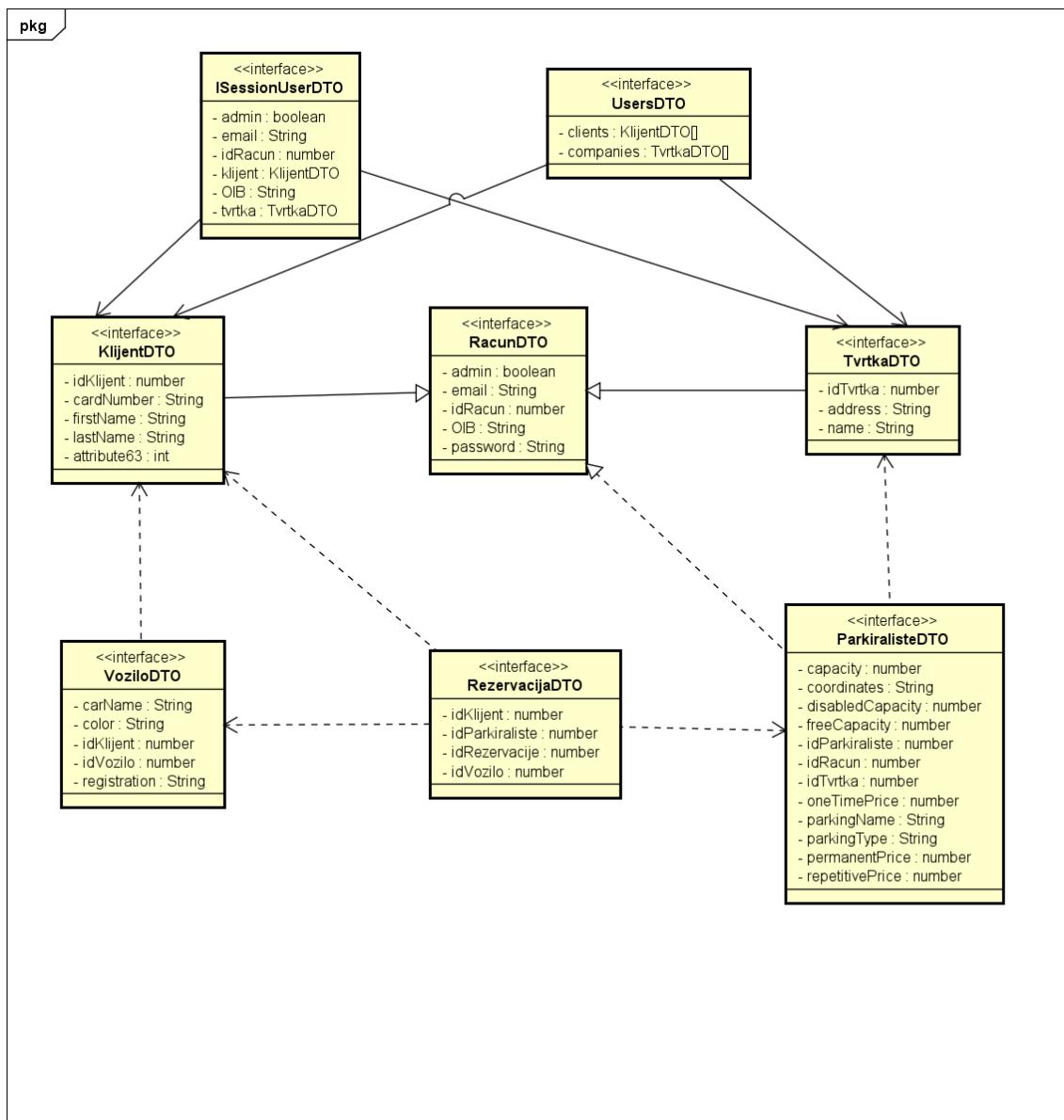
4.1.2 Dijagram baze podataka



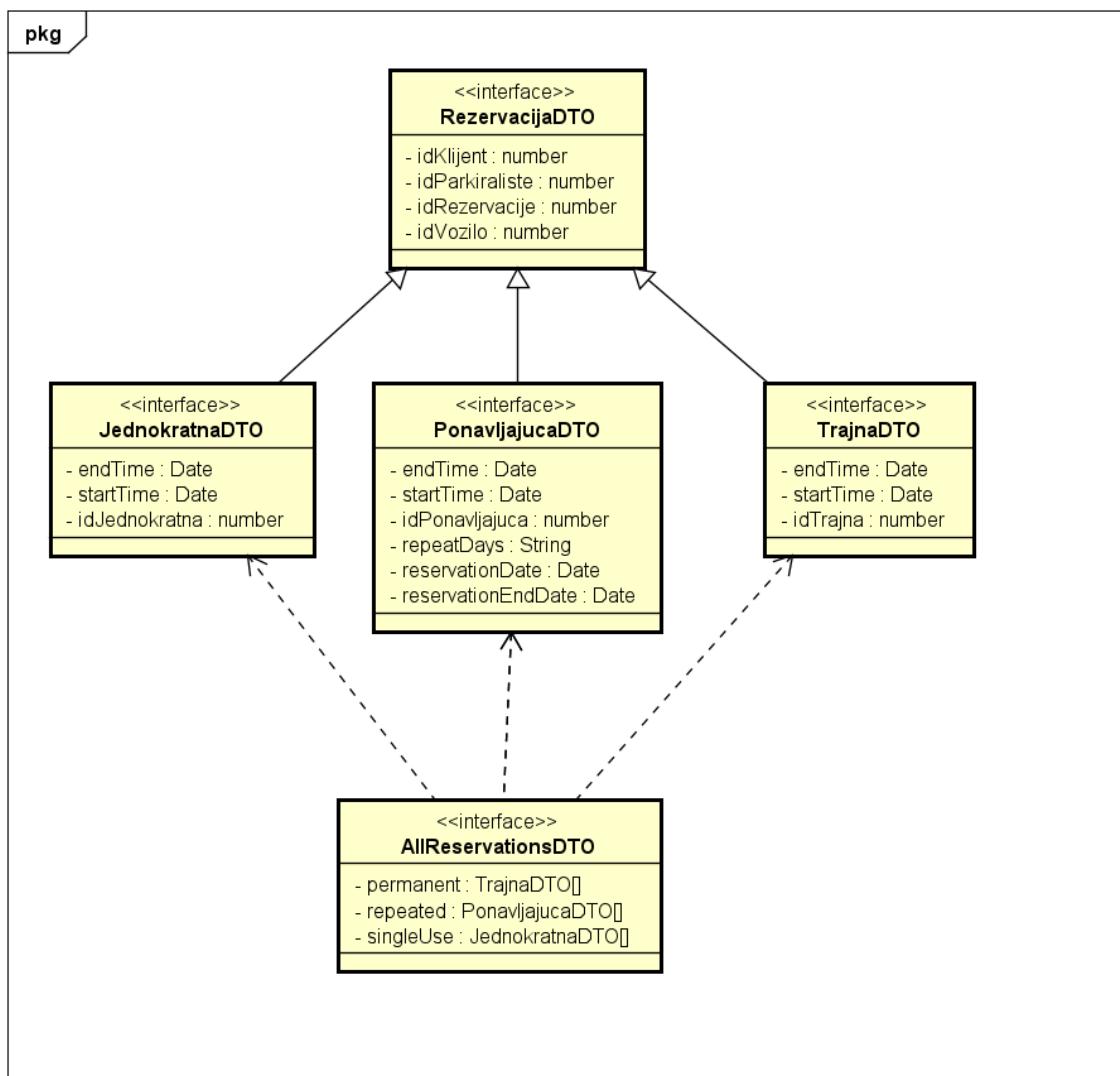
Slika 4.2: Dijagram baze podataka

4.2 Dijagram razreda

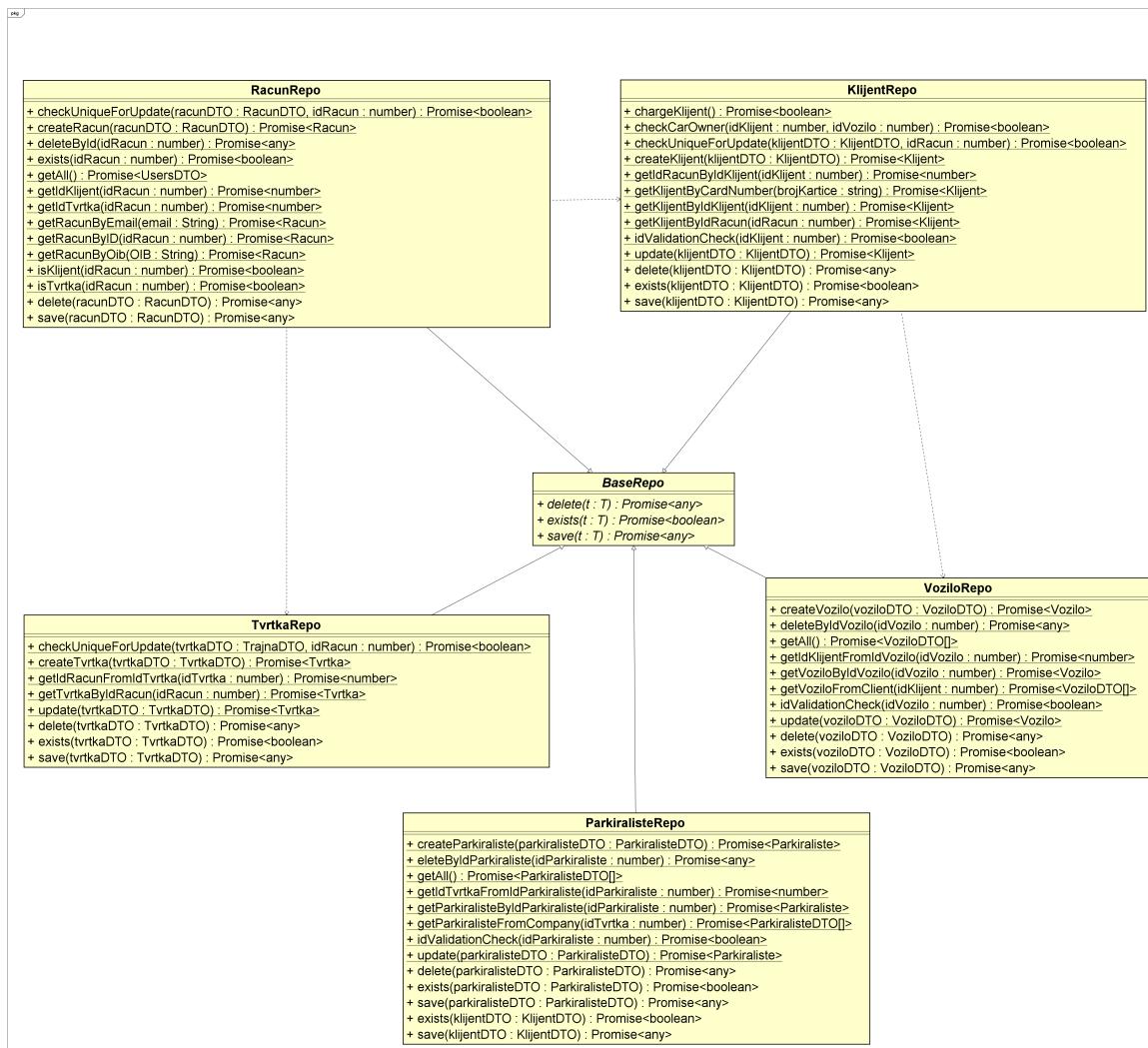
Na slikama 4.3, 4.4, 4.5, 4.6, 4.7 prikazani su razredi koji pripadaju *backend* dijelu MVC arhitekture. Razredi na slici 4.3 oponašaju tablice iz baze podataka kojima se pristupa preko ORM-a i pomoću njih se komunicira s bazom podataka. Slika 4.4 prikazuje DTO (*Data transfer object*) razrede koji spremaju podatke kojima kontroleri komuniciraju s *frontend* dijelom aplikacije i modelima. Na slici 4.5 nalaze se razredi koji komuniciraju s bazom podataka i izvršavaju upite koje kontroleri zatraže. Kontroleri su prikazani na slici 4.6. Svi kontrolери nasljeđuju osnovni kontroler koji sadrži metode za slanje odgovora, dok ostali kontroleri nude metode za izvršavanje određenog zahtjeva na određenoj putanji. Slika 4.7 prikazuje jednu vrstu pomoćnih razreda koje smo koristili za prebacivanje podataka iz domene modela u domenu koja se koristi na *frontend* dijelu aplikacije.



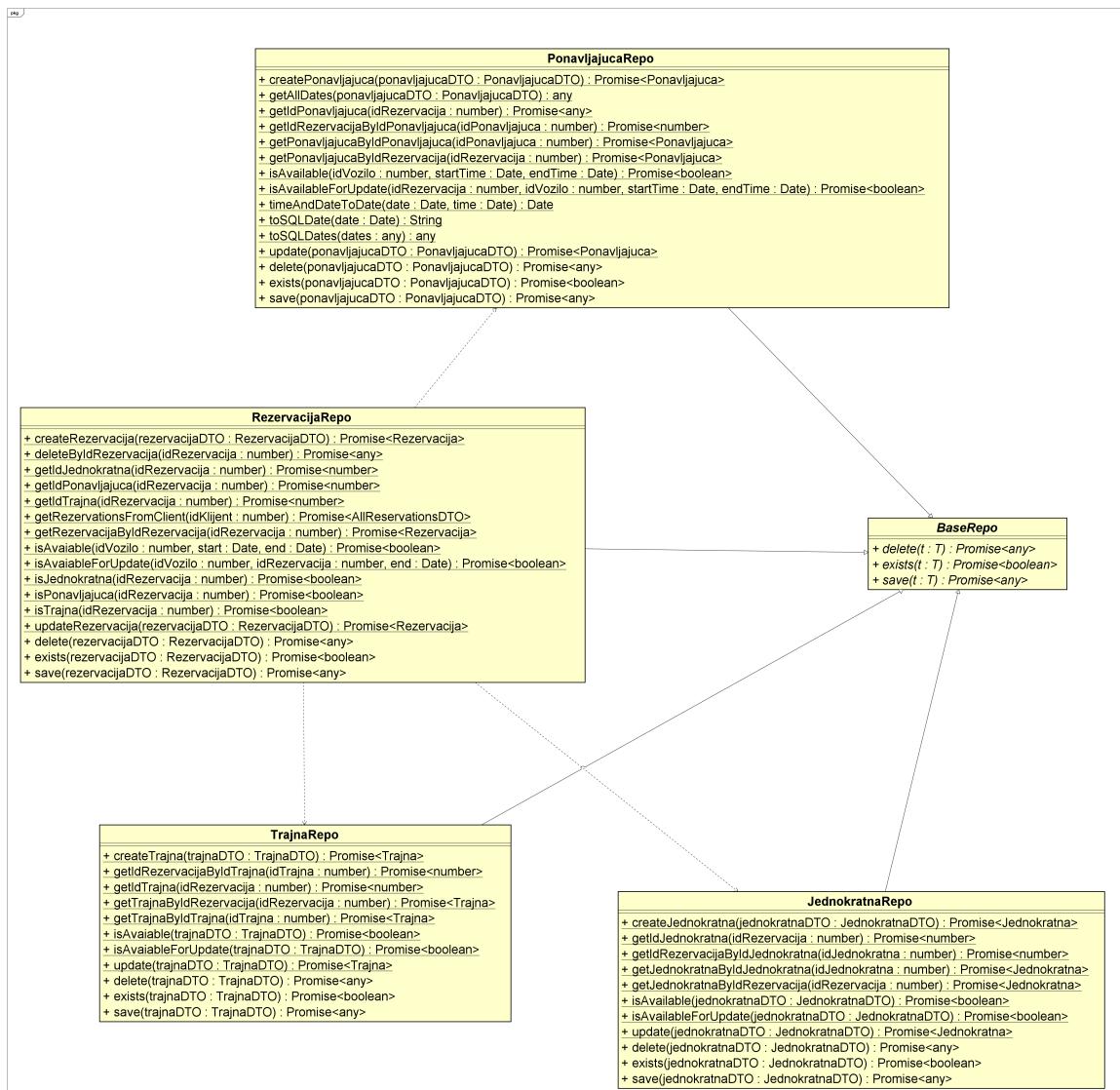
Slika 4.3: Dijagram razreda za DTO (1.dio)



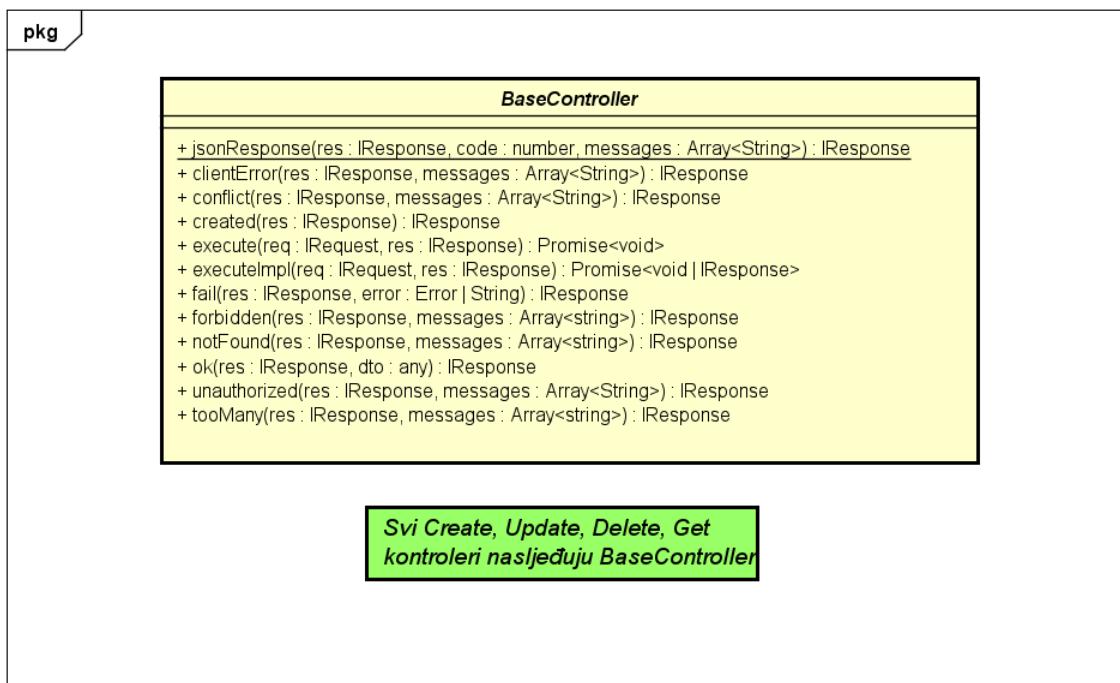
Slika 4.4: Dijagram razreda za DTO (2.dio)



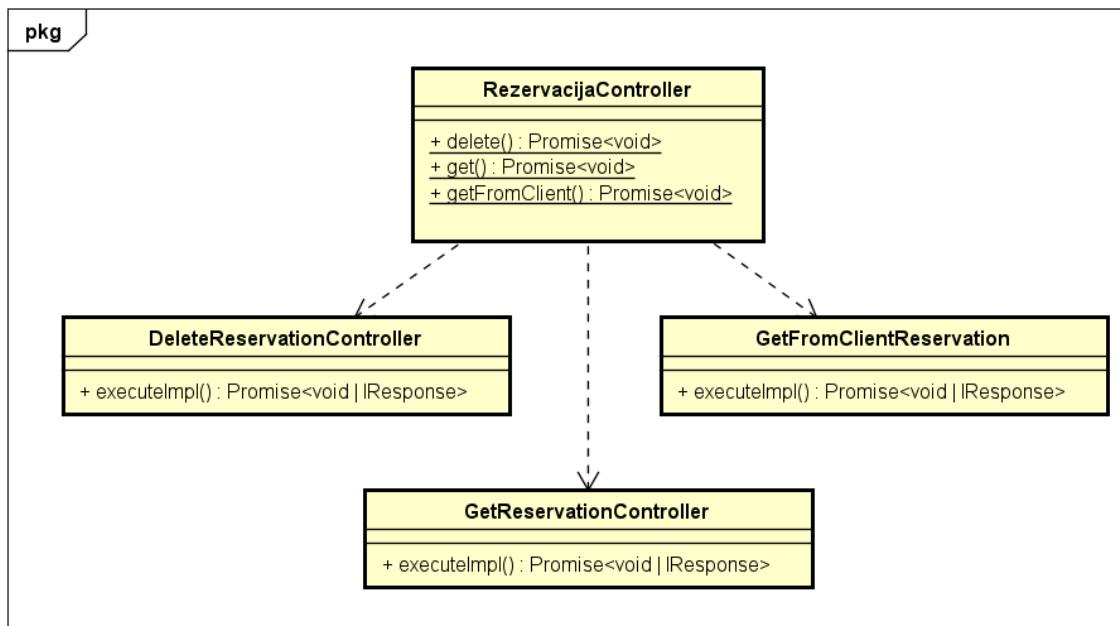
Slika 4.5: Dijagram razreda za repo (1.dio)



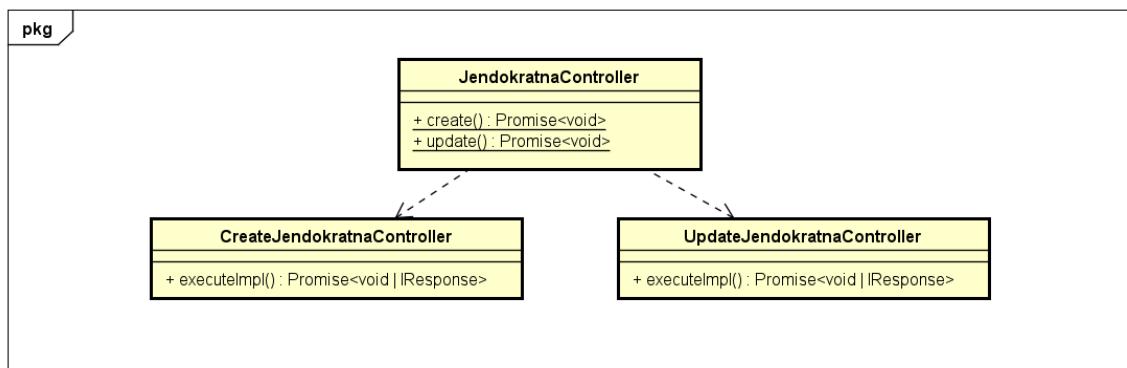
Slika 4.6: Dijagram razreda za repo (2.dio)



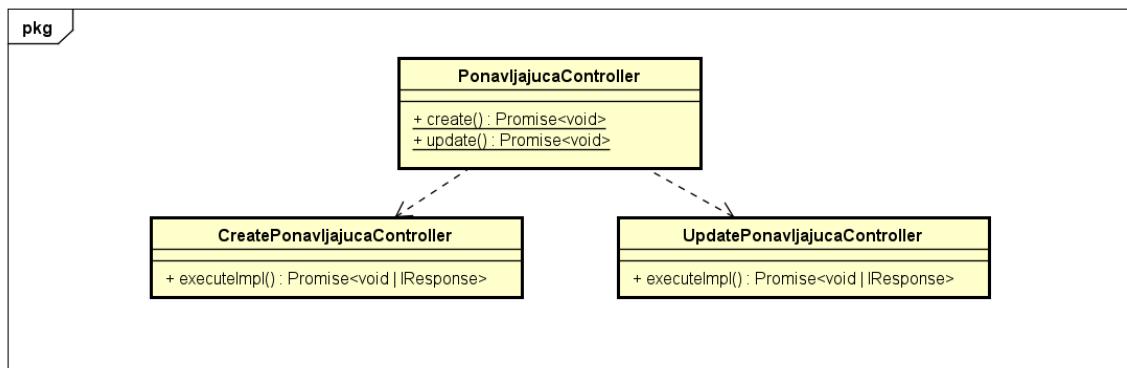
Slika 4.7: Dijagram razreda za BaseController



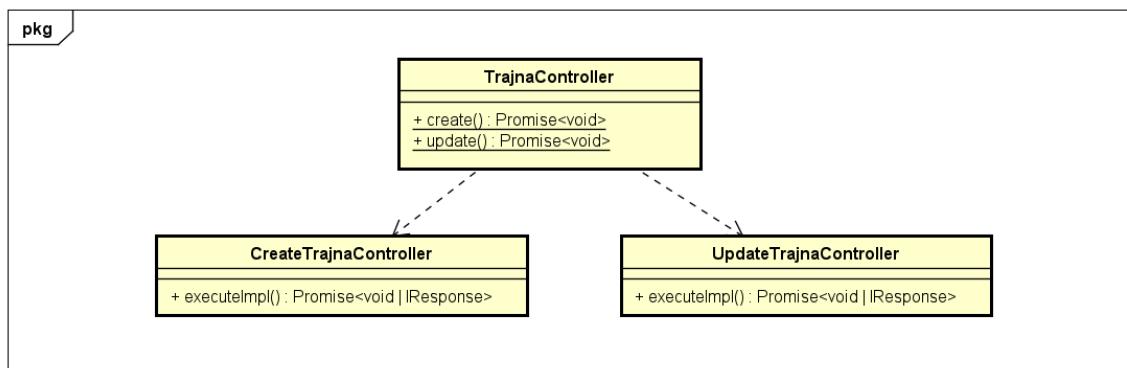
Slika 4.8: Dijagram razreda za Rezervacija Controller



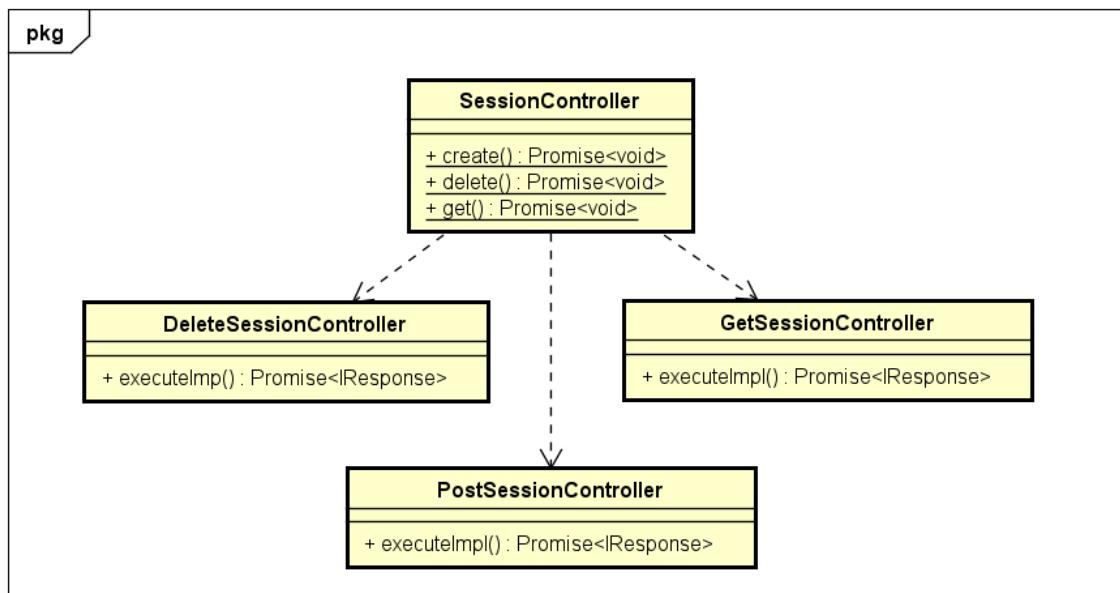
Slika 4.9: Dijagram razreda za Jednodokratna Controller



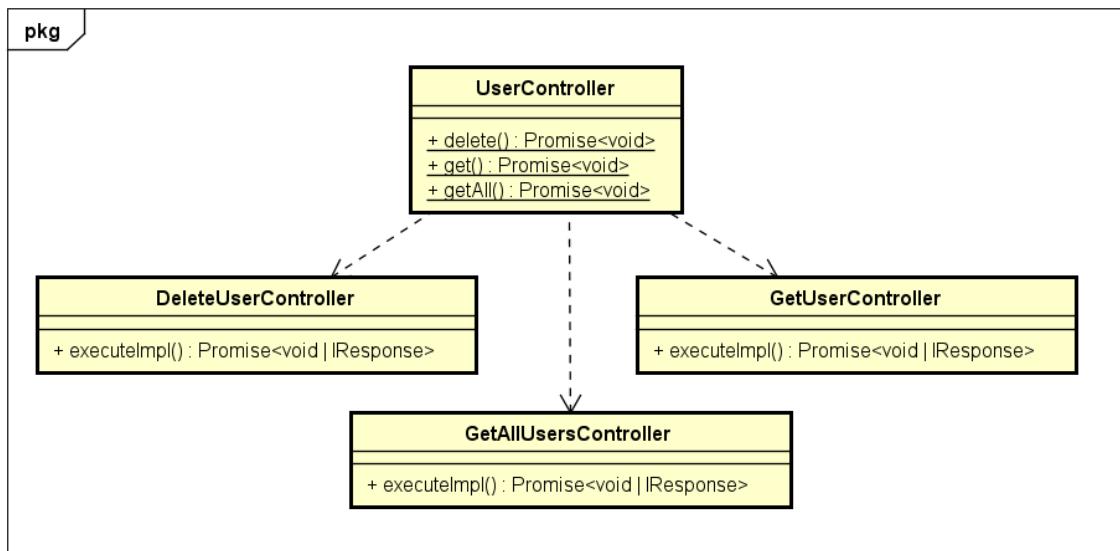
Slika 4.10: Dijagram razreda za Ponavljajuca Controller



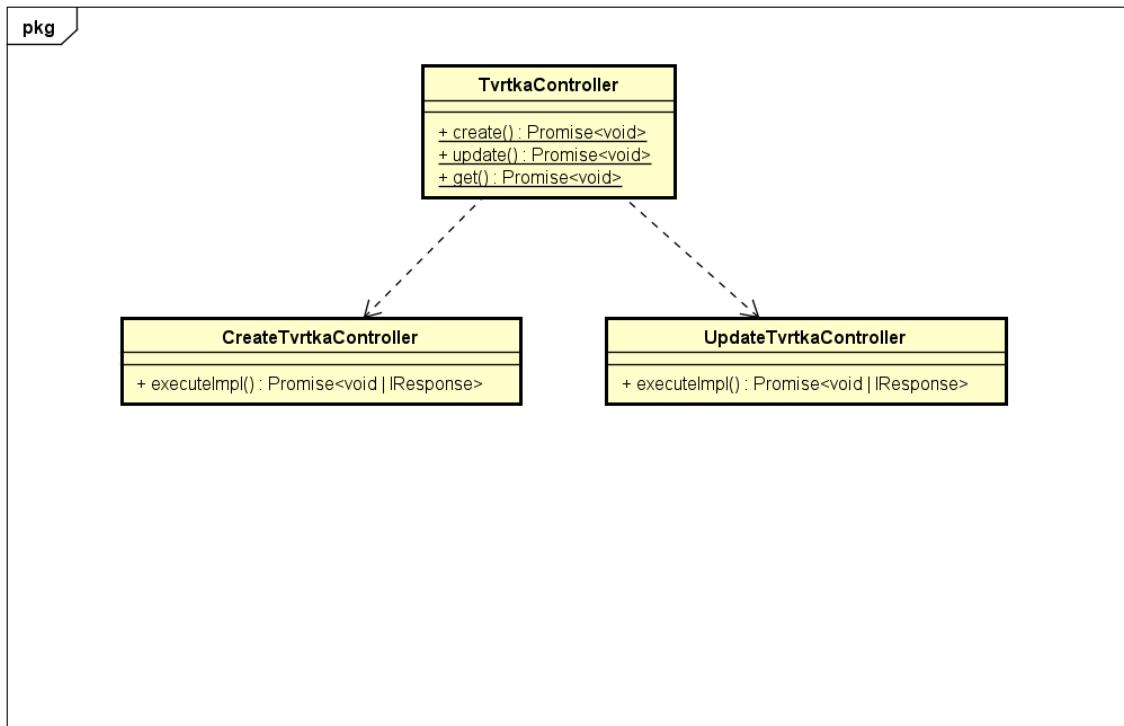
Slika 4.11: Dijagram razreda za Trajna Controller



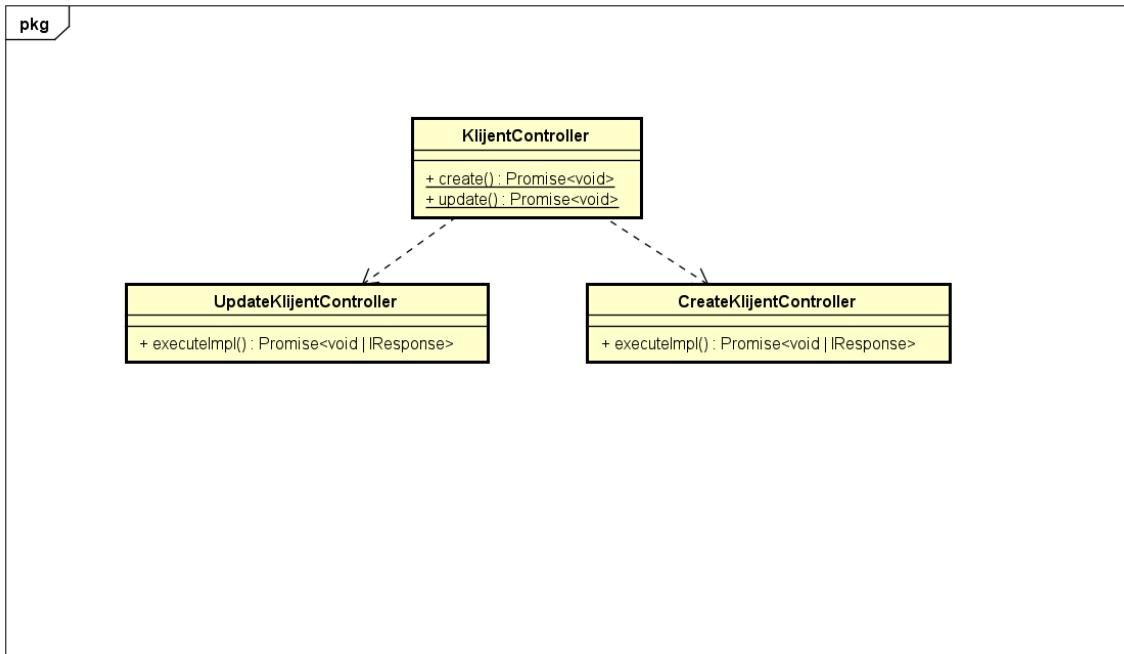
Slika 4.12: Dijagram razreda za Session Controller



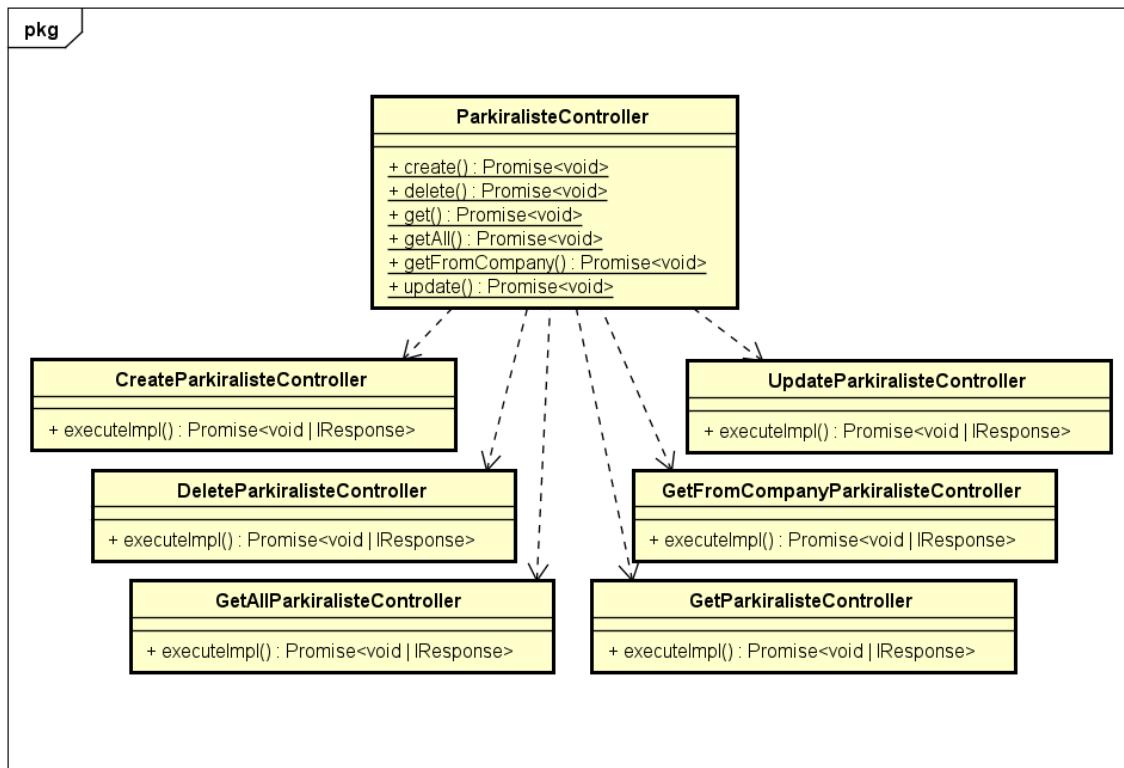
Slika 4.13: Dijagram razreda za User Controller



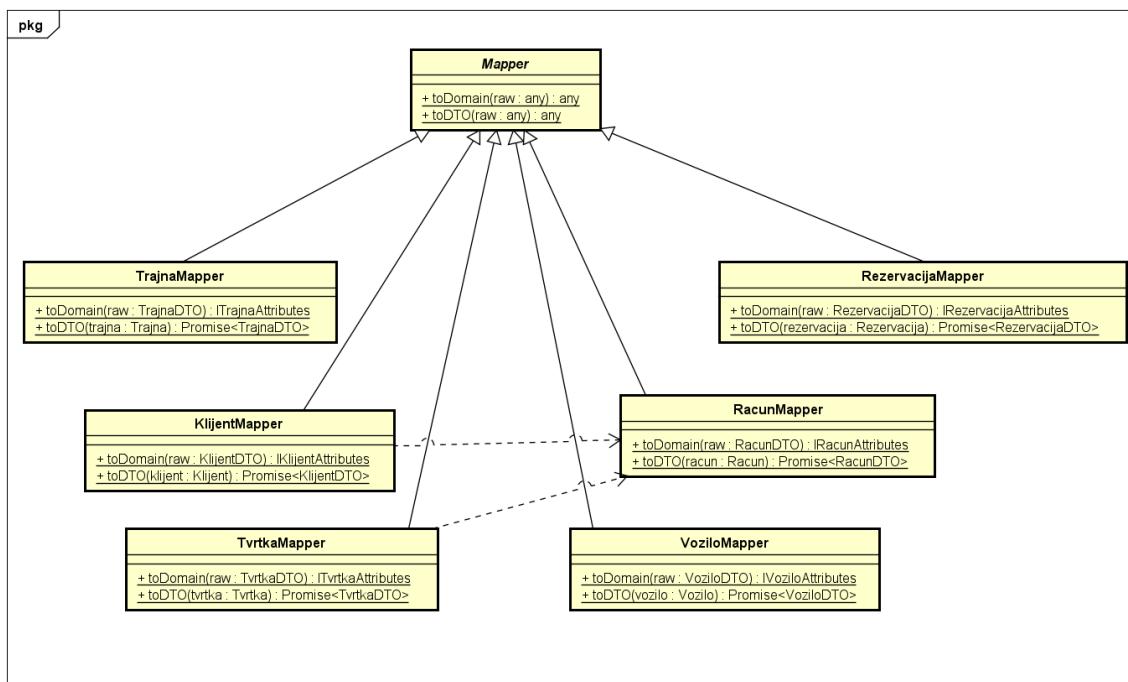
Slika 4.14: Dijagram razreda za Tvrta Controller



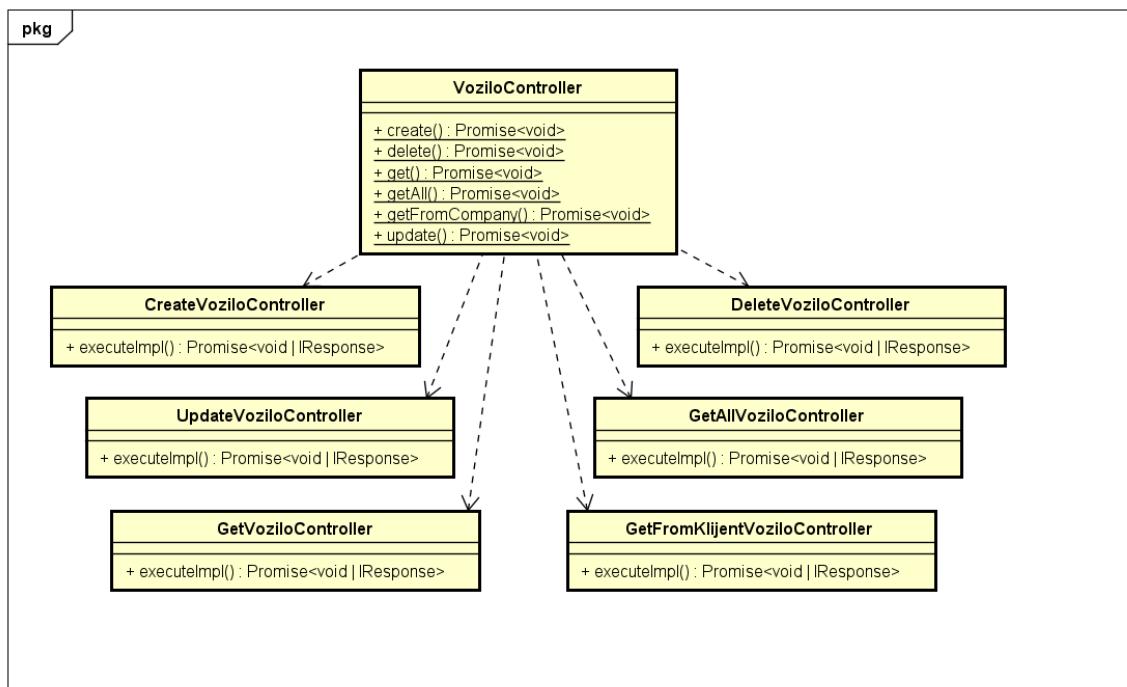
Slika 4.15: Dijagram razreda za Klijent Controller



Slika 4.16: Dijagram razreda za Parkiraliste Controller



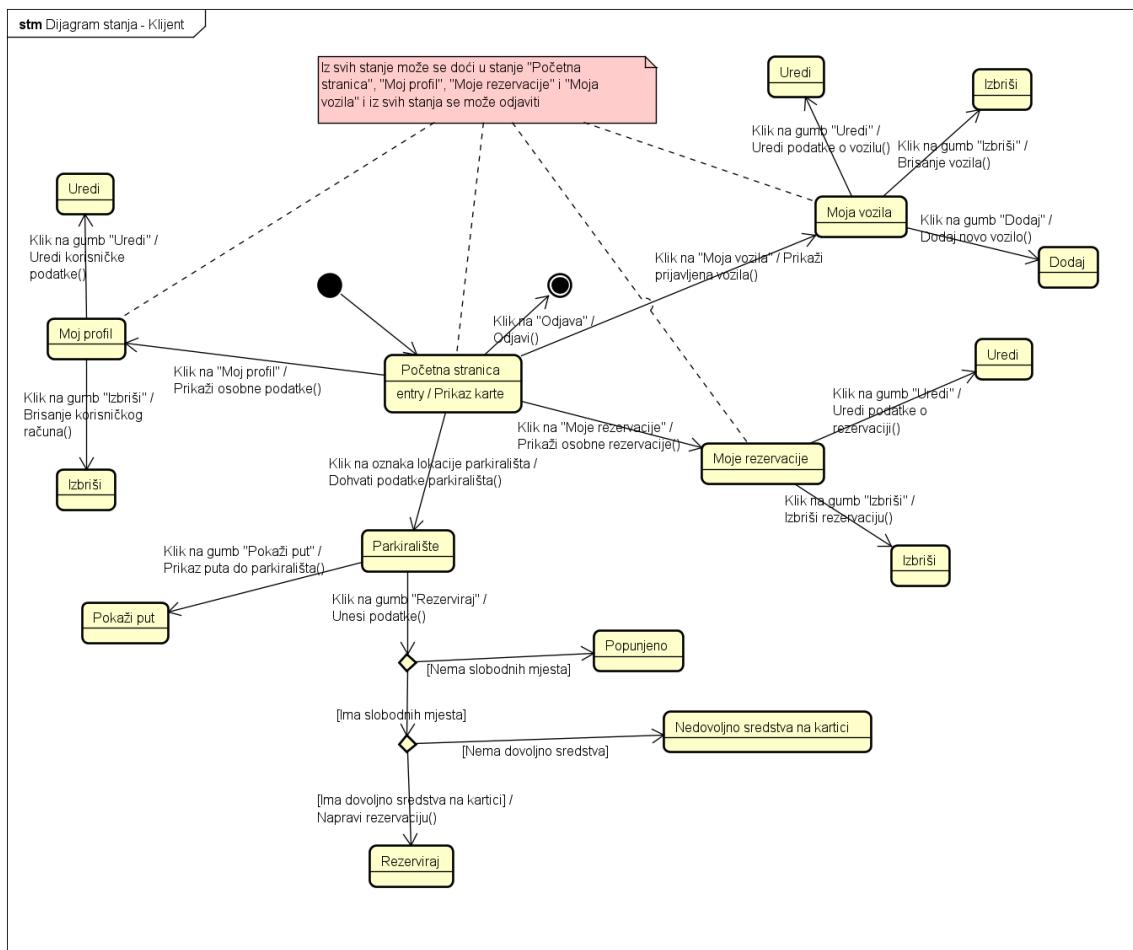
Slika 4.17: Dijagram razreda za Mappers



Slika 4.18: Dijagram razreda za Vozilo Controller

4.3 Dijagram stanja

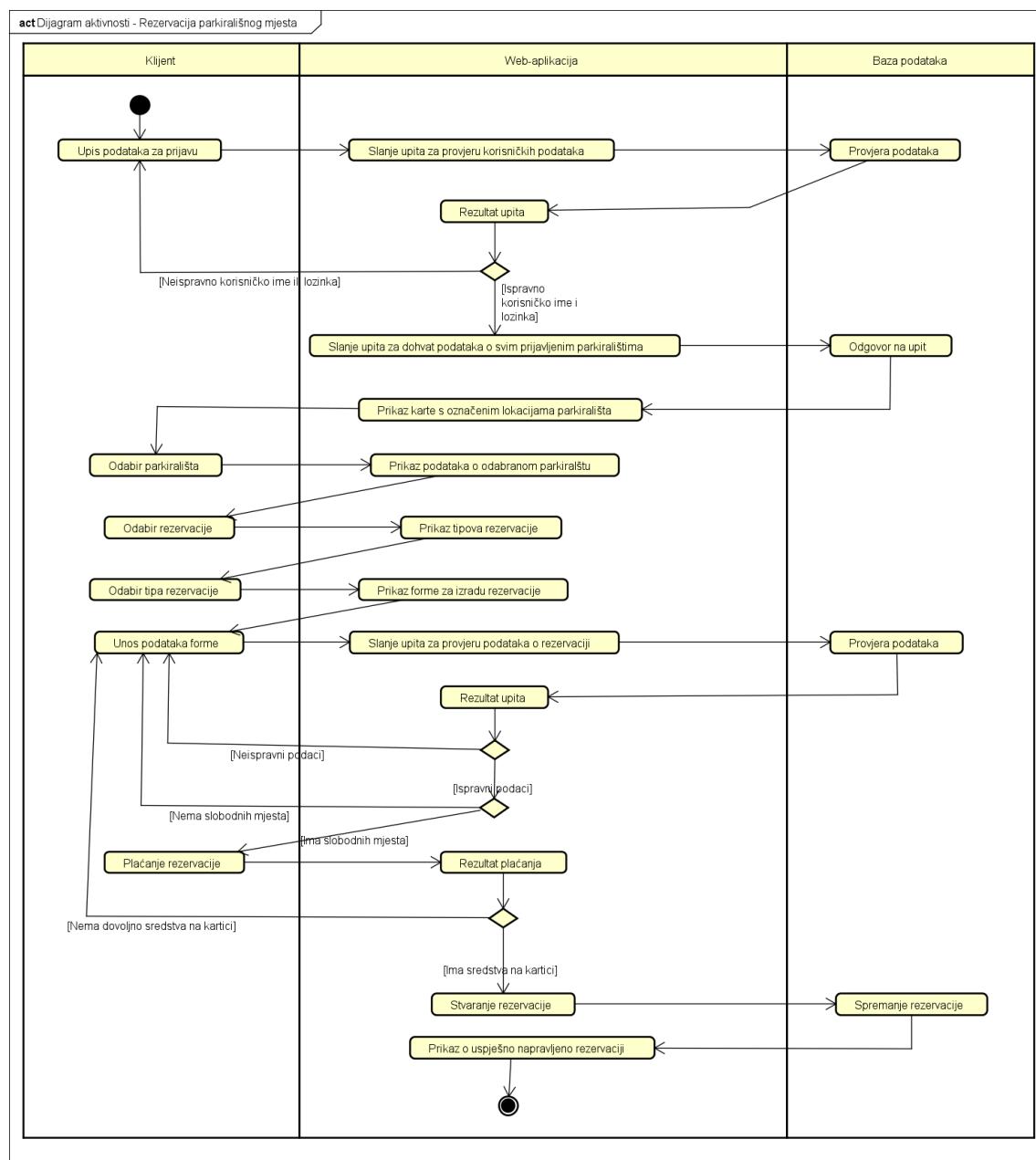
Dijagram stanja pokazuje stanje objekata te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici 4.8 prikazan je dijagram stanja za klijenta. Nakon prijave, klijentu se prikazuje karta na kojoj su prikazana sva prijavljena parkirališta. Sa strane se nalazi izbornik u kojem može odabrati neke mogućnosti. Odabirom na "Moj profil" klijentu se prikazuju korisnički podaci koje može urediti ili može obrisati račun. Odabirom na "Moje rezervacije" otvara se popis vlastitih rezervacija koje klijent može urediti ili obrisati. Odabirom na "Moja vozila" otvara se popis osobnih prijavljenih vozila, klijent može urediti podatke o vozilu, može izbrisati vozilo ili dodati novo. Odabirom oznake parkirališta na karti klijentu se nudi opcija "Pokaži put" koja omogućava navigaciju do odabranom parkiralištu i "Rezerviraj" koja omogućava stvaranje rezervacije parkirališnog mesta.



Slika 4.19: Dijagram stanja

4.4 Dijagram aktivnosti

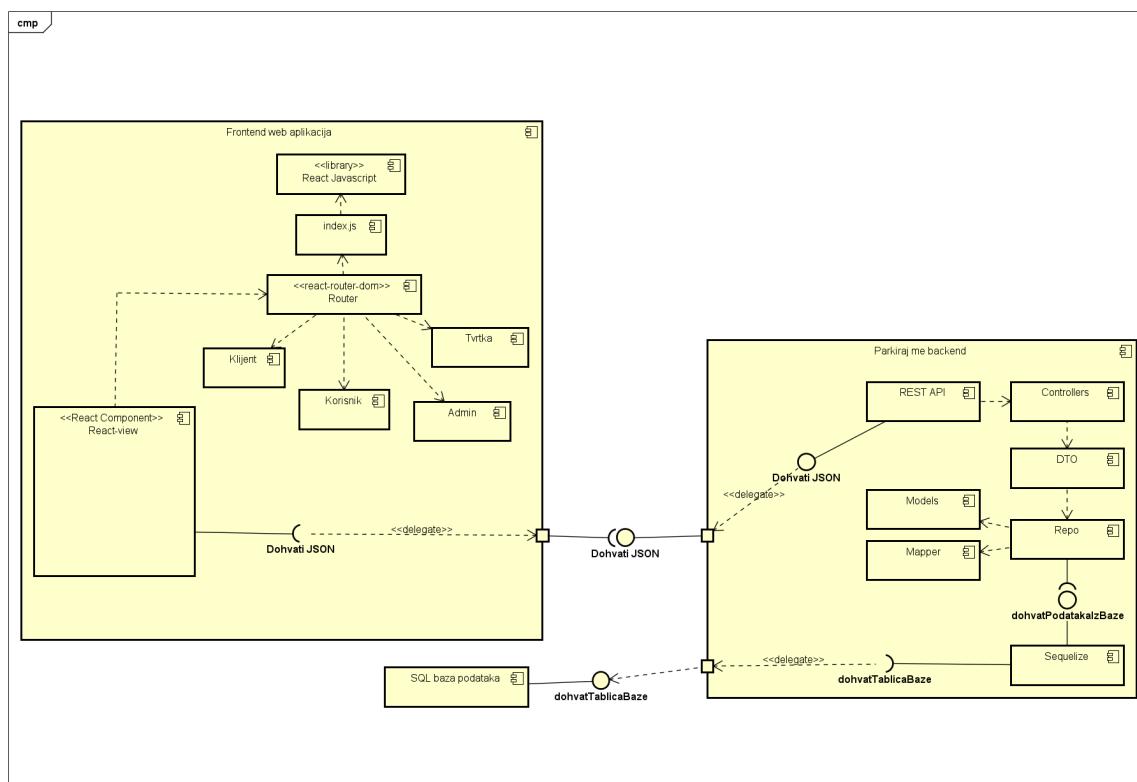
Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. U modeliranju toka upravljanja novi korak izvršava se nakon završenog prethodnog. Na slici 4.9 prikazan je dijagram aktivnosti rezervacije parkirališnog mjestra. Klijent se prijavi u sustav, bira parkiralište na kojem želi napraviti rezervaciju, nakon čega bira tip rezervacije. Prikazuje mu se forma za unos podataka o rezervaciji koju klijent ispunjava te odabire opciju dodavanja rezervacije nakon čega slijedi plaćanje. Ukoliko su svi podaci ispravni, ima slobodnih mesta i plaćanje je uspješno rezervacija se dodaje u sustav.



Slika 4.20: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti je strukturni, statički dijagram koji služi za vizualizaciju organizacije i međuvisnosti između implementacijskih komponenata te odnos programske potpore prema okolini. Sve Javascript datoteke u frontend dijelu aplikacije ovise o React knjižnici iz koje dohvaćaju gotove komponente poput gumba, izbornika i svega ostalog čime netko komunicira s aplikacijom. Router je komponenta koja na upit iz url-a određuje koja će se datoteka poslužiti na aplikaciji. Frontend dio aplikacije sastoji se od Javascript datoteka koje su logički raspoređene u cjeline te su nazvane po tipovima aktora i njihovim akcijama. REST API poslužuje podatke koji pripadaju backend dijelu aplikacije. Sequelize je zadužen za dohvaćanje tablica iz baze podataka. Za rad s bazom backend dio aplikacije koristi Repo koji je povezan Modelima i Mapperima. Model je apstrakcija koja predstavlja tablice u bazi, a Mapper ovisno o zahtjevu pretvara podatke u potrebnii oblik. Podaci koji su pristigli iz baze šalju se u obliku DTO-a MVC arhitekturi. Controller prima zahtjev, obrađuje ga i šalje dalje.



Slika 4.21: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije WhatsApp¹ za manje i kraće dogovore, a za dulje i veće dogovore komunikacija je realizirana korištenjem aplikacije Microsoft Teams². Za izradu UML dijagrama korišten je alat Astah UML³. Kao sustav za upravljanje izvornim kodom Git⁴. Udaljeni rezitorij projekta je dostupan na web platformi GitLab⁵.

Kao razvojno okruženje korišten je Microsoft Visual Studio Code⁶. Microsoft Visual Studio Code trenutno je najpopularnije⁷ razvojno okruženje, iznimno je prilagodljiv raznim potrebama programera i potrebama samog programskega jezika.

Aplikacija je izrađena u dva dijela. Backend dio je napisan koristeći radni okvir Express.js⁸ u programskom jeziku TypeScript⁹, a frontend dio je napisan koristeći biblioteku React.js¹⁰ u programskom jeziku JavaScript¹¹.

Express.js radni okvir je javno dostupan, besplatan te otvorenog izvora pružen od strane OpenJS Foundation-a¹². Express.js je jednostavna i prilagodljiv Node.js¹³ radni okvir koja pruža izdržljiv skup značajki za web i mobilne aplikacije.

React.js je JavaScript biblioteka za izradu interaktivnih sučelja unutar web aplikacija. Održavana je od strane Facebooka, a najčešće je korišten kao osnova u razvoju web aplikacija.

¹Whatsapp

²Microsoft Teams

³Astah UML

⁴Git

⁵Gitlab

⁶Microsoft Visual Studio Code

⁷Deset najraširenijih razvojnih okruženja

⁸Express.js

⁹TypeScript

¹⁰React.js

¹¹JavaScript

¹²OpenJS Foundation

¹³Node.js

Korištena baza podataka je PostgreSQL¹⁴. PostgreSQL je besplatna, objektno-relacijska baza podataka sa više od 30 godina aktivnog razvoja.

Cijela aplikacija poslužena je na Heroku¹⁵. Heroku je platforma koja omogućuje jednostavno i sigurno puštanje u pogon i nadziranje aplikacija. Konkretno, koristimo dvije Heroku aplikacije, jednu za backend i jednu za frontend te u sklopu backend aplikacije koristimo i jednu instancu PostgreSQL baze podataka.

¹⁴PostgreSQL

¹⁵Heroku

5.2 Ispitivanje programskog rješenja

Ispitivanje aplikacije proveli smo pomoću testnih radnih okvira Jest¹⁶ za ispitivanje jedinica i Selenium¹⁷ za ispitivanje sustava.

5.2.1 Ispitivanje komponenti

Ispitivanje komponenti provedeno je pomoću Jest¹⁸ testnog radnog okvira. Ispitivali smo metode na *backend* dijelu aplikacije. Očekivani izlaz usporedili smo s povratnim vrijednostima metoda koje testiramo. Svi su testovi prošli.

```
> jest
  PASS tests/mapper.test.ts
  PASS tests/racunValidator.test.ts
  PASS tests/times.test.ts (5.157 s)

Test Suites: 3 passed, 3 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        6.97 s
Ran all test suites.
```

Slika 5.1: Unit testovi

¹⁶Jest

¹⁷www.selenium.dev

¹⁸Jest

```
ts mapper.test.ts x
backend > tests > ts mapper.test.ts > ...
11   color: 'blue',
12   idKlijent: 2,
13   idVozilo: 1,
14 } as VoziloDTO;
15 expect(VoziloMapper.toDomain(voziloDTO)).toStrictEqual({
16   idVozilo: 1,
17   registracija: 'zg8787hn',
18   nazivVozila: 'ford',
19   boja: 'blue',
20   idKlijent: 2,
21 });
22 });
23
24 it('Provjera 2 - Klijent toDomain', () => {
25   const klijentDTO = {
26     idKlijent: 2,
27     firstName: 'Ana',
28     lastName: 'Maric',
29     carNumber: '1298656293745610',
30     idRacun: 3,
31     OIB: '12345678910',
32     admin: false,
33     password: 'password',
34   } as KlijentDTO;
35   expect(KlijentMapper.toDomain(klijentDTO)).toStrictEqual({
36     idKlijent: 2,
37     brojKartice: '1298656293745610',
38     ime: 'Ana',
39     prezime: 'Maric',
40     idRacun: 3,
41   });
42 });
43 });
44
```

Slika 5.2: Kod za test mappera 1

```
TS mapper.test.ts ×

backend > tests > TS mapper.test.ts > ...
1  import { VoziloMapper } from '../mappers/VoziloMapper';
2  import { VoziloDTO } from '../dtos/VoziloDTO';
3  import { KlijentMapper } from '../mappers/KlijentMapper';
4  import { KlijentDTO } from '../dtos/KlijentDTO';
5
6  describe('Provjera funkcija u VoziloMapper', () => {
7    it('Provjera 1 - Vozilo toDomain', () => {
8      const voziloDTO = {
9        registration: 'zg8787hn',
10       carName: 'ford',
11       color: 'blue',
12       idKlijent: 2,
13       idVozilo: 1,
14     } as VoziloDTO;
15     expect(VoziloMapper.toDomain(voziloDTO)).toStrictEqual({
16       idVozilo: 1,
17       registracija: 'zg8787hn',
18       nazivVozila: 'ford',
19       boja: 'blue',
20       idKlijent: 2,
21     });
22   });
23
24  it('Provjera 2 - Klijent toDomain', () => {
25    const klijentDTO = {
26      idKlijent: 2,
27      firstName: 'Ana',
28      lastName: 'Maric',
29      cardNumber: '1298656293745610',
30      idRacun: 3,
31      OIB: '12345678910',
32      admin: false,
33      password: 'password',
34    } as KlijentDTO;
35    expect(KlijentMapper.toDomain(klijentDTO)).toStrictEqual({
36      idKlijent: 2,
37      brojKartice: '1298656293745610',
38      ime: 'Ana'
```

Slika 5.3: Kod za test mappera 2

```

TS racunValidator.test.ts ✘
backend > tests > TS racunValidator.test.ts > ...
40   expect(actual).toStrictEqual(['OIB mora imati samo brojeve']);
41 });
42
43 it('Provjera 4 - Incorrect OIB size', async () => {
44   const klijentDTO = {
45     firstName: 'Però',
46     lastName: 'Peric',
47     cardNumber: '0012312342325623',
48     email: 'peric@mail.com',
49     OIB: '1234568912',
50     password: 'lozinka',
51   } as KlijentDTO;
52   const actual = await RacunValidator.validate(klijentDTO);
53   expect(actual).toStrictEqual(['OIB mora imati 11 brojeva']);
54 });
55
56 it('Provjera 5 - Invalid email', async () => {
57   const klijentDTO = {
58     firstName: 'Però',
59     lastName: 'Peric',
60     cardNumber: '0012312342325623',
61     email: 'peroperic',
62     OIB: '12345678912',
63     password: 'lozinka',
64   } as KlijentDTO;
65   const actual = await RacunValidator.validate(klijentDTO);
66   expect(actual).toStrictEqual(['email je neispravan']);
67 });
68 });
69

```

Slika 5.4: Kod za test validacije računa 1

```

TS racunValidator.test.ts ✘
backend > tests > TS racunValidator.test.ts > ...
27   expect(actual.length).toBe(0);
28 });
29
30 it('Provjera 3 - Incorrect OIB input', async () => {
31   const klijentDTO = {
32     firstName: 'Però',
33     lastName: 'Peric',
34     cardNumber: '0012312342325623',
35     email: 'peric@mail.com',
36     OIB: '1234568912',
37     password: 'lozinka',
38   } as KlijentDTO;
39   const actual = await RacunValidator.validate(klijentDTO);
40   expect(actual).toStrictEqual(['OIB mora imati samo brojeve']);
41 });
42
43 it('Provjera 4 - Incorrect OIB size', async () => {
44   const klijentDTO = {
45     firstName: 'Però',
46     lastName: 'Peric',
47     cardNumber: '0012312342325623',
48     email: 'peric@mail.com',
49     OIB: '1234568912',
50     password: 'lozinka',
51   } as KlijentDTO;
52   const actual = await RacunValidator.validate(klijentDTO);
53   expect(actual).toStrictEqual(['OIB mora imati 11 brojeva']);
54 });
55
56 it('Provjera 5 - Invalid email', async () => {
57   const klijentDTO = {
58     firstName: 'Però',
59     lastName: 'Peric',
60     cardNumber: '0012312342325623',
61     email: 'peroperic',
62     OIB: '12345678912',
63     password: 'lozinka',
64   } as KlijentDTO;
65   const actual = await RacunValidator.validate(klijentDTO);
66   expect(actual).toStrictEqual(['email je neispravan']);
67 });
68 });
69

```

Slika 5.5: Kod za test validacije računa 2

```

1  import { RacunValidator } from '../utils/validators/RacunValidator';
2  import { KlijentDTO } from '../dtos/KlijentDTO';
3
4  describe('Provjera funkcija u RacunValidator', () => {
5    it('Provjera 1 - No email input', async () => {
6      const klijentDTO = {
7        firstname: 'Pero',
8        lastName: 'Perić',
9        cardNumber: '0012312342325623',
10       OIB: '12345678912',
11       password: 'lozinka',
12     } as KlijentDTO;
13     const actual = await RacunValidator.validate(klijentDTO);
14     expect(actual).toStrictEqual(['Email je obvezan']);
15   });
16
17   it('Provjera 2 - Correct input', async () => {
18     const klijentDTO = {
19       firstname: 'Pero',
20       lastName: 'Perić',
21       cardNumber: '0012312342325623',
22       email: 'peric@mail.com',
23       OIB: '12345678912',
24       password: 'lozinka',
25     } as KlijentDTO;
26     const actual = await RacunValidator.validate(klijentDTO);
27     expect(actual.length).toBe(0);
28   });
29
30   it('Provjera 3 - Incorrect OIB input', async () => {
31     const klijentDTO = {
32       firstname: 'Pero',
33       lastName: 'Perić',
34       cardNumber: '0012312342325623',
35       email: 'peric@mail.com',
36       OIB: '12345678912',
37       password: 'lozinka',
38     } as KlijentDTO;
39   });

```

Slika 5.6: Kod za test validacije računa 3

```

1  it('Provjera 1 - checkIsOneHourLongTrue', () => {
2    const actual = ValidatorFunctions.checkIsOneHourLong(
3      new Date('2021-09-09 09:33:00').toISOString(),
4      new Date('2021-09-09 10:33:00').toISOString()
5    );
6    expect(actual).toBe(true);
7  });
8
9  it('Provjera 2 - checkIsOneHourLongFalse', () => {
10   const actual = ValidatorFunctions.checkIsOneHourLong(
11     new Date('2021-09-09 09:33:00').toISOString(),
12     new Date('2021-09-09 10:32:59').toISOString()
13   );
14   expect(actual).toBe(false);
15 });
16
17 it('Provjera 3 - checkIsOneDayLongTrue', () => {
18   const actual = ValidatorFunctions.checkIsOneDayLong(
19     new Date('2021-09-09 09:33:00').toISOString(),
20     new Date('2021-09-10 09:33:00').toISOString()
21   );
22   expect(actual).toBe(true);
23 });
24
25 it('Provjera 4 - checkIsOneDayLongFalse', () => {
26   const actual = ValidatorFunctions.checkIsOneDayLong(
27     new Date('2021-09-09 09:33:00').toISOString(),
28     new Date('2021-09-10 02:32:59').toISOString()
29   );
30   expect(actual).toBe(false);
31 });
32
33 it('Provjera 5 - checkIsOneDayLongTrue', () => {
34   const actual = ValidatorFunctions.checkIsOneDayLong(
35     new Date('2021-09-09 09:33:00').toISOString(),
36     new Date('2021-09-10 09:33:00').toISOString()
37   );
38   expect(actual).toBe(true);
39 });
40
41 it('Provjera 6 - checkIsOneDayLongFalse', () => {
42   const actual = ValidatorFunctions.checkIsOneDayLong(
43     new Date('2021-09-09 09:33:00').toISOString(),
44     new Date('2021-09-10 02:32:59').toISOString()
45   );
46   expect(actual).toBe(false);
47 });
48
49 it('Provjera 7 - checkIsOneDayLongFalse', () => {
50   const actual = ValidatorFunctions.checkIsOneDayLong(
51     new Date('2021-09-09 09:33:00').toISOString(),
52     new Date('2021-09-10 02:32:59').toISOString()
53   );
54   expect(actual).toBe(false);
55 });
56
57 });
58
59 });
60

```

Slika 5.7: Kod za test metoda s datumima 1

```
ts times.test.ts x
backend > tests > TS-times.test.ts > ...
1 import { ValidatorFunctions } from '../utils/validators/ValidatorFunctions';
2
3 describe('Provjera funkcija u ValidatorFunctions', () => {
4   it('Provjera 1 - checkIsStartBeforeEnd', () => {
5     const actual = ValidatorFunctions.checkIsStartBeforeEnd(
6       new Date('2021-01-03').toISOString(),
7       new Date('2021-01-04').toISOString()
8     );
9
10    expect(actual).toBe(true);
11  });
12
13 it('Provjera 2 - checkIsStartBeforeNow', () => {
14   const actual = ValidatorFunctions.checkIsStartBeforeNow(
15     new Date('2021-01-01').toISOString()
16   );
17
18   expect(actual).toBe(false);
19 });
20
21 it('Provjera 3 - checkIsStartBeforeNow6Hours', () => {
22   const actual = ValidatorFunctions.checkIsStartBeforeNow6Hours(
23     new Date('2021-09-09').toISOString()
24   );
25   expect(actual).toBe(true);
26 });
27
28 it('Provjera 4 - checkIsOneHourLongTrue', () => {
29   const actual = ValidatorFunctions.checkIsOneHourLong(
30     new Date('2021-09-09 09:33:00').toISOString(),
31     new Date('2021-09-09 10:33:00').toISOString()
32   );
33   expect(actual).toBe(true);
34 });
35
36 it('Provjera 5 - checkIsOneHourLongFalse', () => {
37   const actual = ValidatorFunctions.checkIsOneHourLong(
38     new Date('2021-09-09 09:33:00').toISOString()
```

Slika 5.8: Kod za test metoda s datumima 2

5.2.2 Ispitivanje sustava

Potrebno je provesti i opisati ispitivanje sustava koristeći radni okvir Selenium¹⁹. 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 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.

Ispitivanjem sustava želimo provjeriti osnovne funkcionalnosti aplikacije koje korisnik susreće. Svi su pokrenuti testovi prošli.

¹⁹<https://www.seleniumhq.org/>

```
seven/frontend on ✘ selenium [?] [?] took 41s
› npm run test-selenium

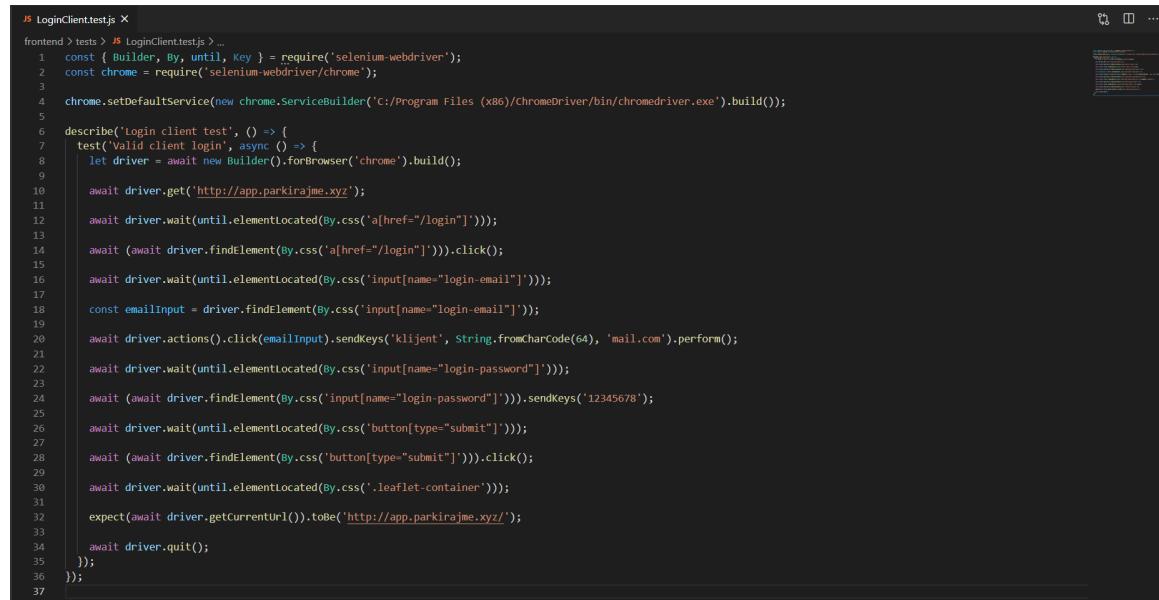
> frontend@0.1.0 test-selenium /Users/kristiandjakovic/Personal/FER/v-semestar/pi/seven/frontend
> jest --runInBand

PASS  tests/AddParking.test.js (5.664 s)
PASS  tests/RegistrationCompany.test.js
PASS  tests/OneTimeReservation.test.js (5.279 s)
PASS  tests/AddVehicle.test.js
PASS  tests/LoginClient.test.js

Test Suites: 5 passed, 5 total
Tests:      5 passed, 5 total
Snapshots:  0 total
Time:       25.141 s, estimated 35 s
Ran all test suites.
```

Slika 5.9: Selenium testovi

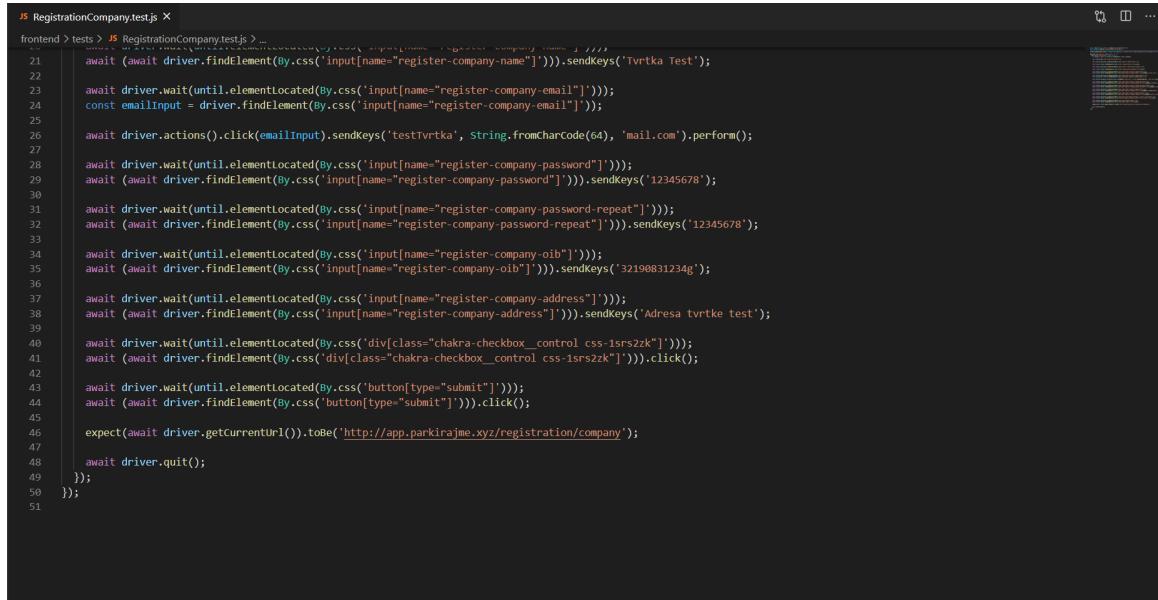
Prvi test ispituje prijavu korisnika na aplikaciju. Nakon uspješne prijave korisnik je preusmjeren na početnu stranicu i prijavljen je.



```
JS LoginClient.test.js ×
frontend > tests > JS LoginClient.test.js >...
1  const { Builder, By, until, Key } = require('selenium-webdriver');
2  const chrome = require('selenium-webdriver/chrome');
3
4  chrome.setDefaultService(new chrome.ServiceBuilder('C:/Program Files (x86)/ChromeDriver/bin/chromedriver.exe').build());
5
6  describe('Login client test', () => {
7    test('Valid Client Login', async () => {
8      let driver = await new Builder().forBrowser('chrome').build();
9
10     await driver.get('http://app.parkirajme.xyz');
11
12     await driver.wait(until.elementLocated(By.css('a[href="/login"]')));
13
14     await (await driver.findElement(By.css('a[href="/login"]'))).click();
15
16     await driver.wait(until.elementLocated(By.css('input[name="login-email"]')));
17
18     const emailInput = driver.findElement(By.css('input[name="login-email"]'));
19
20     await driver.actions().click(emailInput).sendKeys('klijent', String.fromCharCode(64), 'mail.com').perform();
21
22     await driver.wait(until.elementLocated(By.css('input[name="login-password"]')));
23
24     await (await driver.findElement(By.css('input[name="login-password"]'))).sendKeys('12345678');
25
26     await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
27
28     await (await driver.findElement(By.css('button[type="submit"]'))).click();
29
30     await driver.wait(until.elementLocated(By.css('.leaflet-container')));
31
32     expect(await driver.getCurrentUrl()).toBe('http://app.parkirajme.xyz/');
33
34     await driver.quit();
35   });
36 });
37
```

Slika 5.10: Kod za test prijave klijenta

Drugi test provjerava neispravnu registraciju tvrtke kad korisnik unese pogrešan OIB. Na zaslon se ispiše poruka i korisnik ostaje na istoj stranici i omogućava mu se ponovni unos podataka. Ako je registracija uspješna, korisnik se preusmjeri na početnu stranicu aplikacije.

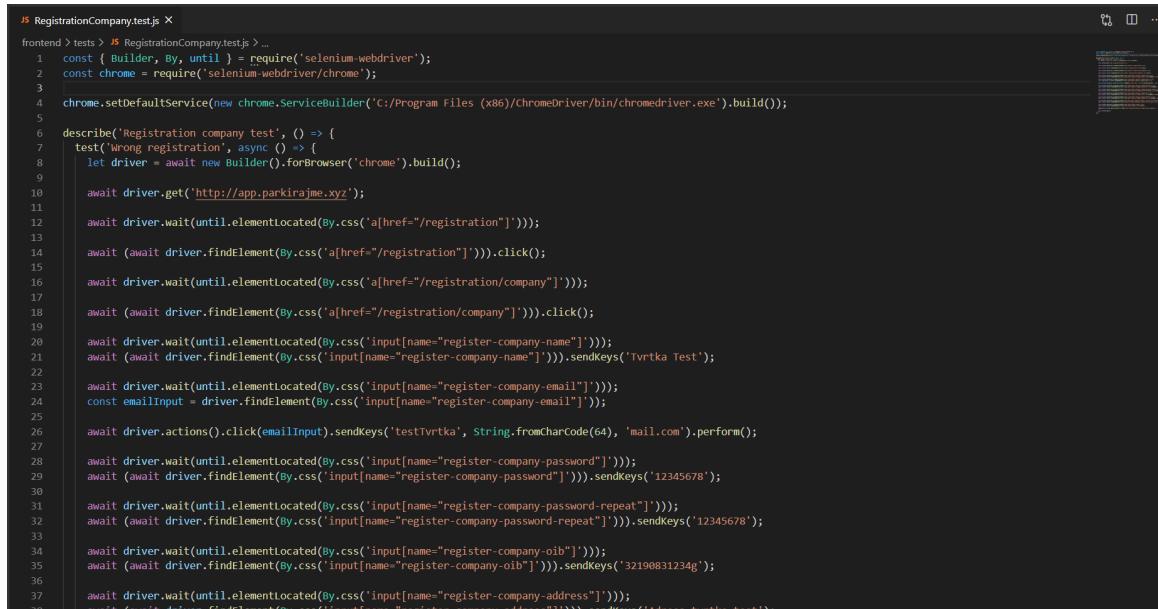


```

1  //<RegistrationCompany.test.js>
2  //> tests > JS RegistrationCompany.test.js ...
3  //> frontend > tests > JS RegistrationCompany.test.js ...
4  //> await (await driver.findElement(By.css('input[name="register-company-name"]'))).sendKeys('Tvrta Test');
5  //> await driver.wait(until.elementLocated(By.css('input[name="register-company-email"]')));
6  //> const emailInput = driver.findElement(By.css('input[name="register-company-email"]'));
7  //> await driver.actions().click(emailInput).sendKeys('testTvrta', String.fromCharCode(64), 'mail.com').perform();
8  //> await driver.wait(until.elementLocated(By.css('input[name="register-company-password"]')));
9  //> await (await driver.findElement(By.css('input[name="register-company-password"]'))).sendKeys('12345678');
10 //> await driver.wait(until.elementLocated(By.css('input[name="register-company-password-repeat"]')));
11 //> await (await driver.findElement(By.css('input[name="register-company-password-repeat"]'))).sendKeys('12345678');
12 //> await driver.wait(until.elementLocated(By.css('input[name="register-company-oib"]')));
13 //> await (await driver.findElement(By.css('input[name="register-company-oib"]'))).sendKeys('32190831234g');
14 //> await driver.wait(until.elementLocated(By.css('input[name="register-company-address"]')));
15 //> await (await driver.findElement(By.css('input[name="register-company-address"]'))).sendKeys('Adresa tvrtke test');
16 //> await driver.wait(until.elementLocated(By.css('div[class="chakra-checkbox__control css-1srs2zk"]')));
17 //> await (await driver.findElement(By.css('div[class="chakra-checkbox__control css-1srs2zk"]'))).click();
18 //> await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
19 //> await (await driver.findElement(By.css('button[type="submit"]'))).click();
20 //> expect(await driver.getCurrentUrl()).toBe('http://app.parkirajme.xyz/registration/company');
21 //> await driver.quit();
22 //> });
23 //> });
24 //> });
25 //> });
26 //> });
27 //> });
28 //> });
29 //> });
30 //> });
31 //> });
32 //> });
33 //> });
34 //> });
35 //> });
36 //> });
37 //> });
38 //> });
39 //> });
40 //> });
41 //> });
42 //> });
43 //> });
44 //> });
45 //> });
46 //> });
47 //> });
48 //> });
49 //> });
50 //> });
51 //> });

```

Slika 5.11: Kod za test registracije tvrtka 1



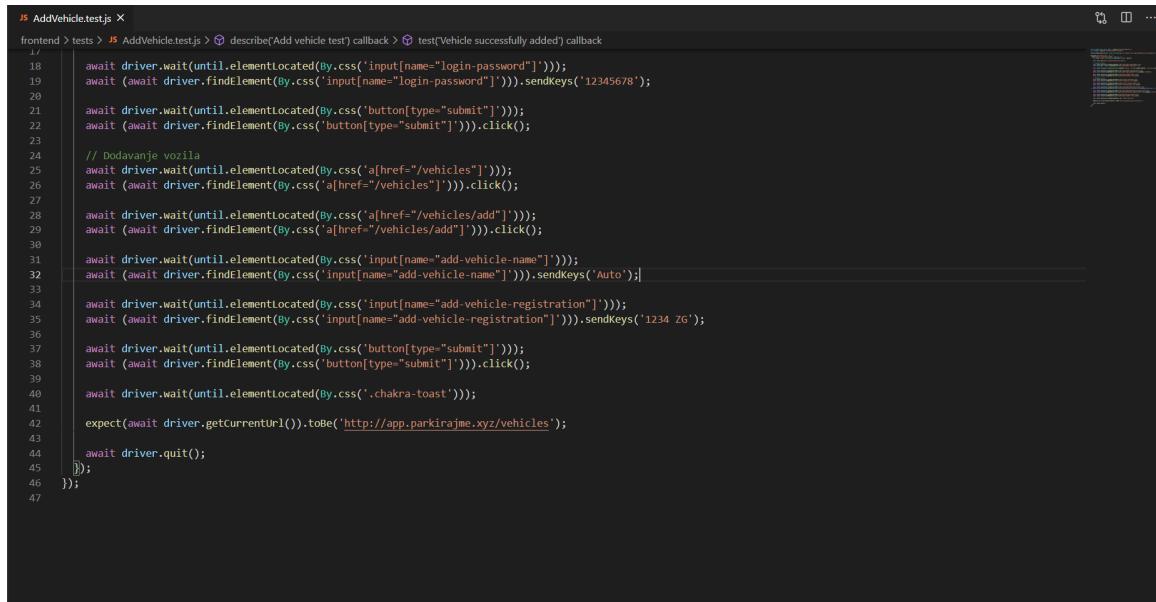
```

1  //<RegistrationCompany.test.js>
2  //> tests > JS RegistrationCompany.test.js ...
3  //> frontend > tests > JS RegistrationCompany.test.js ...
4  //> const { Builder, By, until } = require('selenium-webdriver');
5  //> const chrome = require('selenium-webdriver/chrome');
6  //> chrome.setDefaultService(new chrome.ServiceBuilder('C:/Program Files (x86)/ChromeDriver/bin/chromedriver.exe').build());
7  //> describe('Registration company test', () => {
8  //>   test('Wrong registration', async () => {
9  //>     let driver = await new Builder().forBrowser('chrome').build();
10 //>     await driver.get('http://app.parkirajme.xyz');
11 //>     await driver.wait(until.elementLocated(By.css('a[href="/registration"]')));
12 //>     await (await driver.findElement(By.css('a[href="/registration"]'))).click();
13 //>     await driver.wait(until.elementLocated(By.css('a[href="/registration/company"]')));
14 //>     await (await driver.findElement(By.css('a[href="/registration/company"]'))).click();
15 //>     await driver.wait(until.elementLocated(By.css('input[name="register-company-name"]')));
16 //>     await (await driver.findElement(By.css('input[name="register-company-name"]'))).sendKeys('Tvrta Test');
17 //>     await driver.wait(until.elementLocated(By.css('input[name="register-company-email"]')));
18 //>     const emailInput = driver.findElement(By.css('input[name="register-company-email"]'));
19 //>     await driver.actions().click(emailInput).sendKeys('testTvrta', String.fromCharCode(64), 'mail.com').perform();
20 //>     await driver.wait(until.elementLocated(By.css('input[name="register-company-password"]')));
21 //>     await (await driver.findElement(By.css('input[name="register-company-password"]'))).sendKeys('12345678');
22 //>     await driver.wait(until.elementLocated(By.css('input[name="register-company-password-repeat"]')));
23 //>     await (await driver.findElement(By.css('input[name="register-company-password-repeat"]'))).sendKeys('12345678');
24 //>     await driver.wait(until.elementLocated(By.css('input[name="register-company-oib"]')));
25 //>     await (await driver.findElement(By.css('input[name="register-company-oib"]'))).sendKeys('32190831234g');
26 //>     await driver.wait(until.elementLocated(By.css('input[name="register-company-address"]')));
27 //>     await (await driver.findElement(By.css('input[name="register-company-address"]'))).sendKeys('Adresa tvrtke test');
28 //>     await driver.wait(until.elementLocated(By.css('div[class="chakra-checkbox__control css-1srs2zk"]')));
29 //>     await (await driver.findElement(By.css('div[class="chakra-checkbox__control css-1srs2zk"]'))).click();
30 //>     await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
31 //>     await (await driver.findElement(By.css('button[type="submit"]'))).click();
32 //>     expect(await driver.getCurrentUrl()).toBe('http://app.parkirajme.xyz/registration/company');
33 //>     await driver.quit();
34 //>   });
35 //> });
36 //> });
37 //> });
38 //> });
39 //> });
40 //> });
41 //> });
42 //> });
43 //> });
44 //> });
45 //> });
46 //> });
47 //> });
48 //> });
49 //> });
50 //> });
51 //> });

```

Slika 5.12: Kod za test registracije tvrtka 2

Treći test provjerava dodavanje vozila. Korisnik se prijavi, odabere stranicu 'Moja vozila', klikne na opciju 'Dodaj', popuni podatke i pošalje zahtjev. Nakon što je vozilo uspješno dodano, korisnika se preusmjerava na stranicu s njegovim vozilima.

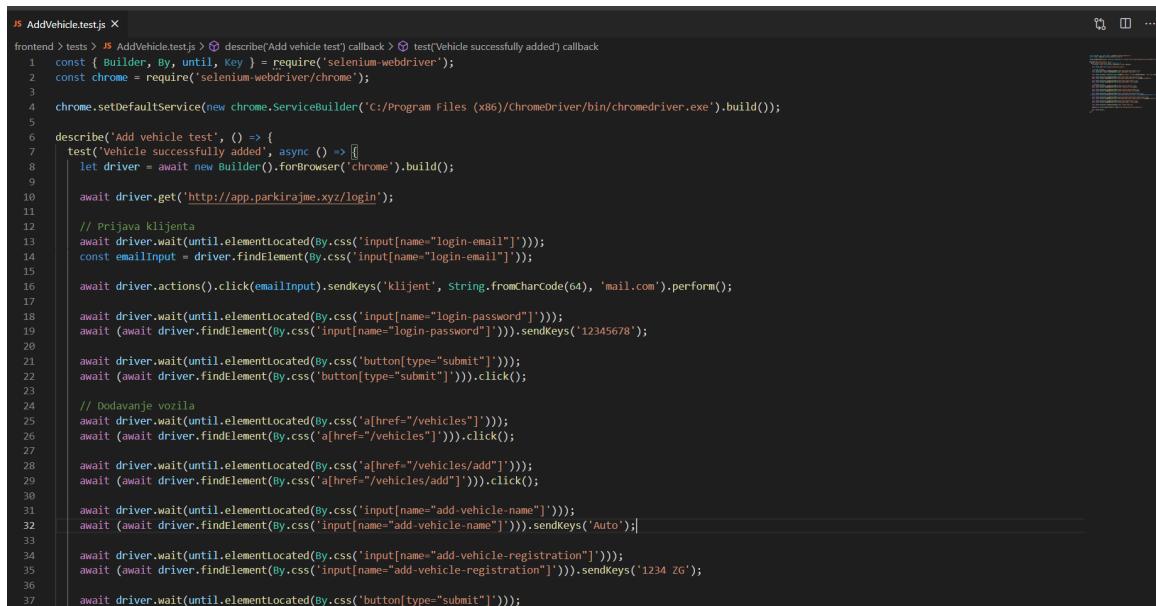


```

15 AddVehicle.test.js ×
frontend > tests > JS AddVehicle.test.js > ⚡ describe('Add vehicle test') callback > ⚡ test('Vehicle successfully added') callback
16
17   await driver.wait(until.elementLocated(By.css('input[name="login-password"]')));
18   await (await driver.findElement(By.css('input[name="login-password"]'))).sendKeys('12345678');
19
20   await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
21   await (await driver.findElement(By.css('button[type="submit"]'))).click();
22
23   // Dodavanje vozila
24   await driver.wait(until.elementLocated(By.css('a[href="/vehicles"]')));
25   await (await driver.findElement(By.css('a[href="/vehicles"]'))).click();
26
27   await driver.wait(until.elementLocated(By.css('a[href="/vehicles/add"]')));
28   await (await driver.findElement(By.css('a[href="/vehicles/add"]'))).click();
29
30   await driver.wait(until.elementLocated(By.css('input[name="add-vehicle-name"]')));
31   await (await driver.findElement(By.css('input[name="add-vehicle-name"]'))).sendKeys('Auto');
32
33   await driver.wait(until.elementLocated(By.css('input[name="add-vehicle-registration"]')));
34   await (await driver.findElement(By.css('input[name="add-vehicle-registration"]'))).sendKeys('1234 ZG');
35
36   await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
37   await (await driver.findElement(By.css('button[type="submit"]'))).click();
38
39   await driver.wait(until.elementLocated(By.css('.chakra-toast')));
40
41 expect(await driver.getCurrentUrl()).toBe('http://app.parkirajme.xyz/vehicles');
42
43 await driver.quit();
44
45 });
46 });
47

```

Slika 5.13: Kod za test dodavanja vozila 1



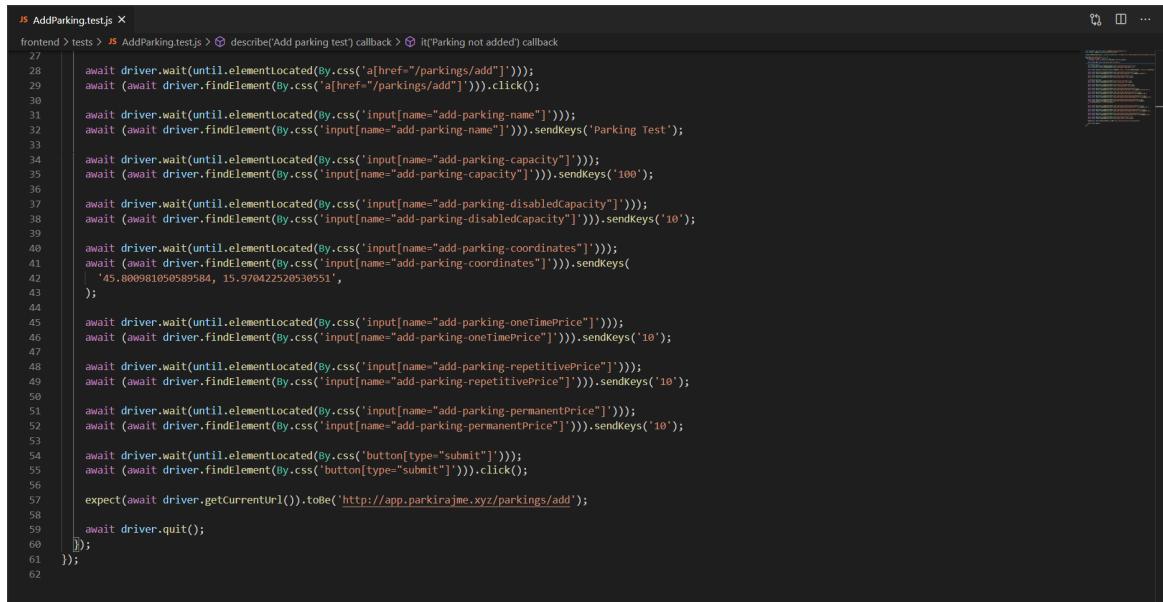
```

JS AddVehicle.test.js ×
frontend > tests > JS AddVehicle.test.js > ⚡ describe('Add vehicle test') callback > ⚡ test('Vehicle successfully added') callback
1  const { Builder, By, until, Key } = require('selenium-webdriver');
2  const chrome = require('selenium-webdriver/chrome');
3
4  chrome.setDefaultService(new chrome.ServiceBuilder('C:/Program Files (x86)/ChromeDriver/bin/chromedriver.exe').build());
5
6  describe('Add vehicle test', () => {
7    test('Vehicle successfully added', async () => [
8      let driver = await new Builder().forBrowser('chrome').build();
9
10     await driver.get('http://app.parkirajme.xyz/login');
11
12     // Prijava klijenta
13     await driver.wait(until.elementLocated(By.css('input[name="login-email"]')));
14     const emailInput = driver.findElement(By.css('input[name="login-email"]'));
15
16     await driver.actions().click(emailInput).sendKeys('klijent', String.fromCharCode(64), 'mail.com').perform();
17
18     await driver.wait(until.elementLocated(By.css('input[name="login-password"]')));
19     await (await driver.findElement(By.css('input[name="login-password"]'))).sendKeys('12345678');
20
21     await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
22     await (await driver.findElement(By.css('button[type="submit"]'))).click();
23
24     // Dodavanje vozila
25     await driver.wait(until.elementLocated(By.css('a[href="/vehicles"]')));
26     await (await driver.findElement(By.css('a[href="/vehicles"]'))).click();
27
28     await driver.wait(until.elementLocated(By.css('a[href="/vehicles/add"]')));
29     await (await driver.findElement(By.css('a[href="/vehicles/add"]'))).click();
30
31     await driver.wait(until.elementLocated(By.css('input[name="add-vehicle-name"]')));
32     await (await driver.findElement(By.css('input[name="add-vehicle-name"]'))).sendKeys('Auto');
33
34     await driver.wait(until.elementLocated(By.css('input[name="add-vehicle-registration"]')));
35     await (await driver.findElement(By.css('input[name="add-vehicle-registration"]'))).sendKeys('1234 ZG');
36
37     await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
38
39   ]);
40
41 ]);
42
43
44
45
46
47

```

Slika 5.14: Kod za test dodavanja vozila 2

Četvrti test provjerava dodavanje parkirališta. Tvrtka se prijavi, odabere stranicu 'Moja parkirališta', klikne na opciju 'Dodaj', popuni podatke i pošalje zahtjev. Nakon što je parkiralište uspješno dodano, tvrtku se preusmjerava na stranicu s njegovim parkiralištima.

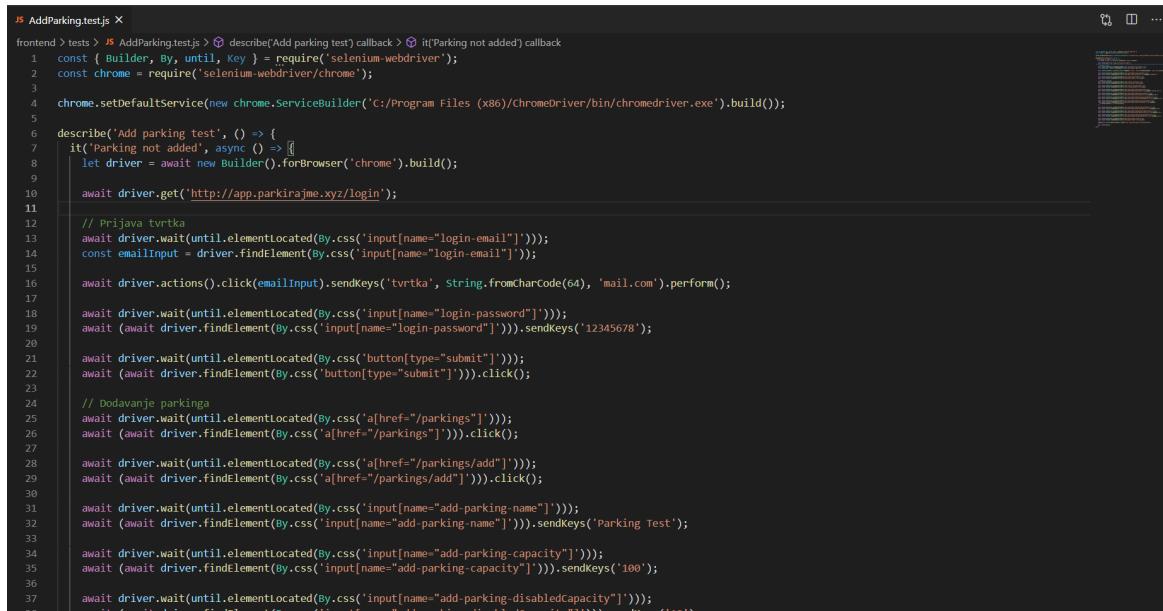


```

JS AddParking.test.js ×
frontend > tests > JS AddParking.test.js > ⚡ describe('Add parking test') callback > ⚡ it('Parking not added') callback
27
28   await driver.wait(until.elementLocated(By.css('a[href="/parkings/add"]')));
29   await (await driver.findElement(By.css('a[href="/parkings/add"]'))).click();
30
31   await driver.wait(until.elementLocated(By.css('input[name="add-parking-name"]')));
32   await (await driver.findElement(By.css('input[name="add-parking-name"]'))).sendKeys('Parking Test');
33
34   await driver.wait(until.elementLocated(By.css('input[name="add-parking-capacity"]')));
35   await (await driver.findElement(By.css('input[name="add-parking-capacity"]'))).sendKeys('100');
36
37   await driver.wait(until.elementLocated(By.css('input[name="add-parking-disabledCapacity"]')));
38   await (await driver.findElement(By.css('input[name="add-parking-disabledCapacity"]'))).sendKeys('10');
39
40   await driver.wait(until.elementLocated(By.css('input[name="add-parking-coordinates"]')));
41   await (await driver.findElement(By.css('input[name="add-parking-coordinates"]'))).sendKeys(
42     '| 45.800981050589584, 15.970422520530551',
43   );
44
45   await driver.wait(until.elementLocated(By.css('input[name="add-parking-oneTimePrice"]')));
46   await (await driver.findElement(By.css('input[name="add-parking-oneTimePrice"]'))).sendKeys('10');
47
48   await driver.wait(until.elementLocated(By.css('input[name="add-parking-repetitivePrice"]')));
49   await (await driver.findElement(By.css('input[name="add-parking-repetitivePrice"]'))).sendKeys('10');
50
51   await driver.wait(until.elementLocated(By.css('input[name="add-parking-permanentPrice"]')));
52   await (await driver.findElement(By.css('input[name="add-parking-permanentPrice"]'))).sendKeys('10');
53
54   await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
55   await (await driver.findElement(By.css('button[type="submit"]'))).click();
56
57   expect(await driver.getCurrentUrl()).toBe('http://app.parkirajme.xyz/parkings/add');
58
59   await driver.quit();
60 });
61 });
62

```

Slika 5.15: Kod za test dodavanja parkinga 1



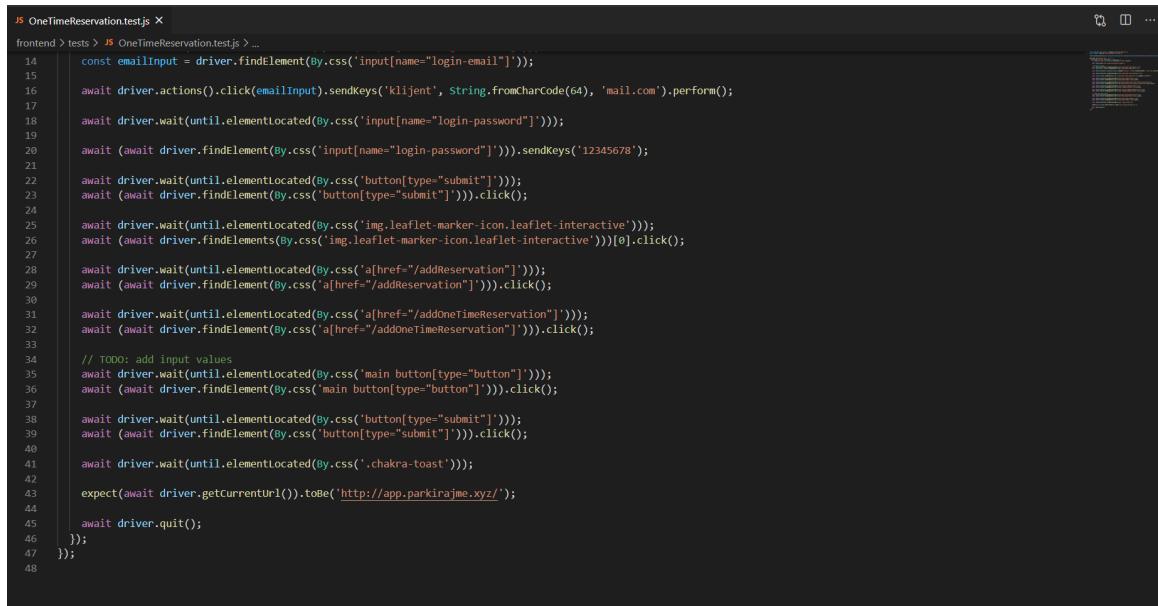
```

JS AddParking.test.js ×
frontend > tests > JS AddParking.test.js > ⚡ describe('Add parking test') callback > ⚡ it('Parking not added') callback
1 const { Builder, By, until, Key } = require('selenium-webdriver');
2 const chrome = require('selenium-webdriver/chrome');
3
4 chrome.setDefaultService(new chrome.ServiceBuilder('C:/Program Files (x86)/Chromedriver/bin/chromedriver.exe').build());
5
6 describe('Add parking test', () => {
7   it('Parking not added', async () => [
8     let driver = await new Builder().forBrowser('chrome').build();
9
10    await driver.get('http://app.parkirajme.xyz/login');
11
12    // Prijava tvrtka
13    await driver.wait(until.elementLocated(By.css('input[name="login-email"]')));
14    const emailInput = driver.findElement(By.css('input[name="login-email"]'));
15
16    await driver.actions().click(emailInput).sendKeys('tvrtka', String.fromCharCode(64), 'mail.com').perform();
17
18    await driver.wait(until.elementLocated(By.css('input[name="login-password"]')));
19    await (await driver.findElement(By.css('input[name="login-password"]'))).sendKeys('12345678');
20
21    await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
22    await (await driver.findElement(By.css('button[type="submit"]'))).click();
23
24    // Dodavanje parkinga
25    await driver.wait(until.elementLocated(By.css('a[href="/parkings"]')));
26    await (await driver.findElement(By.css('a[href="/parkings"]'))).click();
27
28    await driver.wait(until.elementLocated(By.css('a[href="/parkings/add"]')));
29    await (await driver.findElement(By.css('a[href="/parkings/add"]'))).click();
30
31    await driver.wait(until.elementLocated(By.css('input[name="add-parking-name"]')));
32    await (await driver.findElement(By.css('input[name="add-parking-name"]'))).sendKeys('Parking Test');
33
34    await driver.wait(until.elementLocated(By.css('input[name="add-parking-capacity"]')));
35    await (await driver.findElement(By.css('input[name="add-parking-capacity"]'))).sendKeys('100');
36
37    await driver.wait(until.elementLocated(By.css('input[name="add-parking-disabledCapacity"]')));
38    await (await driver.findElement(By.css('input[name="add-parking-disabledCapacity"]'))).sendKeys('10');
39
40    await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
41    await (await driver.findElement(By.css('button[type="submit"]'))).click();
42
43    expect(await driver.getCurrentUrl()).toBe('http://app.parkirajme.xyz/parkings/add');
44
45    await driver.quit();
46 });
47

```

Slika 5.16: Kod za test dodavanja parkinga 2

Peti test provjerava dodavanje jednokratne rezervacije. Prijavljeni klijent odbire željeno parkiralište, klikne na gumb rezerviraj i opciju 'Jednokratna' za jednokratnu rezervaciju. Potom ispunii tražene podatke i odabere opciju 'Rezerviraj'. Opcijom 'Potvrdi' klijent plaća navedenu rezervaciju. Nakon uspješne rezervacije korisnika se preusmjeri na početnu stranicu s kartom.

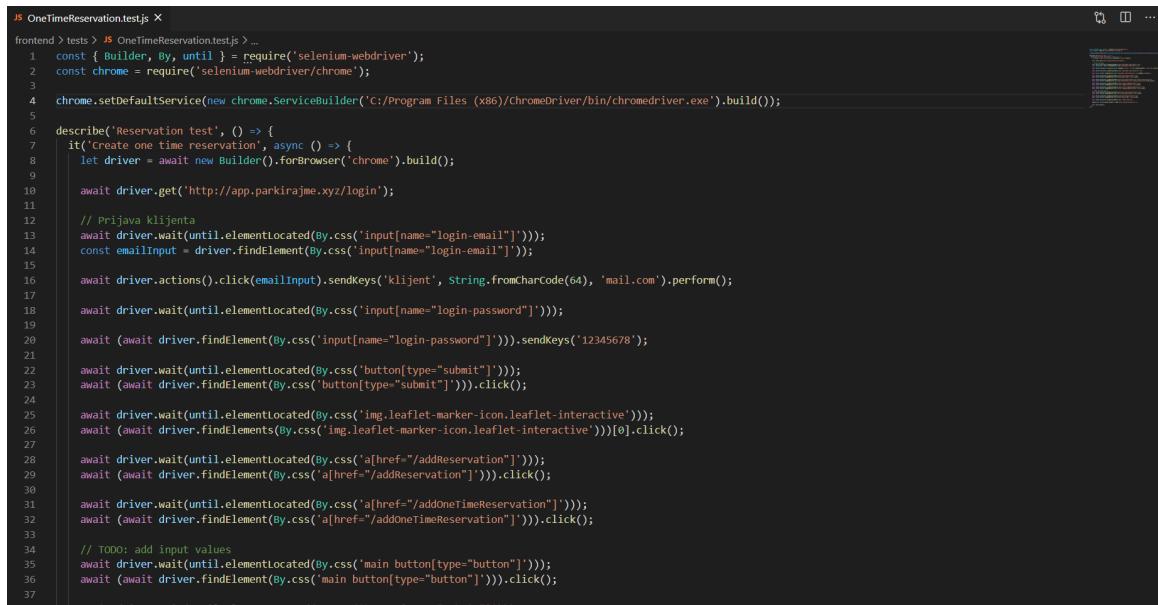


```

JS OneTimeReservation.test.js ×
frontend > tests > JS OneTimeReservation.test.js > ...
14 const emailInput = driver.findElement(By.css('input[name="login-email"]'));
15
16 await driver.actions().click(emailInput).sendKeys('klijent', String.fromCharCode(64), 'mail.com').perform();
17
18 await driver.wait(until.elementLocated(By.css('input[name="login-password"]')));
19
20 await (await driver.findElement(By.css('input[name="login-password"]'))).sendKeys('12345678');
21
22 await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
23 await (await driver.findElement(By.css('button[type="submit"]'))).click();
24
25 await driver.wait(until.elementLocated(By.css('img.leaflet-marker-icon.leaflet-interactive')));
26 await (await driver.findElements(By.css('img.leaflet-marker-icon.leaflet-interactive'))[0].click());
27
28 await driver.wait(until.elementLocated(By.css('a[href="/addReservation"]')));
29 await (await driver.findElement(By.css('a[href="/addReservation"]'))).click();
30
31 await driver.wait(until.elementLocated(By.css('a[href="/addOneTimeReservation"]')));
32 await (await driver.findElement(By.css('a[href="/addOneTimeReservation"]'))).click();
33
34 // TODO: add input values
35 await driver.wait(until.elementLocated(By.css('main button[type="button"]')));
36 await (await driver.findElement(By.css('main button[type="button"]'))).click();
37
38 await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
39 await (await driver.findElement(By.css('button[type="submit"]'))).click();
40
41 await driver.wait(until.elementLocated(By.css('.chakra-toast')));
42
43 expect(await driver.getCurrentUrl()).toBe('http://app.parkirajme.xyz/');
44
45 await driver.quit();
46 });
47 });
48

```

Slika 5.17: Kod za test rezervacije 1



```

JS OneTimeReservation.test.js ×
frontend > tests > JS OneTimeReservation.test.js > ...
1 const { Builder, By, until } = require('selenium-webdriver');
2 const chrome = require('selenium-webdriver/chrome');
3
4 chrome.setDefaultService(new chrome.ServiceBuilder('C:/Program Files (x86)/chromedriver/bin/chromedriver.exe').build());
5
6 describe('Reservation test', () => {
7   it('Create one time reservation', async () => {
8     let driver = await new Builder().forBrowser('chrome').build();
9
10    await driver.get('http://app.parkirajme.xyz/login');
11
12    // Prijava klijenta
13    await driver.wait(until.elementLocated(By.css('input[name="login-email"]')));
14    const emailInput = driver.findElement(By.css('input[name="login-email"]'));
15
16    await driver.actions().click(emailInput).sendKeys('klijent', String.fromCharCode(64), 'mail.com').perform();
17
18    await driver.wait(until.elementLocated(By.css('input[name="login-password"]')));
19
20    await (await driver.findElement(By.css('input[name="login-password"]'))).sendKeys('12345678');
21
22    await driver.wait(until.elementLocated(By.css('button[type="submit"]')));
23    await (await driver.findElement(By.css('button[type="submit"]'))).click();
24
25    await driver.wait(until.elementLocated(By.css('img.leaflet-marker-icon.leaflet-interactive')));
26    await (await driver.findElements(By.css('img.leaflet-marker-icon.leaflet-interactive'))[0].click());
27
28    await driver.wait(until.elementLocated(By.css('a[href="/addReservation"]')));
29    await (await driver.findElement(By.css('a[href="/addReservation"]'))).click();
30
31    await driver.wait(until.elementLocated(By.css('a[href="/addOneTimeReservation"]')));
32    await (await driver.findElement(By.css('a[href="/addOneTimeReservation"]'))).click();
33
34 // TODO: add input values
35 await driver.wait(until.elementLocated(By.css('main button[type="button"]')));
36 await (await driver.findElement(By.css('main button[type="button"]'))).click();
37

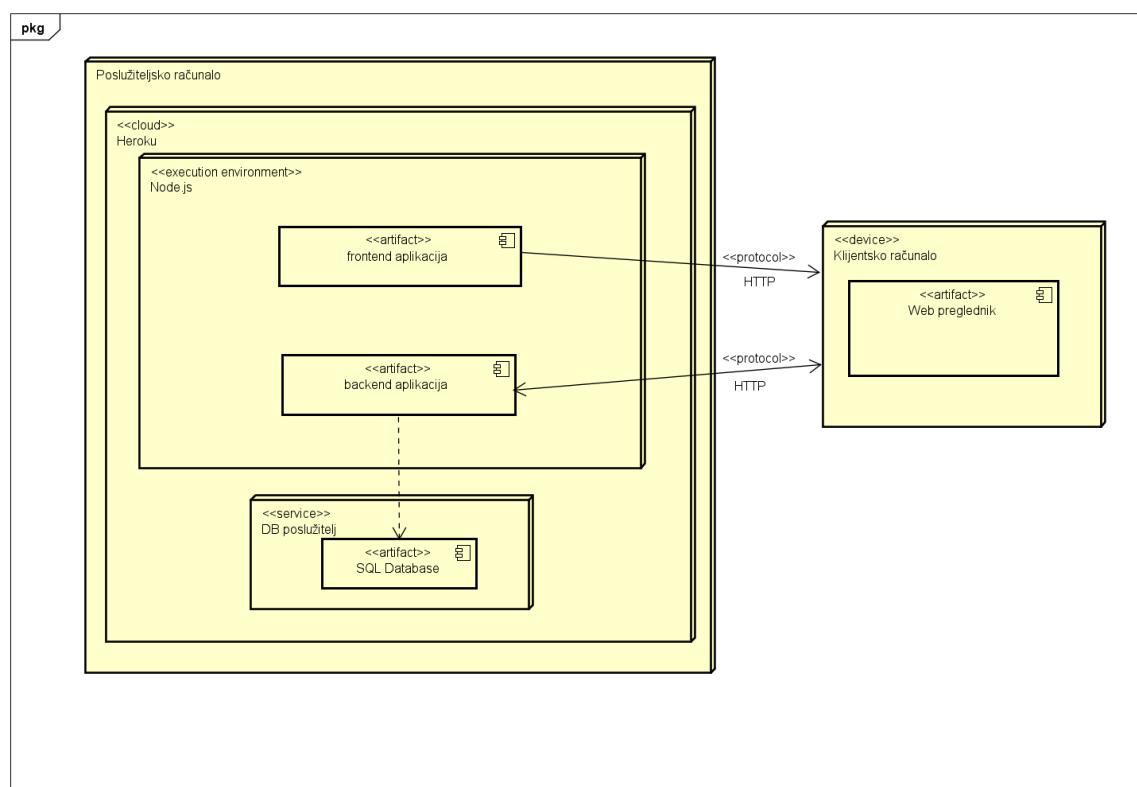
```

Slika 5.18: Kod za test rezervacije 2

5.3 Dijagram razmještaja

Dijagram razmještaja koristi se za prikaz topologije sustava pri čemu je naglasak stavljen na odnose između sklopovskih i programskih komponenti. Oni opisuju korištenu programsku potporu sustava te njegove elemente i okolinu izvođenja. Osnovni su elementi čvorovi, artefakti te spojevi koji predstavljaju komunikacijske puteve između njih.

Klijent putem svojeg web preglednika pristupa web aplikaciji koja se vrti na cloud serveru. Sva komunikacija odvija se putem protokola HTTP. Na poslužiteljskom računalu nalaze se baza podataka te aplikacija.



Slika 5.19: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Prije ikakve radnje, potrebno je klonirati *GitLab* repozitorij sa kodom te otvoriti terminal (i opcionalno sistemski preglednik) u tom direktoriju. Udaljeni *GitLab* repozitorij možete pronaći na [ovoj poveznici](#)

Kako bi kloniranje bilo moguće potrebno je imati instalirani *Git* klijent na računalu. *Git* klijent moguće je preuzeti na [ovoj poveznici](#).

Još jedan uvjet kako bi aplikacija radila lokalno jest instaliran *Node.js*. *Node.js* moguće je preuzeti na [ovoj poveznici](#). Instalacijom *Node.js* aplikacije, instalirat će vam se i [*npm*](#)²⁰ koji je zapravo abrevijacija od *Node Package Manager*. Samo ime govori da je to alat za upravljanje *Node.js* paketima.

²⁰npm

5.4.1 Frontend aplikacija

Lokalna instalacija

Da bismo pokrenuli lokalnu frontend aplikaciju potrebno je unutar terminala pozicionirati se u frontend mapu unutar korijenskog direktorija.

```
cd frontend
```

Sada je potrebno instalirati potrebne pakete za uspješan rad aplikacije. To činimo pokretanjem naredbe

```
npm install
```

Nakon instalacije paketa, potrebno je podesiti i lokalni *host* file. Da bismo uredili *host* file prvo ga je potrebno locirati, a sama lokacija ovisi o operacijskom sustavu. Ukoliko koristimo operacijski sustav na bazi *Linuxa* *host* file se nalazi na lokaciji */etc/hosts*, dok se na *Windows* operacijskom sustavu ta datoteka nalazi na *C:\Windows\System32\drivers\etc\hosts*

Jednom kada smo je locirali, potrebno je na kraj datoteke dodati redak

```
127.0.0.1 frontend-local.parkirajme.xyz
```

Također, potrebno je unutar *frontend* direktorija kreirati datoteku *.env* te u nju dodati sljedeći zapis

```
HOST=frontend-local.parkirajme.xyz
```

Dodavanjem ovih zapisa, završili smo lokalnu instalaciju servera te server možemo pokrenuti naredbom

```
npm run dev
```

Kako bi sve ispravno radilo lokalnom serveru ćemo pristupiti sa sljedeće adrese

```
frontend-local.parkirajme.xyz:3000
```

Heroku instalacija

Kako bismo frontend aplikaciju pustili u pogon, potrebne je kreirati novu aplikaciju na Heroku platformi.

1. Otvoriti heroku.com stranicu i prijaviti se ukoliko to već nismo prethodno učinili.
2. U gornjem desnom kutu možemo vidjeti gumb *New* te kada ga pritisnemo otvara nam se padajuća lista i u njoj odabiremo *Create new app*
3. Otvara nam se forma za kreiranje nove aplikacije te tu navodimo ime aplikacije koje može biti proizvoljno. Također na tom koraku odabiremo i regiju servera i tu je poželjno postaviti server koji je nama najbliži, a to je Europa.
4. Pritisom na *Create app* stvaramo aplikaciju

Sada kada je Heroku aplikacija napravljena potrebno je otvoriti .git/config datoteku unutar korijenskog direktorija aplikacije te na kraj te datoteke potrebno je dodati

```
[remote "heroku-frontend"]  
url = https://git.heroku.com/${imeHerokuAplikacije}.git  
fetch = +refs/heads/*:refs/remotes/heroku-frontend/*
```

te umjesto {imeHerokuAplikacije} potrebno je umetnuti ime novostvorene aplikacije. Kada smo podesili config datoteku sve što je potrebno napraviti je pozicionirati se u direktorij frontend aplikacije te iz terminala pokrenuti

```
npm run deploy
```

Nakon par minuta aplikacija će biti puštena u pogon.

HTTPS lokalna instalacija

Kako bi lokalno pokrenuli aplikaciju sa HTTPS protokolom potrebno je u datoteku *frontend/.env* dodati sljedeći zapis

```
HTTPS=true
```

Sada, prilikom sljedećeg pokretanja lokalnog servera, aplikacija će se pokrenuti na HTTPS protokolu te će browser moći pročitati našu trenutnu lokaciju.

5.4.2 Backend aplikacija

Lokalna instalacija

Da bismo pokrenuli lokalnu backend aplikaciju potrebno je unutar terminala pozicionirati se u backend mapu unutar korijenskog direktorija.

```
cd backend
```

Sada je potrebno instalirati potrebne pakete za uspješan rad aplikacije. To činimo pokretanjem naredbe

```
npm install
```

Nakon instalacije paketa, završili smo lokalnu instalaciju servera te server možemo pokrenuti naredbom

```
npm run dev
```

Sada je server pokrenut te mu možemo pristupiti koristeći naredbeni redak i na-redu curl ili pak neki drugi alat za kreiranje HTTP zahtjeve poput alata [Postman](#)²¹.

²¹Postman

Heroku instalacija

Kako bismo backend aplikaciju pustili u pogon, potrebno je kreirati novu aplikaciju na Heroku platformi.

1. Otvoriti heroku.com stranicu i prijaviti se ukoliko to već nismo prethodno učinili.
2. U gornjem desnom kutu možemo vidjeti gumb *New* te kada ga pritisnemo otvara nam se padajuća lista i u njoj odabiremo *Create new app*
3. Otvara nam se forma za kreiranje nove aplikacije te tu navodimo ime aplikacije koje može biti proizvoljno. Također na tom koraku odabiremo i regiju servera i tu je poželjno postaviti server koji je nama najbliži, a to je Europa.
4. Pritisom na *Create app* stvaramo aplikaciju

Sada kada je Heroku aplikacija napravljena potrebno je otvoriti .git/config datoteku unutar korijenskog direktorija aplikacije te na kraj te datoteke potrebno je dodati

```
[remote "heroku-backend"]
  url = https://git.heroku.com/${imeHerokuAplikacije}.git
  fetch = +refs/heads/*:refs/remotes/heroku-backend/*
```

te umjesto {imeHerokuAplikacije} potrebno je umetnuti ime novostvorene aplikacije. Kada smo podesili config datoteku sve što je potrebno napraviti je pozicionirati se u direktorij backend aplikacije te iz terminala pokrenuti

```
npm run deploy
```

Nakon par minuta aplikacija će biti puštena u pogon.

5.4.3 Baza podataka

Lokalna instalacija

Potrebno je preuzeti *pgAdmin* aplikaciju uz koju standardno dolazi i *PostgreSQL* server. Pokrenuti instalaciju te uz *pgAdmin* klijenta odabrati i *PostgreSQL* server za instalaciju. Nakon instalacije potrebno je postaviti korisnika. Nakon instalacije *pgAdmin* klijent i sam *PostgreSQL* server su pokrenuti.

Lokalna konfiguracija

Nakon instalacije, potrebno je napraviti novu lokalnu bazu.

1. Proširiti *Servers* te, ovisno o preuzetoj verziji *PostgreSQL* servera, proširiti *PostgreSQL XX*
2. Desni klik na *Databases* te *Create* te *Database...*
3. Otvara se modal te kao *Database* vrijednost stavljamo proizvoljni naziv
4. Odabiremo *Spremi* te time stvaramo novu bazu

Nakon stvorene baze potrebno je stvoriti konekcijski *connection string*. Njega ćemo stvoriti što ćemo vrijednosti u vitičastim zagradama zamijeniti s pravim vrijednostima.

`postgres://username:password@hostname:port/database`

1. *username* - inicijalni *username* je *postgres*
2. *password* - lozinka postavljena prilikom instalacije
3. *hostname* - pošto je server lokalni, *hostname* će biti *localhost*
4. *port* - inicijalni priključak *PostgreSQL* servera je *5432*
5. *database* - ime novo-napravljene baze

Nakon generiranog *connection stringa*, kreirat ćemo *.env* datoteku u korijenskom direktoriju backend aplikacije te u njoj napraviti zapis:

`DATABASE_URL=connectionString`

Sada će prilikom pokretanja backend aplikacije, prije samog spajanja s bazom, aplikacija pročitati sadržaj *.env* datoteke i učitati te vrijednosti kao sistemske variable. U *Node.js* aplikaciji tim varijablama imamo pristup preko

```
process.env.{imeVrijednost}
```

Heroku instalacija

1. Otići na Heroku Dashboard
2. Odabratи kreiranu backend aplikaciju
3. Odabratи karticu *Resources*
4. Unutar *Add-ons* tražilice tražiti **Heroku Postgres**
5. Kada se otvori modal, odabratи **Submit Order Form**

I to je to! Server sada ima svoju bazu. Ukoliko se želimo spojiti na tu bazu preko pgAdmin klijenta tada slijedimo sljedeće korake:

1. S *Resources* zaslona odaberemo naš *Heroku Postgres* server
2. Nakon malo duljeg učitavanja, odabiremo karticu *Settings*
3. Odabiremo gumb *View Credentials...*

Sada su nam na ekranu prikazani svi potrebni podaci za spajanje na PostgreSQL server. Sljedeći korak je otići u pgAdmin klijent te kreirati novu konekciju.

1. Otvoriti pgAdmin
2. Desnim klikom kliknuti na *Servers* te odabratи *Create Server...*
3. Postaviti proizvoljno ime (ne utječe na samu konekciju)
4. Odabratи karticu *Connection* te tu popuniti potrebne podatke (*Hostname/address, Maintenance database, Username, Password*)
5. Nakon popunjениh podataka potrebno je kliknuti na gumb *Save*

Sada smo stvorili novu konekciju te iz popisa servera možemo odabratи novo-kreirani te manipulirati s tablicama.

Heroku konfiguracija

Slično kao u lokalnoj konfiguraciji potrebno je igrati se s varijablama. Jedina i vrlo bitna razlika je ta da sve već posloženo te na Heroku serveru već postoji varijabla *DATABASE_URL*. Potrebno je samo pustiti backend aplikaciju u pogon.

6. Zaključak i budući rad

Naš je zadatak bio napraviti web aplikaciju koja će korisnicima omogućiti pretraživanje i rezervaciju slobodnih parkirališnih mesta u gradu Zagrebu svih tvrtki koje odluče nuditi parkiralište preko naše aplikacije.

Rad na aplikaciji podijelili smo u dvije faze: planiranje i izrada dokumentacije i generičkih funkcionalnosti aplikacije te implementacija projekta, ispitivanje i dovršetak dokumentacije.

Prvu fazu započeli smo okupljanjem razvojnog tima. Nakon što smo dobili projektni zadatak, počeli smo s dokumentiranjem i dogovaranjem kako će naša aplikacija izgledati. Tu smo naišli i na prve izazove. Nitko od nas nije radio sličan projekt i nije znao što očekivati niti kako započeti rad na takvom projektu. Mislili smo da trebamo što prije početi programirati, ali čim smo započeli shvatili smo kako prvo trebamo dobro definirati svojstva naše aplikacije, sve zahtjeve i moguće scenarije pa smo to i napravili. Izrađena dokumentacija (obrasci uporabe, dijagram baze podataka itd.) pomogla nam je da se bolje upoznamo sa zadatkom, a kasnije nudila jasne smjernice kako bi naša aplikacija trebala izgledati.

Podijelili smo se u tri grupe koje su radile na određenim dijelovima aplikacije: frontend, backend i baze podataka. Grupa zadužena za bazu podataka se nakon izrade baze pridružila *backendu* i *frontendu*. Prvu smo fazu završili implementacijom registracije i prijave za korisnike.

Druga faza većinom se sastojala od implementacije aplikacije. Jasno definirani zahtjevi napisani u dokumentaciji ubrzali su rad na aplikaciji. Svatko je znao što treba raditi i kako bi to trebalo izgledati na kraju. Na kraju druge faze dovršili smo ispitivanje aplikacije i preostalu dokumentaciju koja će poslužiti za buduće inačice sustava.

Radeći na ovom projektu morali smo riješiti mnoge izazove. Koristili smo potpuno nove tehnologije koje smo prvo morali naučiti da bi ih dobro upotrijebili. Implementacija karte, postavljanje aplikacije na javno dostupnom poslužitelju, komunikacija baze podataka, *backend* i *frontend* dijela samo su neki od izazova koje smo uspješno savladali. Naučili smo raditi u grupi, rješavati konflikte, koristiti nove tehnologije i ne bojati se probati nešto novo.

Neki izazovi su bili tehnički prezahtjevni za nas da ih riješimo u okviru ovog zadatka. Javni web poslužitelj kojeg smo koristili ne omogućuje nam da dohvaćamo trenutnu lokaciju korisnika i spremamo kolačiće na njegovu domenu. Spremanje kolačića smo riješili kupnjom vlastite domene, dok dohvaćanje trenutne lokacije sigurno ostaje kao daljnja mogućnost poboljšanja aplikacije. Uz to, poboljšanje ove aplikacije može se ostvariti izgradnjom mobilne aplikacije što je jedan od budućih proširenja sustava.

Ovaj projektni zadatak bio je vrijedno iskustvo svim članovima grupe. Iskusili smo kako je biti dio tima, raditi stvari u zadanom roku. Naučili smo koliko dobra organizacija i priprema pomažu i ubrzavaju rad na projektu, ali i koliko još imamo za učiti i da će uvijek biti mjesta za napredak.

Popis literature

Kontinuirano osvježavanje

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

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>

Indeks slika i dijagrama

| | | |
|------|--|----|
| 3.1 | Prikaz funkcionalnosti vezanih za korisničke podatke | 22 |
| 3.2 | Pregled ostalih funkcionalnosti | 23 |
| 3.3 | Sekvencijski dijagram za UC4 | 24 |
| 3.4 | Sekvencijski dijagram za UC8 | 26 |
| 3.5 | Sekvencijski dijagram za UC9 | 27 |
| 4.1 | MVC načelo | 29 |
| 4.2 | Dijagram baze podataka | 36 |
| 4.3 | Dijagram razreda za DTO (1.dio) | 38 |
| 4.4 | Dijagram razreda za DTO (2.dio) | 39 |
| 4.5 | Dijagram razreda za repo (1.dio) | 40 |
| 4.6 | Dijagram razreda za repo (2.dio) | 41 |
| 4.7 | Dijagram razreda za BaseController | 42 |
| 4.8 | Dijagram razreda za Rezervacija Controller | 42 |
| 4.9 | Dijagram razreda za Jednokratna Controller | 43 |
| 4.10 | Dijagram razreda za Ponavljajuća Controller | 43 |
| 4.11 | Dijagram razreda za Trajna Controller | 43 |
| 4.12 | Dijagram razreda za Session Controller | 44 |
| 4.13 | Dijagram razreda za User Controller | 44 |
| 4.14 | Dijagram razreda za Tvrtka Controller | 45 |
| 4.15 | Dijagram razreda za Klijent Controller | 45 |
| 4.16 | Dijagram razreda za Parkiraliste Controller | 46 |
| 4.17 | Dijagram razreda za Mappers | 46 |
| 4.18 | Dijagram razreda za Vozilo Controller | 47 |
| 4.19 | Dijagram stanja | 48 |
| 4.20 | Dijagram aktivnosti | 50 |
| 4.21 | Dijagram komponenti | 51 |
| 5.1 | Unit testovi | 54 |
| 5.2 | Kod za test mappera 1 | 55 |
| 5.3 | Kod za test mappera 2 | 55 |

| | |
|--|----|
| 5.4 Kod za test validacije računa 1 | 56 |
| 5.5 Kod za test validacije računa 2 | 56 |
| 5.6 Kod za test validacije računa 3 | 57 |
| 5.7 Kod za test metoda s datumima 1 | 57 |
| 5.8 Kod za test metoda s datumima 2 | 58 |
| 5.9 Selenium testovi | 59 |
| 5.10 Kod za test prijave klijenta | 59 |
| 5.11 Kod za test registracije tvrtka 1 | 60 |
| 5.12 Kod za test registracije tvrtka 2 | 60 |
| 5.13 Kod za test dodavanja vozila 1 | 61 |
| 5.14 Kod za test dodavanja vozila 2 | 61 |
| 5.15 Kod za test dodavanja parkinga 1 | 62 |
| 5.16 Kod za test dodavanja parkinga 2 | 62 |
| 5.17 Kod za test rezervacije 1 | 63 |
| 5.18 Kod za test rezervacije 2 | 63 |
| 5.19 Dijagram razmještaja | 64 |

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 7.listopada.2020
- Prisustvovali: (Svi): M.Ćubela, K.Djaković, L.Matković, L.Strejček, A, Krišto, K.Petković, M.Bakić
- Teme sastanka: Inicijalni sastanak
 - Detalji o tehnologijama
 - Git
 - Latex
 - Podjela tima na 3 pod tima
 - Teme za sljedeći sastanak

2. sastanak

- Datum: 9.listopada.2020
- Prisustvovali: Svi
- Teme sastanka: Odabir tehnologija za projekt
 - Odabir tehnologije
 - AWS
 - ORM
 - JS
 - Teme za sljedeći sastanak

3. sastanak

- Datum: 10.listopada.2020
- Prisustvovali: M.Ćubela, K. Djaković, K.Petković
- Teme sastanka: Inicijalizacija backenda
 - Dogovor o okvirnoj strukturi backenda
 - Raspodjela poslova
 - Pitanja za asistenticu
 - Teme za sljedeći backend sastanak

4. sastanak

- Datum: 10.listopada.2020
- Prisustvovali: K.Djaković, M.Bakić, M.Ćubela
- Teme sastanka: Proučavanje AWS-a
 - Izrada timskog računa (Gmail, AWS)
 - Istrživanje mogućnosti "deploy"-a
 - PostgreSQL - AWS

5. sastanak

- Datum: 12.listopada.2020
- Prisustvovali: K.Djaković, M.Bakić
- Teme sastanka: Odabir tehnologija za projekt
 - Uspostava servera(AWS)

6. sastanak

- Datum: 12.listopada.2020
- Prisustvovali: M.Ćubela, K.Petković
- Teme sastanka: Podjela poslova na backendu
 - Pregled zadataka za backend
 - Rasподjela poslova
 - Pitanja za asistenticu
 - Teme za sljedeći backend sastanak

7. sastanak

- Datum: 14.listopada.2020
- Prisustvovali: Svi
- Teme sastanka: Okvirno definiranje zahtjeva
 - Funkcionalni zahtjevi
 - Nedefinirane radnje
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

8. sastanak

- Datum: 16.listopada.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled primjera projektne dokumentacije
 - Povezivanje entiteta u bazi
 - Pitanja za asistenticu

- Teme za sljedeći sastanak

9. sastanak

- Datum: 21.listopada.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled napretka
 - Podjela zadataka u vezi dokumentacije
 - Teme za sljedeći sastanak

10. sastanak

- Datum: 28.listopada.2020
- Prisustvovali: K.Djaković, L.Matković, M.Bakić, L.Strejček
- Teme sastanka: Podizanje baze, Heroku
 - Prebacivanje s AWS na Heroku
 - Izrada računa
 - Podizanje servera na Heroku

11. sastanak

- Datum: 27.listopada.2020
- Prisustvovali: L.Strejček, A.Krišto
- Teme sastanka: Izrada ER modela
 - Definiranje i izrada ER modela
 - Pregledavanje baze
 - Podjela zadataka
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

12. sastanak

- Datum: 30.listopada.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Dogovor o nesuglasnostima
 - Entitet "administrator"
 - Odluka o korištenju ORM-a
 - UML dijagrami
 - UC - ovi
 - Podjela zadataka
 - Pitanja za asistenticu

13. sastanak

- Datum: 3.studeni.2020
- Prisustvovali: L.Strejček, A.Krišto
- Teme sastanka: Izrada ER modela
 - Definiranje i izrada ER modela
 - Pregledavanje baze
 - Podjela zadataka
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

14. sastanak

- Datum: 4.studeni.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Proučavanje ORM-a
 - Podjela zadataka
 - Pitanja za asistenticu

15. sastanak

- Datum: 5.studeni.2020
- Prisustvovali: L.Strejček, A.Krišto
- Teme sastanka: Dokumentacija baze
 - Pregled strukture dokumentacije
 - Raspodjela poslova
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

16. sastanak

- Datum: 9.studeni.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled izvršenih zadataka
 - Podjela zadataka
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

17. sastanak

- Datum: 9.studeni.2020
- Prisustvovali: Svi + Nikolina Frid

- Teme sastanka: Prezentacija trenutnog stanja
 - Prikaz stanja aplikacije
 - Prikaz stanja dokumentacije
 - Dogovor o koracima napretka
 - Pitanja za asistenticu

18. sastanak

- Datum: 12.studeni.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dokumentacije
 - Provjera pogrešaka u dokumentaciji
 - Pogreška u korištenoj tehnologiji za backend
 - Dogovor o prebacivanju cijelog backenda u TypeScript
 - Ispravljanje dokumentacije
 - Podjela zadataka
 - Pitanja za asistenticu

19. sastanak

- Datum: 13.studeni.2020
- Prisustvovali: K.Djaković, M.Ćubela, K.Petković
- Teme sastanka: Restrukturiranje cijelokupnog backenda u OO paradigmu
 - Dogovor o načinu prebacivanja u TypeScript
 - Raspodjela poslova prebacivanja
 - Pitanja za asistenticu
 - Dogovor za sljedeći sastanak

20. sastanak

- Datum: 14.studeni.2020
- Prisustvovali: K.Djaković, M.Ćubela, K.Petković
- Teme sastanka: Pregled napretka restrukturiranja backenda
 - Pregled napretka
 - Ispravak pogrešaka
 - Dogovor o načinu nastavljanja rekonstrukcije
 - Podjela poslova

21. sastanak

- Datum: 22.studeni.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka

- Pregled izvršenih zadataka
- Podjela zadataka
- Pitanja za asistenticu
- Teme za sljedeći sastanak

22. sastanak

- Datum: 27.studeni.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled izvršenih zadataka
 - Podjela zadataka
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

23. sastanak

- Datum: 3.prosinca.2020
- Prisustvovali: Svi + dr. sc. Nikolina Frid
- Teme sastanka: Kolokviranje prvog dijela projekta
 - Prikaz aplikacije i dokumentacije
 - Ispitivanje pojedinačnih članova tipa
 - Isticanje pogrešaka
 - Dogovor za sljedeće napretke

24. sastanak

- Datum: 7.prosinca.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled izvršenih zadataka
 - Podjela zadataka
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

25. sastanak

- Datum: 15.prosinca.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled izvršenih zadataka
 - Podjela zadataka
 - Pitanja za asistenticu

- Teme za sljedeći sastanak

26. sastanak

- Datum: 22.prosinca.2020
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled izvršenih zadataka
 - Podjela zadataka
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

27. sastanak

- Datum: 3.siječnja.2021
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled izvršenih zadataka
 - Podjela zadataka
 - Pitanja za asistenticu
 - Teme za sljedeći sastanak

28. sastanak

- Datum: 7.siječnja.2021
- Prisustvovali: Svi
- Teme sastanka: Sastanak prije demonstracije alpha inačice
 - Pregled aplikacije
 - Pregled dokumentacije
 - Ispravljanje detalja
 - Pitanja za asistenticu

29. sastanak

- Datum: 8.siječnja.2021
- Prisustvovali: Svi + dr. sc. Nikolina Frid
- Teme sastanka: Demonstracija alpha inačice
 - Demonstracija aplikacije
 - Prikaz dokumentacije
 - Sljedeći koraci

30. sastanak

- Datum: 12.siječnja.2021
- Prisustvovali: Svi

- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled izvršenih zadataka
 - Podjela zadataka
 - Teme za sljedeći sastanak

31. sastanak

- Datum: 14.siječnja.2021
- Prisustvovali: Svi
- Teme sastanka: Pregled dosadašnjeg napretka
 - Pregled izvršenih zadataka
 - Pregled dokumentacije
 - Podjela zadataka
 - Teme za sljedeći sastanak

32. sastanak

- Datum: 15.siječnja.2021
- Prisustvovali: Svi
- Teme sastanka: Pregled dokumentacije
 - Pregled dokumentacije
 - Podjela zadataka nadopune i ispravaka
 - Termin za sljedeći sastanak

33. sastanak

- Datum: 18.siječnja.2021
- Prisustvovali: Svi
- Teme sastanka: Sastanak prije kolokviranja
 - Pregled cijelokupnog projekta
 - Završni ispravci

34. sastanak

- Datum: 20.siječnja.2021
- Prisustvovali: Svi + dr. sc. Nikolina Frid
- Teme sastanka: Kolokviranje projekta
 - Demonstracija završne aplikacije
 - Završna dokumentacija
 - Pojedinačno ispitivanje članova tima

Tablica aktivnosti

| | Marin Ćubela | Marko Bakić | Kristian Djaković | Andrea Krišto | Lea Matković | Kristina Petković | Lucija Strejček |
|----------------------------------|--------------|-------------|-------------------|---------------|--------------|-------------------|-----------------|
| Upravljanje projektom | 8 | | | | | | |
| Opis projektnog zadatka | | | | | 4 | | 4 |
| Funkcionalni zahtjevi | 1 | 3 | 3 | 1 | 1 | 3 | 1 |
| Opis pojedinih obrazaca | 2 | 6 | 6 | 6 | 6 | 6 | 6 |
| Dijagram obrazaca | 4 | | | 4 | | | |
| Sekvencijski dijagrami | | | 5 | | 4 | | 4 |
| Opis ostalih zahtjeva | | | | | | 4 | |
| Arhitektura i dizajn sustava | | 2 | | | | 4 | |
| Baza podataka | 1 | 1 | | 4 | | | 4 |
| Dijagram razreda | 5 | 10 | | 10 | | | |
| Dijagram stanja | | | 3 | | 4 | | |
| Dijagram aktivnosti | | | | | 3 | | 3 |
| Dijagram komponenti | | | | | 3 | | 3 |
| Korištene tehnologije i alati | | | 4 | | | | |
| Ispitivanje programskog rješenja | 2 | | | | | 5 | |
| Dijagram razmještaja | | | | | | 3 | |
| Upute za puštanje u pogon | | | 4 | | | | |
| Dnevnik sastajanja | | 3 | | | | | |
| Zaključak i budući rad | 2 | | | | | | |
| Popis literature | | | | | | | |
| Dodatne funkcionalnosti | | | | | | | |
| <i>izrada baze podataka</i> | | | | 20 | | | 20 |
| <i>izrada frontend-a</i> | | 130 | 60 | | 140 | | 120 |
| <i>izrada backend-a</i> | 140 | | 60 | 120 | | 140 | |
| <i>deploy aplikacije</i> | | 10 | 20 | | | | |

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.