

Programsko inženjerstvo

Ak. god. 2023./2024.

SpotPicker

Dokumentacija, Rev. 2

Grupa: *Lukax4*

Voditelj: *Luka Diktić*

Datum predaje: 19. 1. 2024.

Nastavnik: *Hrvoje Nuić, mag. ing.*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.2 Dijagram obrazaca uporabe	17
3.2.1 Sekvencijski dijagrami	19
3.3 Ostali zahtjevi	21
4 Arhitektura i dizajn sustava	22
4.1 Baza podataka	25
4.1.1 Opis tablica	25
4.1.2 Dijagram baze podataka	29
4.2 Dijagram razreda	29
4.3 Dijagram stanja	35
4.4 Dijagram aktivnosti	36
4.5 Dijagram komponenti	37
5 Implementacija i korisničko sučelje	38
5.1 Korištene tehnologije i alati	38
5.2 Ispitivanje programskog rješenja	40
5.2.1 Ispitivanje komponenti	40
5.2.2 Ispitivanje sustava	41
5.3 Dijagram razmještaja	43
5.4 Upute za puštanje u pogon	45
6 Zaključak i budući rad	47
Popis literature	48

Indeks slika i dijagrama	49
Dodatak: Prikaz aktivnosti grupe	50

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Luka Žmak	25.10.2023.
0.2	Dopisane upute za povijest dokumentacije. Dodane reference.	Maja Mavračić	24.08.2013.
0.5	Dodan <i>Use Case</i> dijagram i jedan sekvencijski dijagram, funkcionalni i nefunkcionalni zah-tjevi i dodatak A	Mirta Pos-njak	26.10.2023.
0.6	Arhitektura i dizajn sustava, algoritmi i strukture podataka	Filip Sučić	27.10.2013.
0.8	Povijest rada i trenutni status implementa-cije, Zaključci i plan dalnjeg rada	*Dominik Poljak	2.11.2023.
0.9	Opisi obrazaca uporabe	Luka Dikić	07.11.2023.
0.10	Preveden uvod	Luka Babić	10.11.2023.
0.11	Sekvencijski dijagrami	Mirta Pos-njak	15.11.2023.
0.12.2	Napravljen dijagonala razreda	Maja Mavračić	16.11.2023.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	svi	17.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.1	Implementacija	Luka Žmak	25.11.2023.
1.2	Dodani dijagram stanja i dijagram aktivnosti	Luka Diktić	25.11.2023.
1.3	Dodan dijagram razmjestaja	Maja Mavračić	5.1.2024.
1.4	Dodan opis tehnologija i alata	Mirta Posnjak	5.1.2024.
1.5	Dodan dio korisnickih uputa	Dominik Poljak	10.1.2024.
1.6	Dodan dijagram komponenti	Filip Sučić	15.1.2024.
1.7	Konacne korisničke upute	Luka Babić	15.1.2024.
1.8	Dodana potrebna instalacija	Luka Žmak	17.1.2024.
1.9	Pregledano i dodane slike s opisima	Luka Diktić	17.1.2024.
2.0	Konacna verzija		

2. Opis projektnog zadatka

Cilj projekta je napraviti web aplikaciju “SpotPicker” koja bi omogućila vozačima osobnih automobila i bicikla da unaprijed rezerviraju svoje parkirno mjesto u garaži i time se riješe svakodnevne muke lutanja po parkingu i traženja mesta.

S korisničke strane, aplikacija se otvara u neregistriranom obliku. Neregistrirani korisnik može pregledavati parkirališta i njihova parkirna mjesta, ali bez informacije o dostupnosti. Zato se novim korisnicima nudi mogućnost registracije, odnosno izrade korisničkog računa i mogućnost prijave za one koji već imaju izrađen račun. Za registraciju su potrebni:

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

Prilikom prve registracije potrebno je potvrditi svoje podatke preko poruke poslane na prethodno unesenu e-mail adresu. Za prijavu su potrebni samo korisničko ime i lozinka. Najvažnija dodatna mogućnost koju aplikacija nudi prijavljenim korisnicima je pregled dostupnih parkirnih mjesta u realnom vremenu. Puni postupak rezervacije parkirnog mesta izgleda ovako:

1. korisnik na karti grada bira lokaciju do koje želi doći
2. aplikacija mu na temelju njegovog odredišta bira najbliži parking

Sljedeći se korak može odviti na dva načina:

3. a) korisnik bira parkirna mjesta koja mu se sviđaju, a zatim se otvara kalendar u kojem može točno odabrati termin rezervacije (datum i vrijeme)

3. b) korisnik bira termin rezervacije (datum i vrijeme), a zatim mu se prikazuju parkirna mjesta koja će tada biti dostupna

Ovdje je bitno napomenuti da i u a) i u b) varijanti najranija moguća rezervacija je dan nakon onog u kojem korisnik vrši rezervaciju. Znači nije moguće rezervirati u okviru dana rezervacije.

4. korisnik sada dobiva dodatnu mogućnost odabira želi li da mu rezervacija bude ponavljajuća ili ne. To je pogotovo pogodno ljudima koji znaju da imaju neke obaveze na npr. tjednoj bazi

5. korisniku se daje opcija plaćanja karticom unaprijed ili plaćanje prilikom dolaska na parking

6. potvrđuje se rezervacija parkirnog mesta, termina, trajanja i načina plaćanja

Korisniku se pruža mogućnost plaćanja karticom, uplaćivanjem novca u novčanik unutar aplikacije nakon kojeg se sredstva mogu koristiti u bilo kojem trenutku. Drugi način štedi vrijeme jer nije potrebno unositi podatke o kartici i vršiti potvrde prilikom svake transakcije.

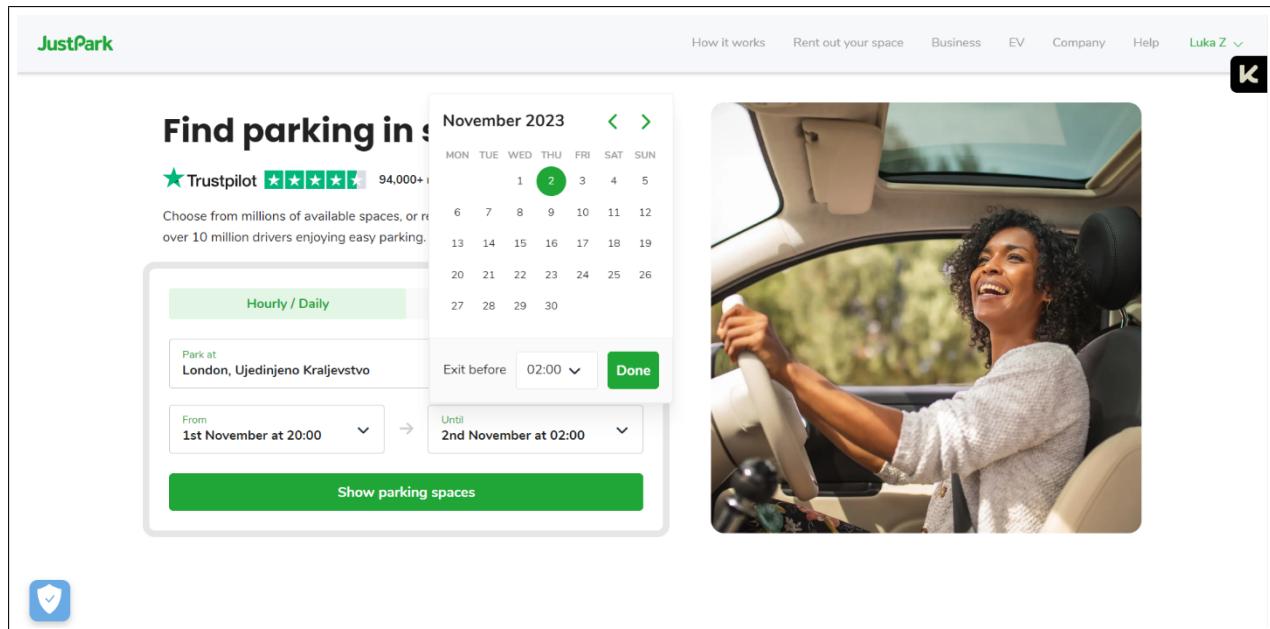
Što se tiče parkirnih mesta za bicikle, njih nije potrebno rezervirati jer se ne naplaćuju. Moguće je jedino pratiti u realnom vremenu ukupan broj dostupnih mesta.

U slučaju da se korisnik prijavljuje kao voditelj parkinga, da bi dobio voditeljska prava prvo je potrebno odobrenje administratora. Voditelj ima mogućnost promjene podataka o svom parkingu. To uključuje izmjenu imena parkinga, dodavanje i uklanjanje slika te promjene cijena. Također može dodavati nova mjesta u slučaju proširenja parkinga.

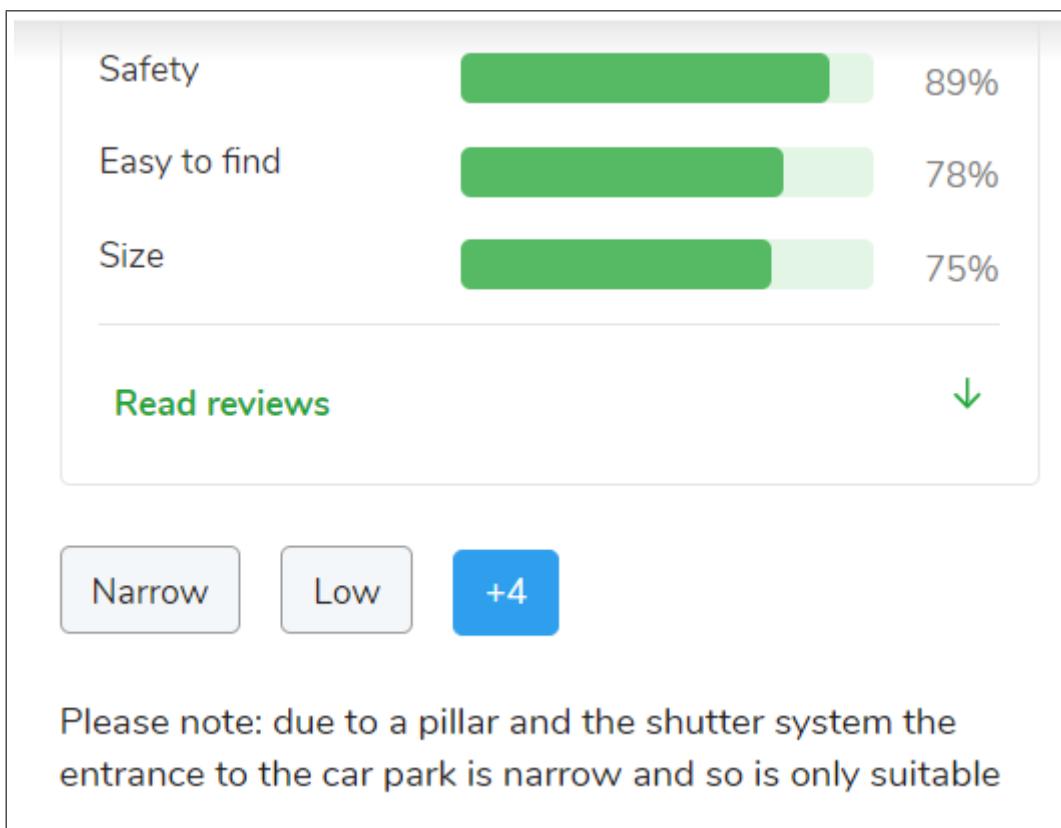
Uloga administratora donosi pravo pregledavanja i mijenjanja osobnih podataka registriranih korisnika. Također, administrator potvrđuje identitet vodi-

telja parkinga prilikom prijave u aplikaciju.

Slično programsko rješenje nudi aplikacija JustPark koja omogućuje rezervaciju parkirnih mjesta u Ujedinjenom Kraljevstvu. Glavno sučelje (slika1) nudi jednastavnu mogućnost odabira lokacije i termina. Zatim se otvara popis parkirališta sortiranih po udaljenosti koju treba pješice prijeći od parking do finalne destinacije. SpotPicker se razlikuje od JustParka po tome što JustPark nudi ostavljanje komentara i ocijenjivanje parkinga (slika2) po raznim kriterijima poput sigurnosti, lakoće snalaženja, širine mjesta itd.



Slika 2.1: Sučelje za odabir termina u JustPark



Slika 2.2: Dodatna mogućnost ocjene parkinga u JustPark

SpotPicker je aplikacija savršena za ljude koji planiraju svoja putovanja ili obaveze po nekoliko dana unaprijed jer će si rezervacijom mjesta ukloniti onaj najgori dio vožnje, a to je lutanje u potrazi za mjestom. Također, pogodno je ljudima koji moraju negdje doći na vrijeme jer ovako mogu značajno bolje procijeniti koliko im je vremena potrebno u vožnji. Isto tako, aplikaciju mogu korisiti i ljudi kojima je parking potreban stalno (npr. jer rade u blizini) jer je moguće da rezervacija bude ponavljajuća iz tjedna u tjedan.

Naravno, bitno je napomenuti da je ovaj projekt moguće nadograditi i poboljšati neke njegove stavke. Tako bi korisni dodatak bio mogućnost poništavanja rezervacije uz povrat novca.

Što se tiče prilagodbe raznim tržištima, za manje gradove koji imaju puno privatnih kuća u centru, bila bi dobra mogućnost da se vlasnici kuća s dvorištem mogu prijaviti da iznajmljuju svoje dvorište kao parkirno mjesto. Vlasnik dvorišta bi se jednak vodio kao i vlasnik većeg parkirališta, a odabir mesta bi djelovao na isti način kao i kod velikih parkinga ovisno o veličini dvorišta. Bitno je primijetiti da projekt uključuje rješenje za parkirališta s jednom razinom, tako da bi jedno dodatno moguće proširenje bilo dodati mogućnost odabira kata garaže i tek onda parkirnog mesta.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Korisnik
2. Administrator
3. Vlasnik parkirališta(naručitelj)
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) na karti pregledati lokacije parkirališta i parkirališna mjesta
 - (b) stvoriti korisnički račun, tj. registrirati se za šta su mu potrebni korisničko ime, lozinka, ime, prezime, slika osobne iskaznice, IBAN i email adresa
2. Registrirani korisnik (sudionik) može:
 - (a) prijaviti se u sustav koristeći korisničko ime i lozinku
 - (b) na karti odabrati adresu do koje želi doći
 - (c) na karti pregledati lokacije parkirališta, parkirališna mjesta i dostupna parkirna mjesta
 - (d) rezervirati parkirno mjesto, termin i trajanje za koje je zainteresiran
 - (e) platiti rezervaciju parkinga
3. Vlasnik parkinga (sudionik) može:
 - (a) unijeti i mijenjati informacije o svom parkiralištu (naziv, opis, fotografija, cjenik)
 - (b) dodavati nova parkirna mjesta
 - (c) vidjeti statistiku zauzetosti parkirališta i parkirališnih mjesta kroz vrijeme

4. Administrator (sudionik) može:

- (a) pregledavati osobne podatke registriranih korisnika
- (b) mijenjati osobne podatke registriranih korisnika
- (c) potvrditi prijavu korisnicima koji se prijavljuju kao vlasnik parkinga

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - registracija korisnika

- **Glavni sudionik:** neregistrirani korisnik
- **Cilj:** izraditi korisnički račun
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. korisnik odabire opciju za registraciju
 2. korisnik upisuje potrebne podatke
 3. u bazu podataka se upisuje novi korisnik
 4. web stranica se preusmjerava na stranicu za prijavu
- **Opis mogućih odstupanja:**
 - 2.a korisničko ime i/ili email su već zauzeti
 1. korisnik dobiva povratnu informaciju o specifičnoj greški i preusmjerjen je nazad na stranicu za registraciju
 - 2.b upisana lozinka nije važeća(nema barem 8 znakova ili ne sadrži barem jedno slovo i brojku)
 1. korisnik dobiva povratnu informaciju o specifičnoj greški i preusmjerjen je nazad na stranicu za registraciju

UC2 -prijava korisnika

- **Glavni sudionik:** neprijavljeni korisnik
- **Cilj:** prijava u korisnički račun
- **Sudionici:** baza podataka
- **Preduvjet:** prethodna registracija
- **Opis osnovnog tijeka:**
 1. korisnik odabire opciju za prijavu
 2. korisnik upisuje svoje korisničko ime i lozinku
 3. korisnik je preusmjeren na početnu stranicu sa svim ovlastima prijavljenog korisnika
- **Opis mogućih odstupanja:**
 - 2.a upisana kombinacija korisničkog imena i lozinke je neispravna
 1. korisnik je vraćen na stranicu za prijavu

UC3-promjena lozinke

- **Glavni sudionik:** registrirani korisnik
- **Cilj:** izmijeniti lozinku na već postojećem korisničkom računu
- **Sudionici:** baza podataka
- **Preduvjet:** prethodna registracija
- **Opis osnovnog tijeka:**
 1. korisnik izabire opciju za mijenjanje lozinku
 2. korisnik upisuje email adresu na koju želi da mu se pošalje verifikacijski kod
 3. korisnik upisuje verifikacijski kod
 4. korisnik upisuje novu lozinku i potvrđuje ju ponovnim upisom
 5. korisnik je preusmjeren na stranicu za prijavu
- **Opis mogućih odstupanja:**
 - 3.a upisani verifikacijski kod nije isti onom poslanom na email
 1. dolazi poruka o grešci i korisnik može opet upisati kod
 - 4.a lozinke se ne poklapaju
 1. dolazi poruka o grešci i korisnik može opet upisati lozнике
- 4.b upisana lozinka nije važeća(nema barem 8 znakova ili ne sadrži barem jedno slovo i brojku)
 1. dolazi poruka o grešci i korisnik može opet upisati loznike

UC4-odabir parkirališta

- **Glavni sudionik:** prijavljeni korisnik
- **Cilj:** izabrati parkiralište na kojem će kasnije rezervirati mjesto
- **Sudionici:** baza podataka
- **Preduvjet:** prethodna prijava
- **Opis osnovnog tijeka:**
 1. korisnik upisuje adresu do koje želi doći
 2. korisnik klikne gumb za pretraživanje
 3. korisniku se na karti iscrta ruta do parkirališta koje je najbliže njegovom odredištu

UC5.1 -rezervacija mjesta s uvjetom određenog mesta

- **Glavni sudionik:** prijavljeni korisnik
- **Cilj:** rezervirati specifično parkirno mjesto

- **Sudionici:** baza podataka
- **Preduvjet:** odabрано parkiralište
- **Opis osnovnog tijeka:**
 1. na karti parkirališta odabire mjesto za koje je zainteresiran
 2. u kalendaru izabire datum
 3. korisniku se prikazuju slobodni sati za odabranu mjesto i datum
 4. korisnik bira datum rezervacije između ponuđenih datuma
 5. korisnik potvrđuje rezervaciju
 6. web stranica preusmjerava korisnika na stranicu za plaćanje
- **Opis mogućih odstupanja:**
 - 4.a sva vremena za taj datum i mjesto su zauzeta
 1. korisnik sam mora odabrati neki drugi datum ili mjesto

UC5.2-rezervacija mjesta s uvjetom datuma

- **Glavni sudionik:** prijavljeni korisnik
- **Cilj:** rezervirati mjesto u specifičnom terminu
- **Sudionici:** baza podataka
- **Preduvjet:** odabranо parkiralište
- **Opis osnovnog tijeka:**
 1. korisnik klikne na ikonu parkirališta
 2. korisnik u kalendaru odabire datum i vrijeme početka parkiranja
 3. korisnik u kalendaru odabire datum i vrijeme završetka parkiranja
 4. korisniku se prikažu dostupna mjesta za odabrani termin na karti parkirališta
 5. korisnik odabire mjesto na karti
 6. korisnik potvrđuje rezervaciju
 7. web stranica se preusmjerava na stranicu za plaćanje
- **Opis mogućih odstupanja:**
 - 4.a niti jedno mjesto se ne prikazuje kao dostupno
 1. korisnik mora odabrati drugi termin
 - 5.a korisnik ne želi platiti karticom, već dolaskom na parking
 1. korisnik će dolaskom na parking platiti rezervaciju

UC6.1-plaćanje prilikom rezervacije

- **Glavni sudionik:** prijavljeni korisnik
- **Cilj:** plaćanje karticom prilikom rezervacije

- **Sudionici:** baza podataka
- **Preduvjet:** odabrano parkirališno mjesto
- **Opis osnovnog tijeka:**
 1. Korisnik odabire mogućnost plaćanja karticom prilikom rezervacije
 2. Korisnik unosi podatke s kartice
 3. Nakon uspješnog plaćanja mjesto je rezervirano
- **Opis mogućih odstupanja:**
 - 2.a odbijena kartica
 1. pojavljuje se poruka o nevaljanosti kartice

UC6.2-plaćanje sredstvima u novčaniku aplikacije

- **Glavni sudionik:** prijavljeni korisnik
- **Cilj:** plaćanje parkirnog mjesta sredstvima unutar aplikacije
- **Sudionici:** baza podataka
- **Preduvjet:** registracija korisnika, dodavanje novaca u novčanik, odabir parkirnog mjesta
- **Opis osnovnog tijeka:**
 1. Korisnik sa svog računa uplaćuje željeni iznos u novčanik aplikacije
 2. Korisnik prilikom rezervacije odabire način plaćanja sredstvima iz novčanika aplikacije
 3. Nakon uspješnog plaćanja mjesto je rezervirano
- **Opis mogućih odstupanja:**
 - 2.a nedovoljno sredstva u novčaniku
 1. pojavljuje se poruka o nedovoljnoj količini sredstava u novčaniku

UC7-dodavanje novaca u novčanik aplikacije

- **Glavni sudionik:** prijavljeni korisnik
- **Cilj:** korištenje sredstava novčanika aplikacije za plaćanje
- **Sudionici:** baza podataka
- **Preduvjet:** prijava korisnika
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za dodavanje sredstava u novčanik aplikacije
 2. Korisnik upisuje podatke kartice te iznos koji želi uplatiti
 3. Nakon uspješne uplate stanje u novčaniku aplikacije se ažurira

UC8-izmjena osobnih podataka korisnika

- **Glavni sudionik:** administrator
- **Cilj:** izmjena osobnih podataka postojećih korisnika
- **Sudionici:** baza podataka
- **Preduvjet:** prijava korisnika kao administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregled korisnika
 2. Administrator otvara željeni profil
 3. Administrator odabire opciju uredi
 4. Administrator izmjenjuje podatke
 5. Promjene se ažuriraju u bazi podataka

UC9-potvrđivanje prijave vlasniku

- **Glavni sudionik:** administrator
- **Cilj:** potvrda uspješne prijave voditelju
- **Sudionici:** baza podataka
- **Preduvjet:** prijava korisnika kao administratora
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregled korisnika
 2. Administrator otvara željeni profil
 3. Administrator odabire opciju potvrdi voditelja
 4. Promjene se ažuriraju u bazi podataka

UC10-unos podataka o parkirnom mjestu

- **Glavni sudionik:** voditelj parkinga
- **Cilj:** unos podataka o parkirnom mjestu
- **Sudionici:** baza podataka
- **Preduvjet:** registracija korisnika, voditeljski račun
- **Opis osnovnog tijeka:**
 1. Voditelj otvara popis parkirališnih mjesta
 2. Voditelj odabire željeno parkirališno mjesto
 3. Voditelj odabire opciju uredi
 4. Voditelj unosi naziv, opis, fotografiju, cijenu itd
 5. Promjene se ažuriraju u bazi podataka

UC11-postavljanje statusa dostupnosti parkirnog mjeseta

- **Glavni sudionik:** voditelj parkinga

- **Cilj:** ucrtava dostupna parkirališna mjesta u kartu
- **Sudionici:** baza podataka
- **Preduvjet:** registracija korisnika, potvrda uloge voditelja
- **Opis osnovnog tijeka:**
 1. Voditelj otvara kartu parkirališnih mjesta
 2. Voditelj odabire željeno parkirališno mjesto
 3. Voditelj odabire opciju dostupno/nedostupno
 4. Promjene se ažuriraju u bazi podataka

UC12-pregledavanje statistike za parkiralište

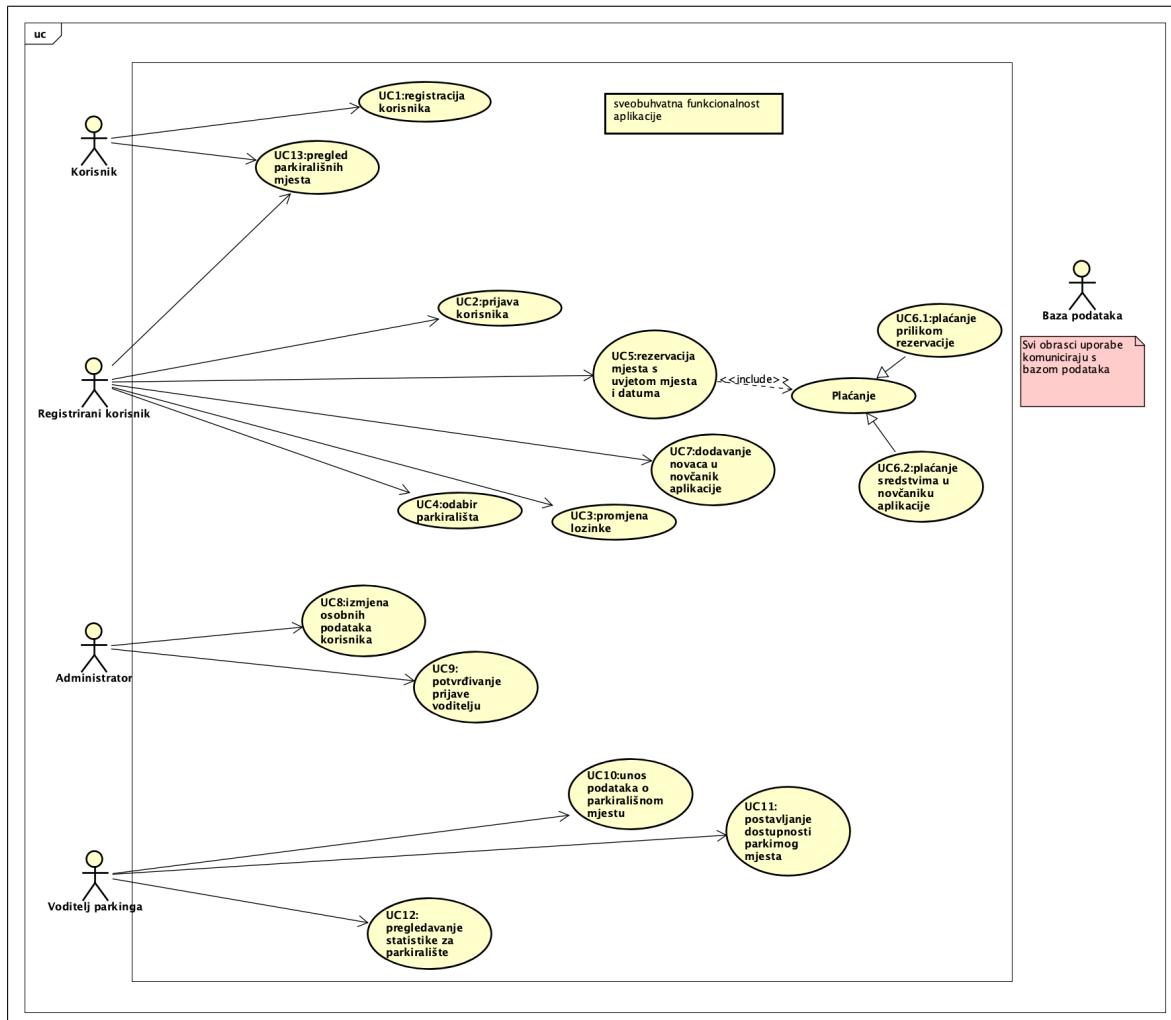
- **Glavni sudionik:** voditelj parkinga
- **Cilj:** pregled statistike zauzetosti parkirališnog mesta
- **Sudionici:** baza podataka
- **Preduvjet:** registracija korisnika, potvrda uloge voditelja
- **Opis osnovnog tijeka:**
 1. Voditelj otvara kartu parkirališnog mesta
 2. Voditelj odabire željeno parkirališno mjesto
 3. Voditelj odabire opciju statistika
 4. Prikazuje se statistika zauzetosti parkirališnog mesta u obliku gafa

UC13 -pregled parkirališnih mjesta

- **Glavni sudionik:** korisnik
- **Cilj:** pregled parkirališnih mjesta i njihova dostupnost
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire kartu parkirališta
 2. Otvara se karta sa prikazom dostupnih mjestas

3.2 Dijagram obrazaca uporabe

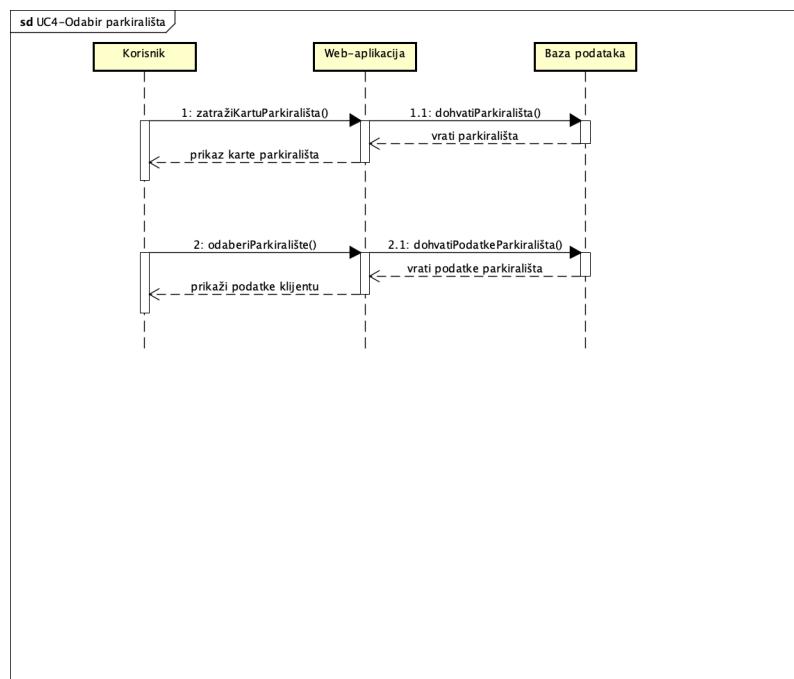
Na slici je prikazan dijagram obrazaca uporabe točnije funkcionalnosti korisnika, registriranog korisnika, administratora te voditelja parkinga.



3.2.1 Sekvencijski dijagrami

Obrazac uporabe UC4 - Odabir parkirališta

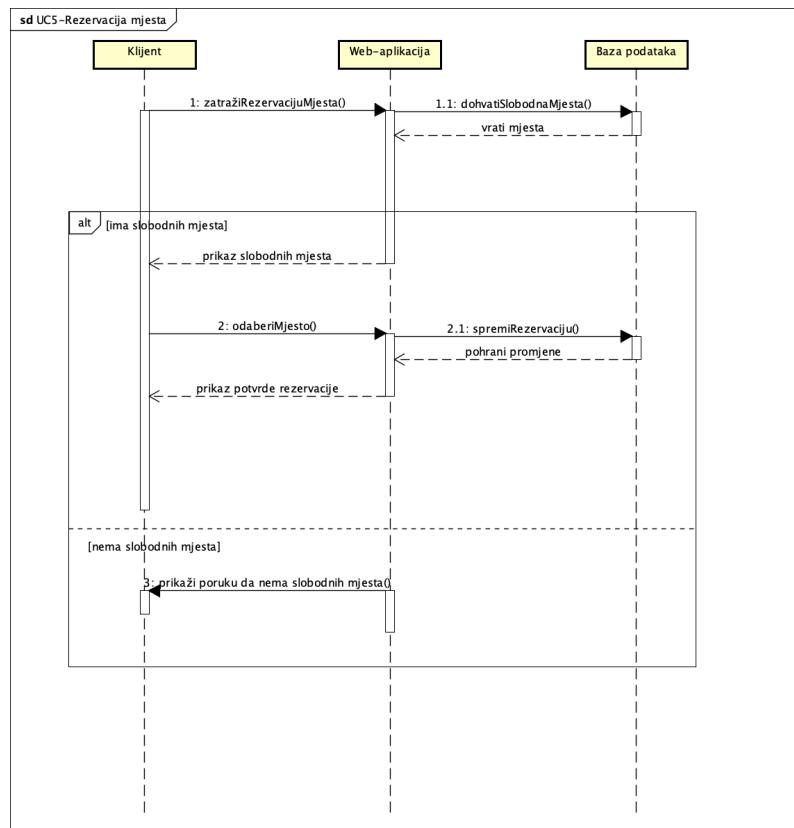
Klijent, nakon što pokrene aplikaciju, šalje zahtjev za pregled svih dostupnih parkirališta. Aplikacija, koristeći bazu podataka, dohvaća popis parkirališta te ih prikazuje korisniku. Nakon pregleda, klijent odabire specifično parkiralište koje ga zanima. Aplikacija zatim prikazuje dodatne informacije o odabranom parkiralištu, uključujući naziv, opis, fotografiju, cjenik i slobodna parkirališna mjesta.



Slika 3.1: Sekvencijski dijagram za UC4

Obrazac uporabe UC5 - Rezervacija mjesta

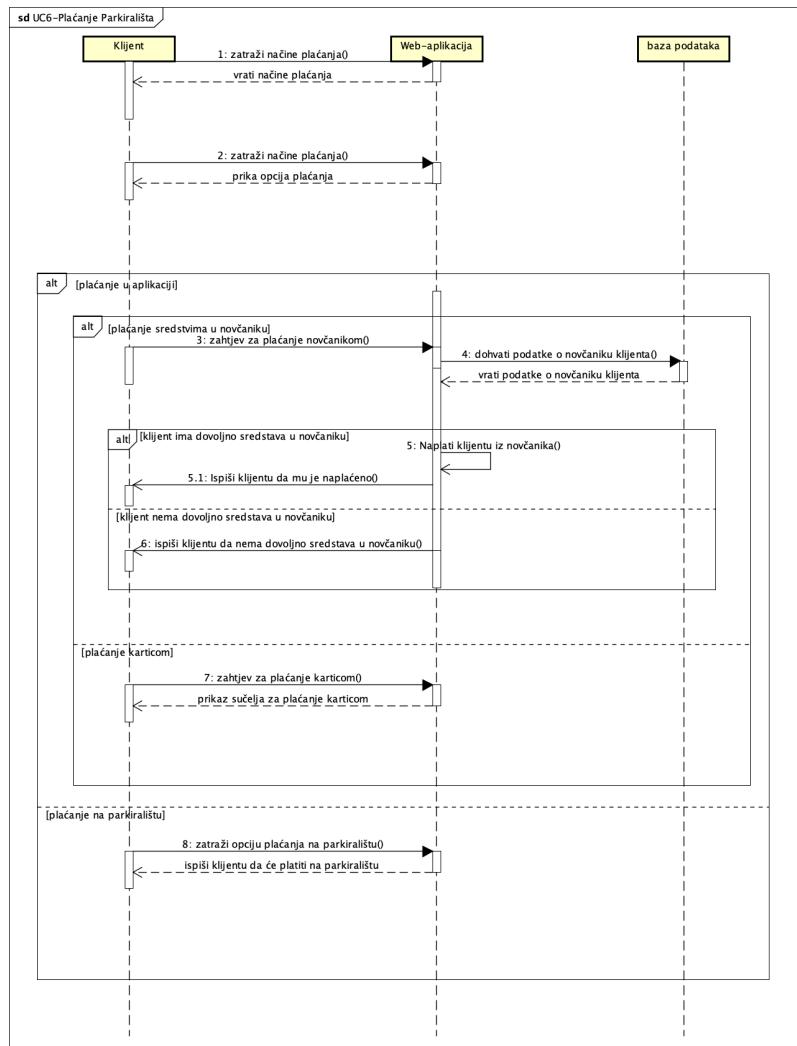
Klijent, nakon što je odabrao željeno parkiralište, pristupa informacijama o dostupnim parkirališnim mjestima. Aplikacija mu prikazuje kartu parkirališta s označenim slobodnim i zauzetim mjestima. Tada klijent šalje upit za odabranim mjestom, a aplikacija provjerava raspoloživost tog mesta. Ukoliko je mjesto slobodno, klijentu se potvrđuje rezervacija, a sustav ažurira informacije o zauzetosti i evidentira rezervaciju. U slučaju da mesta nisu slobodna, aplikacija obavještava klijenta o nedostupnosti te ga vraća na korak odabira novih mesta ili termina.



Slika 3.2: Sekvencijski dijagram za UC5

Obrazac uporabe UC6 - Plaćanje parkirališta

Kada klijent odluči platiti parkiranje, sustav prikazuje opcije plaćanja prilikom rezervacije ili prilikom dolaska na lokaciju parkirališta. Klijent odabire željeni način plaćanja. Ako odabere plaćanje unutar aplikacije, tada aplikacija provjerava stanje novčanika. Rezervacija će se naplatiti ako klijent ima dovoljno sredstava u novčaniku, a u protivnom će mu se prikazati da treba nadopлатit svoj novčanik. Ako se odluči za opciju plaćanja parkinga na parkiralištu, tada će mu aplikacija to prihvati i potvrditi.

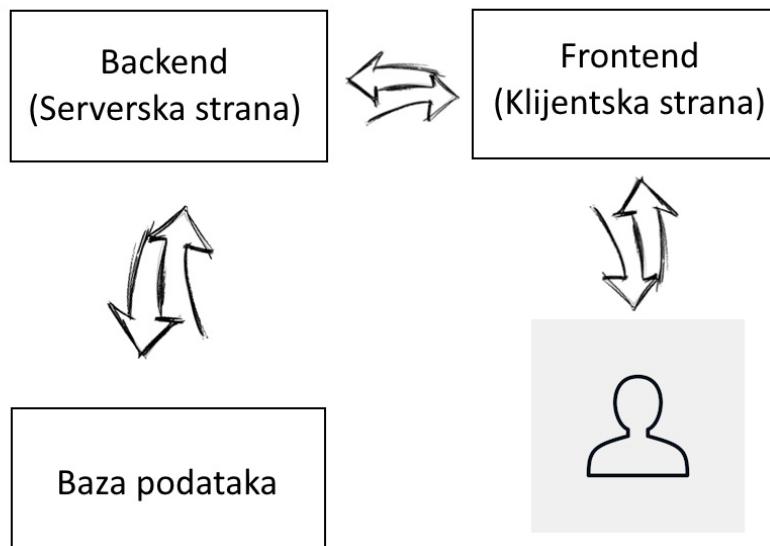


Slika 3.3: Sekvencijski dijagram za UC6

3.3 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu.
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja.
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi.
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orientirane jezike.
- Sustav kao valutu koristi euro.

4. Arhitektura i dizajn sustava



Slika 4.1: Arhitektura sustava

Korisnik aplikaciji pristupa putem web preglednika. Interakciju s aplikacijom ostvaruje preko korisničkog sučelja pomoću kojeg šalje zahtjeve web poslužitelju i prima odgovore.

Programski jezik pomoću kojeg je ostvaren backend web aplikacije je C#, a korišteni radni okvir je .NET. Frontend aplikacije ostvaren je programskim jezikom TypeScript i radnim okvirom Angular. Za razvojno okruženje backend-a odabran je Visual Studio, a frontend-a Visual Studio Code.

.NET je radni okvir namijenjen stvaranju mikroservisa. Mikroservisi su arhitektura razvoja softvera koja podrazumijeva izgradnju aplikacije pomoću niza malih, nezavisnih servisa. Svaki mikroservis obavlja određeni posao i može komunicirati s drugim mikroservisima preko standardiziranih protokola. Primjena mikroservisne arhitekture u razvoju web aplikacija omogućuje agilniji pristup razvoju, povećava skalabilnost, pojednostavljuje održavanje i prilagodljivost sustava promjenama. Svaki mikroservis može se smatrati zasebnim servisom unutar web

aplikacije, obavljajući specifične funkcionalnosti, poput autentifikacije, upravljanja korisnicima, poslovnih logika, itd.

Arhitektura sustava temelji se na MVC (Model-View-Controller) konceptu. MVC koncept predstavlja arhitekturni obrazac koji je podržan od strane .NET radnog okvira, pružajući gotove predloške koji značajno olakšavaju razvoj web aplikacija.

MVC koncept omogućuje nezavisan razvoj pojedinih dijelova aplikacije, što rezultira jednostavnijim ispitivanjem i olakšanim dodavanjem novih svojstava u sustav. Takav pristup razdvaja sustav na tri ključna dijela:

- **Model:** Središnja komponenta sustava koja predstavlja dinamičke strukture podataka neovisne o korisničkom sučelju. Model izravno upravlja podacima, logikom i pravilima aplikacije. Također, prima ulazne podatke od Controllera.
- **View:** Omogućuje različite prikaze istih informacija, poput grafičkog ili tabličnog prikaza podataka.
- **Controller:** Prima ulaze od korisnika i prilagođava ih za prosljeđivanje Modelu ili Viewu. Controller upravlja korisničkim zahtjevima i temeljem njih izvodi daljnju interakciju s ostalim elementima sustava. Osim posredovanja između Modela i Viewa, Controller igra ključnu ulogu u usmjeravanju tijeka aplikacije.

Ovaj trojni raspored omogućuje jasno definiranje odgovornosti svake komponente, čime se postiže modularnost sustava. Model upravlja podacima, View se brine o prezentaciji, dok Controller koordinira interakcijom između njih. Ovaj pristup često rezultira čistim, lako održivim kodom i olakšava daljnji razvoj aplikacije.

Web aplikaciju čine tri osnovna dijela:

- **Frontend**
- **Backend**
- **Baza podataka**

Frontend se sastoji od komponenata i logike. Istu komponentu je moguće koristiti za različite namjene (engl. *reusability*). Angular koristi koncept komponenata

i servisa koji omogućuje ponovnu upotrebu koda. Angular koristi svojstva poput Dependency Injection i Change Detection za poboljšanje performansi. Struktura je ostvarena povezivanjem različitih komponenata unutar hijerarhijske strukture.

Backend je ključna komponenta web aplikacije, sastavljena od sljedećih dijelova:

- **Programsko sučelje za reprezentacijski prijenos stanja (REST API):** Controller je odgovoran za obradu zahtjeva koji dolaze od korisnika ili drugih dijelova aplikacije. On prima HTTP zahtjeve, interpretira ih, te ih proslijedi odgovarajućim dijelovima aplikacije. Nakon obrade zahtjeva, Controller šalje odgovor korisniku, koji se sastoji od statusnog koda, tijela poruke i zaglavlja. Sadržaj u tijelu poruke predstavlja informacije koje korisnik konzumira nakon što su prikazane u web pregledniku.
- **Sloj poslovne logike (Service):** Service je odvojeni sloj koji sadrži poslovnu logiku aplikacije. Ovdje se obrađuju i provjeravaju podaci, izvršavaju poslovne operacije te se komunicira s ostalim dijelovima sustava. Service često služi kao posrednik između Controllera i Repository sloja. Komunikaciju sa slojem poslovne logike Controller ostvaruje pomoću umetanja ovisnosti (dependency injection). Dependency injection je obrazac prema kojemu se u određeni objekt/funkciju umeće neki drugi objekt/funkcija na koji se prvo-bitno spomenuti objekt/funkcija oslanja.
- **Sloj za pristup bazi podataka (Repository):** Repository je odgovoran za komunikaciju s bazom podataka. Ovaj sloj omogućuje izvođenje operacija poput čitanja, pisanja, brisanja i ažuriranja podataka u bazi. Koristi se SQL upitima ili ORM (Object-Relational Mapping) tehnikama za pretvaranje podataka iz relacijske baze podataka u objekte koji se koriste unutar aplikacije. Repository omogućuje komunikaciju s bazom podataka pomoću SQL-a. Objekti iz relacijske baze podataka pretvaraju se u objekte programskog jezika Java pomoću tehnike ORM.

Ova trostruka podjela omogućuje jasnu strukturu i odvajanje odgovornosti unutar backend-a, čineći ga modularnim i lakšim za održavanje.

4.1 Baza podataka

Baza podataka za aplikaciju implementirana je kao relacijska baza podataka, gdje se podaci pohranjuju u obliku redaka ili n-torki te stupaca ili atributa koji zajedno čine tablicu. Ovaj pristup omogućuje strukturiranu organizaciju podataka, gdje svaka tablica predstavlja određenu vrstu informacija, a svaki redak u tablici predstavlja konkretan zapis ili entitet. Relacijski model olakšava učinkovito upravljanje podacima, omogućuje jednostavno pretraživanje i izvođenje kompleksnih upita nad podacima, što čini bazu podataka temeljem za pouzdano čuvanje i manipulaciju informacijama u aplikaciji.

4.1.1 Opis tablica

Glavna komponenta baze je tablica User, koja se puni osobnim podacima unesenim pri registraciji korisnika u sustav. Svakom novom korisniku dodjeljuje se primarni ključ, unikatni identifikacijski broj pomoću kojeg se korisnici u bazi podataka međusobno razlikuju. Budući da korisnik može biti primatelj usluge, davalj usluge i menadžer usluge, postoji atribut roleId koji ovisno o broju označava ulogu korisnika u sustavu. Ako je korisnik menadžer, onda je povezan na tablicu Manager preko stranog ključa userId. Tablica Manager dodatno sadrži atribut confirmedByAdmin koji opisuje je li menadžer potvrđen od strane administratora aplikacije. Kako bi korisnik opće rezervirao parkirno mjesto, postoji tablica Parking koja sadrži svoj unikatni id, id vlasnika. Svako parkiralište ima n parkirališnih mjesta koja su opisana u tablici ParkingPlace sa stranim ključem na id parkirališta u čijem su vlasništvu, te tip mesta uz cijenu i pokazatelj zauzetosti. U sustavu baze podataka također postoji tablica Vehicle koja opisuje vozilo. Vozilo ima strani ključ na id vlasnika, te vlastiti id i naziv. Kako bi sustav rezervacija funkcionirao, postoji tablica Reservation koja povezuje korisnika, parking, parkirno mjesto, vozilo te vrijeme same rezervacije. Svaki korisnik ima svoj novčanik opisan tablicom Wallet sa stranim ključem na korisnika i količinom novca kao atributom. Na samom kraju postoji tablica Transaction koja prati uplate i isplate u sustavu.

USER		
user ID	INT	jedinstveni identifikator korisnika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

USER		
username	VARCHAR	korisničko ime
password	VARCHAR	hash lozinke
name	VARCHAR	ime korisnika
surname	VARCHAR	prezime korisnika
IBAN	VARCHAR(34)	IBAN korisnika koji se koristi prilikom plaćanja
email	VARCHAR	email korisnika
is email confirmed	BOOLEAN	potvrda je li korisnik potvrdio svoj račun preko mail-a
role id	INT	broj korisnikove uloge u sustavu
id image path	VARCHAR	fotografija osobne iskaznice korisnika

MANAGER		
user ID	INT	jedinstveni identifikator korisnika koji je menadžer parkinga, strani ključ na user ID user tablice
confirmed by admin	BOOLEAN	vrijednost je li korisnik potvrđen od strane administratora da je menadžer

PARKING		
parking ID	INT	jedinstveni identifikator parkinga
owner ID	INT	ID na korisnika vlasnika parkinga, strani ključ
parking name	VARCHAR	ime parkinga
parking description	VARCHAR	opis parkinga, koliko ima mesta, lokacija, je li natkriven, je li osvijetljen...

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

PARKING		
parking photo	VARCHAR	fotografija parkinga

PARKING SPACE		
parking space ID	INT	jedinstveni identifikator parkirnog mesta
parking ID	INT	ID na parking u čijem je vlasništvu odabрано parkirno mjesto, strani ključ
type	INT	id vozila ciji tip odgovara odabranom parkirnom mjestu (npr: automobil, bicikl...), strani ključ
status	ID	status zauzetosti parkinga, vrsta podatka je integer jer je s brojem označen status parkinga
latitude	INT	brojčana vrijednost fizičke širine parkinga, u metrima
longitude	INT	brojčana vrijednost fizičke duljine parkinga, u metrima
price	INT	cijena parkirnog mesta, po satu

VEHICLE		
vehicle ID	INT	jedinstveni identifikator vozila
vehicle name	VARCHAR	ime vozila

RESERVATION		
reservation ID	INT	jedinstveni identifikator rezervacije
user ID	INT	ID na korisnika koji je napravio rezervaciju, strani ključ
reservation date	DATETIME	datum i vrijeme rezervacije
duration	TIME	vremenska duljina rezervacije, u satima
vehicle type	INT	tip vozila na čije ime glasi rezervacija, strani ključ na tablicu vehicle
parking space id	INT	jedinstveni identifikator koje je točno parkirno mjesto rezervirano, strani ključ
repeating	BOOLEAN	vrijednost koja pokazuje treba li se rezervacija periodički ponavljati (na dnevnoj, tjednoj ili mjesecnoj razini...)

TRANSACTION		
transaction ID	INT	jedinstveni identifikator transakcije
user ID	INT	ID na korisnika koji je obavio transakciju, strani ključ
type	INT	tip transakcije
amount	FLOAT	količina novca koji je prošao kroz transakciju

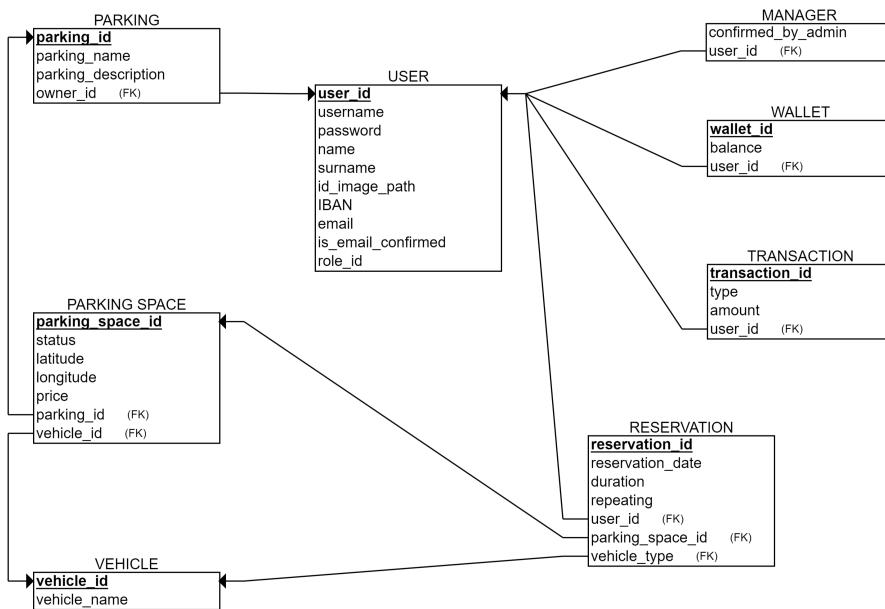
WALLET		
wallet ID	INT	jedinstveni identifikator novčanika
user ID	INT	ID na korisnika u čijem je vlasniku virtualni novčanik, strani ključ

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

WALLET		
balance	FLOAT	količina novca koji se nalazi u novčaniku

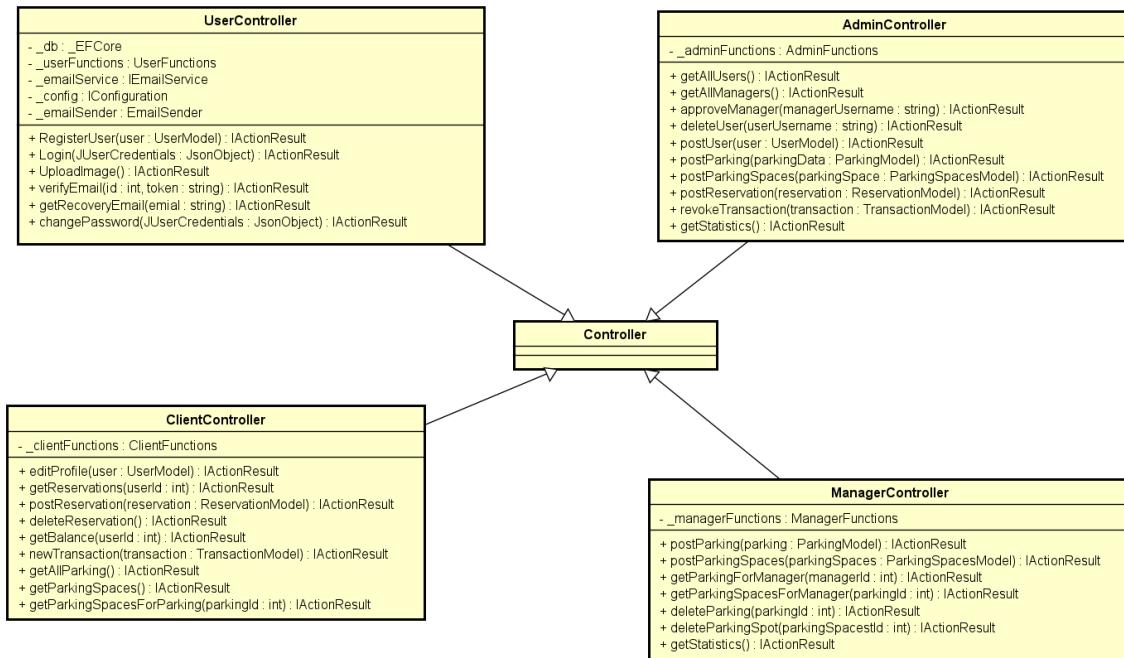
4.1.2 Dijagram baze podataka



Slika 4.2: ER dijagram baze podataka

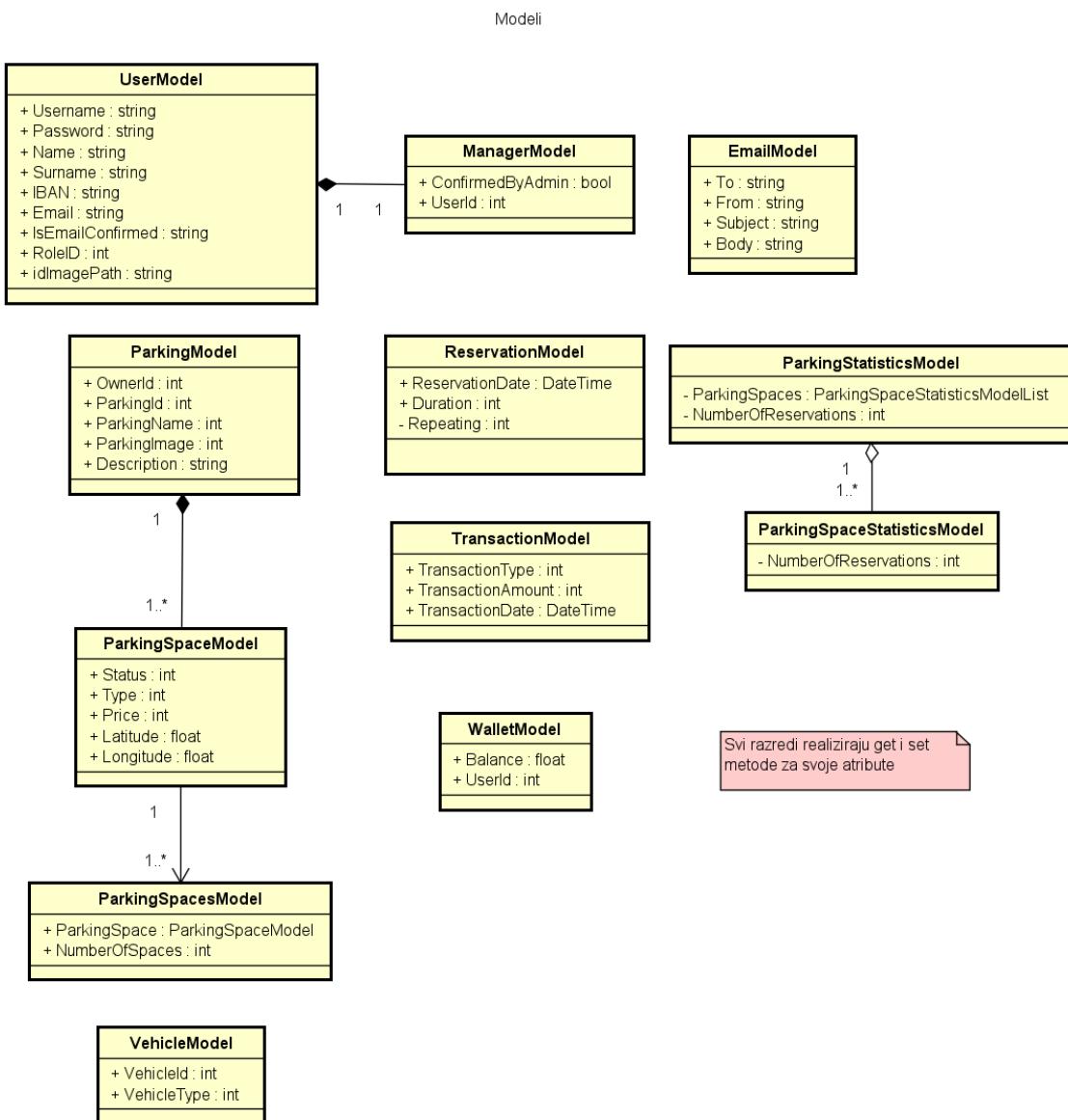
4.2 Dijagram razreda

Na slikama su prikazani razredi („Class“) koji su korišteni za implementaciju backend-a. Na slici 4.3 su prikazani razredi koji nasljeđuju Controller razred. Sve funkcije implementirane u Controller razredu vraćaju ActionResult (podatke i odgovarajući kod). Također, sve funkcije ne komuniciraju direktno s bazom podataka nego pozivaju funkcije iz određenih servisa koji imaju implementiranu tu funkcionalnost. Funkcije u kontrolerima kao parametre primaju odgovarajući model.

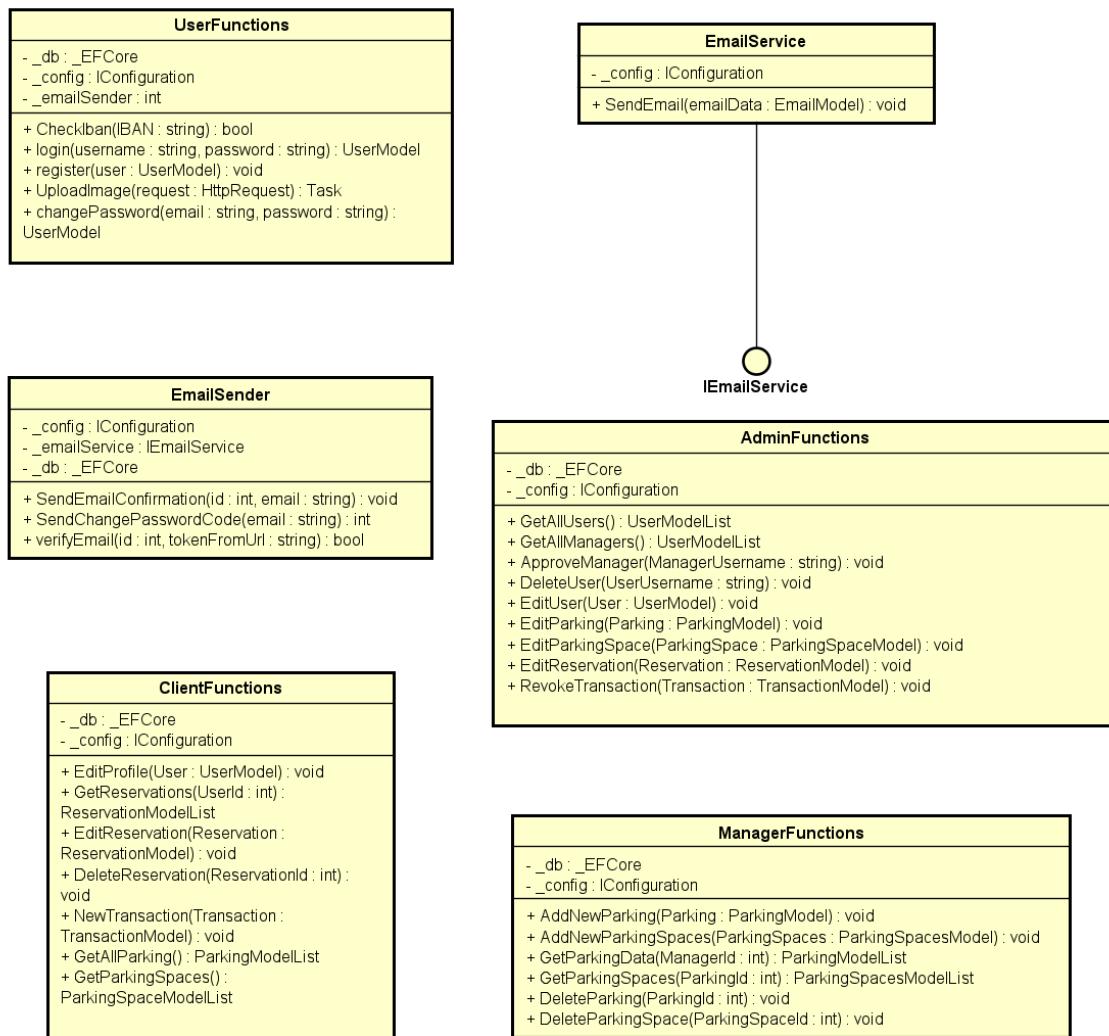


Slika 4.3: dio Controllers

Modeli razreda odražavaju strukturu baze podataka unutar aplikacije. Metode implementirane unutar tih razreda izravno komuniciraju s bazom podataka kako bi do bile tražene informacije. Razred User predstavlja generičnog korisnika aplikacije koji se može registrirati. Na taj razred referira se razred Manager (jer je svaki Manager ujedno i User).

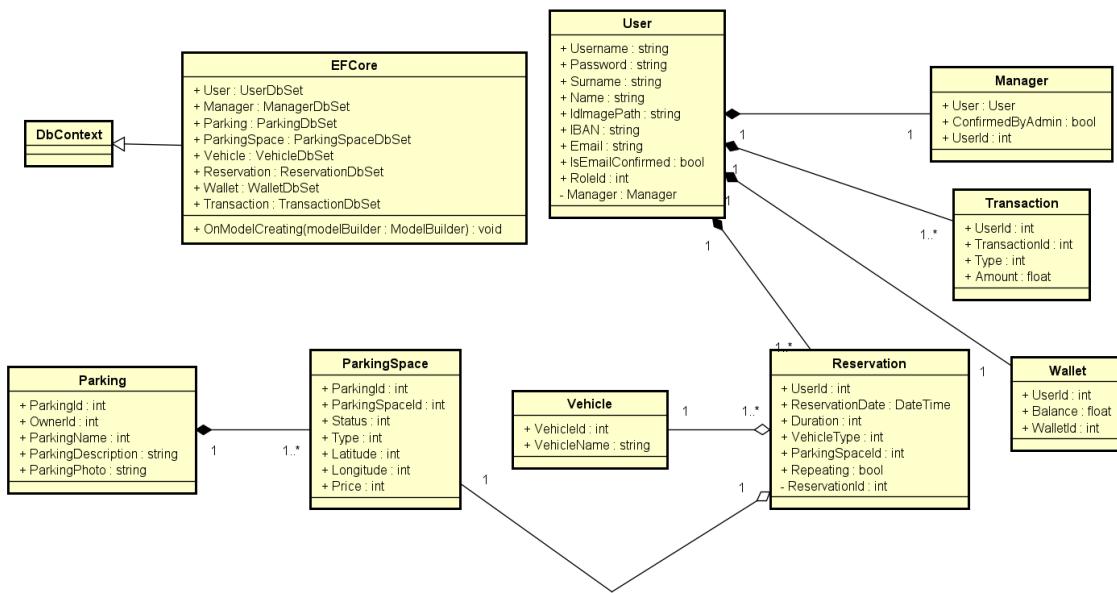


Slika 4.4: dio Models



Slika 4.5: dio Services

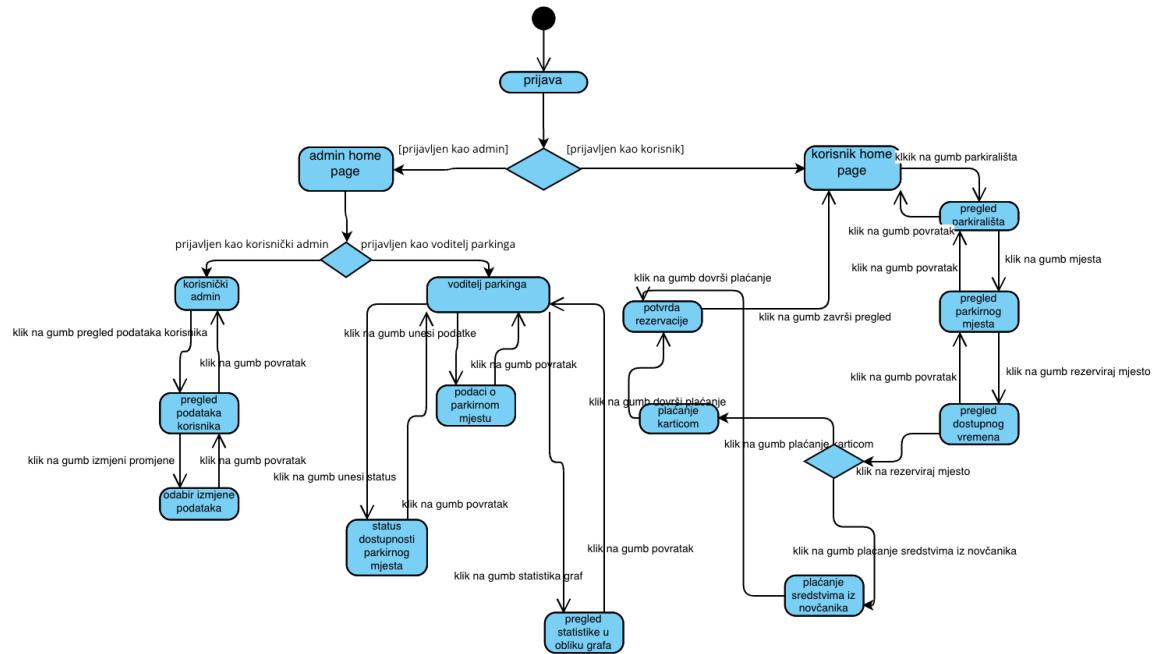
U dijagramu razreda na slici 4.5 prikazani je dio servisa. Svi servisi kao svoj atribut, između ostalih, imaju instancu objekta `EFCore` koji predstavlja kontekst za bazu podataka i instancu `IConfiguration` objekta koji služi za dohvaćanje konstanti. Funkcije definirane u pojedinom razredu dohvaćaju, mijenjaju i dodaju podatke u bazu podataka i manipuliraju podacima koje vraćaju kontroleru koji šalje nazad do korisnika.



Slika 4.6: Reprezentacija baze podataka

4.3 Dijagram stanja

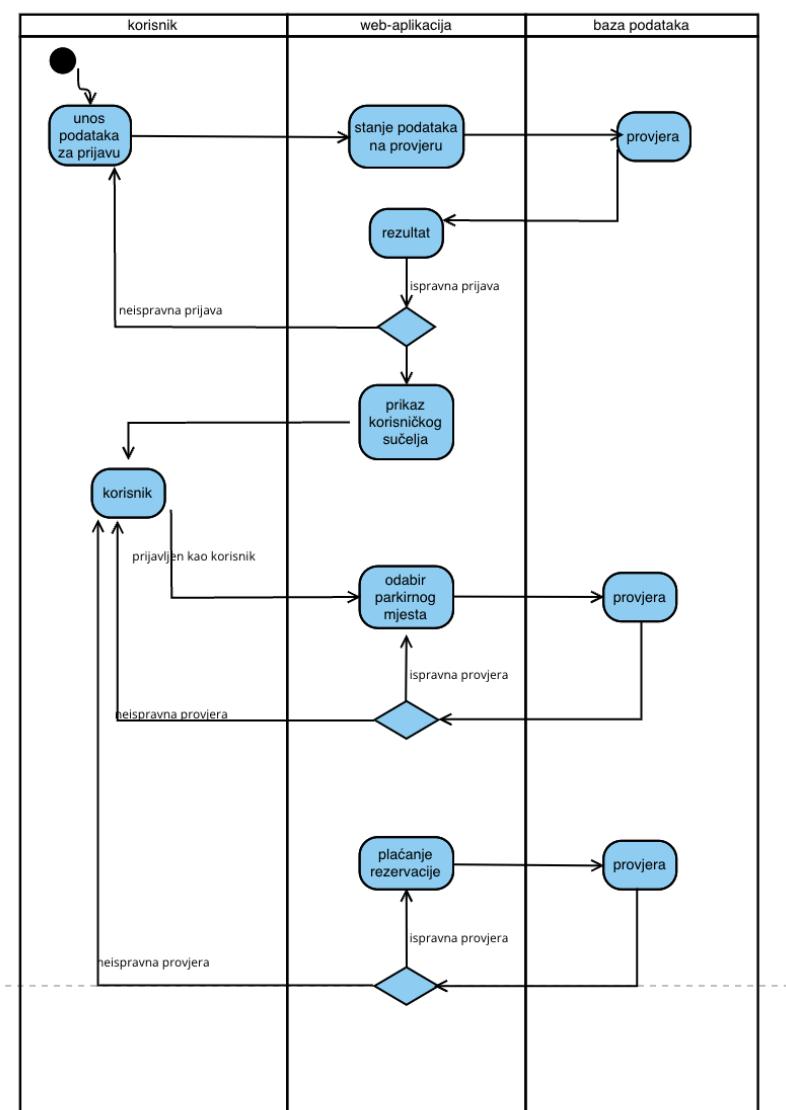
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici je prikazan dijagram stanja za registriranog korisnika bilo kao admin ili korisnik. Nakon prijave admina kao korisničkog admina prikazuje mu se početna stranica na kojoj može pregledati sve registrirane korisnike te izmjenjivati njihove podatke. Ako se admin prijavi kao vlasnik parkingu prikazuje mu se početna stranica na kojoj može unositi podatke o parkirnom mjestu, podatke o dostupnosti parkirnog mjesta te dohvatiti statistiku grafa o parkingu. Korisnik s druge strane kada se prijavi na svojoj početnoj stranici pregledava parkirališta, odabire parkirno mjesto te odabire plaćanje dolaskom na parking ili plaćanje karticom te sredstvima iz novčanika.



Slika 4.7: Dijagram stanja

4.4 Dijagram aktivnosti

Dijagram aktivnosti koristi se za opis modela upravljačkog ili podatkovnog toka. Nije namijenjen modeliranju događaja potaknutog ponašanja. U modeliranju upravljačkog toka, svaki sljedeći korak slijedi nakon završetka prethodnog, pri čemu se naglašava jednostavnost. Na slici je prikazan dijagram aktivnosti koji opisuje proces rezerviranja parkirnog mjesta. Korisnik nakon uspješne prijave odabire parking, parkirališno mjesto, datum i vrijeme. Zatim mu se prikazuju opcije plaćanja te rezervacije gdje nakon uspješne transakcije korisniku se potvrđuje rezervacija.

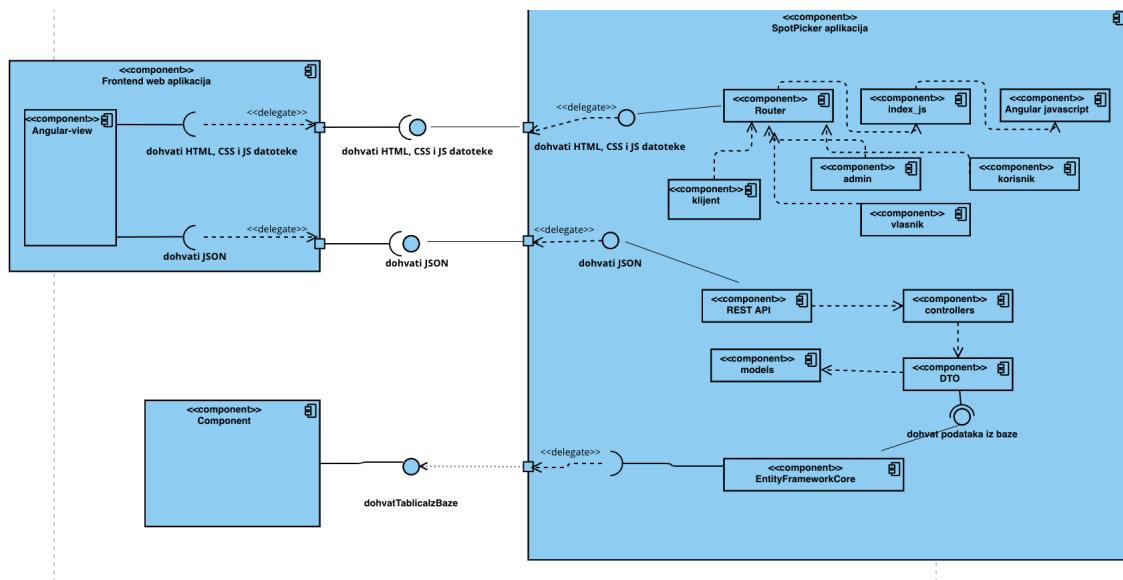


Slika 4.8: Dijagram aktivnosti

4.5 Dijagram komponenti

Dijagram komponenti, prikazan na slici, ilustrira organizaciju i međusobnu povezanost komponenti, unutarnju strukturu te odnose s okolinom. Sustav je dostupan putem dva različita sučelja. Sučeljem za dohvat HTML, CSS i JS datoteka pristupaju datoteke koje pripadaju frontend dijelu aplikacije. Router je komponenta koja, na temelju upita s URL-a, određuje koja datoteka će biti poslužena na sučelju. Frontend se sastoji od niza JavaScript datoteka grupiranih u logičke cjeline nazvane prema tipovima korisnika koji im pristupaju. Sve JavaScript datoteke ovise o Angular biblioteci, iz koje se pozivaju gotove komponente poput gumba, obrazaca i slično.

Sučeljem za dohvat JSON podataka pristupa se REST API komponenti, koja poslužuje podatke pripadajuće backend dijelu aplikacije. EntityFrameworkCore je odgovoran za dohvaćanje tablica iz baze podataka putem SQL upita. Podaci koji stižu iz baze šalju se dalje MVC arhitekturi u obliku DTO (Data Transfer Object).



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Za međusobnu komunikaciju unutar tima korišten je Discord u kojem su članovi tima bili podijeljeni u nekoliko kanala kako bi se organizirali različiti dijelovi projekta. Također za svojevrsnu komunikaciju upotrebljavan je i GitHub što je jedna od najpoznatijih platformi koja omogućuje programerima da imaju repozitorije u kojima zatim mogu stvarati, pohranjivati i upravljati svojim kodovima i projektima. Također moguće je pratiti svaku promjenu. Prilikom izrade projekta GitHub je služio kako bi bila olakšana međusobna komunikacija te kako bi na jednom mjestu bilo dostupno sve što se radi na projektu u svakom trenutku.

Za izradu dokumentacije korišten je sustav za uređivanje teksta LaTex. Ovaj markup jezik se najčešće koristi za izradu znanstvene i tehničke publikacije. Njegove prednosti su lakoća formatiranja velikih datoteka što olakšava i skraćuje vrijeme koje bi se potrošilo na uređivanje samog dokumenta. Prilikom izrade UML dijagrama korišten je Astah UML alat za modeliranje UML dijagrama. Poznat je po tome što je vrlo jednostavan za učenje i korištenje. Također brži je od Excela i drugih alata za crtanje. Astah je primijenjen prilikom izrade dijagrama obrazaca uporabe, sekvensijskih dijagrama, dijagrama baze podataka, dijagrama razreda, stanja, aktivnosti, komponenti te dijagram razmještaja. Za izradu frontenda korišten je Angular te jezik Typescript dok su za backend korišteni .NET Framework i jezik C.

Angular je platforma tj. okvir (eng. Framework) koji služi za razvijanje web aplikacija pomoću TypeScript-a i HTML-a. Kako bi se mogao koristiti Angular potrebno je prethodno instalirati NPM te Node.js. Prednost Angular-a je kod koji je lako čitljiv te koji se može ponovno iskoristiti, također dobra je tehnologija za rad u timu s obzirom da omogućuje usporedan samostalan rad. Radni okvir .NET Framework namijenjen je za izgradnju i izvođenje aplikacija. Korišteni su i Visual Studio Code i Visual Studio. To su integrirana razvojna okruženja (eng. IDE). Koriste se kao platforma za uređivanje izvornog koda te platforma za pokretanje koja se može koristiti za uređivanje, debugiranje, izgradnju koda i samo objavljivanje

aplikacije.

PGAdmin je korisničko grafičko sučelje iskorišteno za interakciju sa Postgresom prilikom izrade baze podataka dok je EntityFramework korišten kao poveznica baze podataka i backend-a. Za dohvat početne informacije o parkirališnim mjestima upotrebljeno je aplikacijsko programsko sučelje Overpass, a za mapu je iskorišten Open Street Map. Također korišten je Nominatim koji služi za pretvaranje koordinata u adresu te OSRM Demo API koji koristi algoritme usmjeravanja za izračun i pronađak najkraćeg puta do odredišta.

1. <https://discord.com/>
2. <https://github.com/>
3. <https://www.latex-project.org/>
4. <https://astah.net/products/astah-uml/>
5. <https://angular.io/>
6. <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework>
7. <https://visualstudio.microsoft.com/>
8. <https://www.pgadmin.org/>
9. <https://learn.microsoft.com/en-us/aspnet/entity-framework>
10. <https://hrbrmstr.github.io/overpass/>
11. <https://www.openstreetmap.org/#map=7/44.523/16.460>
12. <https://nominatim.openstreetmap.org/ui/search.html>
13. <https://project-osrm.org/docs/v5.5.1/api/#introduction>

5.2 Ispitivanje programskog rješenja

Opisujemo provedbu ispitivanja implementiranih funkcionalnosti na razini komponenti i na razini cijelog sustava s prikazom odabralih ispitnih slučajeva.

5.2.1 Ispitivanje komponenti

DistanceFunctionTest

Pri stvaranju instantne rezervacije traži se parkirno mjesto koje je najbliže traženoj destinaciji. Pri tome odlučili smo koristiti Haversinovu formulu. Za ovaj test odbrali smo pouzdan primjer za koji znamo točan odgovor te testirali vraća li funkcija ispravnu vrijednost:

```
$ namespace SpotPickerTests
{
    public class DistanceFunctionTest
    {
        [Fact]
        public void TestDistanceFunction()
        {
            var (c1, c2) = (41.507483, -99.436554);
            var (c3, c4) = (38.504048, -98.315949);
            var result = 347.3;

            Assert.True(Math.Abs(result - SpotPicker.Services.ParkingFunctions
                .HaversineDistance(c1, c2, c3, c4)) <= 0.05);
        }
    }
}
```

Slika 5.1: DistanceFunctionTest

IBanFunctionTest

Pri mijenjanju IBAN-a od strane admina ili registraciji provjerava se validnost unesenog IBAN-a. Zato testiramo funkciju koja to provjerava pomoću ispravnog i neispravnog IBAN-a:

PasswordRegexTest

Pri registraciji unosi se lozinka, minimalna duljina je 8 znakova, uključujući minimalno jednu znamenku, zato testiramo tu funkciju s nekoliko primjera:

```

namespace SpotPickerTests
{
    public class IBanFunctionTest
    {
        [Fact]
        public void TestIBANFunctionValid()
        {
            string validIBAN = "HR7723400092999136896";
            bool result = SpotPicker.Services.UserFunctions.CheckIban(validIBAN);
            Assert.True(result);
        }

        [Fact]
        public void TestIBANFunctionInvalid()
        {
            var invalidIBAN = "progiprojekt";
            var result = SpotPicker.Services.UserFunctions.CheckIban(invalidIBAN
                );
            Assert.False(result);
        }
    }
}

```

Slika 5.2: IBanFunctionTest

```

namespace SpotPickerTests
{
    public class PasswordRegexTest
    {
        [Fact]
        public void TestPasswordRegexFunctionValid()
        {
            // osam charova i barem jedna znamenka
            string validPassword = "strongpassword123";
            bool result = SpotPicker.Services.UserFunctions.checkPasswordRegex(validPassword);
            Assert.True(result);
        }

        [Fact]
        public void TestPasswordRegexFunctionInvalid()
        {
            string invalidPassword1 = "weakpw"; // samo 6 charova
            string invalidPassword2 = "weakpw1"; // barem jedna znamenka, ali 7 charova
            string invalidPassword3 = "weakpwwwww"; // 8 ili vise charova, ali 0 znamenki

            bool result = SpotPicker.Services.UserFunctions.checkPasswordRegex(invalidPassword1) ||
                SpotPicker.Services.UserFunctions.checkPasswordRegex(invalidPassword2) ||
                SpotPicker.Services.UserFunctions.checkPasswordRegex(invalidPassword3);

            Assert.False(result);
        }
    }
}

```

Slika 5.3: PasswordRegexTest

5.2.2 Ispitivanje sustava

LoginTest

Ovo je sistem test koji testira funkcija li login (ulogira se u adminov profil):

AdminChangeTest

Još jedan sistem test, ovaj provjerava funkcija li promjena nečije uloge od strane admina:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace SpotPickerTests
{
    public class LoginTest
    {
        [Fact]
        public void TestLogin()
        {
            ChromeDriver driver = new ChromeDriver();
            string URL = "https://spotpicker.online/login";
            try {
                driver.Navigate().GoToUrl(URL);
                // Console.WriteLine(driver.Title);
                IWebElement username = driver.FindElement(By.XPath("//html/body/app-root/app-login/body/div/div/form/div[1]/input"));
                IWebElement password = driver.FindElement(By.XPath("//html/body/app-root/app-login/body/div/div/form/div[2]/input"));
                username.SendKeys("admin");
                password.SendKeys("test123123");

                IWebElement loginButton = driver.FindElement(By.XPath("//html/body/app-root/app-login/body/div/div/form/button"));
                loginButton.Click();

                WebDriverWait wait = new WebDriverWait(driver, new System.TimeSpan(0, 0, 5));
                wait.Until(wt => wt.FindElement(By.XPath("//html/body/app-root/app-dashboard/div/app-admin/div[1]/div/div[1]")));
                // Console.WriteLine(driver.Title);
                Assert.Equal("https://spotpicker.online/dashboard", driver.Url);
            } finally {
                driver.Dispose();
            }
        }
    }
}
```

Slika 5.4: LoginTest

Za pokretanje testova koristili smo xUnit testing framework, ovo je rezultat:

```

using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace SpotPickerTests
{
    public class AdminChangeTest
    {
        [Fact]
        public void TestChangingPartRole()
        {
            ChromeDriver driver = new ChromeDriver();
            string URL = "https://spotpicker.online/login";
            try
            {
                driver.Navigate().GoToUrl(URL);
                //Console.WriteLine(driver.Title);
                IWebElement username = driver.FindElement(By.XPath("//html/body/app-root/app-login/body/div/form/div[1]/input"));
                IWebElement password = driver.FindElement(By.XPath("//html/body/app-root/app-login/body/div/form/div[2]/input"));
                username.SendKeys("admin");
                password.SendKeys("test123123");

                IWebElement loginButton = driver.FindElement(By.XPath("//html/body/app-root/app-login/body/div/div/form/button"));
                loginButton.Click();

                WebDriverWait wait = new WebDriverWait(driver, new System TimeSpan(0, 0, 5));
                wait.Until(ExpectedConditions.ElementExists(By.XPath("//html/body/app-root/app-dashboard/div/app-admin/div[2]/div[4]/div[6]/div[1]/span[1]")));
                //Console.WriteLine(driver.Title);

                IWebElement child = driver.FindElement(By.XPath("//span[contains(text(), 'tempmail@test.com')]"));
                IWebElement parent = child.FindElement(By.XPath("./.."));
                var children = parent.FindElements(By.XPath("./"));
                var role = int.Parse(children[2].Text);
                children[4].Click();

                Thread.Sleep(1000);

                parent = driver.FindElement(By.XPath("//html/body/app-root/app-dashboard/div/app-admin/div[2]/div[4]/div[1]/div[2]/div[2]/div[1]"));
                children = parent.FindElements(By.XPath("./"));
                var div = children[3];

                children = div.FindElements(By.XPath("./"));
                Thread.Sleep(1000);
                SelectElement select = new SelectElement(div.FindElements(By.XPath("./"))[1]);

                int desiredRole = role == 1 ? 1 : 0;
                select.SelectByIndex(desiredRole);
                Thread.Sleep(1000);
                driver.Navigate().Refresh();

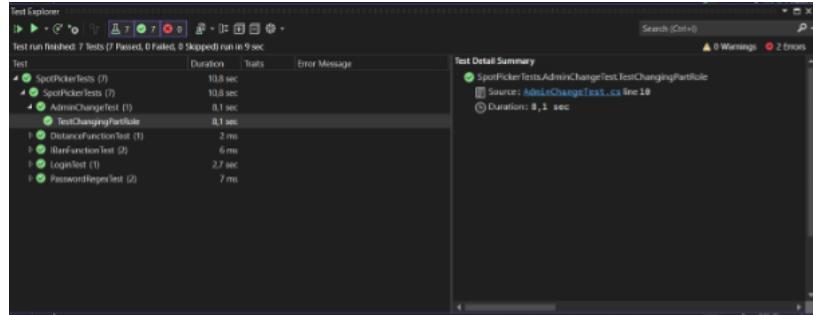
                Thread.Sleep(1000);

                child = driver.FindElement(By.XPath("//span[contains(text(), 'tempmail@test.com')]"));
                parent = child.FindElement(By.XPath("./.."));
                children = parent.FindElements(By.XPath("./"));
                var newRole = int.Parse(children[2].Text);

                Assert.NotEqual(newRole, role);
            }
            finally
            {
                driver.Dispose();
            }
        }
    }
}

```

Slika 5.5: AdminChangeTest

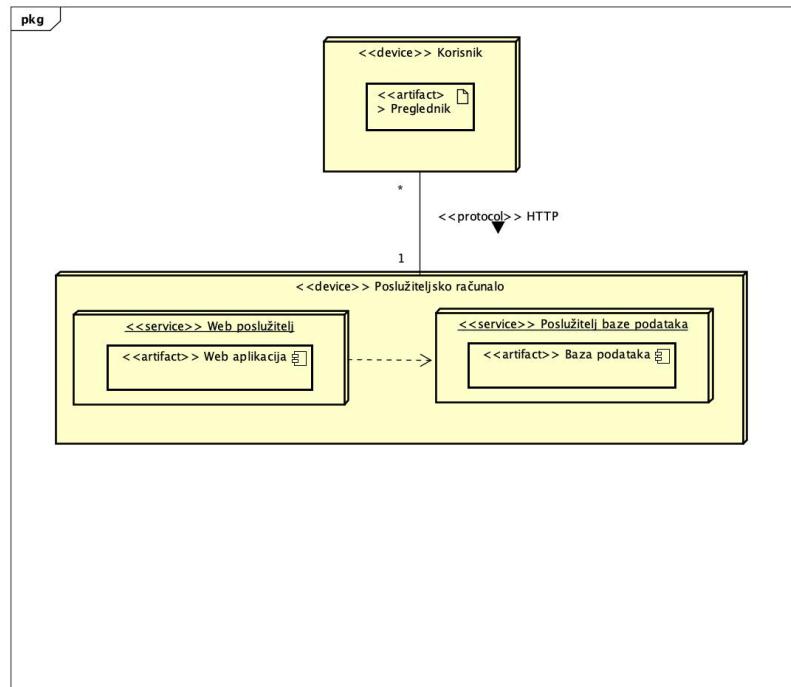


Slika 5.6: xUnit testing framework

5.3 Dijagram razmještaja

Kako bi se bolje razumjela arhitektura sustava koristi se dijagram razmještaja. On prikazuje arhitekturu programskog sustava tako što pokazuje razmještaj programske potpore i samu topologiju sklopolja. Svrha dijagrama razmještaja je pružanje pomoći prilikom donošenja raznih odluka u vezi sustava. Poslužiteljsko računalo može obavljati veći broj radnji u isto vrijeme te ga u istom trenutku može koristiti više korisnika. Web poslužitelj te poslužitelj baze podataka nalaze se na poslužiteljskom računalu, a korisnici koriste web poslužitelj za pristup web aplikacijama.

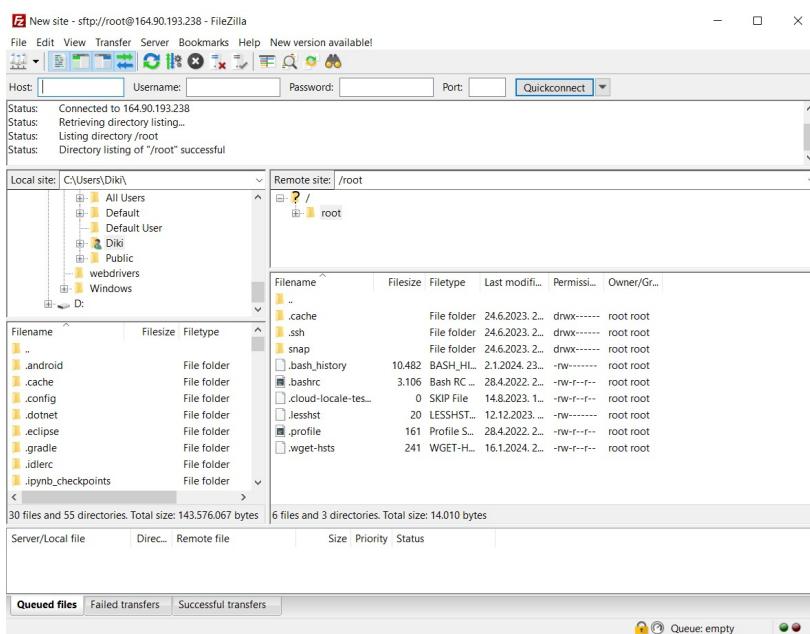
kaciji. Korištena je klijent-poslužitelj arhitektura. Prednosti ove arhitekture je što više klijenata istovremeno može pristupiti poslužitelju te mogu pristupiti funkcionalnostima poslužitelja sa udaljenosti. Komunikacija između računala korisnika, administratora i voditelja parkinga je realizirana putem HTTP veze.



Slika 5.7: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Postoji server odnosno računalo koje je smješteno u Nizozemskoj i na njemu je instaliran Nginx server koji poslužuje našu aplikaciju. Na tom remote računalu se nalazi frontend, backend i baza podataka. Baza podataka je PostgreSQL i ona je otvorena za konekcije svim IP adresama, potrebna je samo lozinka te se mi spojimo sa svojeg računala na tu bazu koja je u Nizozemskoj i onda ažuriramo nove podatke koje postavljamo. Na taj način bazu podataka ažuriramo i postavljamo, a frontend i backend prvo izgradimo i onda te datoteke pomoću FileZilla, software koji nam olakšava povezivanje sa serverom, pomoću SFTP protokola prebacujemo podatke na remote računalo odnosno naš server.



Slika 5.8: prikaz povezivanja u datotečni sustav remote računala

```

user      www-data; ## Default: nobody
worker_processes 5; ## Default: 1
worker_rlimit_nofile 8192;
events {
}

http {
    include /etc/nginx/mime.types;
    include /etc/nginx/proxy.conf;
    limit_req_zone $binary_remote_addr zone=one:10m rate=5r/s;
    server_tokens off;

    sendfile on;
    # adjust keepalive_timeout to the lowest possible value that makes sense
    # for your application
    keepalive_timeout 20;
    client_body_timeout 10; client_header_timeout 10; send_timeout 10;
    error_log /var/log/nginx/error.log debug;

    server {
        listen          443 ssl http2;
        listen [::]:443 ssl http2;
        server_name    mojastvar.hr;
        ssl_certificate /etc/letsencrypt/live/mojastvar.hr-0001/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/mojastvar.hr-0001/privkey.pem; # managed by Certbot
        ssl_protocols   TLSv1.2 TLSv1.3;
        ssl_ciphers     ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
        ssl_session_cache shared:SSL:10m;
        ssl_session_tickets off;
        ssl_stapling off;

        add_header X-Frame-Options DENY;
        add_header X-Content-Type-Options nosniff;

        location /api {
            proxy_pass http://localhost:5000; # Assuming your .NET backend is running on port 5000
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "keep-alive";
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
            limit_req zone=one burst=50 nodelay;
        }

        location /assetz {
            alias /var/www/webapi/assetsz;
        }

        location / {
            index index.html;
            try_files $uri /index.html;
            limit_req zone=one burst=50 nodelay;
        }
    }

    server {
        listen 80;
        server_name _;
        return 301 https://$host$request_uri;
    }
}

```

Slika 5.9: prikaz konfiguracije servera

```

File Edit Format View Help
proxy_redirect          off;
proxy_set_header        Host $host;
proxy_set_header        X-Real-IP $remote_addr;
proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header        X-Forwarded-Proto $scheme;
client_max_body_size   10m;
client_body_buffer_size 128k;
proxy_connect_timeout  90;
proxy_send_timeout     90;
proxy_read_timeout     90;
proxy_buffers          32 4k;

```

Slika 5.10: prikaz konfiguracije proxyja

```

root@ubuntu-s-1vcpu-512mb-10gb-ams3-01:~ Swap usage:  0%          IPv4 address for eth0: 10.18.0.6
99 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

*** System restart required ***
Last login: Fri Nov 17 17:32:31 2023 from 89.172.131.198
root@ubuntu-s-1vcpu-512mb-10gb-ams3-01:~# systemctl status webapi
● webapi.service - SpotPicker API
   Loaded: loaded (/etc/systemd/system/webapi.service; enabled; preset: enabled)
   Active: active (running) since Fri 2023-12-15 06:46:51 UTC; 1 month 2 days ago
     Main PID: 922499 (dotnet)
       Tasks: 15 (limit: 497)
      Memory: 75.9M
         CPU: 8min 13.439s
        Group: /system.slice/webapi.service
               └─ 922499 /usr/bin/dotnet /var/www/webapi/SpotPicker.dll

Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: Executed DbCommand (2ms) [Parameters=@_existingUser_Id_0]
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: SELECT m."UserId", m."ConfirmedByAdmin"
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: FROM "Manager" AS m
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: WHERE m."UserId" = @_existingUser_Id_0
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: LIMIT 1
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: info: Microsoft.EntityFrameworkCore.Database.C
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: Executed DbCommand (2ms) [Parameters=@_existingUser_Id_0]
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: SELECT m."UserId", m."ConfirmedByAdmin"
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: FROM "Manager" AS m
Jan 16 13:45:19 ubuntu-s-1vcpu-512mb-10gb-ams3-01 mojastvar-api[922499]: WHERE m."ConfirmedByAdmin" IS NULL
lines 1-20/20 (END).

```

Slika 5.11: prikaz povezivanja preko konzole na Ubuntu server

6. Zaključak i budući rad

Zadatak naše grupe bio je razvoj web aplikacije pod nazivom "SpotPicker". Ideja aplikacije je da se korisnicima omoguć rezervacija, naplata parkiranja i pregled slobodnih parkirališnih mesta za automobile i bicikle. Nakon 12 tjedana timskog rada, ostvarili smo zadani cilj i projekt je završen. Projekt je bio podijeljen u dvije ključne faze, pri čemu je prva faza obuhvatila okupljanje tima, dodjelu projektnog zadatka te temeljito dokumentiranje zahtjeva. Ova faza pokazala se ključnom za uspješnu provedbu projekta, pružajući čvrst temelj za daljnji razvoj sustava.

Druga faza, iako kraća, zahtjevala je intenzivan rad i samostalnost članova tima. Nedostatak iskustva s određenim alatima i jezicima potaknuo nas je na samostalno učenje kako bismo uspješno ispunili postavljene ciljeve. Dokumentiranje UML dijagrama i izrada popratne dokumentacije bili su ključni za olakšavanje budućeg održavanja i nadogradnje sustava.

Sudjelovanje u ovom projektu bilo je izuzetno vrijedno iskustvo za sve članove tima. Za većinu nas, ovo je bio prvi ozbiljniji grupni projekt, i moram reći da smo se iznimno dobro snašli. Biti deo tima koji nije imao sukoba, gdje je suradnja bila prirodna, a komunikacija na iznimno zadovoljavajućoj razini, pridonijelo je uspješnosti realizacije projekta.

Ovaj projekt predstavljao je prvi ozbiljniji susret s tehnologijama kao što su Git i LaTeX za većinu nas. Tijekom procesa razvoja, stekli smo znanje o korištenju modernih radnih okvira u izradi web aplikacija. Iznimno smo zadovoljni postignutim rezultatima i cijelim timskim radom koji je doprinio ostvarenju tih rezultata.

Ovo iskustvo nas je obogatilo ne samo tehničkim znanjem već i sposobnostima timskog rada i upravljanja projektima. Spremni smo za nove izazove i s veseljem nastavljamo s dalnjim profesionalnim razvojem.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
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. Visual Paradigm, <https://online.visual-paradigm.com/>
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>

Indeks slika i dijagrama

2.1	Sučelje za odabir termina u JustPark	8
2.2	Dodatna mogućnost ocjene parkinga u JustPark	8
3.1	Sekvencijski dijagram za UC4	19
3.2	Sekvencijski dijagram za UC5	20
3.3	Sekvencijski dijagram za UC6	21
4.1	Arhitektura sustava	22
4.2	ER dijagram baze podataka	29
4.3	dio Controllers	30
4.4	dio Models	31
4.5	dio Services	32
4.6	Reprezentacija baze podataka	34
4.7	Dijagram stanja	35
4.8	Dijagram aktivnosti	36
4.9	Dijagram komponenti	37
5.1	DistanceFunctionTest	40
5.2	IBanFunctionTest	41
5.3	PasswordRegexTest	41
5.4	LoginTest	42
5.5	AdminChangeTest	43
5.6	xUnit testing framework	43
5.7	Dijagram razmještaja	44
5.8	prikaz povezivanja u datotečni sustav remote računala	45
5.9	prikaz konfiguracije servera	46
5.10	prikaz konfiguracije proxyja	46
5.11	prikaz povezivanja preko konzole na Ubuntu server	46
6.1	prikaz aktivnosti na repozitoriju	53

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 15. listopad 2023.
- Prisustvovali: Luka Diktić, Luka Babić, Luka Žmak, Maja Mavračić, Mirta Posnjak, Dominik Poljak, Filip Sučić
- Teme sastanka:
 - podjela poslova

2. sastanak

- Datum: 4. studenog 2023.
- Prisustvovali: Luka Diktić, Luka Babić, Luka Žmak, Maja Mavračić, Mirta Posnjak, Dominik Poljak, Filip Sučić
- Teme sastanka:
 - odabir alata i tehnologija
 - definiranje funkcionalnih zahtjeva

3. sastanak

- Datum: 20. prosinca 2023.
- Prisustvovali: Luka Diktić, Luka Babić, Luka Žmak, Maja Mavračić, Mirta Posnjak, Dominik Poljak, Filip Sučić
- Teme sastanka:
 - podjela rada implementacije
 - podjela rada dokumentacije

4. sastanak

- Datum: 10. siječnja 2024.
- Prisustvovali: Luka Diktić, Luka Babić, Luka Žmak, Maja Mavračić, Mirta Posnjak, Dominik Poljak, Filip Sučić
- Teme sastanka:
 - završne podjele zadataka
 - podjela rada dokumentacije

Tablica aktivnosti

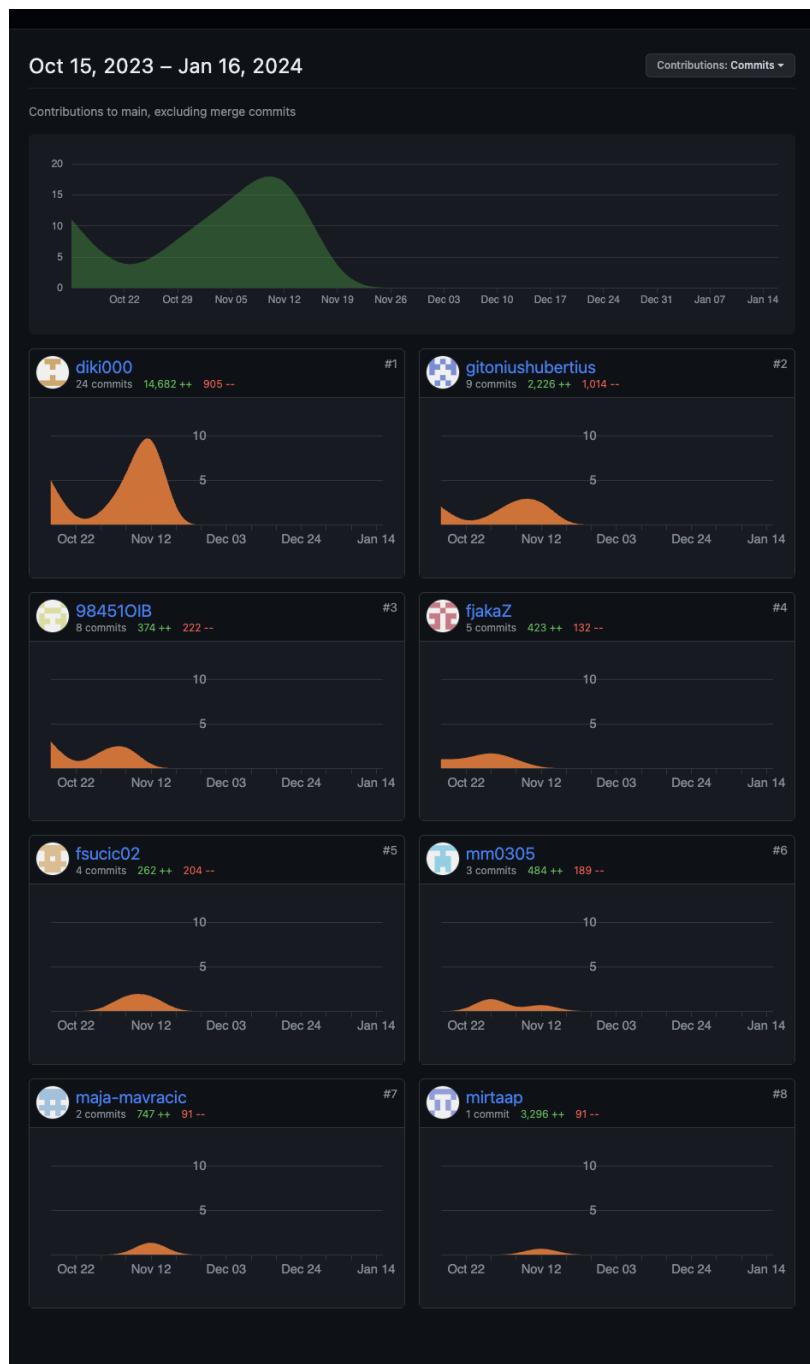
	Luka Diktič	Maja Mavračić	Mirta Posnjak	Filip Sučić	Dominik Poljak	Luka Babić	Luka Žmaka
Upravljanje projektom	30				7	7	
Opis projektnog zadatka							5
Funkcionalni zahtjevi	4	4					
Opis pojedinih obrazaca	5	5					5
Dijagram obrazaca	6						
Sekvencijski dijagrami			6				
Opis ostalih zahtjeva							3
Arhitektura i dizajn sustava	7						
Baza podataka				2	5	5	
Dijagram razreda				4			
Dijagram stanja			4				
Dijagram aktivnosti		4					
Dijagram komponenti						5	
Korištene tehnologije i alati	3						
Ispitivanje programskog rješenja	3						
Dijagram razmještaja		3					
Upute za puštanje u pogon							4
Dnevnik sastajanja	3						
Zaključak i budući rad						4	
Popis literature			2				

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Luka Diktić	Maja Mavračić	Mirta Posnjak	Filip Sučić	Dominik Poljak	Luka Babić	Luka Žmaka
<i>Dodatne stavke kako ste podijelili izradu aplikacije</i>	3						
<i>npr. izrada početne stranice</i>	3	2	2				4
<i>izrada baze podataka</i>					4	4	
<i>spajanje s bazom podataka</i>	5						
<i>back end</i>	10			6	6	6	

Dijagrami pregleda promjena



Slika 6.1: prikaz aktivnosti na repozitoriju