

Programsko inženjerstvo

Ak. god. 2023./2024.

Iznajmi Romobil

Dokumentacija, Rev. 1

Grupa: *ScootShare*

Voditelj: *Ivan Pavelić*

Datum predaje: 17. 11. 2023.

Nastavnik: *Igor Stančin*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	5
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	21
3.2 Ostali zahtjevi	24
4 Arhitektura i dizajn sustava	25
4.1 Baza podataka	26
4.1.1 Opis tablica	26
4.1.2 Dijagram baze podataka	31
4.2 Dijagram razreda	32
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	42
5.3 Dijagram razmještaja	45
5.4 Upute za puštanje u pogon	46
6 Zaključak i budući rad	49
Popis literature	51
Indeks slika i dijagonama	53

Dodatak: Prikaz aktivnosti grupe

54

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatak	Autori	Datum
0.1	Napravljen predložak.	Karmen Korić, Dino Babić, Jan Grbac, Anamarija Jakoubek, Ivan Pavelić, Igor Šoštarko, Leonarda Pribanić	20.10.2023.
0.2	Napisan opis projekta	Karmen Korić, Dino Babić, Jan Grbac, Anamarija Jakoubek, Ivan Pavelić, Igor Šoštarko, Leonarda Pribanić	25.10.2023.
0.3	Dodani funkcionalni zahtjevi i ostali zahtjevi	Karmen Korić, Dino Babić, Jan Grbac, Anamarija Jakoubek, Ivan Pavelić, Igor Šoštarko, Leonarda Pribanić	25.10.2023.
0.4	Dodani dijagrami i opisi obrazaca uporabe.	Karmen Korić, Dino Babić, Anamarija Jakoubek	1.11.2023.
0.5	Dodani sekvencijski dijagrami i njihovi opisi.	Ivan Pavelić, Jan Grbac, Igor Šoštarko, Leonarda Pribanić	2.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.6	Arhitektura i dizajn sustava.	Karmen Korić, Dino Babić, Jan Grbac, Anamarija Jakoubek, Ivan Pavelić, Igor Šoštarko, Leonarda Pribanić	4.11.2023.
0.7	Baza podataka.	Karmen Korić, Dino Babić, Jan Grbac, Anamarija Jakoubek, Ivan Pavelić, Igor Šoštarko, Leonarda Pribanić	4.11.2023.
0.8	Prvi dio dijagrama razreda.	Karmen Korić, Dino Babić, Jan Grbac, Anamarija Jakoubek, Ivan Pavelić, Igor Šoštarko, Leonarda Pribanić	10.11.2023.
0.8	Dodani ostali dijagrami razreda i doradena baza podataka.	Karmen Korić, Dino Babić, Jan Grbac, Anamarija Jakoubek, Ivan Pavelić, Igor Šoštarko, Leonarda Pribanić	13.11.2023.
1.0	Provjera i analiza dokumentacije	Karmen Korić, Dino Babić, Jan Grbac, Anamarija Jakoubek, Ivan Pavelić, Igor Šoštarko, Leonarda Pribanić	13.11.2023.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije "Iznajmi Romobil" koja će korisniku omogućiti da iznajmi vlastiti romobil u vrijeme kada ga ne koristi, a korisnicima koji ne posjeduju romobil dati mogućnost najma romobila.

Prilikom pokretanja aplikacije prikazuje se glavna stranica s oglasima za najam romobila. Neprijavljeni korisnik oglase može samo pregledavati, dok prijavljeni korisnik može i stupiti u kontakt s iznajmljivačem te u konačnici i unajmiti oda-brani romobil.

Ako korisnik nema izrađen račun u aplikaciji, ima mogućnost registracije. Pri-likom registracije, korisnik unosi:

- ime
- prezime
- nadimak
- email adresa
- broj kartice
- lozinka
- kopija osobne iskaznice
- potvrda o nekažnjavanju

Tijekom obrade registracije na strani poslužitelja korisniku se dodjeljuju ovlasti. S obzirom na to da korisnik može biti i iznajmljivač i klijent, jedna će ovlast objedi-niti te uloge. Također, sustav razlikuje i ovlast administrator.

Jednom kada se korisnik registrira te mu administrator prihvati zahtjev za re-gistraciju, ili već od prije ima aktivan račun, ima mogućnost prijave u aplikaciju. Prilikom prijave korisnik unosi korisničko ime i lozinku. Jednom kada potvrdi prijavu, na poslužiteljsku stranu šalju se uneseni podaci, a povratno poslužitelj odgovara ili s greškom ili s JSON web tokenom u kojem su spremljeni ime i pre-zime korisnika te njegove ovlasti.

Ako je prijava uspješno provedena korisnika se prosljeđuje na početnu stra-nicu. Kao što je prethodno navedeno, prijavljeni korisnik sada može iznajmljivati i unajmljivati romobile. Prijavljeni korisnik može pregledavati i mijenjati svoje

osobne podatke. Dodatno mu je omogućeno i postavljanje profilne slike.

Da bi iznajmljivač postavio svoj romobil na iznajmljivanje mora ga registrirati. Prilikom registracije romobila iznajmljivač unosi slike romobila kao dokaz trenutnog stanja. Jednom kada je romobil registriran, iznajmljivač postavlja ponude za iznajmljivanje te za svaku unosi:

- trenutna lokacija romobila
- lokacija povratka romobila
- vrijeme povratka romobila
- cijena iznajmljivanja po prijeđenom kilometru
- iznos novčane kazne u slučaju da romobil ne bude vraćen na vrijeme

Iznajmljivač svoju ponudu može objaviti i na društvenoj mreži Facebook.

Klijent pretražuje oglase i kada se odluči za romobil, može se javiti iznajmljivaču. Komunikacija između klijenta i iznajmljivača odvija se preko poruka putem aplikacije. Iznajmljivač i klijent mogu se detaljnije dogovoriti oko vremena i lokacije preuzimanja romobila. Kada klijent pošalje poruku iznajmljivaču, iznajmljivač dobiva obavijest. Ako stvarno stanje romobila kojeg klijent preuzme ne odgovara stanju kakvo je prikazano na slikama, klijent može podnijeti zahtjev da se stare slike romobila zamijene vlastitim. Također, može i dati kratki opis razlika između starih slika i stvarnog stanja. Ako iznajmljivač smatra da zamjena slika nije utemeljena, tu zamjenu može prijaviti administratoru. Klijent je dužan vratiti romobil u zadano vrijeme i lokaciju povratka, u suprotnom će mu biti izrečena novčana kazna navedena u oglasu.

Administrator je korisnik najveće razine te on ima mogućnost pregleda svih korisnika i može ih brisati. Administrator pregledava prijave iznajmljivača i odlučuje hoće li se zamjena slika izvršiti i li ne. Ako administrator prihvati prijavu iznajmljivača, izmjena slike romobila se neće odviti. Njegova odluka dolazi kao obavijest klijentu i iznajmljivaču. Također, administrator pregledava dostavljenu kopiju osobne iskaznice i potvrdu o nekažnjavanju tije. Ako uspostavi da su dokumenti neispravnji, može odbiti zahtjev za registraciju.

Svaki romobil mora imati mjerac prijeđenih kilometara te on mora biti povezan s aplikacijom kako bi se znalo koliko je kilometara prešao klijent. Kada istekne vrijeme povratka iznajmljivač dobiva obavijest u aplikaciji u kojoj je navedeno koliko je kilometara klijent prešao. Klijent također dobiva obavijest u kojoj su napisani detalji transakcije.

Nakon što je iznajmljivanje završeno, iznajmljivač može ocijeniti klijenta i na-

pisati komentar. Ocjene i komentari su vidljivi na profilima korisnika.

Korisnik na svom profilu može pregledati podatke o prethodnim vožnjama, ili ako se radi o iznajmljivaču podatke o prethodnim najmovima romobila.

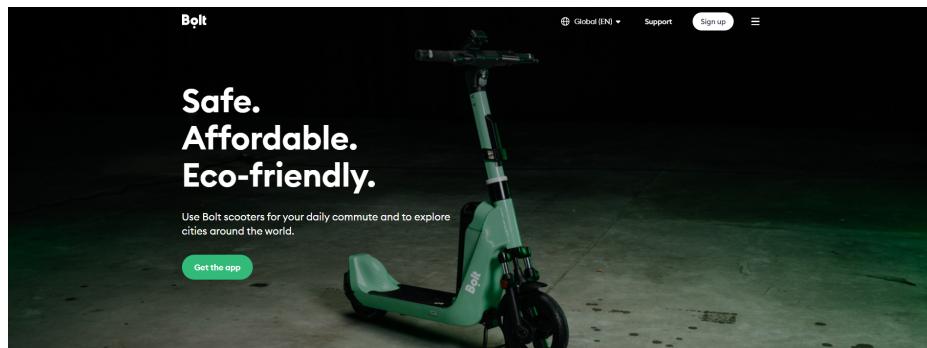
Korist ovog projekta je omogućiti ljudima jednostavan pristup električnim romobilima. Kvalitetan električni romobil je mnogima preskup, tako da oni koji ga ne posjeduju mogu lako i brzo iznajmiti romobil na dane kada im je potreban. Ljudi koji ne koriste svoj romobil, a ne žele da propada u skladištu, mogu ga iznajmiti drugima i na taj način ostvariti dodatan prihod.

U urbanim sredinama gužve u prometu postaju sveprisutni problem koji otežava svakodnevno kretanje i troši vrijeme. U takvim okolnostima, ljudi traže alternativne metode prijevoza koje su brže, praktičnije i održive. I tu vidimo priliku za našu aplikaciju.

Aplikacija bi mogla biti unaprijeđena tako da klijent ima veću mogućnost filtriranja prikazanih oglasa. Na primjer, može pretraživati oglase samo u okolini njegove lokacije ili romobile koji zadovoljavaju njegove zahtjeve. Mogao bi pretraživati romobile čija maksimalna brzina nije manja od 45 km/h ili prema dometu romobila. Također, moguće je aplikaciju učiniti profitabilnijom za vlasnika aplikacije. Pa bi tako vlasnik aplikacije od svake transakcije mogao dobit određeni postotak. Druga mogućnost bi bila da iznajmljivači moraju plaćati mjesecnu pretplatu kako bi mogli dati svoje romobile u najam.

Tvrtka Bolt, koja je u hrvatskoj prepoznatljiva po uslugama dostave hrane i prijevoza, u pojedinim gradovima nudi mogućnost najma romobila. Pa se tako u glavnom gradu Belgije, Briselu, romobili spremni za najam mogu pronaći gotovo u svakoj ulici. Potrebno je imati novčana sredstva u Bolt aplikaciji, na romobilu se skenira QR kod i spremni ste za vožnju. Cijena se također obračunava po prijeđenom kilometru. A jednom kada ste gotovi odjavite se s romobila, romobil se zaključa i spremjan je za idućeg korisnika, a Vama se automatski provede transakcija.

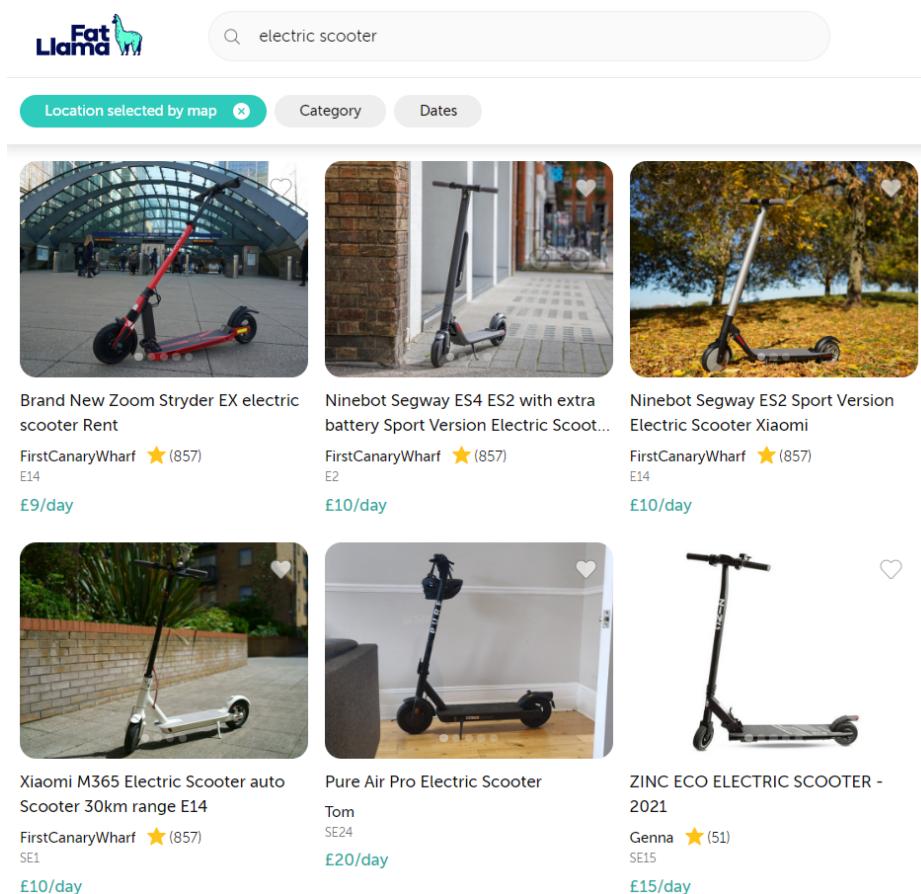
Aplikacija Fat Liama nudi mogućnost najma električnih romobila diljem svijeta. Razlikuje se po tome što je cijena iskazana po danu najma. Korisnik može odabrat Lokaciju, pretražuje ponuđene oglase te ima mogućnost najma na više dana.



Slika 2.1: Početna stranica za najam rombila tvrtke Bolt

The image shows a mobile phone displaying the Bolt app's 'Safety toolkit' section. The screen title is 'How to ride a Bolt scooter'. Below it, there are four sections: 'Find a ride', 'Unlock with the app', 'Follow safety rules', and 'Park with care'. Each section has a small icon and a brief description. To the right of the text, there is a screenshot of the 'Safety toolkit' interface, which includes icons for scooter tracking, insurance, and helmet usage, along with a 'Beginner mode' toggle and a list of safety tips and local rules.

Slika 2.2: Instrukcije za korištenje romobila tvrtke Bolt



Slika 2.3: Najam romobila Fat Llama

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Neregistrirani korisnik
2. Prijavljeni korisnik
3. Klijent
4. Iznajmljivač
5. Administrator

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik može:

- (a) se registrirati u sustav, stvoriti novi korisnički račun sa svim potrebnim podatcima
- (b) pregledavati trenutno dostupne romobile i njihove cijene

2. Prijavljeni korisnik može:

- (a) prijaviti se s postojećim podacima za prijavu u sustav
- (b) pregledavati dostupne romobile
- (c) upravljati svojim osobnim podatcima
- (d) pregledati povijest svojih transakcija

3. Klijent može:

- (a) iznajmljivati dostupne romobile te provjeriti je li slike odgovaraju stvarnom stanju
- (b) zamijeniti sliku koja ne odgovara stvarnom stanju iznajmljenog romobila novom slikom i kratkim opisom
- (c) javiti se iznajmljivaču s porukom i zahtjevom za iznajmljivanje

4. Iznajmljivač može:

- (a) registrirati romobil za što su mu potrebne slike romobila kao dokaz tre-nutnog stanja
- (b) postaviti ponudu za iznajmljivanje i pritom mora unijeti sve relevantne podatke
- (c) objaviti na društvenoj mreži o dostupnosti romobila za iznajmljivanje
- (d) prijaviti administratoru ako je zamjena slika iznajmljenog romobila predložena bez dobrog razloga
- (e) odgovoriti klijentu i prihvati ili odbiti ponudu
- (f) ocijeniti klijenta i napisati komentar

5. Administrator može:

- (a) pregledava prijave iznajmljivača i odlučuje hoće li se zamjena slika izvršiti ili ne
- (b) pregledava dokumente poslane prilikom registracije te registraciju odo-brava ili odbija
- (c) zabraniti pristupa svim korisnicima

6. Baza podataka (sudionik) može:

- (a) pohranjuje sve podatke o korisnicima i njihovim romobilima
- (b) pohranjuje sve podatke o transakcijama

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled romobila

- **Glavni sudionik:** Prijavljeni korisnik
- **Cilj:** Pregledati romobile i njihove cijene
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Prikazani su dostupni romobili
 2. Korisnik odabire romobil
 3. Prikazuju se informacije i slike romobila te uvjeti za iznajmljivanje

UC2 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Korisnik unosi potrebne korisničke podatke
 3. Sustav provjerava ispravnost podataka
 4. Korisnik prima obavijest o uspješnoj registraciji
 5. Sustav preusmjerava korisnika na stranicu s aktivnim oglasima
- **Opis mogućih odstupanja:**
 - 2.a Odabir već zauzetog e-maila, unos korisničkog podatka u nedozvoljenom formatu
 1. Sustav obavještava korisnika o neuspjelom pokušaju registracije i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava registraciju ili odustaje od nje

UC3 - Prijava u sustav

- **Glavni sudionik:** Prijavljeni korisnik
- **Cilj:** Dobiti pristup korisničkom sučelju

- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za prijavu
 2. Korisnik unosi potrebne korisničke podatke
 3. Sustav provjerava ispravnost podataka
 4. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Neispravni korisnički podatci
 1. Sustav obavještava korisnika o neuspjelom pokušaju prijave i vraća ga na stranicu za prijavu

UC4 - Pregled dokumenata poslanih prilikom registracije

- **Glavni sudionik:** Administrator
- **Cilj:** Odobravanje ili odbijanje registracije
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen i novi korisnik je registriran
- **Opis osnovnog tijeka:**
 1. Administrator pregledava dokumente poslane prilikom registracije
 2. Registraciju odobrava ili odbija
 3. Odluka dolazi kao obavijest korisniku

UC5 - Pregled osobnih podataka

- **Glavni sudionik:** Prijavljeni korisnik
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za pregled svojih podataka
 2. Aplikacija prikazuje osobne podatke korisnika

UC6 - Promjena osobnih podataka

- **Glavni sudionik:** Prijavljeni korisnik
- **Cilj:** Promijeniti osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen

- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za pregled svojih podataka
 2. Korisnik odabere opciju za promjenu podataka
 3. Korisnik mijenja svoje osobne podatke
 4. Korisnik spremi promjene
 5. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
 - 3.a Neki podatci nisu popunjeni
 1. Sustav javlja da nisu uneseni svi potrebni podaci
 2. Korisnik dodaje potrebne podatke te završava promjenu ili odustaje od nje

UC7 - Registriranje romobila

- **Glavni sudionik:** Iznajmljivač
- **Cilj:** Registracija novog romobila
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registriranje romobila
 2. Korisnik popunjava tražena podatke
 3. Korisnik spremi promjene
 4. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Nisu popunjena sva polja obrasca
 1. Sustav javlja da nisu uneseni svi podaci
 2. Korisnik dodaje potrebne podatke te završava registraciju ili odustaje od nje

UC8 - Postavljanje ponude za iznajmljivanje

- **Glavni sudionik:** Iznajmljivač
- **Cilj:** Stvoriti novu ponudu za iznajmljivanje romobila
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i romobil je registriran
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za stvaranje nove ponude
 2. Korisnik popunjava potrebne podatke

3. Korisnik sprema promjene
 4. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Nisu popunjena sva polja obrasca
 1. Sustav javlja da nisu uneseni svi podaci
 2. Korisnik dodaje potrebne podatke te završava ponudu ili odustaje od nje

UC9 - Objavljanje ponude na društvenoj mreži

- **Glavni sudionik:** Iznajmljivač
- **Cilj:** Objava na društvenoj mreži o dostupnosti romobila za iznajmljivanje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i postoji ponuda za iznajmljivanje
- **Opis osnovnog tijeka:**
 1. Korisnik odabire oglas
 2. Korisnik objavljuje na društvenoj mreži o dostupnosti romobila za iznajmljivanje

UC10 - Kontaktiranje iznajmljivača

- **Glavni sudionik:** Klijent
- **Cilj:** Podnošenje zahtjeva za iznajmljivanje romobila
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen i postoji romobil za iznajmljivanje
- **Opis osnovnog tijeka:**
 1. Klijent odabire romobil koji želi iznajmit
 2. Klijent se javlja iznajmljivaču s porukom i zahtjevom za iznajmljivanje

UC11 - Prihvatanje ponude

- **Glavni sudionik:** Iznajmljivač
- **Cilj:** Odluka o iznajmljivanju romobila
- **Sudionici:** Baza podataka
- **Preduvjet:** Iznajmljivač je prijavljen i postoji zahtjev za iznajmljivanje
- **Opis osnovnog tijeka:**
 1. Iznajmljivač pregledava zahtjev za iznajmljivanje romobila
 2. Iznajmljivač ga prihvata ili odbija
 3. Odluka dolazi kao obavijest klijentu

UC12 - Iznajmljivanje romobila

- **Glavni sudionik:** Klijent
- **Cilj:** Najam romobila i provjera odgovaraju li slike romobila stvarnom stanju
- **Sudionici:** Baza podataka, iznajmljivač
- **Preduvjet:** Klijent je prijavljen i iznajmljivač je potvrdio zahtjev
- **Opis osnovnog tijeka:**
 1. Klijent izvršava plaćanje
 2. Klijent provjerava je li romobil odgovara slikama iz ponude

UC13 - Zamjena slike romobila

- **Glavni sudionik:** Klijent
- **Cilj:** Zamjena neodgovarajuće slike romobila sa novom slikom i kratkim opisom što je drugačije na slici
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent je prijavljen i iznajmio je romobil
- **Opis osnovnog tijeka:**
 1. Klijent odabire sliku koju želi zamijeniti
 2. Klijent učitava novu sliku i opis što je drugačije
 3. Klijent spremi promjene
 4. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Nisu popunjena sva polja
 1. Sustav javlja da nisu uneseni svi podaci
 2. Korisnik dodaje potrebne podatke te završava zamjenu ili odustaje od nje

UC14 - Prijava pogrešne zamjene

- **Glavni sudionik:** Iznajmljivač
- **Cilj:** Prijava administratoru da je zamjena slika predložena bez dobrog razloga
- **Sudionici:** Baza podataka
- **Preduvjet:** Iznajmljivač je prijavljen i romobil je iznajmljen, izvršena je zamjena slika od strane klijenta
- **Opis osnovnog tijeka:**
 1. Klijent odabire promijenjenu sliku

2. Klijent odabire prijavu zamjene s kratkim opisom
 3. Klijent sprema promjene
 4. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Nisu popunjena sva polja
 1. Sustav javlja da nisu uneseni svi podaci
 2. Korisnik dodaje potrebne podatke te završava prijavu ili odustaje od nje

UC15 - Pregled prijave

- **Glavni sudionik:** Administrator
- **Cilj:** Odluka o zamjeni slike
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen i podnesena je prijava zamjene slike
- **Opis osnovnog tijeka:**
 1. Administrator pregledava prijave iznajmljivača
 2. Donosi odluku hoće li se zamjena slika izvršiti ili ne
 3. Odluka dolazi kao obavijest i klijentu i iznajmljivaču
 4. Baza se ažurira

UC16 - Ocjenjivanje klijenta

- **Glavni sudionik:** Iznajmljivač
- **Cilj:** Ocjena klijenta i objava komentara
- **Sudionici:** Baza podataka
- **Preduvjet:** Iznajmljivač je prijavljen i romobil je iznajmljen
- **Opis osnovnog tijeka:**
 1. Iznajmljivač odabire klijenta
 2. Dodjeljuje mu ocjenu i piše komentar
 3. Ocjena i komentar se objavljuju na profilu.
 4. Baza se ažurira
- **Opis mogućih odstupanja:**
 - 2.a Nisu popunjena sva polja
 1. Sustav javlja da nisu uneseni svi podaci
 2. Korisnik dodaje potrebne podatke te završava ocjenu ili odustaje od nje

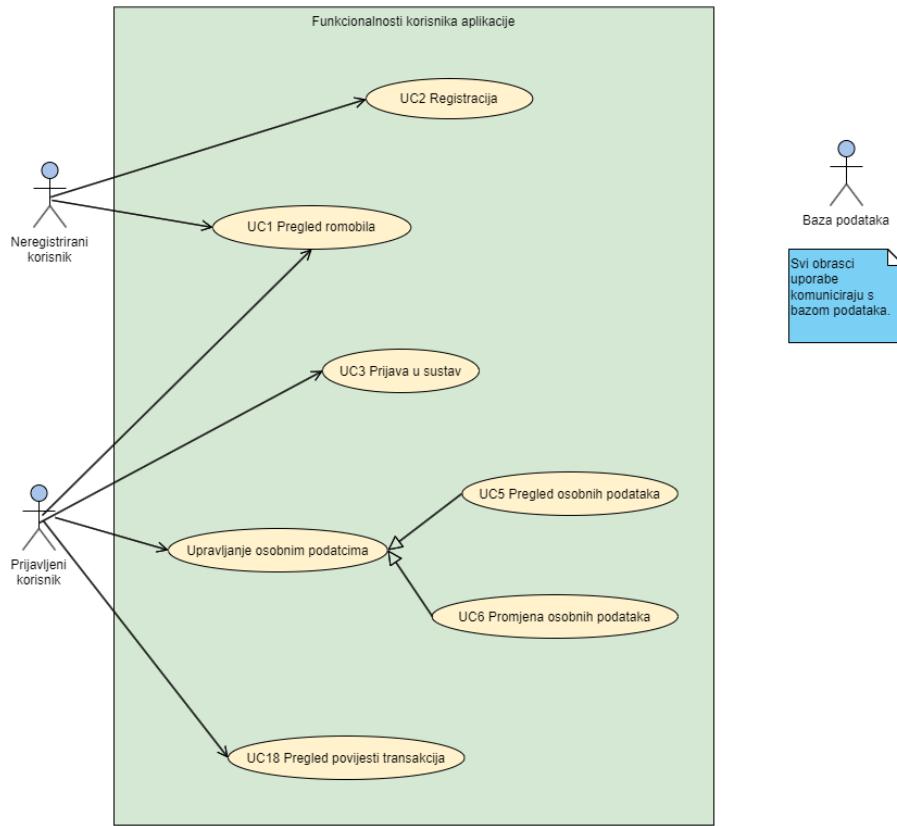
UC17 - Zabrana pristupa korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Zabrana pristupa određenom korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. Administrator odabire korisnika
 2. Zabranjuje mu pristup aplikaciji
 3. Baza se ažurira

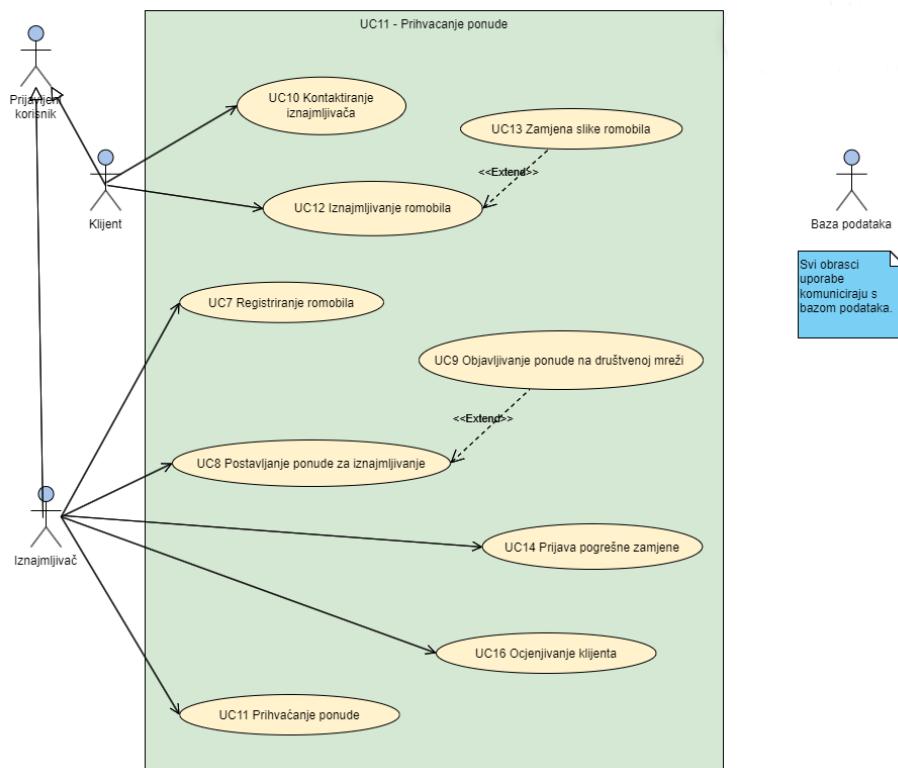
UC18 - Pregled povijesti transakcija

- **Glavni sudionik:** Prijavljeni korisnik
- **Cilj:** Pregledati povijest transakcija
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik odabire pregled svojih transakcija
 2. Aplikacija prikazuje povijest transakcija korisnika

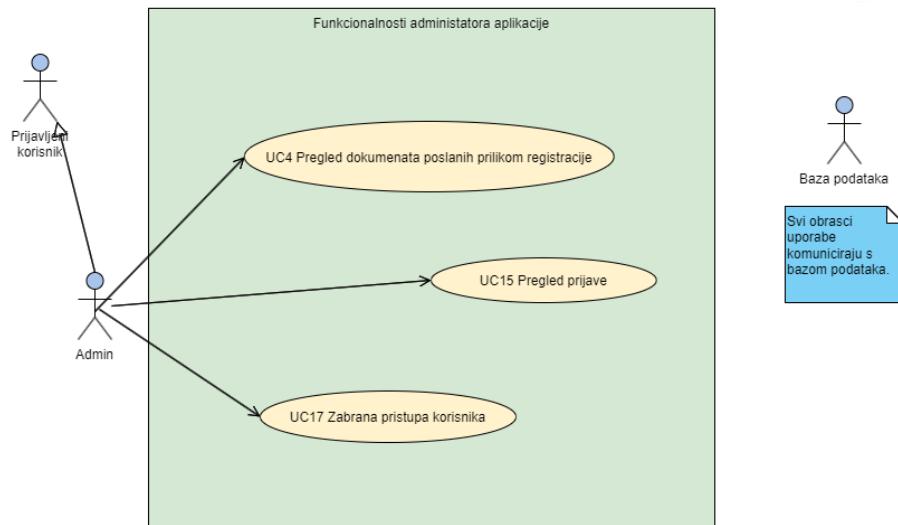
Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnost korisnika



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost klijenta i iznajmljivača

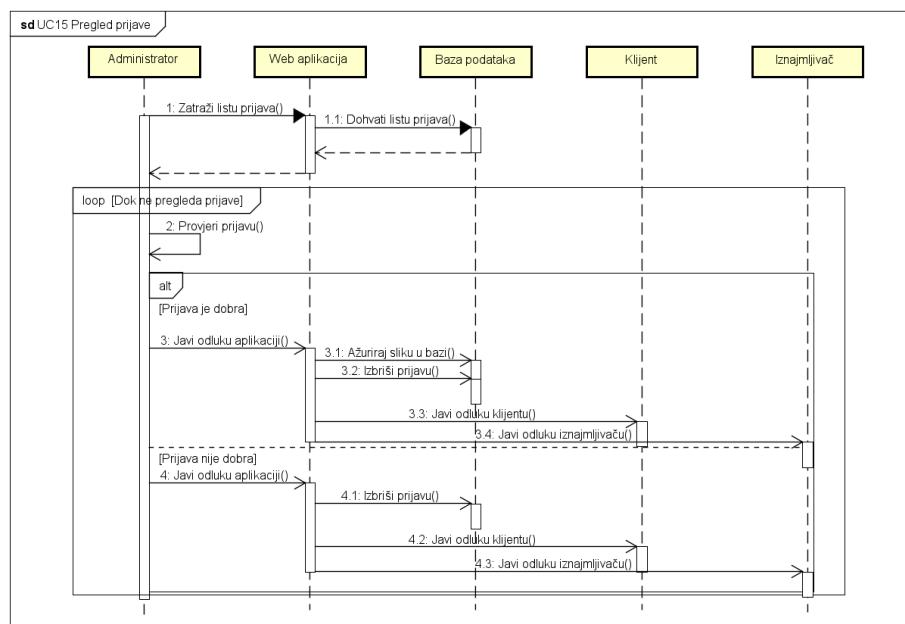


Slika 3.3: Dijagram obrasca uporabe, funkcionalnost administratora

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC15 – Pregled prijave

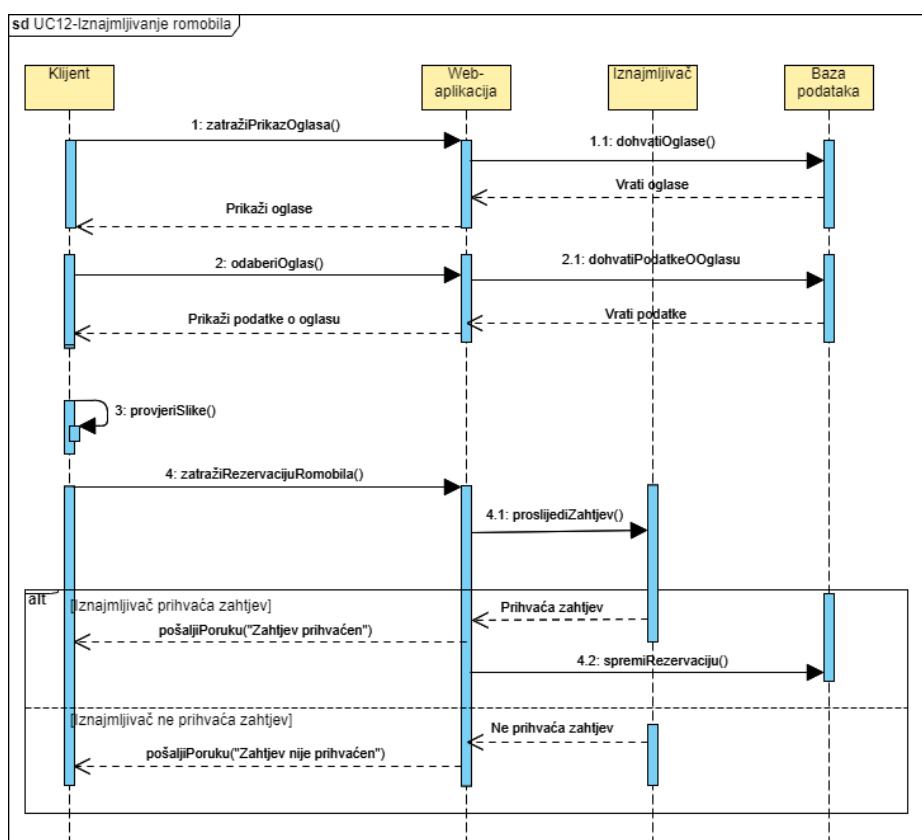
Administrator dobiva obavijest da je iznajmljivač podnio prijavu na zamjenu slika koju je izvršio klijent. Administrator pregledava izmjenjene slike i odlučuje hoće li se zamjena slika izvršiti ili ne. Administratorova odluka dolazi kao obavijest iznajmljivaču i korisniku. Ako je prijava potvrđena nema promjena u bazi, a u slučaju da nije, slike romobila se vraćaju na one prije izmjene.



Slika 3.4: Sekvencijski dijagram za UC15

Obrazac uporabe UC12 – Iznajmljivanje romobila

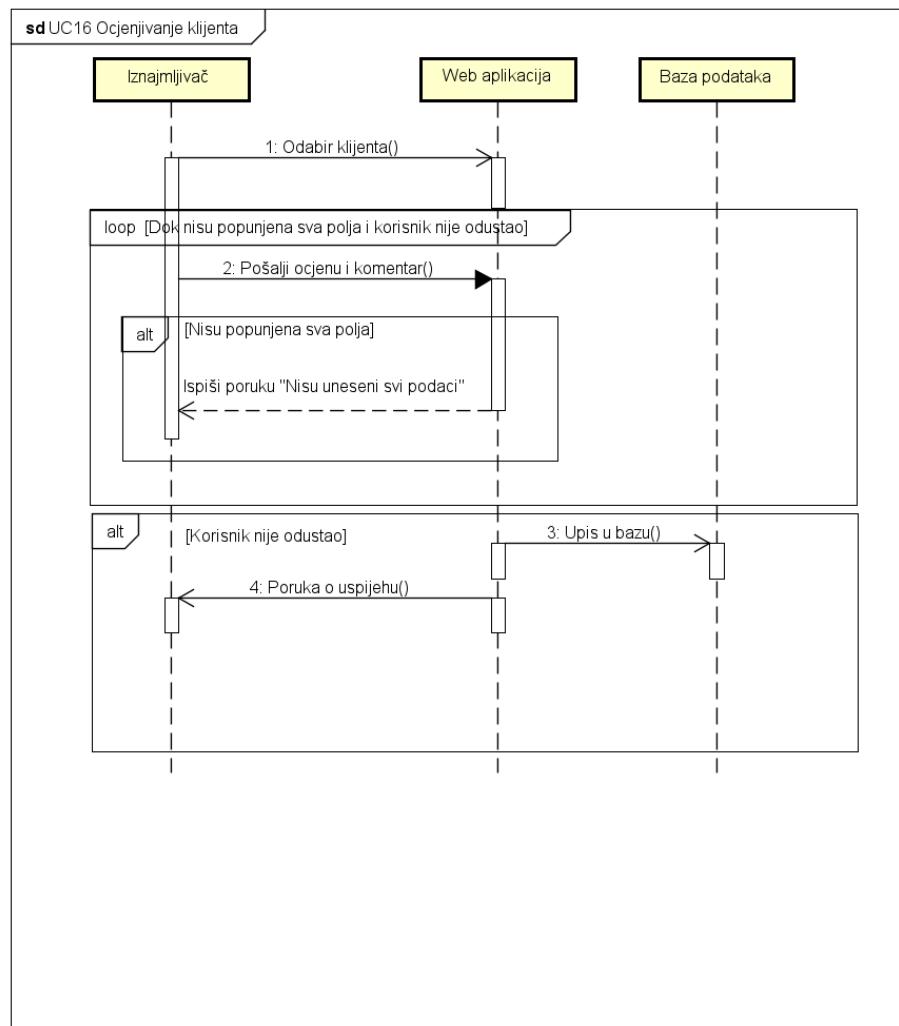
Klijent inicira zahtjev za pregled dostupnih oglasa kako bi odabrao željeni romobil. Poslužitelj potom pristupa trenutnim oglasima iz baze podataka i prikazuje ih klijentu. Nakon odabira oglasa, poslužitelj povlači sve relevantne podatke o odabranom romobilu i predstavlja ih klijentu. Korisnik ima mogućnost provjeriti usklađenost prikazanih fotografija romobila s njegovim stvarnim stanjem. Kada klijent doneše odluku, šalje zahtjev za iznajmljivanje romobila iznajmljivaču. Iznajmljivač ima opciju prihvati ili odbiti zahtjev klijenta. U slučaju prihvata, poslužitelj proslijeđuje relevantne podatke o rezervaciji romobila bazi podataka.



Slika 3.5: Sekvencijski dijagram za UC12

Obrazac uporabe UC16 – Ocjenjivanje klijenta

Iznajmljivač odabire klijenta kojemu ona on dodjeljuje ocjenu i komentar. Ukoliko nisu popunjena oba polja za ocjenu i komentar, aplikacija iznajmljivača šalje poruku da nisu uneseni svi podaci te mo daje opciju ponovnog unosa. Tu poruku iznajmljivač dobiva za svaku takvu nepotpunu objavu. Iznajmljivač može tada odustati od objave ili nadopuniti. Ocjena i komentar su nakon toga vidljivi na profilu klijentata.



Slika 3.6: Sekvencijski dijagram za UC16

3.2 Ostali zahtjevi

- Sustav mora podržavati istovremeni rad više korisnika u stvarnom vremenu.
- Unos i prikaz tekstualnog sadržaja putem korisničkog sučelja i sustava trebaju podržavati hrvatsku abecedu, uključujući dijakritičke znakove.
- Izvršavanje dijela programa koji pristupa bazi podataka ne smije trajati dulje od nekoliko sekundi.
- Implementacija sustava treba biti izvedena kao web aplikacija koristeći objektno-orientirane jezike.
- Pogrešna uporaba korisničkog sučelja ne smije utjecati na funkcionalnost i rad sustava.
- Korisnicima mora biti jednostavno koristiti ScootShare sučelje bez potrebe za detaljnim uputama.
- Nadogradnje sustava ne smiju narušavati postojeće funkcionalnosti.
- Sustav koristi EUR kao svoju valutu.
- Veza s bazom podataka mora biti sigurna, brza i otporna na vanjske pogreške.
- Pristup sustavu mora biti omogućen iz javne mreže putem HTTPS-a.

4. Arhitektura i dizajn sustava

Odabrali smo arhitekturu klijent-poslužitelj zbog njezine skalabilnosti i jasne podjele odgovornosti između klijentskog i poslužiteljskog dijela sustava. Ova arhitektura pruža učinkovitu komunikaciju između korisničkog sučelja i servera.

Sustav je organiziran kao *klijent-poslužitelj*, pri čemu klijent sadržava korisničko sučelje i logiku za interakciju s korisnikom. S druge strane, poslužitelj obrađuje zahtjeve klijenata, pristupa bazi podataka i upravlja poslovnim logikama.

Aplikacija je strukturirana na slojevima, gdje frontend koristi React tehnologiju, koja obuhvaća korisničko sučelje i logiku za prikaz podataka korisnicima. Backend, s druge strane, koristi kombinaciju Java, JavaScript i Spring Boot tehnologija, obrađuje poslovnu logiku, upravlja pristupom bazi podataka i komunicira s klijentima. Ova struktura temelji se na *Model-View-Controller (MVC)* arhitekturi, gdje model predstavlja podatke i poslovnu logiku, view je odgovoran za prikazivanje podataka, a controller upravlja korisničkim zahtjevima.

Klijent radi na različitim uređajima, uključujući računala, pametne telefone i tablete, putem web preglednika. Poslužitelj je smješten u lokalnom data centru s potrebnim resursima za podršku poslužiteljskoj logici i bazi podataka. Baza podataka djeluje kao centralno spremište za pohranu podataka, s pristupom koji je kontroliran od strane poslužitelja.

Klijent inicira zahtjeve i prikazuje podatke korisnicima, dok poslužitelj obrađuje zahtjeve, upravlja poslovnim logikama te komunicira s bazom podataka.

Za komunikaciju između klijenta i poslužitelja koristimo HTTP/HTTPS, osiguravajući time sigurnu razmjenu podataka. Za pouzdan prijenos podataka između različitih komponenti sustava koristi se TCP/IP.

Razvojna okolina uključuje korištenje Eclipse i Visual Studio Code za razvoj aplikacije, pružajući programerima učinkovite alate za pisanje, testiranje i održavanje

koda tijekom razvojnog ciklusa.

4.1 Baza podataka

Za potrebe našeg sustava koristiti 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:

- User
- Authority
- Scooter
- ScooterImages
- Listing
- Message
- ImageChangeRequest
- Rental
- Rating
- Transaction

4.1.1 Opis tablica

User entitet sadržava sve važne informacije o korisniku aplikacije. Sadrži atribute: email, nadimak, lozinku, ime, prezime, broj kartice za plaćanje, presliku osobne iskaznice te potvrdu o nekažnjavanju. Ovaj entitet u vezi je Many-to-One s entitetom Authority, sa entitetom Scooter, Message te Rental.

User		
userID	INT	Jedinstveni identifikator korisnika
firstName	VARCHAR	Ime korisnika
lastName	VARCHAR	Prezime korisnika
password	VARCHAR	Hash lozinke

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

User		
CardNumber	VARCHAR	Broj kartice za plaćanje
email	VARCHAR	Email adresa korisnika
idCard	MEDIUMTEXT	Kopija osobne iskaznice
CertificateOfNo CriminalRecord	MEDIUMTEXT	Potvrda o nekažnjavanju korisnika

Authority entitet pohranjuje informacije o ovlastima korisnika. Sadrži atribute: jedinstveni identifikator autorizacije, identifikator korisnika kojem ovlast pripada i naziv ovlasti. U vezi je One-To-Many s entitetom User preko atributa jedinstvenog identifikatora korisnika.

Authority		
authorityID	INT	Jedinstveni identifikator autorizacije
userID	INT	Jedinstveni identifikator korisnika
authority	VARCHAR	Naziv ovlasti

Scooter entitet pohranjuje informacije o romobilu koji se iznajmljuje. Sadrži atribute: jedinstveni identifikator romobila te jedinstveni identifikator vlasnika. U vezi je One-to-Many sa entitom User preko njegovog identifikatora te Many-To-One sa entitetom ScooterImages.

Scooter		
scooterID	INT	Jedinstveni identifikator romobila
ownerID	INT	Jedinstveni identifikator vlasnika romobila

Listing entitet pohranjuje informacije o oglasu za iznajmljivanje romobila. Sadrži atribute: jedinstveni identifikator oglasa, jedinstveni identifikator romobila, lokaciju romobila, lokaciju povratka romobila, krajnje vrijeme povratka romobila, cijenu najma po prijeđenom kilometru, iznos novčane kazne u slučaju ne vraćanja romobila na vrijeme te status oglasa. U vezi je Many-To-One s entitetom Scooter

preko identifikatora romobila.

Listing		
listingID	INT	Jedinstveni identifikator oglasa
scooterID	INT	Jedinstveni identifikator romobila
location	VARCHAR	Lokacija romobila
returnLocation	VARCHAR	Lokacija za povratak romobila
returnByTime	DATETIME	Vrijeme do kada romobil mora biti vraćen
pricePerKilometer	VARCHAR	Cijena iznajmljivanja po kilometru
lateReturnPenalty	VARCHAR	Iznos zakasnine kod vraćanja romobila
status	VARCHAR	Status oglasa (aktivno, završeno)

Message entitet sadržava sve važne informacije o porukama između iznajmljivača i unajmitelja. Sadrži atribute: jedinstveni identifikator poruka, identifikator pošiljatelja i primatelja poruke, vrijeme slanja poruke, tekst poruke i jedinstveni identifikator oglasa na kojeg se korisnik javlja. Ovaj entitet u vezi je Many-to-One s User preko jedinstvenog identifikatora pošiljatelja, u vezi Many-to-One s User preko jedinstvenog identifikatora primatelja i u vezi Many-to-One s Listing preko jedinstvenog identifikatora oglasa.

Message		
messageID	INT	Jedinstveni identifikator poruke
userFromID	INT	Jedinstveni identifikator pošiljatelja
userToID	INT	Jedinstveni identifikator primatelja
listingID	INT	Jedinstveni identifikator oglasa
content	VARCHAR	Sadržaj poruke
sentAt	DATETIME	vrijeme slanja poruke

ImageChangeRequest entitet pohranjuje zahtjeve korisnika za promjenu slike romobila kojeg su unajmili. Sadrži atribute: jedinstveni identifikator najma romobila, novu sliku romobila, identifikator trenutne fotografije koju želimo zamijeniti

te razlog zamjene priloženu uz zahtjev. U vezi je One-To-One sa ScooterImages te Many-To-One sa Rental entitetom.

ImageChangeRequest		
imageChange RequestID	INT	Jedinstveni identifikator zahtjeva za promjenu slike romobila
replacementImage	MEDIUMTEXT	Slika stanja romobila kojom želimo zamijeniti postojeću
message	VARCHAR	Razlog zamjene slike romobila
rentalID	INT	Jedinstveni identifikator najma romobila
imageID	INT	Jedinstveni identifikator fotografije romobila

Rental entitet sadržava sve važne informacije o najmu romobila. Sadrži attribute: jedinstveni identifikator najma romobila, jedinstveni identifikator unajmitelja, jedinstveni identifikator oglasa, početno vrijeme najma romobila i završno vrijeme najma romobila. Ovaj entitet u vezi je Many-to-One s entitetom User preko identifikatora unajmitelja, također Many-to-One s entitetom Listing. U vezi One-to-One s entitetom Transaction te u vezi One-to-One s entitetom Rating preko atributa jedinstveni identifikator najma romobila.

Rental		
rentalID	INT	Jedinstveni identifikator najma romobila
renterID	INT	Jedinstveni identifikator unajmitelja
listingID	INT	Jedinstveni identifikator oglasa
rentalTimeStart	DATETIME	Početno vrijeme najma romobila
rentalTimeEnd	DATETIME	Završno vrijeme najma romobila

Rating entitet sadrži sve bitne informacije o ocjenjivanju korisnika. Sadrži attribute: jedinstveni identifikator ocjene, identifikator najma romobila, ocjenu korisnika, komentar te vrijeme postavljanja ocjene. U vezi je One-To-One sa entitetom

Rental preko identifikatora najma romobila.

Rating		
ratingID	INT	Jedinstveni identifikator ocjene
rentalID	INT	Jedinstveni identifikator najma romobila
grade	VARCHAR	Ocjena korisnika
comment	VARCHAR	Komentar
ratingTime	DATETIME	Vrijeme ocjenjivanja korisnika

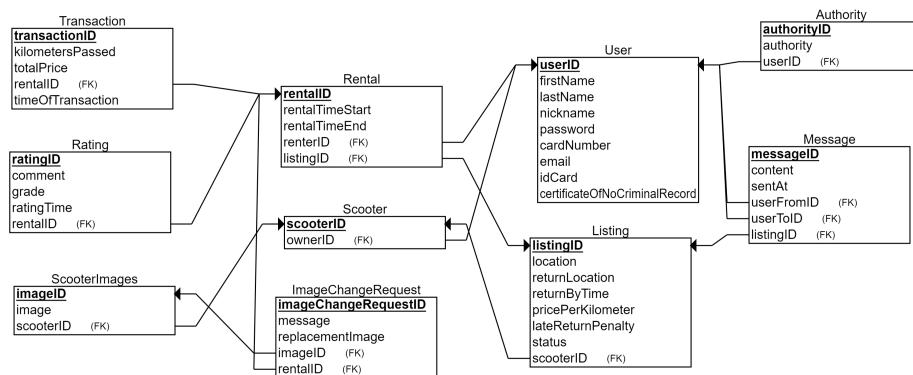
Transaction Ovaj entitet pohranjuje informacije o jednoj transakciji. Sadrži atribute: identifikator transakcije, identifikator najma, broj prijeđenih kilometara te ukupnu cijenu najma. U vezi je One-To-One s entitetom Rental.

Transaction		
transactionID	INT	Jedinstveni identifikator transakcije
rentalID	INT	Jedinstveni identifikator najma romobila
totalPrice	VARCHAR	Ukupna cijena najma romobila
kilometersPassed	VARCHAR	Kilometri prijeđeni romobilom
timeOfTransaction	DATETIME	Vrijeme transakcije

ScooterImages entitet sadrži fotografije vezane za romobil koji se iznajmljuje. Sadrži atribute: identifikator fotografije, identifikator romobila te samu sliku romobila. U vezi je Many-To-One sa entitetom Scooter preko identifikatora romobila.

ScooterImages		
imageID	INT	Jedinstveni identifikator fotografije romobila
scooterID	INT	Jedinstveni identifikator romobila
image	MEDIUMTEXT	Fotografija romobila

4.1.2 Dijagram baze podataka

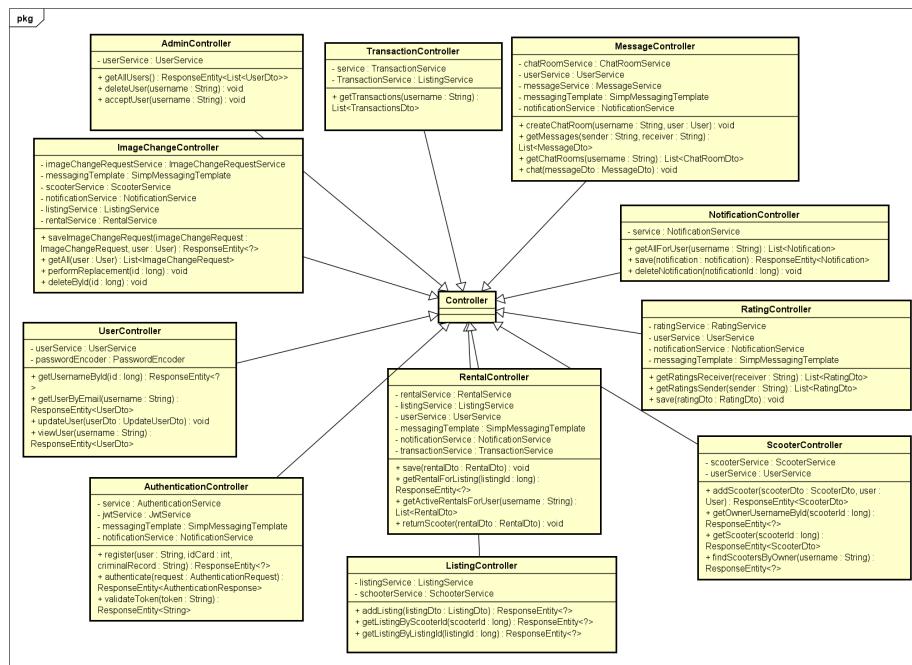


Slika 4.1: E-R dijagram baze podataka

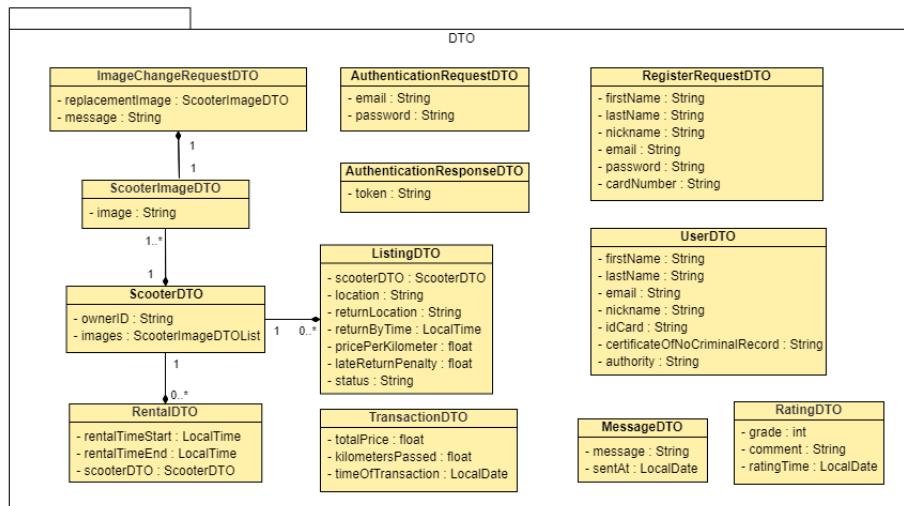
4.2 Dijagram razreda

Dijagrami razreda predstavljaju skup razreda koji su dio backend segmenta MVC arhitekture. Ti razredi nasljeđuju Controller razred. Metode unutar tih razreda manipuliraju s DTO (Data Transfer Object) koji su dohvaćeni kroz implementirane metode u Entitet i Servisi razredima. Kontrolni razredi implementiraju metode koje vraćaju JSON datoteke s odgovarajućim HTML status kodom.

Razredi su logički grupirani prema pravima pristupa određenih aktera. Ova organizacija pomaže smanjiti prenapučenost unutar dijagrama, gdje su prikazane samo ovisnosti između razreda koji pripadaju istom segmentu dijagrama. Informacije o nazivima i tipovima atributa unutar razreda omogućuju zaključivanje vrste ovisnosti među različitim razredima.



Slika 4.2: Dijagram razreda - dio Controllers

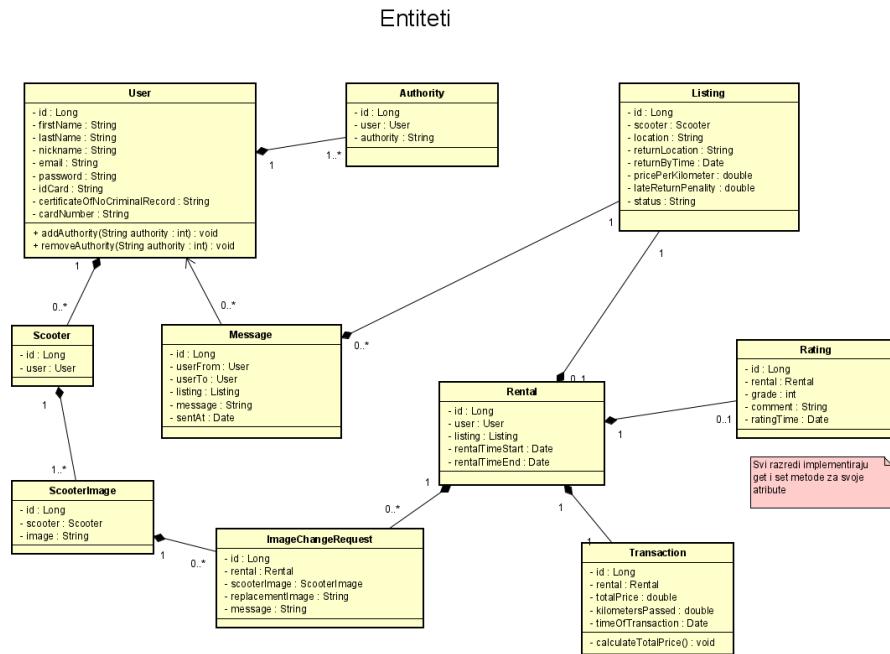


Slika 4.3: Dijagram razreda - dio Data Transfer Objects

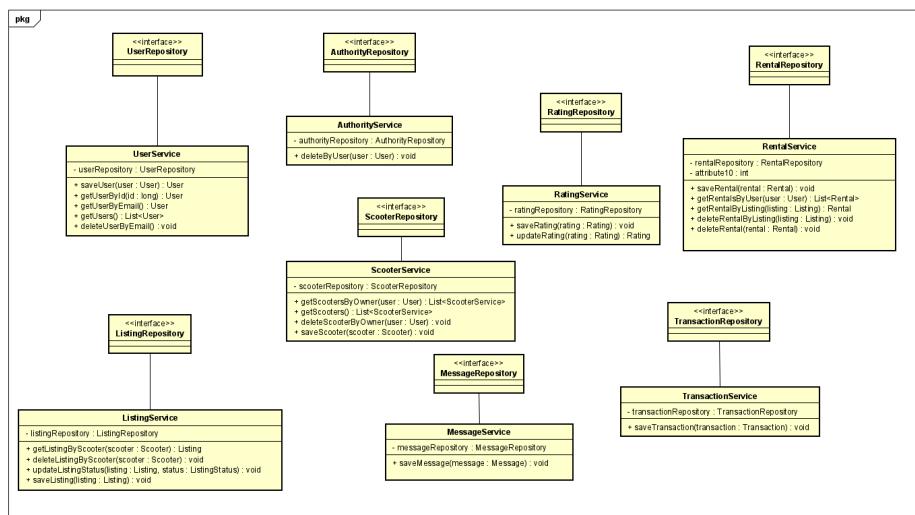
Servisi i Entiteti predstavljaju dvije ključne komponente koji primjenjuju arhitektonski obrazac poznat kao "Service-Oriented Architecture" (SOA) ili "Domain-Driven Design" (DDD).

Entiteti predstavljaju objekte u bazi podataka koji imaju svoj identitet. Svaki entitet ima svoje primarne i sekundarne ključeve kojima su vezani za ostale entitete.

Servisi predstavljaju komponente sustava koje obavljaju određene operacije ili funkcionalnosti. Oni sadrže poslovnu logiku i nude specifične usluge koje se mogu pozvati iz drugih dijelova sustava.



Slika 4.4: Dijagram razreda - dio Entiteti

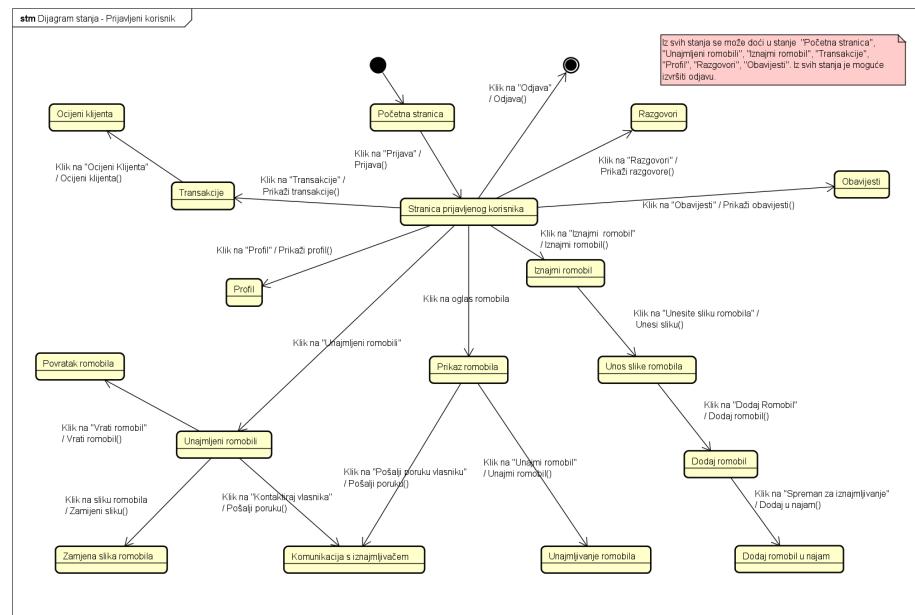


Slika 4.5: Dijagram razreda - dio Servisi

4.3 Dijagram stanja

bice

bice

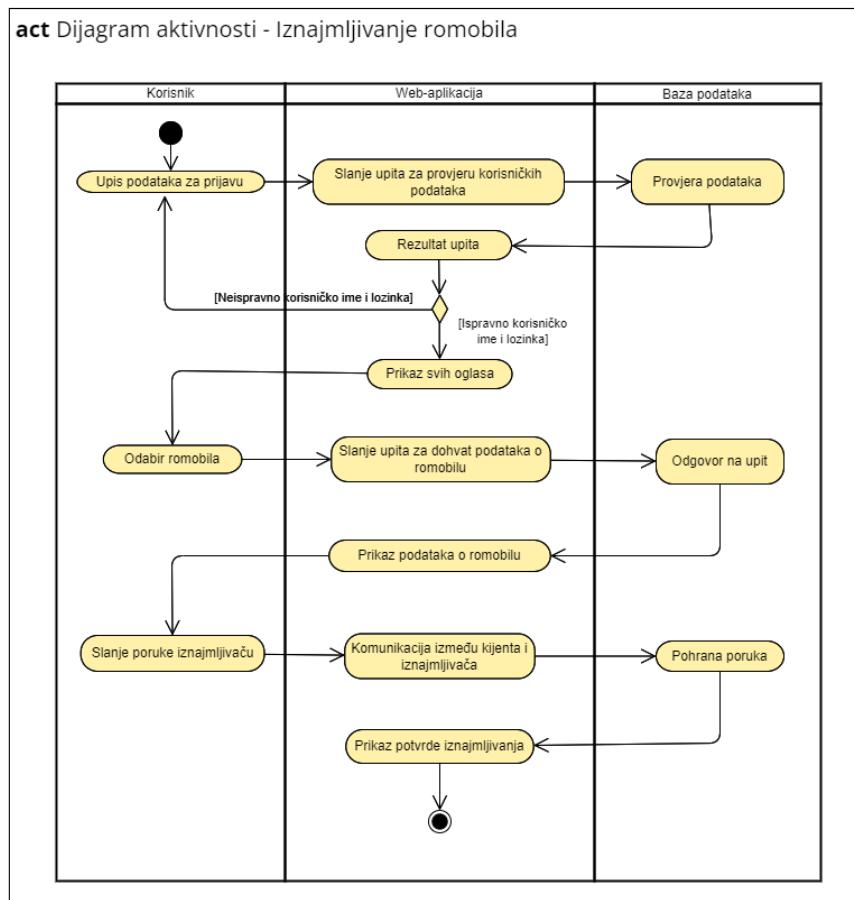


Slika 4.6: Dijagram stanja - Prijavljeni korisnik

4.4 Dijagram aktivnosti

Dijagram aktivnosti primjenjuje se za modeliranje upravljačkog i podatkovnog toka. Pogodan je za opisivanje sinkronizacije i konkurentnosti. Dijagram aktivnosti ne primjenjuje se za modeliranje događajima poticanog ponašanja. U modeliranju toka upravljanja novi korak se izvršava nakon završenog prethodnog.

Dijagram aktivnosti 4.7 prikazuje jednostavan proces iznajmljivanja romobila. Korisnik se prijavljuje, odabire romobil, te nakon kontakta s iznajmljivačem, proces iznajmljivanja postaje aktivan.



Slika 4.7: Dijagram aktivnosti

4.5 Dijagram komponenti

dio 2. revizije

Potrebno je priložiti dijagram komponenti s pripadajućim opisom. Dijagram komponenti treba prikazivati strukturu cijele aplikacije.

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu odvijala se videopozivima putem aplikacije Microsoft Teams¹. Za izradu UML dijagrama korišteni su alati Astah UML² i Visual Paradigm³. Git⁴ je korišten kao sustav za upravljanje kodom. Projekt je dostupan na web platformi GitHub⁵

Za razvoj aplikacije korištena su razvojna okruženja IntelliJ IDE⁶ i Visual Studio Code⁷. IntelliJ jedano je od najpopularnijih razvojnih okruženja za razvoj aplikacija u programskom jeziku Java. Izgradila ga je tvrtka JetBrains. Pruža bogatu podršku za razvoj desktop i web aplikacija. Visual Studio Code je također vrlo popularan uređivač za pisanje programskog koda, osobita za razvoj frontend dijela aplikacija jer pruža izrazito dobru podršku za frotend radne okvire. On je pod vlasništvom tvrtke Microsoft.

Aplikacija je napisana koristeći radni okvir Spring Boot⁸ i jezik Java⁹ za izradu *backenda* te React¹⁰ i jezik JavaScript¹¹ za izradu *frontenda*. React je bilioteka izražena u JavaScriptu koja olakšava izradu korisničkog sučelja. React je održava tvrtka Meta. Radni okvir Spring Boot nadograđuje mogućnosti samog programskog jezika Java te time uvelike olakšava razvoj web aplikacija. Nudi niz gotovih funkcionalnosti koji povećavaju produktivnost i efikasnost programera.

Za bazu podataka korišten je PostgreSQL¹².

¹<https://www.microsoft.com/en-us/microsoft-teams/group-chat-software>

²<https://astah.net/products/astah-uml/>

³<https://online.visual-paradigm.com/>

⁴<https://git-scm.com/>

⁵<https://github.com/>

⁶<https://www.jetbrains.com/idea/>

⁷<https://code.visualstudio.com/>

⁸<https://spring.io/projects/spring-boot/>

⁹<https://www.oracle.com/java/>

¹⁰<https://react.dev/>

¹¹<https://www.javascript.com/>

¹²<https://www.postgresql.org/>

5.2 Ispitivanje programskog rješenja

Za provođenje testiranja naše Spring Boot aplikacije koristili smo biblioteke JUnit i Mockito. One su nam omogućile pouzdano i sustavno testiranje raličitih dijelova aplikacije. Za ispitivanje sustava koristili smo alat Selenium.

5.2.1 Ispitivanje komponenti

U nastavku ovog poglavlja, pronaći ćete detaljan opis šest odabralih ispitnih slučajeva koji obuhvaćaju redovne situacije, rubne uvjete te testiranje ponašanja sustava u situacijama iznimki. Uz opis ispitnih slučajeva, priložili smo i odgovarajući izvorni kôd svakog testa, kao i rezultate izvođenja ispita u razvojnem okruženju, što omogućuje transparentnost i preglednost provedenih ispitivanja.

Prvi test koji smo odradili je dohvaćanje romobila preko njegovog identifikatora. U ovom smo testu testirali servisni sloj naše aplikacije te set test uredno proveo.

```
@Test
@DisplayName("Should retrieve scooter by id")
public void shouldFindScooterById() {
    Scooter scooter = new Scooter(1L, null, new ArrayList<String>(), new ArrayList<Listing>());
    Mockito.when(scooterRepository.findById(1L)).thenReturn(Optional.of(scooter));
    Scooter retrieved = scooterService.findById(1L);
    Assertions.assertThat(scooter.getId()).isEqualTo(retrieved.getId());
}
```

Slika 5.1: Testiranje dohvaćanja romobila preko identifikatora

U sljedećem testu također smo testirali servisni sloj i to dohvaćanje poruka za dva korisnika. Dakle, u testu definiramo dva korisnika te provjeravamo vraća li nam naša aplikacija tražene poruke.

```
@Test
@DisplayName("Should find all the messages for user sender and user receiver")
public void shouldFindMessagesForSenderAndReceiver() {
    String sender = "user1";
    String receiver = "user2";

    List<Message> messages = new ArrayList<>();
    messages.add(new Message(1L, User.builder().username(sender).build(),
        User.builder().username(receiver).build(),
        ChatRoom.builder().id(sender + "_" + receiver).build(), new Date(), "Hello user 2"));
    messages.add(new Message(2L, User.builder().username(receiver).build(),
        User.builder().username(sender).build(),
        ChatRoom.builder().id(sender + "_" + receiver).build(), new Date(), "Hello user 1"));

    Mockito.when(messageRepository.findMessagesBySenderAndReceiver(sender + "_" + receiver))
        .thenReturn(messages);

    List<MessageDto> retrievedMessageDtos = messageService.findByUsers(sender, receiver);
    Assertions.assertThat(retrievedMessageDtos.size()).isEqualTo(messages.size());
    Assertions.assertThat(retrievedMessageDtos.get(0).getSender())
        .isEqualTo(messages.get(0).getSender().getUsername());
    Assertions.assertThat(retrievedMessageDtos.get(0).getContent())
        .isEqualTo(messages.get(0).getContent());
    Assertions.assertThat(retrievedMessageDtos.get(1).getSender())
        .isEqualTo(messages.get(1).getSender().getUsername());
    Assertions.assertThat(retrievedMessageDtos.get(1).getContent())
        .isEqualTo(messages.get(1).getContent());
}
```

Slika 5.2: Testiranje dohvaćanja poruka za zadene korisnike

```

    @Test
    @DisplayName("Should add new listing")
    @WithMockUser(username = "test")
    public void testAddingNewListing() throws Exception {
        ListingDto listingDto = ListingDto.builder()
            .scooterId(1L)
            .location("Zagreb")
            .build();

        Mockito.when(scooterService.findById(1L)).thenReturn(
            Scooter.builder().id(1L).build());
        mockMvc.perform(post("/api/listings/add")
            .contentType(MediaType.APPLICATION_JSON_VALUE)
            .content(convertObjectToJson(listingDto))
            .andExpect(status().is(200)));
    }

    @Test
    @DisplayName("Should throw exception listing for given id does not exist")
    @WithMockUser(username = "test")
    public void testThrowExceptionForNonExistingListing() throws Exception {
        assertThatThrownBy(() ->
            mockMvc.perform(get("/api/listings/getOneListing/1"))
                .andExpect(status().is(500)))
            .hasCauseInstanceOf(NullPointerException.class);
    }
}

```

Slika 5.3: Testiranje ListingController-a

Sada ćemo prikazati testiranje ListingController. U prvom testu testirali smo dodavanje novog oglasa te je očekivani ishod odgovor poslužitelja sa novim oglasom, dok smo u drugom pokušali dohvati oglas za nepostojeći romobil. U drugom testu očekujemo da poslužitelj baci novu NullPointerException iznimku. Oba testa su se izvršila ispravno.

Korisnik mora moći pregledavati romobile koje je trenutno iznajmio pa smo testirali i RentalController i to metodu koja osigurava dohvaćanje trenutnih najmova. U testu definiramo koje objekte tipa Rental očekujemo te kada primimo odgovor od poslužitelja provjeravamo jesmo li ih uistinu i primili.

```

    @Test
    @DisplayName("Should get all the rentals for given user")
    @WithMockUser(username = "test")
    public void testGetRentalsForUser() throws Exception {
        Rental rental1 = Rental.builder()
            .id(1L)
            .listing(Listing.builder()
                .id(1L)
                .scooter(Scooter.builder().owner(User.builder().username("user2").build()).build())
                .build())
            .scooterRenter(User.builder().id(1L).username("user1").build())
            .build();
        Rental rental2 = Rental.builder()
            .id(2L)
            .listing(Listing.builder()
                .id(2L)
                .scooter(Scooter.builder().owner(User.builder().username("user3").build()).build())
                .build())
            .scooterRenter(User.builder().id(1L).username("user1").build())
            .build();
        List<Rental> rentals = new ArrayList<>();
        rentals.add(rental1);
        rentals.add(rental2);
        User user = User.builder().id(1L).username("user1").build();

        Mockito.when(rentalService.findRentalsByUser(user))
            .thenReturn(rentals);

        Mockito.when(userService.findUserByUsername("user1"))
            .thenReturn(user);

        mockMvc.perform(get("/api/rentals/getRentalsForUser/user1"))
            .andExpect(status().is(200))
            .andExpect(content().contentType(MediaType.APPLICATION_JSON_VALUE))
            .andExpect(jsonPath("$.size()", Matchers.is(2)))
            .andExpect(jsonPath("$.get(0).listingId", Matchers.is(1)))
            .andExpect(jsonPath("$.get(1).listingId", Matchers.is(2)))
            .andExpect(jsonPath("$.get(0).rentalUser.username", Matchers.is("user1")))
            .andExpect(jsonPath("$.get(1).rentalUser.username", Matchers.is("user1")))
            .andExpect(jsonPath("$.get(0).scooterOwner", Matchers.is("user2")))
            .andExpect(jsonPath("$.get(1).scooterOwner", Matchers.is("user3")));
    }
}

```

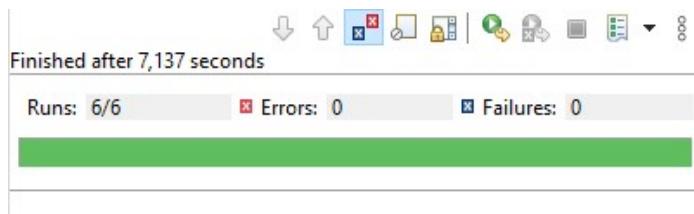
Slika 5.4: Testiranje dohvaćanja najmova za korisnika

Na kraju, provjerili smo što će se dogodit ako pokušamo izbrisat ocjenu i komentar. Poslužitelj je prepoznao da ta funkcionalnost nije implementirana te ispravno odgovorio sa statusom 404 Not found.

```
@Test  
@DisplayName("Should return status 404 Not Found since deleting of ratings is not implemented")  
public void testDeletingRatingShouldNotBeImplemented() throws Exception {  
    mockMvc.perform(delete("/api/ratings/delete/1")  
        .andExpect(status().is(404));  
}
```

Slika 5.5: Testiranje brisanja komentara i ocjena

Sljedeća slika prikazuje da su se svi testovi uspješno proveli.



Slika 5.6: Uspješno provođenje testova

5.2.2 Ispitivanje sustava

Za ispitivanje sustava koristili smo alat Selenium WebDriver. Kako bi ga mogli koristiti u projekt smo dodali dependency za Selenium WebDriver. Također, bilo je potrebno i instalirati Chrome WebDriver.

U prvom testu testirali smo dvije funkcionalnosti: dodavanje novog romobila i dodavanje oglasa za taj romobil. Korisnik se prvo prijavi u aplikaciju. S obzirom na to da su svi testovi započinjali s prijavom korisnika u aplikaciju, prijavu smo izdvojili u zasebnu funkciju. Koristeći objekt tipa WebDriver provjerili smo je li se korisnik uspješno prijavio u sustav te ako je, pozicionirali smo se na stranicu za pregled romobila. Da bi dodali novi romobil, bilo je potrebno unijeti sliku romobila i to smo učinili metodom sendKeys koju nudi WebDriver. Kliknuli smo na gumb za dodavanje romobila te provjerili je li se romobil uspješno dodao. Zatim smo kliknuli na gumb za dodavanje oglasa, ponovo metodom sendKeys unijeli sve zahtijevane informacije za dodavanje oglasa, dodali oglas i na kraju provjerili je li se oglas uspješno dodao.

```

@test
@DisplayName("User logs in application, goes to rent scooter page, adds new scooter to his scooter list and adds new listing for that scooter")
public void testAddScooterAndListing() {
    System.setProperty("webdriver.chrome.driver", "C:\\Program Files (x86)\\chromedriver-wind64\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("https://scoot-share.onrender.com/");

    loginAsUser(driver, username: "dino", password: "dinobabic");

    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    wait.until(ExpectedConditions.not(ExpectedConditions.urlContains("faction: /login")));
    Assertions.assertThat(driver.getCurrentUrl()).contains("/login");

    driver.findElement(By.className("hamburger-menu-button")).click();
    driver.findElement(By.className("rent-scooter-btn")).click();
    driver.findElement(By.className("add-scooter-image-input")).sendKeys(KeysToSend: "C:\\Users\\korisnik\\Downloads\\scooter-3.jpg");
    driver.findElement(By.className("submit-scooter-btn")).click();

    Assertions.assertEquals(expected: 1, driver.findElements(By.tagName("tr")).size());

    driver.findElement(By.className("ready-to-rent-btn")).click();
    driver.findElement(By.className("location-input")).sendKeys(KeysToSend: "Zagreb");
    driver.findElement(By.className("return-location-input")).sendKeys(KeysToSend: "Zagreb");
    driver.findElement(By.className("price-per-kilometer-input")).sendKeys(KeysToSend: "2.5");
    driver.findElement(By.className("penalty-input")).sendKeys(KeysToSend: "25");
    driver.findElement(By.className("create-listing-btn")).click();

    Assertions.assertTrue(driver.findElement(By.className("scooter-rented-label")).isDisplayed());
}

```

Slika 5.7: Testiranje dodavanja romobila i oglasa za romobil

```

private void loginAsUser(WebDriver driver, String username, String password) {
    driver.findElement(By.className("hamburger-menu-button")).click();
    driver.findElement(By.className("login-btn")).click();

    driver.findElement(By.className("username-input")).sendKeys(KeysToSend: username);
    driver.findElement(By.className("password-input")).sendKeys(KeysToSend: password);

    driver.findElement(By.className("submit-login-btn")).click();
}

```

Slika 5.8: Funkcija za prijavu korisnika preko objekta tipa WebDriver

U sljedećem testu provjerili smo mogućnost unajmljivanja romobila. Prijavili smo se u aplikaciju i odabrali smo prvi oglas na početnoj stranici. Očekivano ponašanje je da nas aplikacija usmjeri na stranicu koja prikazuje detalje o odrabljeno oglasu te smo to i testirali. Ako je ta provjera uspješno prošla, na stranici smo kliknuli gumb za unajmljivanje romobila. Testirali smo je li nas aplikacija preusmjerila na stranicu s prikazom trenutno unajmljenih romobila i je li broj trenutno unajmljenih romobila jednak jedan.

```

@Test
@DisplayName("User logs into application and rents scooter")
public void testRentScooter() {
    System.setProperty("webdriver.chrome.driver", "C:\\Program Files (x86)\\chromedriver-wind64\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("https://scoot-share.onrender.com/");

    loginAsUser(driver, username: "dino", password: "dinobabic");

    driver.findElement(By.className("listing-0")).click();
    Assertions.assertTrue(driver.getCurrentUrl().contains("/listing"));

    driver.findElement(By.className("rent-btn")).click();
    Assertions.assertTrue(driver.getCurrentUrl().contains("/my-rentals"));

    Assertions.assertEquals(expected: 1, driver.findElements(By.className("rental-item")).size());
}

```

Slika 5.9: Testiranje unajmljivanja romobila

U posljednjem testu provjerili smo funkcionalnost dopisivanja između dva ko-

```

    @Test
    @DisplayName("Testing chat between two users")
    public void testChat() throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "C:\\\\Program Files (x86)\\\\chromedriver-win64\\\\chromedriver.exe");

        WebDriver driverDino = new ChromeDriver();
        driverDino.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driverDino.get("https://scoot-share.onrender.com/");

        WebDriver driverMarko = new ChromeDriver();
        driverMarko.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driverMarko.get("https://scoot-share.onrender.com/");

        loginAsUser(driverDino, username: "dino", password: "dinobabic");
        loginAsUser(driverMarko, username: "marko", password: "markomarkic");

        driverDino.findElement(By.className("hamburger-menu-button")).click();
        driverDino.findElement(By.className("chat-btn")).click();
        Assertions.assertTrue(driverDino.getCurrentUrl().contains("/chat"));

        driverMarko.findElement(By.className("hamburger-menu-button")).click();
        driverMarko.findElement(By.className("chat-btn")).click();
        Assertions.assertTrue(driverMarko.getCurrentUrl().contains("/chat"));

        driverDino.findElement(By.className("room-marko")).click();
        driverMarko.findElement(By.className("room-dino")).click();

        Assertions.assertEquals(expected: 0, driverDino.findElements(By.className("message")).size());
        Assertions.assertEquals(expected: 0, driverMarko.findElements(By.className("message")).size());

        driverDino.findElement(By.className("message-content-input")).sendKeys(_keysToSend: "Pozdrav, Marko");
        driverDino.findElement(By.className("send-message-btn")).click();

        Thread.sleep( millis: 500);

        Assertions.assertEquals(expected: 1, driverDino.findElements(By.className("message")).size());
        Assertions.assertEquals(expected: 1, driverMarko.findElements(By.className("message")).size());

        driverMarko.findElement(By.className("message-content-input")).sendKeys(_keysToSend: "Pozdrav, Dino");
        driverMarko.findElement(By.className("send-message-btn")).click();

        Thread.sleep( millis: 500);

        Assertions.assertEquals(expected: 2, driverDino.findElements(By.className("message")).size());
        Assertions.assertEquals(expected: 2, driverMarko.findElements(By.className("message")).size());
    }
}

```

Slika 5.10: Testiranje razmjene poruka između korisnika

risnika. Kreirali smo dva objekta tipa WebDriver, jedan za korisnika Dino i jedan za korisnika Marko. Prijavili smo oba korisnika u aplikaciju te smo ih koristeći njihove WebDriver objekte pozicionirali na stranicu za dopisivanje. Testirali smo je li početni broj poruka između ova dva korisnika jednak nuli. Nakon toga korinik Dino je poslao poruku korisniku Marko. S obzirom na to da je potrebno određeno vrijeme da poruka pristigne do Marka, pozvali smo Thread.sleep(500). Kada se dretva probudila, provjerili smo je li sada broj poruka jednak jedan. Slično smo ponovili još jednom, samo što je sada Marko slao poruku Dini.

5.3 Dijagram razmještaja

dio 2. revizije

Potrebno je umetnuti **specifikacijski dijagram razmještaja** i opisati ga. Moguće je umjesto specifikacijskog dijagrama razmještaja umetnuti dijagram razmještaja instanci, pod uvjetom da taj dijagram bolje opisuje neki važniji dio sustava.

5.4 Upute za puštanje u pogon

Našu aplikaciju odlučili smo pustit u pogon preko platforme Render. U ovom poglavlju objasnit ćemo korake koje smo proveli kako bi uspješno pustili u pogon aplikaciju.

Konfiguracija baze podataka

Na web stranici Render potrebno je odabrat izradu nove baze podataka te unijeti željeno ime baze te verziju PostgreSQL-a. Sljedeća slika prikazuje izradu baze podataka.

New PostgreSQL

[Read the docs](#)

Name	scoot_share_db	
Database	Optional The PostgreSQL "dbname"	randomly generated unless specified
User	Optional	randomly generated unless specified
Region	The region where your PostgreSQL instance runs. Services must be in the same region to communicate privately and you currently have services running in Frankfurt.	
PostgreSQL Version	15	
Datadog API Key	Optional The API key to use for sending metrics to Datadog. Setting this will enable Datadog monitoring.	

Slika 5.11: Izrada baze podataka

Konfiguracija backenda

Kako bi mogli pustit u pogon našu poslužiteljsku stranu potrebno je učiniti određene izmjene. Potrebno je preuređiti application.properties file na način da se dodaju globalne varijable koje će se koristit na Render-u. Također, treba dodati i Dockerfile koji sadrži niz instrukcija za izgradnju Docker kontejnera.

Nakon što je prvi korak odrđen potrebno je konfigurirati backend na Renderu. Odabiremo novi Web Service, nakon toga povežemo naš github repository s Ren-

Slika 5.12: Dockerfile i application.properties

derom.

Name	scoot_share_backend
Region	Frankfurt (EU Central)
Branch	deploy
Root Directory <small>Optional</small>	/Izvornikod
Runtime	Docker

Slika 5.13: Kreiranje backenda na Renderu

Konfiguracija frontend-a Potrebno je u našoj frontend React aplikaciji u dатотeci package.json dodati ovisnosti prema http-proxy-middleware, dotenv, express. Zatim treba dodati datoteku /src/setupProxy.js koja služi kao proxy server za lokalni development (redirecta api pozive na localhost:8080), odnosno kad se koristi react-scripts start skripta. Posljednja datoteka koju treba dodati je app.js u kojoj se nalazi express server za produkcijski proxy i posluzivanje frontend-a. Jednom kada su sve izmjene napravljene i pohranjene u repozitorij kreirajmo Web Servis na Renderu za frontend.

Name	scoot_share_frontend
Region	Frankfurt (EU Central)
Branch	deploy
Root Directory <small>Optional</small>	/Izvornikod/scoot_share_frontend
Runtime	Node
Build Command	/Izvornikod/scoot_share_frontend/ \$ yarn build

Slika 5.14: Kreiranje frontend-a na Renderu

Renderu će trebati jedno vrijeme da pokrene aplikacije te kada završi prikazat će URL adresu preko koje možemo pristupiti našoj aplikaciji. S obzirom da je Ren-

der nudi besplatno puštanje u pogon za očekivat je da će aplikacija raditi sporije, no za naše potrebe je to bilo sasvim dovoljno.

6. Zaključak i budući rad

Zadatak naše grupe bila je izrada web aplikacije za iznajmljivanje romobila. Glavne funkcionalnosti aplikacije su pregled romobila, iznajmljivanje i unajmljivanje romobila. Tijekom čitavog semestra radili smo na razvoju projekta te smo nakon gotovo 15 tijedana projekt uspješno priveli kraju.

U prvoj fazi projekta, naš tim se okupio radi razvoja aplikacije. Ovaj početni korak uključivao je dodjelu projektnog zadatka te usmjeren rad na dokumentiranje zahtjeva. Znali smo da će nam bez detaljnog i kvalitetnog dokumentiranja zahtjeva biti puno teže te smo zato ozbiljno prisupili tom dijelu. Tijekom ove faze, posvetili smo se izradi obrazaca uporabe, sekvensijskih dijagrama, modela baze podataka te dijagrama razreda. Ovi dokumenti pokazali su se ključnim prilikom suradnje s podtimovima odgovornim za razvoj backend-a i frontend-a. Vizualni prikazi idejnih rješenja problemskih zadataka, uključujući dijagrame i obrasce, pokazali su se izuzetno korisnima. Njihova izrada omogućila je jasno razumijevanje koncepta i uštedjela mnogo vremena tijekom drugog ciklusa razvoja, posebice kada su se pojavile nedoumice oko implementacije rješenja.

U drugoj fazi projekta, fokus se pomaknuo prema implementaciji naše aplikacije. S progresijom rada na aplikaciji, suočavali smo se s naprednjim koncepcijama, što je zahtjevalo dodatno samostalno proučavanje alata i programske jezika. Osim što smo aktivno radili na implementaciji aplikacije, bilo je nužno posvetiti se dokumentiranju preostalih zahtjeva i izradi potrebne popratne dokumentacije. Drugom fazom dominirala je intenzivna faza samostalnog rada gdje su članovi tima aktivno istraživali i primjenjivali stečena znanja u rješavanju izazova tehničkog razvoja. Ovaj period samostalnosti pridonio je značajnom jačanju kompetencija tima, otvarajući put uspješnoj realizaciji postavljenih projektnih ciljeva. Ovaj intenzivan rad na razvoju aplikacije omogućio nam je dublje razumijevanje tehničkih aspekata projekta.

Korištenje videopoziva putem aplikacije Microsoft Teams pružilo nam je izvrsnu platformu za održavanje produktivnih sastanaka, olakšavajući fluidnu razmjenu informacija i znanja unutar tima.

Za većinu članova tima ovo je bio prvi put da rade na razvoju aplikacije u timu,

stoga je to izuzetno vrijedno iskustvo za sve nas. S ozbirom da ćemo u budućnosti gotovo sigurno raditi u timovima, drago nam je da smo se s tim načinom rada susreli vćć na faksu. Iako aplikacija ima svojih mana i mjesta za napredak, ponosni smo na postignuće koje smo ostvarili.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzzemris>
2. Nikolina Frid, Alan Jović, Modeliranje programske potpore UML-dijagramima
[https://www.fer.unizg.hr/_download/repository/UML_zadaci_za_vjezbu_-nadopuna_sveucilisnog_prirucnika\[1\].pdf](https://www.fer.unizg.hr/_download/repository/UML_zadaci_za_vjezbu_-nadopuna_sveucilisnog_prirucnika[1].pdf)
3. Spring Boot dokumentacija <https://spring.io/projects/spring-boot/>
4. React dokumentacija <https://react.dev/learn>
5. WebSocket Wikipedia <https://en.wikipedia.org/wiki/WebSocket>
6. WebSocket tutorial <https://www.youtube.com/watch?v=7T-HnTE6v64&t=2961s>
7. The Unified Modeling Language, <https://www.uml-diagrams.org/>
8. Astah Community, <http://astah.net/editions/uml-new>

Indeks slika i dijagrama

2.1 Početna stranica za najam rombila tvrtke Bolt	8
2.2 Instrukcije za korištenje romobila tvrtke Bolt	8
2.3 Najam romobila Fat Liama	9
3.1 Dijagram obrasca uporabe, funkcionalnost korisnika	19
3.2 Dijagram obrasca uporabe, funkcionalnost klijenta i iznajmljivača .	20
3.3 Dijagram obrasca uporabe, funkcionalnost administratora	20
3.4 Sekvencijski dijagram za UC15	21
3.5 Sekvencijski dijagram za UC12	22
3.6 Sekvencijski dijagram za UC16	23
4.1 E-R dijagram baze podataka	31
4.2 Dijagram razreda - dio Controllers	32
4.3 Dijagram razreda - dio Data Transfer Objects	33
4.4 Dijagram razreda - dio Entiteti	34
4.5 Dijagram razreda - dio Servisi	34
4.6 Dijagram stanja - Prijavljeni korisnik	35
4.7 Dijagram aktivnosti	36
5.1 Testiranje dohvaćanja romobila preko identifikatora	40
5.2 Testiranje dohvaćanja poruka za zadene korisnike	40
5.3 Testiranje ListingController-a	41
5.4 Testiranje dohvaćanja najmova za korisnika	41
5.5 Testiranje brisanja komentara i ocjena	42
5.6 Uspješno provođenje testova	42
5.7 Testiranje dodavanja romobila i oglasa za romobil	43
5.8 Funkcija za prijavu korisnika preko objekta tipa WebDriver	43
5.9 Testiranje unajmljivanja romobila	43
5.10 Testiranje razmjene poruka između korisnika	44
5.11 Izrada baze podataka	46
5.12 Dockerfile i application.properties	46

5.13 Kreiranje backenda na Renderu	47
5.14 Kreiranje frontenda na Renderu	47

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 20. listopada 2023.
- Prisustvovali: Anamarija Jakoubek, Dino Babić, Jan Grbac, Igor Šoštarko, Karem Korić, Leonarda Pribanić, Ivan Pavelić
- Teme sastanka: Upoznavanje, Osnove projekta
 - Prvo upoznavanje, razmjenjivanje kontaktnih podataka i izrada skupne grupe.
 - Biranje imena projekta, tehnologije koje će se koristiti, tko bi se klijim dijelom htio baviti te koliko ćemo se često sastajati.

2. sastanak

- Datum: 24. listopada 2023.
- Prisustvovali: Anamarija Jakoubek, Dino Babić, Jan Grbac, Igor Šoštarko, Karem Korić, Leonarda Pribanić, Ivan Pavelić
- Teme sastanka: Funkcionalni zahtjevi, Dijagrami, Arhitektura i Baza podataka
 - Navođenje i razrada funkcionalnih i ostalih zahtjeva, koje će mogućnosti imati korisnici, a koji administratori
 - Razrađeni dijagrami i napravljene prve skice opisa obrazaca uporabe temeljene na funkcionalnim zahtjevima, također napisani prvi sekvencijski dijagrami koji opisuju kako će aplikacija funkcionirati u danim slučajevima
 - Dogovor oko arhitekture klijent-poslužitelj koju ćemo koristiti kroz projekt. Za frontend je odabrana React, a za backend kombinacija Java, JavaScript i Spring Boot tehnologija. Prve skice baze podataka, tablica za entitete i postavljanje primarnih i stranih ključeva

3. sastanak

- Datum: 7. studenog 2023.
- Prisustvovali: Anamarija Jakoubek, Dino Babić, Jan Grbac, Igor Šoštarko,

Karemn Korić, Leonarda Pribanić, Ivan Pavelić

- Teme sastanka: Dijagrami razreda, Ažuriranje baze podataka

- Crtanje dijagrama razreda, specifikacija metoda koje povlače Controller razred, definiranje entiteta i servisa koje će entiteti koristiti
- Dorada i ispravak baze podataka, dodavanje novih atributa u entitete ili cijele koji zamjenjuju nepotrebne funkcionalnosti baze podataka

4. sastanak

- Datum: 15. studenog 2023.
- Prisustvovali: Anamarija Jakoubek, Dino Babić, Jan Grbac, Igor Šoštarko, Karemn Korić, Leonarda Pribanić, Ivan Pavelić
- Teme sastanka: Dovršavanje prvog dijela dokumentacije, Objava i testiranje web-aplikacije
 - Ažuriranje slika, dijelova teksta, tablica i ostalih stavki prije prve predaje
 - Deploy aplikacije na Redner i testiranje funkcionalnosti prijave, registracije, odobravanje registracija i osnovnih funkcionalnosti web-stranice

Tablica aktivnosti

	Ivan Pavelić	Jan Grbac	Anamarija Jakoubek	Karmen Korić	Dino Babić	Leonarda Pribanić	Igor Šoštarko
Upravljanje projektom	5	3			4		
Opis projektnog zadatka			2	5			
Funkcionalni zahtjevi			3	3			
Opis pojedinih obrazaca			1	4			
Dijagram obrazaca	4	4	4		4	4	4
Sekvencijski dijagrami	2	2	2		2	2	2
Opis ostalih zahtjeva	1	1	1		1	1	1
Arhitektura i dizajn sustava						3	4
Baza podataka	2	4	2	2	2	2	2
Dijagram razreda			2		4		
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Korištene tehnologije i alati							
Ispitivanje programskog rješenja							
Dijagram razmještaja							
Upute za puštanje u pogon							
Dnevnik sastajanja	1					1	
Zaključak i budući rad							

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Ivan Pavelić	Jan Grbac	Anamarija Jakoubek	Karmen Korić	Dino Babić	Leonarda Pribanić	Igor Šoštarko
Popis literature							
<i>Izrada početne stranice</i>			2	3			
<i>Frontend</i>			1	1	5		
<i>Izrada baze podataka</i>	4						5
<i>Spajanje s bazom podataka</i>						4	
<i>Backend</i>		5					

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.