Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

EasyRent

Tim: <TG16.2>

EasyRent

- Fran Galić
- Jakov Biloglav
- Karlo Brzak
- Ivan Dujmić
- Ivan Džanija
- Vilim Hrupelj
- Matija Kukić

Nastavnik: Vlado Sruk

Projekt: EasyRent – Web aplikacija za iznajmljivanje automobila

1. Potencijalna korist projekta EasyRent

EasyRent je inovativna platforma za iznajmljivanje automobila koja pojednostavljuje, ubrzava i čini najam vozila transparentnijim. Ovaj projekt donosi višestruke koristi kako za korisnike tako i za vlasnike vozila.

Prednosti za korisnike

- **Brza i efikasna rezervacija**
  Platforma omogućava korisnicima jednostavno rezerviranje vozila putem online sustava, eliminirajući potrebu za osobnim dolaskom ili složenim procedurama.
- **Fleksibilnost i personalizacija**
  Korisnici mogu pretraživati vozila prema različitim parametrima (tip vozila, cijena, lokacija), čime im je osigurana prilagodljivost i lakoća odabira.
- **Transparentnost i povjerenje**
  Sustav ocjenjivanja i recenzija pruža korisnicima uvid u kvalitetu usluge, što olakšava donošenje informiranih odluka.

Prednosti za vlasnike vozila

- **Automatizacija i smanjenje troškova**
  Automatizirani procesi za upravljanje rezervacijama i transakcijama omogućuju smanjenje administrativnih troškova te ubrzavaju obradu rezervacija, čime se smanjuje potreba za ljudskom intervencijom.
- **Bolja kontrola nad flotama**
  Platforma nudi vlasnicima vozila jednostavnije upravljanje vozilima i mogućnost bržeg odgovaranja na recenzije.

Financijska fleksibilnost

- **Sigurno i jednostavno plaćanje**
  Integrirani sustavi plaćanja omogućuju jednostavno plaćanje putem virtualnog novčanika, PayPal-a, Stripe-a i drugih servisa, čime se platforma prilagođava međunarodnim korisnicima.
- **Bonusi i popusti**
  Korisnici mogu iskoristiti bonuse i popuste za buduće najmove, što potiče njihovu lojalnost i dugoročno korištenje platforme.

2. Postojeća slična rješenja

Na tržištu postoje već etablirane platforme za iznajmljivanje vozila poput **Rentcars**, **Hertz**, **Enterprise**, i **Turo**, no **EasyRent** se izdvaja u nekoliko ključnih aspekata.

- **Interakcija korisnika i vlasnika vozila**
  EasyRent omogućava korisnicima izravnu komunikaciju s vlasnicima vozila putem ugrađenog chat sustava. Korisnici mogu pregovarati o uvjetima najma i cijeni, što omogućava dogovore o posebnim uvjetima, kao što su:
  - Popusti za dugoročne najmove
  - Fleksibilni uvjeti preuzimanja vozila izvan radnog vremena
- **Sustav virtualnog novčanika i nagrađivanja korisnika**
  EasyRent implementira sustav bonusa i popusta za buduće najmove, čime se potiče dugoročno korištenje platforme. Korisnici mogu iskoristiti dodatne poticaje, za razliku od većine konkurentskih platformi koje nude samo osnovne opcije plaćanja (kartice, PayPal).
- **Personalizirane preporuke**
  EasyRent koristi povratne informacije i povijest rezervacija za prilagodbu preporuka, poboljšavajući korisničko iskustvo i olakšavajući donošenje odluka.
- **Višestruki korisnički profili**
  EasyRent nudi različite korisničke uloge, uključujući:

  - **Korisnike (klijente)**
  - **Vlasnike vozila**
  - **Administratore**

  Administratori imaju pristup interakcijama između korisnika i vlasnika te mogu intervenirati u slučaju nesuglasica ili nepravilnosti, osiguravajući sigurnost i transparentnost na platformi.

3. Skup korisnika koji bi mogao biti zainteresiran za ostvareno rješenje

EasyRent cilja širok spektar korisnika, uključujući:

- **Turiste**
  Turisti mogu lako rezervirati vozila za putovanje, koristeći filtere prema cijeni, vrsti vozila i lokaciji kako bi pronašli najpovoljniju opciju za svoj boravak.
- **Poslovne korisnike**
  Poslovni korisnici, koji često trebaju vozila za službene potrebe, cijene fleksibilnost i brzinu online rezervacija, što štedi vrijeme i olakšava planiranje.
- **Vlasnike vozila**
  Privatni vlasnici i tvrtke mogu koristiti platformu za dodatni prihod, s mogućnošću postavljanja cijena i dostupnosti, kao i jednostavne komunikacije s korisnicima.
- **Startupove i mala poduzeća**
  Tvrtke bez vlastite flote mogu koristiti EasyRent za povoljan najam vozila, prilagođavajući najam svojim poslovnim potrebama.
- **Ekološki osviještene korisnike**
  Korisnici zainteresirani za ekološke opcije mogu preferirati najam električnih vozila na EasyRent-u, što doprinosi održivom poslovanju i smanjenju emisije ugljičnog dioksida.

4. Mogućnost prilagodbe rješenja

EasyRent nudi visoku razinu prilagodbe za različite potrebe i tržišta, uključujući:

- **Podršku za globalno tržište**
  Platforma može podržavati više jezika i valuta, čineći je pogodnom za različite regije i zemlje.
- **Integracija s vanjskim platformama**
  Uz opcije plaćanja kao što su PayPal i Stripe, moguće je dodati lokalne platne sustave i aplikacije za prijevoz kako bi se korisnicima omogućile dodatne opcije.

- **Prilagodba za specifične vrste vozila**
  Sustav se može proširiti za luksuzna vozila, električna vozila ili vozila za dostavu.
- **Mogućnosti upravljanja flotama**
  Za tvrtke s većim flotama, platforma omogućuje praćenje statusa vozila, optimizaciju cijena najma i analizu performansi.
- **Prilagodba korisničkog iskustva**
  Platforma može uključivati personalizirane popuste, obavijesti putem mobilnih aplikacija, te preporuke temeljene na AI tehnologiji.

5. Opseg projektnog zadatka

Opseg projektnog zadatka uključuje razvoj i implementaciju sljedećih ključnih funkcionalnosti:

- **Frontend sučelje**
  Razvoj frontend sučelja u **Next.js** i **React** omogućit će korisnicima pregledavanje vozila, filtriranje rezultata, pregled recenzija i jednostavno rezerviranje.
- **Integrirani chat sustav**
  Omogućit će izravnu komunikaciju između korisnika i vlasnika vozila.
- **Backend poslovna logika**
  Backend u **Django** (Python) omogućit će upravljanje korisnicima, vozilima, rezervacijama i plaćanjima.
- **Baza podataka**
  Implementacija baze podataka u **PostgreSQL** za pohranu podataka o korisnicima, vozilima, transakcijama i povijesti rezervacija.
- **Integracija s vanjskim servisima**
  Korištenje servisa kao što su **PayPal**, **Stripe**, **Google Maps**, **OAuth 2.0** te SMS i e-mail notifikacije za sigurno plaćanje i obavijesti.
- **Hosting i deployment**
  Aplikacija će biti postavljena na **Render**, **Verace** i **Neon** za jednostavnost, skalabilnost i pouzdanost.

Opseg uključuje razvoj minimalnog održivog proizvoda (MVP), uz mogućnost nadogradnje funkcionalnosti u kasnijim fazama.

6. Moguće nadogradnje projektnog zadatka

EasyRent platforma može se proširiti na sljedeće načine:

- **Mobilne aplikacije**
  Razvoj aplikacija za iOS i Android omogućit će korisnicima lakši pristup platformi.

- **Praćenje vozila u stvarnom vremenu**
  Integracija GPS sustava za praćenje statusa vozila tijekom najma.
- **Personalizirani sustav nagrađivanja**
  Ocjene korisnika i prepoznavanje preferencija mogu omogućiti personalizirane popuste i promocije.
- **Automatsko prilagođavanje cijena**
  AI sustav može automatski prilagođavati cijene na temelju tržišnih uvjeta i povijesti rezervacija.
- **Napredne analize i preporuke**
  Platforma može nuditi prijedloge za vozila prema korisničkim preferencijama, proširenu analitiku za vlasnike vozila te dodatne opcije za osiguranje i povezane usluge.

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Kriterij prihvaćanja
F-001	Sustav omogućuje registraciju i prijavu korisnika	Visoki	Sustav uspiješno registrira i prijavljuje korisnike
F-002	Korisnici mogu pregledavati, filtrirati i rezervirati vozila, te upravljati i pregladavati trenutne i prošle rezervacije.	Visoki	Korisnici mogu pregledavati vozila i upravljati rezervacijama
F-003	Korisnici mogu plaćati putem virtualnog novčanika ili vanjskih servisa	Visoki	Korisnici mogu izvršiti uplatu putem različitih platnih sustava
F-004	Sustav ocjenjivanja omogućuje korisnicima ocjenjivanje vozila i vlasnika te pregled recenzija	Srednji	Korisnici mogu ocijeniti vozila i vlasnike te pregledavati recenzije
F-005	Sustav preporuka na temelju ocjena i recenzija sugerira korisnicima vozila s najboljim ocjenama	Srednji	Sustav preporuka nudi vozila s najboljim ocjenama i recenzijama
F-006	Vlasnici mogu dodavati, uređivati i upravljati vozilima, te pratiti prihode i analizu recenzija	Visoki	Vlasnici mogu upravljati vozilima, pregledati prihode i recenzije
F-007	Vlasnici i korisnici mogu putem chat sustava pregovarati o uvjetima najma.	Srednji	Vlasnici i korisnici mogu komunicirati i dogovarati uvjete najma
F-008	Administrator može upravljati korisnicima (dodavanje, uklanjanje, pregled informacija)	Visoki	Administrator može pregledavati i upravljati korisnicima i vlasnicima

ID zahtjeva	Opis	Prioritet	Kriterij prihvaćanja
F-009	Administratori mogu pratiti aktivnosti, moderirati chat, recenzije i upravljati financijskim transakcijama	Visoki	Administratori mogu pratiti sve aktivnosti i intervenirati po potrebi
F-010	Administratori mogu kreirati promocije i postavljati istaknute ponude.	Srednji	Administratori mogu upravljati promocijama i korisnicima/vlasnicima
F-011	Administratori mogu rješavati pritužbe i sporove između korisnika i vlasnika vozila u svim navedenim sustavima.	Visoki	Administratori mogu rješavati sporove i donositi odluke o povratima
F-012	Sustav mora moći slati pripadajuće obavijesti na email adresu pripadajećeg korsnika	Srednji	Sustav uspiješno šalje mailove korisnicima sustava

Ostali zahtjevi

Nefunkcionalni zahtjevi

Zahtjevi performansi

ID zahtjeva	Opis	Prioritet
NF-1.1	Stranica treba učitati unutar 2 sekunde na prosječnoj mrežnoj vezi.	Visoki
NF-1.2	Sustav mora podržavati do 1000 istovremenih korisnika bez usporavanja.	Visoki

Zahtjevi sigurnosti

ID zahtjeva	Opis	Prioritet
NF-2.1	Svi osjetljivi podaci (lozinke, osobni podaci) moraju biti šifrirani koristeći moderne algoritme.	Visoki
NF-2.2	Sustav mora imati zaštitu od napada poput SQL injekcija i XSS.	Visoki

Zahtjevi za korisničko iskustvo

ID zahtjeva	Opis	Prioritet
NF-3.1	Sučelje mora biti intuitivno i jednostavno za navigaciju.	Visoki
NF-3.2	Responsivni dizajn koji omogućuje optimalno korištenje na mobilnim uređajima.	Visoki

Zahtjevi za održavanje

ID zahtjeva	Opis	Prioritet
NF-4.1	Kod mora biti dobro dokumentiran kako bi olakšao buduće izmjene i nadogradnje.	Visoki
NF-4.2	Sustav mora biti lako proširiv za dodavanje novih funkcionalnosti.	Visoki

Zahtjevi za oporavak

ID zahtjeva	Opis	Prioritet
NF-5.1	Redovite sigurnosne kopije podataka moraju se provoditi svakodnevno.	Visoki
NF-5.2	Sustav mora omogućiti brz oporavak u slučaju kvara.	Visoki

Domenski zahtjevi

ID zahtjeva	Opis	Prioritet
DR-6.1	Sustav mora provoditi automatske provjere registracije vozila prema važećim zakonima i propisima države u kojoj se vozilo iznajmljuje, uključujući provjeru važeće prometne dozvole i osiguranje.	Visoki
DR-6.2	Platforma mora osigurati prikupljanje i pohranu osobnih podataka korisnika u skladu s GDPR-om (General Data Protection Regulation) ili sličnim zakonima o zaštiti privatnosti podataka.	Visoki
DR-6.3	Platforma mora biti usklađena sa zakonima o elektroničkom plaćanju, uključujući osiguranje zaštićenih plaćanja putem vanjskih servisa poput PayPala, u skladu s pravilima zaštite potrošača.	Visoki

ID zahtjeva	Opis	Prioritet
DR-6.4	Sustav mora automatski pratiti i evidentirati sve transakcije, uključujući plaćanja, povrate i troškove povezane s najmom, u skladu sa zakonima o zaštiti potrošača i računovodstvenim standardima.	Visoki

---

Dionici

1. **Korisnici sustava** – krajnji korisnici koji koriste sustav za iznajmljivanje vozila.
2. **Automobilske kuće (davatelji vozila)** – tvrtke koje stavljaju vozila na raspolaganje za najam, dodaju informacije o vozilima i upravljaju dostupnošću.
3. **Administratori sustava** – osobe koje nadgledaju rad sustava, moderiraju sadržaj i korisničke račune te osiguravaju da sve funkcionalnosti rade ispravno.
4. **Razvojni tim** – tim odgovoran za održavanje, ažuriranje i razvoj sustava, implementaciju novih funkcionalnosti i ispravak grešaka.

Aktori i njihovi funkcionalni zahtjevi:

1. **Korisnik (iznajmljivač, neregistrirani)** (inicijator) može:
   - Napraviti novi korisnički račun
   - Pregledati dostupna vozila
   - Filtrirati vozila prema raznim kriterijima (npr. cijena, tip vozila, lokacija)
   - Pogledati detaljne informacije o vozilu
2. **Korisnik (iznajmljivač, registrirani)** (inicijator) može:
   - Prijaviti se i registrirati na sustav
   - Pregledati dostupna vozila
   - Filtrirati vozila prema raznim kriterijima (npr. cijena, tip vozila, lokacija)
   - Pogledati detaljne informacije o vozilu
   - Izabrati vozilo za najam i započeti proces rezervacije > - unijeti podatke o vremenu najma (datum preuzimanja i povratka) > - potvrditi rezervaciju i izvršiti uplatu
   - Pregledati i upravljati svojim rezervacijama
   - Ocijeniti i recenzirati iskustvo nakon završenog najma
   - Pregledati svoje ocjene i povijest iznajmljivanja
   - Komunicirati sa vlasnicima vozila i pregovarati uvjete najma
   - Platiti uz pomoć digitalnog novčanika integriranog u stranici
3. **Automobilska kuća (davatelj vozila)** (inicijator) može:

- Prijaviti se i registrirati na sustav kao poslovni korisnik
- Dodavati nova vozila u sustav s detaljnim informacijama (npr. marka, model, godine proizvodnje, cijena najma)
- Upravljati postojećim vozilima u sustavu > - ažurirati podatke o vozilu > - promijeniti dostupnost vozila
- Primati i pregledavati rezervacije korisnika
- Upravljati rezervacijama (potvrditi, odbiti ili promijeniti rezervaciju)
- Pregledati povijest najma svojih vozila
- Primati povratne informacije i ocjene korisnika
- Uvid u statistike vozila i transakcija
- Dogovarati putem chat-a uvjete najma sa korisnicima

4. **Administrator sustava** (sudionik) može:
   - Prijaviti se u administracijski panel sustava
   - Upravljati korisničkim računima > - odobriti, deaktivirati ili suspendirati korisničke račune
   - Moderirati sadržaj dodan od strane korisnika i automobilskih kuća
   - Pregledati i upravljati novim oglasima vozila
   - Pratiti i analizirati statistike sustava
   - Primati i odgovarati na prijave problema ili pritužbe korisnika
   - Upravljati sigurnosnim postavkama i pravnim obavijestima sustava
   - Razrješiti poteškoće pri transakcijama i akcijama sustava.

Obrasci uporabe

Kategorija	Use Caseovi
1. Visokorazinski dijagram obrazaca uporabe cijelog sustava	UC1: Registracija korisnika UC2: Prijava korisnika UC6: Sustav preporuka UC7.1: Dodavanje i Uređivanje Vozila od strane Vlasnika UC8: Komunikacija putem chat sustava UC13: Slanje obavijesti e-mailom
2. Dijagram obrazaca uporabe za ključne funkcionalnosti	UC3.1: Pregled vozila UC3.2: Rezervacija vozila UC4: Plaćanje putem virtualnog novčanika ili vanjskih servisa UC5: Ocjenjivanje vozila i vlasnika UC10: Praćenje aktivnosti od strane administratora
3. Dijagram obrazaca uporabe za korisničke role	UC3.1: Pregled vozila UC3.2: Rezervacija vozila UC5: Ocjenjivanje vozila i vlasnika UC6: Sustav preporuka UC8: Komunikacija putem chat sustava

Kategorija	Use Caseovi
4. Dijagram obrazaca uporabe za osnovne poslovne procese	UC7.1: Dodavanje i Uređivanje Vozila od strane Vlasnika UC7.2: Praćenje Prihoda i Recenzija Vozila od strane Vlasnika UC9: Upravljanje korisnicima od strane administratora UC11: Kreiranje promocija UC12: Rješavanje pritužbi i sporova
5. Dijagram obrazaca uporabe za kritične sustave i integracije	UC4: Plaćanje putem virtualnog novčanika ili vanjskih servisa UC9: Izdavanje potvrda o upisu UC12: Rješavanje pritužbi i sporova

Opis obrazaca uporabe

UC1: Registracija korisnika

- **Glavni sudionik**: Korisnik
- **Cilj**: Korisnik se registrira u sustav kako bi mogao koristiti funkcionalnosti sustava.
- **Sudionici**: Korisnik, Sustav
- **Preduvjet**: Korisnik nije prethodno registriran u sustavu.
- **Opis osnovnog tijeka**:
  1. Korisnik otvara početnu stranicu sustava.
  2. Korisnik odabire opciju “Registracija”(F-001).
  3. Sustav prikazuje obrazac za unos podataka (ime, prezime, e-mail, lozinka, potvrda lozinke).
  4. Korisnik unosi potrebne podatke i potvrđuje unos.
  5. Sustav provodi validaciju podataka (provjera ispravnosti e-maila, jačine lozinke, itd.).
  6. Sustav šalje e-mail korisniku za potvrdu registracije.
  7. Korisnik otvara e-mail i klikne na poveznicu za potvrdu registracije.
  8. Sustav potvrđuje registraciju i omogućava korisniku pristup sustavu.
- **Moguća odstupanja**:
  - Ako e-mail već postoji u sustavu, sustav obavještava korisnika da je e-mail već registriran i nudi opciju za oporavak lozinke.
  - Ako korisnik ne potvrdi e-mail unutar određenog vremena, sustav poništava registraciju.

UC2: Prijava korisnika

- **Glavni sudionik**: Korisnik
- **Cilj**: Korisnik se prijavljuje na svoj račun kako bi pristupio funkcionalnostima sustava.
- **Sudionici**: Korisnik, Sustav
- **Preduvjet**: Korisnik je već registriran u sustavu.

- **Opis osnovnog tijeka**:
  1. Korisnik otvara početnu stranicu sustava.
  2. Korisnik odabire opciju "Prijava"(F-001).
  3. Sustav prikazuje obrazac za prijavu (e-mail, lozinka).
  4. Korisnik unosi svoje korisničke podatke (e-mail i lozinka) i potvrđuje prijavu.
  5. Sustav provodi validaciju unesenih podataka (provjera ispravnosti lozinke i e-maila).
  6. Ako su podaci ispravni, sustav omogućuje korisniku pristup svom korisničkom sučelju.
  7. Ako korisnik zaboravi lozinku, sustav nudi opciju za oporavak lozinke putem e-maila.
- **Moguća odstupanja**:
  - Ako korisnik unese pogrešne podatke (neispravan e-mail ili lozinku), sustav obavještava korisnika i omogućava mu da ponovo unese podatke.
  - Ako korisnik zaboravi lozinku, sustav nudi opciju za oporavak lozinke putem e-maila.

UC3.1: Pregled vozila

- **Glavni sudionik**: Korisnik
- **Cilj**: Pregledati dostupna vozila prema različitim filtrima.
- **Sudionici**: Korisnik, Sustav
- **Opis osnovnog tijeka**:
  1. Korisnik otvara katalog vozila(F-002).
  2. Sustav prikazuje vozila s opcijama filtriranja (lokacija, cijena, vrsta vozila).
  3. Korisnik koristi dostupne filtre za prilagodbu pregleda vozila.
  4. Korisnik odabire željeno vozilo i pregledava detalje (opis, slika, cijena, dostupnost).
- **Moguća odstupanja**:
  - Ako nema vozila koja zadovoljavaju korisnikove filtere, sustav prikazuje obavijest o nedostatnoj ponudi.
  - Ako korisnik odustane od pregleda, sustav vraća korisnika na početnu stranicu ili prethodni ekran.

UC3.2: Rezervacija vozila

- **Glavni sudionik**: Korisnik
- **Cilj**: Rezervirati odabrano vozilo.
- **Sudionici**: Korisnik, Sustav
- **Preduvjet**: Korisnik je prijavljen.

- **Opis osnovnog tijeka**:
  1. Korisnik odabire opciju "Rezerviraj" na stranici vozila(F-002).
  2. Sustav prikazuje potvrdu o dostupnosti vozila za odabrane datume.
  3. Korisnik potvrđuje rezervaciju.
  4. Sustav potvrđuje rezervaciju i prikazuje detalje rezervacije (datum, cijena, uvjeti).
- **Moguća odstupanja**:
  - Ako vozilo nije dostupno na odabrane datume, sustav nudi alternativne datume ili vozila.
  - Ako korisnik prekine postupak rezervacije, status rezervacije se ne mijenja i korisnik se vraća na prethodni ekran.

UC4: Plaćanje putem virtualnog novčanika ili vanjskih servisa

- **Glavni sudionik**: Korisnik
- **Cilj**: Izvršiti plaćanje za rezervaciju.
- **Sudionici**: Korisnik, Sustav, Platni servis
- **Preduvjet**: Korisnik je odabrao vozilo za rezervaciju.
- **Opis osnovnog tijeka**:
  1. Korisnik odabire opciju plaćanja(F-003).
  2. Sustav prikazuje mogućnosti (virtualni novčanik, kartica, PayPal).
  3. Korisnik odabire metodu plaćanja i unosi podatke.
  4. Sustav šalje zahtjev platnom servisu.
  5. Platni servis potvrđuje transakciju.
  6. Sustav prikazuje potvrdu o uspješnom plaćanju.

Moguća odstupanja: - Ako transakcija ne uspije, korisnik prima obavijest o pogrešci. - Ako korisnik odustane, rezervacija se ne potvrđuje.

UC5: Ocjenjivanje vozila i vlasnika

- **Glavni sudionik**: Korisnik
- **Cilj**: Ocijeniti vozilo i vlasnika te pregledati recenzije.
- **Sudionici**: Korisnik, Sustav
- **Preduvjet**: Korisnik je završio najam.
- **Opis osnovnog tijeka**:
  1. Korisnik otvara stranicu prethodne rezervacije(F-002).
  2. Sustav prikazuje opciju ocjenjivanja(F-004).
  3. Korisnik unosi ocjenu i komentar.
  4. Sustav pohranjuje ocjenu i ažurira prosječnu ocjenu.

Moguća odstupanja: - Ako korisnik unese neprimjeren komentar, sustav šalje obavijest administratoru. - Ako korisnik ne završi ocjenjivanje, sustav ne pohranjuje podatke.

UC6: Sustav preporuka

- **Glavni sudionik**: Korisnik
- **Cilj**: Prikazati vozila s najboljim ocjenama i recenzijama.
- **Sudionici**: Korisnik, Sustav
- **Preduvjet**: Korisnik je prijavljen.
- **Opis osnovnog tijeka**:
  1. Korisnik otvara katalog vozila(F-002).
  2. Sustav prikazuje preporučena vozila na temelju ocjena i recenzija(F-005).
  3. Korisnik pregledava detalje preporučenih vozila.
  4. Sustav omogućuje rezervaciju izravno iz preporučenog popisa.

Moguća odstupanja: - Ako nema dovoljno podataka za preporuke, sustav prikazuje najnovija vozila. - Ako korisnik filtrira rezultate, preporuke se ažuriraju.

UC7.1: Dodavanje i uređivanje vozila od strane vlasnika

- **Glavni sudionik**: Vlasnik
- **Cilj**: Dodavati nova vozila ili uređivati postojeća vozila u sustavu.
- **Sudionici**: Vlasnik, Sustav
- **Preduvjet**: Vlasnik je prijavljen u sustav.
- **Opis osnovnog tijeka**:
  1. Vlasnik otvara stranicu za upravljanje vozilima.
  2. Sustav prikazuje popis postojećih vozila.
  3. Vlasnik odabire opciju za dodavanje novog vozila ili uređivanje postojećeg(F-006).
  4. Vlasnik unosi ili ažurira podatke o vozilu (npr. marka, model, cijena, dostupnost, opis vozila).
  5. Sustav pohranjuje promjene i ažurira bazu podataka.
- **Moguća odstupanja**:
  - Ako vlasnik unese neispravne podatke, sustav prikazuje obavijest o grešci.
  - Ako vozilo već ima aktivne rezervacije, uređivanje nije moguće dok se one ne završe.

UC7.2: Praćenje prihoda i recenzija vozila od strane vlasnika

- **Glavni sudionik**: Vlasnik

- **Cilj**: Praćenje prihoda generiranih od vozila, te pregled recenzija korisnika.
- **Sudionici**: Vlasnik, Sustav
- **Preduvjet**: Vlasnik je prijavljen u sustav.
- **Opis osnovnog tijeka**:
  1. Vlasnik otvara stranicu za analitiku prihoda i recenzija(F-006).
  2. Sustav prikazuje statistiku prihoda za vozila, uključujući ukupnu zaradu, broj rezervacija i prihod po vozilu.
  3. Vlasnik pregledava recenzije korisnika za svako vozilo, uključujući ocjene i komentare.
  4. Sustav generira izvještaje o prihodima i recenzijama za određeno vremensko razdoblje (npr. dnevno, tjedno, mjesečno).
- **Moguća odstupanja**:
  - Ako vozilo nije ostvarilo nikakav prihod u određenom vremenskom razdoblju, sustav prikazuje obavijest da nema prihoda za to vozilo.
  - Ako vozilo nema recenzije, sustav prikazuje obavijest da nisu ostavljene recenzije.

UC8: Komunikacija putem chat sustava

- **Glavni sudionik**: Korisnik i Vlasnik
- **Cilj**: Pregovarati o uvjetima najma.
- **Sudionici**: Korisnik, Vlasnik, Sustav
- **Preduvjet**: Korisnik ili vlasnik je prijavljen i odabiru započeti chat sa drugom partijom.
- **Opis osnovnog tijeka**:
  1. Korisnik ili vlasnika odabire opciju za pokretanje razgovora(F-007).
  2. Sustav otvara chat sučelje.
  3. Korisnik ili vlasnika unosi poruku i šalje je.
  4. Primatelj prima poruku i odgovara.
  5. Chat zapis se čuva u sustavu.

Moguća odstupanja: - Ako korisnik napusti chat prije slanja poruke, poruka se ne pohranjuje.

UC9: Upravljanje korisnicima od strane administratora

- **Glavni sudionik**: Administrator
- **Cilj**: Upravljanje korisnicima, uključujući dodavanje, uklanjanje i pregled podataka.
- **Sudionici**: Administrator, Sustav
- **Preduvjet**: Administrator je prijavljen.

- **Opis osnovnog tijeka**:
  1. Administrator otvara sučelje za upravljanje korisnicima(F-008).
  2. Sustav prikazuje popis korisnika s osnovnim podacima.
  3. Administrator odabire opciju dodavanja, uređivanja ili brisanja korisnika.
  4. Administrator unosi potrebne izmjene.
  5. Sustav pohranjuje promjene i ažurira bazu podataka.

Moguća odstupanja: - Ako administrator unese neispravne podatke, sustav prikazuje obavijest o pogrešci. - Ako pokušava obrisati korisnika s aktivnim rezervacijama, sustav prikazuje upozorenje.

UC10: Praćenje aktivnosti od strane administratora

- **Glavni sudionik**: Administrator
- **Cilj**: Pratiti aktivnosti, moderirati chat i upravljati financijskim transakcijama.
- **Sudionici**: Administrator, Sustav
- **Preduvjet**: Administrator je prijavljen u sustav.
- **Opis osnovnog tijeka**:
  1. Administrator otvara sučelje za praćenje aktivnosti(F-009).
  2. Sustav prikazuje sve aktivne chatove, recenzije i transakcije.
  3. Administrator odabire aktivnost koju želi pregledati.
  4. Administrator poduzima odgovarajuće radnje (npr. blokira korisnika, poništava transakciju).

Moguća odstupanja: - Ako je chat već zatvoren, nije moguća intervencija.

UC11: Kreiranje promocija

- **Glavni sudionik**: Administrator
- **Cilj**: Kreirati promocije
- **Sudionici**: Administrator, Sustav
- **Preduvjet**: Administrator je prijavljen u sustav.
- **Opis osnovnog tijeka**:
  - Administrator odabire opciju za kreiranje promocije(F-010).
  - Unosi podatke o promociji (npr. popusti, trajanje).

Moguća odstupanja: - Ako podaci za promociju nisu potpuni, sustav prikazuje upozorenje. -Ako korisnik ima aktivne rezervacije, nije moguće promijeniti cijenu.

UC12: Rješavanje pritužbi i sporova

- **Glavni sudionik**: Administrator

- **Cilj**: Rješavati pritužbe i sporove između korisnika i vlasnika.
- **Sudionici**: Administrator, Korisnik, Vlasnik, Sustav
- **Preduvjet**: Administrator je prijavljen u sustav.
- **Opis osnovnog tijeka**:
  1. Administrator otvara modul za pritužbe(F-011).
  2. Sustav prikazuje popis prijavljenih sporova.
  3. Administrator odabire spor i pregledava detalje.
  4. Donosi odluku (prihvaćanje, odbijanje ili djelomični povrat).
  5. Sustav obavještava obje strane o ishodu(F-012).

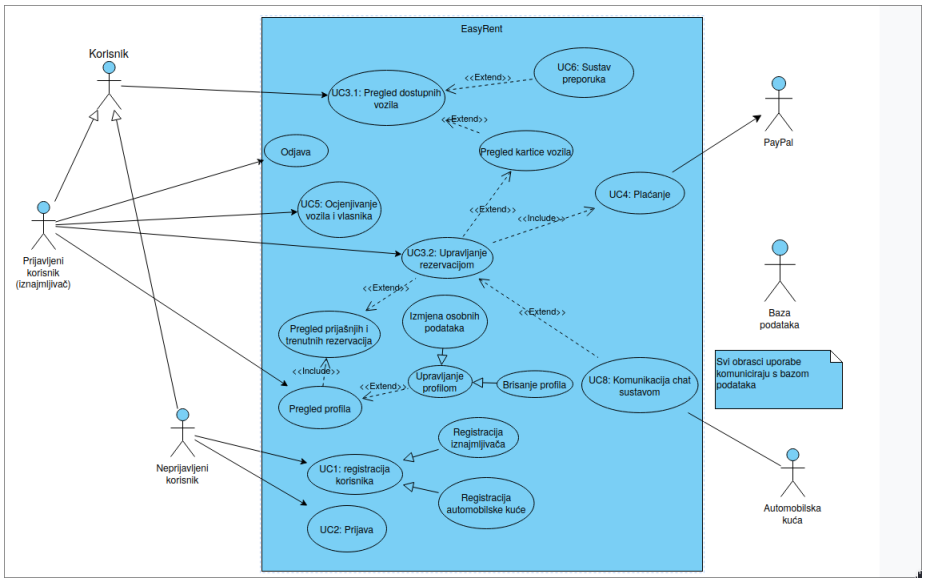
Moguća odstupanja: - Ako nema dovoljno dokaza za spor, administrator traži dodatne informacije. - Ako sustav ne može povezati pritužbu s transakcijom, prikazuje upozorenje.

UC13: Slanje obavijesti e-mailom

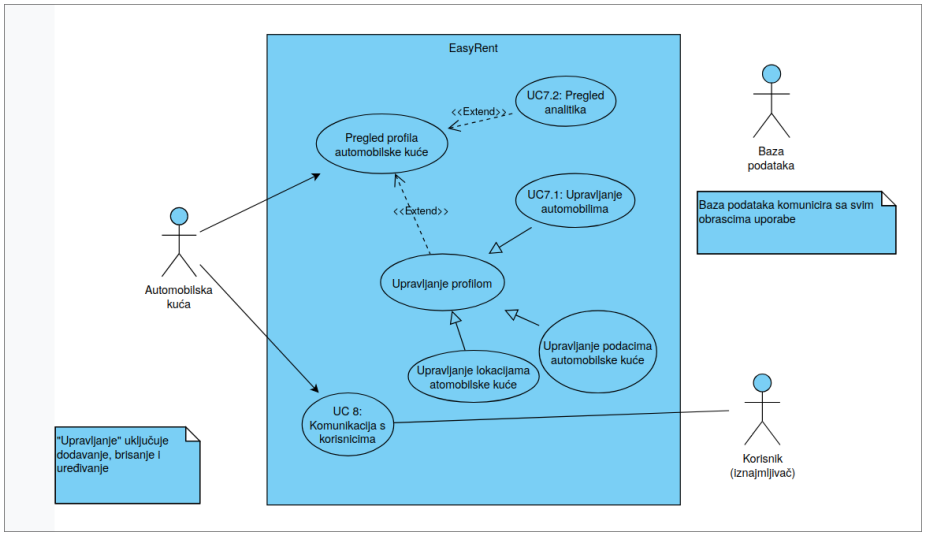
- **Glavni sudionik**: Sustav
- **Cilj**: Obavijestiti korisnike o važnim događajima.
- **Sudionici**: Sustav, Korisnik
- **Preduvjet**: Korisnik je uspiješno registriran.
- **Opis osnovnog tijeka**:
  1. Sustav detektira važan događaj (npr. potvrda rezervacije, promjena statusa).
  2. Sustav generira e-mail s potrebnim informacijama.
  3. E-mail se šalje korisniku(F-012).
  4. Korisnik prima obavijest i poduzima potrebne radnje.

Moguća odstupanja: - Ako e-mail nije dostavljen, sustav ponovno pokušava slanje. - Ako korisnik otkaže obavijesti, sustav ih više ne šalje.

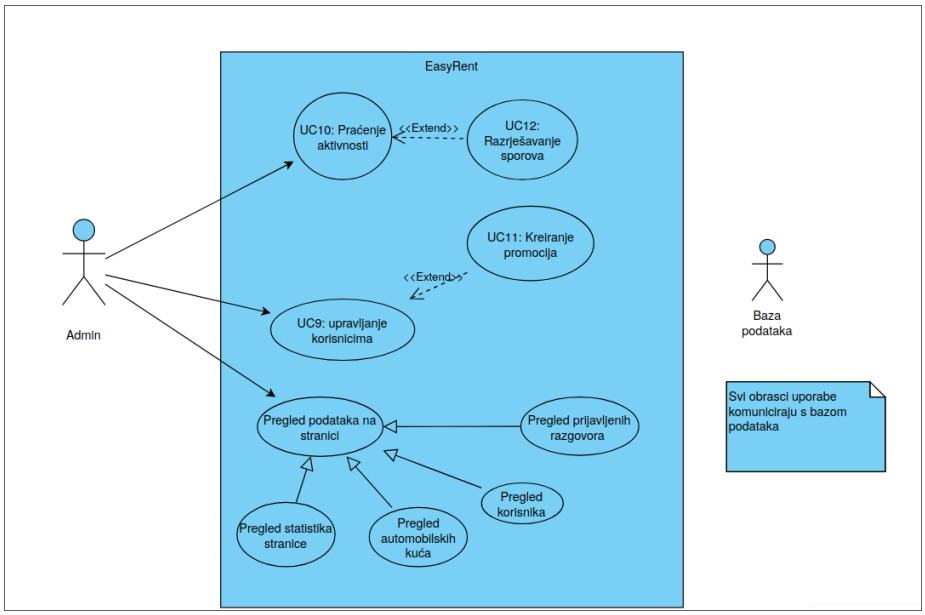
Dijagrami obrazaca uporabe



Slika 3.1 Funkcionalnosti iznajmljivača i neprijavljenog korisnika



Slika 3.2 Funkcionalnosti automobilske kuće

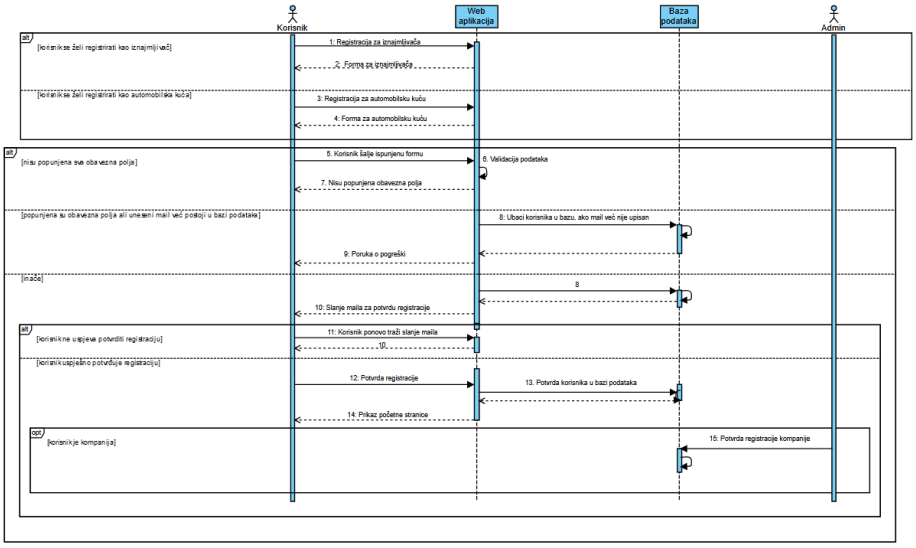


Slika 3.3 Funkcionalnosti administratora

Sekvencijski dijagrami

Sekvencijski dijagram za registraciju korisnika (UC1 i UC2)

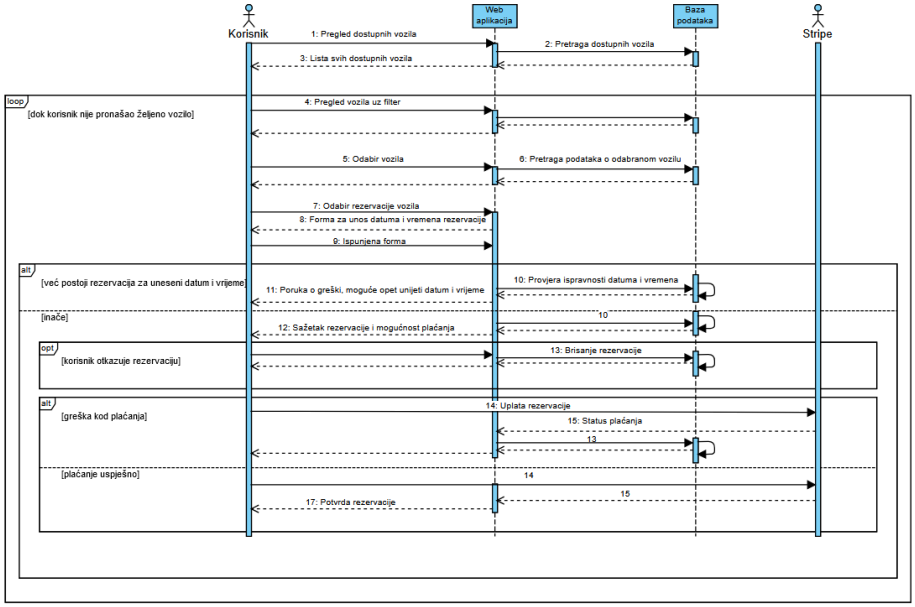
Korisnik šalje zahtjev za registraciju kao iznajmljivač ili automobilska kuća. Ovisno o ulozi korisnika aplikacija šalje odgovarajući obrazac. Obrazac za iznajmljivača sadrži polja za ime, prezime, mail, broj telefona, broj vozačke dozvole te polja za lozinku i potvrdu lozinke. Obrazac za automobilske kuće je nešto složeniji te sadrži polja za ime i mail kompanije, kontaktni broj telefona, lokaciju kompanije, radno vrijeme po danima te polja za lozinku i potvrdu lozinke. Nakon što korisnik ispuni obrazac, aplikacija ga validira te, ako neka obavezna polja nisu popunjena ili se polja za lozinke ne podudaraju, obavještava korisnika o problemu. Korisnik može ispraviti obrazac i poslati ga ponovo. Ako obrazac ne sadrži navedene pogreške, aplikacija provjerava postoji li već korisnik s istom e-mail adresom u bazi podataka. Ako postoji aplikacija obavještava korisnika o problemu te ga on može ispraviti i poslati novi obrazac. Ako se mail ne nalazi u bazi podataka, aplikacija na unesenu e-mail adresu šalje poveznicu za potvrdu registracije. Ako korisnik ne potvrdi registraciju unutar zadanog vremena, moguće je zatražiti ponovno slanje poveznice. Ako korisnik potvrdi e-mail, njegov se profil unosi u bazu podataka te se korisnika preusmjerava na početnu stranicu aplikacije (početnu stranicu iznajmljivača ili auto kuće, ovisno o ulozi korisnika).



Slika 3.4 Sekvencijski dijagram za registraciju (UC1 i UC2)

Sekvencijski dijagram za rezervaciju i pregled dostupnih vozila(UC3.1 i UC3.2)

Kada korisnik pristupa stranici za pregled automobila, aplikacija iz baze podataka dohvaća listu najpopularnijih vozila. Korisnik ima mogućnost filtriranja vozila po raznim parametrima (cijena, broj sjedala, model, ocjena, tip mjenjača...). Korisnik može odabrati određeno vozilo, nakon čega mu web aplikacija prikazuje karticu vozila s dodatnim parametrima, recenzijama. Ukoliko korisnik želi rezervirati automobil, aplikacija mu šalje obrazac za rezervaciju koji sadrži polja za datum, vrijeme i lokaciju polazišta i odredišta. Nakon slanja ispunjene forme aplikacija pomoću baze podataka provjerava je li automobil dostupan za navedeno vrijeme i mjesto te ako nije vraća poruku o pogreški. Ako je forma valjana rezervacija se upisuje u bazu podataka te se korisniku šalje sažetak rezervacije i mogućnost plaćanja pomoću Stripe-a. Korisnik prije plaćanja ima mogućnost otkazivanja narudžbe čime se ona briše iz baze podataka. Ukoliko korisnik želi platit, preusmjerava ga se na stranicu Stripe-a. Stripe web aplikaciji javlja status plaćanja. Ako plaćanje nije uspjelo rezervacija se briše iz baze podataka, inače se korisniku šalje potvrda rezervacije.



Slika 3.5 Sekvencijski dijagram za rezervaciju i pregled dostupnih vozila (UC3.1 i UC3.2)

Provjera uključenosti funkcionalnosti u obrasce uporabe

Funkcionalni zahtjev	Obrazac uporabe
F-001	UC-1, UC-2
F-002	UC-3.1, UC-3.2
F-003	UC-4
F-004	UC-5
F-005	UC-6
F-006	UC-7.1, UC-7.2
F-007	UC-8
F-008	UC-9
F-009	UC-10
F-010	UC-11
F-011	UC-12
F-012	UC-13

Arhitektura sustava

Opis arhitekture

Stil arhitekture

EasyRent koristi klijent-poslužitelj arhitektonski stil s odvojenim backend i frontend komponentama koje komuniciraju korištenjem REST API-ja. Frontend (React) ostvaruje korisničko sučelje koje omogućuje interakciju korisnika s backendom, dok backend (Django) provodi radnu logiku, komunicira s bazom podataka i šalje odgovore frontendu. Ovaj stil je odabran zbog jednostavnosti održavanja i razvoja, modularnosti i skalabilnosti *(literatura 12. točka)*, što ga čini idealnim izborom za mali tim s ograničenim vremenskim okvirima. Tim je razdvojen na dva podtima koji mogu raditi nezavisno prema dogovorenim specifikacijama te se pri kraju dvije komponente mogu spojiti u funkcionalnu cjelinu. Promjene se mogu zasebno provoditi na frontend i backend komponenti. Opseg aplikacije je manji te nema potrebe za finiju granulaciju komponenata.

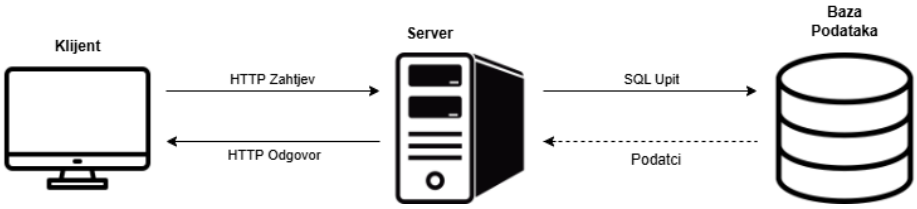


Figure 1: Dijagram Sustava drawio

Slika 4.1 Dijagram sustava

Podsustavi

- **Korisnički podsustav (Frontend)** - Podsustav odgovoran za klijentsku stranu i korisničko sučelje. Prikazuje podatke korisnicima i omogućava im interakciju s aplikacijom. U cilju mu je ostvariti intuitivno i responzivno sučelje za jednostavno i potpuno korištenje. Razvijen s pomoću React radnog okvira.
- **Podsustav za upravljanje podacima, zahtjevima i logikom (Backend)** - Obrađuje zahtjeve dobivene od klijenta, upravlja poslovnom logikom i pristupa bazi podataka. Na njemu se odvijaju autentikacija i autorizacija korisnika putem OAuth 2.0, upravljanje rezervacijama automobila i uređivanje profila. Mora pokriti sve sigurnosne provjere neovisno obavljale se one na drugim podsustavima ili ne. Razvijen s pomoću Django radnog okvira, koji pruža infrastrukturu za izgradnju sigurnih i skalabilnih aplikacija.
- **Podsustav za pohranu podataka (Baza podataka)** - U bazi podataka su pohranjeni svi podaci aplikacije. Nužno je ostvariti sigurnost i integritet podataka. Koristi se PostgreSQL.

- **Google Maps API (dio frontenda)** - Služi kompanijama za odabir lokacija poslovanja i za prikaz lokacija kompanija i pojedinih vozila korisnicima.
- **OAuth 2.0 (dio backenda)** - Omogućuje korisnicima sigurnu prijavu preko vanjskog identifikacijskog sustava (npr. Google) te po potrebi osigurava pristup resursima aplikacije prema njihovim privilegijama.
- **Stripe** - Služi za izvršavanje plaćanja i bilježenje transakcija. Koristi se u testnom načina rada za koji se može koristit broj kartice 4242 4242 4242 4242.
- **SwaggerUI** - Pomoćni sustav za olakšavanje testiranja i implementiranja krajnjih točaka backenda.

Preslikavanje na radnu platformu

- **Render (Backend)** - Render nudi vrlo jednostavno hostanje backenda i baze podataka, uključujući automatsko skaliranje, sigurnosne kopije i visoku dostupnost. Podržava Django i PostgreSQL uz mnoge druge programske jezike i baze podataka.
- **Vercel (Frontend)** - Vercel je optimiziran za frontend aplikacije te nudi besplatan plan za hosting statičkih aplikacija i automatsko skaliranje resursa.
- **Neon (Baza podataka)** - Vrlo jednostavan host PostgreSQL baze podataka bez poslužitelja. Besplatna licensa dopušta brzinu i memoriju puno veću nego za potrebe ovoga projekta.

Spremišta podataka

Relacijska baza podataka PostgreSQL je odabrana zbog svoje skalabilnosti i podrške za složene upite, što je ključno za upravljanje velikim brojem entiteta te omogućavanje naprednih funkcionalnosti poput pretrage i filtriranja. PostgreSQL nudi podršku za transakcije, što omogućuje sigurnost i integritet podataka. Baza će spremati podatke o korisnicima, kompanijama, lokacijama, vozilima, stanjima vozila, iznajmljivanjima, recenzijama, razgovorima i ostalo. Django i PostgreSQL su vrlo kompatibilni.

Mrežni protokoli

Primarni mrežni protokol koji se koristi za komunikaciju između frontenda i backenda je HTTP. HTTP omogućava jednostavnu razmjenu podataka između klijenta i poslužitelja putem REST API-ja. Frontend šalje zahtjeve prema backendu koji ih obrađuje i vraća odgovarajuće odgovore.

Globalni upravljački tok

Kada korisnik izvrši zahtjev na klijentskoj strani, ta radnja šalje HTTP zahtjev prema backendu. Backend prihvaća zahtjev te ga obrađuje, a po potrebi povlači podatke iz baze podataka ili ih pohranjuje u bazu. Zatim backend vraća odgovor

frontendu u JSON formatu. Frontend, ovisno o odgovoru, može prikazati neke podatke, preusmjeriti korisnika, prikazati grešku itd.

Sklopovskoprogramski zahtjevi

Poslužiteljska strana nije velike zahtjevnosti sa strane sklopovlja te ne predstavlja problem pokretanju na odabranom servisu za hosting. Korisnička strana je optimizirana za moderne preglednike te ne ovisi o korištenom operacijskom sustavu. Prikaz je prilagođen za različite vrste zaslona. Preporučuje se uređaj s najmanje 4 GB RAM-a, dvojezgrenim procesorom i osnovnom grafičkom akceleracijom jer aplikacija sadrži velik broj slika. Preporučuje se brza i stabilna internetska veza.

Obrazloženje odabira arhitekture

Klijent-poslužitelj stil je odabran zbog jasne podjele odgovornosti, vremenskih okvira tima te malog opsega projekta. Većina članova tima je izrazila želju za radom na jednoj od strana, a monolitska arhitektura bi također zahtijevala puno više vremena za usvajanje svih potrebnih tehnologija. React za frontend je odabran zbog visoke prisutnosti u poslovnom svijetu čime je na raspolaganju više materijala te zbog bogatog ekosustava komponenti koji znatno pojednostavljuje izradu aplikacije. Django također dolazi s mnogim ugrađenim alatima i komponentama koje je lagano integrirati u naš sustav. Svi članovi tima već imaju dobro razumijevanje JavaScripta dok je Python poznat kao čitljiv jezik koji je jednostavan za naučiti i koristiti. Naša aplikacija nema visoke zahtjeve koji zahtijevaju korištenje specifičnih radnih okvira za veću uspješnost. Razmatrano je i korištenje Springa za backend, ali se pokazalo da više članova tima ima preferencu za korištenjem Djangoa.

Organizacija sustava na visokoj razini

EasyRent sustav koristi klijent-poslužitelj arhitekturu s relacijskom bazom podataka. Korisnici preko grafičkog sučelja mogu vršiti zahtjeve koji se šalju na backend koji potom njih obrađuje uz pomoć baze podataka, po potrebi ih sprema u bazu te na frontend šalje odgovarajući odgovor koji se prikazuje korisniku u prikladnom obliku.

Organizacija aplikacije

Backend

Backend je implementiran s pomoću Djangoa. Backend je odgovoran za obradu poslovne logike, autentifikaciju korisnika (uz pomoć OAuth 2.0 za vanjske identifikacijske sustave), upravljanje podacima i komunikaciju s bazom podataka. Backend se sastoji od više aplikacija od kojih svaka ostvaruje jednu cjelinu. Na primjer ‘src’ ostvaruje sve vezano za autentikaciju i autorizaciju dok ‘home’ ostvaruje funkcionalnosti početne stranice. Jezgrena aplikacija je ‘backend’ koja

sadrži 'settings.py' s generalnim postavkama sustava i 'urls.py' za povezivanje svih aplikacija. Svaka aplikacija sadrži nekoliko funkcionalnih datoteka: * **models.py** - Služi za definiranje modela prema osmišljenoj bazi koji se onda migriraju u bazu (Django generira bazu koristeći modele). * **admin.py** - Služi za registriranje modela s administracijskim sučeljem djangoa koje dopušta pregled i uređivanje podataka u bazi podataka. * **views.py** - Sadrži logiku za rukovanje HTTP zahtjevima i odgovorima. * **serializers.py** - Služi za formatiranje oblika tijela zahtjeva i odgovora. Za ovaj projekt su se koristili za formatiranje SwaggerUI primjera. * **urls.py** - Mapira views.py metode s putanjama koje frontend može koristiti za slanje zahtjeva. * **Ostale** - Po potrebi se mogu definirati datoteke za dodatne potrebe poput tests.py, serializers.py, forms.py, singals.py, tokens.py, itd.

Frontend

Frontend je implementiran s pomoću Reacta i Next.js-a, uz korištenje Chakra UI-a za stiliziranje i izradu komponenata korisničkog sučelja. Frontend je odgovoran za prikaz podataka korisniku, autentifikaciju korisnika, zaštitu ruta te upravljanje navigacijom unutar aplikacije. Struktura frontenda: * **Mapa app** - app/auth: Sadrži login i registracijske stranice, omogućavajući korisnicima kreiranje računa i prijavu. app/(auth-user) i app/(auth-company): Ove mape služe za stranice specifične za ulogu korisnika ili kompanije. Na primjer, unutar mape user i company nalaze se page.tsx datoteke koje predstavljaju različite korisničke profile. app/home: Glavna stranica aplikacije, dostupna gostima i korisnicima, ali ne i firmama koje nemaju opciju iznajmljivanja vozila već ih se preusmjerava na njihov profil. * **Mapa components** - Komponente su podijeljene na nekoliko mapa, uključujući core, features, i shared. shared/auth: Sadrži komponente za autentifikaciju, kao što su AuthUserHeader. shared/Header i Footer: Komponente za zaglavlje i podnožje koje su vidljive na različitim stranicama aplikacije. features/cars i company: Sadrže specifične funkcionalnosti za prikaz automobila i kompanija. * **Mapa fetchers** - Sadrži fetcher.ts, funkciju za dohvaćanje podataka s backenda putem HTTP zahtjeva. Fetcher koristi credentials: 'include' kako bi omogućio slanje kolačića za autentifikaciju. * **Mapa mutation** - Ovdje se nalaze funkcije za specifične zahtjeve poput authCompany.ts, auth.ts, i login.ts koji obrađuju logiku autentifikacije, registracije i prijave korisnika. * **Mapa styles** - Sadrži colors.ts i easy-rent-light-theme.ts za definiranje prilagođenih boja i tema koje aplikacija koristi, što olakšava održavanje dosljednog izgleda korisničkog sučelja.

Interakcija između frontenda i backenda

- **HTTP zahtjevi i odgovori** - Frontend šalje HTTP zahtjeve na backend koristeći fetcher funkciju. Svi zahtjevi koriste credentials: 'include', što znači da će kolačići za autentifikaciju automatski biti uključeni, čime se osigurava da su zahtjevi sigurni. Backend zahtjevima rukuje u 'views.py' datoteci odgovarajuće aplikacije metodom mapiranom u 'urls.py'.

- **Prijava i sesija** Nakon uspješne prijave, backend šalje kolačić s tokenom koji se pohranjuje u pregledniku. Frontend koristi te podatke kako bi provjerio korisničku prijavu i zaštitio pristup zaštićenim stranicama.
- **Zaštita ruta** Na osnovu korisničkog statusa (prijavljen ili ne), frontend upravlja navigacijom i osigurava da samo ovlašteni korisnici mogu pristupiti određenim dijelovima aplikacije, kao što su korisnički i kompanijski profili.

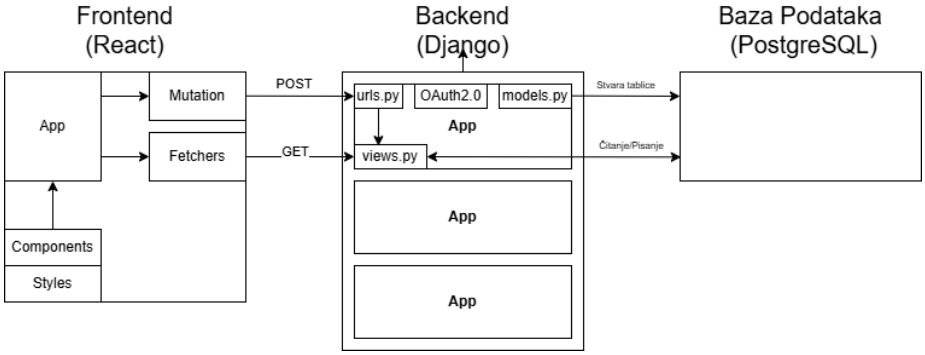
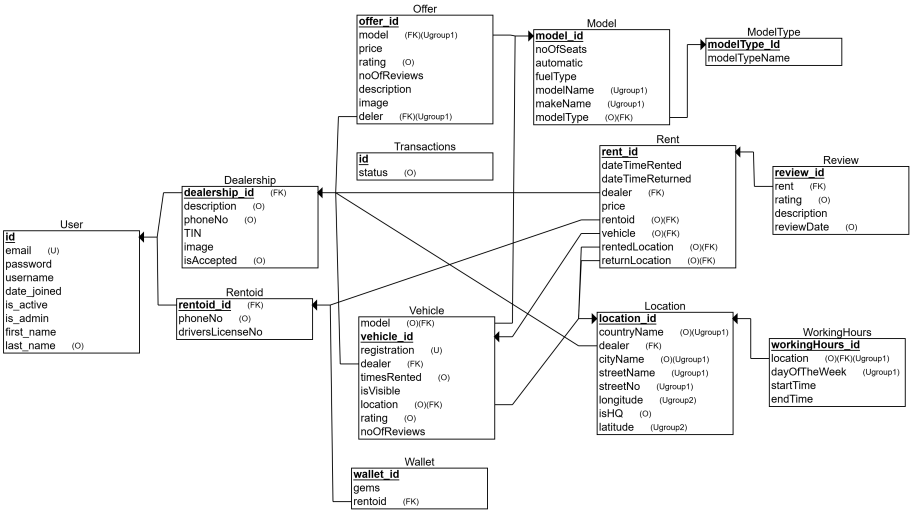


Figure 2: Dijagram Strukture drawio

Slika 4.2 Dijagram organizacije aplikacije

Baza podataka



Slika 4.3 Shema baze podataka

Napomene:

(public) označava informacije dostupne svima.
(private) označava informacije dostupne samo uključenim stranama i administratoru.
Za realizaciju baze korišten je *Django framework* zbog kojeg su automatski generirani primarni ključevi pod oznakom `_id`
Tablica `User` je također automatski stvorena i preddefinirana > (Za više informacija Django dokumentacija navedena je u literaturi od točaka 7 do 10)

Sve tablice su otvorene za proširenje

Tablice:

User (private)

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator korisnika
email	VARCHAR(254) (Jedinstven)	Email korisnika
password	VARCHAR(50)	Lozinka korisnika
username	-	-
date_joined	DATE	Datum registracije korisnika
is_active	BOOLEAN	Status aktivnosti korisnika
is_admin	BOOLEAN	Status administratora
first_name	VARCHAR(50)	Ime korisnika
last_name	VARCHAR(50) (Opcionalno)	Prezime korisnika

1. Sadrži podatke za prijavu klijenata `Dealership` i `Rentoid`.
2. PK: `id` SK: `email`
3. Tablicu je automatski generirao Django framework
4. `username` je trenutačno samo artefakt Django arhitekture. Nema konkretnu svrhu.
5. `first_name` i `last_name` koriste se i za kuće i za korisnike, ali kuće ostavljaju prezime prazno

Rentoid (public)

Atribut	Tip podatka	Opis varijable
rentoid_id	INT	Jedinstveni identifikator korisnika
phoneNo	VARCHAR(20) (Opcionalno)	Broj telefona korisnika
driversLicenseNo	VARCHAR(16)	Broj vozačke dozvole korisnika

1. Izveden iz `User`
2. Sadrži osnovne podatke o rentoidu.
3. Povijest najma/recenzija može se pregledati u tablici `Rent`
4. Podatci o trenutačnom stanju računa nalaze se u tablici `Wallet`

Dealership (public)

Atribut	Tip podatka	Opis varijable
dealership_id	INT	Jedinstveni identifikator autokuće
description	TEXT (Opcionalno)	Opis autokuće
phoneNo	VARCHAR(20) (Opcionalno)	Broj telefona autokuće
TIN	VARCHAR(16)	Porezni identifikacijski broj autokuće
image	BINARY	Naslovna slika autokuće
isAccepted	BOOLEAN (Opcionalno)	Status prihvaćenosti

1. Izveden iz `User`
2. Sadrži osnovne podatke o autokući.
3. Ponude autokuće dostupne su u tablici `Offer`
4. Dostupna vozila pohranjena su u tablici `Vehicle`
5. U tablici `Location` navedene su sve lokacije gdje kuća drži automobile uključujući i HQ adresu

Location (public)

Atribut	Tip podatka	Opis varijable
location_id	INT	Jedinstveni identifikator za lokaciju
dealership	INT (FK)	vlasnik lokacije (autokuća)
countryName	VARCHAR(50) (Opcionalno)	Država
cityName	VARCHAR(50) (Opcionalno)	Grad
streetName	VARCHAR(100)	Naziv ulice
streetNo	VARCHAR(10)	Broj ulice
isHQ	BOOLEAN (Opcionalno)	Oznaka sjedišta
latitude	NUMERIC(9 6)	Geografska širina
longitude	NUMERIC(9 6)	Geografska dužina

1. Sadrži sve dostupne lokacije i kome pripadaju.
2. PK: `location_id`, SK1:`{countryName, cityName, streetName, streetNo}`, SK2: `{latitude, longitude}`

3. Autokuća može imati više lokacija.
4. `isHQ` označava je li lokacija sjedište autokuće

WorkingHours (public)

Atribut	Tip podatka	Opis varijable
workingHours_id	INT	Jedinstveni identifikator za radno vrijeme
location	INT (FK) (Opcionalno)	ID lokacije
dayOfTheWeek	ENUM	Dan u tjednu
startTime	TIME	Vrijeme početka rada
endTime	TIME	Vrijeme završetka rada

1. Drži informacije o početku i završetku radnog vremena lokacije za dane u tjednu.
2. PK: `workingHours_id` SK: `{location, dayOfTheWeek}`
3. `dayOfTheWeek` ima format prva tri slova dana velikim početnim slovom npr. `Mon` za ponedjeljak

Wallet (private)

Atribut	Tip podatka	Opis varijable
wallet_id	INT	Jedinstveni identifikator novčanika
rentoid	INT	ID vlasnika (rentoida)
gems	INT	broj gemova na računu

1. Podatci o digitalnom novčaniku
2. PK: `wallet_id`
3. gemovi su valuta koju koristi EasyRent i jedan gem je ekvivalentan jednom europskom centu

Transactions (private)

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator transakcije
status	VARCHAR(20) Opcionalno	Status transakcije

1. Bilježi broj i status transakcija
2. PK: `id`

Offer (public)

Atribut	Tip podatka	Opis varijable
offer_id	INT	Jedinstveni identifikator za ponudu
price	DECIMAL	Cijena ponude
rating	DECIMAL (Opcionalno)	Ocjena ponude
noOfReviews	INT	Broj recenzija
description	TEXT (Opcionalno)	Opis ponude
picture	BINARY	Slika ponude
model	INT (FK)	ID modela
dealer	INT (FK)	ID prodajnog salona

1. Detalji o cijeni i dostupnosti određenog modela vozila određene autokuće.
2. PK: `offer_id` SK: `{dealer, model}`
3. `price` odražava trenutačnu cijenu modela koja može varirati tijekom vremena.
4. `rating` i `noOfReviews` ažuriraju se svakom novom recenzijom.

Model (public)

Atribut	Tip podatka	Opis varijable
model_id	INT	Jedinstveni identifikator modela
noOfSeats	INT	Broj sjedala
automatic	BOOLEAN	Automatski prijenos da/ne
fuelType	ENUM	Vrsta goriva
modelName	VARCHAR(50)	Naziv modela
makeName	VARCHAR(50)	Naziv proizvođača
modelType	INT (FK) (Opcionalno)	ID Tipa modela

1. Sadrži dostupne informacije o modelu vozila.
2. PK: `model_id`, SK: `{modelName, makeName}`
3. `fuelType` poprima znakovne vrijedonsti: `G` za benzin, `D` za Diesel ili `E` za električno napajanje

ModelType (public)

Atribut	Tip podatka	Opis varijable
modelType_Id	INT	Jedinstveni identifikator za tip modela
modelTypeName	ENUM	Naziv tipa modela

1. Dodatne infomacije o tipu modela
2. PK: `modelTypeID`
3. `modelTypeName` poprima vrijednosti `SUV`, `LIMO` ili `COMPACT`

Vehicle (private)

Atribut	Tip podatka	Opis varijable
vehicle_id	INT	Jedinstveni identifikator vozila
registration	VARCHAR(20) (jedinstveno)	Registracijski broj vozila
model	INT (FK) (Opcionalno)	ID modela
dealer	INT (FK)	ID vlasnika (autokuće)
timesRented	INT (Opcionalno)	ukupan broj najmova
rating	DECIMAL (Opcionalno)	ukupna ocjena vozila
noOfReviews	INT	ukupan broj recenzija
location	INT (FK) (Opcionalno)	ID lokacije vozila
IsVisible	BOOLEAN	Prikazuje li se vozilo u objavama

1. Informacije o određenom vozilu.
2. Dostupno samo autoriziranim korisnicima (vlasnicima).
3. PK: `vehicle_id` PK: `registration`
4. `location` označava trenutačnu lokaciju vozila, inače je vozilo iznajmljeno.
5. `timesRented`, `rating` i `noOfREviews` ažuriraju se nakon svakog iznajmljivanja
6. `isVisible` Određuje hoće li se vozilo prikazivati u objavama kompanije

Rent (private)

Atribut	Tip podatka	Opis varijable
rent_id	INT	Jedinstveni identifikator za najam
vehicle	INT (FK) (Opcionalno)	ID vozila koje se unajmljuje
dealer	INT (FK) (Opcionalno)	ID kuće koja iznajmljuje vozilo
rentoid	INT (FK) (Opcionalno)	ID osobe koja unajmljuje vozilo
dateTimeRented	DATETIME	Datum i vrijeme početka najma
dateTimeReturned	DATETIME	Datum i vrijeme povrata vozila

Atribut	Tip podatka	Opis varijable
rentedLocation	INT (Opcionalno)	ID lokacije preuzimanja
returnLocation	INT (Opcionalno)	ID lokacije povratka
price	DECIMAL(10 2) (Opcionalno)	Cijena najma

1. Sadrži privatne podatke o najmu.
2. PK: `rent_id`
3. `price` se u trenutku čita iz tablice **`Offer`**, ali se trajno pohranjuje radi obračuna zarade.
4. `dealer` se pohranjuje u slučaju da je vozilo izbrisano iz baze podataka iz kojeg se inače može čitati isti podatak

Review (public)

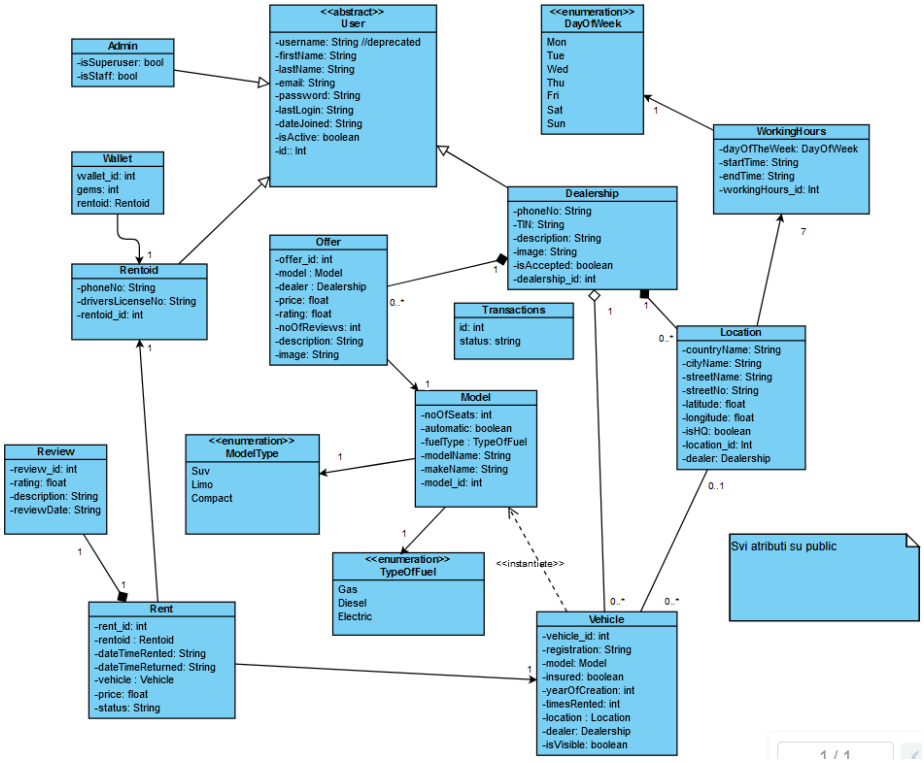
Atribut	Tip podatka	Opis varijable
review_id	INT	Jedinstveni identifikator za recenziju
rent	INT (FK)	ID najma na koji je ostavljena recenzja
rating	INT (Opcionalno)	Ocjena recenzije
description	TEXT	Tekst recenzije
reviewDate	DATE (Opcionalno)	Datum recenzije

1. Sadrži detalje o recenziji.
2. PK: `review_id`
3. Zapisi u recenziji se dodaju u trenutku kada se definira `dateTimeReturned` i ostaju prazni dok ih rentoid ne ispuni što može biti sve do isteka vremenskog roka.
4. Samo se polje `rating` mora definirati kako bi se recenzija smatrala ispunjenom, dok ostala polja mogu ostati nedefinirana
5. Polje `rating` je broj od 1 do 5
6. Ako rentoid odbije napraviti recenziju (što je potrebno eksplicitno definirati ili pri isteku vremena) tek onda se prazni zapis miče iz tablice
7. Recenzije koje su jednom definirane ne mogu se promijeniti niti maknuti iz tablice

Dijagram razreda

Na slici 5.4 prikazan je dijagram razreda za modele. Modeli služe za preslikavanje tablica baze podataka u backend program. Svaka instanca

modela predstavlja jednu n-torku baze podataka. Modeli izvučeni iz baze podataka pretvaraju se u JSON format koji se šalje na frontend.



Slika 4.4 Dijagram razreda

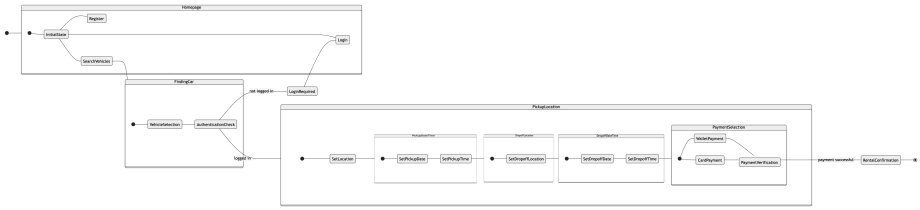
Dinamičko ponašanje aplikacije

Dinamičko ponašanje aplikacije odnosi se na način na koji objekti u sustavu evoluiraju kroz vrijeme, uključujući prijelaze između različitih stanja. To uključuje aktivnosti, događaje, odluke i interakcije unutar aplikacije. UML dijagrami stanja omogućuju vizualizaciju tih promjena i olakšavaju razumijevanje dinamike sustava.

Razumijevanje promjena stanja neophodno je za pravilno funkcioniranje aplikacije jer pruža uvid u interakcije među objektima, komponentama i korisnicima tijekom rada sustava. Korištenjem UML dijagrama stanja i aktivnosti moguće je vizualizirati prijelaze i stanja objekata, identificirati potencijalne probleme, osigurati točnu implementaciju te poboljšati komunikaciju među članovima tima.

UML dijagrami stanja

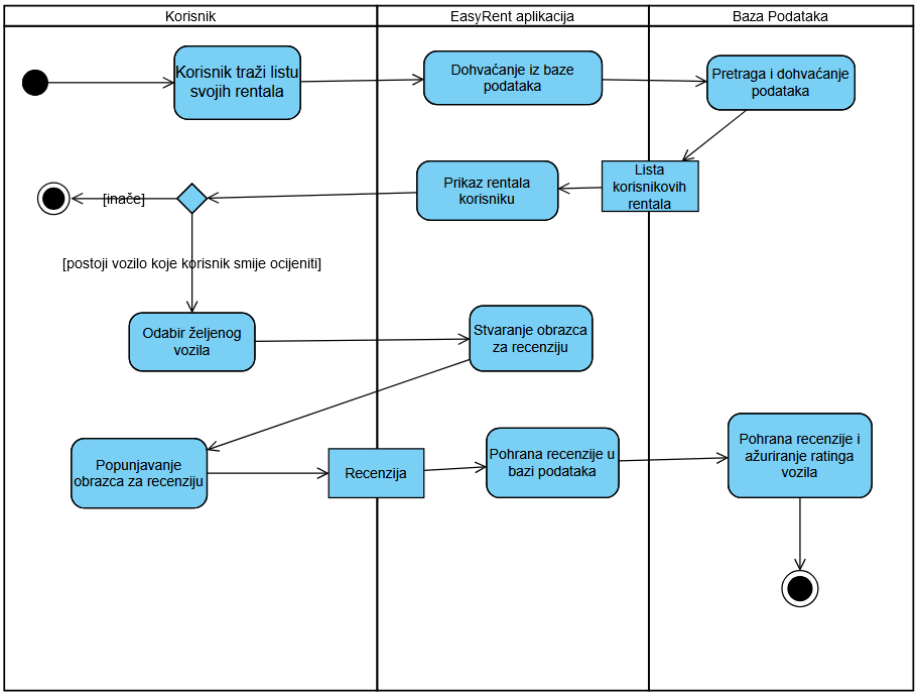
Na slici 4.5 prikazan je dijagram stanja za proces iznajmljivanja vozila. Kada korisnik otvori početnu stranicu aplikacije može se registrirati, ulogirati ili pretraživati vozila s različitim filtrima. Pretraživati vozila može i korisnik koji nije ulogiran, ali ako želi iznajmiti vozilo mora se ulogirati, što ga vraća na početnu stranicu. Kada korisnik pronađe odgovarajuće vozilo prikazuje mu se forma za iznajmljivanje. Korisnik prvo bira lokaciju s koje želi preuzeti vozilo, a aplikacija mu vraća kalendar na kojem su prikazani svi mogući datumi i vremena kada se vozilo može preuzeti na toj lokaciji. Kada korisnik odabere i vrijeme preuzimanja vozila dozvoljava mu se odabir lokacije na kojoj će ostaviti vozilo, a koja mora pripadati istoj kompaniji kojoj pripada i vozilo. Na kraju aplikacija prikazuje kalendar za odabir datuma i vremena vraćanja vozila. Na kalendaru je prikazano kada korisnik najkasnije mora vratiti vozilo (ako je isto vozilo rezervirano za neki termin u budućnosti). Kada korisnik popuni formu, omogućuje mu se plaćanje. Korisnik može platiti karticom ili svojim virtualnim EasyRent novčanikom. Ako je plaćanje uspješno, korisniku se prikazuje poruka da je vozilo uspješno iznajmljeno.



Slika 4.5 Dijagram stanja za iznajmljivanje vozila

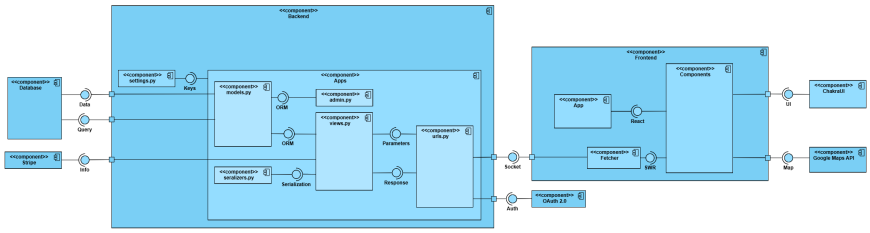
UML dijagrami aktivnosti

Dijagram aktivnosti prikazuje tijek izvršavanja određenog procesa. Osim za razumijevanje toka podataka unutar aplikacije, koristi se za analizu poslovnih procesa. Dijagram aktivnosti (slika 4.6) prikazuje tok podataka pri ostavljanju recenzije. Ulogirani korisnik na svojem profilu može vidjeti listu svojih (prošlih, tekućih i budućih) rentala koje aplikacija dohvaća iz baze podataka. Korisnik smije ocijeniti vozilo ako se ono nalazi u listi njegovih prošlih rentala te ako on to vozilo još nije ocijenio za tu instancu rentala. Ako takav rental ne postoji proces ostavljanja recenzije se završava. Ako postoji barem jedan prošli rental kojega korisnik nije ocijenio, korisnik može zatražiti obrazac za recenziju tog rentala. Korisnik popunjava obrazac (ostavlja ocjenu između 1 i 5 te komentar) i šalje recenziju na web aplikaciju. Aplikacija pohranjuje recenziju u bazi podataka gdje se dodatno ažuriraju prosječna ocjena i broj recenzija za vozilo kojem pripada ocijenjeni rental.



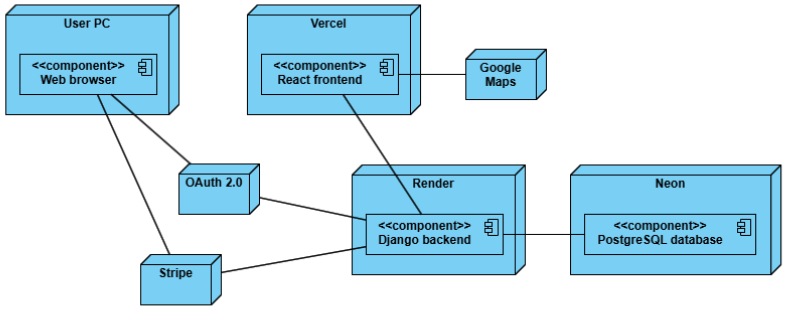
Slika 4.6 Dijagram aktivnosti za ostavljanje recenzije

Na slici 5.1 prikazan je dijagram komponenti. Dijagram prikazuje arhitekturu frontenda i backenda te sučelja kojim oni komuniciraju te kojim backend komunicira s bazom podataka. Backend je kodiran koristeći Django framework te se sastoji od settings.py datoteke te nekoliko app mapa. U settings.py datoteci se čuvaju API ključevi npr. za komunikaciju s bazom podataka i Stripe-om. App mape smanjuju međuovisnost i povećavaju koheziju unutar backenda, a također olakšavaju podjelu posla unutar tima jer svaki član tima može u svojoj mapi implementirati svoj dio funkcionalnosti. Svaka app mapa se sastoji od svojih modela, koji služe za preslikavanje redaka baze podataka u objekte, view datoteke, u kojoj se nalaze endpoint funkcije, url datoteke, koja služi kako bi se endpoint funkcije povezale s URL putanjama te datoteke sa serijalizatorima koji modele pretvaraju u JSON format koji se šalje frontendu. Frontend je kodiran koristeći React biblioteke. Frontend komunicira s google maps API-jem za prikaz mapa.



Slika 5.1 Dijagram komponenti

Na slici 5.2 prikazan je dijagram razmještaja. Sustav koristi klijent-poslužitelj arhitekturu. Frontend, backend i baza podataka su svi razmješteni na različitim serverima. Frontend se nalazi na vercelovom serveru, backend na renderovom, a baza podataka na neon serveru. Serveri međusobno i s korisnikom komuniciraju koristeći protokol HTTP, a korisnik za login može koristiti i OAuth2.0.



Slika 5.2 Dijagram razmještaja

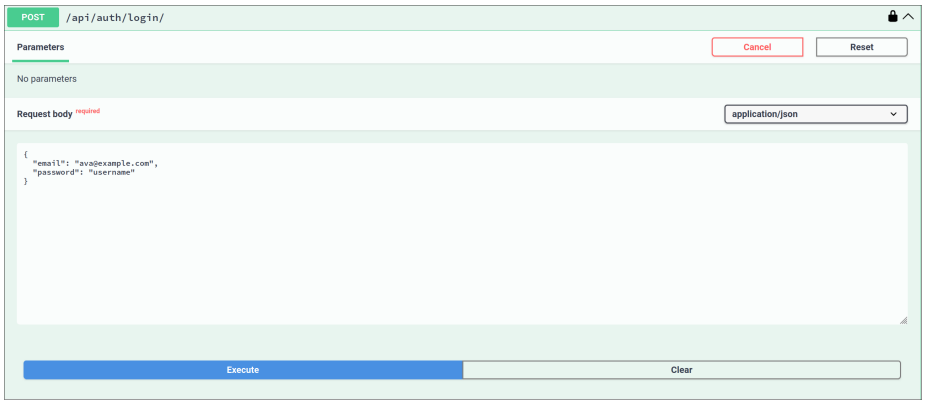
Ispitivanje programskog rješenja

Ispitivanje komponenti

Napomena: koristi se SwaggerUI za testiranje, pokrene se lokalna grana i pristupi localhost:8000/api/schema/docs, kod serializera je view.py od točnog appleta iz /api/{applet}

Postoje i scripte koje programski provode iste testove u repozitoriju.

Prikaz rada SwaggerUI-a



Slika 6.1 Ulaz u SwaggerUI

Slika 6.2 Izlaz is SwaggerUIa

Ispit komponente 1 - kupnja valute u wallet

/api/wallet/buyGems

Preduvjet: User mora biti ulogiran

Ulaz očekivanog ponašanja: - amount: 5000

Izlaz - status: 200, “link na stripe checkout”

Ulaz rubnog slučaja: - amount: 2

Izlaz - status: 404, “The Checkout Session’s total amount due must add up to at least €0.50 eur”

Ulaz nepodržanog ponašanja: - amount: test

Izlaz - status: 400, “Nan provided”

Ispit komponente 2 - provjera transakcije u bazi podataka

/api/wallet/checkTransaction/{transaction_id}

Preduvjet: User mora biti ulogiran

Ulaz očekivanog ponašanja: - transaction_id: 5

Izlaz - status: 200, "Successful transaction."

Ulaz rubnog slučaja: - transaction_id: 30

Izlaz - status: 200, "Unsuccessful transaction."

Ulaz neopodržanog ponašanja: - transaction_id: test

Izlaz - Django error, nije moguće ni unijeti takav url bez alata poput curla.

Ispit komponente 3 - login

/api/auth/login/

Ulaz očekivanog ponašanja: - email: ava@example.com - password: username

Izlaz - status: 200, "Successful login"

Ulaz rubnog slučaja: - email: ava@example.com - password: kriva_lozinka

Izlaz - status: 400, "Invalid password"

Ulaz rubnog slučaja: - email: ava@example.com.com - password: kriva_lozinka

Izlaz - status: 400, "Invalid email"

Ispit komponente 4 - earnings

Preduvjet: Kompanija mora biti ulogirana

Ulaz očekivanog ponašanja: - year: 1

Izlaz Status: 200


```
{
  "results": {
    "totalEarnings": "23471.04",
    "yearEarnings": 0,
    "totalRentals": 18,
    "yearRentals": 0,
    "monthlyEarnings": {
      "1": 0,
      "2": 0,
```

```
      "3": 0,
      "4": 0,
      "5": 0,
      "6": 0,
      "7": 0,
      "8": 0,
      "9": 0,
      "10": 0,
      "11": 0,
      "12": 0
    }
  },
  "isLastPage": true
}
```


Ulaz rubnog ponašanja: - year: 30

Izlaz - isti izlaz kao gore jer se dohvati zadnja dostupna godina

Ulaz neočekivanog ponašanja: - year: -0.1

Izlaz - sucess: "0" - message: "Company does not exist or has no rents yet!"

Ispit komponente 5 - ponude najbolje vrijednosti(best offer)

/api/home/best-value/

Preduvjet: User mora biti ulogiran

Ulaz očekivanog ponašanja: - Max number of offers per page: 1 - Page number, starts with: 1

Izlaz


```
{
  "offers": [
    {
      "image": "http://127.0.0.1:8000/media/offers/citroenc1.jpeg",
      "companyName": "Last minute",
      "makeName": "Citroen",
      "modelName": "C1",
      "noOfSeats": 5,
      "automatic": false,
      "price": "42.00",
      "rating": 5,
      "noOfReviews": 1,
      "offer_id": 259
    }
```

```
  ],
  "last": false
}
```


Ulaz rubnog ponašanja: - Max number of offers per page: 4 - Page number, starts with: 2

Izlaz


```
{
  "offers": [
    {
      "image": "http://127.0.0.1:8000/media/offers/skodafabia.jpeg",
      "companyName": "enterprise",
      "makeName": "Skoda",
      "modelName": "Fabia",
      "noOfSeats": 4,
      "automatic": true,
      "price": "48.00",
      "rating": 5,
      "noOfReviews": 1,
      "offer_id": 100
    },
    {
      "image": "http://127.0.0.1:8000/media/offers/citroenc3.jpeg",
      "companyName": "dollar",
      "makeName": "Citroen",
      "modelName": "C3",
      "noOfSeats": 5,
      "automatic": true,
      "price": "55.00",
      "rating": 5,
      "noOfReviews": 1,
      "offer_id": 412
    },
    {
      "image": "http://127.0.0.1:8000/media/offers/fiatpanda.jpeg",
      "companyName": "avantcar",
      "makeName": "Fiat",
      "modelName": "Panda",
      "noOfSeats": 4,
      "automatic": true,
      "price": "59.00",
      "rating": 5,
      "noOfReviews": 1,
      "offer_id": 144
    },
```

```
    {
      "image": "http://127.0.0.1:8000/media/offers/fiat500.jpeg",
      "companyName": "dollar",
      "makeName": "Fiat",
      "modelName": "500",
      "noOfSeats": 4,
      "automatic": false,
      "price": "63.00",
      "rating": 5,
      "noOfReviews": 1,
      "offer_id": 223
    }
  ],
  "last": false
}
```


Ulaz neočekivanog ponašanja: - Max number of offers per page: 0 - Page number, starts with: 0

Izlaz - Prvih 9 offera u istom formatu kao prije.

Ispit komponente 6 - user-info

/api/auth/user-info/

Ulaz očekivanog ponašanja: - Ulogirani korisnik

Izlaz - role: “user”, - name: “ava white”

Ulaz rubnog ponašanja: - Ulogirana kompanija

Izlaz - role: “company”, - name: “Adios d.o.o”

Ulaz neočekivanog ponašanja -nitko nije ulogiran

Izlaz - role: “guest”

Navedenim pristupom sa SwaggerUI je moguće testirati svaki endpoint i podsustav. Ponovno, svaki test je dostupan u repozitoriju componentTest mapi.

Ispitivanje sustava

Ispit sustava 1 - login

Ulaz očekivanog ponašanja: - email - lozinka

Izlaz - povratak na homepage

Ispit sustava 2 - tražilica autiju

Ulaz: - pickupDate - dropoffDate - pickupLocation - dropoffLocation - pickupTime - dropoffTime

Izlaz - prikaz dostupnih autiju za dane parametre

Ispit sustava 3 - filtracija nakon inicijalne pretrage

Ulaz: - Seats - Car Type - Transmission

Izlaz - prikaz dostupnih autiju za dane parametre

Ispit sustava 4 - traženje s neispravnim parametrima (rubni slučajevi)

Ulaz: - pickupDate - dropoffDate - pickupLocation - dropoffLocation - pickupTime - dropoffTime

Izlaz - prikaz da ne postoje dostupna vozila s danim parametrima

Video of selenium test

Korištene tehnologije i alati

1. Programski jezici

- **Python 3.11/3.12** Python je odabran zbog svoje čitljivosti, jednostavnosti korištenja i bogate zajednice te ekosustava. S obzirom na opseg projekta i dano vrijeme, tim je zaključio da je Python savršen izbor kao visoko apstraktni programski jezik koji se brzo uči.
- **TypeScript 5** TypeScript je korišten za izradu dinamičkog frontenda aplikacije. Budući da je JavaScript standardni jezik za rad s web preglednicima, njegova primjena je logičan izbor u kombinaciji s React frameworkom. TypeScript je proširenje JavaScripta uz mogućnosti statičkog tipiziranja, što olakšava održavanje koda i smanjuje broj grešaka tijekom razvoja. Njegova integracija s Reactom poboljšava produktivnost tima.

–

2. Radni okviri i biblioteke

- **React 18 (Next.js 15.0.3)** React je korišten za razvoj frontenda zbog svoje fleksibilnosti, modularne arhitekture i mogućnosti jednostavne integracije s TypeScriptom. Next.js omogućava server-side rendering (SSR) i poboljšava performanse stranice, što je ključno za bolji korisnički doživljaj. Dio tima je imao prethodno iskustvo s Reactom.
- **Django 5.1.2** Za razvoj serverskog dijela aplikacije korišten je Django Django omogućuje veliku razinu apstrakcije pri korištenju i modeliranju baze podataka sa svojim integriranim modelima. Također pruža veliku mogućnost skalabilnosti i sigurnosti sa “out-of-the-box” modelima i

mogućnostima podjela na manje aplikacije. Serverski dio aplikacije podijeljen je na aplikacije radi preglednosti i skalabilnosti te je svaka od tih aplikacija posebno pogodna za daljnje skaliranje

–

3. Baza podataka

- **PostgreSQL 17** PostgreSQL je relacijska baza podataka odabrana zbog svoje stabilnosti, skalabilnosti i bogatog skupa značajki. Svi članovi tima već imaju iskustva s ovom bazom, što ubrzava razvoj. Njegova kompatibilnost s Django ORM-om omogućuje bržu implementaciju složenih funkcionalnosti.

–

4. Razvojni alati

- **VSCode** Visual Studio Code je korišten kao primarni alat za razvoj zbog svoje brzine, bogatog skupa ekstenzija i podrške za više jezika, uključujući Python i JavaScript.
- **NeoVim** NeoVim je korišten od strane članova tima koji preferiraju minimalističke alate s naprednim mogućnostima prilagodbe. Njegova brzina i podrška za LSP (Language Server Protocol) omogućuju efikasan rad.
- **Git 2.47** Git je korišten za verzioniranje koda i suradnju na projektu.

–

5. Alati za ispitivanje

- **SwaggerUI OAS 3.0** SwaggerUI je korišten za dokumentaciju i testiranje API-ja. Omogućava lako razumijevanje strukture API-ja i testiranje njegovih funkcionalnosti bez dodatnih alata.
- **Selenium 4.28** Selenium je korišten za automatsko testiranje web sučelja.

–

6. Alati za razmještaj

- **Nisu korišteni**

–

7. Cloud platforma

- **Render** Render je korišten za hostanje backend aplikacije zbog svoje jednostavnosti i podrške za automatsko razmještanje nakon promjena na GitHub repozitoriju.
- **Vercel** Vercel je korišten za hostanje frontenda. Njegova optimizacija za Next.js projekte omogućuje brzu isporuku sadržaja i odlične performanse.
- **Neon** Neonje odabran za hostanje PostgreSQL baze podataka zbog svojih performansi, podrške za skaliranje i fleksibilnosti potrebne za naš projekt.

Lokalno pokretanje

Instalacija alata

- Potrebno je instalirati sljedeće:
  - https://nodejs.org/en
  - https://www.python.org/downloads/
  - https://www.postgresql.org/download/
  - https://git-scm.com/

Namještanje backenda:

```
git clone https://github.com/fran-galic/EasyRent.git
cd backend
pip install -r requirements.txt
```

- Izraditi PostgreSQL bazu podataka
- U backend/backend/settings.py navigirati do objekta DATABASES i podesiti atribute da odgovaraju PostgreSQLu (česti portovi su 5432 i 5433)
- Migracija baze:

```
cd backend
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
python manage.py runserver
```

- Backend bi trebao biti dostupan na 127.0.0.1:8000
- Otvori 127.0.0.1:8000/admin
- Idi na SITES Sites i promijeni example na
  - Domain name: 127.0.0.1:8000
  - Display name: localhost

(Opcionalno) Postavljanje slanja e-maila za potvrdu:

- Postaviti sljedeće u settings.py (ili .env u backend direktoriju)

```
GOOGLE_CLIENT_ID=
GOOGLE_CLIENT_SECRET=
HOST_PASSWORD=
HOST_EMAIL=
```

- Ponovno otići na admin stranicu
- Dodati Social Accounts Social applications
  - Provider: Google
  - Name: google
  - Client id: GOOGLE_CLIENT_ID
  - Secret key: GOOGLE_CLIENT_SECRET
  - dodati 127.0.0.1:8000 u sites

(Opcionalno) Dodavanje plaćanja sa Stripeom

- Postaviti sljedeće u settings.py (ili .env u backend direktoriju)

```
STRIPE_PK=
STRIPE_SK=
STRIPE_WH=
```

- Preuzimanje stripe-cli alata.
- Login kroz stripe-cli, koji nam daje odgovarajući webhook koji se stavlja u .env(STRIPE_WH)
- Pokretanje event listenera na localhost:8000/api/wallet/webhook

```
stripe login
stripe listen --forward-to localhost:8000/api/wallet/webhook
```

Namještanje frontenda:

- U drugom terminalu pokrenuti:

```
cd front
npm install
npm run dev
```

- Frontend bi trebao biti dostupan na localhost:3000

(Opcionalno) Namještanje Google Mapsa:

- Za korištenje Google Maps API-ja lokalno, dodajte API ključ u `.env.local` datoteku u frontend direktoriju.
- *Kreirajte ili uredite `.env.local`** i dodajte:

```
NEXT_PUBLIC_GOOGLE_MAPS_API_KEY=YOUR_GOOGLE_MAPS_API_KEY
```

- Umjesto `YOUR_GOOGLE_MAPS_API_KEY` unesite svoj API ključ iz Google Cloud konzole.
- Na Google Cloud konzoli potrebno je omogućiti sljedeće API-je za ispravan rad aplikacije:
  - Maps JavaScript API – za prikaz interaktivnih karata na web aplikaciji.
  - Geocoding API – za pretvaranje adresa u geografske koordinate i obrnuto.
  - Places API – za dohvaćanje informacija o mjestima (npr. restorani, trgovine).
  - Directions API – za izračunavanje ruta i navigaciju između lokacija.

Backend deploy (Render) i database deploy (Neon)

- Izraditi korisničke račune za navedene servise
- Povezati GitHub repozitorij s Renderom i odabrati direktorij za deploy
- Postaviti environment na točnu verziju pythona
- Postaviti "Build Command":

```
pip install -r requirements.txt && python manage.py collectstatic --no input
```

- Postaviti "Start Command":

```
gunicorn backend.wsgi
```

- Izraditi novu bazu podataka na Neonu
- Kopirati connection string i postaviti ga u settings.py ili .env na backendu
- Na backendu u settings.py (ili u .env) podesiti sljedeće varijable:

```
SECRET_KEY=
ALL_HOST=
DEBUG=
DATABASE=
STRIPE_PK=
STRIPE_SK=
STRIPE_WH=
```

- Na stripe Dashboardu na developers > webhook napraviti novi webhook i omogućiti Checkout i charge eventove
- Dobiveni webhook key staviti u STRIPE_WH u .env datoteci
- Na Neon Dashboardu idi na Settings > Connection Details i postavi pravilo pristupa:
  - Allowed IPs: Stavi 0.0.0.0/0 ako razvijaš lokalno (ne zaboravi ovo ograničiti u produkciji)
  - SSL: Preuzmi ca.crt certifikat s Neona i dodaj ga projektu ako koristite SSL

- Pokrenuti naredbe za migraciju baze

```
python manage.py makemigrations
python manage.py migrate
```

Frontend deploy (Vercel)

Vercel je platforma za hosting frontend aplikacija, posebno optimizirana za **Next.js**, koji su upravo oni razvili. Vercel omogućuje jednostavan i automatiziran proces deploya povezivanjem s GitHub repozitorijem, što znači da se aplikacija automatski ažurira svaki put kada se napravi push na glavnu granu (`main`).

- Prijavite se na Vercel koristeći GitHub račun.
- Povežite željeni GitHub repozitorij s Vercelom.
- Vercel automatski prepoznaje Next.js projekt i primjenjuje odgovarajuće postavke.
- Nakon što je repozitorij povezan, svaka promjena koja se **push-a** na `main` granu automatski pokreće novi deploy i izvodi sljedeće:
  - Instalaciju ovisnosti
  - Generiranje builda
  - Postavljanje na produkciju
- Potrebne environment varijable (npr. API ključevi) mogu se definirati unutar Vercel dashboarda pod sekcijom Settings > Environment Variables.
- U slučaju grešaka tijekom deploya, Vercel pruža detaljne logove unutar sučelja kako biste brzo identificirali i riješili problem.
- Također je moguće pokrenuti lokalni build pomoću naredbe:

  ```
  vercel build
  ```

- Ako je potrebno, deploy se može pokrenuti ručno unutar Vercel dashboarda klikom na "Deploy" ili korištenjem CLI naredbe:

  ```
  vercel --prod
  ```

Osvrt na izradu projektnog zadatka

Ovaj projekt zahtijevao je puno više truda nego što smo na početku očekivali kao tim. Puno smo toga naučili, ali usput smo morali riješiti i brojne tehničke probleme. Većina članova tima se nikada ranije nije susrela s razvojem softvera, pa smo od samog početka morali učiti rad s frameworksima poput **Reacta** i **Next.jsa** za frontend te **Djangoa** za backend. Također, bilo je nužno savladati povezivanje backenda i frontenda kako bi aplikacija radila ispravno.

Uspjeli smo implementirati većinu funkcionalnosti koje smo zamislili, no neke stvari jednostavno nismo stigli realizirati, poput **chata između kompanija i**

korisnika. Osim toga, u procesu rada shvatili smo da određene funkcionalnosti nemaju smisla ili ih je bilo potrebno prilagoditi.

Također smo naučili kako **deployati aplikaciju**, i to kako za backend, tako i za frontend. Uz to, usvojili smo vještine pisanja **kvalitetne dokumentacije**, što je izuzetno važno za buduće developere i druge dionike projekta kako bi mogli razumjeti koncept aplikacije, njene funkcionalnosti i korisničke zahtjeve.

Tehnički izazovi

Naišli smo na niz tehničkih izazova tijekom razvoja, a neki od njih uključuju:

- **Deploy aplikacije** – Imali smo poteškoće s postavljanjem aplikacije na cloud servise, no uspješno smo ih riješili korištenjem **Vercela** za frontend i **Rendera** za backend.
- **Integracija sustava plaćanja i Google mapa** – Početne poteškoće smo riješili kroz detaljno proučavanje dokumentacije i debugiranje.
- **Povezivanje frontenda i backenda** – Rješavali smo probleme koji su nastajali zbog nekompatibilnosti API poziva i razlika u podacima između ova dva dijela sustava.
- **Pisanje kvalitetne dokumentacije** – Kroz projekt smo naučili kako strukturirati i dokumentirati aplikaciju na način koji je koristan za tim i buduće korisnike.
- **Broj svojstava** - Smatramo da smo pretjerali s brojem svojstava koje naša aplikacija pruža te zbog toga nismo mogli realizirati sve na željenoj razini kvalitete i urednosti. Također, kako su se članovi tima poboljšavali u korištenju novonaučenih alata, tako su vidjeli mnoga moguća poboljšanja u prethodno napravljenom, ali s manjkom vremena za provođenje promjena.

Stečena i potrebna znanja

Kroz rad na projektu stekli smo vrijedna znanja u:

- Razvoju frontend aplikacija pomoću **Reacta** i **Next.jsa**,
- Backend razvoju s **Django frameworkom**,
- Postavljanju aplikacije na cloud servise,
- Timskom radu i učinkovitoj komunikaciji unutar tima.
- Bolji osjećaj za potrebnim vremenom za izradu određenog broja svojstava.

Za brže i kvalitetnije ostvarenje projekta korisno bi bilo imati:

- **Bolje znanje backend arhitekture** kako bi se posao bolje organizirao,
- **Iskustvo s DevOps alatima** za efikasniji deployment i CI/CD procese,
- **Dublje razumijevanje sigurnosnih aspekata** aplikacije kako bi se smanjili potencijalni sigurnosni rizici.

Perspektive za nastavak rada

Smatram da bi realizacija aplikacije za stvarnu upotrebu u industriji zahtijevala dodatnu optimizaciju, uključujući:

- Pisanje modularnijeg i efikasnijeg koda,
- Implementaciju naprednijih algoritama za pretraživanje,
- Rješavanje sigurnosnih nedostataka kako bi aplikacija bila spremna za produkcijsko okruženje.

Što se tiče nastavka rada na projektu, većina članova tima, uključujući mene, smatra da smo napravili dovoljno dobar posao za potrebe kolegija. Nastavak rada na projektu u industrijskom okruženju zahtijevao bi dodatno vrijeme i resurse, ali trenutačno smo fokusirani na druge akademske obaveze.

Neimplementirane funkcionalnosti

Jedina funkcionalnost koju nismo uspjeli implementirati je:

- **Komunikacija između korisnika i kompanija (chat sustav)**.

Iako projekt ima prostora za poboljšanja, smatram da smo postigli jako dobar rezultat s obzirom na okolnosti i znanja koja smo imali na početku. Tim je bio odličan, a timski rad i redovita komunikacija su se pokazali ključnima za uspjeh projekta.

Upute za korištenje dnevnika promjena

- Svaka promjena projekta/dokumentacije treba biti zabilježena u tablici.
- **Rev.**: Unesite verziju promjene (npr. 0.1, 0.2).
- **Opis promjene/dodatka**: Kratko opišite što je promijenjeno.
- **Autori**: Unesite imena članova tima koji su radili na toj promjeni.
- **Datum**: Unesite datum kada je promjena napravljena.
- **Status**: Označite status promjene (npr. "Završeno", "U tijeku", "Odobreno").

Rev.	Opis promjene/dodatka	Autori	Datum	Status
0.1	Napravljen predložak	Fran Galić	24.10.2024.	Završeno
0.2	Dodavanje zahtjeva i obrazaca	Matija Kukić	31.10.2024.	Nedovršeno
0.4	Dodana aktivnost 1. sastanka i git workflow projekta	Fran Galić	1.11.2024.	Završeno
0.5	Dodani inicijalni dijagrami obrazaca uporabe i sekvencijski dijagrami	Karlo Brzak	4.11.2024.	Nedovršeno
0.6	Promjena strukture dokumentacije zahtjeva	Matija Kukić	6.11.2024.	Završeno
0.7	Dijagram i opis baze podataka	Vilim Hrupelj	10.11.2024.	Završeno
0.7	Promjena strukture dokumentacije zahtjeva	Matija Kukić	13.11.2024.	Završeno

Rev.	Opis promjene/dodatka	Autori	Datum	Status
0.8	Dodana aktivnost 2. i 3. sastanka	Galić Fran	14.11.2024.	Završeno
0.9	Dodan dijagram razreda, popravljeni sekvencijski dijagrami	Karlo Brzak	15.11.2024.	Završeno
0.10	Dokumentacija zahtjeva završena	Matija Kukić	15.11.2024.	Završeno
0.11	Dodana arhitektura sustava	Ivan Dujmić	15.11.2024.	Završeno
1.1	Dijagram stanja	Ivan Džanija	21.1.2025.	Završeno
1.2	Tehnologije za implementaciju aplikacije	Ivan Džanija	22.1.2025.	Nedovršeno
1.3	Ispitivanje programskog rješenja	Matija Kukić	23.1.2025	Nedovršeno
1.4	Promjena strukture dokumentacije za drugu reviziju	Ivan Dujmić	23.1.2025.	Završeno
1.5	Ažuriranje informacija dokumentacije za drugu reviziju	Ivan Dujmić	23.1.2025.	Završeno
1.6	Detaljniji opis tehnologija za implementaciju aplikacije	Ivan Dujmić	23.1.2025.	Završeno
1.7	Upute za puštanje u pogon	Ivan Dujmić, Ivan Džanija, Matija Kukić, Fran Galić	24.1.2025.	Završeno
1.8	Zaključak i budući rad	Fran Galić	24.1.2025.	Završeno
1.9	Ključni izazovi i rješenja	Fran Galić	24.1.2025.	Završeno
1.10	Poboljšanje ispitivanja programskog rješenja	Matija Kukić	24.1.2025.	Završeno
1.11	Arhitektura komponenata i razmještaja	Ivan Dujmić, Vilim Hrupelj	24.1.2025.	Završeno
1.12	Ažuriranje dijagrama i opisa baze podataka	Vilim Hrupelj	24.1.2025.	Završeno
1.13	Ažuriranje dijagrama razreda	Karlo Brzak	24.1.2025.	Završeno
1.14	Dijagram aktivnosti	Karlo Brzak	24.1.2025.	Završeno
1.15	Dijagram pregleda promjena	Ivan Dujmić	24.1.2025.	Završeno
1.16	Ispitivanje sustava	Matija Kukić	24.1.2025.	Završeno
1.17	Ažuriranje sekvencijskog dijagrama	Karlo Brzak	24.1.2025.	Završeno

Kontinuirano osvježavanje

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

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/

books/SE

5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new
7. https://docs.djangoproject.com/en/5.1/topics/auth/
8. https://django-allauth.readthedocs.io/en/latest/
9. https://docs.djangoproject.com/en/5.1/topics/db/models/
10. https://docs.allauth.org/en/dev/socialaccount/configuration.html
11. VisualParadigm https://online.visual-paradigm.com/
12. Subhash Chandra Yadav, Sanjay Kumar Singh - An Introduction to CLIENT/SERVER COMPUTING (https://aagasc.edu.in/cs/books/client-server-computing.pdf)

Dnevnik sastajanja

Kontinuirano osvježavanje

1. sastanak

- **Datum**: 22. listopada 2024.
- **Prisustvovali**: Biloglav Jakov, Brzak Karlo, Dujmić Ivan, Džanija Ivan, Galić Fran, Hrupelj Vilim, Kukić Matija Luka
- **Teme sastanka**:
  - **Sastanci**:
    - Sljedeći tjedan sastanak uživo, dogovor po potrebi.
    - Utorkom u 13:00.
    - Svaki član treba ukratko objasniti što je napravio.
  - **Uloge**:
    - **Baza podataka**: Vilim
    - **Frontend**: Jakov, Fran
    - **Backend**: Karlo, Ivan Dujmić, Ivan Džanija
    - **DevOps**: Matija (praćenje commitova, verzioniranje, rješavanje konflikata, GitHub poslovi)
    - **Dokumentacija**: Matija (zajednička), svaki član za svoj dio
  - **Tehnologije**:
    - **Backend**: Python (Django)
    - **Frontend**: Next.js, React
    - **Baza podataka**: PostgreSQL
  - **Prvi zadaci**:
    - SVI naučiti Git i GitHub:
      1. Prijava (ulogiraš se)
      2. Pullaš promjene

        3. Otvaraš novu granu
        4. Radiš promjene i commitaš kada napraviš značajan dio
        5. Pushanje i otvaranje pull requesta kada završiš značajan dio
        6. Matija provjerava i upravlja.
    - Razrada specifikacija aplikacije (za prezentaciju).
    - Skica u Figmi (Ivan Dujmić).
    - Prezentacija (Fran).
    - Drugi tjedan učiti tehnologije.
    - Bolje specificirati aplikaciju drugi tjedan.
- **Specifikacije aplikacije**:
    - **Uloge**:
        - Gost
        - Korisnik
        - Prodajna kuća
        - Admin
    - **Register**:
        - Za korisnika
        - Za kuću
    - **Login**
    - **Home page**:
        - Najpopularniji
        - Pretraživanje
    - **Listing page**
    - **Profil**:
        - Pregled
        - Uređivanje
    - **Car page**:
        - Unos novog automobila
        - Informacije o automobilu
    - **Stranica kuće**:
        - Ivan će dodatno specificirati dok radi u Figmi.

2. sastanak

- **Datum**: 29. listopada 2024.
- **Trajanje**: 5 sati
- **Prisustvovali**: Biloglav Jakov, Brzak Karlo, Dujmić Ivan, Džanija Ivan, Galić Fran, Hrupelj Vilim, Kukić Matija Luka

Teme sastanka:

- **Uvod**:
    - Kratko predstavljanje sastanka i potvrđivanje dnevnog reda.
    - Odabir zapisničara.
- **Sljedeći sastanak**:

  - Dogovor o sljedećem sastanku u utorak u 13:00 (predviđeno kraće trajanje).

- **Izvještaj o napretku**:
  - Svaki član tima iznosi kratki pregled rada do sada te planove do kraja tjedna.
  - Fokus na učenje novih tehnologija i korištenje Git-a i GitHub-a, uz savjete i pomoć po potrebi.

- **Zadaci za ovaj tjedan**:
  - Učenje novih tehnologija koje su relevantne za projekt.
  - Pisanje 4. poglavlja dokumentacije (zadužen Matija).
  - Razrada GitHub workflowa kako bi se definirali jasni koraci za commit i praćenje verzija.
  - Razrada arhitekture sustava

- **Rad na arhitekturi sustava**:
  - Razmotreni su glavni funkcionalni dijelovi aplikacije, uključujući:
    * **Home page**: Elementi poput kontaktiranja admina, prozora za prijavu/registraciju, i pretrage automobila prema lokaciji, datumu i vremenu.
    * **Register/Login**: Opcije registracije za korisnike i kompanije, uz različite zahtjeve za svaki tip korisnika (npr. broj vozačke dozvole za korisnike).
    * **Listing page**: Prikaz automobila s filterima za sjedala, vrstu vozila, cijenu i druge parametre.
    * **Profil korisnika**: Mogućnosti pregleda, uređivanja profila i praćenja rezervacija.
    * **Profil kompanije**: Više tabova za pregled i upravljanje vozilima, pregled zarade, lokacija i recenzija.
    * **Stranica automobila**: Detalji o pojedinom vozilu, uključujući mogućnost rezervacije, pregled recenzija i statistike o korištenju.
    * **Admin panel**: Upravljanje korisnicima i kompanijama, moderiranje recenzija i izvještaja, pristup statistikama cijele stranice.

- **Zaključak i zadaci do sljedećeg sastanka**:
  - Svi članovi dokumentiraju svoj rad i unose broj sati uloženih u zadatke.
  - Svaki član zadužen je za kontinuirano ažuriranje dokumentacije.
  - Priprema za sljedeći sastanak:
    * Razrada prvih dijelova aplikacije.
    * Završavanje ostalih poglavlja dokumentacije.
    * Dovršetak GitHub workflowa.

3. sastanak

- **Datum**: 5. studenoga 2024.

- **Trajanje**: 2 sata
- **Prisustvovali**: Biloglav Jakov, Brzak Karlo, Dujmić Ivan, Džanija Ivan, Galić Fran, Hrupelj Vilim

Teme sastanka:

- **Uvod:**
  - Kratko predstavljanje sastanka i potvrđivanje dnevnog reda.
  - Odabir zapisničara.
  - Potvrđeno da će svi dogovori oko koda i razvoja projekta biti na GitHubu osim kada raspravljamo o arhitekturi sustava (diskusije o arhitekturi odvijat će se uživo).
  - Dogovoreno je da se službena dokumentacija piše na hrvatskom jeziku, dok se razvoj sustava provodi na engleskom jeziku.
- **Sljedeći sastanak:**
  - Dogovoren je sljedeći sastanak za utorak u 13:00 (važan finalni sastanak).
- **Izvještaj o napretku:**
  - Svaki član tima je izvijestio o dosadašnjem radu:
    - **Jakov:** Učio React i Next.js (uključujući React hooks i Chakra UI).
    - **Karlo Brzak:** Izradio početne UML dijagrame, te učio backend.
    - **Vilim:** Izradio shemu baze podataka, opis baze te dodatno poboljšao strukturu.
    - **Ivan Dujmić:** Popravio dizajn u Figmi i uveo GitHub Project za tim.
    - **Fran Galić:** Razvio GitHub workflow, dopunio dokumentaciju i organizirao sastanak.
    - **Matija:** Radio na dokumentaciji.
    - **Ivan Džanija:** Učio backend.
- **Zajednički pregled tema i objašnjenja:**
  - **Fran:** Prikazao gdje je napisao wiki i Project Workflow na GitHubu, objasnio svrhu i način korištenja.
  - **Ivan:** Prikazao izmjene na Figma dizajnu i predstavio GitHub Project za upravljanje zadacima.
  - **Vilim:** Objasnio dizajn baze podataka i funkcionalnosti koje podržava. Diskutirano je treba li dodati još poboljšanja ili izmjena.
- **Upoznavanje tima s krajnim rokom za 1. ciklus i očekivanjima projekta:**
  - **Rok za predaju:** Petak, 15. 11. 2024.
  - **Zadaci do tada:**

    * Dovršiti 1., 2. i 4. poglavlje dokumentacije s UML dijagramima.
    * Izraditi 5. poglavlje dokumentacije - razrada arhitekture sustava.
    * Implementirati osnovne funkcionalnosti:
        1. Login i register (za korisnika).
        2. Prikaz različitih automobila za korisnika i gosta (diskusija oko ovog prijedloga).
    * Postaviti početnu verziju stranice na deployment platformu (dogovoriti tko će se time baviti).
    * Prokomentirati podjelu zadataka i organizirati rad (tko radi što i do kada).

- **Diskusija:**
  - **Pitanja za raspravu:**
    * Kako ćemo povezati bazu podataka s frontendom i backendom?
    * Trebamo li bazu unaprijed popuniti početnim podacima?
    * Koji su koraci za deployment cijelog sustava?
    * Tko će se baviti deploymentom i drugim povezanim zadacima?
    * Ima li još nešto važno što trebamo uzeti u obzir?
- **Definiranje konačnih zadataka do sljedećeg sastanka:**
  - **Matija:** Dovršiti poglavlja 1., 2. i 4. dokumentacije. Sve mora biti precizno i detaljno razrađeno.
  - **Fran i Jakov:** Izraditi front-end za Home Page i stranice za prijavu i registraciju te uskladiti projektne pakete kako bi svi koristili iste biblioteke i alate.
  - **Backend tim (Ivan Džanija, Ivan Dujmić, Karlo Brzak):** Implementirati funkcionalnosti za login i register koristeći OAuth2.0, omogućiti dohvat podataka o vozilima za front-end te uspostaviti projekt s usklađenim paketima i dev alatima. Također istražiti mogućnosti povezivanja s bazom podataka.
  - **Vilim:** Dovršiti bazu podataka, istražiti načine za punjenje baze početnim podacima, istražiti opcije za deployment baze i dopuniti sigurnosne postavke.

4. sastanak

- **Datum i vrijeme**: 5. studenoga 2024., 13:00
- **Mjesto**: FER
- **Zapisničar**: Fran Galić

Agenda:

1. **Uvod**
    - Predstavljanje dnevnog reda sastanka.
    - Odabir zapisničara.

2. **Ključne točke do petka – Rasprava i dogovor**
   - **Finalni zadaci koji moraju biti dovršeni do petka**:
     - **Dokumentacija**: Dovršiti poglavlja 1., 2. i 4. do sljedećeg sastanka (zadužen: Matija).
     - **Frontend - Register i Login stranica**: Završiti registraciju i prijavu korisnika (zadužen: Jakov).
     - **Backend - API endpointi**: Izraditi API za registraciju, prijavu, popis "Top Rated" i "Best Value" automobila (zadužen: Backend tim).
     - **Baza podataka**: Unijeti početne podatke (zadužen: Backend tim).
     - **Povezivanje frontenda i backenda**: Usuglasiti elemente frontenda i backenda te po potrebi prilagoditi sadržaj (zaduženi: Fran i Jakov - frontend; netko iz backend tima - backend).
     - **Čišćenje i organizacija koda**: Organizirati i očistiti kod za bolju preglednost (zaduženi: svi članovi tima).
     - **Deployment**: Postaviti početnu verziju stranice u produkciju (zadužen: član backend tima).
     - **Opis arhitekture sustava**: Dokumentirati arhitekturu sustava (zadužen: član backend tima).
     - **Shema baze podataka**: Ažurirati shemu baze i dokumentaciju na wiki stranici (zadužen: Vilim).
     - **Priprema prezentacije za petak**: Fran i svaki član tima priprema prezentaciju svojih dijelova projekta.
3. **Razvoj aplikacije**
   - **Frontend**:
     - Završiti cijeli frontend:
       - Home page za registriranog korisnika.
       - Stranica za gosta.
       - Defaultna stranica za autokuću.
       - Stranice za registraciju i prijavu korisnika.
   - **Backend**:
     - Osigurati API točke za registraciju i login.
     - Implementirati OAuth2.0 za prijavu putem Google-a.
     - Definirati zahtjeve i odgovor tijekom autentifikacije.
     - Implementirati API-je za dohvat podataka "Top Rated" i "Best Value" automobila za frontend.
   - **Povezivanje frontend-a s backend-om**:
     - Omogućiti razmjenu podataka i prilagoditi elemente po potrebi.
   - **Baza podataka**:
     - Dodati inicijalne podatke u bazu (npr. automobili).
     - Ažurirati shemu baze i dokumentaciju.
   - **Deployment**:
     - Deployati frontend, backend i bazu podataka.

4. **Dokumentacija**
   - **Poglavlja 1., 2. i 4.**:
     - Detaljno razraditi i dodati dijagrame.
   - **Arhitektura sustava (Poglavlje 5)**:
     - Prilagoditi shemu baze i ažurirati opise.
5. **Završni zadaci i priprema za demonstraciju**
   - Razraditi prezentaciju stranice i definirati uloge svakog člana.
   - Razmisliti o prikazu funkcionalnosti i modula asistentu.

5. sastanak

- **Datum i vrijeme**: 12. studenoga 2024., 13:00
- **Mjesto**: FER
- **Zapisničar**: Fran Galić

Agenda i zapisnik:

1. **Uvod**
   - Sastanak je trajao kraće nego obično. Glavna tema bila je pregled napretka i definiranje daljnjih koraka.
2. **Pregled napretka**
   - Prošli smo što je sve do sada napravljeno i koji su dijelovi još uvijek u radu.
3. **Zadaci i dogovori**

   Backend tim:
   - Rješavanje postojećih bugova vezanih za sigurnost.
   - Osigurati da login i registracija rade ispravno.
   - Podaci se moraju slati frontend timu na odgovarajući način.

   Frontend tim:
   - Popraviti i očistiti dosadašnji kod.
   - Razlomiti veće dijelove koda u komponente radi bolje organizacije.
   - Zajedno s backend timom raditi na ispravnom funkcioniranju login i register funkcionalnosti.
   - Urediti pristup korisnicima tako da svaki korisnik može pristupiti samo stranicama za koje ima odgovarajuće ovlasti.

   Promjene u zaduženjima:
   - Vilim Hrupelj prelazi iz baze podataka na frontend tim.
   - Kukić, uz dosadašnje zadatke vezane za dokumentaciju, započinje s upoznavanjem backend sustava i pomagati će backend timu.

4. **Dogovor za sljedeći tjedan**
    - Početak intenzivnijeg rada na projektima.
    - Na sljedećem sastanku definirati sve preostale zadatke i detaljno planirati daljnje korake.

Tablica aktivnosti

Upute za korištenje tablice aktivnosti

- Svaka aktivnost je navedena u lijevom stupcu.
- Članovi tima su navedeni u zaglavlju kolona.
- U odgovarajuću ćeliju upisujete **broj sati** koji ste uložili u rad na određenoj aktivnosti.
- Ako niste radili na nekoj aktivnosti, ostavite ćeliju praznom.
- Tablicu redovito ažurirajte i dodajte nove aktivnosti po potrebi.
- Nakon što popunite tablicu, spremite promjene i pushajte ih na GitHub wiki.

Kontinuirano osvježavanje

Aktivnost	Fran Galić	Jakov Biloglav	Karlo Brzak	Ivan Dujmić	Ivan Džanija	Vilim Hrupelj	Matija Kukić
Upravljanje projektom	20			4			
Opis projektnog zadatka	4						5
Funkcionalni zahtjevi	2						8
Opis pojedinih obrazaca			0.5				10
Dijagram obrazaca			3				3
Sekvencijski dijagrami			5				2
Opis ostalih zahtjeva							
Arhitektura i dizajn sustava	8	6	1	28			
Baza podataka				20		11	3
Dijagram razreda			5			3	
Dijagram stanja					5		
Dijagram aktivnosti			2				

Aktivnost	Fran Galić	Jakov Biloglav	Karlo Brzak	Ivan Dujmić	Ivan Džanija	Vilim Hrupelj	Matija Kukić
Dijagram komponenti				3		3	
Korištene tehnologije i alati	1	5	4	6	5		3
Ispitivanje programskog rješenja		3	5	20	25		10
Dijagram razmještaja				1			
Upute za puštanje u pogon				2			1
Dnevnik sastajanja	2			2			
Zaključak i budući rad	2						
Popis literature				1			
Izrada početne stranice	6	3					
Izrada baze podataka				2		8	3
Spajanje s bazom podataka			5	2	10	5	
Backend			70	90	100		70
Frontend	120	10		2		42	

Dijagrami pregleda promjena

U nastavku su prikazani dijagrami preuzeti iz GitHub Contributors stranice.

Ključni izazovi i rješenja

Zaključno

Tijekom izrade projekta, tim se suočio s nizom izazova koji su utjecali na dinamiku razvoja i organizaciju rada. Unatoč poteškoćama, kroz zajednički trud i prilagodbu strategija, projekt je uspješno doveden do završne faze.

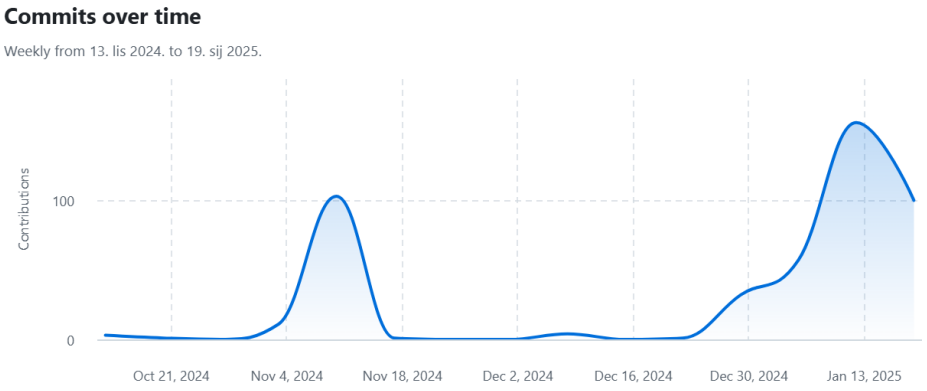


Figure 3: Commits Over Time

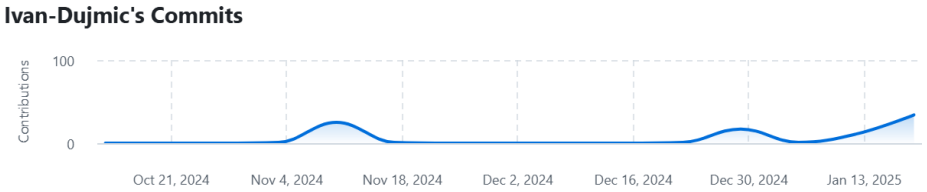


Figure 4: Ivan-Dujmic's Commits

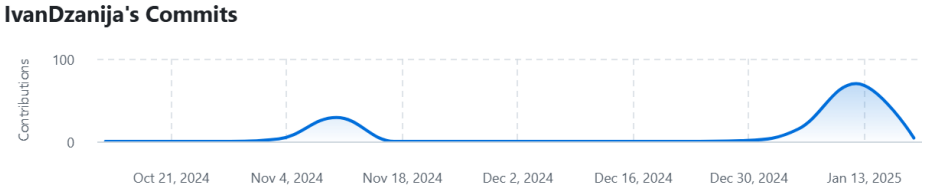


Figure 5: IvanDzanija's Commits

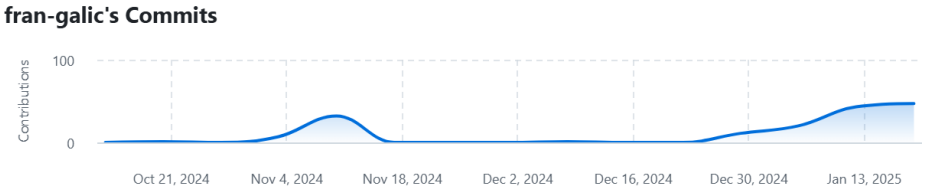


Figure 6: fran-galic's Commits

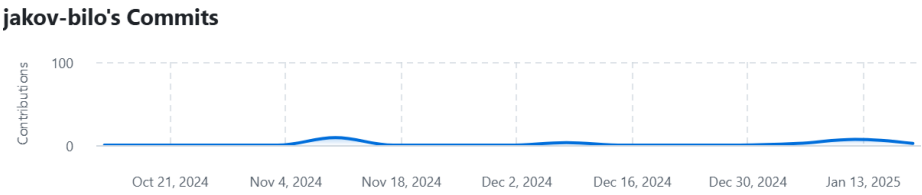


Figure 7: jakov-bilo's Commits

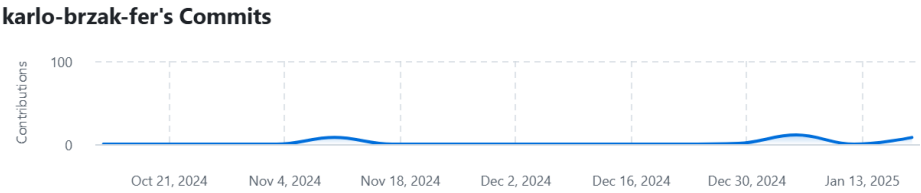


Figure 8: karlo-brzak-fer's Commits

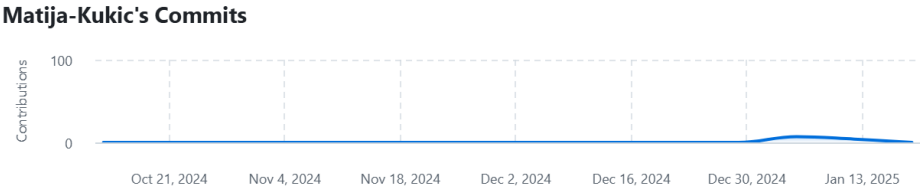


Figure 9: Matija-Kukic's Commits

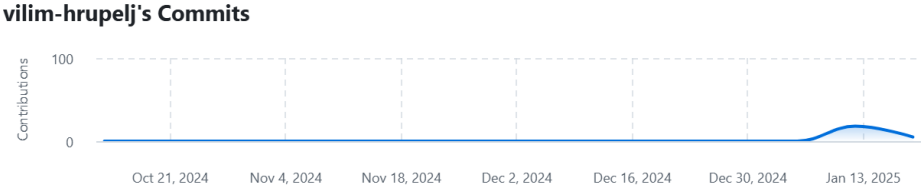


Figure 10: vilim-hrupelj's Commits

Opis izazova

Glavni izazovi koji su prepoznati tijekom rada na projektu uključuju:

- **Kašnjenje u razvoju** – Nedostatak iskustva u radu s novim tehnologijama poput **Reacta, Next.jsa** i **Djangoa** rezultirao je sporijim napretkom. Paralelno izvođenje drugih akademskih obaveza također je otežalo usredotočenost isključivo na ovaj projekt. Broj svojstava koji smo si odredili je bio prevelik za količinu vremena i veličinu tima.
- **Tehnički problemi** – Tim se susreo s poteškoćama prilikom povezivanja frontenda i backenda, konfiguracije cloud deploymenta te integracije vanjskih servisa kao što su **Google Maps API** i **Stripe** sustav plaćanja.
- **Timska organizacija** – Koordinacija rada i jasno definiranje zadataka predstavljali su izazov, pri čemu je bilo potrebno uskladiti rad svih članova kako bi se izbjegli konflikti u radu na različitim dijelovima sustava.
- **Dokumentacija** – Osiguravanje kvalitetne i jasne dokumentacije bilo je ključno za razumijevanje projekta, kako unutar tima, tako i za buduće sudionike projekta.
- **Motivacija tima** – Održavanje radne discipline i timskog duha bilo je ključno, ali zahtjevno u periodima kada su se pojavili problemi ili nepredviđene poteškoće.

Rješenja i naučene lekcije

Kako bi se prevladali ovi izazovi, primijenjena su sljedeća rješenja:

- **Jasno definiranje plana rada** – Kao voditelj tima, preuzeo sam odgovornost za postavljanje jasne vizije i smjera razvoja kako bi svaki član imao jasan uvid u konačni cilj projekta.
- **Redovita komunikacija i nadzor** – Uspostavljeni su redoviti sastanci i praćenje napretka kroz česte provjere statusa zadataka, što je omogućilo pravovremeno prepoznavanje problema i njihovo rješavanje.
- **Tehnička rješenja** – Problemi s deploymentom riješeni su korištenjem **Vercela** za frontend i **Rendera** za backend. Izazovi vezani uz integraciju vanjskih servisa riješeni su proučavanjem službene dokumentacije i testiranjem.
- **Bolja raspodjela zadataka** – Iako je bilo prostora za poboljšanja u upravljanju zadacima, zadaci su s vremenom bolje usklađeni prema znanjima i vještinama pojedinih članova.
- **Timski duh i motivacija** – Kao voditelj projekta, shvatio sam da je jedan od ključnih zadataka motivirati tim i stvoriti radnu atmosferu u kojoj će se svi osjećati uključeno i imati jasnu sliku o svojoj ulozi u projektu.
- **Kvalitetniji kod** – Iako nije bilo dovoljno vremena za optimizaciju u potpunosti, nastojalo se održati modularnost i čitljivost koda kako bi bio što jednostavniji za buduće nadogradnje.
- **Određivanje zadatka** - Bolje odrediti očekivana svojstva projekta prema predviđenom vremenu i prethodnim znanjima tima.

Zaključak

Projekt je omogućio stjecanje novih znanja i iskustava, kako tehničkih tako i organizacijskih. Tim je kroz rad usvojio važnost dobre komunikacije, jasnog definiranja ciljeva i pravovremenog rješavanja problema.

Ako bi se projekt razvijao za stvarnu industrijsku upotrebu, bilo bi potrebno:

- Optimizirati performanse i sigurnost aplikacije,
- Poboljšati modularnost i fleksibilnost koda,
- Razraditi naprednije funkcionalnosti koje bi aplikaciju učinile konkurentnijom na tržištu.

Funkcionalnost koja nije implementirana u aplikaciji je **sustav komunikacije između korisnika i kompanija (chat)**, koji zbog vremenskih ograničenja nije bio moguće realizirati.

Kao voditelj projekta, smatram da smo ostvarili zadane ciljeve u skladu s fakultetskim zahtjevima te da je tim pokazao iznimnu sposobnost u prilagodbi i učenju. Iako bi daljnji razvoj projekta zahtijevao dodatne resurse i angažman, trenutačni rezultat predstavlja solidnu osnovu za buduće nadogradnje. Timski rad i međusobna podrška pokazali su se ključnima za uspješan završetak projekta.