

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Auto Servis

Tim: <TG05.3>

REST-API

Nastavnik: Vlado Sruk

1.1 Uvod

Projekt **Aplikacija za auto servis** razvija se u sklopu kolegija *Programsko inženjerstvo* s ciljem digitalizacije poslovanja i komunikacije između korisnika (vlasnika vozila) i servisera.

Tradicionalno, proces prijave vozila na servis uključuje telefonske pozive, ručne evidencije i nejasne rokove, što dovodi do čestih nesporazuma, kašnjenja i neučinkovitosti.

Ovaj projekt predviđa razvoj web aplikacije koja omogućuje jednostavnu prijavu, pregled statusa popravka i elektroničku komunikaciju između korisnika i osoblja servisa.

1.2 Cilj projekta

Glavni cilj projekta je **automatizirati i pojednostaviti proces prijave i obrade vozila u autoservisima**.

Sustav će omogućiti:

- prijavu vozila putem internetskog sučelja
- odabir termina servisa
- automatsko obavješćavanje korisnika o promjenama statusa prijave
- administrativno praćenje podataka o vozilima, terminima i servisima

Time se postiže transparentnost procesa i smanjuje potreba za ručnom komunikacijom i papirologijom.

1.3 Problematika postojećeg sustava

Trenutni način poslovanja u većini autoservisa temelji se na:

- ručnoj evidenciji vozila
- telefonskim dogovorima s korisnicima
- nedostatku uvida u zauzetost servisera
- nepostojanju sustava za automatske obavijesti

Ovakav sustav uzrokuje česte pogreške, preklapanje termina, gubitak informacija i nezadovoljstvo korisnika. Osim toga, korisnici nemaju mogućnost praćenja statusa servisa u stvarnom vremenu niti elektroničke evidencije svojih prijava.

1.4 Opis predloženog rješenja

Predloženo rješenje je centralizirana web aplikacija koja povezuje sve sudionike – korisnike, servisere i administratore.

Sustav omogućuje:

- **registraciju i autentifikaciju korisnika** putem OAuth2 (Google prijava)
 - **unos i uređivanje podataka o vozilu**
 - **prijavu vozila na servis i odabir termina**
 - **automatsko obavješćavanje e-mailom** o promjenama statusa prijave
 - **uvid u povijest servisa** i mogućnost preuzimanja izvještaja
 - **administratorima** generiranje izvještaja (PDF, XML, XLSX) i praćenje statistike servisa
-

1.5 Potencijalna korist projekta

Implementacija ovakvog sustava donosi višestruke koristi:

Korisnik	Dobiva
Vlasnik vozila	Brzu prijavu i uvid u status servisa bez potrebe za pozivima.
Serviser	Lakšu organizaciju posla, pregled svih prijava i termina, mogućnost dodavanja napomena.
Administrator / vlasnik servisa	Digitalne evidencije, praćenje opterećenosti serviseri i generiranje izvještaja.
Cjelokupna organizacija	Veća učinkovitost, manje pogrešaka i veće zadovoljstvo korisnika.

Projekt donosi i širu društvenu korist – modernizaciju poslovanja i smanjenje nepotrebne papirologije u autoindustriji.

1.6 Postojeća slična rješenja

Na tržištu postoje aplikacije koje korisnicima omogućuju praćenje održavanja vozila i servisnih aktivnosti. Primjeri takvih rješenja su:

Naziv rješenja	Opis	Razlika u odnosu na ovaj projekt
CARFAX Car Care	Aplikacija za praćenje povijesti vozila, servisnih intervala i podsjetnika na održavanje.	Namijenjena samo korisnicima; nema prijavu vozila na servis ni ulogu serviseri.
Drivvo	Aplikacija za vođenje evidencije troškova, goriva i održavanja više vozila.	Ne podržava rezervacije termina ni komunikaciju s autoservisom.

1.7 Skup korisnika

Aplikaciju bi koristili:

- **privatni korisnici** – vlasnici osobnih automobila koji žele jednostavno prijaviti vozilo na servis
- **serviseri i tehničari** – koji vode i obrađuju prijave, planiraju radni raspored
- **administratori / vlasnici servisa** – za nadzor rada, izvještavanje i analitiku

Potencijalno bi se sustav mogao prilagoditi i za:

- **autopraonice**
- **vulkanizerske radionice**
- **sisteme najma vozila** – gdje bi se koristila funkcionalnost rezervacije zamjenskog vozila

1.8 Mogućnost prilagodbe i proširenja

Zbog modularne arhitekture sustava, moguće su sljedeće nadogradnje:

- dodavanje **notifikacija putem SMS-a ili mobilne aplikacije**
 - integracija s **vanjskim ERP sustavima** za fakturiranje
 - statističke analize i praćenje učinkovitosti serviser
 - proširenje sustava na **mobilnu aplikaciju (Android/iOS)**
-

1.9 Opseg projektnog zadatka

Trenutna verzija projekta uključuje sljedeće funkcionalnosti:

- prijava i autentifikacija korisnika (OAuth2)
 - unos i upravljanje vozilima
 - prijava vozila na servis i upravljanje terminima
 - ažuriranje statusa servisa i dodavanje napomena
 - automatsko obavješćavanje korisnika putem e-maila
 - generiranje i preuzimanje izvještaja
-

1.10 Moguće nadogradnje

Projekt se može proširiti u sljedećim smjerovima:

- **mobilna aplikacija** za korisnike i servisere
- **sustav za ocjenjivanje** kvalitete servisa i serviser
- **automatsko prepoznavanje vozila** putem registarske oznake
- **AI prediktivna analiza** za planiranje kapaciteta servisa
- **integracija s IoT uređajima** (npr. pametni senzori vozila za dijagnostiku)

U ovom poglavlju definirani su svi **funkcionalni** i **nefunkcionalni zahtjevi** sustava *Aplikacija za auto servis*, kao i **dionici (akteri)** koji sudjeluju u njegovu radu.

Ova analiza predstavlja temelj za daljnju specifikaciju sustava i izradu obrazaca uporabe.

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriterij prihvatanja
F-001	Sustav omogućuje korisniku prijavu u sustav.	Visok	Zahtjev dionika	Korisnik se može prijaviti i pristupiti svom profilu.
F-002	Sustav omogućuje korisniku dodavanje vozila (marka, model, registracija, godina).	Visok	Zahtjev dionika	Vozilo se sprema u bazu i vidljivo je korisniku.
F-003	Korisnik može prijaviti vozilo na servis te odabrati termin.	Visok	Zahtjev dionika	Prijava se pohranjuje i korisnik dobiva potvrdu e-mailom.
F-004	Sustav omogućuje korisniku pregled statusa prijave.	Srednji	Zahtjev dionika	Korisnik vidi trenutni status svake svoje prijave.
F-005	Serviser može pregledati zaprimljene prijave.	Visok	Zahtjev dionika	Prijave su vidljive u listi i filtrirane po statusu.
F-006	Serviser može ažurirati status prijave (zaprimljeno, u obradi, završeno, odgođeno).	Visok	Zahtjev dionika	Status se promijeni i korisniku se automatski pošalje e-mail.
F-007	Serviser može dodati napomenu na prijavu.	Srednji	Zahtjev dionika	Napomena se prikazuje unutar detalja prijave.
F-008	Sustav omogućuje korisniku rezervaciju zamjenskog vozila.	Srednji	Povratne informacije korisnika	Zamjensko vozilo se rezervira i označava kao nedostupno u sustavu.
F-009	Administrator može generirati izvještaje o servisima u PDF, XML ili XLSX formatu.	Srednji	Dokument projektnog zadatka	Izveštaj se uspješno generira i može se preuzeti.
F-010	Sustav omogućuje korisnicima pregled osnovnih informacija o servisu.	Srednji	Zahtjev dionika	Svi korisnici mogu otvoriti javnu stranicu s prikazom informacija o servisu.
F-011	Sustav generira i pohranjuje PDF obrazac pri predaji i preuzimanju vozila (s potpisima korisnika i serviseru).	Visok	Dokument projektnog zadatka	Prilikom predaje ili preuzimanja vozila generira se PDF s podacima o korisniku, vozilu i serviseru.
F-012	Sustav automatski šalje podsjetnik korisniku ako je termin odgođen za više od 3 dana.	Srednji	Dokument projektnog zadatka	Nakon promjene termina s pomakom većim od 3 dana, korisnik automatski prima e-mail obavijest.

ID zahtjeva	Opis	Prioritet	Izvor	Kriterij prihvatanja
F-013	Administrator može upravljati podacima o servisima (naziv, lokacija, radno vrijeme).	Srednji	Zahtjev dionika	Promjene se pohranjuju i odmah vidljive na korisničkom sučelju.
F-014	Administrator može dodavati, uređivati i brisati serviser te ih povezivati s određenim servisima.	Visok	Zahtjev dionika	Novi serviseri se uspješno dodaju i prikazuju u sustavu.
F-015	Administrator može pregledavati i deaktivirati korisničke račune.	Srednji	Zahtjev dionika	Korisnik se nakon deaktivacije više ne može prijaviti.

Nefunkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-001	Sustav mora koristiti siguran OAuth2 protokol za autentikaciju korisnika.	Visok
NF-002	Sustav mora osigurati komunikaciju putem HTTPS protokola.	Visok
NF-003	Sustav mora omogućiti odziv stranice unutar 2 sekunde.	Srednji
NF-004	Sustav mora omogućiti rad na mobilnim uređajima (responsivni dizajn).	Srednji
NF-005	Sustav mora biti implementiran s relacijskom bazom podataka (PostgreSQL).	Visok
NF-006	Sustav mora imati jednostavno održavanje i dobro dokumentiran kôd.	Srednji
NF-007	Sustav mora automatski slati e-mail obavijesti korisnicima kod promjena statusa prijave i odgoda termina duljih od tri dana.	Visok
NF-008	Sustav mora biti dostupan 24/7 uz pouzdanost veću od 99%.	Srednji
NF-009	Sustav mora omogućiti prikaz lokacije auto servisa korištenjem Google Maps servisa.	Srednji

Dionici i akteri

U sustavu *Aplikacija za auto servis* sudjeluju sljedeći dionici (akteri):

Oznaka	Naziv aktera	Uloga	Funkcionalnosti (povezani zahtjevi)
A-1	Neregistrirani korisnik	Inicijator	Može pregledati osnovne informacije o servisu (F-010) te se registrirati putem OAuth2 (F-001).

Oznaka	Naziv aktera	Uloga	Funkcionalnosti (povezani zahtjevi)
A-2	Registrirani korisnik	Inicijator	Može dodavati vozila (F-002), prijaviti vozilo na servis i odabrati termin (F-003), pregledavati statuse prijave i povijest servisa (F-004), rezervirati zamjensko vozilo (F-008) te preuzeti PDF obrazac pri predaji ili preuzimanju vozila (F-011).
A-3	Serviser	Sudionik	Može pregledavati zaprimljene prijave (F-005), ažurirati statuse prijave (F-006), dodavati napomene (F-007) te sudjelovati u generiranju PDF obrazaca (F-011).
A-4	Administrator	Sudionik	Može upravljati podacima o servisima (F-013), serviserima (F-014), korisnicima (F-015) te generirati i izvoziti izvještaje (F-009).
A-5	E-mail sustav	Vanjski sustav	Automatski šalje potvrde i obavijesti korisnicima (F-003, F-006, F-012, NF-007).
A-6	OAuth2 servis (Google)	Vanjski servis	Omogućuje autentikaciju korisnika (F-001, NF-001).
A-7	Google Maps servis	Vanjski servis	Omogućuje prikaz lokacije servisa u sklopu javne stranice s informacijama (F-010, NF-009).

Zaključak

Definirani funkcionalni i nefunkcionalni zahtjevi predstavljaju temelj za daljnju izradu specifikacije sustava. Njima su jasno obuhvaćene sve osnovne i napredne funkcionalnosti aplikacije — od prijave korisnika, prijave vozila i upravljanja servisima do generiranja izvještaja i automatskog obavješćavanja korisnika.

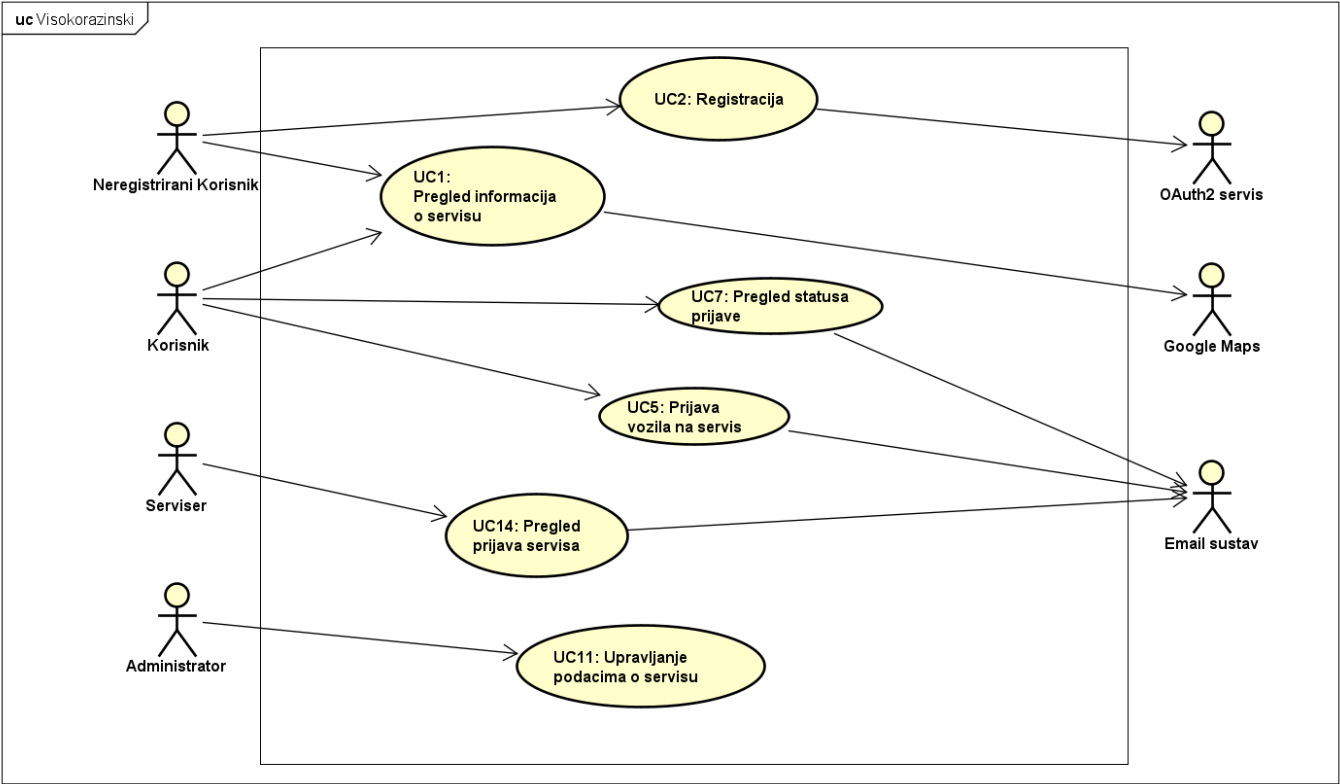
U ovom poglavlju prikazani su obrasci uporabe (Use Case) i njihovi opisi, koji detaljno prikazuju način na koji korisnici, serviseri, administratori i vanjski sustavi međusobno komuniciraju sa sustavom.

Napomena: Use-case dijagrami ne prikazuju internu arhitekturu sustava, no svi obrasci uporabe komuniciraju s bazom podataka radi spremanja i dohvaćanja podataka.

3.1 Dijagrami obrazaca uporabe

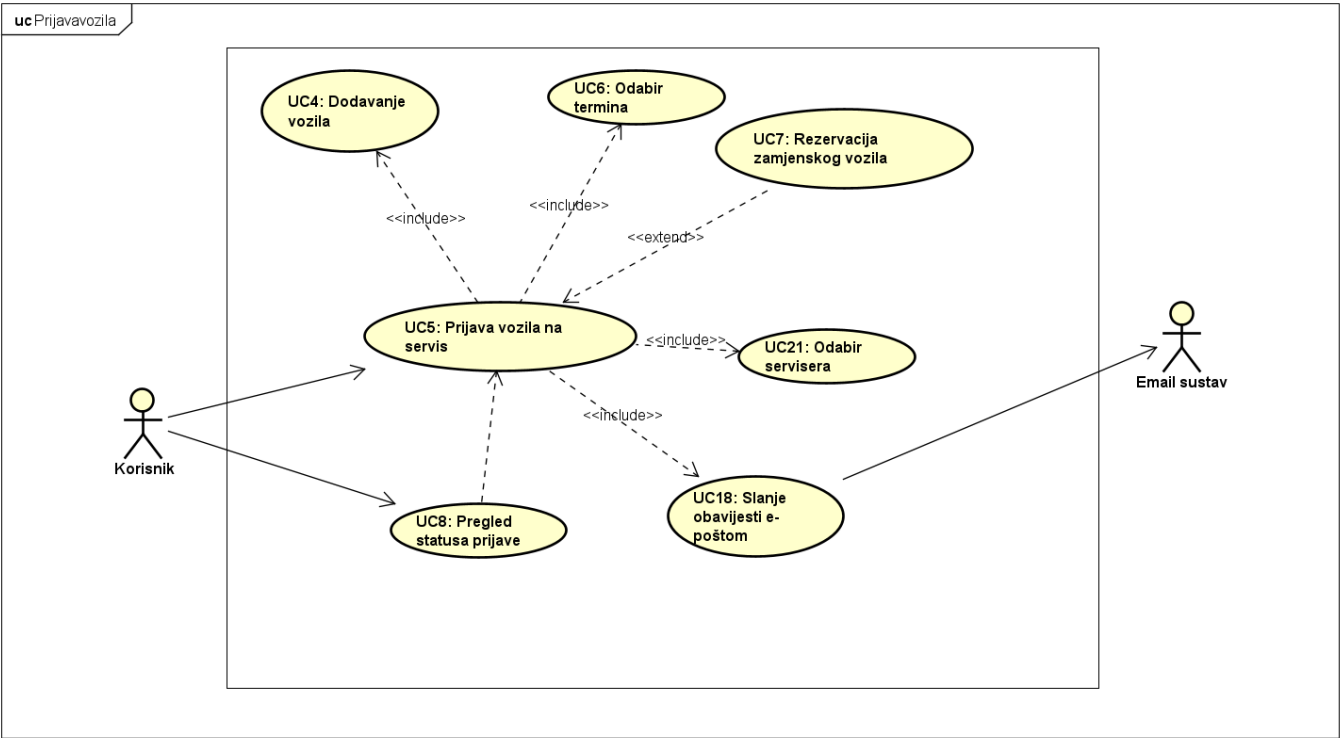
Visokorazinski dijagram

Prikazuje najvažnije funkcionalnosti sustava i sve glavne aktere na jednoj visokoj razini.



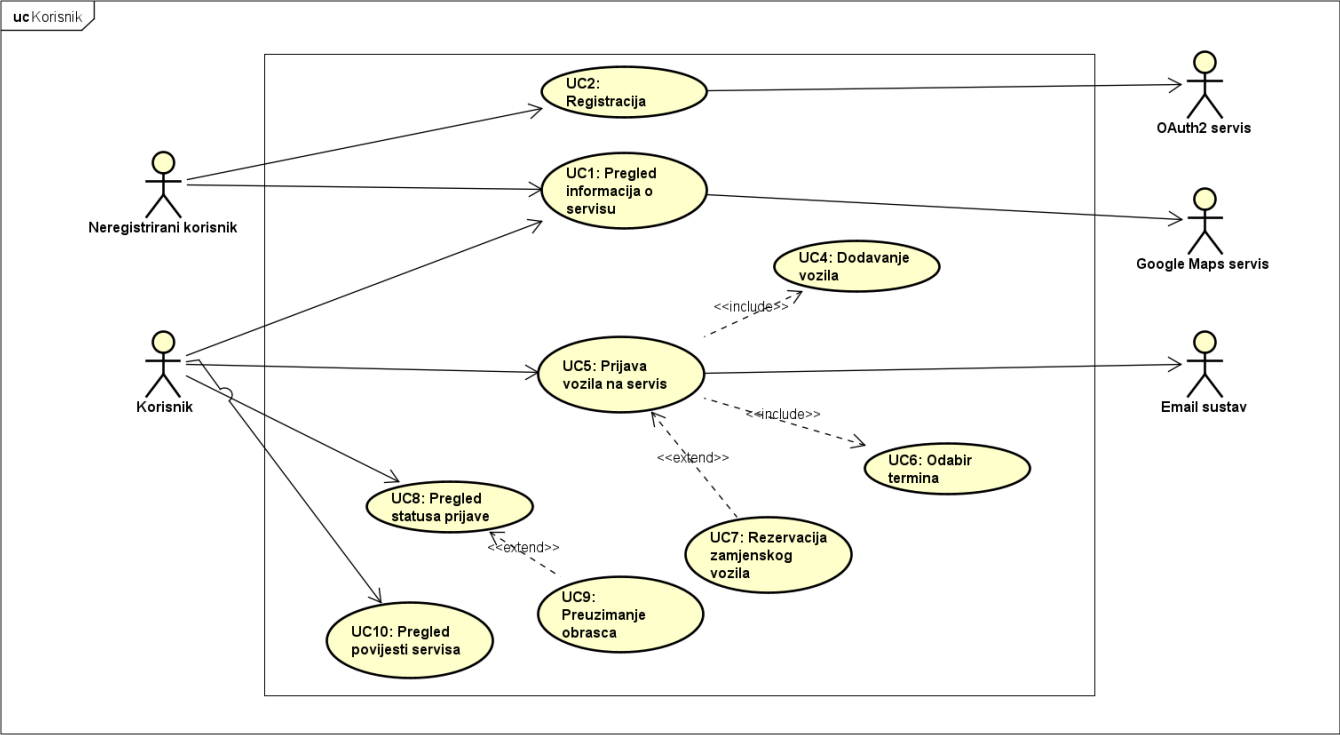
Dijagram ključne značajke – Prijava vozila

Prikazuje detaljan tijek prijave vozila i sve povezane obrasce uporabe.



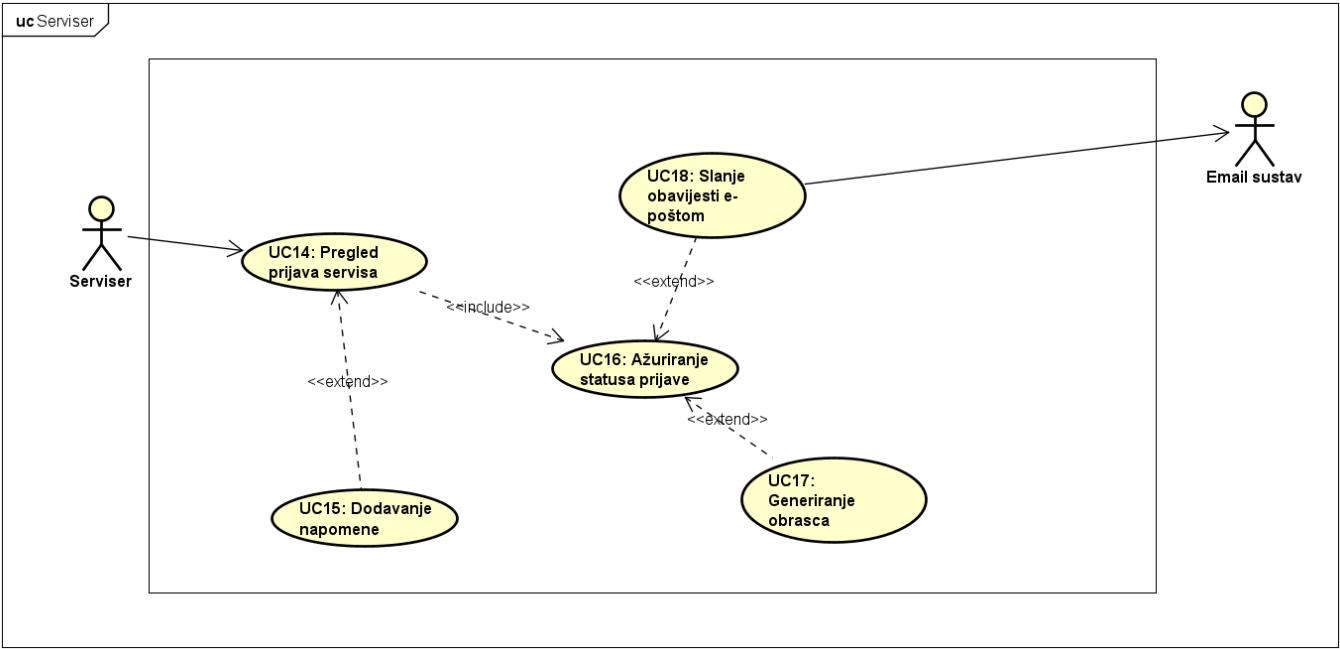
Dijagram po ulozi – Korisnik

Prikazuje sve funkcionalnosti dostupne registriranom i neregistriranom korisniku.



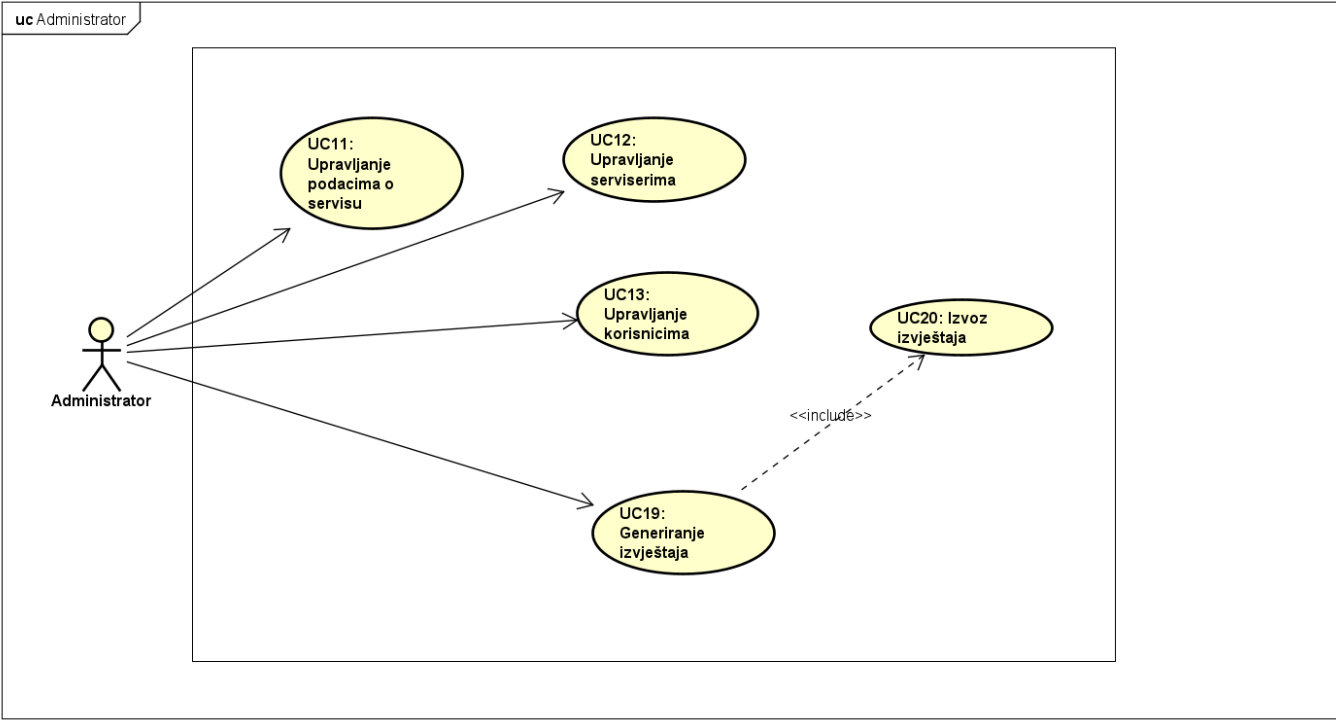
Dijagram po ulozi – Serviser

Prikazuje funkcionalnosti serviseru u procesu obrade prijave.



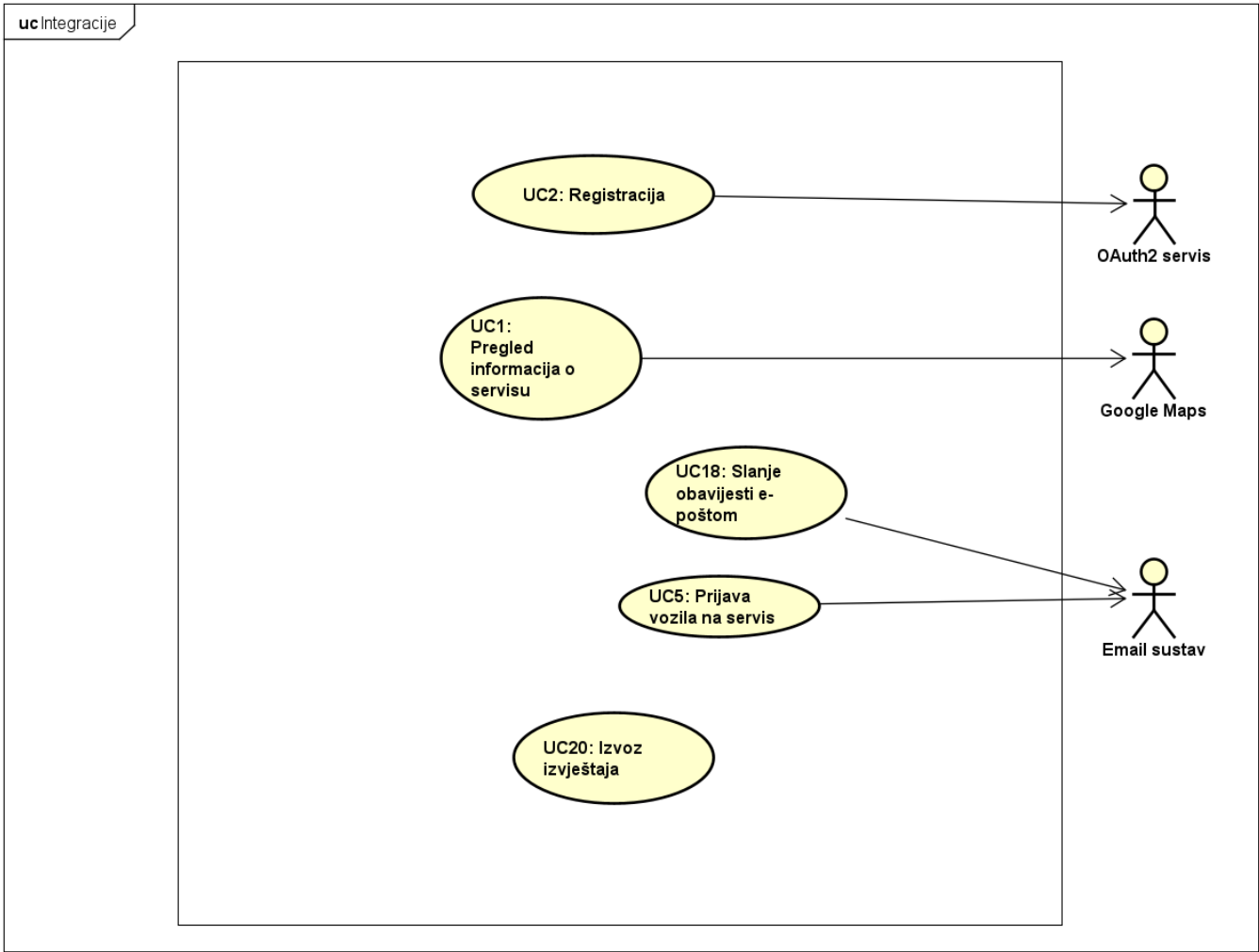
Dijagram po ulozi – Administrator

Prikazuje administrativne funkcionalnosti i upravljanje sustavom.



Dijagram integracija

Prikazuje interakcije sustava s vanjskim servisima poput OAuth2, e-mail sustava i Google Maps.



3.2 Opis obrazaca uporabe

U nastavku su prikazani detaljni opisi svih obrazaca uporabe.

Svaki opis sadrži glavnog sudionika, cilj, sudionike, preduvjete, osnovni tijek, moguća odstupanja i poveznice na funkcionalne zahtjeve (F-ID).

UC1 – Pregled osnovnih informacija o servisu

Glavni sudionik: Neregistrirani korisnik

Cilj: Pregled osnovnih informacija o servisu (radno vrijeme, lokacija, kontakt).

Sudionici: Google Maps servis

Preduvjet: Nema (korisnik nije prijavljen).

Osnovni tijek:

1. Korisnik odabire opciju „O servisu“. (F-011)
2. Sustav prikazuje osnovne informacije o servisu.
3. Sustav prikazuje točnu lokaciju servisa putem Google Maps servisa. (NF-009)

Odstupanja:

- Neuspješno dohvaćanje podataka s Google Mapsa → prikazuje se poruka o pogrešci.
-

UC2 – Registracija korisnika putem OAuth2

Glavni sudionik: Neregistrirani korisnik

Cilj: Stvaranje korisničkog računa pomoću vanjskog OAuth2 servisa.

Sudionici: OAuth2 servis

Preduvjet: Korisnik nije registriran.

Osnovni tijek:

1. Korisnik odabire opciju „Registracija“. (F-001)
2. Sustav preusmjerava korisnika na OAuth2 servis (Google).
3. Korisnik se autentificira i autorizira pristup.
4. Sustav pohranjuje korisničke podatke i potvrđuje registraciju.

Odstupanja:

- Autentifikacija neuspješna → prikazuje se poruka o neuspjeloj registraciji.
-

UC4 – Dodavanje vozila

Glavni sudionik: Registrirani korisnik

Cilj: Dodavanje novog vozila u sustav.

Preduvjet: Korisnik mora biti prijavljen.

Osnovni tijek:

1. Korisnik odabire „Dodaj vozilo“. (F-003)
2. Unosi podatke (marka, model, registracija, godina).

3. Sustav provjerava jedinstvenost registracije.
4. Sustav sprema vozilo u bazu i prikazuje potvrdu.

Odstupanja:

- Unesena registracija već postoji → prikazuje se poruka o pogrešci.
-

UC5 – Prijava vozila na servis

Glavni sudionik: Registrirani korisnik

Cilj: Prijaviti vozilo na servis i odabrati termin.

Sudionici: E-mail sustav

Preduvjet: Korisnik mora imati registrirano vozilo.

Osnovni tijek:

1. Korisnik odabire „Prijava vozila”. (F-004)
2. Sustav prikazuje popis vozila korisnika.
3. Korisnik odabire termin (UC6).
4. Po potrebi rezervira zamjensko vozilo (UC7).
5. Sustav sprema prijavu i šalje potvrdu e-mailom. (F-004, NF-007)

Odstupanja:

- Nema slobodnih termina → sustav nudi druge datume.
 - Nema dostupnih zamjenskih vozila → prikazuje se obavijest.
-

UC6 – Odabir termina

Glavni sudionik: Korisnik

Cilj: Odabrati slobodan termin za servis.

Preduvjet: U tijeku je prijava vozila (UC5).

Osnovni tijek:

1. Sustav prikazuje sve slobodne termine. (F-004)
2. Korisnik odabire željeni termin.
3. Sustav sprema termin u prijavu.

Odstupanja:

- Termin više nije dostupan → sustav nudi drugi.
-

UC7 – Rezervacija zamjenskog vozila

Glavni sudionik: Korisnik

Cilj: Rezervirati zamjensko vozilo tijekom servisa.

Preduvjet: U tijeku je prijava vozila (UC5).

Osnovni tijek:

1. Korisnik odabire opciju „Rezerviraj zamjensko vozilo”. (F-009)
2. Sustav prikazuje dostupna zamjenska vozila.

3. Korisnik odabire vozilo i potvrđuje rezervaciju.

Odstupanja:

- Nema dostupnih vozila → prikazuje se poruka.
-

UC8 – Pregled statusa prijave

Glavni sudionik: Korisnik

Cilj: Pregled trenutnog statusa svih prijava.

Preduvjet: Korisnik mora imati barem jednu prijavu.

Osnovni tijek:

1. Korisnik otvara „Moje prijave“. (F-005)
 2. Sustav prikazuje statute (zaprimitljeno, u obradi, završeno, odgođeno).
 3. Ako je status „završeno“, korisnik može preuzeti PDF (UC9).
-

UC9 – Preuzimanje obrasca (PDF)

Glavni sudionik: Korisnik

Cilj: Preuzeti PDF obrazac o predaji ili preuzimanju vozila.

Preduvjet: Servis mora biti završen.

Osnovni tijek:

1. Korisnik odabire završenu prijavu. (F-012)
2. Sustav dohvaća već generirani PDF iz prethodnog procesa (UC17).
3. Korisnik preuzima dokument.

Odstupanja:

- Greška pri generiranju → prikazuje se poruka o neuspjelom preuzimanju.
-

UC10 – Povijest servisa

Glavni sudionik: Korisnik

Cilj: Pregled povijesti prijave i popravaka.

Preduvjet: Postoji barem jedna završena prijava.

Osnovni tijek:

1. Korisnik otvara „Povijest servisa“. (F-005)
 2. Sustav prikazuje sve završene prijave.
-

UC11 – Upravljanje podacima o servisu

Glavni sudionik: Administrator

Cilj: Dodavanje, uređivanje i brisanje podataka o servisu.

Preduvjet: Administrator prijavljen.

Osnovni tijek:

1. Administrator otvara „Postavke servisa“. (F-014)
 2. Uređuje ili dodaje podatke.
 3. Sustav sprema promjene u bazu.
-

UC12 – Upravljanje serviserima

Glavni sudionik: Administrator

Cilj: Dodavanje, uređivanje i brisanje serviseri te njihovo povezivanje sa servisima.

Preduvjet: Administrator prijavljen.

Osnovni tijek:

1. Administrator otvara „Serviseri“. (F-015)
 2. Pregledava, dodaje i povezuje servisere.
 3. Sustav sprema promjene.
-

UC13 – Upravljanje korisnicima

Glavni sudionik: Administrator

Cilj: Pregled i deaktivacija korisničkih računa.

Preduvjet: Administrator prijavljen.

Osnovni tijek:

1. Administrator otvara „Korisnici“. (F-016)
2. Pregledava popis korisnika.
3. Deaktivira odabranog korisnika.

Odstupanja:

- Korisnik ima aktivne prijave → prikazuje se upozorenje.
-

UC14 – Pregled prijava (serviser)

Glavni sudionik: Serviser

Cilj: Pregled svih prijava dodijeljenih serviseru. (F-006)

Osnovni tijek:

1. Serviser otvara „Prijave servisa“.
 2. Sustav prikazuje dodijeljene prijave.
 3. Serviser pregledava detalje i statuse.
-

UC15 – Dodavanje napomene

Glavni sudionik: Serviser

Cilj: Dodavanje napomene na prijavu. (F-008)

Preduvjet: Otvorena prijava (UC14).

Osnovni tijek:

1. Serviser odabire opciju „Dodaj napomenu“.

2. Unosi opis radnje.
 3. Sustav sprema napomenu.
-

UC16 – Ažuriranje statusa prijave

Glavni sudionik: Serviser

Cilj: Promjena statusa prijave. (F-007)

Osnovni tijek:

1. Serviser otvara prijavu.
 2. Odabire novi status (zaprimljeno, u obradi, završeno, odgođeno).
 3. Sustav sprema promjenu i pokreće UC17 i UC18.
-

UC17 – Generiranje obrasca (PDF)

Glavni sudionik: Sustav

Sekundarni sudionik: Serviser

Cilj: Automatsko generiranje PDF obrasca o predaji/preuzimanju vozila. (F-012)

Osnovni tijek:

1. Sustav prepoznaje završenu prijavu.
 2. Generira PDF obrazac s podacima o korisniku, vozilu i serviseru.
 3. Sprema ga u tablicu obrazac.
-

UC18 – Slanje obavijesti e-poštom

Glavni sudionik: Sustav

Sekundarni sudionik: E-mail sustav

Cilj: Obavijestiti korisnika o promjeni statusa prijave ili odgodi. (F-013, NF-007)

Osnovni tijek:

1. Sustav prepoznaje promjenu statusa.
 2. Ako je odgoda veća od 3 dana, šalje e-mail podsjetnik.
 3. E-mail sustav isporučuje poruku.
-

UC19 – Generiranje izvještaja

Glavni sudionik: Administrator

Cilj: Izrada izvještaja o poslovanju i prijavama. (F-010)

Osnovni tijek:

1. Administrator odabire parametre (vremenski raspon, tip).
 2. Sustav dohvaća podatke i generira izvještaj.
-

UC20 – Izvoz izvještaja

Glavni sudionik: Administrator

Cilj: Izvoz izvještaja u PDF, XML ili XLSX format. (F-010)

Preduvjet: Izvještaj mora biti generiran.

Osnovni tijek:

1. Administrator odabire format izvoza.
 2. Sustav generira datoteku i omogućuje preuzimanje.
-

UC21 – Odabir serviser

Glavni sudionik: Korisnik

Cilj: Odabir dostupnog serviser

Preduvjet: U tijeku je proces prijave vozila (UC5).

Osnovni tijek:

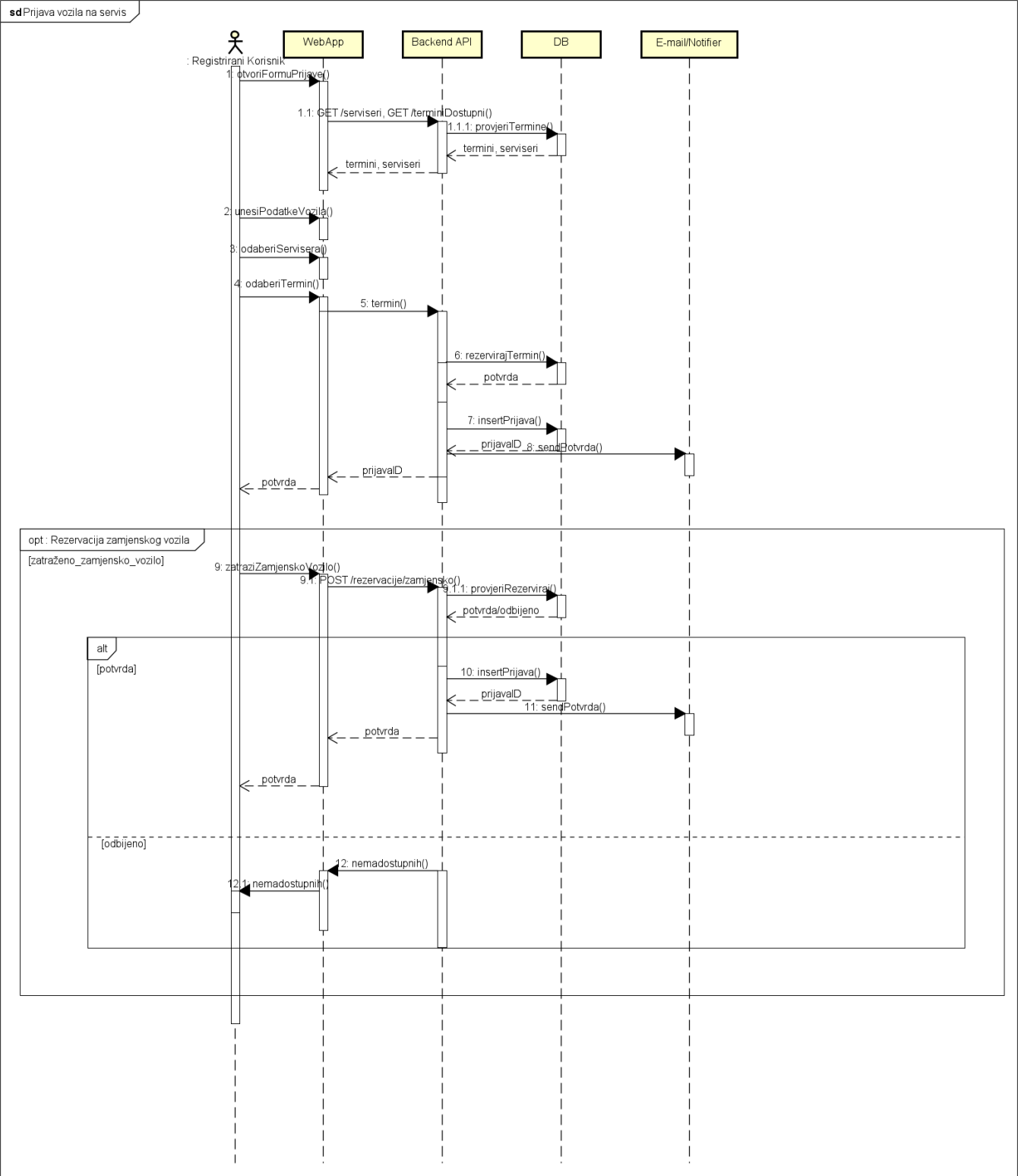
1. Sustav prikazuje listu dostupnih serviser
2. Korisnik odabire serviser.
3. Sustav sprema odabir serviser

Odstupanja:

- Nema dostupnih serviser → sustav prikazuje obavijest i predlaže alternativne termine.
-

Sekvencijski dijagrami

Prijava vozila na servis



Kratki opis sekvencijskog dijagrama: Prijava vozila na servis

Dijagram prikazuje proces u kojem korisnik putem web aplikacije prijavljuje vozilo na servis te po potrebi rezervira zamjensko vozilo.

1. Dohvaćanje podataka za formu

Korisnik otvara formu za prijavu vozila.
Web aplikacija (WebApp) od Backend API-ja dohvaća listu servisa i dostupne termine.
Backend API preuzima podatke iz baze podataka i vraća ih WebApp-u.

2. Unos podataka korisnika

Korisnik unosi podatke o vozilu, odabire servis i termin.

WebApp šalje Backend API-ju zahtjev za spremanje prijave.

3. Rezervacija termina i spremanje prijave

Backend API provjerava dostupnost odabranog termina u bazi podataka.

Ako je termin slobodan, rezervira se i sprema se prijava u bazu.

Backend API zatim vraća potvrdu WebApp-u.

4. Opcionalno: Rezervacija zamjenskog vozila

Ako korisnik zatraži zamjensko vozilo, WebApp šalje zahtjev Backend API-ju.

Backend API provjerava dostupnost zamjenskog vozila u bazi.

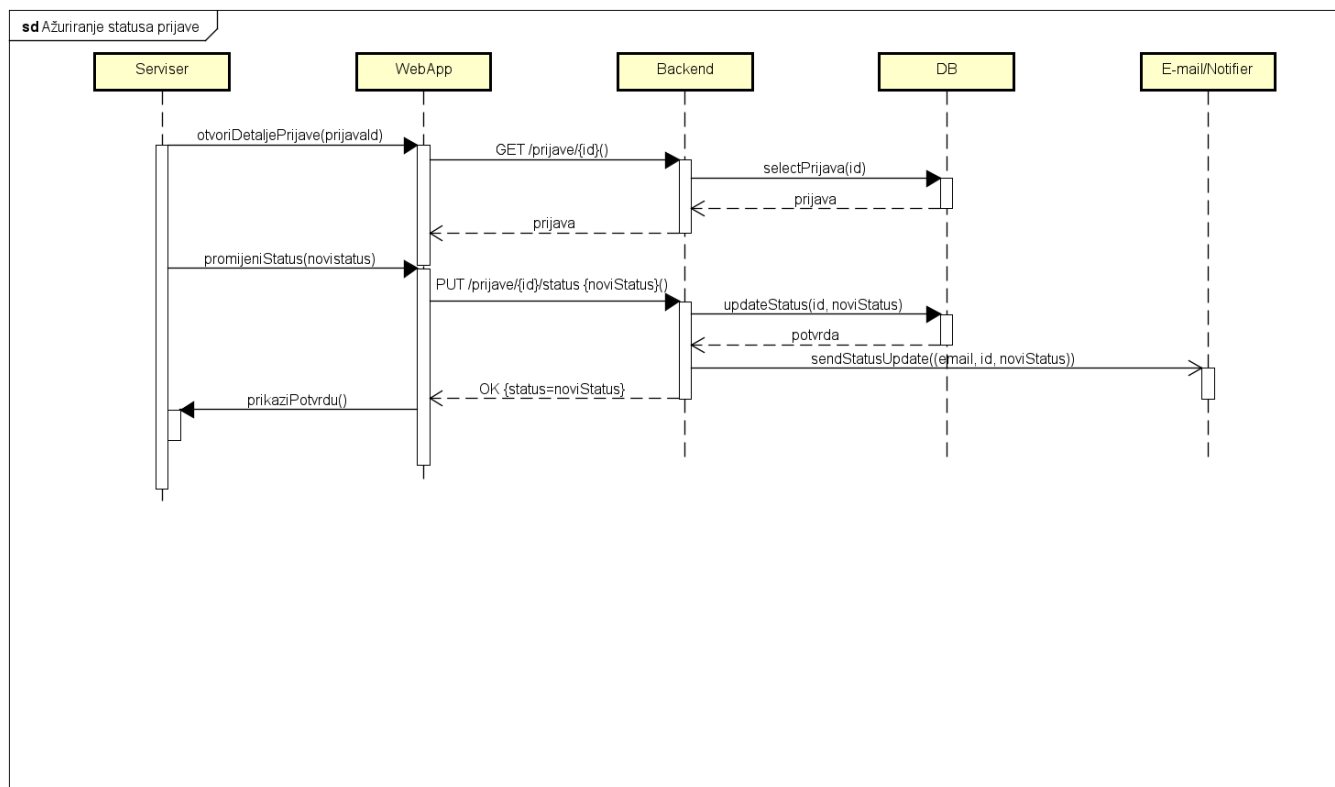
Mogući ishodi:

- Ako je zamjensko vozilo dostupno, rezervacija se sprema u bazu i korisniku se vraća potvrda.
- Ako nije dostupno, WebApp korisniku prikazuje obavijest o nedostupnosti.

5. Slanje završne potvrde

E-mail Notifier šalje korisniku e-mail s potvrdom prijave servisa i eventualne rezervacije zamjenskog vozila.

Ažuriranje statusa prijave



Kratki opis sekvencijskog dijagrama: Ažuriranje statusa prijave

Dijagram prikazuje kako serviser ažurira status postojeće prijave putem web aplikacije, pri čemu se novi status sprema u bazu podataka, a korisniku se šalje obavijest o promjeni.

1. Otvaranje detalja prijave

Serviser preko WebApp-a otvara detalje određene prijave.
WebApp traži od Backend sustava podatke o toj prijavi.
Backend dohvaća podatke iz baze i prosljeđuje ih WebApp-u.

2. Promjena statusa

Serviser u sučelju odabire novi status prijave.
WebApp zatim šalje Backend sustavu informaciju o promjeni statusa.

3. Spremanje promjene

Backend zapisuje novi status prijave u bazu podataka.
Baza potvrđuje da je promjena uspješno spremljena.

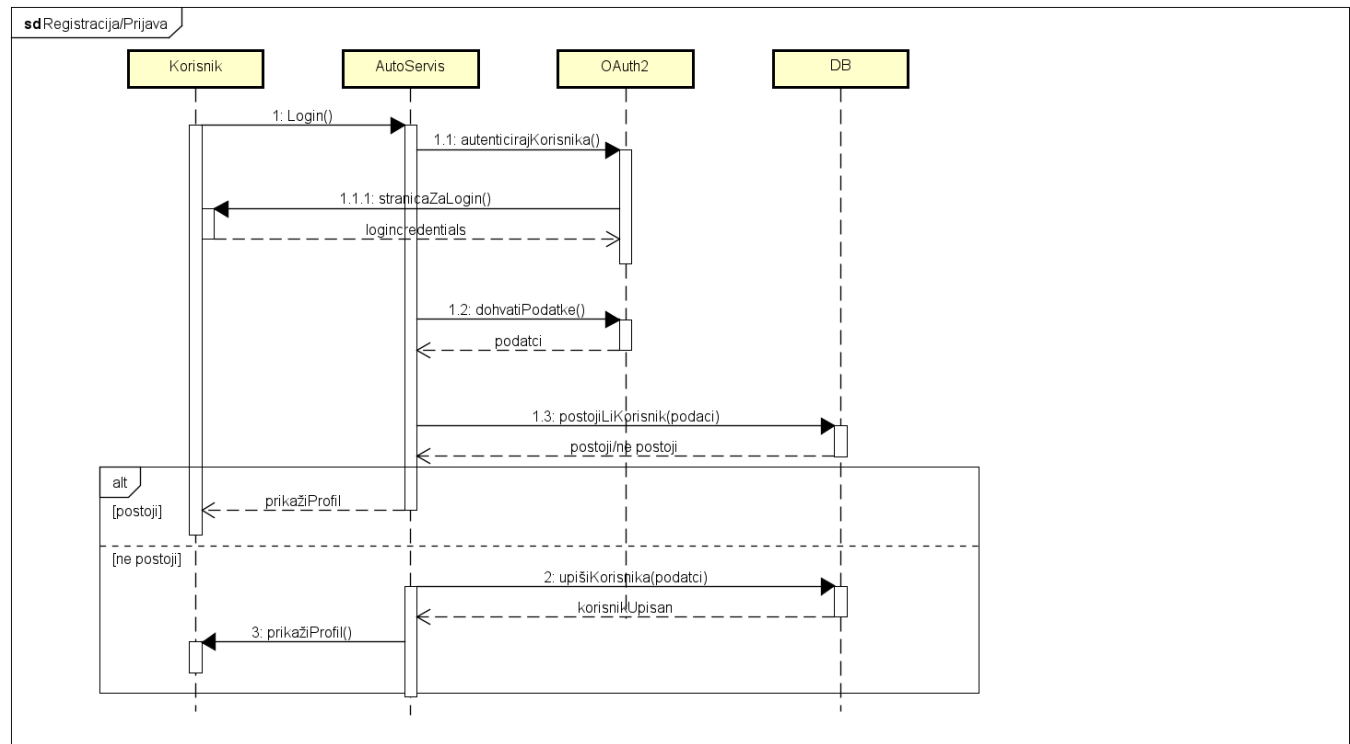
4. Slanje obavijesti korisniku

Nakon što je status ažuriran, Backend obavještava sustav za slanje e-mailova, koji korisniku šalje obavijest o novom statusu prijave.

5. Povratna informacija serviseru

Backend šalje WebApp-u potvrdu da je promjena uspješno provedena, a WebApp prikazuje serviseru poruku o uspješnom ažuriranju statusa.

Registracija/Prijava



Kratki opis sekvencijskog dijagrama: Registracija/Prijava

Dijagram prikazuje proces prijave korisnika putem vanjskog OAuth sustava, kao i automatsko kreiranje profila ako korisnik još ne postoji u aplikacijskoj bazi.

1. Pokretanje prijave

Korisnik pokreće prijavu kroz AutoServis aplikaciju.

Aplikacija preusmjerava korisnika na OAuth sustav za autentikaciju.

2. Autentikacija i dohvaćanje podataka

Korisnik se prijavljuje putem OAuth sustava.

OAuth vraća aplikaciji osnovne podatke o korisniku nakon uspješne autentikacije.

3. Provjera postojanja korisnika

Aplikacija provjerava u bazi podataka postoji li korisnik s dobivenim podacima.

Baza može vratiti jedan od dva rezultata:

- korisnik postoji
- korisnik ne postoji

4. Alternativni tokovi

Ako korisnik postoji

Aplikacija korisniku prikazuje njegov profil i omogućuje mu nastavak rada.

Ako korisnik ne postoji

Aplikacija sprema korisnikove podatke u bazu (automatska registracija).

Nakon uspješnog upisa korisnika, aplikacija prikazuje korisnički profil.

3.3 Tablica povezanosti obrazaca uporabe i funkcionalnih zahtjeva

UC	Naziv	Povezani F-ID
UC1	Pregled informacija o servisu	F-010
UC2	Registracija korisnika (OAuth2)	F-001
UC4	Dodavanje vozila	F-002
UC5	Prijava vozila na servis	F-003, F-008
UC6	Odabir termina	F-003
UC7	Rezervacija zamjenskog vozila	F-008

UC	Naziv	Povezani F-ID
UC8	Pregled statusa prijave	F-004
UC9	Preuzimanje PDF obrasca	F-011
UC10	Povijest servisa	F-004
UC11	Upravljanje podacima o servisu	F-013
UC12	Upravljanje serviserima	F-014
UC13	Upravljanje korisnicima	F-015
UC14	Pregled prijava (serviser)	F-005
UC15	Dodavanje napomene	F-007
UC16	Ažuriranje statusa prijave	F-006
UC17	Generiranje obrasca	F-011
UC18	Slanje e-mail obavijesti	F-006, F-012
UC19	Generiranje izvještaja	F-009
UC20	Izvoz izvještaja	F-009
UC21	Odabir serviser	F-003

Arhitektura i dizajn sustava

Stil arhitekture

Sustav je izrađen prema **klijent–poslužiteljskom** arhitektonskom stilu.

Korisničko sučelje (frontend) je **React SPA**, dok je poslužiteljski dio (backend) **Java Spring Boot** koji izlaže **REST API**.

Ovaj stil je odabran jer:

- pruža **jasnu separaciju odgovornosti** između prikaza (React) i poslovne logike/podataka (Spring),
- omogućuje **neovisno skaliranje** frontenda i backenda,
- olakšava **CI/CD i testiranje** zasebnih dijelova,
- REST API omogućuje kasniju **reupotrebu** (npr. mobilna aplikacija).

Podsustavi

1. Autentikacija i autorizacija (Spring Security)

Upravljanje korisničkim identitetom i pristupnim pravima; OAuth2 integracija.

2. Upravljanje korisnicima

Registracija, ažuriranje profila i uloge (administrator, serviser, korisnik).

3. Servisni nalozi i vozila

Evidencija vozila, otvaranje/praćenje naloga, poslovna pravila (statusi, napomene).

4. Rasporedi i termini

Rezervacije termina.

5. Zamjenska vozila

Evidencija i rezervacija zamjenskih vozila.

6. Izvještavanje i dokumenti (PDF)

Generiranje zapisnika/potvrda u **PDF** formatu (npr. iText/OpenPDF/PDFBox).

7. Integracije

- **Google Maps API** za prikaz lokacije i ruta (na frontendu).
- Email obavijesti/podsjetnici (SMTP ili servis).

Preslikavanje na radnu platformu

- **Razvoj:** lokalno (Spring Boot aplikacija + PostgreSQL baza; opcionalno Docker Compose).
- **Produkcija (cloud):** **GitHub Actions** za build/test/deploy pipeline.

Prednosti:

- jednostavan lokalni setup i **ponovljiv build**,
- brza i automatizirana isporuka u oblak,
- mogućnost horizontalnog skaliranja backenda.

Spremišta podataka

- **Relacijska baza podataka: PostgreSQL** (ACID, relacije, indeksi).
- Pristup bazi preko **Spring Data JPA (Hibernate)**; mogućnost pisanja prilagođenih upita (JPQL/SQL) gdje je potrebno.
- Pohrana generiranih PDF-ova: datotečni sustav ili objektna pohrana (ovisno o okruženju).

Mrežni protokoli

- **HTTP(S) + REST** za komunikaciju između React SPA i Spring Boot API-ja (JSON).
- **SMTP/Email API** za slanje obavijesti.

Obrazloženje odabira arhitekture

Izbor arhitekture temeljen na principima oblikovanja

Arhitektura sustava temelji se na **klijent-poslužiteljskom modelu** s odvojenim frontendom i backendom. Frontend je izrađen u **Reactu** kao Single Page Application (SPA), dok je backend implementiran pomoću

Spring Boot frameworka.

Komunikacija između slojeva odvija se putem **REST API-ja**, pri čemu su svi zahtjevi autentificirani i autorizirani putem **OAuth2 protokola**.

Ovakav pristup omogućuje jasan razdvajanje odgovornosti između prikaza i poslovne logike te slijedi ključne principe suvremenog arhitektonskog dizajna:

- **Visoka kohezija** – svaki modul (korisnici, vozila, nalozi, PDF generiranje, izvještavanje) ima jasno definiranu odgovornost.
- **Niska povezanost** – frontend komunicira s backendom isključivo putem REST API sučelja, čime se osigurava neovisnost slojeva.
- **Modularnost** – backend je podijeljen u slojeve (kontroleri, servisi, repozitoriji), što omogućuje jednostavno testiranje i održavanje.
- **Sigurnost** – autentikacija i autorizacija implementirane su pomoću **Spring Security + OAuth2**, čime se postiže pouzdana zaštita korisničkih podataka.
- **Skalabilnost** – frontend i backend mogu se neovisno skalirati, ovisno o opterećenju sustava.
- **Održavanje i proširivost** – standardizirani slojevi i REST komunikacija omogućuju jednostavno dodavanje novih funkcionalnosti.

Prednosti odabrane arhitekture

- **Paralelan razvoj** – frontend i backend razvijaju se i testiraju neovisno.
- **Jednostavno održavanje** – svaka promjena u jednom sloju ne utječe izravno na drugi.
- **Reusabilnost** – REST API se može koristiti i za buduće mobilne aplikacije ili vanjske integracije.
- **Sigurna autentikacija** – korisnički pristup kontrolira se pomoću **OAuth2**, koji omogućuje pouzdanu prijavu preko vanjskih identitetskih pružatelja (Google, GitHub, Auth0...).
- **Integracija s vanjskim servisima** – jednostavno povezivanje s **Google Maps API-jem** i servisima za slanje email obavijesti.

Organizacija sustava na visokoj razini

Klijent–poslužitelj arhitektura

Aplikacija je organizirana prema **klijent–poslužiteljskom modelu**, pri čemu je korisničko sučelje (frontend) odvojeno od poslužiteljske logike (backend).

- **Frontend:** Implementiran pomoću **React** frameworka kao **Single Page Application (SPA)**.
Korisničko sučelje omogućuje interaktivno iskustvo, dinamičko učitavanje podataka i prikaz statusa vozila, naloga i termina u stvarnom vremenu.
React aplikacija komunicira s poslužiteljem putem **REST API-ja** i koristi **OAuth2** autentikaciju za pristup zaštićenim resursima.
- **Backend:** Implementiran pomoću **Spring Boot** frameworka u programskom jeziku **Java**.
Backend je odgovoran za obradu zahtjeva, provedbu poslovne logike, upravljanje korisnicima, generiranje PDF dokumenata i komunikaciju s bazom podataka.

Komunikacija između slojeva unutar backenda organizirana je prema principima **MVC (Model–View–Controller)**, uz dodatne slojeve za servise i repozitorije.

Baza podataka

- **Tip:** Relacijska baza podataka
- **Tehnologija:** PostgreSQL

Služi za pohranu podataka o korisnicima, vozilima, servisnim nalogima, terminima i zamjenskim vozilima. Pristup bazi implementiran je putem **Spring Data JPA (Hibernate)**, koji omogućuje:

- automatsko mapiranje Java entiteta na relacijske tablice
- upotrebu **JPQL** i **native SQL** upita po potrebi
- jednostavno dohvaćanje i spremanje podataka kroz repozitorije

Prednosti PostgreSQL-a:

- podrška za složene upite i relacije
- transakcijska sigurnost (ACID)
- visoka stabilnost i mogućnost skaliranja

Opis tablica

osoba

Atribut	Tip podataka	Opis varijable
id_osoba	Primarni ključ	(PK) Jedinstveni identifikator osobe.
ime	VARCHAR(100)	Ime osobe.
prezime	VARCHAR(100)	Prezime osobe.
email	VARCHAR(100), UNIQUE	Valjana email adresa.
telefon	VARCHAR(20)	Kontakt broj telefona.
uloga	VARCHAR(50)	Uloga: korisnik, serviser, administrator.
oauth_id	VARCHAR(255), UNIQUE	Identifikator vanjske autentifikacije (OAuth).

marka

Atribut	Tip podataka	Opis varijable
id_marka	Primarni ključ	(PK) Identifikator marke vozila.
naziv	VARCHAR(50), UNIQUE	Naziv marke (npr. Audi, BMW).

model

Atribut	Tip podataka	Opis varijable
---------	--------------	----------------

Atribut	Tip podataka	Opis varijable
id_model	Primarni ključ	(PK) Identifikator modela vozila.
id_marka	Vanjski ključ	(FK) Povezuje model s tablicom marka.
naziv	VARCHAR(50)	Naziv modela vozila.

vozilo

Atribut	Tip podataka	Opis varijable
id_vozilo	Primarni ključ	(PK) Identifikator vozila.
id_osoba	Vanjski ključ	(FK) Vlasnik vozila → tablica osoba.
id_model	Vanjski ključ	(FK) Model vozila → tablica model.
registracija	VARCHAR(20), UNIQUE	Registarska oznaka vozila.
godina_proizvodnje	INT	Godina proizvodnje (≤ trenutna godina).

servis

Atribut	Tip podataka	Opis varijable
id_servis	Primarni ključ	(PK) Identifikator servisnog centra.
ime_servisa	VARCHAR(100)	Naziv servisa.
lokacija	TEXT	Lokacija servisa.

serviser

Atribut	Tip podataka	Opis varijable
id_serviser	Primarni ključ	(PK) Identifikator serviser.
id_osoba	Vanjski ključ	(FK) Povezuje serviser s tablicom osoba.
id_servis	Vanjski ključ	(FK) Povezuje serviser sa servisom.
je_li_voditelj	BOOLEAN	Oznaka je li serviser vođitelj servisa.

termin

Atribut	Tip podataka	Objasnjenje
id_termin	Primarni ključ	(PK) Identifikator termina servisa.
datum_vrijeme	TIMESTAMP	Datum i vrijeme termina.
zauzet	BOOLEAN	Oznaka je li termin zauzet.

prijava_servisa

Atribut	Tip podataka	Opis varijable
id_prijava	Primarni ključ	(PK) Identifikator prijave servisa.
id_vozilo	Vanjski ključ	(FK) Vozilo koje je prijavljeno.
id_serviser	Vanjski ključ	(FK) Dodijeljeni serviser.
id_termin	Vanjski ključ	(FK) Termin servisa.
status	VARCHAR(50)	Status: zaprimljeno, u obradi, završeno, odgođeno.
datum_prijave	TIMESTAMP	Kada je prijava kreirana.
datum_predaje	TIMESTAMP	Kada je vozilo predano servisu.
datum_preuzimanja	TIMESTAMP	Kada je vozilo preuzeto.
napomena_vlasnika	TEXT	Napomena vlasnika vozila.

zamjena_vozilo

Atribut	Tip podataka	Opis varijable
id_zamjena	Primarni ključ	(PK) Identifikator zamjenskog vozila.
id_model	Vanjski ključ	(FK) Model vozila → tablica model.
registracija	VARCHAR(20), UNIQUE	Registarska oznaka zamjenskog vozila.
dostupno	BOOLEAN	Trenutna dostupnost vozila.

rezervacija_zamjene

Atribut	Tip podataka	Opis varijable
id_rezervacija	Primarni ključ	(PK) Identifikator rezervacije.
id_prijava	Vanjski ključ	(FK) Povezuje rezervaciju s prijavom servisa.
id_zamjena	Vanjski ključ	(FK) Povezuje rezervaciju sa zamjenskim vozilom.
datum_od	DATE	Početni datum rezervacije.
datum_do	DATE	Završni datum rezervacije.

napomena_servisera

Atribut	Tip podataka	Opis varijable
id_napomena	Primarni ključ	(PK) Identifikator napomene.
id_prijava	Vanjski ključ	(FK) Povezana prijava servisa.
datum	TIMESTAMP	Vrijeme unosa napomene.
opis	TEXT	Sadržaj napomene.

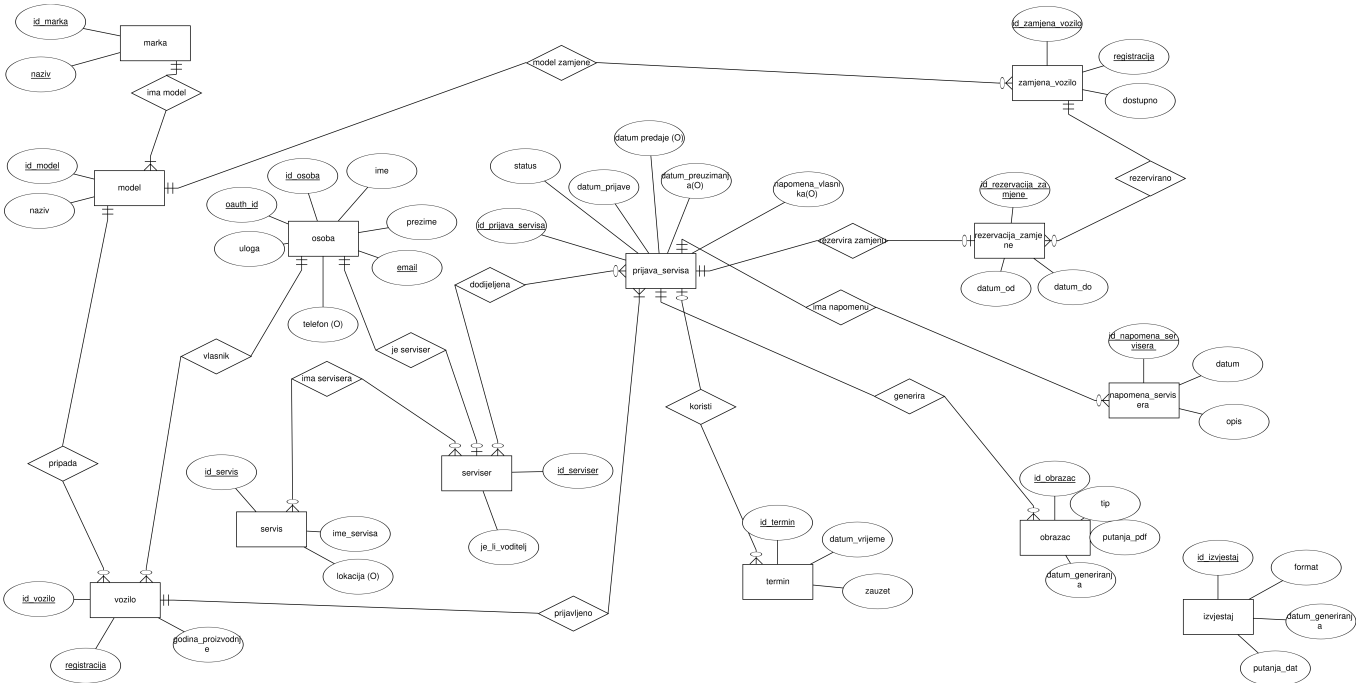
obrazac

Atribut	Tip podataka	Opis varijable
id_obrazac	Primarni ključ	(PK) Identifikator obrasca.
id_prijava	Vanjski ključ	(FK) Povezana prijava servisa.
tip	VARCHAR(20)	Tip obrasca: predaja ili preuzimanje.
putanja_pdf	TEXT	Putanja do generiranog PDF dokumenta.
datum_generiranja	TIMESTAMP	Vrijeme generiranja dokumenta.

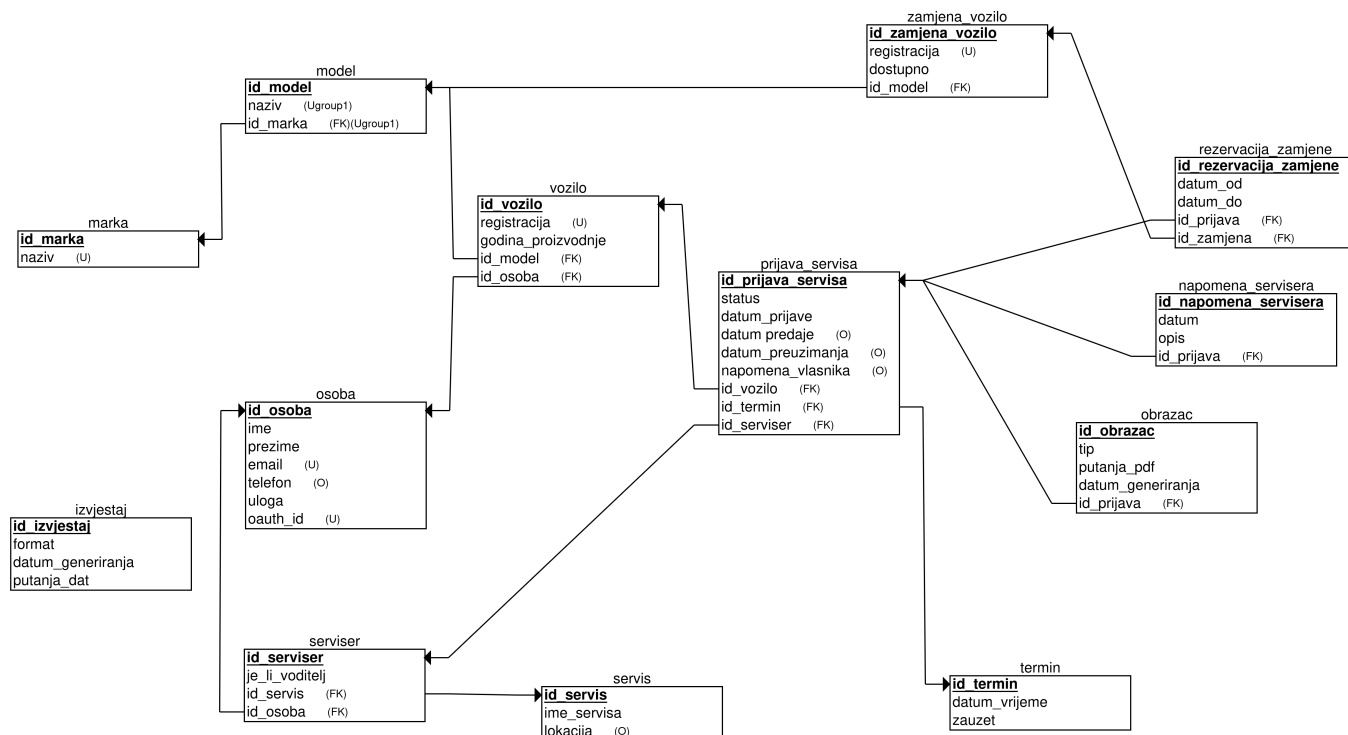
izvjestaj

Atribut	Tip podataka	Opis varijable
id_izvjestaj	Primarni ključ	(PK) Identifikator izvještaja.
format	VARCHAR(20)	Format izvještaja: pdf, xml, xlsx.
datum_generiranja	TIMESTAMP	Vrijeme generiranja izvještaja.
putanja_dat	TEXT	Putanja do datoteke izvještaja.

ER dijagram baze podataka

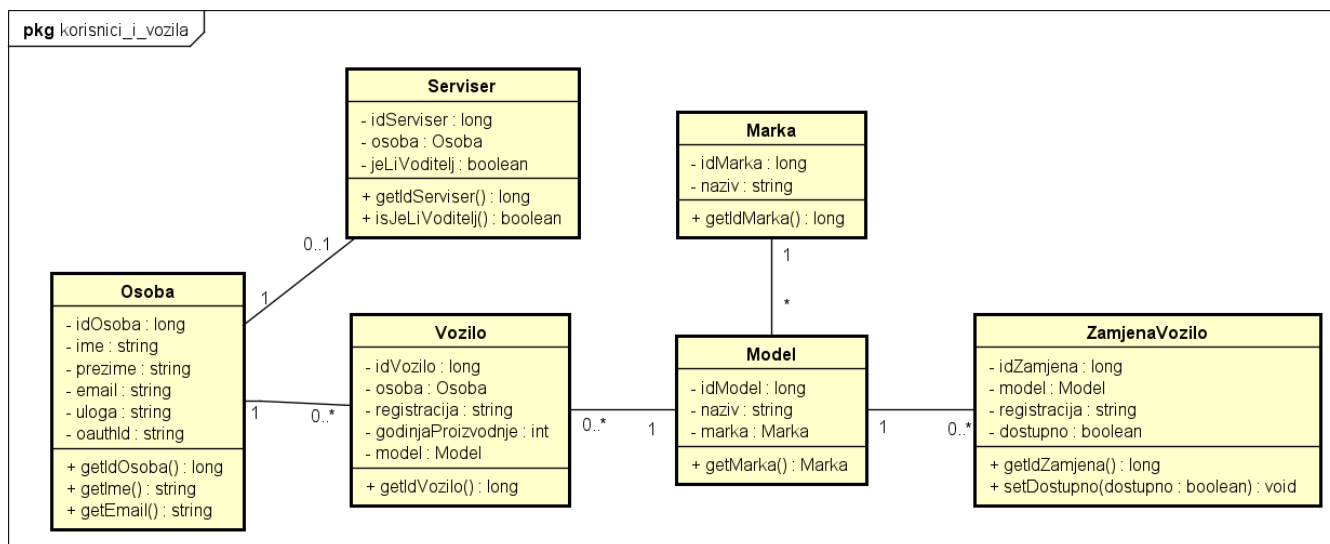


Relacijska shema baze podataka



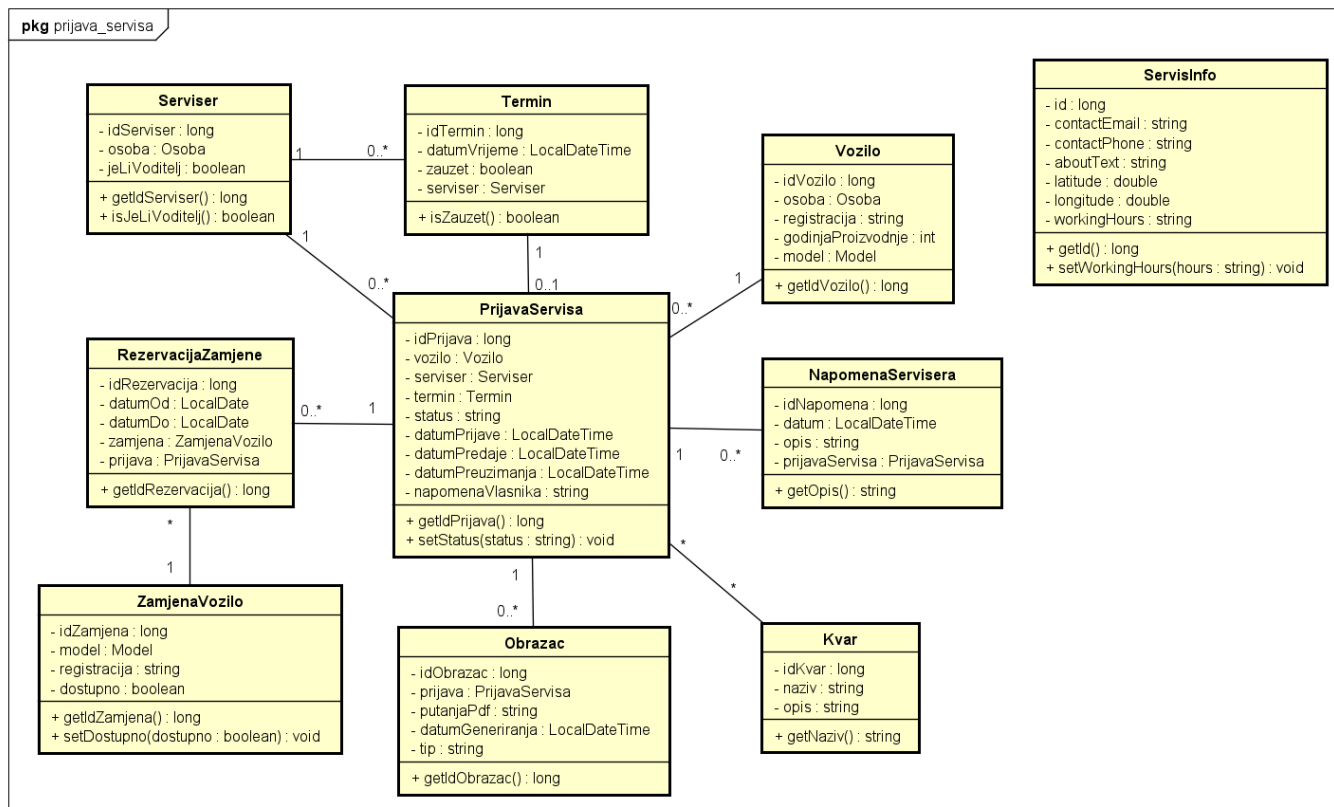
Dijagram razreda – Korisnici i vozila

Dijagram prikazuje osnovne domenske razrede za korisnike i vozila. Razred **Osoba** predstavlja korisnika sustava, a jedna osoba može imati više **Vozilo**. Razred **Vozilo** povezan je s razredima **Model** i **Marka**, dok se **ZamjenaVozilo** koristi kao vozilo dostupno za posudbu tijekom trajanja servisa. Razred **Serviser** predstavlja zaposlenika servisa povezanog s razredom **Osoba**.



Dijagram razreda – Prijava servisa

Dijagram prikazuje proces prijave servisa: kreiranje **PrijavaServisa** za odabrano **Vozilo**, dodjelu **Serviser** i rezervaciju **Termin**. Na prijavu se vežu evidentirani **Kvarovi**, **NapomeneServisera** te generirani **Obrasci** (PDF). U slučaju potrebe moguće je kreirati **RezervacijaZamjene** koja povezuje prijavu i **ZamjenaVozilo**. Klasa **ServisInfo** sadrži opće informacije o servisu (kontakt, opis, radno vrijeme i lokacija) koje se prikazuju korisnicima te nije povezana s ostalim razredima jer predstavlja globalne podatke (konfiguraciju) servisa, a ne podatke vezane uz pojedinu prijavu.



Grafičko korisničko sučelje (GUI)

Korisničko sučelje implementirano u **Reactu** omogućuje:

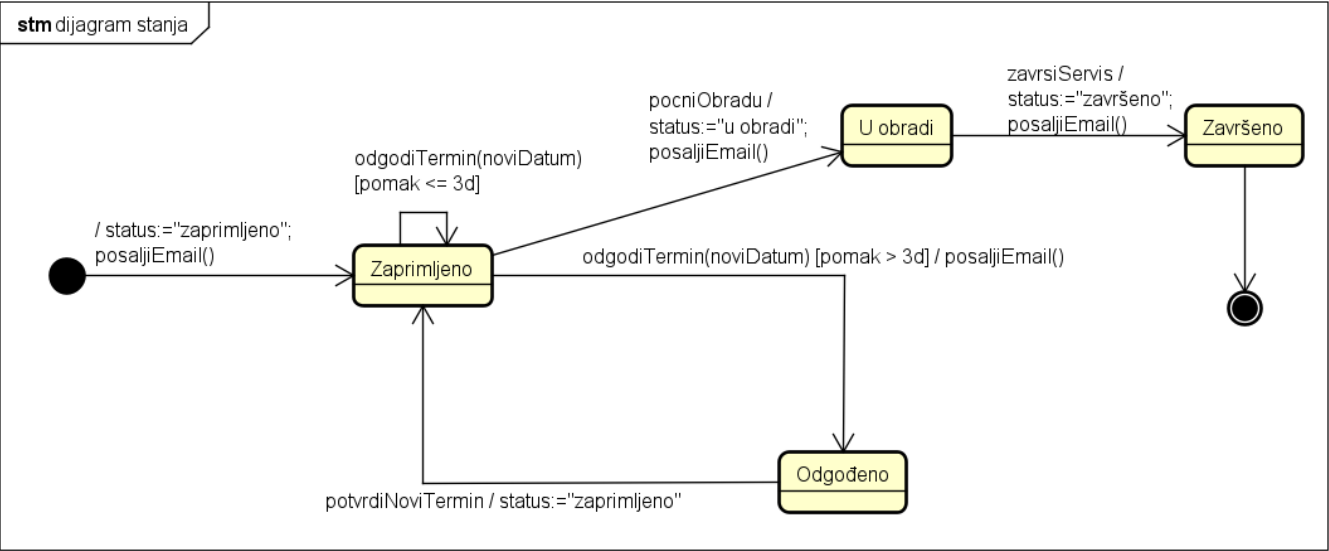
- responzivni dizajn (prilagođen desktop i mobilnim uređajima),
- prikaz liste vozila i servisnih naloga,
- prikaz statusa popravaka i zamjenskih vozila,
- prikaz lokacije servisa putem **Google Maps API-ja**,
- autentikaciju korisnika putem **OAuth2** prijave.

Dinamičko ponašanje aplikacije

Dinamičko ponašanje aplikacije prikazano je pomoću UML dijagrama stanja i aktivnosti, koji opisuju promjene stanja objekata te tijek izvršavanja ključnih procesa unutar sustava.

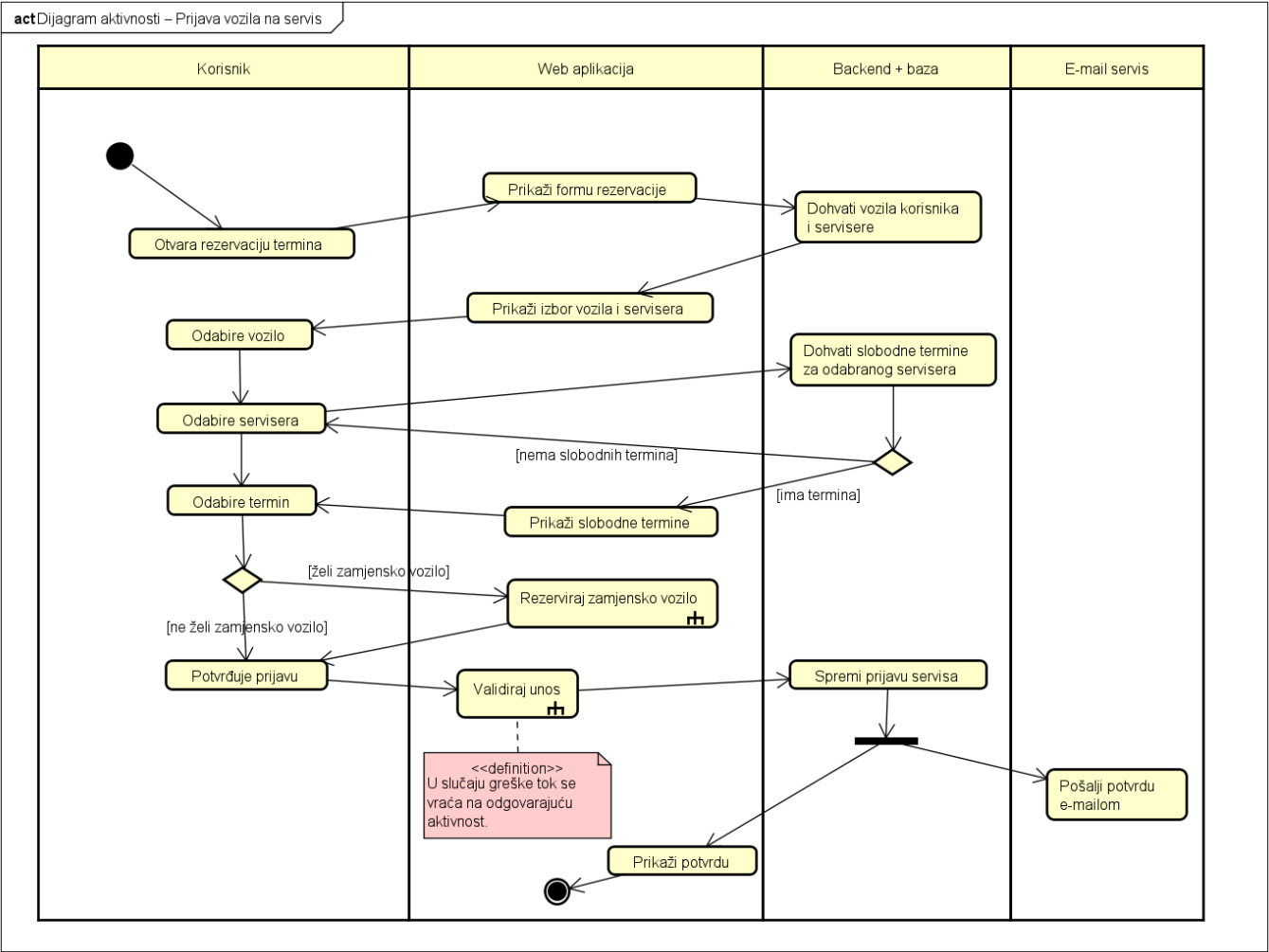
UML dijagram stanja

Dijagram stanja prikazuje promjene stanja servisne prijave tijekom njezina životnog ciklusa, ovisno o događajima i poslovnim pravilima sustava.



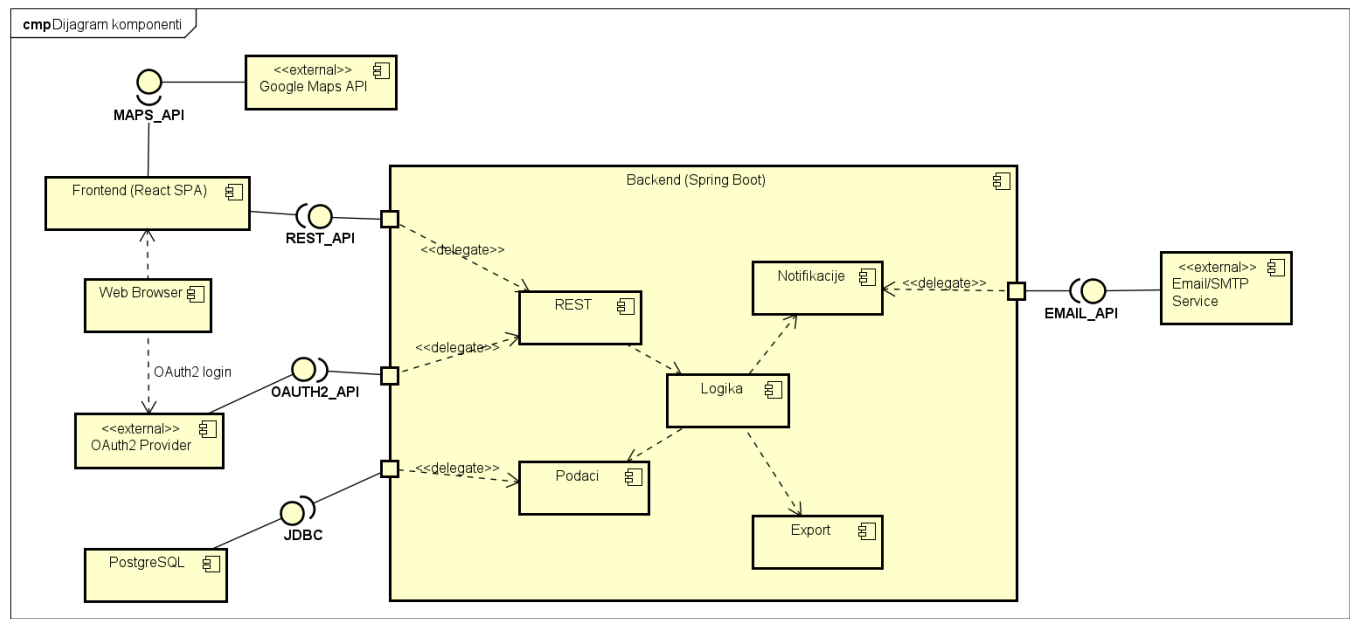
UML dijagram aktivnosti

Dijagram aktivnosti prikazuje tijek izvršavanja procesa obrade servisne prijave, uključujući slijed aktivnosti i donošenje odluka unutar sustava.



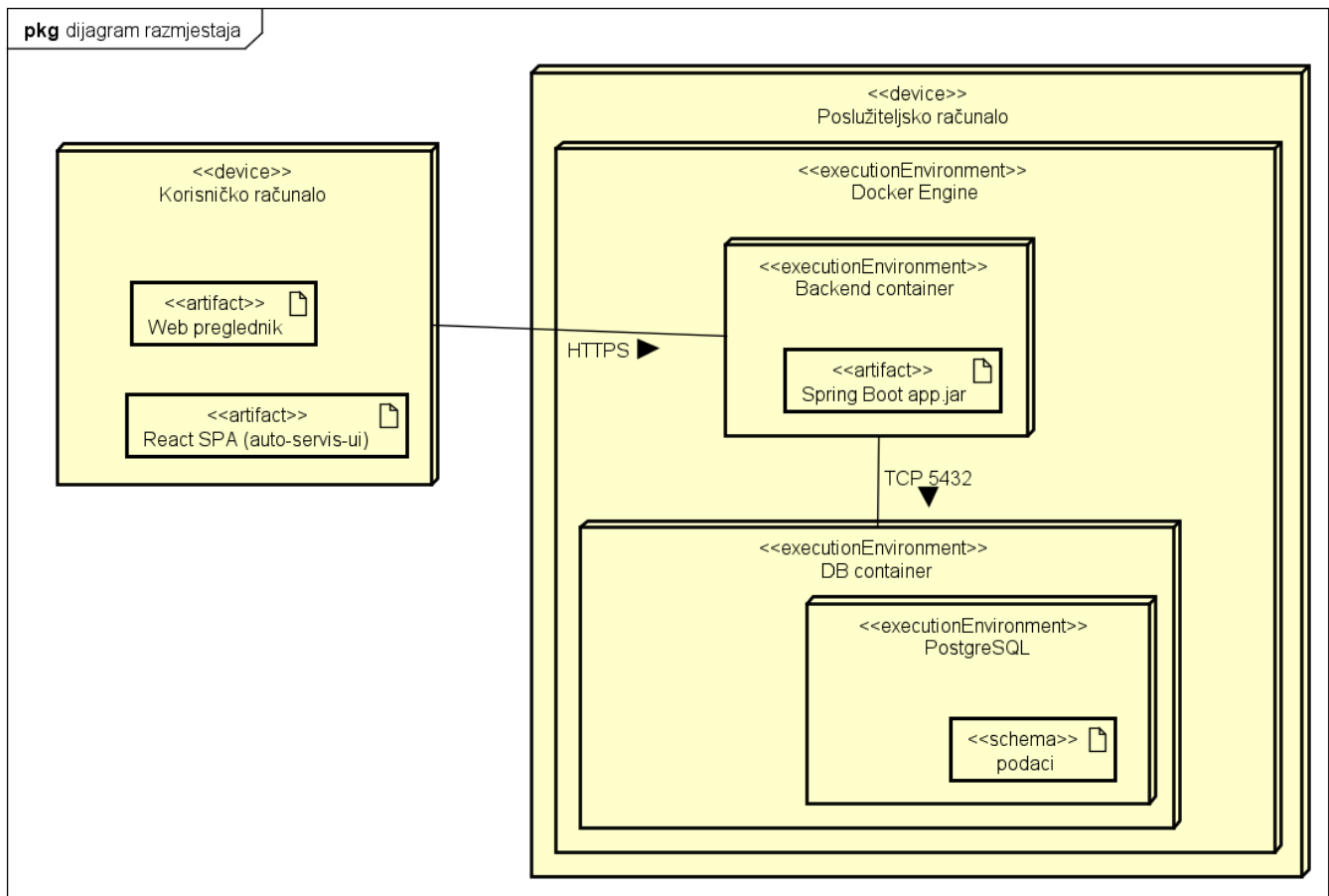
Dijagram komponentenata

Dijagram komponenta prikazuje logičku strukturu sustava i podjelu aplikacije na frontend, backend, bazu podataka i vanjske servise. Komponente komuniciraju putem REST API-ja, čime je omogućena jasna separacija odgovornosti i jednostavno održavanje sustava.



Dijagram razmještaja

Dijagram razmještaja prikazuje fizički raspored komponenata sustava na klijentskom uređaju, poslužitelju i bazi podataka te način njihove komunikacije putem mrežnih protokola.



6. ISPITIVANJE PROGRAMSKOG RJEŠENJA

6.1. Ispitivanje komponenti

Ispitivanje komponenti provedeno je korištenjem JUnit 5 okvira u kombinaciji sa Spring Boot Test okvirom. Svi testovi izvršavaju se u izoliranom okruženju s H2 in-memory bazom podataka. Ukupno je implementirano **11 testova komponenti** raspoređenih u dvije klase:

PrijavaServisaServiceIntegrationTest - Testiranje prijave servisa (5 testova)

TerminServiceIntegrationTest - Testiranje stvaranja termina (6 testova)

Test komponente 1: Kreiraj novu prijavu servisa sa svim poljima

Ulazi:

- Vozilo ID (BMW 320i, registracija ZG-123-AB)
- Serviser ID (Marko Marković)
- Termin ID (25.01.2026. u 09:00)
- Napomena: "Molim detaljnu provjeru motora"
- Kvar ID: "Motor se ne pokreće"
- Datum početka: 25.01.2026.
- Datum završetka: 26.01.2026.

Koraci ispitivanja:

1. Kreiranje kvara u bazi podataka

2. Popunjavanje DTO objekta sa svim potrebnim podacima
3. Poziv metode `prijavaServisaService.createPrijava(dto, vlasnikId)`
4. Provjera da rezultat nije `null` i da ima ID
5. Dohvat prijave iz baze podataka
6. Provjera svih pohranjenih atributa (napomena, vozilo, serviser)

Očekivani izlaz:

- Prijava je uspješno kreirana u bazi
- Sve veze (vozilo, serviser, termin, kvar) su ispravno povezane
- Napomena vlasnika je ispravno pohranjena

Dobiveni izlaz:

- Prijava uspješno kreirana s ID-om
- Vozilo ID se podudara: ZG-123-AB
- Serviser ID se podudara: Marko Marković
- Napomena: "Molim detaljnu provjeru motora" ispravno pohranjena
- Kvar "Motor se ne pokreće" povezan s prijavom

Status: Test prolazi

Test komponente 2: Prijava sa zamjenskim vozilom**Ulazi:**

- Vozilo ID (BMW 320i)
- Serviser ID (Marko Marković)
- Termin ID (25.01.2026. u 09:00)
- Napomena: "Trebam zamjensko vozilo"
- Zamjensko vozilo ID (Volkswagen Golf, ZG-456-CD)
- Datum početka: 25.01.2026.
- Datum završetka: 27.01.2026.
- Nema kvarova

Koraci ispitivanja:

1. Kreiranje zamjenskog vozila (Volkswagen Golf) u bazi
2. Popunjavanje DTO objekta s ID-om zamjenskog vozila
3. Poziv metode `createPrijava` sa zamjenskim vozilom
4. Provjera da je prijava kreirana
5. Provjera da su datumi ispravno pohranjeni

Očekivani izlaz:

- Prijava kreirana s vezom na zamjensko vozilo
- Datumi početka i završetka pohranjeni

Dobiveni izlaz:

- Prijava kreirana s ID-om
- Zamjensko vozilo (Volkswagen Golf, ZG-456-CD) povezano
- Datumi ispravno pohranjeni

Status: Test prolazi

Test komponente 3: Prijava s vremenskim rokom servisa

Ulazi:

- Vozilo ID (BMW 320i)
- Serviser ID (Marko Marković)
- Termin ID (25.01.2026. u 09:00)
- Napomena: "Trebam servis prije puta"
- Datum početka: 25.01.2026.
- Datum završetka: 30.01.2026.
- Nema kvarova

Koraci ispitivanja:

1. Definiranje datumskog raspona (5 dana)
2. Kreiranje DTO objekta s datumima
3. Poziv metode `createPrijava`
4. Provjera da je prijava kreirana
5. Dohvat iz baze i provjera datuma
6. Validacija da je `datumDo` nakon `datumOd`

Očekivani izlaz:

- Prijava kreirana s datumskim rasponom
- `datumDo` (30.01.) je nakon `datumOd` (25.01.)

Dobiveni izlaz:

- Prijava kreirana uspješno
- Datum završetka (30.01.2026) je nakon datuma početka (25.01.2026)
- Vremenski raspon od 5 dana ispravno pohranjen

Status: Test prolazi

Test komponente 4: Prijava s većim brojem kvarova

Ulazi:

- Vozilo ID (BMW 320i)
- Serviser ID (Marko Marković)
- Termin ID (25.01.2026. u 09:00)
- Napomena: "Tri problema kako je navedeno"
- Kvarovi:

1. "Brisači ne rade"
 2. "Hladnjak ne hladi kako treba"
 3. "Buka od motora je preglasna"
- Datum početka: 25.01.2026.
 - Datum završetka: 28.01.2026.

Koraci ispitivanja:

1. Kreiranje 3 različita kvara u bazi
2. Popunjavanje DTO objekta s listom od 3 kvara
3. Poziv metode `createPrijava`
4. Provjera da je prijava kreirana
5. Dohvat iz baze i brojanje povezanih kvarova

Očekivani izlaz:

- Prijava kreirana sa 3 povezana kvara
- Svi kvarovi ispravno pohranjeni i povezani

Dobiveni izlaz:

- Prijava kreirana uspješno
- Povezano 3 kvara s prijavom
- Svi nazivi kvarova ispravno pohranjeni

Status: Test prolazi

Test komponente 5: Prijava sa minimalnim podacima**Ulazi:**

- Vozilo ID (BMW 320i)
- Serviser ID (Marko Marković)
- Termin ID (25.01.2026. u 09:00)
- Napomena: "Servis"
- Zamjensko vozilo: `null`
- Datumi: `null`
- Kvarovi: prazan `List`

Koraci ispitivanja:

1. Kreiranje minimalnog DTO objekta (samo obavezna polja)
2. Poziv metode `createPrijava` s minimalnim podacima
3. Provjera da je prijava kreirana
4. Dohvat iz baze i provjera napomene

Očekivani izlaz:

- Prijava kreirana čak i s minimalnim podacima
- Napomena "Servis" pohranjena

- Neobavezna polja ostaju `null`

Dobiveni izlaz:

- Prijava kreirana uspješno
- Napomena "Servis" pohranjena
- Sustav radi ispravno i s minimalnim podacima

Status: Test prolazi

Test komponente 6: Kreiraj i spremi termin u bazu**Ulazi:**

- Datum i vrijeme: 01.02.2026. u 10:00
- Serviser ID (Jovan Jovanović)

Koraci ispitivanja:

1. Kreiranje objekta `Termin` s datumom i vremenom
2. Postavljanje servisera
3. Spremanje termina u bazu (`terminRepository.save`)
4. Provjera da termin ima ID
5. Provjera da je vrijeme ispravno pohranjeno
6. Dohvat termina iz baze pomoću ID-a
7. Usporedba dohvaćenog vremena s originalnim

Očekivani izlaz:

- Termin uspješno spremljen u bazu
- ID dodijeljen automatski
- Vrijeme pohranjeno točno (01.02.2026. 10:00)
- Termin može biti dohvaćen iz baze

Dobiveni izlaz:

- Termin spremljen s automatski generiranim ID-om
- Vrijeme 01.02.2026. 10:00 ispravno pohranjeno
- Termin uspješno dohvaćen iz baze
- Dohvaćeno vrijeme se podudara s originalnim

Status: Test prolazi

Test komponente 7: Provjeri je li termin zauzet**Ulazi:**

- Termin 1: 01.02.2026. u 10:00, `zauzet = false`
- Termin 2: 01.02.2026. u 11:00, `zauzet = true`
- Serviser ID (Jovan Jovanović)

Koraci ispitivanja:

1. Kreiranje termina 1 sa statusom `zauzet = false`
2. Kreiranje termina 2 sa statusom `zauzet = true`
3. Spremanje oba termina u bazu
4. Provjera da je termin 1 slobodan (`isZauzet() == false`)
5. Provjera da je termin 2 zauzet (`isZauzet() == true`)

Očekivani izlaz:

- Termin 1 ima status "slobodan" (`zauzet = false`)
- Termin 2 ima status "zauzet" (`zauzet = true`)
- Sustav ispravno čuva status zauzetosti

Dobiveni izlaz:

- Termin 1 je slobodan (status = false)
- Termin 2 je zauzet (status = true)
- Atribut `zauzet` ispravno pohranjen i dohvaćen

Status: Test prolazi

Test komponente 8: Testiraj više termina isti dan s različitim vremenima**Ulazi:**

- Termin 1: 01.02.2026. u 08:00
- Termin 2: 01.02.2026. u 09:30
- Termin 3: 01.02.2026. u 11:00
- Termin 4: 01.02.2026. u 14:00
- Serviser ID (Jovan Jovanović)

Koraci ispitivanja:

1. Kreiranje 4 termina za isti dan s različitim vremenima
2. Spremanje svih termina u bazu
3. Dohvat svih termina iz baze (`findAll`)
4. Provjera da je broj termina ≥ 4
5. Provjera da su vremena u ispravnom kronološkom redoslijedu

Očekivani izlaz:

- Sva 4 termina spremljena
- Vremena su u rasponima: 08:00 < 09:30 < 11:00 < 14:00
- Sustav podržava više termina istog dana

Dobiveni izlaz:

- Spremljeno 4 termina
- Vrijeme 1 (08:00) je prije vremena 2 (09:30)
- Vrijeme 2 (09:30) je prije vremena 3 (11:00)

- Vrijeme 3 (11:00) je prije vremena 4 (14:00)
- Sustav ispravno upravlja višestrukim terminima

Status: Test prolazi

Test komponente 9: Kreiranje termina sa null datumom

Ulazi:

- Datum i vrijeme: null
- Serviser ID (Jovan Jovanović)

Koraci ispitivanja:

1. Kreiranje objekta Termin s datumom null
2. Postavljanje serviser
3. Pokušaj spremanja termina u bazu (terminRepository.saveAndFlush)
4. Očekivanje iznimke zbog NOT NULL constrainta
5. Provjera da je iznimka povezana sa NULL constraintom

Očekivani izlaz:

- Baza podataka odbija spremiti termin sa null datumom
- Baca se iznimka (DataIntegrityViolationException ili ConstraintViolationException)
- Poruka iznimke sadrži riječi "NULL", "not allowed" ili "cannot be null"
- Termin nije spremljen u bazu

Dobiveni izlaz:

- Iznimka uspješno bačena (org.h2.jdbc.JdbcSQLIntegrityConstraintViolationException)
- SQL poruka: "NULL not allowed for column DATUM_VRIJEME"
- Termin nije spremljen u bazu
- Constraint iz baze ispravno proveden

Status: Test prolazi

Test komponente 10: Testiraj sortiranje termina po vremenu

Ulazi:

- Termin 1: 01.02.2026. u 09:00
- Termin 2: 01.02.2026. u 14:00
- Termin 3: 02.02.2026. u 13:00
- Termin 4: 03.02.2026. u 10:00
- Termin 5: 03.02.2026. u 15:00
- Termini spremljeni u neuređenom redoslijedu: 5, 1, 3, 2, 4

Koraci ispitivanja:

1. Kreiranje 5 termina s različitim vremenima

2. Spremanje u nasumičnom redoslijedu (5, 1, 3, 2, 4)
3. Dohvat svih termina iz baze
4. Sortiranje termina po vremenu (compareTo metoda)
5. Provjera da je prvi termin kronološki najraniji
6. Provjera da je broj termina ≥ 5

Očekivani izlaz:

- Nakon sortiranja, termini su u kronološkom redoslijedu
- Prvi termin je 01.02.2026. u 09:00 (ili raniji)
- Broj termina je minimalno 5

Dobiveni izlaz:

- Svi termini uspješno spremljeni
- Sortiranje po vremenu radi ispravno
- Prvi termin nakon sortiranja je najraniji (01.02. 09:00)
- Ukupno ≥ 5 termina

Status: Test prolazi

Test komponente 11: Dohvaćanje termina po nepostojećem ID-u**Ulazi:**

- ID termina: 99999L (ne postoji u bazi)

Koraci ispitivanja:

1. Pokušaj dohvatiti termin sa ID-om 99999 (terminRepository.findById)
2. Provjera da je rezultat prazan Optional
3. Dohvat svih termina iz baze
4. Provjera da nijedan termin u bazi nema ID 99999

Očekivani izlaz:

- findById(99999L) vraća prazan Optional
- Optional.isPresent() vraća false
- U bazi ne postoji termin s tim ID-om
- Nema iznimki ili grešaka

Dobiveni izlaz:

- findById(99999L) vratio prazan Optional
- Optional.isPresent() == false
- Nijedan termin u bazi nema ID 99999
- Metoda ispravno rukuje nepostojećim podacima

Status: Test prolazi

6.1.1. Sažetak rezultata testiranja komponenti

Broj testa	Naziv testa	Status	Kategorija
1	Kreiraj prijavu servisa sa svim poljima	Prošao	Redovni slučaj
2	Prijava sa zamjenskim vozilom	Prošao	Redovni slučaj
3	Prijava sa vremenskim rokom servisa	Prošao	Redovni slučaj
4	Prijava sa većim brojem kvarova	Prošao	Rubni uvjet
5	Prijava sa minimalnim podacima	Prošao	Rubni uvjet
6	Kreiraj i spremi termin u bazu	Prošao	Redovni slučaj
7	Provjeri da li je termin zauzet	Prošao	Redovni slučaj
8	Više termina isti dan s različitim vremenima	Prošao	Rubni uvjet
9	Kreiranje termina sa null datumom	Prošao	Izazivanje pogreške
10	Sortiranje termina po vremenu	Prošao	Redovni slučaj
11	Dohvaćanje termina po nepostojećem ID-u	Prošao	Nepostojeća funkcionalnost

Ukupno testova: 11

Prošlo: 11 (100%)

Nije prošlo: 0 (0%)

6.2. Ispitivanje sustava

Ispitivanje sustava provedeno je korištenjem Selenium WebDriver okvira s ChromeDriver upraviteljem. Testovi simuliraju korisničku interakciju s web sučeljem i provjeravaju funkcionalnost end-to-end. Ukupno je implementirano **5 testova sustava**.

Svaki test uključuje:

- Demo usporavanje (pauze od 2 sekunde) za bolje praćenje izvršavanja
- Automatsko snimanje screenshotova u ključnim koracima
- Svi screenshotovi spremljeni su u [docs/system-test-screenshots/](#) direktoriju

Test sustava 1: Učitavanje početne stranice i osnovna navigacija

Ulazi:

- URL: <http://localhost:3000/>

Koraci ispitivanja:

1. Otvaranje početne stranice u pregledniku Chrome
2. Provjera da naslov stranice sadrži "Autoservis"
3. Screenshot: [test1_01_homepage.png](#)

4. Pauza 2s (demo)
5. Klik na navigacijski link "Početna"
6. Screenshot: `test1_02_nav_home.png`
7. Pauza 2s
8. Klik na navigacijski link "Kontakt"
9. Screenshot: `test1_03_nav_kontakt.png`
10. Provjera da URL sadrži `/kontakt`

Očekivani izlaz:

- Naslov stranice sadrži "Autoservis"
- Navigacija "Početna" uspješno funkcionira
- Navigacija "Kontakt" uspješno funkcionira
- URL se mijenja u `/kontakt`

Dobiveni izlaz:

- Naslov: "Autoservis" prisutan
- Klik na "Početna" radi
- Klik na "Kontakt" radi
- URL: `http://localhost:3000/kontakt`

Screenshotovi:

- `test1_01_homepage.png` - Početna stranica
- `test1_02_nav_home.png` - Navigacija na početnu stranicu
- `test1_03_nav_kontakt.png` - Kontakt stranica

Status: Test prolazi

Test sustava 2: Navigacija kroz sve admin stranice aplikacije

Ulazi:

- URL: `http://localhost:3000/`
- Korisnik: Administrator (`test.selenium@example.com`)

Koraci ispitivanja:

1. Simulacija prijave administratora

- Postavljanje JWT tokena u sessionStorage
- Refresh stranice da React učita token
- Screenshot `test2_00_prijava_uspjesna.png`
- Pauza 2s

2. Navigacija na Kontakt stranicu

- Otvaranje početne stranice
- Klik na link "Kontakt"
- Provjera da URL sadrži `/kontakt`

- Screenshot test2_01_kontakt.png
- Pauza 2s

3. Navigacija na Servis stranicu (admin)

- Klik na link "Servis"
- Provjera da URL sadrži /servis
- Screenshot test2_02_servis.png
- Pauza 2s

4. Navigacija na Zamjenska vozila stranicu (admin)

- Klik na link "Zamjenska vozila"
- Provjera da URL sadrži /zamjene
- Screenshot test2_03_zamjene.png
- Pauza 2s

5. Navigacija na Statistika stranicu (admin)

- Klik na link "Statistika"
- Provjera da URL sadrži /statistika
- Screenshot test2_04_statistika.png
- Pauza 2s

6. Navigacija na Osobe stranicu (admin)

- Klik na link "Osobe"
- Provjera da URL sadrži /osobe
- Screenshot test2_05_osobe.png
- Pauza 2s

Očekivani izlaz:

- Prijava kao administrator uspješna
- Sve admin navigacijske poveznice funkcioniraju
- URL-ovi se ispravno mijenjaju
- Stranice se učitavaju

Dobiveni izlaz:

- Prijava: JWT token postavljen, korisnik prijavljen kao administrator
- Kontakt: URL = /kontakt
- Servis: URL = /servis
- Zamjenska vozila: URL = /zamjene
- Statistika: URL = /statistika
- Osobe: URL = /osobe
- Administrator vidi sve admin funkcionalnosti

Screenshotovi:

- [test2_00_prijava_uspjesna.png](#) - Prijava administratora (Statistika stranica)

- `test2_01_kontakt.png` - Kontakt stranica
- `test2_02_servis.png` - Servis stranica
- `test2_03_zamjene.png` - Zamjenska vozila stranica
- `test2_04_statistika.png` - Statistika stranica
- `test2_05_osobe.png` - Osobe stranica

Status: Test prolazi

Test sustava 3: Pristup zaštićenim stranicama i provjera backend veze

Ulazi:

- URL: `http://localhost:3000/servis`

Koraci ispitivanja:

1. Otvaranje zaštićene stranice `/servis`
2. Screenshot `test3_01_servis_rezultat.png`
3. Provjera da se prikazuje neki sadržaj (bez greške)
4. Pauza 2s
5. Navigacija na početnu stranicu
6. Screenshot `test3_02_home_after_servis.png`

Očekivani izlaz:

- Stranica `/servis` se učitava (može prikazati login ili prazan sadržaj)
- Nema JavaScript greške
- Povratak na početnu stranicu radi

Dobiveni izlaz:

- Stranica `/servis` učitana
- Bez JavaScript greške
- Povratak na početnu stranicu uspješan

Screenshotovi:

- `test3_01_servis_rezultat.png` - Servis stranica
- `test3_02_home_after_servis.png` - Povratak na početnu stranicu

Status: Test prolazi

Test sustava 4: Provjera nepostojeće stranice (404 handling)

Ulazi:

- URL: `http://localhost:3000/nepostojeca-stranica-12345`

Koraci ispitivanja:

1. Otvaranje nepostojeće stranice

2. Screenshot `test4_01_404_page.png`
3. Pauza 2s
4. Provjera da naslov stranice nije prazan
5. Navigacija natrag na početnu stranicu
6. Screenshot `test4_02_return_home.png`

Očekivani izlaz:

- Stranica se učitava (može prikazati 404 poruku ili redirect)
- Aplikacija ne pada (no crash)
- Povratak na početnu stranicu radi

Dobiveni izlaz:

- Aplikacija ne pada
- Naslov stranice prisutan
- Poboljšanje: Mogla bi postojati custom 404 stranica

Screenshotovi:

- `test4_01_404_page.png` - Nepostojeća stranica
- `test4_02_return_home.png` - Povratak na početnu stranicu

Status: Test prolazi

Test sustava 5: Nepostojeći API endpoint (backend resilience)**Ulazi:**

- URL: `http://localhost:3000/`
- JavaScript console provjera

Koraci ispitivanja:

1. Otvaranje konzole developera
2. Otvaranje početne stranice
3. Screenshot `test5_01_console_check.png`
4. Pauza 2s
5. Provjera da nema kritičnih JavaScript grešaka u konzoli
6. Navigacija na drugu stranicu
7. Screenshot `test5_02_final_state.png`

Očekivani izlaz:

- Aplikacija se učitava i s neuspjelim API pozivima
- Nema nekontroliranih JavaScript grešaka
- UI ostaje funkcionalan

Dobiveni izlaz:

- Aplikacija stabilna

- Nema kritičnih grešaka
- UI funkcionalan čak i ako backend nije dostupan

Screenshotovi:

- `test5_01_console_check.png` - Provjera konzole
- `test5_02_final_state.png` - Finalno stanje

Status: Test prolazi

6.2.1. Sažetak rezultata testiranja sustava

Broj testa	Naziv testa	Status	Kategorija
1	Učitavanje početne stranice i osnovna navigacija	Prošao	Redovni slučaj
2	Navigacija kroz sve admin stranice aplikacije	Prošao	Redovni slučaj
3	Pristup zaštićenim stranicama	Prošao	Rubni uvjet
4	Nepostojeća stranica (404)	Prošao	Rubni uvjet
5	Nepostojeći API endpoint	Prošao	Nepostojeća funkcionalnost

Ukupno testova: 5

Prošlo: 5 (100%)

Nije prošlo: 0 (0%)

6.3. Ukupni rezultati ispitivanja

Vrsta testiranja	Broj testova	Prošlo	Nije prošlo	Postotak
Ispitivanje komponenti	11	11	0	100%
Ispitivanje sustava	5	5	0	100%
UKUPNO	16	16	0	100%

6.4. Otkrivene greške i problemi

Tijekom testiranja otkriveno je **0 kritičnih grešaka (bugova)**. Sustav radi stabilno i prema specifikacijama.

6.5. Tehnička konfiguracija testnog okruženja

Ispitivanje komponenti (JUnit 5 + Spring Boot Test)

Okviri i biblioteke:

- JUnit 5 (Jupiter)
- Spring Boot Test 3.3.3

- H2 in-memory baza podataka (verzija 2.2.224)
- Hibernate 6.5.2.Final
- Mockito 5.12.0

Konfiguracija:

- Profil: `@ActiveProfiles("test")`
- Transakcije: `@Transactional` (rollback nakon svakog testa)
- Baza: H2 u PostgreSQL modu za kompatibilnost

Pokretanje testova:

```
mvn test -Dtest=PrijavaServisaServiceIntegrationTest
mvn test -Dtest=TerminServiceIntegrationTest
```

Ispitivanje sustava (Selenium WebDriver)

Okviri i biblioteke:

- Selenium WebDriver 4.19.1
- WebDriverManager 5.6.2 (automatska ChromeDriver konfiguracija)
- JUnit 5
- Chrome browser (verzija 144)

Konfiguracija:

- Automatsko upravljanje ChromeDriver-om
- Implicitno čekanje: 10 sekundi
- Demo pauza: 2 sekunde između koraka
- Screenshot direktorij: `docs/system-test-screenshots/`

Napomena: Testovi generiraju screenshotove u `target/screenshots/`, ali finalni rezultati se kopiraju u `docs/system-test-screenshots/` za potrebe dokumentacije i verzioniranja.

Preduvjeti:

- Backend mora biti pokrenut na `http://localhost:8080`
- Frontend mora biti pokrenut na `http://localhost:3000`

Pokretanje testova:

```
mvn test -Dtest=AutoservisSystemTest
```

Upravljanje screenshotovima:

- Automatsko imenovanje: `001_test1_01_homepage.png`
- Format: timestamp + sekvencijski broj + naziv koraka
- Screenshot-ovi se akumuliraju

6.6. Zaključak

Svi provedeni testovi (ukupno 16) su uspješno prošli, što potvrđuje:

1. **Stabilnost komponenti** - Svi servisi za prijavu servisa i termine rade ispravno u različitim scenarijima (redovni slučajevi, rubni uvjeti, opterećenje)
2. **Funkcionalnost sustava** - Korisničko sučelje je potpuno funkcionalno, navigacija radi bez greške, a aplikacija uspješno degradira čak i kada backend nije dostupan
3. **Kvaliteta koda** - Integracija između baze podataka, backend servisa i frontend sučelja funkcionira bez grešaka
4. **Pouzdanost** - Sustav je testiran s većom količinom podataka i pokazuje stabilnost

Korištene tehnologije i alati

Programski jezici

Za razvoj klijentskog dijela aplikacije korišten je **JavaScript (ES2024)**. Suvremena verzija jezika omogućuje deklarativan razvoj korisničkog sučelja, asinkrono programiranje te modularnu strukturu koda, što olakšava održavanje i daljnje proširenje aplikacije.

Poslužiteljski dio aplikacije razvijen je korištenjem **Jave 17 (LTS)**. Odabrana verzija pruža visoku razinu stabilnosti, sigurnosti i performansi te je pogodna za razvoj složenih poslovnih aplikacija.

Radni okviri i biblioteke

Frontend aplikacija implementirana je pomoću **biblioteke React (verzija 19.2.0)**. React se temelji na komponentnom pristupu koji omogućuje ponovnu iskoristivost komponenti i učinkovito upravljanje stanjem aplikacije, uz optimizirano renderiranje putem virtualnog DOM-a.

Backend je razvijen korištenjem **okvira Spring Boot (verzija 3.3.3)**. Spring Boot pojednostavljuje konfiguraciju aplikacije i omogućuje brzu izradu REST servisa, integraciju s bazom podataka, validaciju podataka i implementaciju sigurnosnih mehanizama.

Baza podataka

Kao primarna baza podataka koristi se **PostgreSQL** (produksijsko okruženje). PostgreSQL je relacijska baza podataka poznata po pouzdanosti, skalabilnosti i dobroj podršci za rad s kompleksnim upitima.

Za potrebe testiranja koristi se **H2 in-memory baza podataka (verzija 2.2.224)**, konfigurirana u PostgreSQL kompatibilnom načinu rada, čime se osigurava konzistentnost između testnog i produkcijskog okruženja.

Razvojni alati

Razvoj aplikacije proveden je korištenjem razvojnog okruženja **Visual Studio Code**, koje omogućuje učinkovito pisanje i debugiranje koda uz bogatu podršku dodataka.

Za upravljanje izvornim kodom koristi se **Git** sustav za kontrolu verzija, koji omogućuje praćenje promjena, timski rad i jednostavno vraćanje na prethodne verzije koda.

Alati za ispitivanje

Za jedinično i integracijsko testiranje backend komponenata korišten je **JUnit 5** u kombinaciji s **okvirom Spring Boot Test**. Za simulaciju ovisnosti koristi se **Mockito (verzija 5.12.0)**, čime se omogućuje testiranje u izoliranom okruženju.

Sustavsko i UI testiranje provedeno je pomoću **Selenium WebDriver (verzija 4.19.1)** alata, koji omogućuje automatizirano testiranje korisničkog sučelja u stvarnom web pregledniku.

Alati za razmještaj

Za pakiranje i razmještaj aplikacije korišten je **Docker**. Docker omogućuje pakiranje aplikacije u kontejnere, čime se osigurava konzistentno izvođenje u različitim okruženjima te pojednostavljuje proces implementacije i distribucije.

Cloud platforma

Aplikacija je razmještena na **Render** cloud platformi, koja omogućuje hostanje web aplikacija uz automatski deployment, HTTPS podršku i jednostavno upravljanje okruženjima.

1. Instalacija

1.1 Preduvjeti

Za lokalno pokretanje projekta potrebno je imati instalirano:

- Git (za kloniranje repozitorija)
- Java JDK 17+ (backend je Spring Boot)
- Maven 3.9+ (build/back-end ovisnosti)
- Node.js 18+ i npm (frontend je React)
- PostgreSQL 14+ (baza podataka)

Napomena: verzije mogu biti i novije, bitno je da su kompatibilne.

1.2 Preuzimanje izvornog koda

Izvorni kod aplikacije nalazi se u Git repozitoriju. Repozitorij se klonira na lokalno računalo pomoću sljedećih naredbi:

```
git clone https://github.com/ivanosoric/PROGI-grupa-5.3.git
cd PROGI-grupa-5.3
```

Projekt ima dva dijela:

- backend (Spring Boot)
- frontend (React)

1.3 Instalacija backend ovisnosti (Spring Boot)

Backend aplikacija nalazi se u direktoriju backend. Ovisnosti i izvršna .jar datoteka generiraju se pomoću Maven alata:

```
cd backend
mvn clean package -DskipTests
```

Nakon uspješnog izvršavanja naredbe, generirana aplikacija nalazi se u direktoriju target.

1.4 Instalacija frontend ovisnosti (React)

Frontend dio aplikacije nalazi se u direktoriju frontend. Instalacija potrebnih JavaScript ovisnosti provodi se pomoću alata npm:

```
cd frontend
npm install
```

Time se instaliraju sve ovisnosti definirane u datoteci package.json.

1.5 Instalacija baze podataka (PostgreSQL)

Aplikacija koristi PostgreSQL relacijsku bazu podataka. Prije prvog pokretanja sustava potrebno je kreirati novu bazu podataka (npr. autoservis) te inicijalizirati strukturu tablica i početne podatke.

Inicijalizacija baze podataka provodi se pomoću grafičkog alata pgAdmin.

Postupak uključuje:

- kreiranje nove baze podataka u PostgreSQL sustavu
- otvaranje alata Query Tool unutar pgAdmin sučelja
- izvršavanje SQL skripte baza_auto_servis.sql, koja definira strukturu tablica i početne podatke

2. Postavke

Ovaj dio opisuje konfiguraciju aplikacije potrebnu za ispravan rad backenda i frontenda u lokalnom okruženju.

2.1 Konfiguracija baze podataka

Aplikacija koristi PostgreSQL relacijsku bazu podataka. Struktura tablica i početni podaci inicijaliziraju se izvršavanjem SQL skripte prije prvog pokretanja sustava.

Parametri povezivanja definiraju se putem environment varijabli koje Spring Boot koristi za konfiguraciju datasource-a.

2.2 Environment varijable

Osjetljive konfiguracijske vrijednosti definiraju se putem environment varijabli u datoteci `.env`.

Backend `.env` konfiguracija:

```
ADMIN_EMAILS=
BREVO_API_KEY=
BREVO_FROM_EMAIL=

FRONTEND_URL=

GOOGLE_CALLBACK_URL=
GOOGLE_CLIENT_ID=
GOOGLE_CLIENT_SECRET=

JWT_SECRET=

SPRING_DATASOURCE_URL=
SPRING_DATASOURCE_USERNAME=
SPRING_DATASOURCE_PASSWORD=
```

Frontend `.env` konfiguracija:

```
REACT_APP_API_URL=
REACT_APP_GOOGLE_MAPS_API_KEY=
```

2.3 Opis varijabli

Administracija i autorizacija

ADMIN_EMAILS

Popis e-mail adresa administratora sustava. Vrijednosti se navode kao lista odvojena zarezima.

JWT_SECRET

Tajni ključ za potpisivanje JWT tokena. Mora biti dovoljno dug i nasumičan.

E-mail servis (Brevo)

`BREVO_API_KEY` API ključ za Brevo servis koji se koristi za slanje e-mail obavijesti.

`BREVO_FROM_EMAIL` E-mail adresa s koje se šalju poruke korisnicima.

Frontend integracija

FRONTEND_URL URL frontend aplikacije. Koristi se npr. za linkove u e-mailovima i OAuth redirekcije.

OAuth2 (Google)

GOOGLE_CLIENT_ID Identifikator aplikacije dobiven od Google OAuth2 servisa.

GOOGLE_CLIENT_SECRET Tajni ključ povezan s OAuth2 klijentom.

GOOGLE_CALLBACK_URL URL na koji Google vraća korisnika nakon uspješne autentifikacije.

Baza podataka

SPRING_DATASOURCE_URL JDBC URL PostgreSQL baze podataka.

SPRING_DATASOURCE_USERNAME Korisničko ime za pristup bazi podataka.

SPRING_DATASOURCE_PASSWORD Lozinka za pristup bazi podataka.

Frontend varijable

REACT_APP_API_URL URL backend aplikacije kojem se frontend povezuje putem REST API-ja.

REACT_APP_GOOGLE_MAPS_API_KEY API ključ za Google Maps servis koji se koristi za prikaz lokacije servisa unutar aplikacije.

3. Pokretanje aplikacije

Ovaj dio opisuje pokretanje aplikacije u razvojnome i produkcijskom okruženju. Aplikacija se sastoji od dva dijela: backend (Spring Boot) i frontend (React).

3.1 Pokretanje u razvojnom okruženju

Pokretanje backenda Backend se pokreće iz direktorija backend. Prije pokretanja mora biti ispravno konfigurirana datoteka .env te dostupna PostgreSQL baza podataka.

Pokretanje se provodi Maven alatom:

```
cd backend
mvn spring-boot:run
```

Pokretanje frontenda Frontend se pokreće iz direktorija frontend. Prije pokretanja mora biti postavljena datoteka .env, posebno varijabla REACT_APP_API_URL koja upućuje na backend.

Pokretanje u razvojnom načinu rada:

```
cd frontend
npm start
```

Provjera rada aplikacije Nakon uspješnog pokretanja:

frontend je dostupan na URL-u definiranom razvojnim poslužiteljem (<http://localhost:3000>)

backend je dostupan na portu koji je konfiguriran u Spring Boot postavkama (<http://localhost:8080>)

4. Upute za administratore

Ovaj dio opisuje osnovne administracijske aktivnosti koje su potrebne nakon puštanja aplikacije u pogon. Administratorske funkcionalnosti namijenjene su održavanju sustava, nadzoru rada aplikacije i upravljanju korisnicima.

4.1 Pristup aplikaciji

Pristup aplikaciji provodi se putem vanjske OAuth2 autentifikacije. Ovisno o dodijeljenoj ulozi, sustav korisniku nakon prijave prikazuje odgovarajuće korisničko sučelje.

Administratorska uloga dodjeljuje se korisnicima čije se e-mail adrese nalaze u konfiguracijskoj varijabli `ADMIN_EMAILS`.

4.2 Upravljanje korisnicima i servisom

Administrator ima mogućnost:

- pregleda registriranih korisnika
- pregleda i upravljanja serviserima
- unosa i izmjene osnovnih podataka o servisu
- generiranje statistike o radu servisa

Ove funkcionalnosti omogućuju osnovno održavanje sustava bez potrebe za izravnim pristupom bazi podataka.

4.3 Nadzor rada aplikacije

Za praćenje rada aplikacije administrator može koristiti:

- aplikacijske logove (u lokalnom ili produkcijskom okruženju)
- administratorske funkcionalnosti dostupne kroz backend API

U slučaju korištenja cloud platforme (Render), logovi su dostupni putem administratorskog sučelja same platforme.

4.4 Održavanje baze podataka

Administrator je odgovoran za:

- redovito sigurnosno kopiranje baze podataka
- nadzor integriteta podataka
- korištenje odvojene baze podataka za razvojno i produkcijsko okruženje

Izravne izmjene podataka u bazi preporučuju se samo u iznimnim slučajevima.

4.5 Ažuriranje aplikacije

Ažuriranje aplikacije provodi se objavom nove verzije izvornog koda u Git repozitorij.

U produkcijskom okruženju aplikacija je postavljena na Render platformu. Nakon izmjene koda, deploy se pokreće ručno putem administratorskog sučelja Render platforme.

Prilikom deploja Render:

- preuzima najnoviju verziju izvornog koda iz repozitorija
- instalira potrebne ovisnosti
- izvršava build aplikacije
- ponovno pokreće backend servis

5. Primjer za Render platformu (Cloud deploy)

Za produkcijsko okruženje aplikacija je postavljena na cloud platformu Render. Render omogućuje jednostavno postavljanje web aplikacija povezanih s Git repozitorijem te upravljanje deploy procesom putem web sučelja.

5.1 Priprema repozitorija

Kako bi aplikacija bila prikladna za deploy na Render platformu, repozitorij mora sadržavati:

- backend aplikaciju s definiranim build procesom (Spring Boot + Maven)
- frontend aplikaciju s definiranim build procesom (React)
- konfigurirane environment varijable koje se postavljaju unutar Render sučelja

Nijedna osjetljiva konfiguracija (API ključevi, lozinke) ne nalazi se u samom repozitoriju.

5.2 Konfiguracija servisa na Renderu

Na Render platformi kreira se novi Web Service koji se povezuje s GitHub repozitorijem projekta.

Tijekom konfiguracije definiraju se:

- repozitorij i grana iz koje se deploja aplikacija
- build i start postavke servisa
- environment varijable potrebne za rad aplikacije (prema backend .env konfiguraciji)

Frontend i backend koriste vlastite produkcijske environment varijable koje su definirane unutar Render sučelja.

5.3 Deploy aplikacije

Deploy aplikacije na Render platformi pokreće se ručno putem administratorskog sučelja.

Prilikom pokretanja deploja Render:

- preuzima najnoviju verziju izvornog koda iz repozitorija
- instalira potrebne ovisnosti
- izvršava build aplikacije

- pokreće backend servis u produkcijskom okruženju

Tijekom deploya i nakon njegovog završetka dostupni su logovi koji omogućuju provjeru uspješnosti pokretanja aplikacije.

5.4 Pristup aplikaciji

Nakon uspješno završenog deploya aplikacija je dostupna putem javnog URL-a koji generira Render platforma: <https://autoservis-frontend.onrender.com>.

Pristup aplikaciji moguć je putem internetskog preglednika, bez potrebe za dodatnom instalacijom na korisničkoj strani.

Administratorske funkcionalnosti dostupne su prijavljenim korisnicima s administratorskom ulogom, definiranom putem konfiguracije sustava.

5.5 Ograničenja

U produkcijskom okruženju potrebno je uzeti u obzir sljedeća ograničenja:

- aplikacija ovisi o dostupnosti Render platforme
- deploy se mora pokrenuti ručno nakon svake izmjene koda
- performanse ovise o odabranom planu uslugeTijekom izrade projektnog zadatka razvijena je web aplikacija **Auto Servis** koja služi za podršku poslovanju auto servisa. Cilj aplikacije bio je olakšati i digitalizirati procese zaprimanja vozila, zakazivanja termina, praćenja statusa popravka te upravljanja zamjenskim vozilima. Projekt je razvijan postupno, kroz više faza, uz redovito praćenje napretka putem sustava za verzioniranje i projektne dokumentacije.

Zaključak i budući rad

Tijekom razvoja susreli smo se s različitim tehničkim i organizacijskim izazovima, posebno vezanim uz integraciju vanjskog OAuth2 servisa za prijavu korisnika, izradu i povezivanje baze podataka te usklađivanje poslovnih procesa unutar aplikacije. Svi uočeni problemi uspješno su riješeni tijekom implementacije, a razvijena aplikacija zadovoljava postavljene zahtjeve i omogućuje stabilan i pouzdan rad.

Rad na projektu omogućio je članovima tima stjecanje novih znanja i praktičnih iskustava iz analize zahtjeva, UML modeliranja, dizajna sustava, timskog rada te korištenja suvremenih razvojnih alata. Iako je aplikacija u konačnoj verziji u potpunosti funkcionalna, u budućnosti ju je moguće dodatno unaprijediti dodavanjem novih funkcionalnosti, poboljšanjem performansi i prilagodbom rješenja složenijim poslovnim potrebama.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. Astah Community, <http://astah.net/editions/uml-new>
3. The Unified Modeling Language, <https://www.uml-diagrams.org/>
4. React, <https://react.dev/learn>
5. Spring Boot – <https://spring.io/projects/spring-boot>
6. PostgreSQL dokumentacija – <https://www.postgresql.org/docs/>

Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Kreiran osnovni kostur dokumentacije i početno strukturiranje sadržaja	Ivano Sorić	20.10.2024
0.2	Dodana uvodna sekcija i opis poslovnog problema	Ivan Petrović	22.10.2024
0.3	Provedena analiza korisničkih potreba i izrađen početni popis funkcionalnosti	Ivano Sorić	25.10.2024
0.4	Izrađena specifikacija korisničkih uloga i definiran tok interakcija	Ivan Petrović	27.10.2024
0.5	Dodani dijagrami slučajeva korištenja (Use Case)	Ivan Petrović	30.10.2024
0.6	Kreirana struktura baze podataka i dodani ER dijagrami	Katarina Bencun	02.11.2024
0.7	Opisana logika glavnih poslovnih procesa i definirani tokovi podataka	Ivano Sorić	05.11.2024
0.8	Dodana poglavlja o arhitekturi sustava i tehničkim pretpostavkama	Ivan Petrović	07.11.2024
0.9	Provedena revizija dokumentacije, ispravci i usklađivanje terminologije	Ivano Sorić	10.11.2024
1.0	Dovršena finalna verzija dokumentacije i priprema za predaju	Ivan Petrović	12.11.2024
1.1	Nadopunjena arhitektura i dizajn sustava	Ivan Petrović	15.1.2026
1.2	Arhitektura komponenata i razmještaja	Ivan Petrović	22.1.2026
1.3	Ispitivanje programskog rješenja	Adrijan Lazanja	22.1.2026
1.4	Tehnologije za implementaciju aplikacije	Ivan Petrović	22.1.2026
1.5	Upute za puštanje u pogon	Katarina Bencun	23.1.2026

Dnevnik sastajanja

1. sastanak

Datum: 15. listopada 2024.
Prisustvovali: Ivan Petrović, Ivano Sorić, Katarina Bencun, Adrijan Lazanja, Mia Bagarić, Domagoj Vitković

Teme sastanka

Podjela uloga u timu
Frontend: Mia Bagarić, Domagoj Vitković
Backend: Adrijan Lazanja, Katarina Bencun
Dokumentacija: Ivan Petrović, Ivano Sorić

Uspostava komunikacije
Otvoren Discord server
Kreirana WhatsApp grupa

2. sastanak

Datum: 22. listopada 2024.
Prisustvovali: Ivan Petrović, Ivano Sorić, Katarina Bencun, Adrijan Lazanja, Mia Bagarić, Domagoj Vitković

Teme sastanka

Definiranje funkcionalnih i nefunkcionalnih zahtjeva
Raspravljene osnovne funkcionalnosti sustava
Kreiran inicijalni popis zahtjeva

Raspodjela zadataka
Frontend: izrada početne stranice i UI strukture
Backend: priprema temeljnog API sloja i konfiguracija OAuth2
Dokumentacija: postavljanje strukture wiki stranica

3. sastanak

Datum: 29. listopada 2024.
Prisustvovali: Ivan Petrović, Ivano Sorić, Katarina Bencun, Adrijan Lazanja, Mia Bagarić, Domagoj Vitković

Teme sastanka

Pregled napretka članova tima
Planiranje sljedeće faze razvoja
Frontend: priprema stranice za unos podataka o vozilu
Backend: izrada API logike za dodavanje vozila i spremanje u bazu
Dokumentacija: ažuriranje specifikacije zahtjeva i dijagrama

4. sastanak

Datum: 5. studenog 2024.
Prisustvovali: Ivan Petrović, Ivano Sorić, Katarina Bencun, Adrijan Lazanja, Mia Bagarić, Domagoj Vitković

Teme sastanka

Revizija postojećeg koda i dokumentacije
Dogovor oko implementacije ključnih funkcionalnosti
Frontend: UI za pregled i dodavanje vozila
Backend: dovršavanje funkcionalnosti za spremanje vozila

Backend: integracija OAuth2 prijave
Dokumentacija: izrada Use Case dijagrama i ažuriranje arhitekture

5. sastanak

Datum: 12. studenog 2024.
Prisustvovali: Ivan Petrović, Ivano Sorić, Katarina Bencun, Adrijan Lazanja, Mia Bagarić, Domagoj Vitković

Teme sastanka

Stanje projekta

- Uspješno implementirane funkcionalnosti:
- dodavanje vozila na stranicu autoservisa
 - autentikacija putem OAuth2

Planovi za sljedeće korake

- Daljnje proširenje funkcionalnosti servisa
- Dorade na sučelju i optimizacija korisničkog iskustva
- Ažuriranje i finalizacija dokumentacije

6. sastanak

Datum: 14. siječnja 2026.
Prisustvovali: Ivan Petrović, Ivano Sorić, Katarina Bencun, Adrijan Lazanja, Mia Bagarić, Domagoj Vitković

Teme sastanka

- Većina planiranih funkcionalnosti je implementirana
- Priprema sustava za konačnu predaju
- Dogovor oko raspodjele preostalih zadataka

Prikaz aktivnosti grupe

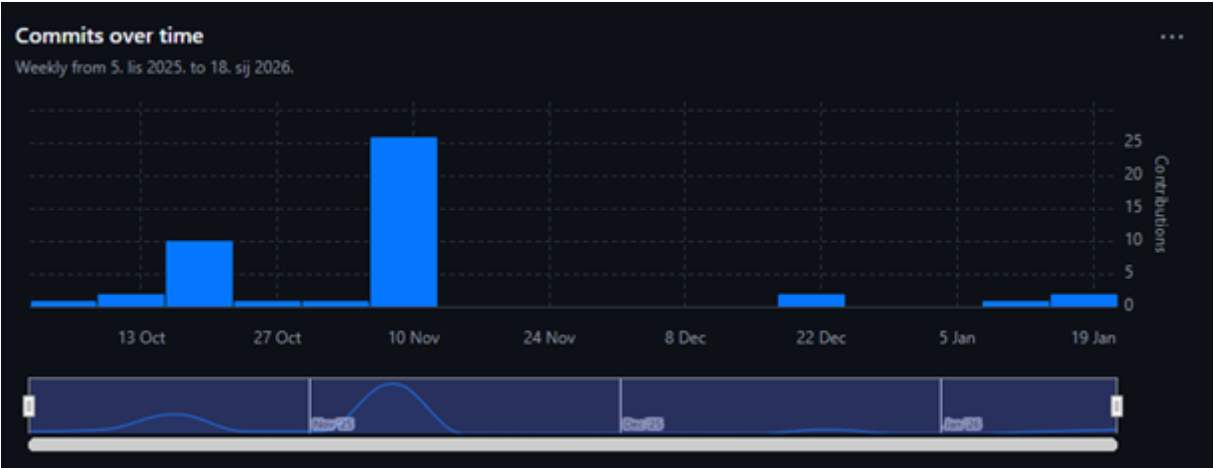
Napomena: Doprinos članova u pojedinim aktivnostima prikazan je u postocima te predstavlja relativni udio rada pojedinog člana u odnosu na ukupni rad uložen u pojedinu aktivnost.

Aktivnost	Ivan Petrović	Ivano Sorić	Katarina Bencun	Adrijan Lazanja	Mia Bagarić	Domagoj Vitković
Upravljanje projektom	-	70%	30%	-	-	-
Opis projektnog zadatka	10%	20%	20%	20%	10%	20%
Funkcionalni zahtjevi	70%	20%	-	-	10%	-

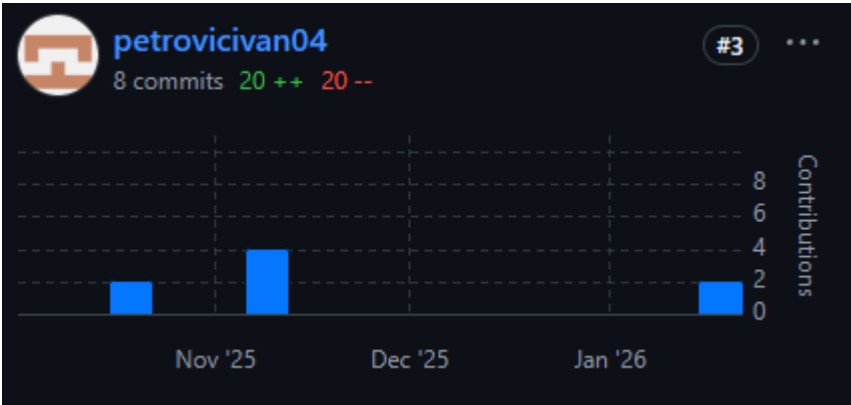
Aktivnost	Ivan Petrović	Ivano Sorić	Katarina Bencun	Adrijan Lazanja	Mia Bagarić	Domagoj Vitković
Opis pojedinih obrazaca	70%	30%	-	-	-	-
Dijagram obrazaca	40%	40%	20%	-	-	-
Sekvencijski dijagrami	-	100%	-	-	-	-
Arhitektura i dizajn sustava	40%	40%	-	-	-	-
Baza podataka	-	-	100%	-	-	-
Dijagram razreda	90%	10%	-	-	-	-
Dijagram aktivnosti	100%	-	-	-	-	-
Dijagram komponenti	100%	-	-	-	-	-
Ispitivanje programskog rješenja	-	-	-	100%	-	-
Dijagram razmještaja	100%	-	-	-	-	-
Upute za puštanje u pogon	-	-	100%	-	-	-
Izrada aplikacije	5%	20%	20%	20%	10%	25%
Izrada prezentacije	-	-	-	-	100%	-

Dijagram aktivnosti kroz vrijeme

Prikazani dijagram aktivnosti prikazuje commitove po tjednima, pri čemu završne promjene iz tekućeg tjedna nisu u potpunosti prikazane zbog načina generiranja GitHub statistika.



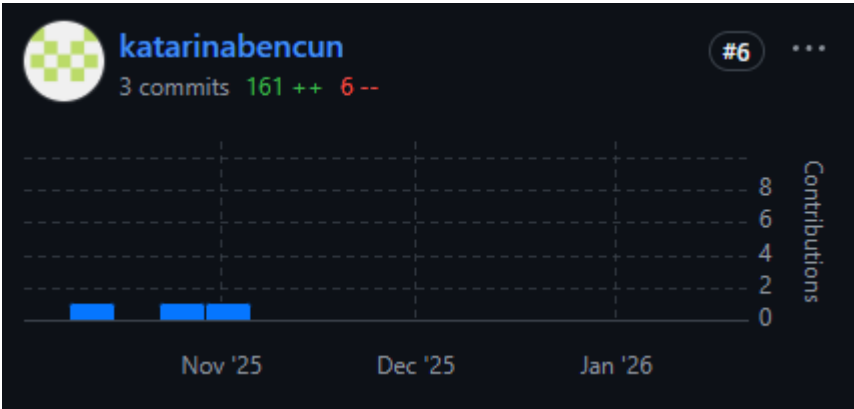
Ivan Petrović



Ivano Sorić



Katarina Bencun



Adrijan Lazanja



Mia Bagarić



Domagoj Vitković



Ključni izazovi i rješenja

Tijekom izrade projekta susreli smo se s nekoliko tehničkih i organizacijskih izazova koji su zahtijevali dodatni trud i prilagodbu tijekom rada. Najveći problemi odnosili su se na povezivanje frontenda i backend-a, implementaciju osnovnih funkcionalnosti aplikacije te zajednički rad u Git repozitoriju, gdje je ponekad dolazilo do manjih konflikata prilikom spajanja promjena. Dodatni izazov bila je integracija autentifikacije korisnika te usklađivanje promjena u backend API-ju s korisničkim sučeljem aplikacije. Također, bilo je potrebno obratiti pažnju na ispravan tijek rada aplikacije i obradu različitih rubnih slučajeva.

Navedeni problemi rješavani su postupno, kroz iterativni razvoj, bolju podjelu zadataka među članovima tima i češću komunikaciju. Tijekom rada stekli smo iskustvo u planiranju razvoja, koordinaciji timskog rada i korištenju Git-a, što je na kraju rezultiralo stabilnom i funkcionalnom aplikacijom te boljom organizacijom rada unutar tima.