Cilj i problematika zadatka

Cilj ovog projektnog zadatka je razviti platformu koja bi olakšala život vlasnicima pasa jednostavnim pronalaženjem provjerenih šetača pasa, čime se olakšava svakodnevna briga za njihove ljubimce.

Problematika koju ovaj projekt nastoji razriješiti odnosi se na nedostatak pouzdanih informacija o šetačima i organizaciji cjelokupne šetnje. Problematiku rješavamo razvojem web aplikacije koja bi omogućila registraciju korisnika u ulozi šetača ili vlasnika pasa kreiranjem detaljnih profila pasa i šetača. Nadalje, korisnicima će biti omogućeno pregledavanje i filtriranje termina prema različitim značajkama, rezervacija termina te plaćanje usluga putem vanjskih servisa poput PayPala ili kreditnih kartica. Platforma će uključivati integrirani kalendar koji bi omogućio pregled slobodnih termina i integrirani chat sustava koji bi olakšao komunikaciju između vlasnika i šetača.

Korisnički zahtjevi

Vlasnik pasa putem platforme kreira profile svojih pasa s osnovnim informacijama poput imena, pasmine, starosti, zdravstvenih napomena, razine energije, socijalizacije s drugim psima i dopuštenih poslastica. Pri odabiru šetnje vlasnici imaju mogućnost pregledavanja profila šetača, filtriranja po lokaciji, cijeni i ocjenama šetača, te hoće li šetnja biti individualna ili grupna. Tijekom rezervacije odabiru odgovarajući termin i polazišnu adresu.

Šetači pasa kreiraju profil s informacijama poput kontakta, fotografije i popisa oglašenih termina šetnji s njihovom cijenom i trajanjem. Na njihovim profilima se prikazuju ocjene i recenzije prethodnih klijenata. Za razliku od vlasnika pasa, šetači moraju platiti mjesečnu ili godišnju članarinu za korištenje platforme.

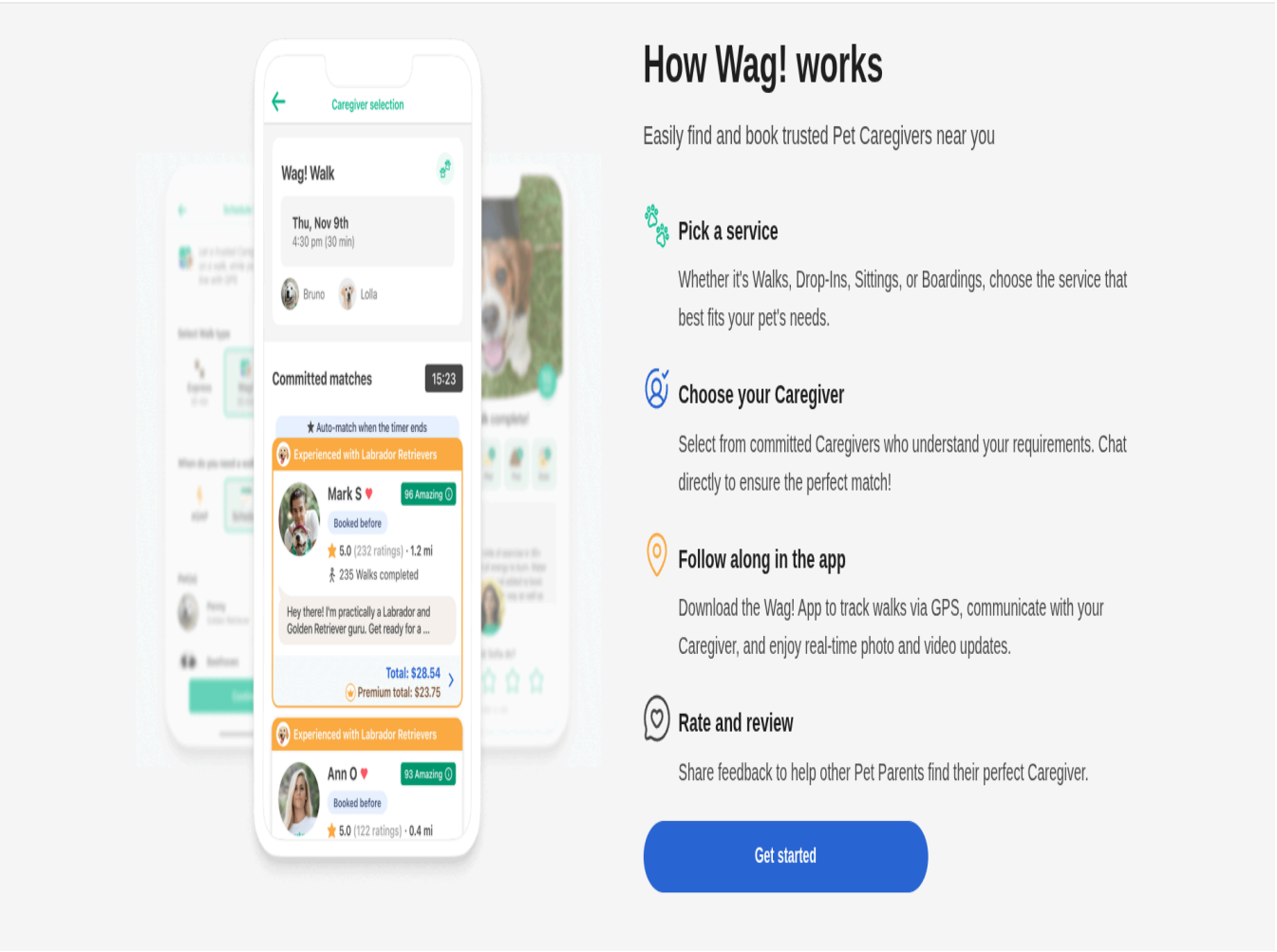
Registracija i prijava korisnika ostvaruju se korištenjem vanjskih servisa za autentifikaciju.

Potencijalna korist projekta

Vlasnicima pasa se štedi vrijeme i smanjuje stres, jer im omogućava da na jednostavan način pronađu pouzdanog šetača. Šetačima pasa pruža priliku promocije svoje usluge i povećanje broja klijenata. Integracijom modernih digitalnih servisa omogućuje jednostavan odabir termina i način plaćanja, čime projekt ispunjava sve potrebe korisnika.

Postojeća slična rješenja

Jedno od popularnijih već postojećih rješenja je aplikacija Wag!. Razlika u odnosu na ovaj projekt je u tome što predložena platforma nudi detaljnije informacije o psu, što omogućuje bolje usklađivanje psa i šetača.



Skup korisnika

Ciljna skupina korisnika ove platforme su vlasnici pasa koji zbog svojih obveza nemaju dovoljno vremena za svakodnevne šetnje. Sekundarna skupina su šetači koji žele promovirati svoje usluge i povećati broj klijenata.

Moguće nadogradnje projektnog zadatka

1. Mogućnost brige od drugim ljubimcima

Osim brige o psima, platforma bi mogla uključiti opciju pružanja usluga brige o ljubimcima druge vrste.

2. Dodatne usluge šetača

Osim pružanja usluge šetanja, mogu se pružiti usluge čuvanja ljubimca u svojem ili domu vlasnika, kratak posjet radi prehrane ili jednostavne provjere stanja ljubimaca,

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Registracija korisnika	Visok	Zahtjev dionika	Korisnik se može registrirati unosom korisničkog imena i lozinke, ili putem Google e-maila.
F-002	Prijava korisnika u sustav	Visok	Zahtjev dionika	Korisnik se uspješno prijavljuje pomoću korisničkog imena i lozinke, ili putem Google e-maila.
F-003	Odabir uloge korisnika	Visok	Zahtjev dionika	Korisnik prilikom prvog pristupa aplikaciji odabire želi li biti šetač pasa ili vlasnik pasa. Unosi osnovne podatke o sebi.
F-004	Brisanje profila korisnika	Visok	Zahtjev dionika	Korisnik uklanja svoj profil.
F-005	Plaćanje članarine	Visok	Zahtjev dionika	Šetač plaća mjesečnu ili godišnju članarinu za pristup platformi.
F-006	Mogućnost postavljanja termina šetnje	Visok	Zahtjev dionika	Šetač uređuje svoj popis oglašenih termina šetnji koristeći vanjsku uslugu kalendara. Za svaki termin šetnje šetač postavlja tip šetnje (individualna ili grupna), cijenu i trajanje.
F-007	Mogućnost registracije više pasa	Visok	Zahtjev dionika	Vlasnik može registrirati više pasa, pri čemu svaki pas ima svoj profil.
F-008	Uređivanje profila pasa	Visok	Zahtjev dionika	Svaki pas ima profil s podacima: ime, pasmina, starost, zdravstvene napomene, razina energije, socijalizacija s drugim psima i dopuštene poslastice.
F-009	Pregled i rezervacija termina šetnje	Visok	Zahtjev dionika	Vlasnik može pregledavati dostupne termine šetača putem vanjske usluge kalendara. Prilikom rezervacije vlasnik odabire: datum, vrijeme, trajanje šetnje, tip šetnje, polaznu adresu i dodane napomene.
F-010	Usluga plaćanja šetnje	Visok	Zahtjev dionika	Vlasnik može platiti šetnju gotovinom, PayPalom ili kreditnom karticom putem vanjskog platnog servisa.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-011	Mogućnost otkazivanja rezervacije	Visok	Zahtjev dionika	Vlasnik može otkazati rezerviranu šetnju najkasnije 24 sata prije početka termina.
F-012	Prihvat šetnje od strane šetača	Visok	Zahtjev dionika	Nakon što vlasnik rezervira termin, šetač prima obavijest i može prihvatiti ili odbiti šetnju.
F-013	Komunikacija između šetača i vlasnika	Visok	Zahtjev dionika	Nakon potvrde šetnje vlasnik i šetač mogu komunicirati putem integrirane web chat usluge. Tijekom šetnje šetač može poslati poruku ili fotografiju.
F-014	Pretraživanje šetača po kriterijima	Visok	Zahtjev dionika	Vlasnici imaju mogućnost pretražiti šetače prema lokaciji, cijeni i prosječnoj ocjeni korisnika.
F-015	Obavijest o novim šetačima	Visok	Zahtjev dionika	Vlasnici se mogu pretplatiti na obavijesti o novim šetačima koji se registriraju na platformi.
F-016	Sustav ocjena i recenzija	Visok	Zahtjev dionika	Nakon obavljene šetnje vlasnik može ocijeniti šetača ocjenom (1–5), dodati komentar i fotografije. Prosječna ocjena prikazuje se na profilu šetača.
F-017	Uloga administratora	Visok	Zahtjev dionika	Administrator sustava postavlja cijenu članstva i upravlja korisnicima.

Ostali Zahtjevi

Nefunkcijski zahtjevi

ID zahtjeva	Opis	Prioritet
NF-01	Aplikacija mora biti jednostavna (intuitivna) za korištenje	Visok
NF-02	Aplikaciju treba moći koristiti više korisnika u isto vrijeme	Visok
NF-03	Aplikaciji se pristupa preko HTTPS protokola	Visok
NF-04	Aplikacija treba pružati jednaku funkcionalnost i korisničko iskustvo bez obzira na uređaj i preglednik na kojem se koristi, razlučivost i veličinu ekrana	Visok
NF-05	Korisnik nepredviđenim korištenjem ne bi trebao moći na nijedan način poremetiti rad aplikacije	Visok

ID zahtjeva	Opis	Prioritet
NF-06	Sustav treba podržavati hrvatski jezik	Nisko

Dionici

U sustavu **PawPal** dionici su: administrator, vlasnik pasa, šetač pasa, neprijavljeni korisnik, razvojni tim i naručitelj.
Inicijatori su oni dionici koji izravno koriste sustav i pokreću procese, poput registracije, prijave, uređivanja profila, kreiranja termina šetnji, rezerviranja termina ili komunikacije između korisnika.
S druge strane, razvojni tim i naručitelj imaju sudioničku ulogu jer sudjeluju u izradi, nadzoru i evaluaciji sustava, ali ne pokreću procese unutar aplikacije.

U nastavku se nalazi pregled svih dionika s pripadajućim funkcionalnostima.

Tablica dionika

ID aktera	Naziv dionika	Uloga	Funkcionalnosti
A-1	Administrator	Inicijator	F-017
A-2	Vlasnik pasa	Inicijator	F-002, F-003, F-007, F-008, F-009, F-010, F-011, F-013, F-014, F-015
A-3	Šetač pasa	Inicijator	F-002, F-003, F-004, F-005, F-006, F-012, F-013, F-014, F-016
A-4	Neprijavljeni korisnik	Inicijator	F-001, F-002
A-5	Razvojni tim	Sudionik	-
A-6	Naručitelj	Sudionik	-

Kratko objašnjenje dionika

- **Administrator** - upravlja korisnicima i sustavom, određuje cijene i kontrolira komunikaciju i sigurnost.
- **Vlasnik pasa** - registrira pse, pregledava i rezervira termine, komunicira sa šetačima i uređuje svoj profil.
- **Šetač pasa** - nudi termine šetnji, uređuje profil, prihvaća rezervacije, komunicira s vlasnicima i prima ocjene.
- **Neprijavljeni korisnik** - može se registrirati i prijaviti u sustav.
- **Razvojni tim** - održava, razvija i nadograđuje funkcionalnosti sustava.
- **Naručitelj** - definira zahtjeve sustava i nadzire isporuku projekta.

Obrasci uporabe

Opis obrazaca uporabe

UC-001 – Registracija korisnika

- Glavni sudionik: Korisnik
- Cilj: Korisnik se može registrirati u sustav.
- Sudionici: Baza podataka, Vanjski servis za autentifikaciju
- Preduvjet: Korisnik nije registriran

Opis osnovnog tijeka

> 1. Korisnik otvara obrazac za registraciju.
> 2. Korisnik unosi korisničko ime i lozinku.
> 3. Sustav sprema podatke i potvrđuje uspješnu registraciju.

ILI

> 1. Korisnik pritisne na 'Sign in with Google'.
> 2. Korisnik odabire Google račun.
> 3. Korisnik odobrava pristup.
> 4. Sustav sprema podatke i potvrđuje uspješnu registraciju.

Opis mogućih odstupanja

> 1. 1. a) Ako korisnik ostavi neko polje prazno, sustav prikazuje poruku o obveznim poljima.
> 2. 1. b) Ako korisničko ime već postoji, sustav traži od korisnika da unese drugo.

UC-002 – Prijava korisnika u sustav

- Glavni sudionik: Korisnik
- Cilj: Korisnik se može prijaviti u aplikaciju.
- Sudionici: Baza podataka, Vanjski servis za autentifikaciju
- Preduvjet: Korisnik je registriran.

Opis osnovnog tijeka

> 1. Korisnik otvara ekran za prijavu.
> 2. Korisnik unosi korisničko ime i lozinku.
> 3. Sustav prijavljuje korisnika i prikazuje početnu stranicu.

ILI

> 1. Korisnik pritisne na 'Sign in with Google'.
> 2. Korisnik odabire Google račun.

3. Sustav prijavljuje korisnika i prikazuje početnu stranicu.

Opis mogućih odstupanja

1. 1. a) Ako su uneseni neispravni podaci, sustav prikazuje poruku o pogrešnoj prijavi.
2. 1. a) Ako korisnik zaboravi lozinku, sustav nudi mogućnost obnove lozinke.

UC-003 – Odabir uloge korisnika

- Glavni sudionik: Korisnik
- Cilj: Odabrati ulogu
- Sudionici: Baza podataka
- Preduvjet: Korisnik je prijavljen.

Opis osnovnog tijeka

1. Korisnik odabire ulogu (šetač ili vlasnik).
2. Korisnik potvrđuje odabir.

UC-004 – Brisanje korisničkog računa

- Glavni sudionik: Korisnik
- Cilj: Korisnik može obrisati vlastiti korisnički račun.
- Sudionici: Baza podataka
- Preduvjet: Korisnik je registriran.

Opis osnovnog tijeka

1. Korisnik odabire opciju za brisanje.
2. Korisnik unosi podatke u obrazac za brisanje računa.
3. Korisnik potvrđuje brisanje računa.
4. Sustav briše korisnički račun i obavještava korisnika o uspješnom brisanju.

Opis mogućih odstupanja

1. a) Ako korisnik unese nepotpune ili neispravni podatke, sustav prikazuje poruku o pogrešci.
2. a) Ako korisnik odustane od brisanja, sustav prekida proces i vraća ga na prethodni ekran.
3. a) Ako dođe do pogreške u bazi podataka, sustav obavještava korisnika da brisanje nije dovršeno i nudi ponovni pokušaj

UC-005 – Plaćanje mjesečne/godišnje članarine

- Glavni sudionik: Šetač
- Cilj: Platiti članarinu.
- Sudionici: Baza podataka, Vanjska usluga za plaćanje
- Preduvjet: Šetač nije platio članarinu.

Opis osnovnog tijeka

> 1. Šetač otvara svoj profil.
> 2. Šetač odabire opciju plaćanja mjesečne ili godišnje članarine.
> 3. Sustav preusmjerava korisnika na vanjsku platnu uslugu.
> 4. Šetač unosi podatke o plaćanju i potvrđuje transakciju.
> 5. Sustav zaprima potvrdu plaćanja i ažurira status članarine.
> 6. Sustav sprema promjene i ažurira javni profil.

Opis mogućih odstupanja

> 3. a) Ako vanjska platna usluga nije dostupna, omogućuje ponovni pokušaj kasnije.
> 4. a) Ako korisnik unese neispravne podatke o kartici, sustav traži ispravan unos.
> 5. a) Ako korisnik odustane od plaćanja, sustav prekida proces i ne ažurira članstvo.
> 6. a) Ako potvrda o uplati nije primljena, sustav označava plaćanje kao neuspješno i obavještava korisnika.

UC-006 – Unos šetačevih detalja o šetnji

- Glavni sudionik: Šetač
- Cilj: Definirati detalje za oglašene termine šetnji.
- Sudionici:Vanjska kalendarska usluga, Baza podataka
- Preduvjet: Šetač ima aktivan profil.

Opis osnovnog tijeka

> 1. Šetač stvara termin šetnje u svom kalendaru.
> 2. Šetač unosi tip šetnje (individualna ili grupna).
> 3. Šetač unosi cijenu i trajanje šetnje.
> 4. Sustav sprema detalje.

Opis mogućih odstupanja

> 3. a) Ako šetač unese nepotpune ili neispravni podatke, sustav prikazuje poruku o pogrešci.
> 4. a) Ako dođe do pogreške pri spremanju, sustav nudi ponovni pokušaj.

UC-007 – Registracija više pasa

- Glavni sudionik: Vlasnik
- Cilj: Vlasnik otvara više profila pasa unutar jednog korisničkog računa.
- Sudionici: Baza podataka
- Preduvjet: Vlasnik je prijavljen.

Opis osnovnog tijeka

1. Vlasnik otvara odjeljak s profilima pasa.
2. Vlasnik odabire opciju za dodavanje novog profila.
3. Vlasnik unosi osnovne podatke o psu.
4. Sustav sprema podatke i dodaje psa u profil vlasnika.

Opis mogućih odstupanja

3. a) Ako vlasnik unese nepotpune ili neispravni podatke, sustav prikazuje poruku o pogrešci.
4. a) Ako dođe do pogreške u bazi podataka, sustav nudi ponovni pokušaj spremanja.

UC-008 – Uređivanje profila pasa

- Glavni sudionik: Vlasnik
- Cilj: Vlasnik ima mogućnost ažurirati podatke o psu.
- Sudionici: Vlasnik, Baza podataka
- Preduvjet: Pas je registriran.

Opis osnovnog tijeka

1. Vlasnik otvara profil psa.
2. Vlasnik uređuje ime, pasminu, starost i zdravstvene napomene.
3. Vlasnik unosi razinu energije, socijalizaciju i dopuštene poslastice.
4. Vlasnik sprema promjene.
5. Sustav sprema podatke.

Opis mogućih odstupanja

2. a) Ako vlasnik unese nepotpune ili neispravni podatke, sustav prikazuje poruku o pogrešci.
3. a) Ako vlasnik odustane prije spremanja, sustav ne pohranjuje promjene.

UC-009 – Pregled i rezervacija termina šetnje

- Glavni sudionik: Vlasnik
- Cilj: Pregledati i rezervirati željeni termin šetnje.
- Sudionici: Vlasnik, Šetač, Vanjska kalendarska usluga, Baza podataka
- Preduvjet: Vlasnik ima registriranog psa.

Opis osnovnog tijeka

1. Vlasnik otvara kalendar šetača.
2. Vlasnik pregledava dostupne termine.
3. Vlasnik odabire termin i unosi detalje: datum, vrijeme, trajanje, tip šetnje, adresu i dodatne napomene.
4. Sustav potvrđuje rezervaciju i obavještava šetača.

Opis mogućih odstupanja

> 3. a) Ako vlasnik unese nepotpune ili neispravni podatke, sustav prikazuje poruku o pogrešci.

UC-010 – Plaćanje šetnje

- Glavni sudionik: Vlasnik
- Cilj: Vlasnik može platiti rezerviranu šetnju.
- Sudionici: Vlasnik, Vanjski platni servis
- Preduvjet: Šetnja je rezervirana.

Opis osnovnog tijeka

> 1. Vlasnik odabire način plaćanja.
> 2. Sustav preusmjerava korisnika na vanjski platni servis.
> 3. Vlasnik unosi podatke i potvrđuje uplatu.
> 4. Platni servis vraća potvrdu sustavu.
> 5. Sustav označava šetnju kao plaćenu.

Opis mogućih odstupanja

> 3. a) Ako vlasnik unese neispravne podatke o plaćanju, sustav traži ponovni unos.
> 4. b) Ako korisnik odustane od plaćanja, sustav prekida proces i vraća ga na ekran rezervacija.

UC-011 – Otkazivanje rezervacije

- Glavni sudionik: Vlasnik
- Cilj: Vlasnik može otkazati rezerviranu šetnju.
- Sudionici: Vlasnik, Baza podataka
- Preduvjet: Šetnja je rezervirana.

Opis osnovnog tijeka

> 1. Vlasnik otvara rezervaciju.
> 2. Vlasnik otkazuje rezervaciju.
> 3. Sustav otkazuje rezervaciju.
> 4. Sustav vraća uplatu.

Opis mogućih odstupanja

> 2. a) Ako vlasnik odustane od otkazivanja, proces se prekida i rezervacija ostaje aktivna.
> 3. a) Ako je preostalo manje od 24 sata do zakazane šetnje, sustav ne dopušta otkazivanje i prikazuje upozorenje.

UC-012 – Šetač prihvaća/odbija šetnju

- Glavni sudionik: Šetač
- Cilj: Šetač prihvaća ili odbija šetnju nakon rezervacije.
- Sudionici: Šetač, Vlasnik, Baza podataka
- Preduvjet: Postoji nova rezervacija.

Opis osnovnog tijeka

> 1. Sustav obavještava šetača o novoj rezervaciji.
> 2. Šetač pregledava detalje šetnje.
> 3. Šetač odabire opcije “Prihvati” ili “Odbij”.
> 4. Vlasnik dobiva obavijest o prihvatu rezervacije.

Opis mogućih odstupanja

> 2. a) Ako šetač ne odabere jednu od opcija u zadanom roku, sustav zatvara rezervaciju.

UC-013 – Komunikacija između šetača i vlasnika

- Glavni sudionik: Šetač, Vlasnik
- Cilj: Šetač i vlasnik mogu međusobno komunicirati.
- Sudionici: Šetač, Vlasnik, Vanjska chat usluga
- Preduvjet: Šetnja je potvrđena.

Opis osnovnog tijeka

> 1. Sustav omogućuje pristup chatu nakon potvrde šetnje.
> 2. Šetač i vlasnik razmjenjuju poruke i fotografije.

Opis mogućih odstupanja

> 1. a) Sustav upozorava korisnika u slučaju slanja nepodržanog formata ili veličine datoteke.

UC-014 – Pretraživanje šetača po kriterijima

- Glavni sudionik: Vlasnik
- Cilj: Vlasnik može pronaći šetača prema željenim kriterijima.
- Sudionici: Vlasnik
- Preduvjet: Vlasnik je prijavljen.

Opis osnovnog tijeka

> 1. Vlasnik otvara tražilicu šetača.
> 2. Vlasnik odabire kriterije: lokacija, cijena, prosječna ocjena.
> 3. Vlasnik može pregledati detalje profila šetača.

Opis mogućih odstupanja

> 3. a) Ako nema šetača koji odgovaraju kriterijima, sustav prikazuje obavijest o nedostatku rezultata.

UC-015 – Obavijest o novim šetačima

- Glavni sudionik: Vlasnik
- Cilj: Primati obavijesti o novim šetačima.
- Sudionici: Vlasnik, Baza podataka
- Preduvjet: Vlasnik ima aktivan profil.

Opis osnovnog tijeka

> 1. Vlasnik se pretplaćuje na obavijesti.
> 2. Sustav bilježi pretplatu.
> 3. Sustav šalje obavijest pretplaćenim vlasnicima.

Opis mogućih odstupanja

> 3. a) Ako vlasnik poništi pretplatu, sustav prestaje slati obavijesti.

UC-016 – Sustav ocjena i recenzija

- Glavni sudionik: Vlasnik
- Cilj: Vlasnik može ocijeniti šetača nakon šetnje.
- Sudionici: Vlasnik, Šetač, Baza podataka
- Preduvjet: Šetnja je završena.

Opis osnovnog tijeka

> 1. Vlasnik otvara dovršenu šetnju.
> 2. Vlasnik unosi ocjenu, komentar i opcionalno fotografije.
> 3. Sustav ažurira i prikazuje prosječnu ocjenu na profilu šetača.

Opis mogućih odstupanja

> 2. a) Ako vlasnik ne unese ocjenu, sustav ne dopušta slanje recenzije.

UC-017 – Upravljanje korisnicima i cijenom članstva

- Glavni sudionik: Administrator
- Cilj: Upravljati korisnicima i postaviti cijenu članstva.
- Sudionici: Administrator, Baza podataka
- Preduvjet: Administrator je prijavljen u sustav.

Opis osnovnog tijeka

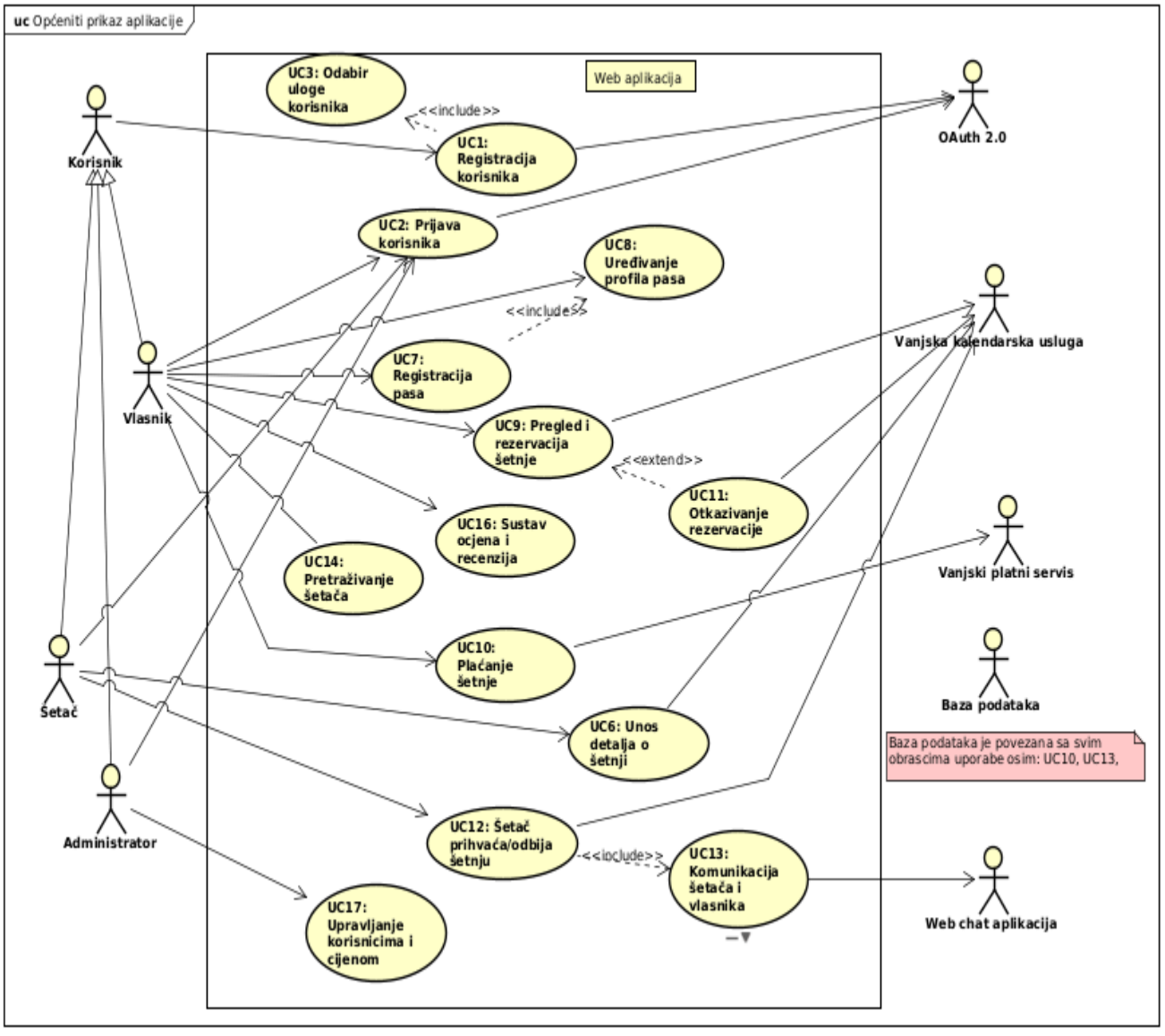
> 1. Administrator postavlja ili mijenja cijenu članstva.
> 2. Administrator pregledava popis korisnika.
> 3. Administrator briše korisničke račune.

Opis mogućih odstupanja

> 1. a) Ako administrator unese neispravan iznos, sustav prikazuje upozorenje.
> 2. a) Ako administrator slučajno odabere pogrešnog korisnika, sustav traži dodatnu potvrdu prije brisanja.

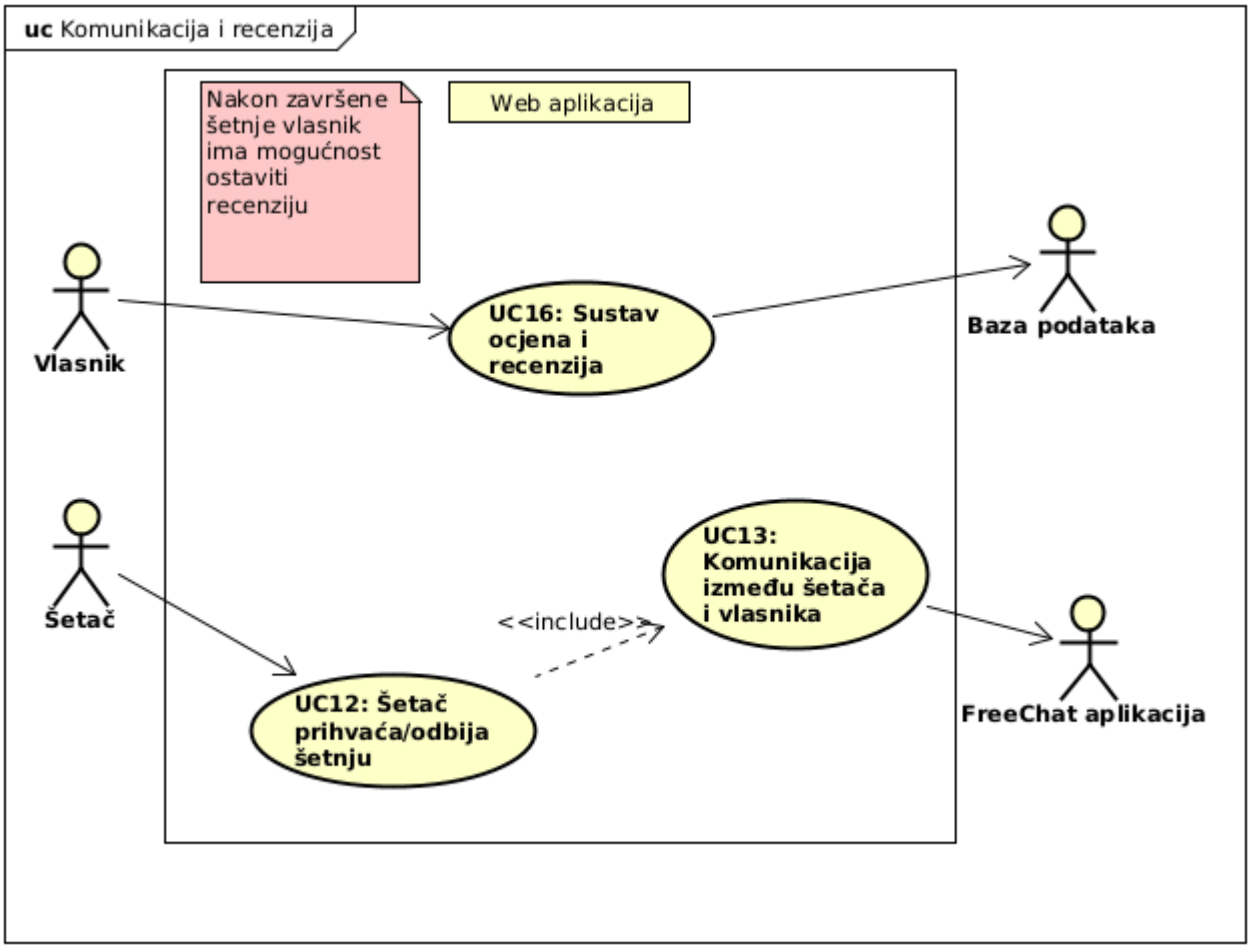
Dijagrami obrazaca uporabe

1. Visokorazinski dijagram obrazaca uporabe cijelog sustava

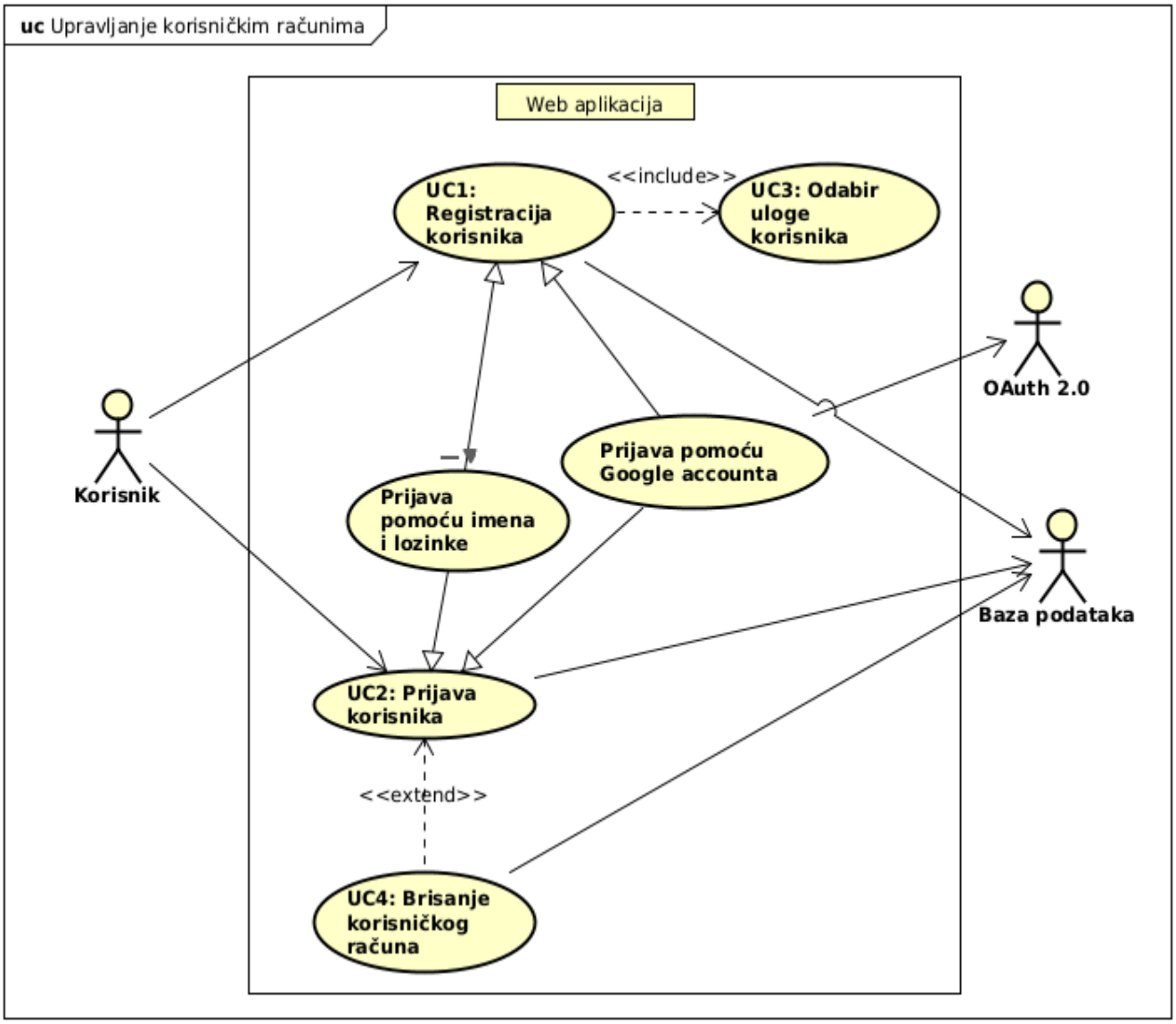


2. Dijagram obrazaca uporabe za ključne značajke

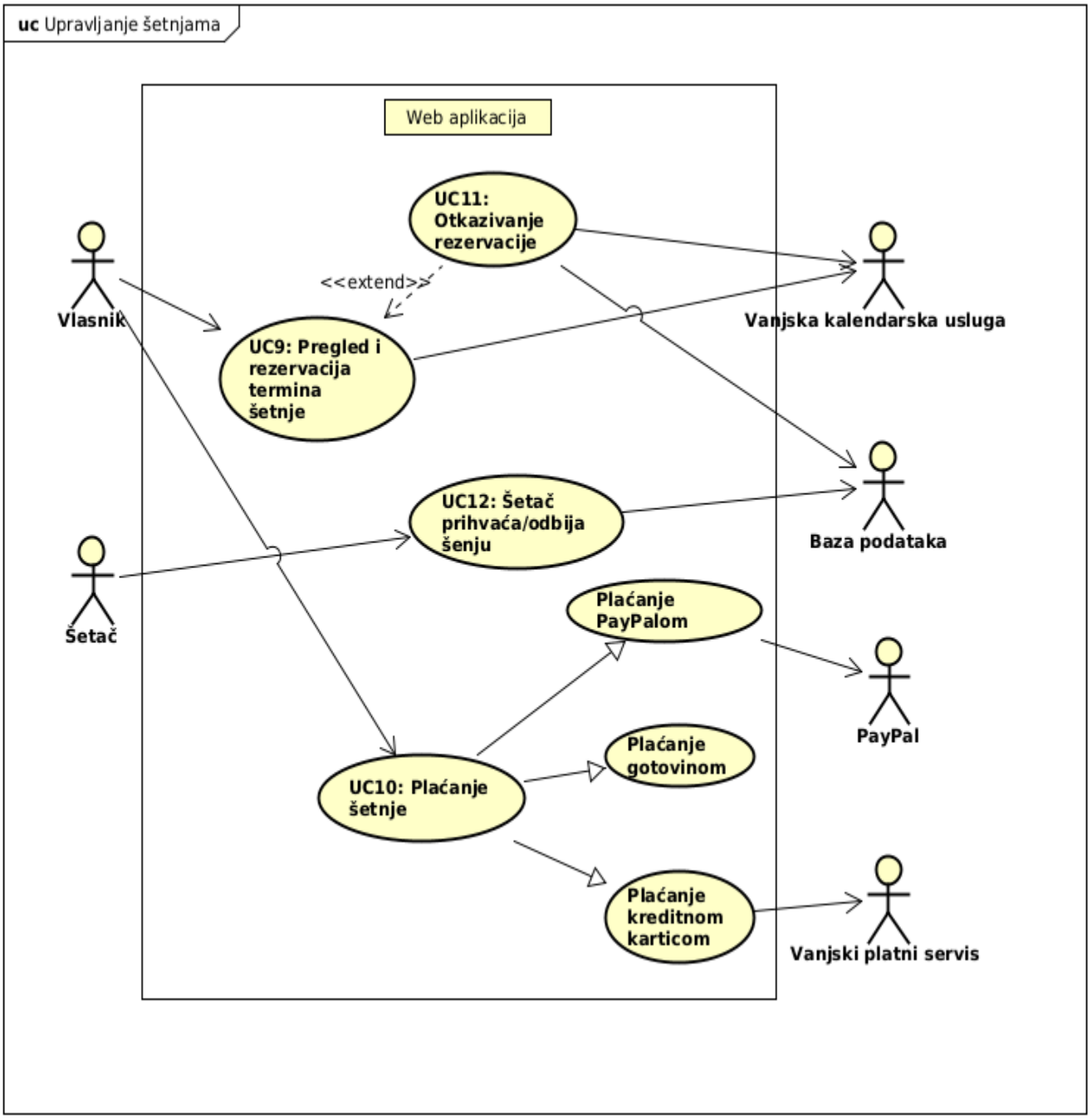
1.) Dijagram usluge komunikacije i recenzija



2.) Dijagram upravljanja korisničkim računima

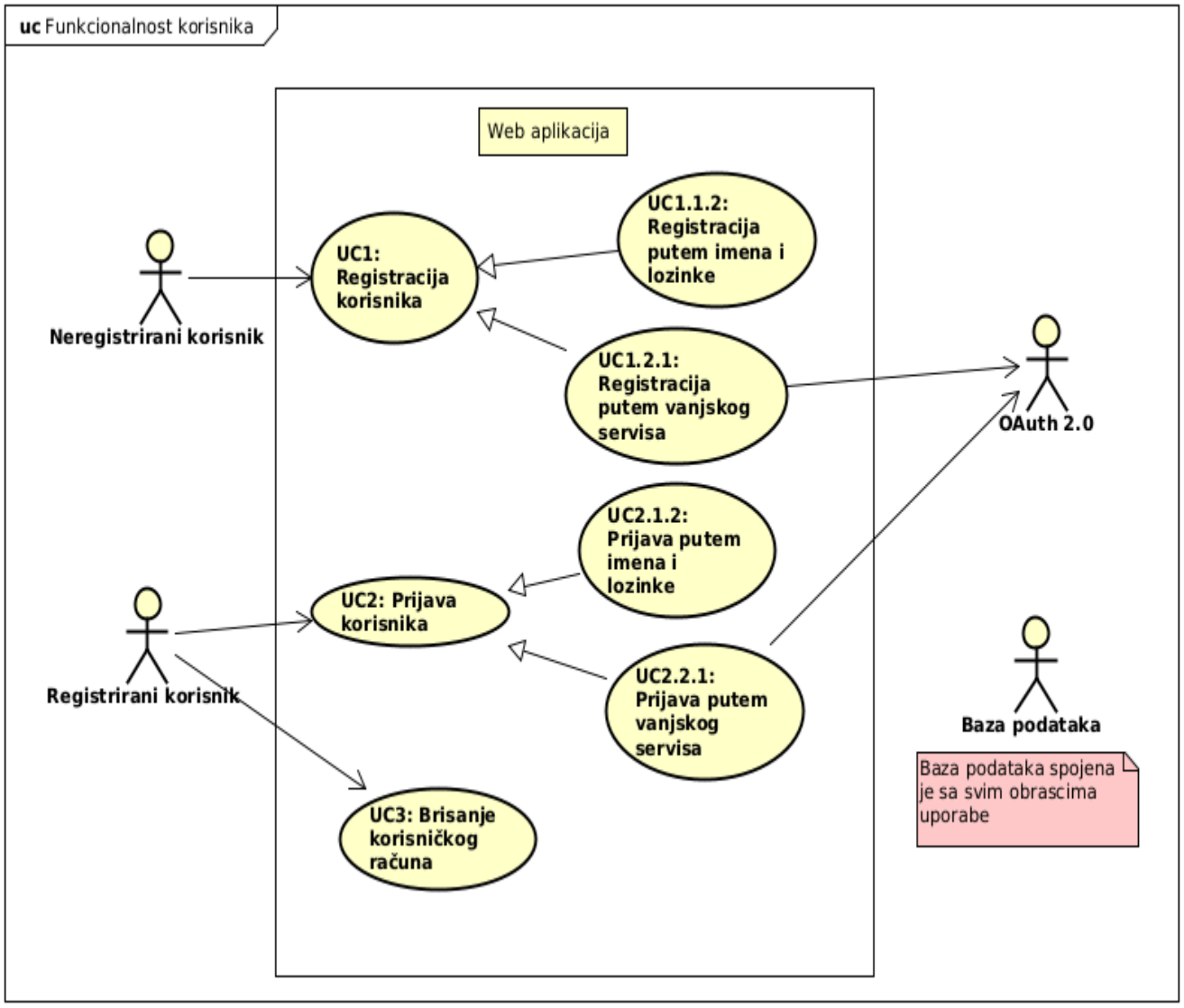


3.) Dijagram upravljanja rezervacijom šetnji

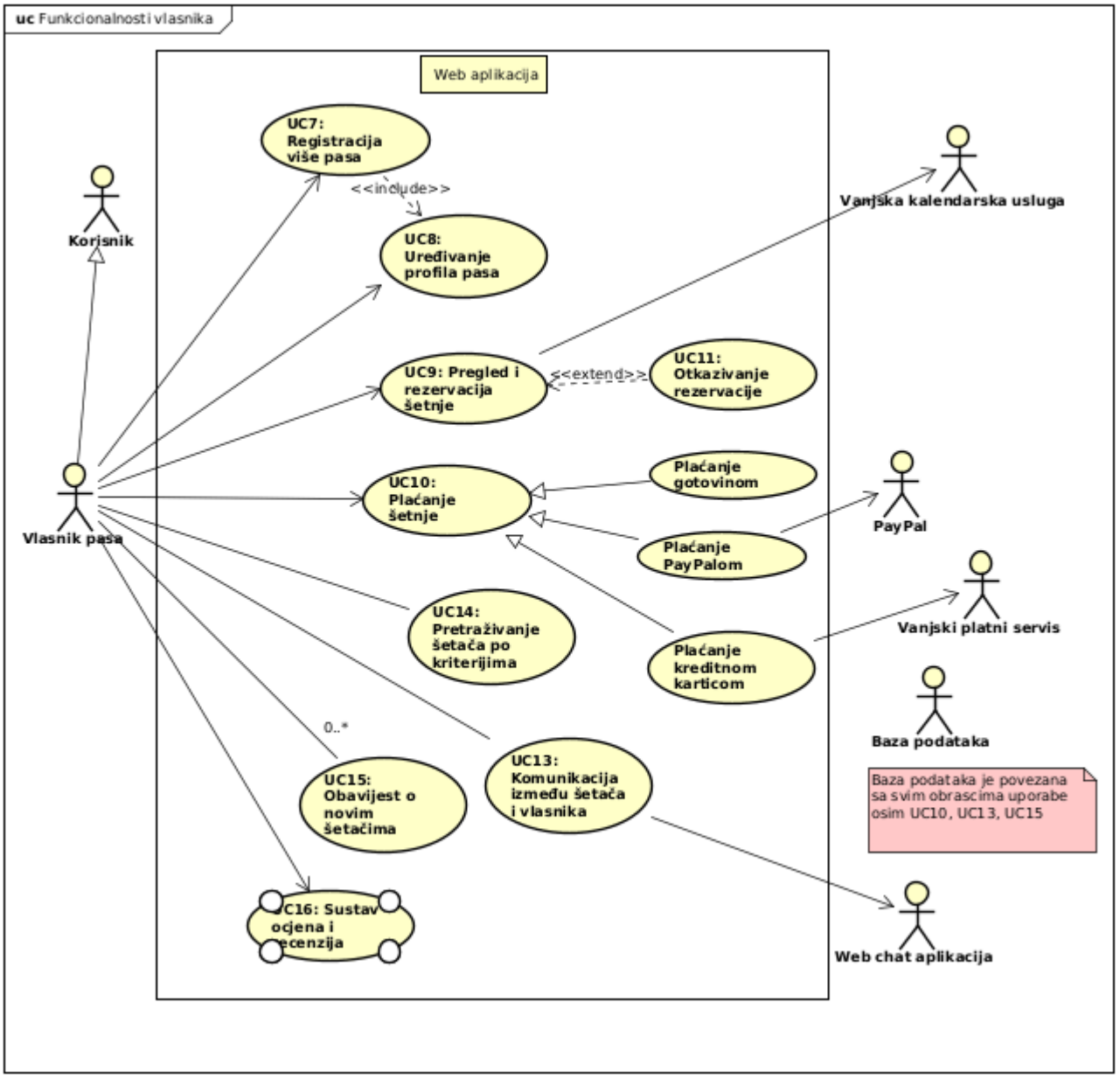


3. Dijagram obrazaca uporabe za korisničke uloge

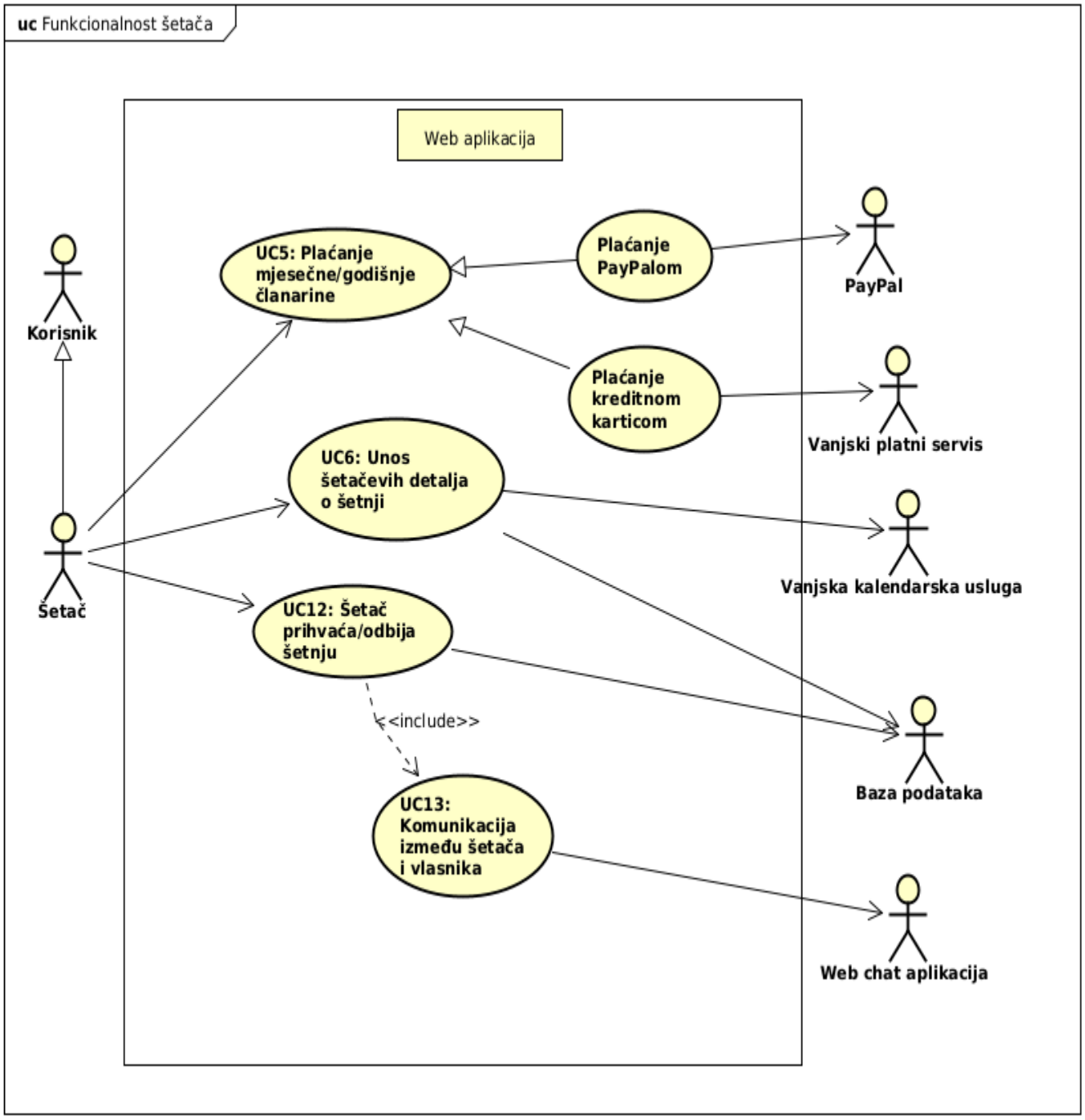
1.) UC dijagram za funkcionalnosti korisnika


uc Funkcionalnost korisnika
Web aplikacija
UC1: Registracija korisnika
UC1.1.2: Registracija putem imena i lozinke
UC1.2.1: Registracija putem vanjskog servisa
Neregistrirani korisnik
OAuth 2.0
UC2.1.2: Prijava putem imena i lozinke
UC2: Prijava korisnika
UC2.2.1: Prijava putem vanjskog servisa
Registrirani korisnik
Baza podataka
Baza podataka spojena je sa svim obrascima uporabe
UC3: Brisanje korisničkog računa

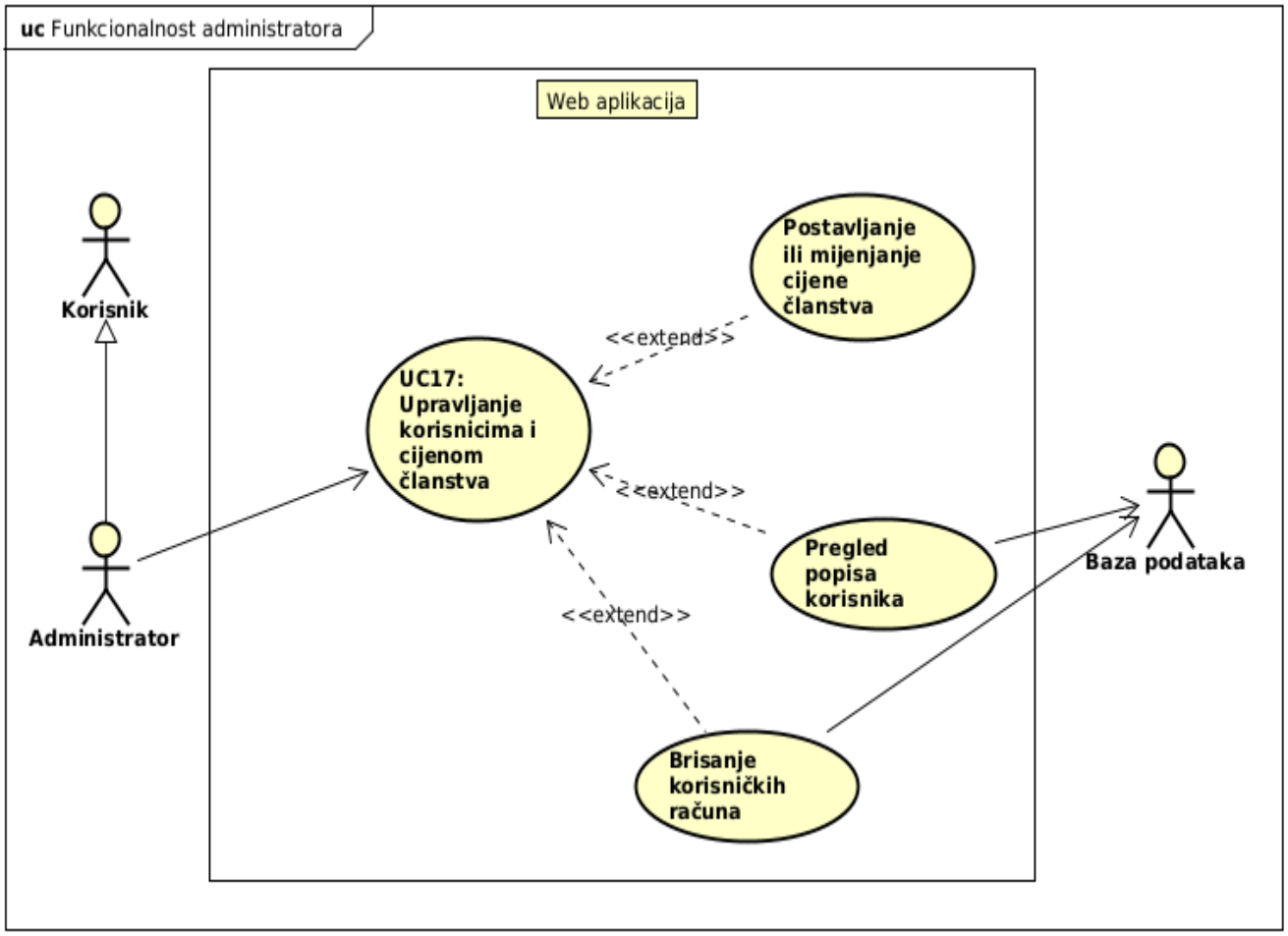
2.) UC dijagram za funkcionalnost vlasnika



3.) UC dijagram za funkcionalnost šetača

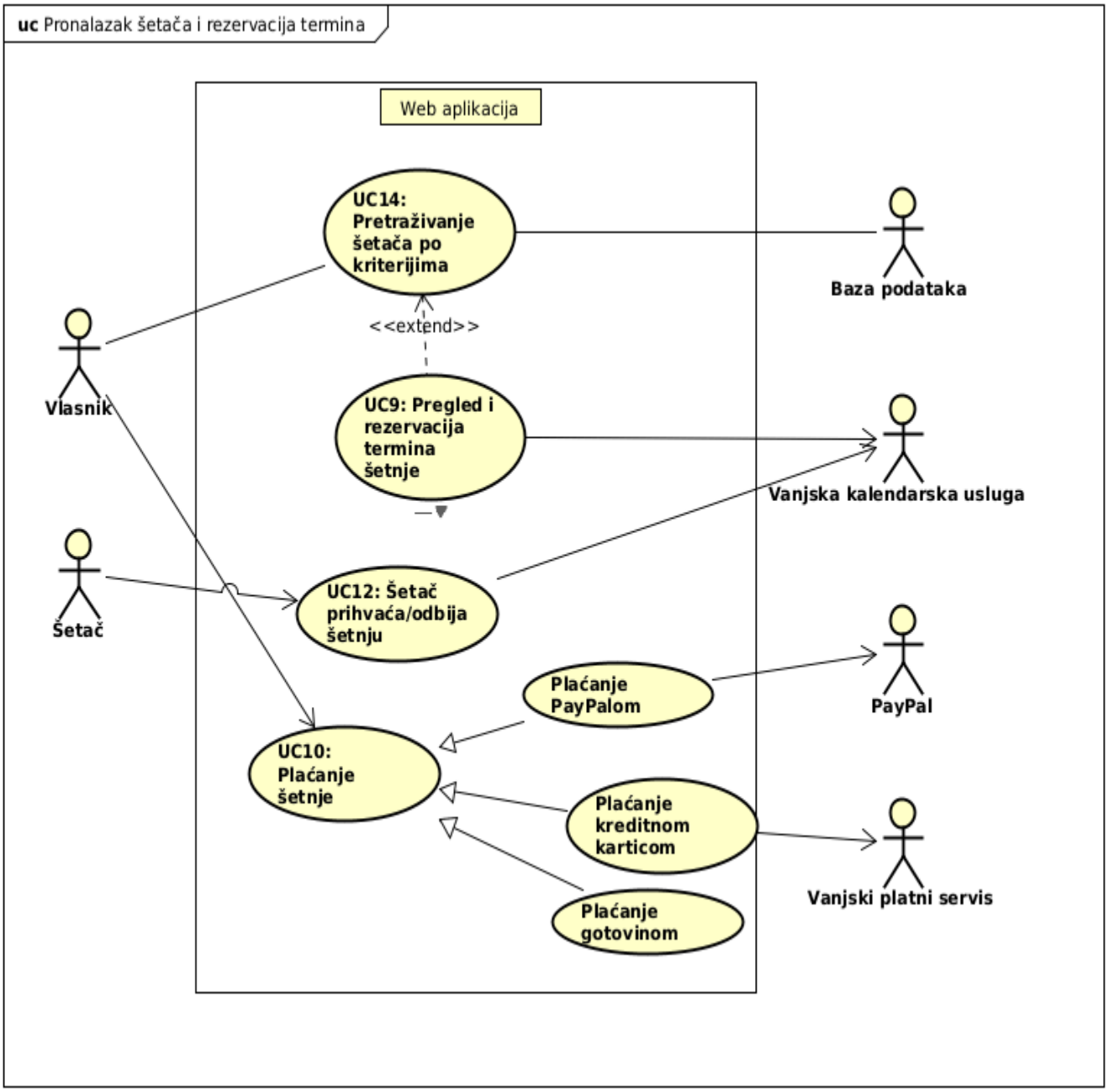


4.) UC dijagram za funkcionalnost administratora

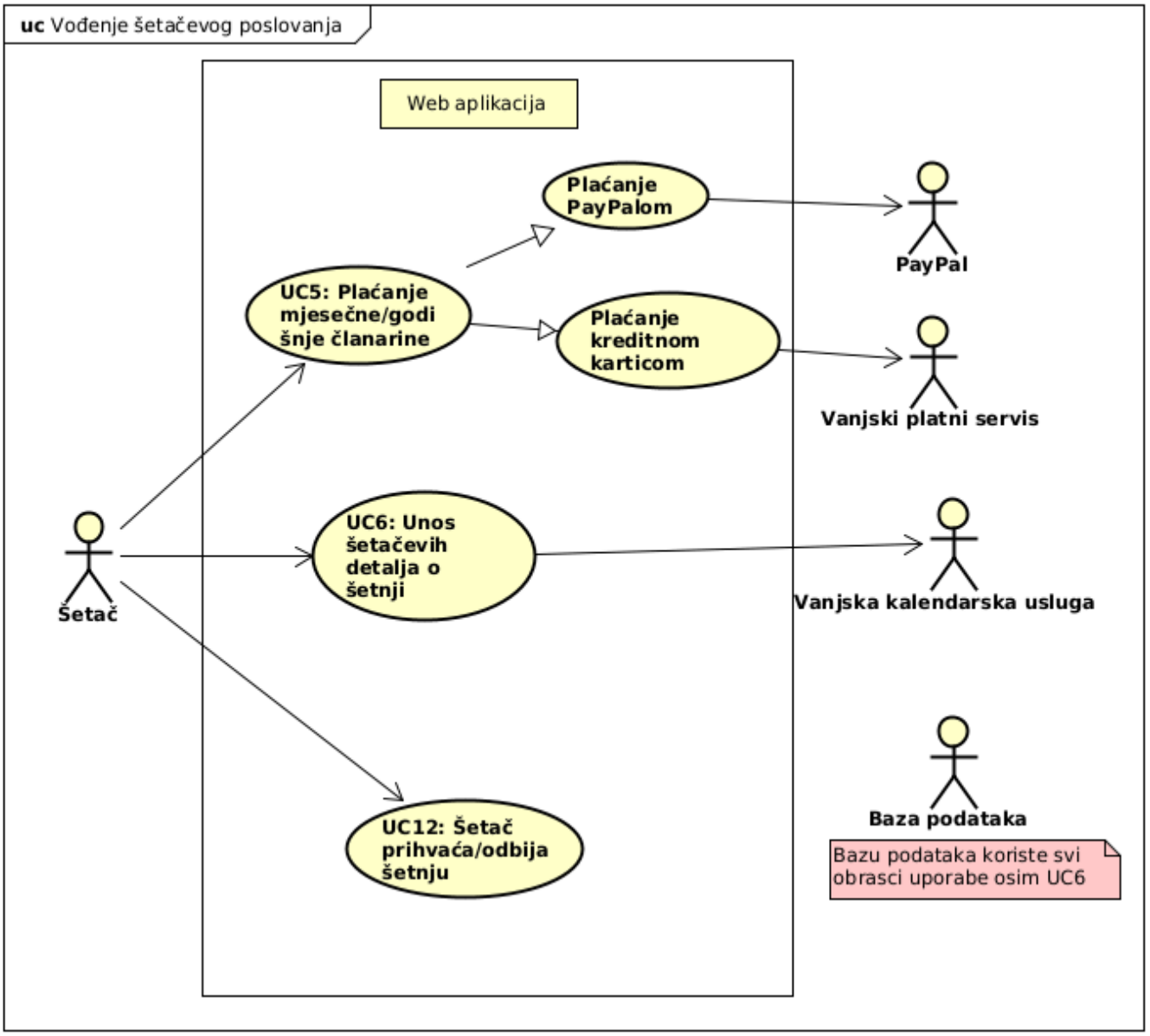


4. Dijagram obrazaca uporabe za osnovne poslovne procese

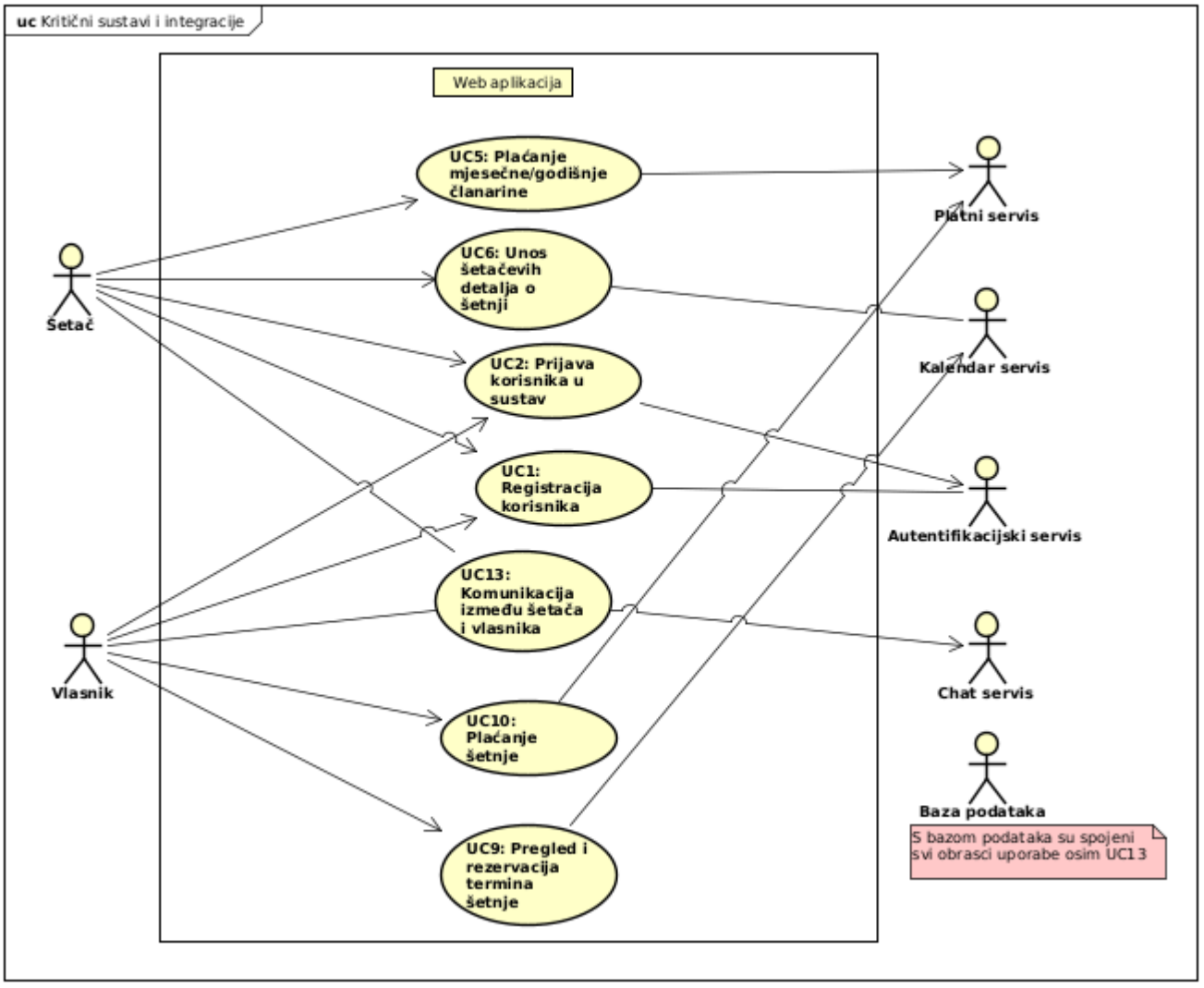
1.) Dijagram za pronalazak šetača i rezervaciju termina


uc Pronalazak šetača i rezervacija termina
Web aplikacija
UC14: Pretraživanje šetača po kriterijima
Baza podataka
<<extend>>
Vlasnik
UC9: Pregled i rezervacija termina šetnje
Vanjska kalendarska usluga
Šetač
UC12: Šetač prihvaća/odbija šetnju
Plaćanje PayPalom
PayPal
UC10: Plaćanje šetnje
Plaćanje kreditnom karticom
Vanjski platni servis
Plaćanje gotovinom

2.) Dijagram za vođenje šetačevog poslovanja



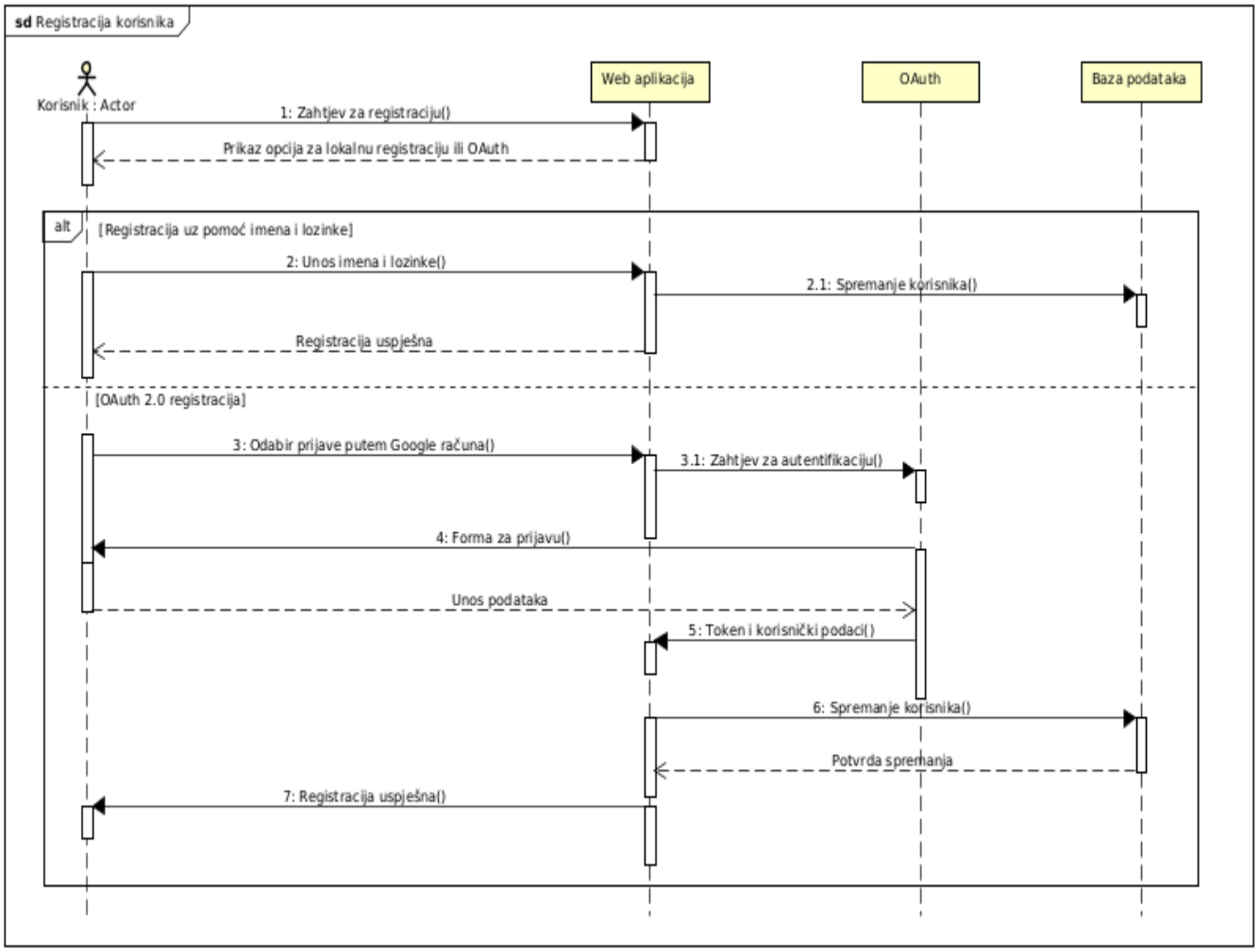
5. Dijagram obrazaca uporabe za kritične sustave i integracije



Sekvencijski dijagrami

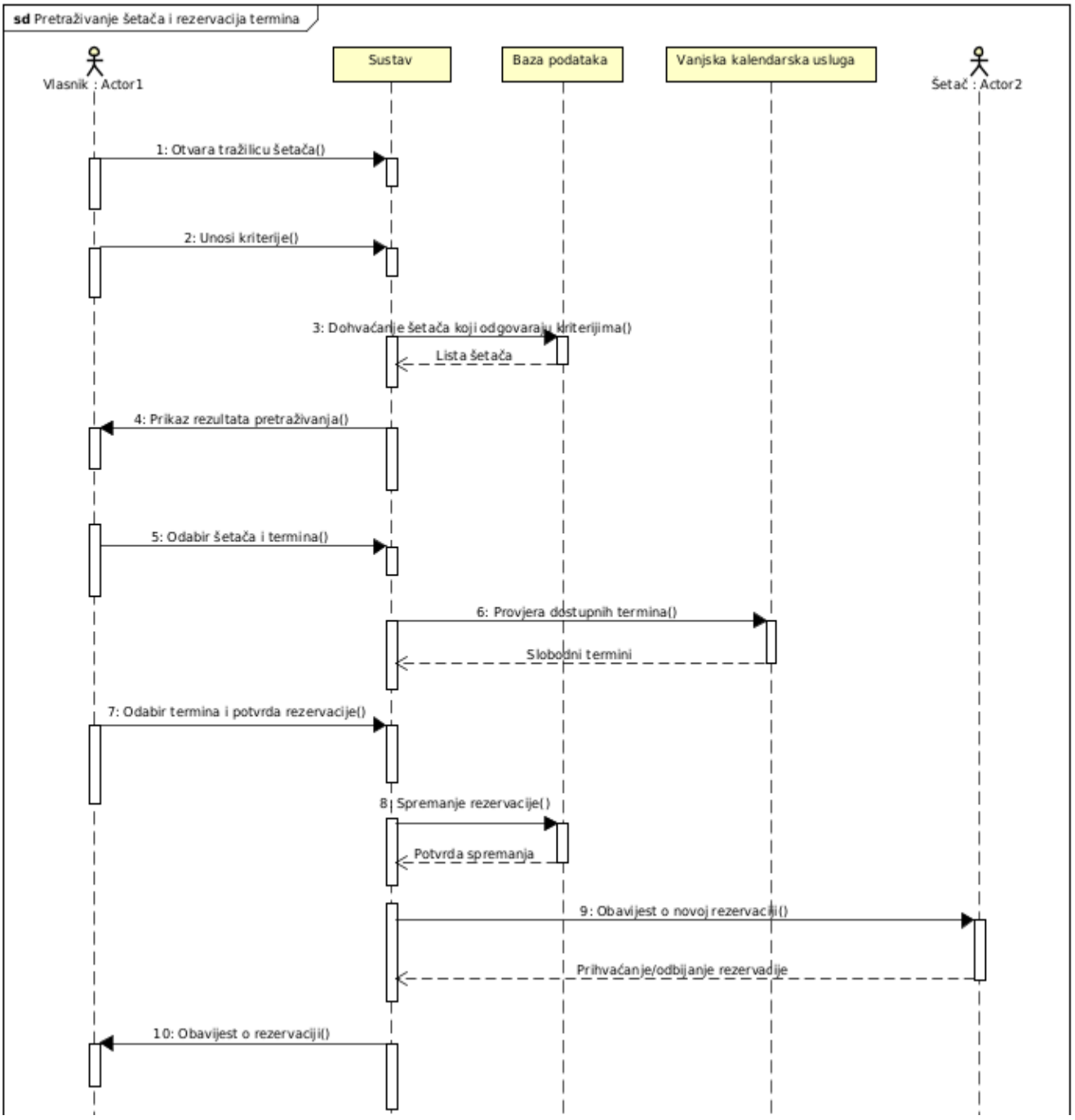
Funkcionalnosti za registraciju korisnika

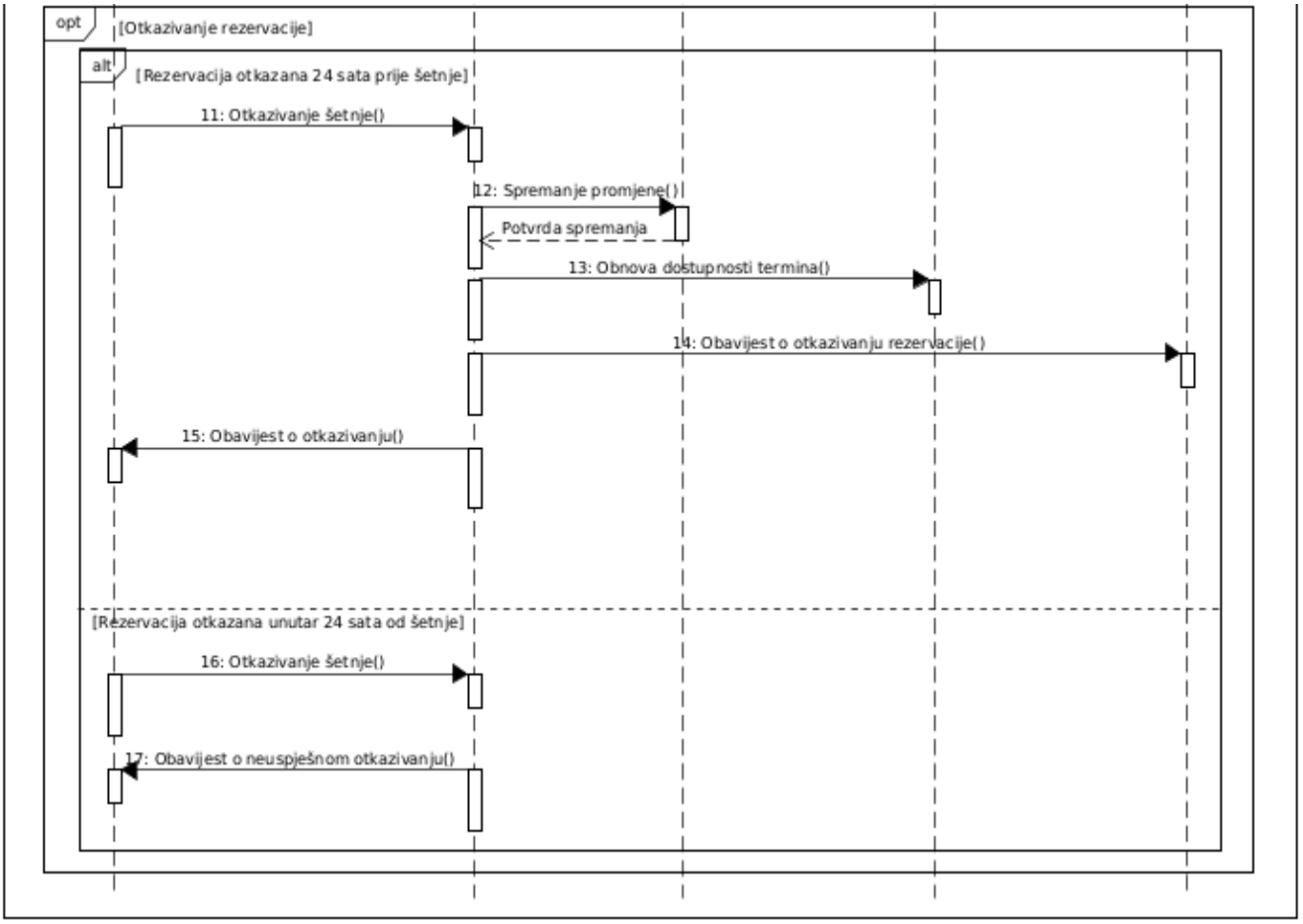
Sekvencijski dijagram prikazuje proces registracije korisnika u web aplikaciji s dvije metode autentifikacije. Proces započinje kada korisnik šalje zahtjev za registracijom web aplikaciji, koja mu nudi izbor između lokalne registracije ili OAuth autentifikacije. U prvom slučaju korisnik bira registracije s pomoću imena i lozinke koje unosi direktno u aplikaciju, nakon čega web aplikacija sprema podatke u bazu podataka, a baza vraća potvrdu spremanja. Konačno, aplikacija obavještava korisnika da je registracija uspješna. Alternativna opcija predstavlja OAuth 2.0 registraciju. Korisnik odabire prijavu putem Google računa, što pokreće zahtjev za autentifikacijom prema OAuth servisu. OAuth servis tada prikazuje formu za prijavu, gdje korisnik unosi svoje Google informacije. Nakon uspješne autentifikacije, OAuth generira pristupni token i korisničke podatke koje šalje web aplikaciji. Aplikacija zatim u sprema te podatke u vlastitu bazu, prima potvrdu spremanja, te zatim obavještava korisnika o uspješnoj registraciji.



Funkcionalnost za pretraživanje šetača i rezervaciju termina

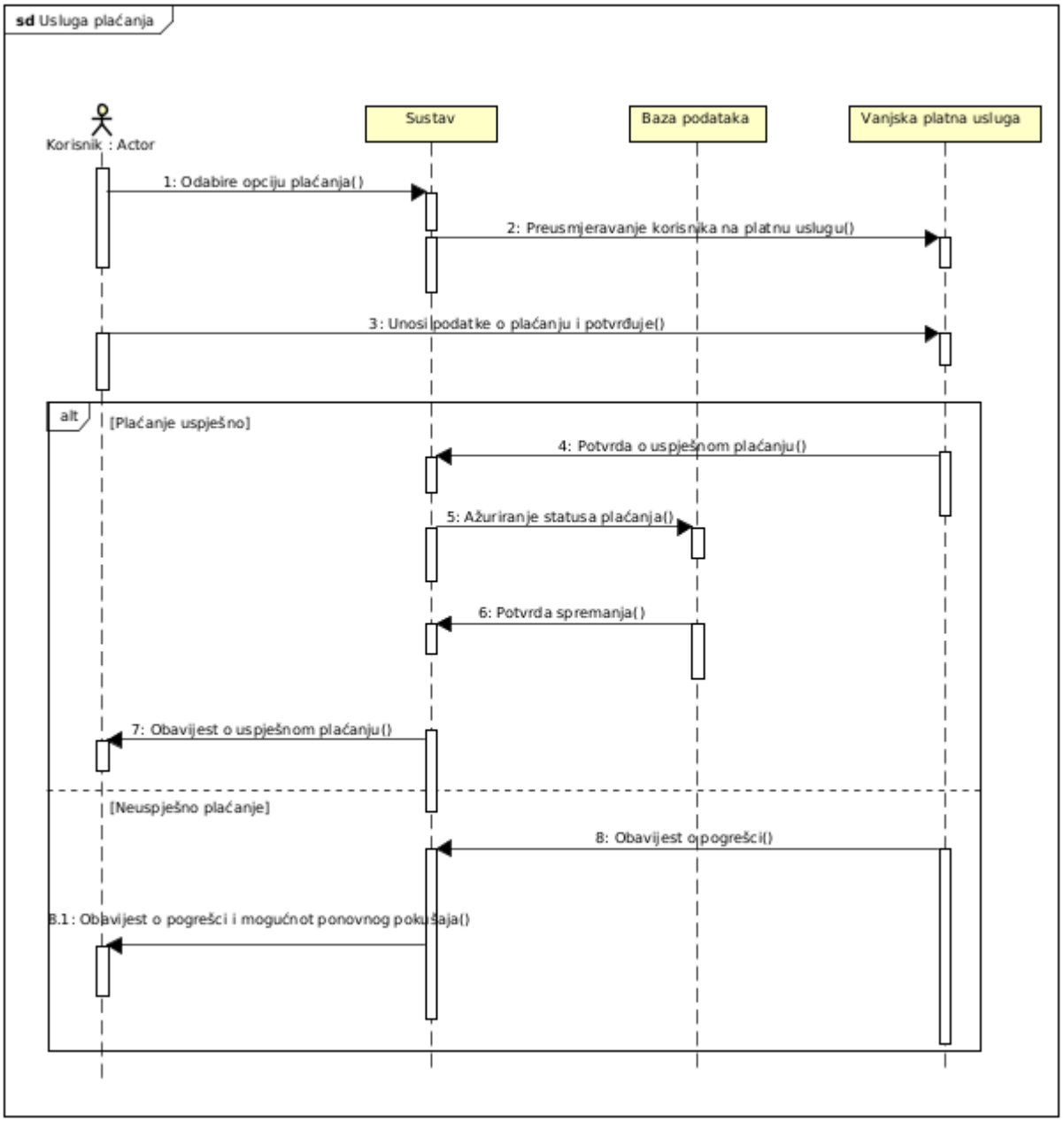
Sekvencijski dijagram prikazuje proces pretraživanja šetača, rezervaciju termina i otkazivanje istog. Proces započinje kada Vlasnik otvara pretraživanje i unosi kriterije. Sustav potom dohvaća odgovarajuće šetače iz baze podataka i prikazuje rezultate vlasniku. Nakon što vlasnik odabere šetača i termin, sustav provjerava dostupnost tog termina putem vanjske kalendarske usluge. Vlasnik zatim odabire termin i potvrđuje rezervaciju, koju sustav sprema u bazu podataka. Nakon uspješnog spremanja, sustav šetaču šalje obavijest o novoj rezervaciji. Nakon što šetač prihvati ili odbije rezervaciju, sustav obavještava vlasnika o statusu rezervacije.
Dijagram uključuje i opcionalni blok za otkazivanje rezervacije. U slučaju da je rezervacija otkazana 24 sata prije šetnje, sustav sprema promjenu, obnavlja dostupnost termina u vanjskoj kalendarskoj usluzi, te šalje obavijest o otkazivanju i šetaču i vlasniku. U slučaju da vlasnik pokuša otkazati rezervaciju unutar 24 sata do šetnje, sustav ga obavještava o neuspješnom otkazivanju.


sd Pretraživanje šetača i rezervacija termina
Vlasnik : Actor1
Sustav
Baza podataka
Vanjska kalendarska usluga
Šetač : Actor2
1: Otvara tražilicu šetača()
2: Unosi kriterije()
3: Dohvaćanje šetača koji odgovaraju kriterijima()
Lista šetača
4: Prikaz rezultata pretraživanja()
5: Odabir šetača i termina()
6: Provjera dostupnih termina()
Slobodni termini
7: Odabir termina i potvrda rezervacije()
8: Spremanje rezervacije()
Potvrda spremanja
9: Obavijest o novoj rezervaciji()
Prihvaćanje/odbijanje rezervadije
10: Obavijest o rezervaciji()



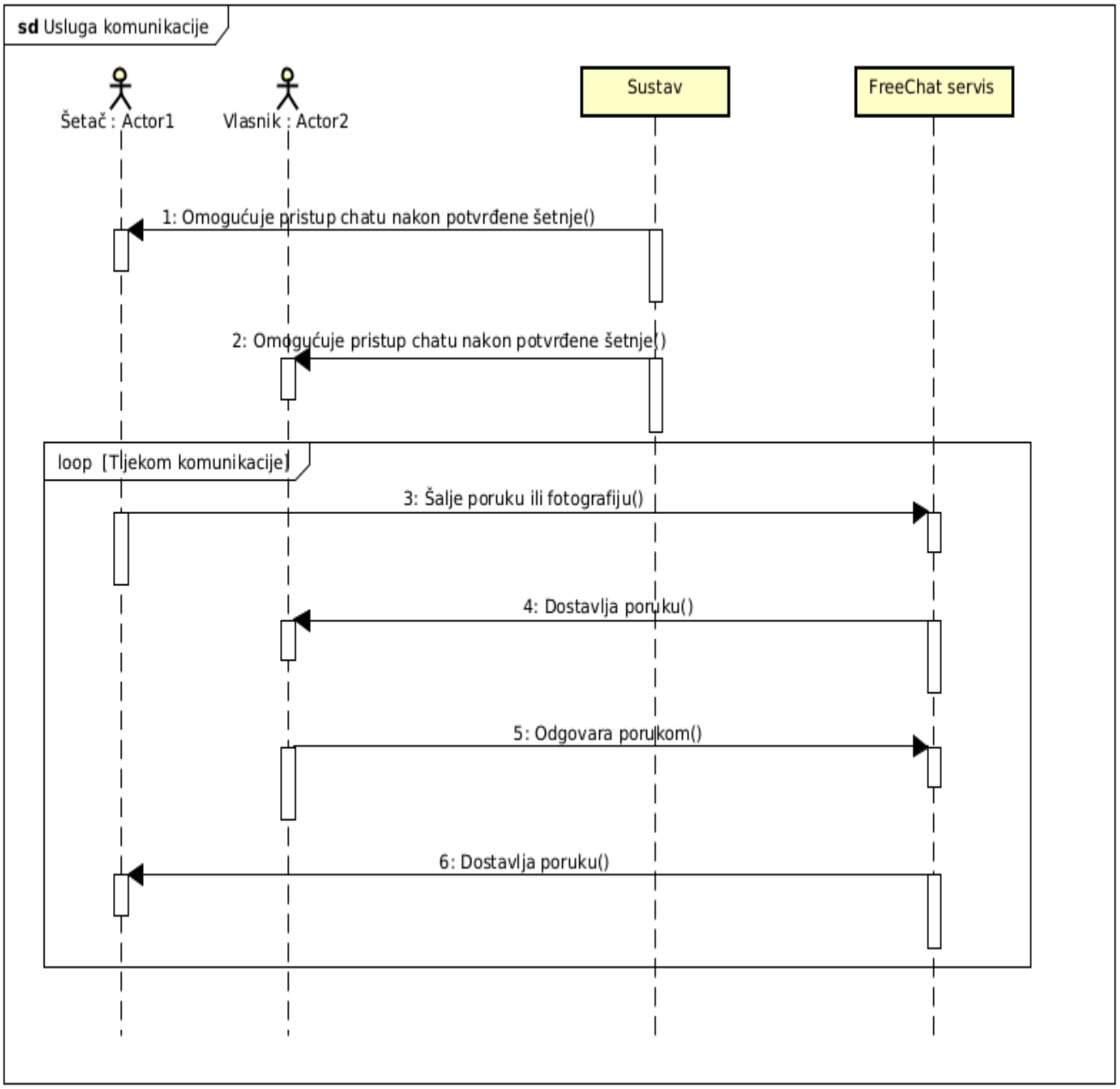
Funkcionalnost za uslugu plaćanja

Priloženi sekvencijski dijagram prikazuje proces usluge plaćanja. Proces započinje kada korisnik odabire opciju plaćanja i šalje zahtjev sustavu. Sustav zatim preusmjerava korisnika na vanjsku platnu uslugu radi autentifikacije i obrade plaćanja. U sljedećem koraku sustav unosi podatke o plaćanju i potvrđuje transakciju s vanjskom platnom uslugom.Dijagram dalje prikazuje dva alternativna scenarija. U prvom scenariju u kojem se plaćanje izvršava uspješno, vanjska platna usluga sustavu šalje potvrdu o uspješnom plaćanju. Sustav zatim ažurira status plaćanja u bazi podataka, koja potvrđuje spremanje promjena, i konačno obavještava korisnika o uspješno izvršenom plaćanju.
U alternativnom scenariju, gdje se transakcija nije obavila, vanjska platna usluga šalje obavijest o pogrešci u sustavu, koja zatim obavještava korisnika o grešci i mogućim razlozima o neuspješnom pokušaju plaćanja.



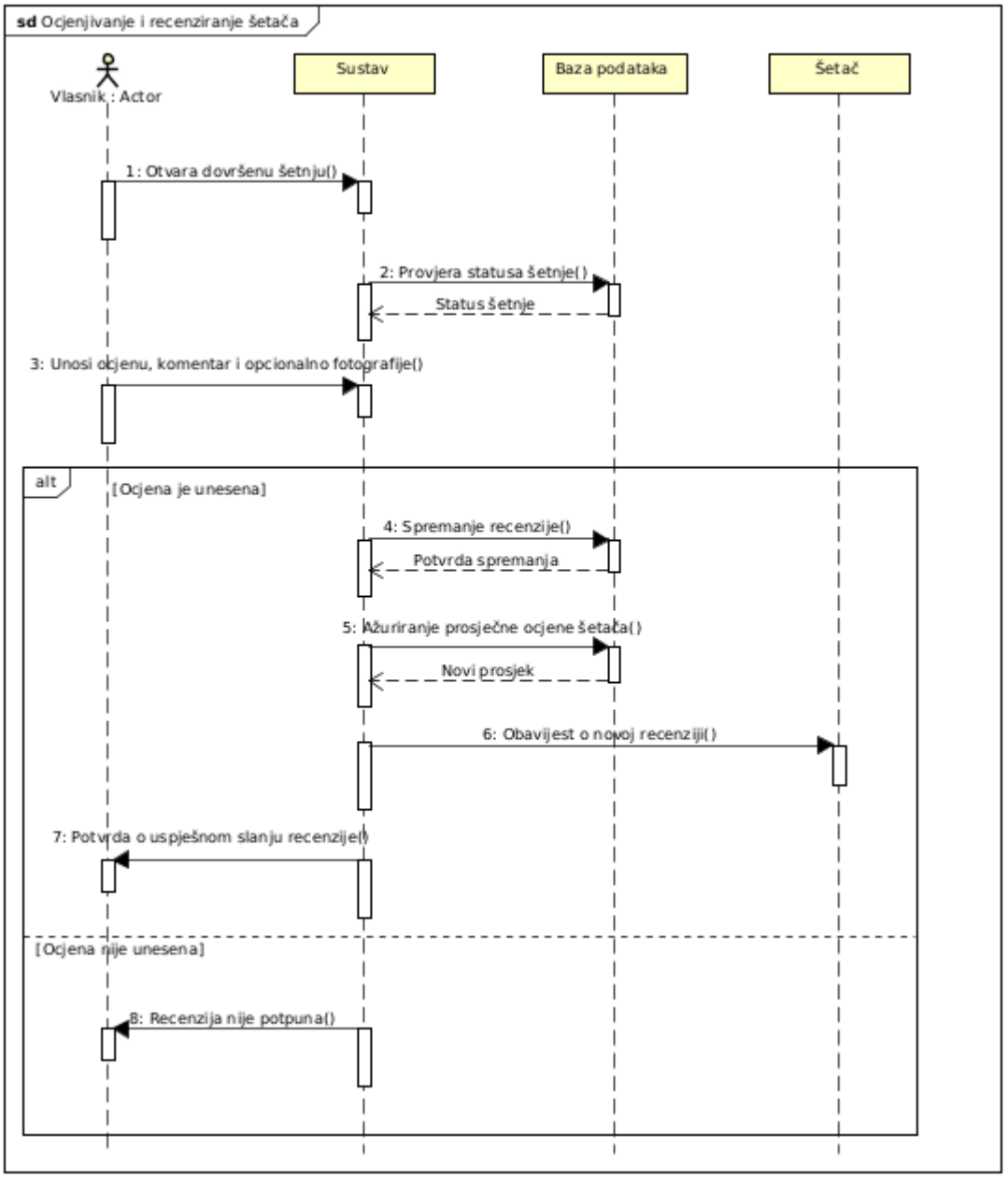
Funkcionalnost za međusobnu komunikaciju korisnika

Priloženi sekvencijski dijagram prikazuje proces usluge komunikacije između dva korisnika putem chat sustava. Proces započinje kada sustav omogućuje vlasniku i šetaču pristup chatu nakon što je šetnja potvrđena. Dijagram zatim prikazuje petlju koja predstavlja kontinuiranu razmjenu poruka između korisnika. Vlasnik šalje poruku ili fotografiju putem sustava prema FreeChat aplikaciji. Šetač zatim ima mogućnost pročitati i odgovoriti na primljenu poruku. Konačno, FreeChat aplikacija dostavlja odgovor natrag vlasniku, zatvarajući krug komunikacije. Ova petlja se može ponavljati neograničen broj puta tijekom trajanja razgovora, omogućujući dvosmjernu komunikaciju.



Funkcionalnost za ocjenjivanje i recenziju šetača

Priloženi sekvencijski dijagram prikazuje proces ocjenjivanja i recenziranja šetača. Proces započinje kada vlasnik u otvara dovršenu šetnju u sustavu. Sustav zatim provjerava status šetnje u bazi podataka, koja vraća informaciju o statusu šetnje, nakon čega vlasnik unosi ocjenu, komentar i opcionalnu fotografiju. Dijagram dalje prikazuje dva alternativna scenarija. U prvom scenariju u kojem vlasnik upisuje ocjenu u recenziju, sustav sprema unesenu recenziju u bazu podataka, koja potvrđuje uspješno spremanje. Sustav zatim ažurira prosječnu ocjenu šetača u bazi podataka, koja vraća novi izračunati prosjek i direktno šalje šetaču obavijest o novoj recenziji. Konačno, sustav vraća potvrdu vlasniku o uspješnom slanju recenzije. Alternativni scenarij pokriva situaciju kada vlasnik ne unese ocjenu. U tom slučaju, sustav šalje poruku vlasniku da recenzija nije potvrđena i proces se prekida bez spremanja ikakvih podataka.



Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

ID obrasca uporabe	Naziv obrasca uporabe	Obuhvaćeni funkcionalni zahtjevi
UC-001	Registracija korisnika	F-001
UC-002	Prijava korisnika u sustav	F-002
UC-003	Odabir uloge korisnika	F-003
UC-004	Brisanje korisničkog računa	F-004
UC-005	Plaćanje mjesečne/godišnje članarine	F-005

ID obrasca uporabe	Naziv obrasca uporabe	Obuhvaćeni funkcionalni zahtjevi
UC-006	Unos šetačevih detalja o šetnji	F-006
UC-007	Registracija više pasa	F-007
UC-008	Uređivanje profila pasa	F-008
UC-009	Pregled i rezervacija termina šetnje	F-009
UC-010	Plaćanje šetnje	F-010
UC-011	Otkazivanje rezervacije	F-011
UC-012	Šetač prihvaća šetnju	F-012
UC-013	Komunikacija između šetača i vlasnika	F-013
UC-014	Pretraživanje šetača po kriterijima	F-014
UC-015	Obavijest o novim šetačima	F-015
UC-016	Sustav ocjena i recenzija	F-016
UC-017	Upravljanje korisnicima i cijenom članstva	F-017

Arhitektura i dizajn sustava

Opis arhitekture

- **Stil arhitekture:** Klijent-poslužitelj (frontend: React, backend: .NET)
- **Glavne komponente:**
    - Autentifikacija i autorizacija
    - Upravljanje korisnicima
    - Upravljanje psima
    - Upravljanje šetnjama i rezervacijama
    - Komunikacija i obavijesti
    - Upravljanje plaćanjima i članarinama
    - Administracija sustava
- **Glavne karakteristike:**
    - Jasan razdvoj frontend i backend za lakše održavanje i razvoj
    - Frontend šalje HTTP zahtjeve backendu (REST API) koji upravlja logikom i bazom
    - JWT autentifikacija
    - Vanjski servisi za plaćanje i kalendar
    - PostgreSQL kao relacijska baza podataka

Obrazloženje odabira arhitekture

Aplikacija **PawPal** temelji se na klijent-poslužitelj arhitekturi koja jasno odvaja frontend i backend funkcionalnosti.

Ovako razdvojen sustav omogućava da se svaka komponenta aplikacije razvija, testira i održava neovisno. Ovakav pristup olakšava dugoročnu nadogradnju sustava, kao i potencijalnu migraciju pojedinih dijelova (npr. prelazak na mobilnu aplikaciju).

Backend aplikacije u potpunosti je odgovoran za autentifikaciju, autorizaciju, upravljanje korisnicima i poslovnu logiku, dok frontend korisniku prikazuje sučelje te komunicira s poslužiteljem preko HTTP zahtjeva.

Ovaka arhitektura dugoročno osigurava veću fleksibilnost za buduće nadogradnje sustava - primjerice, dodavanje modula za plaćanje, kalendare šetnji ili sustav ocjenjivanja korisnika.

Razmatrane alternativne arhitekture

U ranoj fazi razvoja razmatrali smo mogućnost implementacije klasične MVC arhitekture (Model-View-Controller), u kojoj su sve komponente integrirane u jedinstveni sustav.

Zbog toga smo odlučili koristiti modernu klijent-poslužitelj arhitekturu u kojoj React upravlja korisničkim sučeljem (kasnije faze), a backend je razvijen u Spring Bootu, što omogućuje jednostavno upravljanje sigurnošću, rad s bazom podataka i obradu poslovne logike.

Prednosti:

- Bolja razdvojenost odgovornosti
- Skalabilnost i lakše održavanje
- Mogućnost zamjene jednog dijela sustava bez utjecaja na ostatak

Nedostatak klasičnog MVC-a bio bi slabije korisničko iskustvo i manja fleksibilnost sučelja na različitim uređajima.

Organizacija sustava na visokoj razini

Sustav je podijeljen u tri osnovne cjeline:

- **Klijent**
  Korisničko sučelje izrađeno u HTML-u (Thymeleaf), u budućnosti zamjenjivo Reactom. Korisnik pristupa funkcijama poput registracije, prijave i odabira uloge.
- **Poslužitelj**
  Backend izrađen u Spring Bootu upravlja poslovnom logikom, autentifikacijom, obradom podataka i komunikacijom s bazom podataka.
- **Baza podataka**
  Relacijska baza (H2 u razvoju, PostgreSQL u produkciji) s tablicama za korisnike, šetače, vlasnike i pse. Entiteti su povezani putem JPA relacija (1:1, 1:N).

Organizacija aplikacije

Backend aplikacija

- **Kontroleri**
  - AuthController - upravlja registracijom, prijavom i postavljanjem sesije
  - ProfileController - postavlja korisničku ulogu (OWNER ili WALKER)
  - UserController - dohvaća popis korisnika
- **Servisi**
  - RegistrationService - logika registracije korisnika
  - SetupService - logika postavljanja uloge i povezivanja korisnika s profilima
- **Repozitoriji**
  - UserRepository - rad s tablicom users
  - OwnerRepository - spremanje i dohvat OWNER entiteta
  - WalkerRepository - spremanje i dohvat WALKER entiteta
- **Entiteti**
  - UserEntity - osnovni korisnik (username, lozinka, uloga)
  - OwnerEntity - vlasnik psa (povezan s DogEntity)
  - WalkerEntity - šetač psa
  - DogEntity - podaci o psu (pasmina, dob, energija...)

Frontend aplikacija

- HTML predlošci koriste Thymeleaf
    - registration.html - registracija korisnika
    - login.html - prijava korisnika
    - setup.html - odabir uloge i unos podataka
    - homepage.html - osnovna stranica (placeholder)
- CSS i statički sadržaji definirani u static/
- Sučelje omogućuje responzivan prikaz

Baza podataka

Odabrali smo relacijsku bazu podataka kao trajni sloj sustava. U razvoju koristimo H2, dok se u produkciji planira PostgreSQL zbog bolje sigurnosti, skalabilnosti i upravljanja transakcijama.

Opis tablica

users

Atribut	Tip podataka	Opis varijable
id	bigint	Primarni ključ
username	text	Jedinstveno korisničko ime
email	text	Email adresa korisnika
password	text	Lozinka (hashirana)
enabled	boolean	Status korisnika (aktivan/neaktivan)
role	text	Korisnička uloga: OWNER ili WALKER

owners

Atribut	Tip podataka	Opis varijable
username	text	FK → users.username (primarni ključ)

walkers

Atribut	Tip podataka	Opis varijable
username	text	FK → users.username (primarni ključ)
name	text	Ime šetača

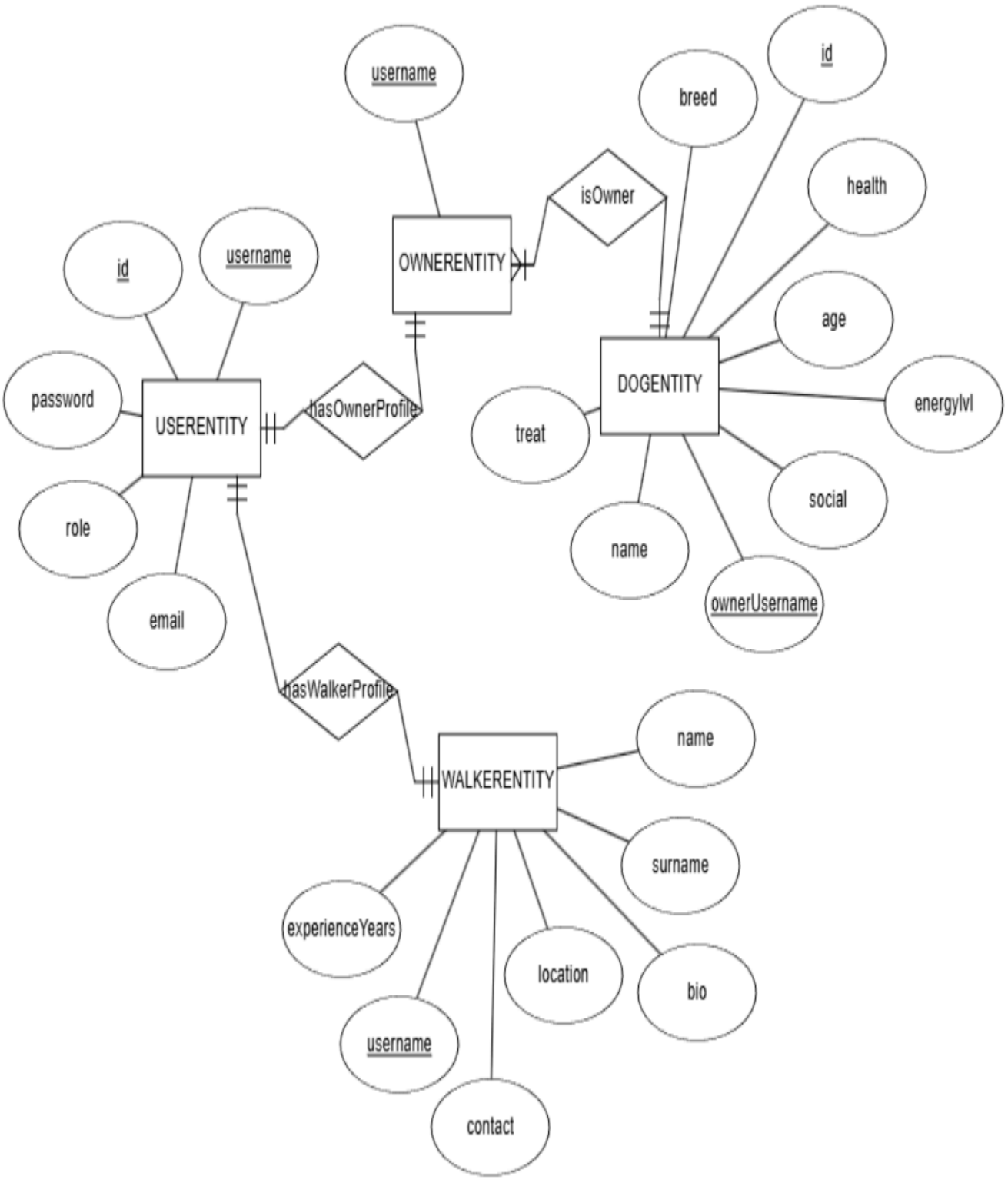
Atribut	Tip podataka	Opis varijable
surname	text	Prezime šetača
contact	text	Kontakt podaci
location	text	Grad ili područje rada
experienceYears	int	Broj godina iskustva
bio	text	Biografski opis

dogs

Atribut	Tip podataka	Opis varijable
id	bigint	Primarni ključ
name	text	Ime psa
breed	text	Pasmina
age	text	Dob psa
energylvl	text	Razina energije
treat	text	Omiljeni tretman
health	text	Zdravstveno stanje
social	text	Socijaliziranost
ownerUsername	text	FK → owners.username

ER dijagram baze podataka

ER dijagram baze podataka prikazuje osnovne entitete, njihove atribute te veze koje opisuju logičku strukturu podataka u aplikaciji PawPal.



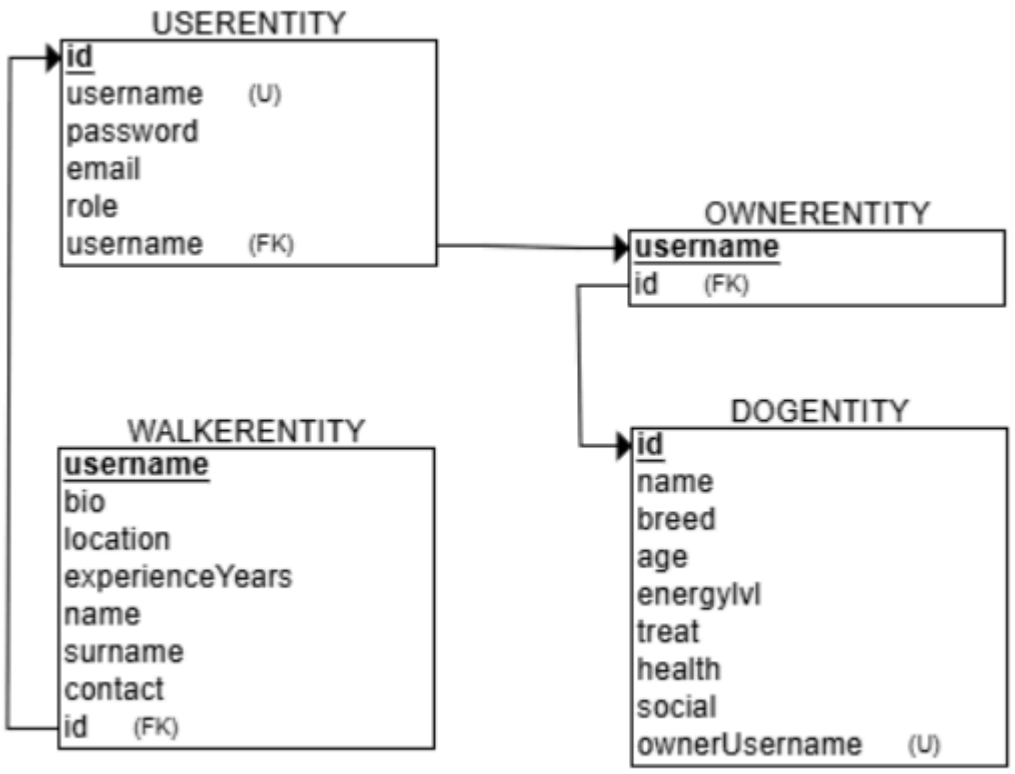
Temeljni entitet sustava je USERENTITY, koji predstavlja registriranog korisnika web aplikacije. Svaki korisnik pri registraciji odabire ulogu (OWNER ili WALKER), čime se određuje koje će dodatne podatke korisnik imati i koje funkcionalnosti može izvršavati.

Korisnik se identificira korisničkim imenom (username), koje se koristi i kao primarni ključ u entitetima OWNERENTITY i WALKERENTITY. Time se ostvaruje 1:1 odnos između osnovnog korisnika i njegovog proširenog profila.

Vlasnici pasa (OWNERENTITY) mogu imati jedan ili više pasa, što je prikazano kao 1:N veza prema DOGENTITY. Svaki pas pohranjen je u bazi s detaljima poput imena (name), dobi (age), pasmine (breed), zdravstvenog stanja (health) i razine energije (energylvl). Veza pas-vlasnik ostvarena je atributom ownerUsername, koji služi kao strani ključ.
Šetači pasa (WALKERENTITY) pohranjuju dodatne informacije specifične za tu ulogu, poput lokacije rada (location), biografskog opisa (bio), godina iskustva (experienceYears) i kontakt podataka.

REL shema baze podataka

Relacijska shema baze podataka, dobivena transformacijom ER dijagrama u relacijski model, prikazuje konkretne tablice koje će biti pohranjene u sustavu, njihove atribute, primarne i strane ključeve te kardinalitete veza.



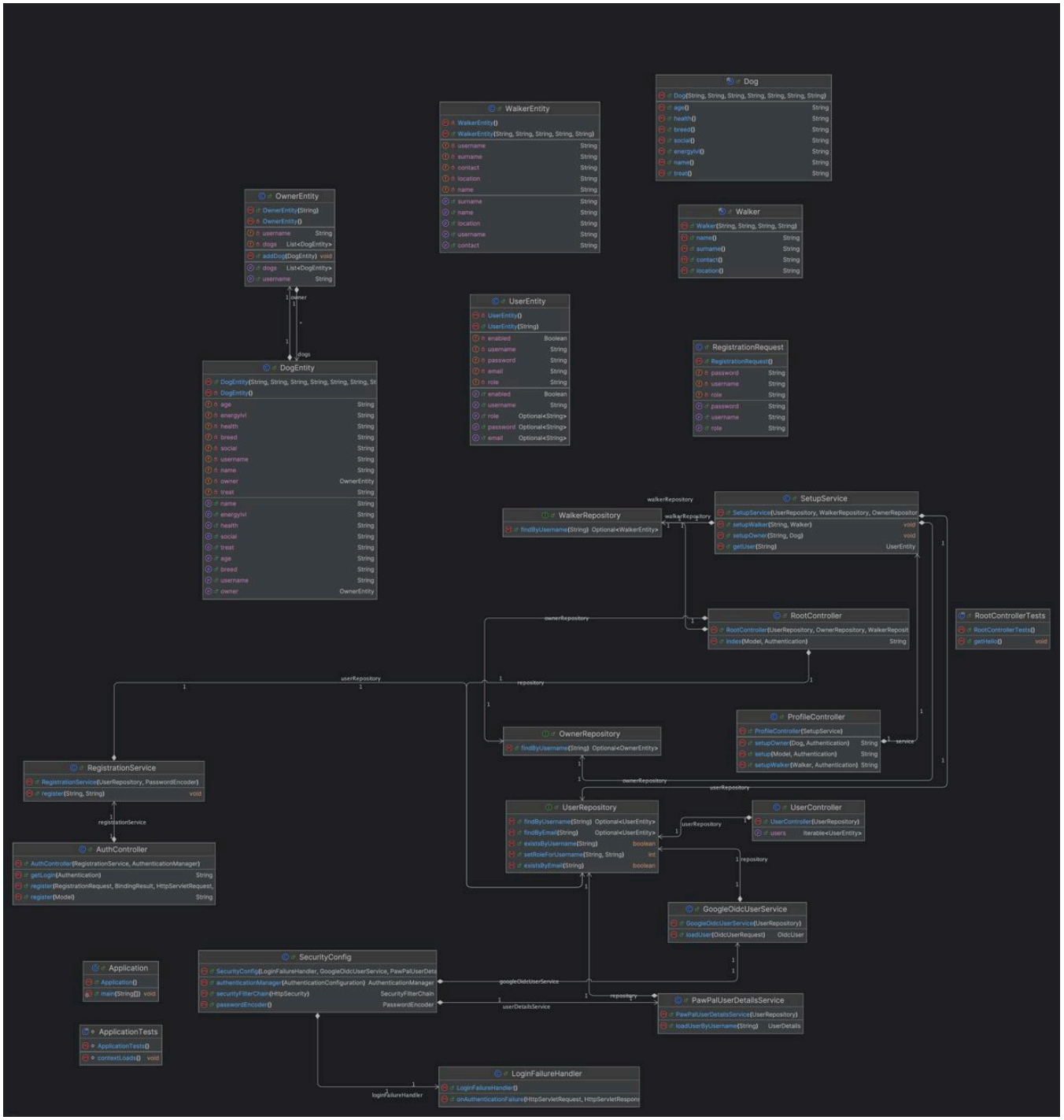
Tablica USERENTITY predstavlja središnju točku sustava te sadrži sve osnovne korisničke podatke. Njezini primarni atributi omogućuju autentifikaciju, autorizaciju i prikaz korisničkih profila u aplikaciji.

Specijalizirane uloge korisnika prikazane su kroz dvije dodatne tablice - OWNERENTITY i WALKERENTITY. Obje koriste atribut username kao primarni ključ i istovremeno strani ključ koji referencira tablicu USERENTITY. Time je osigurano da svaki vlasnik ili šetač nužno mora biti registrirani korisnik.

Tablica DOGENTITY sadrži sve informacije o psima te uključuje atribut ownerUsername kao strani ključ koji ostvaruje relaciju 1:N između vlasnika i njegovih pasa. Atributi pasa pohranjuju se u jednostavne tipove podataka, što omogućuje učinkovitu pohranu i dohvat.

Dijagram razreda

Dijagram razreda prikazuje strukturu backend sustava aplikacije PawPal, implementirane u Spring Bootu, te prikazuje komunikaciju između kontrolera, servisa, entiteta i repozitorija. Dijagram ilustrira kako su logičke komponente aplikacije povezane u cjelinu te kako pojedini razredi surađuju kako bi omogućili funkcionalnosti registracije, prijave te upravljanja profilima i podacima o psima.



Središnji dio dijagrama čine entiteti UserEntity, OwnerEntity, WalkerEntity i DogEntity, koji odgovaraju tablicama u bazi podataka. Svaki entitet mapira se putem JPA anotacija na relacijsku bazu i posjeduje odgovarajuće atribute i veze.

Servisni sloj sadrži poslovnu logiku - npr. validaciju podataka, provjeru postojanja korisnika, kreiranje profila nakon registracije ili dohvat informacija o psima. Ti servisi direktno komuniciraju s repozitorijima koji koriste Spring Data JPA za pristup bazi podataka.

Kontroleri (AuthController, ProfileController, UserController) izlažu REST API krajnjim korisnicima. Oni primaju HTTP zahtjeve, pozivaju metode servisnog sloja i vraćaju odgovarajuće JSON odgovore.
Dijagram razreda omogućuje vizualni uvid u arhitekturu sustava, prikazuje tok podataka između slojeva i potvrđuje da backend slijedi tipičan MVC princip (Model-View-Controller), čime se postiže modularnost, čitljivost i lako održavanje aplikacije.

Dinamičko ponašanje aplikacije

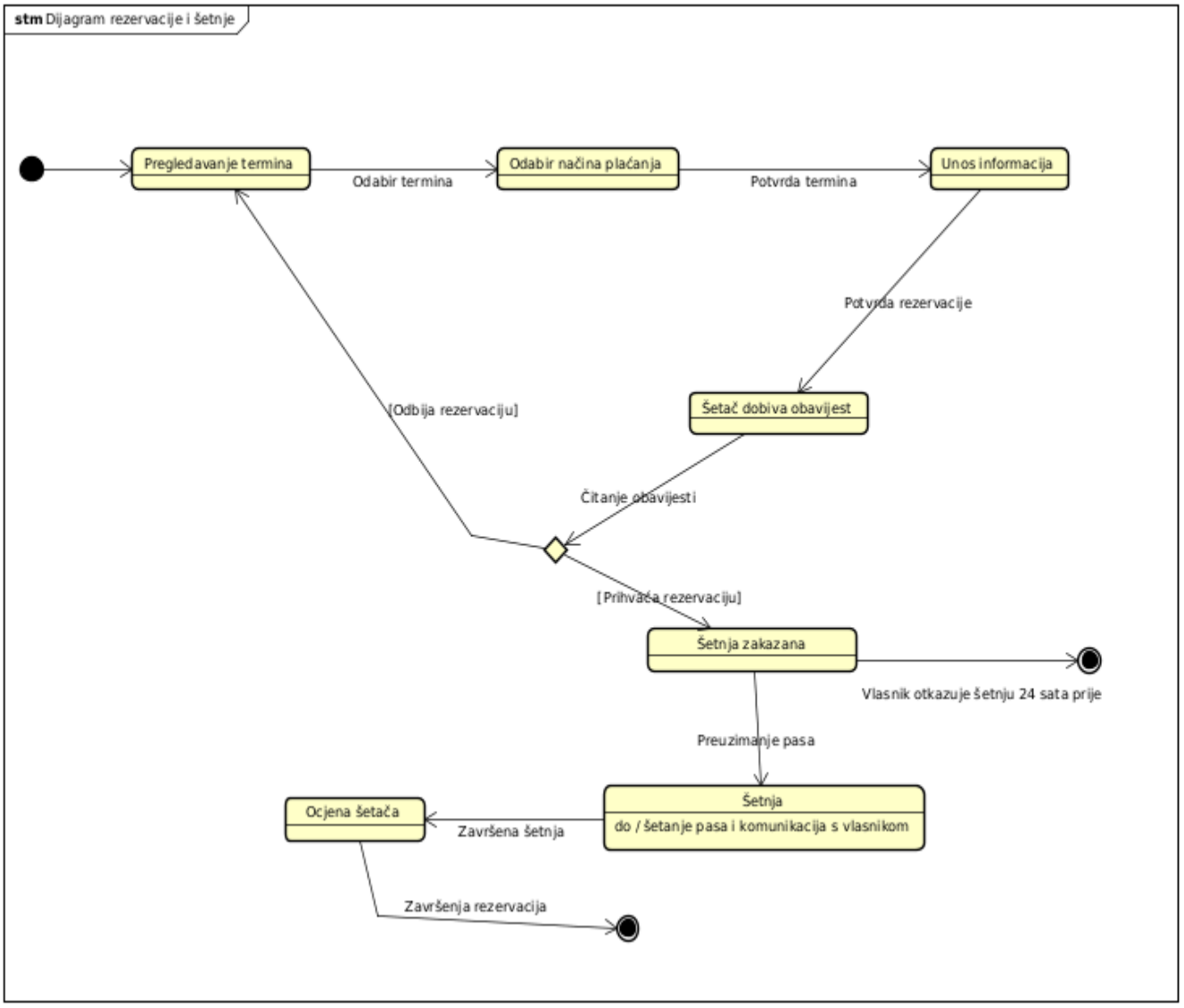
UML dijagrami stanja

Rezervacija termina i šetanje pasa

Korisnik započinje proces rezervacije šetnje pregledavanjem dostupnih termina. Nakon što odabere željeni termin, prelazi na odabir načina plaćanja. Kada su termin i način plaćanja potvrđeni, korisnik unosi potrebne podatke za rezervaciju. Time završava proces izrade rezervacije.

Nakon potvrde rezervacije, šetač dobiva obavijest o novoj rezervaciji.Šetač može pregledati obavijest i odlučiti hoće li rezervaciju prihvatiti ili odbiti. U slučaju da šetač odbije rezervaciju, proces rezervacije se prekida i završava bez zakazane šetnje.
Ako šetač prihvati rezervaciju, šetnja se smatra zakazanom. U tom trenutku vlasnik još uvijek ima mogućnost otkazati šetnju, ali najkasnije 24 sata prije dogovorenog termina, čime se proces također završava.

Ako rezervacija nije otkazana, u dogovoreno vrijeme dolazi do preuzimanja psa te započinje sama šetnja, tijekom koje se obavlja šetanje psa i komunikacija između šetača i vlasnika. Nakon završetka šetnje, vlasnik ima mogućnost ocijeniti šetača.



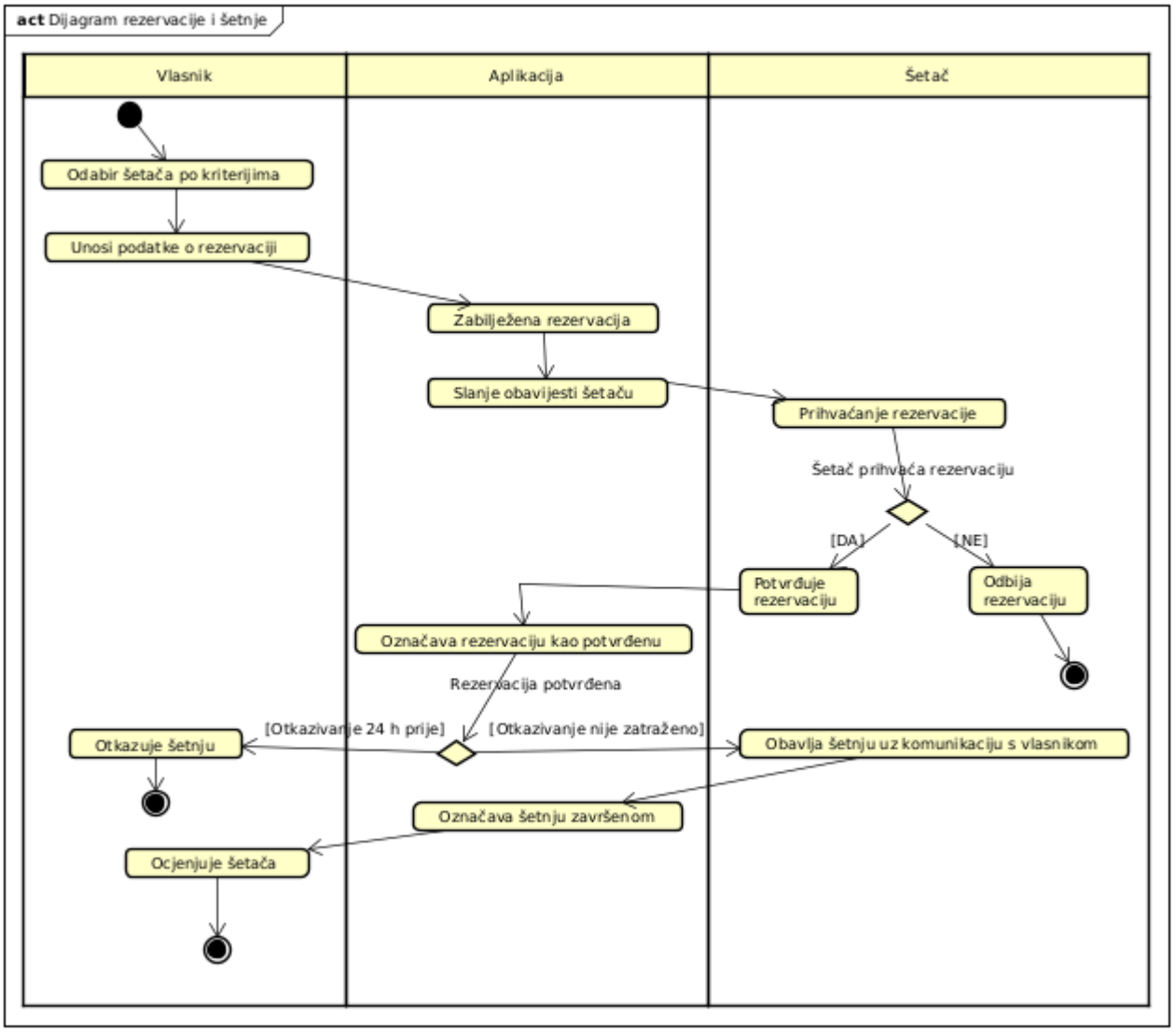
UML dijagrami aktivnosti

Rezervacija termina i šetanje pasa

Proces započinje tako da vlasnik pasa u aplikaciji odabire šetača prema zadanim kriterijima. Nakon odabira šetača, vlasnik unosi podatke o rezervaciji, čime se šalje zahtjev za šetnju. Aplikacija bilježi rezervaciju i automatski šalje obavijest odabranom šetaču. Šetač potom razmatra zahtjev i odlučuje hoće li rezervaciju prihvatiti ili odbiti. Ako šetač odbije, proces se prekida i šetnja se ne realizira.

Ako šetač prihvati rezervaciju, aplikacija je označava kao aktivnu. Rezervacija tada čeka termin šetnje, pri čemu vlasnik može otkazati najkasnije 24 sata prije početka. Ako vlasnik otkaže unutar tog roka, proces se završava.

Ako otkazivanje nije zatraženo, šetač u dogovoreno vrijeme obavlja šetnju uz komunikaciju s vlasnikom. Nakon završetka šetnje, aplikacija
označava šetnju kao završenu, a vlasnik može ocijeniti šetača, čime se cijeli postupak uspješno zaključuje.

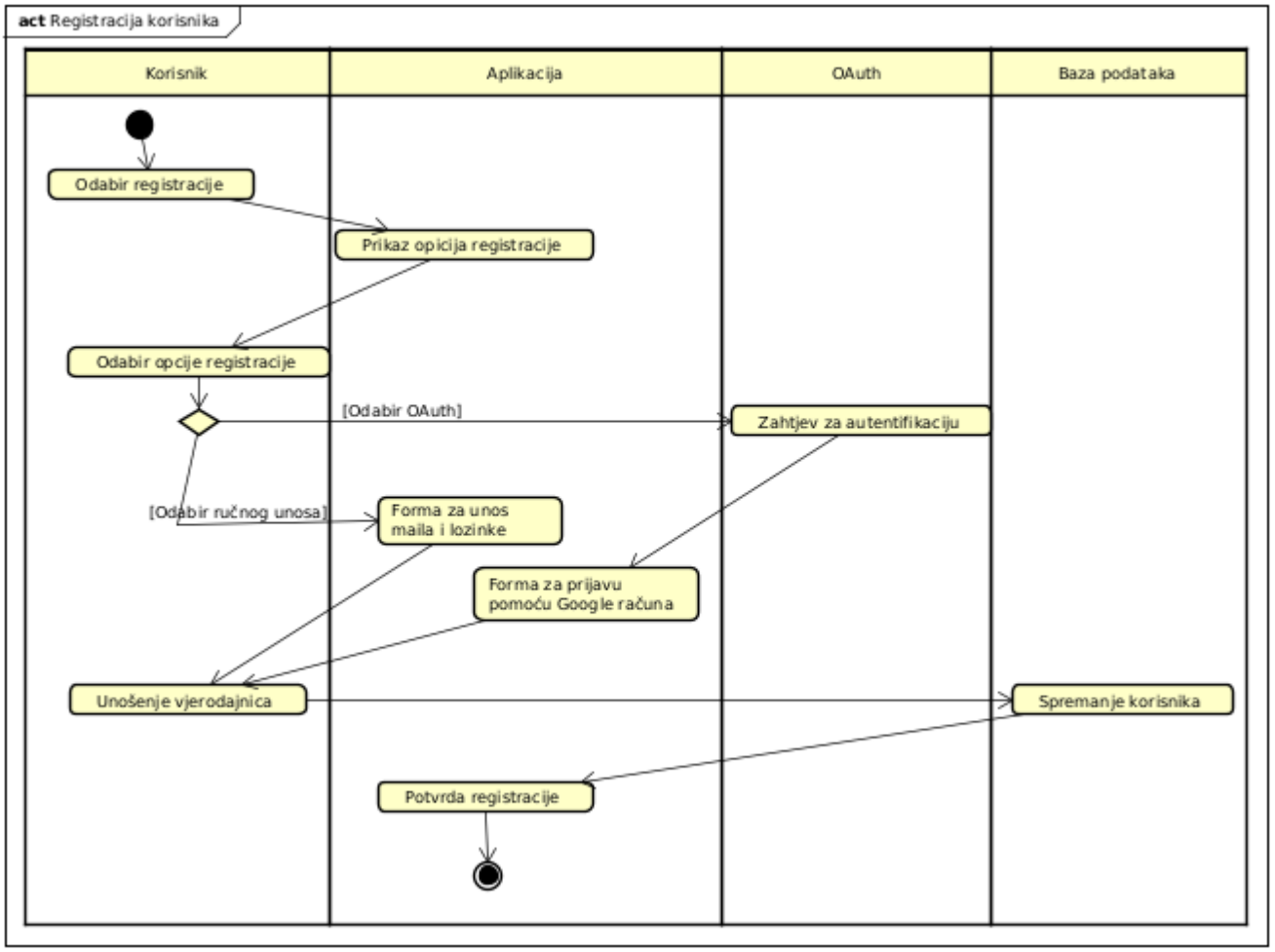


Registracija korisnika

Proces registracije započinje kada korisnik u aplikaciji odabere opciju registracije. Aplikacija prikazuje dostupne načine registracije, nakon čega korisnik bira između registracije mailom i lozinkom ili registracije putem vanjskog OAuth sustava.

Ako korisnik odabere OAuth registraciju, aplikacija šalje zahtjev vanjskom OAuth servisu za autentifikaciju. Nakon uspješne autentifikacije, korisnički podaci se prosljeđuju aplikaciji i spremaju u bazu podataka, čime je registracija dovršena.

U slučaju da korisnik odabere ručni unos, aplikacija prikazuje obrazac za unos e-mail adrese i lozinke. Korisnik unosi svoje vjerodajnice, a aplikacija ih obrađuje i sprema u bazu podataka. Nakon uspješnog spremanja podataka, registracija se potvrđuje.

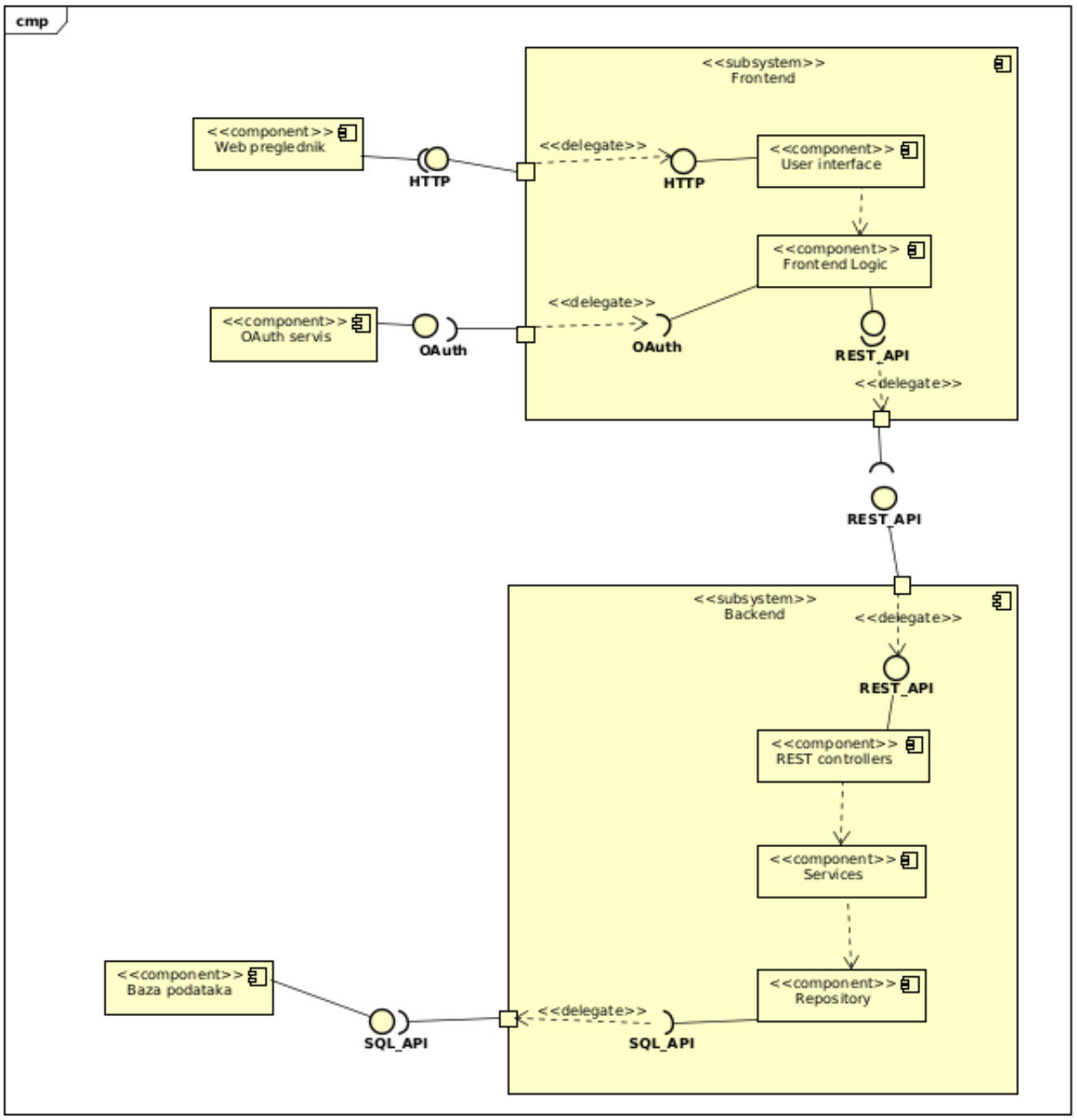

act Registracija korisnika
Korisnik
Aplikacija
OAuth
Baza podataka
Odabir registracije
Prikaz opicija registracije
Odabir opcije registracije
[Odabir OAuth]
Zahtjev za autentifikaciju
[Odabir ručnog unosa]
Forma za unos maila i lozinke
Forma za prijavu pomoću Google računa
Unošenje vjerodajnica
Spremanje korisnika
Potvrda registracije

Dijagram komponenata

Ovaj dijagram opisuje korisnikov pristup aplikaciji putem web preglednika, koji putem HTTP protokola komunicira s frontendom. Unutar frontenda nalaze se komponenta korisničkog sučelja i frontend logika, pri čemu korisničko sučelje služi za interakciju s korisnikom, a frontend logika za obradu zahtjeva i pripremu podataka.

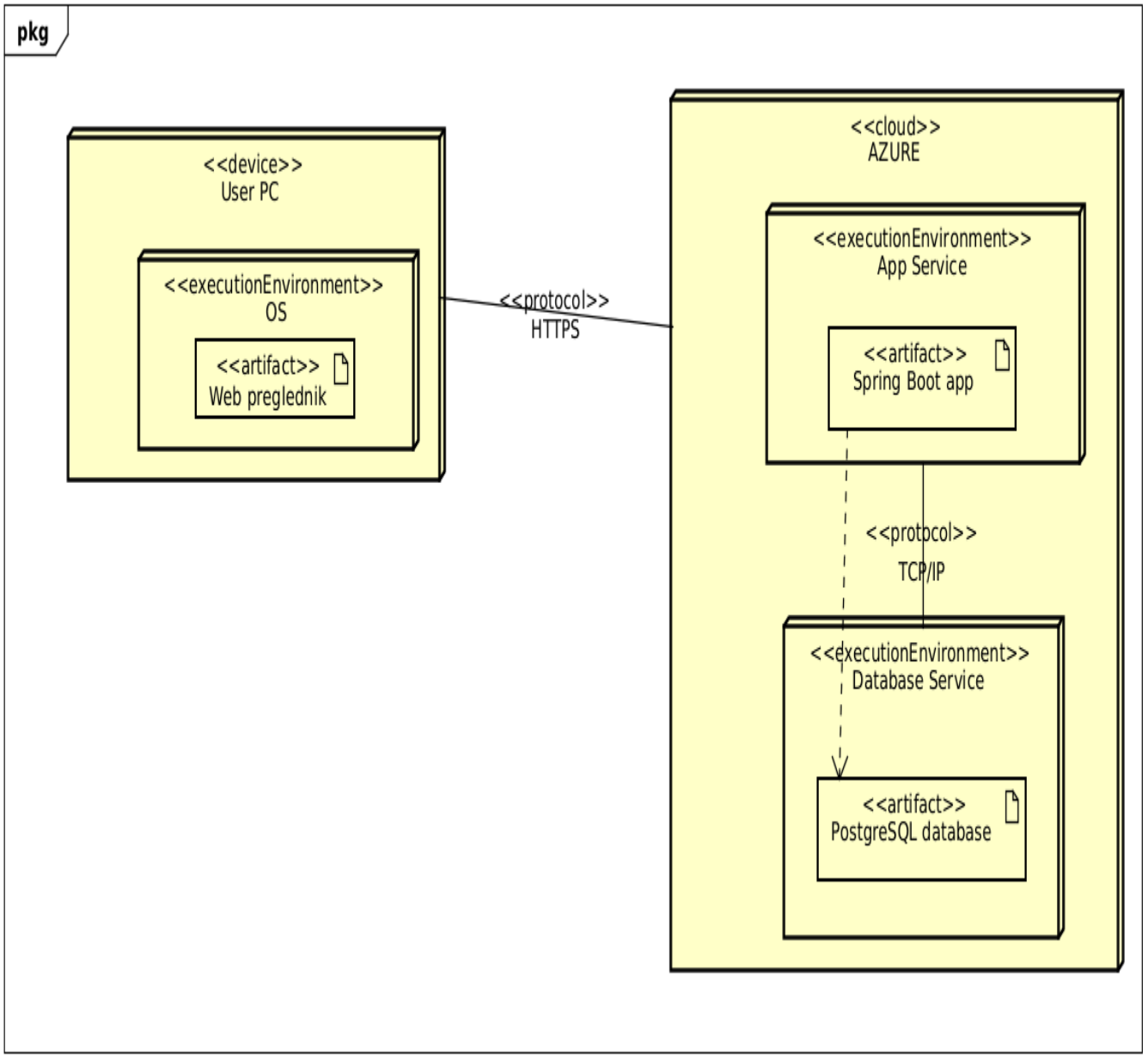
Frontend podsustav koristi vanjski OAuth servis za autentikaciju i autorizaciju korisnika, čime se osigurava kontrolirani pristup aplikaciji. Nakon uspješne autentikacije, frontend putem REST API-ja šalje zahtjeve prema backend podsustavu.

Backend je organiziran u više slojeva kako bi se jasno razdvojile odgovornosti. REST kontroleri zaprimaju zahtjeve s frontenda i prosljeđuju ih servisnom sloju, koji sadrži poslovnu logiku aplikacije. Servisi zatim koriste repozitorije za pristup podacima, a repozitoriji komuniciraju s bazom podataka putem SQL API-ja.



Dijagram razmještaja

Ovaj dijagram prikazuje arhitekturu web aplikacije kroz UML dijagram razmještaja. Korisnik pristupa sustavu putem web preglednika na svom računalu, koji se izvršava unutar operacijskog sustava. Komunikacija s aplikacijom odvija se putem HTTPS protokola prema Azure cloud okruženju. U Azureu je aplikacija smještena u App Service izvršnom okruženju kao Spring Boot aplikacija. Aplikacija dalje komunicira s PostgreSQL bazom podataka putem TCP/IP protokola, pri čemu je baza smještena u zasebnom database servisu.


pkg
<<device>>
User PC
<<executionEnvironment>>
OS
<<artifact>>
Web preglednik
<<protocol>>
HTTPS
<<cloud>>
AZURE
<<executionEnvironment>>
App Service
<<artifact>>
Spring Boot app
<<protocol>>
TCP/IP
<<executionEnvironment>>
Database Service
<<artifact>>
PostgreSQL database

Ispitivanje komponenti

1. ispitni slučaj (Redovan slučaj)

1. **Funkcionalnost:**
   Provjerava se funkcionalnost dodavanja psa postojećem vlasniku.
2. **Ispitni slučaj:**
   - **ulazni podaci:** username: "testUser", Dog objekt: {id: 1L, name: "Rex", breed: "Labrador", age: "3", energyLevel: "High", allowedTreats: "Yes", healthcare: "Healthy", personality: "Friendly"}
   - **očekivani rezultat:** Vlasnik "testUser" ima jednog dodanog psa u svojem popisu.
   - **dobiveni rezultat:** Test uspješno prolazi. Pas je dodan u vlasnikov popis pasa.
3. **Postupak provođenja ispitivanja:**
   Kreira se vlasnik "testUser" i pohranjuje se u simulirani repozitorij. Zatim se kreira objekt psa `Dog` i poziva metoda `ownerService.addDog("testUser", dog)`. Provjerava se veličina liste pasa vlasnika (`assertEquals(1, owner.getDogs().size())`) te se potvrđuje da je repozitorij pozvan za spremanje vlasnika (`verify(ownerRepository).save(owner)`).
4. **Izvorni kod:**

```
@Test
@DisplayName("Dodavanje psa vlasniku")
void addingDogToOwner() {
    when(ownerRepository.findByUsername("testUser")).thenReturn(Optional.of(owner));

    Dog dog = new Dog(
            id: 1L,
            name: "Rex",
            breed: "Labrador",
            age: "3",
            energyLevel: "High",
            allowedTreats: "Yes",
            healthcare: "Healthy",
            personality: "Friendly"
    );

    ownerService.addDog( username: "testUser", dog);

    assertEquals( expected: 1, owner.getDogs().size());
    verify(ownerRepository).save(owner);
}
```

2. ispitni slučaj (Slučaj izazivanja pogreške)

1. **Funkcionalnost:**
   Testira se ponašanje metode `addDog` kada se pokušava dodati pas vlasniku koji ne postoji.
2. **Ispitni slučaj:**
   - **ulazni podaci:** username: "nonexistent", Dog objekt: {id: 1L, name: "Rex", ...}
   - **očekivani rezultat:** Metoda baca `NoSuchElementException`, jer vlasnik s navedenim korisničkim imenom ne postoji.
   - **dobiveni rezultat:** Test prolazi; greška je ispravno bačena.
3. **Postupak provođenja ispitivanja:**
   Kreira se objekt psa `Dog`. Pokušava se pozvati metoda `ownerService.addDog("nonexistent", dog)` i provjerava da je bačena greška `NoSuchElementException`.
4. **Izvorni kod:**

```
@Test
@DisplayName("Dodavanje psa nepostojećem vlasniku")
void addingDogToNonexistingOwner() {

    Dog dog = new Dog(
            id: 1L,
            name: "Rex",
            breed: "Labrador",
            age: "3",
            energyLevel: "High",
            allowedTreats: "Yes",
            healthcare: "Healthy",
            personality: "Friendly"
    );

    assertThrows(NoSuchElementException.class, () -> {
        ownerService.addDog( username: "nonexistent", dog);
    });
}
```

3. ispitni slučaj (Rubni slučaj)

1. **Funkcionalnost:**
   Provjerava se funkcionalnost dodavanja psa s minimalnim unosom podataka.
2. **Ispitni slučaj:**
   - **ulazni podaci:** username: "testUser", Dog objekt: {id: 0L, name: "A", breed: "", age: "", energyLevel: "", allowedTreats: "", healthcare: "", personality: ""}
   - **očekivani rezultat:** Metoda stvara psa s minimalnom količinom podataka.
   - **dobiveni rezultat:** Test prolazi.

3. **Postupak provođenja ispitivanja:**
   Poziva se metoda `ownerService.addDog("testUser", dog)` i ispituje ima li pas, navedenog vlasnika "testUser", ime "A".
4. **Izvorni kod:**

```
@Test
@DisplayName("Dodavanje psa s minimalnim podacima")
void dogWithMinimalData() {

    Dog dog = new Dog(
            id: 0L,
            name: "A",
            breed: "",
            age: "",
            energyLevel: "",
            allowedTreats: "",
            healthcare: "",
            personality: ""
    );

    assertDoesNotThrow(() -> ownerService.addDog( username: "testUser", dog));
    assertEquals( expected: "A", owner.getDogs().get(0).getName());
}
```

4. ispitni slučaj (Redovan slučaj)

1. **Funkcionalnost:**
   Testira se funkcionalnost prijave registriranog korisnika u sustav.
2. **Ispitni slučaj:**
   - **ulazni podaci:** username: "testUser", password: "Password123"
   - **očekivani rezultat:** Sustav uspješno učitava korisnike, te vraća ispravne korisničke podatke.
   - **dobiveni rezultat:** Test prolazi. Korisnik je aktivan, a korisničko ime i lozinka odgovaraju očekivanim vrijednostima.
3. **Postupak provođenja ispitivanja:**
   Kreira se objekt korisnika s definiranim korisničkim imenom, lozinkom i statusom aktivacije. Simulira se dohvaćanje korisnika iz repozitorija. Poziva se metoda `userDetailService.loadUserByUsername("testUser")`, nakon čega se provjerava ispravnost korisničkih podataka.
4. **Izvorni kod:**

```java
@Test
@DisplayName("Prijava korisnika u sustav")
void userLogin() {
    UserEntity testUser = new UserEntity( username: "testUser");
    testUser.setPassword("Password123");
    testUser.setEnabled(true);

    when(userRepository.findByUsername("testUser"))
            .thenReturn(Optional.of(testUser));

    UserDetails result = userDetailsService.loadUserByUsername("testUser");

    assertTrue(result.isEnabled());
    assertEquals( expected: "testUser", result.getUsername());
    assertEquals( expected: "Password123", result.getPassword());

    verify(userRepository, times( wantedNumberOfInvocations: 2)).findByUsername("testUser");
}
```

5. ispitni slučaj (Redovan slučaj)

1. **Funkcionalnost:**
   Testira se funkcionalnost brisanja korisnika iz sustava.
2. **Ispitni slučaj:**
   - **ulazni podaci:** username: "userToDelete"
   - **očekivani rezultat:** Korisnik se uspješno briše iz sustava.
   - **dobiveni rezultat:** Test prolazi. Metoda za brisanje je uspješno pozvana.
3. **Postupak provođenja ispitivanja:**
   Poziva se metoda `registrationService.delete(username)`. Provjerava se da je metoda `deleteByUsername` u repozitoriju pozvana točno jedanput s odgovarajućim korisničkim imenom.
4. **Izvorni kod:**

```java
@Test
@DisplayName("Obriši korisnika po korisničkom imenu")
void deleteUserByUserName() {

    String username = "userToDelete";

    registrationService.delete(username);

    verify(userRepository, times( wantedNumberOfInvocations: 1)).deleteByUsername(username);
}
```

6. ispitni slučaj (Slučaj nepostojeće funkcionalnosti)

1. **Funkcionalnost:**
   Povjerava se ponašanje sustava prilikom prijave korisnika koji nije registriran.
2. **Ispitni slučaj:**
   - **ulazni podaci:** username: "unregisteredUser"
   - **očekivani rezultat:** Sustav baca `UsernameNotFoundException` jer korisnik s navedenim korisničkim imenom ne postoji.
   - **dobiveni rezultat:** Test prolazi. Greška je bačena.
3. **Postupak provođenja ispitivanja:**
   Pokušava se učitati korisnik pozivom metode `userDetailsService.loadUserByUsername(username)` s korisničkim imenom koje ne postoji u sustavu. Povjerava se da poruka greške odgovara očekivanoj vrijednosti `"User not found"`.
4. **Izvorni kod:**

```
@Test
@DisplayName("Login neregistriranog korisnika")
void loginOfUnregisteredUser() {

    String username = "unregisteredUser";

    UsernameNotFoundException exception = assertThrows(
            UsernameNotFoundException.class,
            () -> userDetailsService.loadUserByUsername(username)
    );

    assertEquals( expected: "User not found", exception.getMessage());
}
```

Ispitivanje sustava

1. Šetač stvara termin (Rubni slučaj)

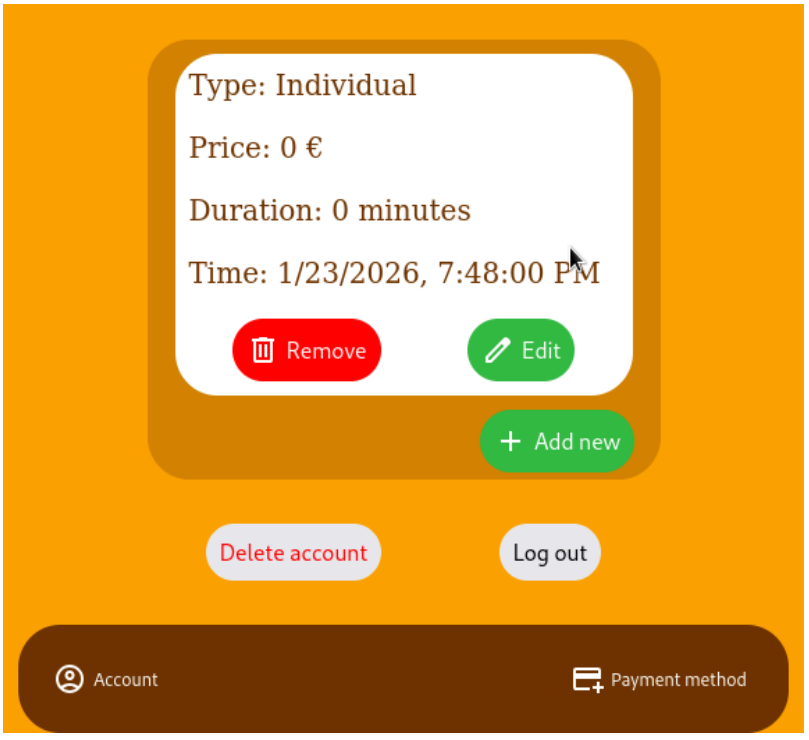
1. **Ulazi:**
   - **price:** 0
   - **duration:** 0
   - **datetime:** 2026-01-23T07:48
2. **Koraci ispitivanja:**
   - Prijava korisnika u sustav.
   - Odabir opcije za stvaranje novog termina.
   - Unos potrebnih podataka za termin.
   - Spremanje termina pritiskom na gumb za potvrdu.

3. **Očekivani izlaz:**
    - Sustav stvara termin unatoč rubnim vrijednostima unosa.
4. **Dobiveni izlaz:**

Running 'Dodavanje termina'

1. open on https://pawpal.click/ OK
2. setWindowSize on 1596x982 OK
3. click on id=add OK
4. click on id=price OK
5. type on id=price with value 0 OK
6. click on id=duration OK
7. type on id=duration with value 0 OK
8. click on id=datetime OK
9. click on id=datetime OK
10. click on id=datetime OK
11. type on id=datetime with value 2026-01-23T07:48 OK
12. type on id=datetime with value 2026-01-23T19:48 OK
13. click on id=submit OK

'Dodavanje termina' completed successfully



2. Vlasnik psa pravi profil novom psu (Redovan slučaj)

1. **Ulazi:**
    - **name:** "Cucak"
    - **breed:** "Rotveiler"
    - **age:** 17

- **energyLevel:** 9000
- **allowedTreats:** "Čips"
- **healthcare:** "NO"
- **personality:** "Agressive"

2. **Koraci ispitivanja:**
   - Prijava korisnika u sustav.
   - Odabir opcije za stvaranje profila novog psa.
   - Unos svih potrebnih podataka o psu.
   - Spremanje profila pritiskom na gumb za potvrdu.
3. **Očekivani izlaz:**
   - Sustav uspješno stvara i prikazuje profil novog psa.
4. **Dobiveni izlaz:**

Running 'Dodavanje novog psa'

1. open on https://pawpal.click/edit/owner/add OK
2. setWindowSize on 1596x982 OK
3. click on name=name OK
4. type on name=name with value Cucak OK
5. click on name=breed OK
6. type on name=breed with value Rotveiler OK
7. click on name=age OK
8. type on name=age with value 17 OK
9. click on name=energyLevel OK
10. type on name=energyLevel with value 9000 OK
11. click on name=allowedTreats OK
12. type on name=allowedTreats with value Čips OK
13. click on name=healthcare OK
14. type on name=healthcare with value NO OK
15. click on name=personality OK
16. type on name=personality with value Agressive OK
17. click on css=p:nth-child(1) > input OK

'Dodavanje novog psa' completed successfully

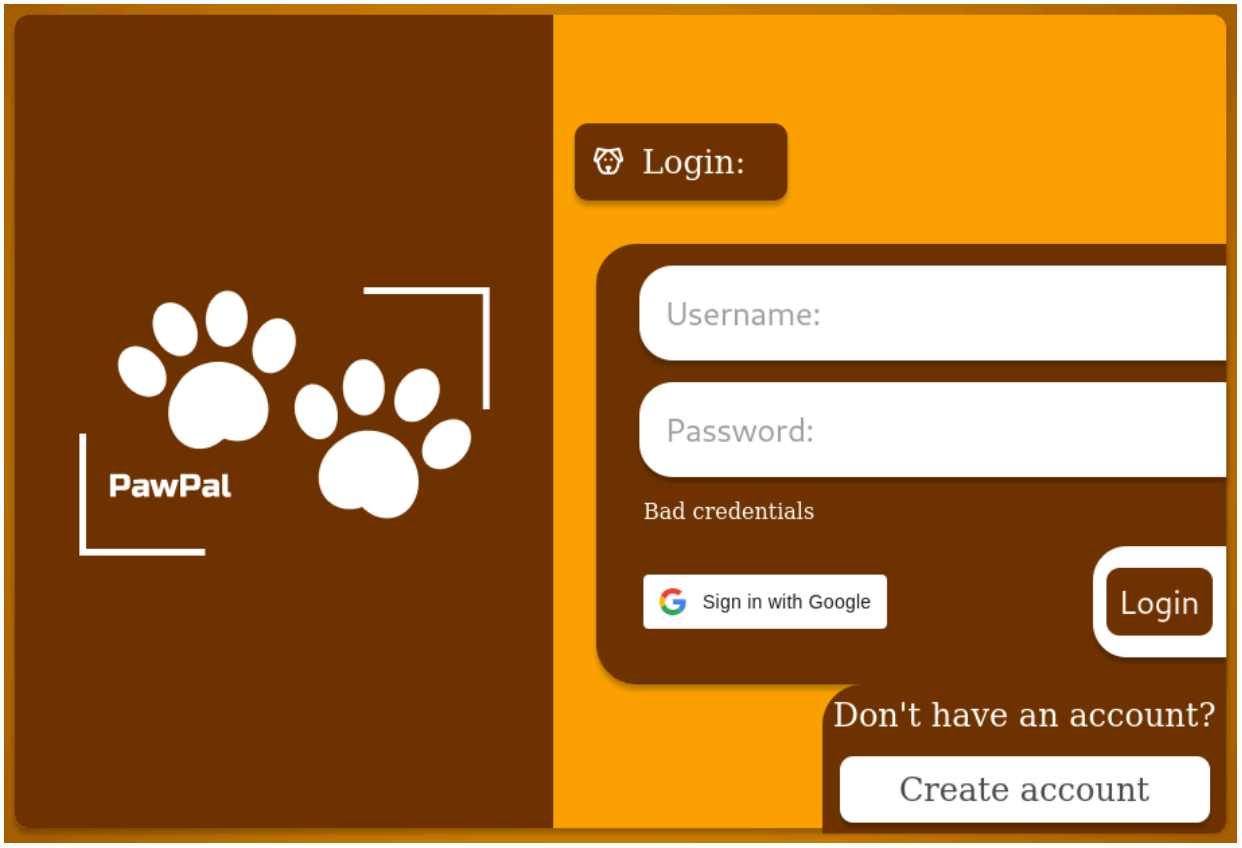
3. Prijava korisnika bez registracije (Slučaj poziva nepostojećih funkcionalnosti)

1. **Ulazi:**
    - **name:** "Jakov"
    - **password:** "pass"
2. **Koraci ispitivanja:**
    - Pokretanje aplikacije.
    - Unos korisničkog imena i lozinke nepostojećeg korisnika.
    - Pokušaj prijave u sustav.
    - Sustav javlja grešku `Bad credentials`.
3. **Očekivani izlaz:**
    - Sustav odbija prijavu i prikazuje grešku `Bad credentials`.
4. **Dobiveni izlaz:**

Running 'Prijava u sustav neregistriranog korisnika'

1. open on /login OK
2. setWindowSize on 1596x982 OK
3. click on name=username OK
4. type on name=username with value Jakov OK
5. sendKeys on name=username with value ${KEY_DOWN} OK
6. sendKeys on name=username with value ${KEY_ENTER} OK
7. type on name=password with value pass OK
8. click on css=.login_button input OK

'Prijava u sustav neregistriranog korisnika' completed successfully



4. Brisanje profila psa (Redovan slučaj)

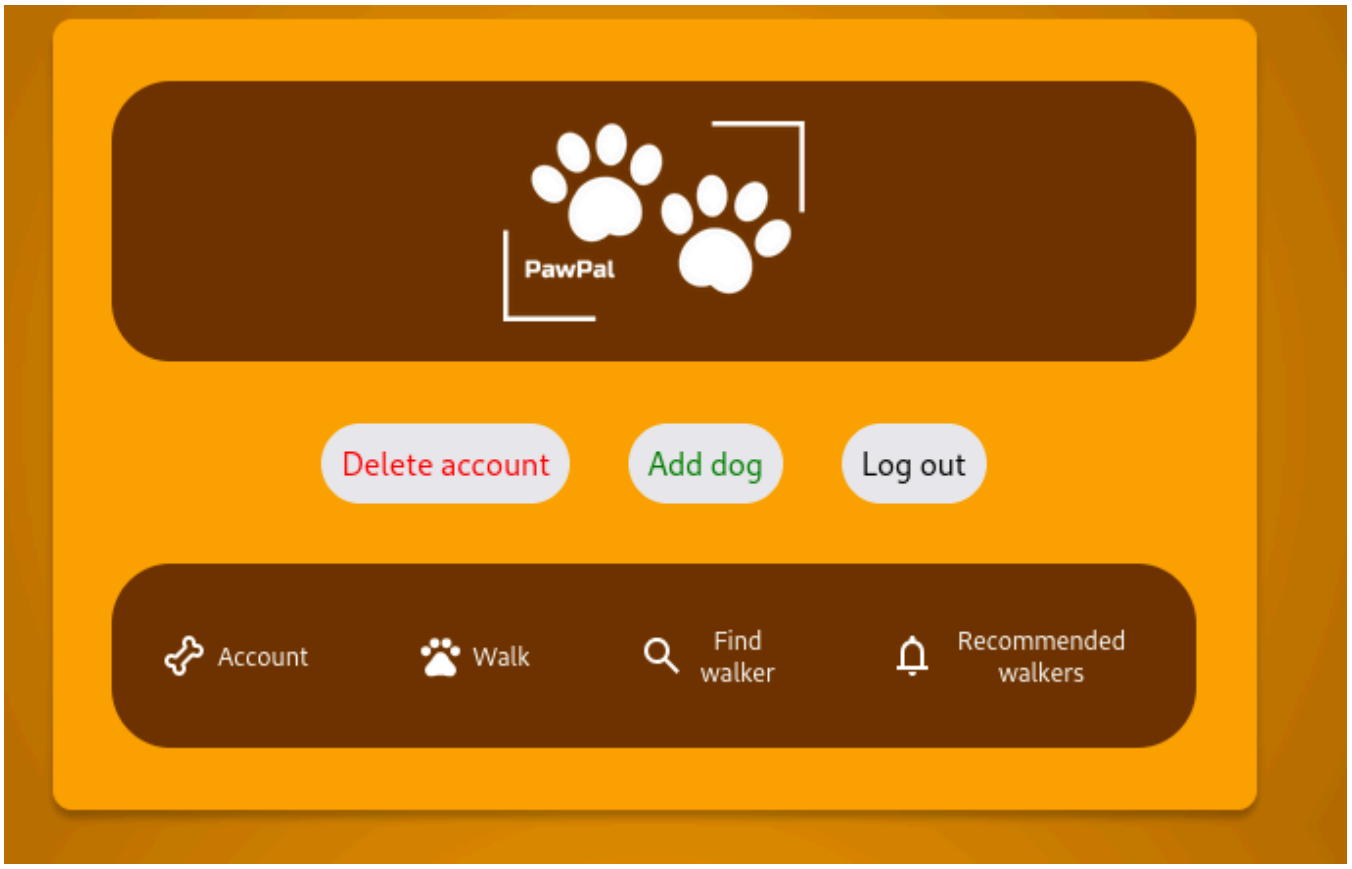
1. **Ulazi:**
   - Nema ulaznih podataka.
2. **Koraci ispitivanja:**
   - Prijava korisnika u aplikaciju.
   - Odabir opcije za brisanje profila psa.
   - Sustav prikazuje upozorenje o trajnom brisanju profila.
   - Vlasnik potvrđuje brisanje.
   - Profil psa se uklanja iz sustava.
3. **Očekivani izlaz:**
   - Profil psa je trajno uklonjen iz sustava.

4. **Dobiveni izlaz:**

Running 'Uklanjanje psa'

1. open on https://pawpal.click/ OK
2. setWindowSize on 1596x982 OK
3. chooseOkOnNextConfirmation OK
4. click on css=.dogData:nth-child(1) .delete > span OK
5. assertConfirmation on This dog will be permanently removed and you won't be able to undo it! OK
6. webdriverChooseOkOnVisibleConfirmation OK
7. assertAlert on Selected dog has been successfully removed. OK

'Uklanjanje psa' completed successfully



Korištene tehnologije i alati

1. Programski jezici

HTML i CSS

HTML i CSS koriste se za izradu strukture i vizualnog izgleda web aplikacije. HTML definira sadržaj stranice, dok se CSS koristi za stilizaciju i prilagodbu izgleda korisničkog sučelja.

Java 17

Java je objektno orijentirani programski jezik koji se koristi za razvoj backend dijela aplikacije. Odabrana je zbog svoje stabilnosti, sigurnosti i široke primjene u izradi serverskih aplikacija.

JavaScript ES2023

JavaScript je programski jezik koji se koristi na klijentskoj strani aplikacije. Omogućuje interaktivnost i dinamičko ponašanje web stranica te poboljšava korisničko iskustvo.

2. Radni okviri i biblioteke

Spring Boot 3.5.6

Spring Boot je Java radni okvir koji se koristi za razvoj backend dijela aplikacije. Pojednostavljuje konfiguraciju aplikacije, omogućuje brzi razvoj te dolazi s ugrađenim web poslužiteljem. Često se koristi za izradu REST API-ja i poslovne logike.

Thymeleaf 3.1.3

Thymeleaf je server-side template engine koji se koristi za generiranje HTML stranica. Omogućuje jednostavno povezivanje backend podataka s korisničkim sučeljem te izradu preglednih i lako održivih predložaka.

3. Baza podataka

PostgreSQL 42.7.7

PostgreSQL je relacijska baza podataka poznata po svojoj pouzdanosti i stabilnosti. Podržava napredne SQL značajke i ACID principe, a dobro se integrira sa Spring Boot aplikacijama.

4. Razvojni alati

IntelliJ IDEA 21.0.7

IntelliJ IDEA je razvojno okruženje koje omogućuje pisanje, uređivanje i testiranje Java koda. Nudi napredne alate za refaktoriranje i snažnu podršku za Spring Boot projekte.

GitHub

GitHub se koristi za verzioniranje izvornog koda i timsku suradnju. Omogućuje praćenje promjena, rad s granama te jednostavno upravljanje projektom kroz pull requestove.

5. Alati za ispitivanje

JUnit 5.12.2

JUnit je framework za jedinično testiranje Java aplikacija. Omogućuje pisanje testova koji provjeravaju ispravnost poslovne logike i pomažu u ranom otkrivanju grešaka.

Selenium IDE

Selenium IDE je alat za automatizirano testiranje korisničkog sučelja. Omogućuje snimanje korisničkih interakcija u pregledniku i njihovo ponovno izvođenje kao testnih scenarija.

6. Razmještaj i cloud platforma

Microsoft Azure

Microsoft Azure je cloud platforma koja se koristi za razmještaj aplikacije. Omogućuje hosting, skaliranje i nadzor aplikacije te integraciju s bazom podataka i sigurnosnim postavkama.

8.1 Instalacija

Lokalno okruženje (razvoj/test)

- Instaliran JDK 17
- Git
- IDE po izboru

Baza podataka

- Za produkcijsko okruženje preporučeno je koristiti PostgreSQL

Vanjske integracije

- Za funkcionalnosti vezane uz Google Calendar i prijavu preko Google računa potrebno je imati Google OAuth2 vjerodajnice (client id/secret) i odgovarajuće postavke u aplikaciji

---

8.2 Postavke

1. Klonirati repozitorij
    - `https://github.com/JanCerovac/PawPal`
2. Projekt se nalazi u direktoriju:
    - `projekt/`
3. Najvažnije lokacije u projektu:
    - Backend (Spring Boot): `projekt/src/main/java/root/`
    - Konfiguracija: `projekt/src/main/resources/application.properties`
    - HTML predlošci (Thymeleaf): `projekt/src/main/resources/templates/`
    - Statički resursi (CSS/JS): `projekt/src/main/resources/static/`

8.3 Pokretanje aplikacije

U nastavku su upute za pokretanje aplikacije u dva okruženja

Razvojno okruženje

- Klonirati repozitorij i otvoriti direktorij `projekt/` kao backend projekt
- U konfiguraciji (`application.properties`) provjeriti da su postavke baze i integracija u razvojnom režimu ispravne za lokalno pokretanje
- Pokrenuti aplikaciju na jedan od uobičajenih načina
    - Pokretanjem glavne Spring Boot klase iz IDE-a
    - Ili pokretanjem kroz build alat definiran u projektu
- Provjeriti da je aplikacija dostupna u pregledniku na lokalnoj adresi

Produkcijsko okruženje

- Pripremiti produkcijske postavke
    - Postaviti produkcijsku bazu (preporučeno PostgreSQL)
    - Postaviti produkcijske konfiguracije (najčešće kroz environment varijable ili produkcijski profil)
    - Postaviti vjerodajnice za vanjske servise ako se koriste (npr. Google OAuth2, Google Calendar, PayPal)
- Izgraditi aplikaciju u izvršni artefakt (JAR) koristeći build alat koji je definiran u projektu
- Pokrenuti aplikaciju kao servis ili proces na produkcijskom poslužitelju ili platformi za deploy
- Provjeriti da je aplikacija dostupna putem javnog URL-a

8.4 Upute za administratore

Pristup administratorskom sučelju

- Administratorskom sučelju se pristupa putem rute `/admin`
- Prije pristupa administratorskom sučelju potrebno je prijaviti se u aplikaciju kao admin korisnik
- Zadani podaci za prijavu administratora su
    - username: `admin`
    - password: `1234`

Administratorske funkcionalnosti

- Brisanje korisnika iz sustava
- Postavljanje cijene članarine (mjesečna i godišnja)

Redovito održavanje

- Arhiviranje baze podataka
    - Periodično izraditi sigurnosnu kopiju baze prema pravilima korištene baze i okruženja
- Pregled logova
    - Redovito pregledavati logove aplikacije na platformi ili poslužitelju kako bi se uočile greške
- Ažuriranje aplikacije
    - Povlačenje novih verzija iz Git repozitorija i ponovno pokretanje aplikacije nakon deploya nove verzije
    - Provjera osnovnih funkcionalnosti nakon ažuriranja (prijava, setup tok, admin sučelje)

8.5 Deployment

Aplikacija je Spring Boot web aplikacija i može se deployati na bilo koju platformu koja podržava Java runtime i mrežni pristup.
Preporučeni koraci:

1. Pripremiti produkcijsku bazu
2. Postaviti produkcijske konfiguracije
    - parametri baze podataka
    - OAuth2/Google vjerodajnice
3. Izgraditi aplikaciju (Gradle build) i pokrenuti JAR
4. Provjeriti dostupnost aplikacije kroz javni URL

Opis pristupa aplikaciji na javnom poslužitelju

Aplikaciji se pristupa putem javne poveznice

- https://pawpal.click

Korištenje aplikacije

- Aplikaciji se pristupa kroz internetski preglednik
- Za korištenje funkcionalnosti potrebno je prijaviti se ili registrirati
- Pri prvom pristupu nakon registracije korisnik odabire ulogu (OWNER ili WALKER) kroz setup tok

Pristup administratorskom sučelju

- Prijaviti se u aplikaciju s admin korisnikom (`admin` / `1234`)

Ograničenja sustava

- Aplikacija ovisi o dostupnosti i ispravnoj konfiguraciji vanjskih servisa koji se koriste u funkcionalnostima sustava, posebno prijava putem Google računa, integracija kalendara i plaćanja
- Stabilnost sustava ovisi i o dostupnosti baze podataka i mrežne povezanosti između aplikacije i baze, u slučaju prekida moguća je privremena nedostupnost funkcionalnosti koje ovise o bazi

9. Zaključak i budući rad

Sažetak

Tijekom izrade projekta **PawPal** cilj nam je bio razviti web aplikaciju koja olakšava povezivanje vlasnika pasa i šetača pasa kroz jednostavnu registraciju, odabir uloge te postavljanje osnovnih podataka o korisniku i psima. Projekt je razvijan timski u sklopu kolegija Programsko inženjerstvo, pri čemu smo implementirali brojne funkcionalnosti aplikacije i napravili dokumentaciju koja prati izradu projekta.

Trenutna verzija aplikacije predstavlja temelj za mogući daljnji razvoj, a dokumentacija pokriva zahtjeve, arhitekturu, dizajn i bazu podataka sukladno smjernicama kolegija.

Tehnički izazovi

Najveći izazovi tijekom razvoja odnosili su se na:

- Učenje i usklađivanje tehnologija i alata potrebnih za izradu cjelovitog web sustava (frontend, backend, baza, dokumentacija)
- Integraciju komponenti (povezivanje korisničkih formi, kontrolera, servisnog sloja i baze podataka), uz održavanje povezanosti između modela i implementacije
- Razvoj aplikacije pod vremenskim ograničenjem
- Timsko usklađivanje i raspodjela posla, pri čemu je bilo važno održavati jasnu komunikaciju i dogovor oko odluka dizajna i funkcionalnosti

Unatoč izazovima, većina prepreka rješavana je postupno kroz velik broj razvojnih verzija aplikacije, kontinuirano testiranje i pregled implementacije te usklađivanje s dokumentacijom i zadanim zahtjevima.

Stečena znanja i iskustva

Razvoj projekta omogućio nam je stjecanje praktičnih znanja i iskustava u sljedećim područjima:

- Razvoj web aplikacije
- Modeliranje baze podataka (entiteti, veze, ER i relacijski model) te povezivanje modela s implementacijom
- Timski rad - koordinacija, podjela zadataka, praćenje napretka i donošenje kompromisa
- Izrada tehničke dokumentacije u GitHub Wiki formatu

Lakše izvođenje projekta

Kroz rad na projektu postalo je jasno da kvalitetno izvođenje sličnih zadataka značajno olakšavaju:

- prijašnje iskustvo s korištenim tehnologijama i alatima
- jasno definirani zadaci i raspodjela odgovornosti
- redovno i precizno vođenje dokumentacije usporedno s implementacijom

- redovita provjera točnosti i konzistentnosti - npr. pregledi koda, dogovori o strukturi, provjera konzistentnosti modela i koda

---

Perspektive za nastavak rada

Ako bi se razvoj aplikacije nastavio izvan okvira kolegija, sljedeći koraci bi uključivali:

- proširenje funkcionalnosti prema definiranim zahtjevima
- poboljšanje korisničkog iskustva i napredak vizualne reprezentacije aplikacije
- dodatno učvršćivanje sigurnosti
- po potrebi razdvajanje i modularizacija komponenata kako bi sustav bio lakše održiv i poboljšan u budućnosti

---

Opis neimplementiranih funkcionalnosti

U ovoj tablici će se navesti funkcionalnosti koje nisu implementirane u trenutnoj fazi projekta, uz njihove identifikatore i kratki opis.

ID zahtjeva	Opis

A. Dnevnik promjena dokumentacije

Ovaj dnevnik promjena evidentira ključne promjene u repozitoriju projekta PawPal. Zapisi su izrađeni na temelju povijesti commitova na grani `main`, pri čemu je svaki commit evidentiran kao zasebna revizija dokumentacije.

Rev.	Opis promjene/dodatka	Autor	Datum
0.1	Initial commit	Dominik Grus	12.10.2025.
0.2	Create LICENSE	Dominik Grus	12.10.2025.
0.3	Create CODE_OF_CONDUCT.md	Dominik Grus	12.10.2025.
0.4	Update README.md	Dominik Grus	12.10.2025.
0.5	Dodana početna struktura projekta.	Jakov Andrić	20.10.2025.
0.6	Dodan user journey mapa napravljena u FigJam-u	Damjan Mamula	21.10.2025.
0.7	Dodani funkcijski zahtjevi sustava	Dominik Grus	21.10.2025.
0.8	Add files via upload	Marija Ilić	22.10.2025.
0.9	Dodan backend za login.	Jakov Andrić	30.10.2025.
1.0	Dodan static HTML i CSS za login page	Damjan Mamula	30.10.2025.
1.1	Merge branch 'dev'	Jakov Andrić	02.11.2025.
1.2	Izmjene na login.html i login.css te dodana nova stranica za registraciju novih korisnika	Damjan Mamula	05.11.2025.
1.3	Implementiran OAuth2 za prijavu preko Google-a.	Jakov Andrić	06.11.2025.
1.4	Završena prva verzija login i registration funkcija.	Jakov Andrić	06.11.2025.
1.5	Dodajem prvu verziju backend-a za postavljanje profila (OWNER ili WALKER)	Jakov Andrić	07.11.2025.
1.6	Dodan HTML, CSS i JS za profile setup page	Damjan Mamula	11.11.2025.
1.7	Merge branch 'main'	Damjan Mamula	11.11.2025.

Rev.	Opis promjene/dodatka	Autor	Datum
1.8	Završen endpoint /setup za određivanje korisničke uloge.	Jakov Andrić	11.11.2025.
1.9	Dodan homepage.html	Marija Ilić	13.11.2025.
2.0	Dodan homepage.css	Marija Ilić	13.11.2025.
2.1	Dodane ikone za toolbar	Marija Ilić	13.11.2025.
2.2	Implementiran backend za homepage.html.	Jakov Andrić	13.11.2025.
2.3	Izmjena na CSS-u za Setup page, loše poravnanje za neke web preglednike	Damjan Mamula	13.11.2025.
2.4	Uklonjen test primjer iz homepage.html i popravljena veličina fonta u homepage.css	Marija Ilić	14.11.2025.
2.5	Dodan link za stranicu.	Jakov Andrić	14.11.2025.
2.6	Dodane nove funkcionalnosti u homepage.html, prilagođen homepage.css i dodane nove ikone.	Marija Ilić	13.01.2026.
2.7	Dodan payment.html i prilagođen homepage.css	Marija Ilić	15.01.2026.
2.8	Dodan backend za Google Calendar.	Jakov Andrić	15.01.2026.
2.9	Uređen kod.	Jakov Andrić	15.01.2026.
3.0	Izmjene na homepage.html (dodan walk page za ownera i popup window) te u homepage.css	Damjan Mamula	16.01.2026.
3.1	Dodani popUps za potvrdu rezervacije	Damjan Mamula	21.01.2026.
3.2	Dodan html, css i js za Admin korisnika	Damjan Mamula	21.01.2026.

B. Popis literature

U izradi projekta korišteni su sljedeći izvori (službena dokumentacija tehnologija, priručnici i ostali materijali):

1. Dokumentacija za Spring Boot
   https://docs.spring.io/spring-boot/docs/current/reference/html/
2. Dokumentacija za Spring Framework
   https://docs.spring.io/spring-framework/reference/
3. Dokumentacija za Spring Security
   https://docs.spring.io/spring-security/reference/
4. Dokumentacija za Spring Data JPA
   https://docs.spring.io/spring-data/jpa/docs/current/reference/html/
5. Dokumentacija za PostgreSQL
   https://www.postgresql.org/docs/
6. Dokumentacija za Thymeleaf
   https://www.thymeleaf.org/documentation.html
7. Dokumentacija za Google Identity
   https://developers.google.com/identity
8. Dokumentacija za Google Calendar API
   https://developers.google.com/calendar/api
9. Dokumentacija za PayPal Developers
   https://developer.paypal.com/docs/
10. ERDPlus
    https://erdplus.com/
11. ChatGPT
    https://chatgpt.com/

Dnevnik sastajanja

1. sastanak

- **Datum:** 08.10.2025.
- **Prisustvovali:** D. Mamula, J. Andrić, J. Cerovac, V. Magdić, D. Grus, M. Ilić, G. Mihaljević
- **Teme:**
    - Uvod u projekt i ciljeve prve predaje
    - Dogovor oko načina rada (repozitorij, grane, pravila commitanja)
    - Dogovor oko strukture dokumentacije i podjele zaduženja

2. sastanak

- **Datum:** 19.10.2025.
- **Prisustvovali:** D. Mamula, J. Andrić, J. Cerovac, V. Magdić, D. Grus, M. Ilić, G. Mihaljević
- **Teme:**
    - Pregled funkcionalnih zahtjeva
    - Dogovor oko osnovnog izgleda aplikacije
    - Usklađivanje backend i frontend plana rada

3. sastanak

- **Datum:** 03.11.2025.
- **Prisustvovali:** D. Mamula, J. Andrić, J. Cerovac, V. Magdić, D. Grus, M. Ilić, G. Mihaljević
- **Teme:**
    - Uvođenje mogućnosti prijave
    - Dogovor oko toka postavljanja profila i korisničke uloge
    - Usklađivanje modela podataka s dogovorenim tokovima

4. sastanak

- **Datum:** 24.11.2025.
- **Prisustvovali:** D. Mamula, J. Andrić, J. Cerovac, V. Magdić, D. Grus, M. Ilić, G. Mihaljević
- **Teme:**
    - Plan sljedeće faze razvoja
    - Pregled stanja dokumentacije i što je preostalo za dopunu

5. sastanak

- **Datum:** 12.01.2026.
- **Prisustvovali:** D. Mamula, J. Andrić, J. Cerovac, V. Magdić, D. Grus, M. Ilić, G. Mihaljević
- **Teme:**

- Usklađivanje integracija vanjskih servisa s ostatkom sustava
- Dogovor oko UI nadogradnji i završnih dorada
- Pregled preostalih priloga dokumentacije i plan dovršetka