Programsko inženjerstvo ak.god 2025./2026.

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

ClayPlay

Tim: <TG15.3>

Ime tima: DevNopes

Nastavnik: Vlado Sruk

1. Uvod

Keramika je sve popularniji oblik kreativnog izražavanja i opuštanja, kako među hobistima tako i među profesionalcima. Upravo zbog rastuće potražnje za kvalitetnim radionicama, postoji potreba za izradom moderne digitalne platforme koja omogućuje jednostavno povezivanje zainteresiranih polaznika i organizatora radionica keramike. Ovaj dokument detaljno opisuje korisničke zahtjeve, cilj projekta, razradu problema, analizu postojećih rješenja, moguće nadogradnje i druge važne aspekte ovog sustava.

---

2. Cilj projektnog zadatka

Cilj projekta je izraditi **web-platformu** koja će služiti kao centralno mjesto za:

- pregled, prijavu i plaćanje termina radionica keramike
- promociju održanih radionica
- prodaju gotovih keramičkih radova
- promociju i organizaciju izložbi
- međusobnu komunikaciju između polaznika i organizatora

Platforma omogućuje dvosmjernu interakciju između korisnika (polaznika) i kreatora sadržaja (organizatora), čime se potiče zajednica i olakšava pristup kulturno-umjetničkim sadržajima.

---

3. Razrada problematike

Trenutno ne postoji jedinstvena lokalizirana digitalna platforma koja omogućava organizatorima radionica keramike da u nekoliko klikova:

- objave detalje radionice
- prikupe prijave i izvrše naplatu
- promoviraju svoj rad i prodaju proizvode

S druge strane, polaznici se često moraju oslanjati na društvene mreže, oglase ili preporuke kako bi saznali za radionice, što otežava pristup i smanjuje vidljivost manjih lokalnih umjetnika.

Problemi koje projekt rješava:

- **Raspršenost informacija**: Radionice se oglašavaju na različitim kanalima (Facebook, Instagram, web stranice), što korisnicima otežava usporedbu i planiranje.
- **Nedostatak automatiziranog sustava za prijave i plaćanje**.
- **Nema integrirane trgovine za prodaju radova organizatora i polaznika.**
- **Ne postoji arhiva i promocija održanih radionica.**

---

4. Potencijalna korist projekta

Ovaj projekt donosi višestruke koristi za sve uključene strane:

Za polaznike:

- Jednostavan pregled i prijava na radionice
- Sigurno online plaćanje
- Mogućnost praćenja omiljenih organizatora i proizvoda
- Sudjelovanje u izložbama i zajednici

Za organizatore:

- Vidljivost i promocija kroz vlastiti javni profil
- Automatsko upravljanje prijavama i plaćanjima
- Dodatni prihod putem prodaje radova
- Alati za promociju i arhiviranje aktivnosti

Za tržište:

- Razvoj kreativne zajednice
- Promocija lokalne umjetnosti i rukotvorina
- Digitalizacija sektora edukacije u umjetnosti

---

5. Postojeća slična rješenja

Iako postoje opće platforme za rezervaciju termina, nijedna nije specijalizirana za radionice keramike. Neki primjeri sličnih rješenja uključuju:

5.1 Airbnb Experiences

https://www.airbnb.com/experiences

- **Fokus**: razne aktivnosti, uključujući umjetničke radionice.
- **Razlike**: Nema specijalizaciju za keramiku, ne nudi online trgovinu ni izložbe.

5.2 Eventbrite

https://www.eventbrite.com

- **Fokus**: organizacija evenata, seminara, radionica
- **Razlike**: Platforma je preopćenita, ne nudi trgovinu ni dugoročnu interakciju između korisnika.

5.3 Etsy

https://www.etsy.com

- **Fokus**: prodaja ručno rađenih proizvoda.
- **Razlike**: Nema funkcionalnost za radionice, prijave ni izložbe.

---

6. Skup korisnika

Primarne korisničke skupine:

- **Polaznici radionica**: hobisti, studenti, kreativci
- **Organizatori**: umjetnici, keramičari, studiji keramike

Sekundarne skupine:

- **Kupci proizvoda**: svi zainteresirani za jedinstvene ručne radove
- **Administratori platforme**: tehnička podrška i podrška zajednici

---

7. Mogućnost prilagodbe rješenja

Platforma je zamišljena kao modularna i prilagodljiva za različite scenarije:

- **Jezična lokalizacija**: podrška za više jezika (npr. HR, EN, DE)
- **Prilagodba za druge oblike radionica**: slikanje, izrada nakita, glazba
- **Prilagodba mobilnim uređajima**: responzivni dizajn ili mobilna aplikacija
- **Pristupačnost**: podrška za korisnike s invaliditetom (npr. čitači ekrana)

---

8. Opseg projektnog zadatka

Ključne funkcionalnosti:

- Registracija korisnika (OAuth 2.0)
- Upravljanje profilima organizatora
- Kalendar radionica (integracija s Google kalendarom)
- Prijave i plaćanja (PayPal i kreditne kartice)
- Trgovina keramičkih radova
- Organizacija i promocija izložbi
- Sustav recenzija i komentara
- Obavijesti i pretplate
- Administracija korisnika i članarina

---

9. Moguće nadogradnje

Projekt može rasti kroz brojne funkcionalne nadogradnje:

a) Mobilna aplikacija

- Brži pristup sadržaju
- Push obavijesti o novim radionicama i izložbama

b) Video radionice

- Mogućnost online edukacije i snimljenih tečajeva

c) Certifikati i priznanja

- Polaznici dobivaju digitalne certifikate po završetku

d) Community forum

- Povezivanje članova zajednice, razmjena iskustava i savjeta

e) Statistika i analitika

- Za organizatore: statistika prijava, prodaje i interakcije
- Za administratore: analiza aktivnosti i rasta zajednice

Zaključak

Razvoj ovakve platforme ne samo da modernizira i digitalizira sektor kreativnih radionica, već aktivno doprinosi rastu lokalne umjetničke scene. Kroz ciljane funkcionalnosti, prilagodljivost i jednostavno korisničko sučelje, projekt rješava stvarne probleme i otvara prostor za buduće inovacije i širenje na druge segmente kreativne edukacije.

---

1. Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Sustav omogućuje registraciju i prijavu korisnika kao polaznika ili organizatora.	Visok	Tekst zadatka	Korisnik odabire ulogu, ispunjava podatke i uspješno se registrira.
F-002	Sustav omogućuje prijavu korisnika putem vanjskih servisa za autentifikaciju.	Visok	Tekst zadatka	Korisnik se može prijaviti koristeći odabrani vanjski servis.
F-003	Sustav omogućuje uređivanje javnog profila organizatora.	Srednji	Tekst zadatka	Organizator može urediti osobne podatke, sliku, adresu i kontakt.
F-004	Sustav omogućuje organizatoru dodavanje novih radionica s pripadajućim detaljima.	Visok	Tekst zadatka	Organizator unosi sve potrebne informacije o radionici, koje su nakon spremanja vidljive korisnicima.
F-005	Sustav omogućuje polaznicima pregled i rezervaciju termina radionica.	Visok	Tekst zadatka	Polaznik vidi dostupne termine i može uspješno rezervirati mjesto.
F-006	Sustav omogućuje prikaz termina radionica putem integriranog kalendara.	Srednji	Tekst zadatka	Termini radionica su vidljivi i ažurirani u kalendaru.
F-007	Sustav omogućuje korisnicima plaćanje termina radionica putem integriranih servisa.	Visok	Tekst zadatka	Plaćanje se izvršava uspješno i potvrda se pohranjuje u sustav.
F-008	Sustav omogućuje organizatorima plaćanje članarine prema odabranom planu.	Visok	Tekst zadatka	Organizator odabire plan i uspješno izvršava plaćanje.
F-009	Sustav omogućuje korisnicima otkazivanje rezervacija najkasnije 48 sati prije početka radionice.	Srednji	Tekst zadatka	Sustav onemogućuje otkazivanje unutar 48 sati i prikazuje poruku potvrde pri valjanom otkazivanju.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-010	Sustav sadrži internetsku trgovinu keramičkih radova.	Visok	Tekst zadatka	Proizvodi su kategorizirani i dostupni za kupnju putem sučelja.
F-011	Sustav omogućuje kupcima pregled, filtriranje i kupovinu proizvoda.	Visok	Tekst zadatka	Kupac može pretraživati proizvode po cijeni i kategoriji te ih dodavati u košaricu.
F-012	Sustav omogućuje korisnicima ostavljanje recenzija i ocjena proizvoda nakon kupnje.	Srednji	Tekst zadatka	Korisnik može unijeti ocjenu i komentar koji su javno vidljivi.
F-013	Sustav omogućuje organizaciju i promociju izložbi keramičkih radova polaznika.	Srednji	Tekst zadatka	Izložba sadrži opis, slike, prijave i status odobrenja.
F-014	Sustav omogućuje polaznicima prijavu za sudjelovanje na izložbi.	Srednji	Tekst zadatka	Prijava se bilježi i organizator ima mogućnost odobrenja.
F-015	Sustav omogućuje sudionicima objavu komentara i fotografija nakon održane izložbe.	Nizak	Tekst zadatka	Komentari i slike su prikazani na stranici izložbe.
F-016	Sustav omogućuje korisnicima pretplatu na obavijesti o novim radionicama i proizvodima.	Srednji	Tekst zadatka	Pretplaćeni korisnici primaju obavijest u roku od 5 minuta nakon objave novog sadržaja.
F-017	Sustav omogućuje administratorima upravljanje korisnicima i odobravanje profila.	Visok	Tekst zadatka	Administrator može pregledavati, uređivati i odobravati korisničke račune.
F-018	Sustav omogućuje administratorima definiranje i ažuriranje cijena članarina.	Srednji	Tekst zadatka	Cijene se prikazuju u sučelju i primjenjuju pri registraciji organizatora.
F-019	Sustav mora primjenjivati kontrolu pristupa prema korisničkim ulogama (polaznik, organizator, administrator) i ograničiti pristup nedozvoljenim funkcijama.	Visoki	Tekst zadatka	Ako korisnik pokuša pristupiti funkciji koja nije dozvoljena njegovoj ulozi sustav mora prikazati poruku o nedozvoljenom pristupu.

2. Nefunkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-1.1	Sustav mora biti responzivan i prilagođen prikazu na uređajima različitih veličina ekrana (računala, tableti, mobilni uređaji).	Niski
NF-1.2	Sustav mora imati intuitivno korisničko sučelje koje omogućuje korisniku izvršavanje glavnih funkcija unutar najviše tri koraka.	Visok
NF-1.3	Vrijeme učitavanja svake stranice ne smije prelaziti 3 sekunde pri prosječnoj brzini internetske veze od 10 Mbps.	Srednji
NF-1.4	Sustav mora podržavati istovremeni rad najmanje 100 aktivnih korisnika bez značajnog pada performansi (manje od 10% usporenja).	Srednji
NF-1.5	Sustav mora sinkronizirati vanjske kalendare i servise unutar 10 sekundi od promjene podataka.	Srednji
NF-2.1	Sustav mora osigurati sigurnu autentifikaciju korisnika korištenjem protokola koji podržavaju dvofaktorsku provjeru identiteta.	Srednji
NF-2.2	Sustav mora pohranjivati i obrađivati osobne podatke korisnika u skladu s važećom GDPR regulativom.	Niski
NF-2.3	Sav mrežni promet između klijenta i poslužitelja mora biti šifriran korištenjem sigurnosnog protokola s certifikatom.	Visoki
NF-3.1	Sustav mora biti izrađen na način koji omogućuje jednostavnu izmjenu i proširenje funkcionalnosti u roku kraćem od 2 dana po izmjeni.	Srednji
NF-3.2	Sustav mora sadržavati tehničku dokumentaciju koja uključuje opis arhitekture, modula i API-ja, priručnik za korištenje s opisom osnovnih funkcionalnosti i postupaka rada te plan implementacije koji omogućuje postavljanje sustava u novo okruženje u roku od najviše 4 sata.	Visoki

---

3. Dionici i aktori

ID aktora	Uloga	Opis	Povezani funkcionalni zahtjevi
A-1	**Korisnik**	Krajnji korisnik koji sudjeluje u radionicama i kupuje proizvode.	F-001, F-002, F-005, F-009, F-011, F-012, F-014, F-015, F-016

ID aktora	Uloga	Opis	Povezani funkcionalni zahtjevi
A-2	**Organizator**	Osoba koja vodi radionice, prodaje radove i sudjeluje u izložbama.	F-001, F-002, F-003, F-004, F-007, F-008, F-010, F-013, F-017
A-3	**Administrator**	Osoba koja upravlja sustavom, cijenama članarina i korisnicima.	F-017, F-018
A-4	**Vanjski servisi**	Usluge koje se integriraju sa sustavom (OAuth, Google kalendar, PayPal).	F-002, F-006, F-007, F-008
A-5	**ClayPlay aplikacija (sustav)**	Slanje obavijesti pretplaćenim korisnicima, validacija vremena otkazivanja	F-009, F-016
A-6	**Baza podataka**	Pasivni sudionik koji pohranjuje, dohvaća i ažurira sve podatke sustava (korisnici, profili, radionice, rezervacije, plaćanja, proizvodi, izložbe, članarine).	F-001—F-019
A-7	**Banka**	Omogućuje kartično plaćanje.	F-007, F-008, F-011

Obrasci uporabe

UC1 – Prijava putem OAuth 2.0 servisa

Glavni sudionik: Korisnik
Cilj: Pristupiti sustavu korištenjem vanjske autentikacije
Sudionici: Vanjski servis za autentikaciju (OAuth 2.0)
Preduvjet: Korisnik ima račun na odabranom vanjskom servisu (npr. Google, GitHub)

Opis osnovnog tijeka:

1. Korisnik odabire opciju *"Prijava putem Google/GitHub"*.
2. Sustav preusmjerava korisnika na vanjski servis za autentifikaciju.
3. Korisnik unosi vjerodajnice na vanjskom servisu.
4. Vanjski servis potvrđuje identitet i vraća korisnika u aplikaciju.
5. Sustav automatski kreira sesiju i dodjeljuje korisničku ulogu.
6. Korisnik dobiva pristup platformi.

Opis mogućih odstupanja:

- 3.a Korisnik unosi neispravne vjerodajnice
  → Vanjski servis prikazuje poruku o pogrešci.
  → Korisnik može pokušati ponovno ili odustati od prijave.
- 4.a Vanjski servis nije dostupan
  → Sustav prikazuje poruku "Vanjski servis trenutačno nije dostupan. Pokušajte kasnije."

UC2 – Registracija novog korisnika

Glavni sudionik: Korisnik
Cilj: Kreirati novi korisnički račun kao polaznik ili organizator
Sudionici: Baza podataka
Preduvjet: Korisnik nije prethodno registriran

Opis osnovnog tijeka:

1. Korisnik odabire opciju *"Registracija"*.
2. Korisnik unosi osobne podatke (ime, e-mail, lozinku, ulogu).

3. Sustav provjerava valjanost i jedinstvenost e-mail adrese.
4. Sustav sprema korisničke podatke u bazu podataka.
5. Korisnik prima poruku o uspješnoj registraciji i mogućnosti prijave.

Opis mogućih odstupanja:

- 3.a E-mail adresa već postoji u bazi
  → Sustav prikazuje poruku "Račun s ovom adresom već postoji."
- 2.a Neispravan format unosa (npr. prekratka lozinka)
  → Sustav prikazuje poruku o pogrešci i vraća korisnika na ispravak.

UC3 – Uređivanje profila korisnika

Glavni sudionik: Organizator/korisnik

Cilj: Ažurirati javni profil s osobnim i poslovnim informacijama
Sudionici: Baza podataka
Preduvjet: Organizator/korisnik je prijavljen u sustav

Opis osnovnog tijeka:

1. Organizator/korisnik otvara svoj profil i odabire opciju *"Uredi profil"*.
2. Uređuje naziv studija, opis, adresu, kontakt i profilnu sliku.
3. Sustav validira unesene podatke.
4. Sustav sprema promjene u bazu podataka.
5. Ažurirani podaci postaju vidljivi na javnoj stranici profila.

Opis mogućih odstupanja:

- 2.a Neispravan format datoteke slike
  → Sustav prikazuje upozorenje i traži ponovno učitavanje ispravne datoteke.
- 4.a Baza nije dostupna
  → Sustav prikazuje poruku "Trenutno nije moguće spremiti promjene."

UC4 – Kreiranje i uređivanje radionice

Glavni sudionik: Organizator
Cilj: Dodati novu radionicu keramike s pripadajućim detaljima
Sudionici: Baza podataka, Google kalendar servis
Preduvjet: Organizator ima aktivno članstvo

Opis osnovnog tijeka:

1. Organizator odabire *"Nova radionica"*.
2. Unosi naziv, opis, datum, trajanje, lokaciju, broj mjesta i cijenu.
3. Sustav validira podatke i provjerava dostupnost termina.
4. Sustav sprema radionicu u bazu i sinkronizira je s kalendarom.
5. Polaznici mogu vidjeti radionicu na javnom popisu.

Opis mogućih odstupanja:

- 3.a Termin se preklapa s postojećom radionicom
  → Sustav prikazuje upozorenje i predlaže alternativni termin.
- 4.a Google kalendar nije dostupan
  → Sustav pohranjuje podatke lokalno i ponavlja sinkronizaciju kasnije.

UC5 – Pregled i rezervacija radionica

Glavni sudionik: Polaznik
Cilj: Pregledati dostupne radionice i rezervirati mjesto
Sudionici: Sustav, Google kalendar
Preduvjet: Polaznik je prijavljen

Opis osnovnog tijeka:

1. Polaznik otvara stranicu s radionicama.
2. Sustav dohvaća i prikazuje popis radionica iz baze i kalendara.
3. Polaznik odabire željenu radionicu i klikne *"Rezerviraj"*.
4. Sustav provjerava raspoloživa mjesta.
5. Polaznik potvrđuje rezervaciju.
6. Sustav pohranjuje rezervaciju i prikazuje poruku o uspjehu.

Opis mogućih odstupanja:

- 4.a Nema slobodnih mjesta
  → Sustav prikazuje poruku "Sva mjesta su popunjena."
- 5.a Korisnik odustane prije potvrde
  → Sustav otkazuje postupak bez promjena u bazi.

UC6 – Plaćanje termina radionice

Glavni sudionik: Polaznik
Cilj: Platiti rezervirani termin radionice putem vanjskog servisa
Sudionici: PayPal servis / sustav kartičnog plaćanja
Preduvjet: Postoji potvrđena rezervacija radionice

Opis osnovnog tijeka:

1. Polaznik odabire *"Plati"* na rezerviranoj radionici.
2. Sustav prikazuje iznos i metode plaćanja (PayPal, kartica).
3. Polaznik odabire željeni način plaćanja.
4. Sustav preusmjerava korisnika na vanjski servis.
5. Plaćanje se izvršava i potvrđuje.
6. Sustav bilježi transakciju i označava radionicu kao plaćenu.

Opis mogućih odstupanja:

- 5.a Plaćanje odbijeno
  → Sustav prikazuje poruku "Plaćanje nije uspjelo."
- 4.a Vanjski servis nije dostupan
  → Sustav prikazuje poruku i omogućuje ponovno pokušavanje.

UC7 – Odabir i plaćanje članarine

Glavni sudionik: Organizator
Cilj: Platiti mjesečnu ili godišnju članarinu
Sudionici: PayPal servis / sustav kartičnog plaćanja
Preduvjet: Organizator ima aktivan račun

Opis osnovnog tijeka:

1. Organizator otvara stranicu *"Članstvo"*.
2. Odabire plan (mjesečni ili godišnji).
3. Sustav prikazuje cijenu i način plaćanja.
4. Organizator izvršava plaćanje putem odabranog servisa.
5. Sustav bilježi transakciju i aktivira članstvo.
6. Organizator dobiva potvrdu o uspješnom plaćanju.

Opis mogućih odstupanja:

- 4.a Plaćanje odbijeno
  → Sustav prikazuje poruku "Plaćanje nije uspjelo."
- 5.a Neuspjelo spremanje transakcije
  → Sustav prikazuje upozorenje i traži ponovno učitavanje stranice.

UC8 – Kupovina proizvoda u online trgovini

Glavni sudionik: Korisnik (organizator ili polaznik)
Cilj: Pregledati, filtrirati i kupiti keramički proizvod
Sudionici: PayPal servis / sustav kartičnog plaćanja
Preduvjet: Kupac ima korisnički račun

Opis osnovnog tijeka:

1. Korisnik otvara online trgovinu.
2. Filtrira proizvode po kategoriji ili cijeni.
3. Odabire željeni proizvod i dodaje ga u košaricu.
4. Korisnik prelazi na blagajnu i odabire način plaćanja.
5. Sustav preusmjerava na PayPal/kartični servis.
6. Nakon uspješnog plaćanja, kupac dobiva potvrdu i kontakt prodavača.

Opis mogućih odstupanja:

- 5.a Plaćanje neuspješno
  → Sustav prikazuje poruku o grešci.
- 3.a Proizvod nije dostupan
  → Sustav prikazuje obavijest "Proizvod više nije na zalihi."

UC9 – Prijava na izložbu i odobrenje

Glavni sudionik: Polaznik
Cilj: Prijaviti se za sudjelovanje na izložbi i čekati odobrenje organizatora
Sudionici: Organizator, Baza podataka
Preduvjet: Izložba je otvorena za prijave

Opis osnovnog tijeka:

1. Polaznik pregledava dostupne izložbe.
2. Odabire željenu izložbu i klikne *"Prijavi se"*.
3. Sustav bilježi prijavu i šalje obavijest organizatoru.
4. Organizator pregledava prijave.
5. Organizator odobrava ili odbija prijavu.
6. Polaznik prima obavijest o statusu prijave.

Opis mogućih odstupanja:

- 2.a Izložba je zatvorena za prijave
  → Sustav prikazuje poruku “Prijave su završene.”
- 5.a Organizator ne reagira unutar roka
  → Sustav šalje automatski podsjetnik.

UC10 – Upravljanje korisnicima i članarinama (Administrator)

Glavni sudionik: Administrator
Cilj: Odobriti nove profile, upravljati korisnicima i definirati cijene članarina
Sudionici: baza podataka
Preduvjet: Administrator je prijavljen

Opis osnovnog tijeka:

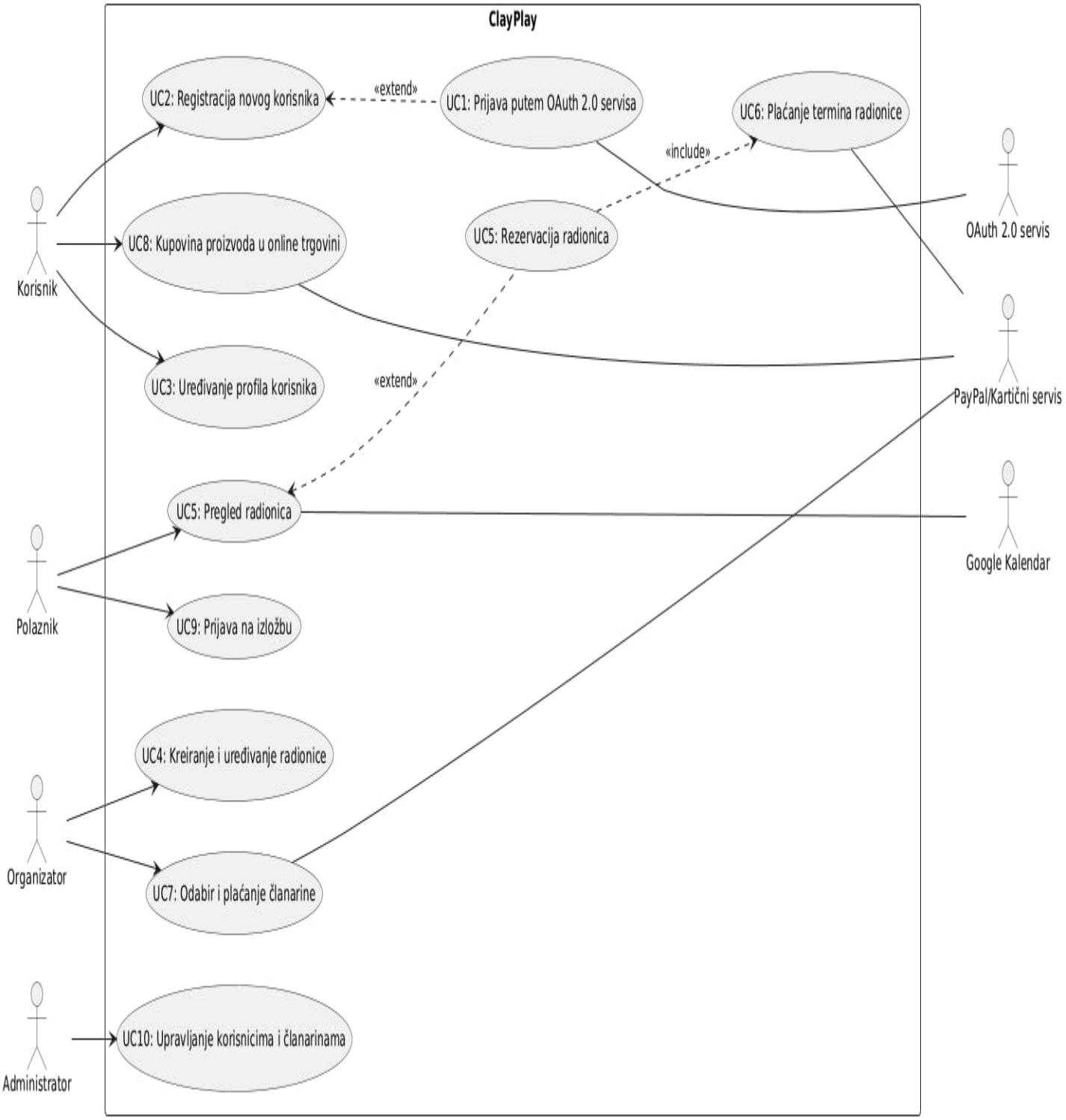
1. Administrator otvara nadzornu ploču.
2. Pregledava nove i postojeće korisnike.
3. Odobrava ili blokira korisničke račune.
4. Otvara sekciju *“Cijene članarina”*.
5. Ažurira iznose mjesečnih i godišnjih članarina.
6. Sustav sprema promjene i prikazuje ih organizatorima.

Opis mogućih odstupanja:

- 3.a Administrator pokuša odobriti već odobren račun
  → Sustav prikazuje poruku “Račun je već aktivan.”
- 5.a Sustav ne može spremiti nove cijene
  → Sustav prikazuje upozorenje “Neuspjelo ažuriranje podataka.”

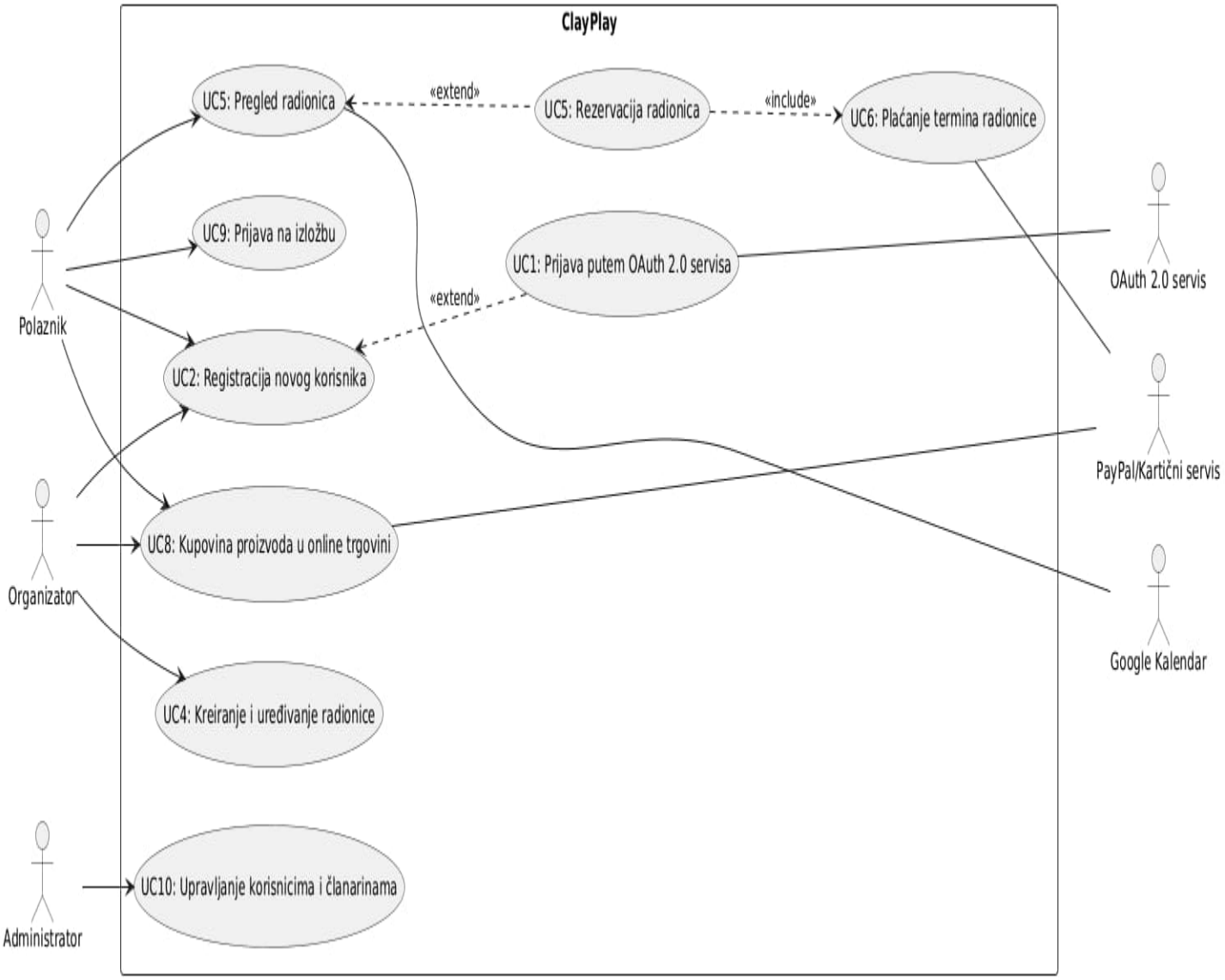
Dijagram obrazaca uporabe

1. Visokorazinski dijagram obrazaca uporabe cijelog sustava



Prikazuje glavne funkcionalnosti sustava i osnovne interakcije između sustava i ključnih korisničkih uloga (svaki korisnik, polaznik, organizator, administrator). Služi za brzo razumijevanje opsega i ciljeva sustava.

2. Dijagram obrazaca uporabe za ključne značajke

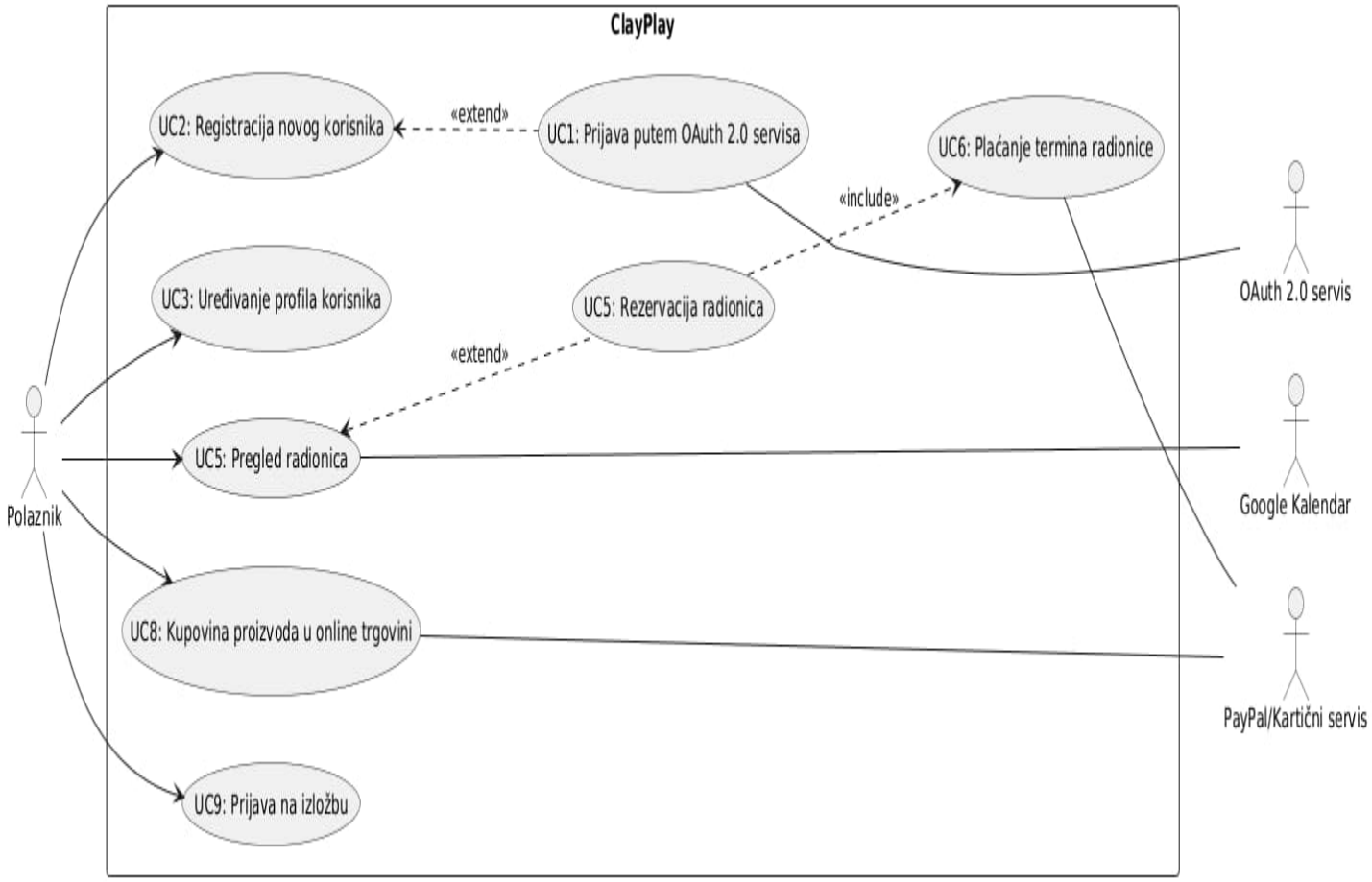


Detaljnije prikazuje ključne funkcionalnosti sustava kao što su prijava, registracija i procesi plaćanja te njihovu povezanost s vanjskim servisima (OAuth, PayPal, kartični sustavi).

3. Dijagram obrazaca uporabe prema korisničkim ulogama

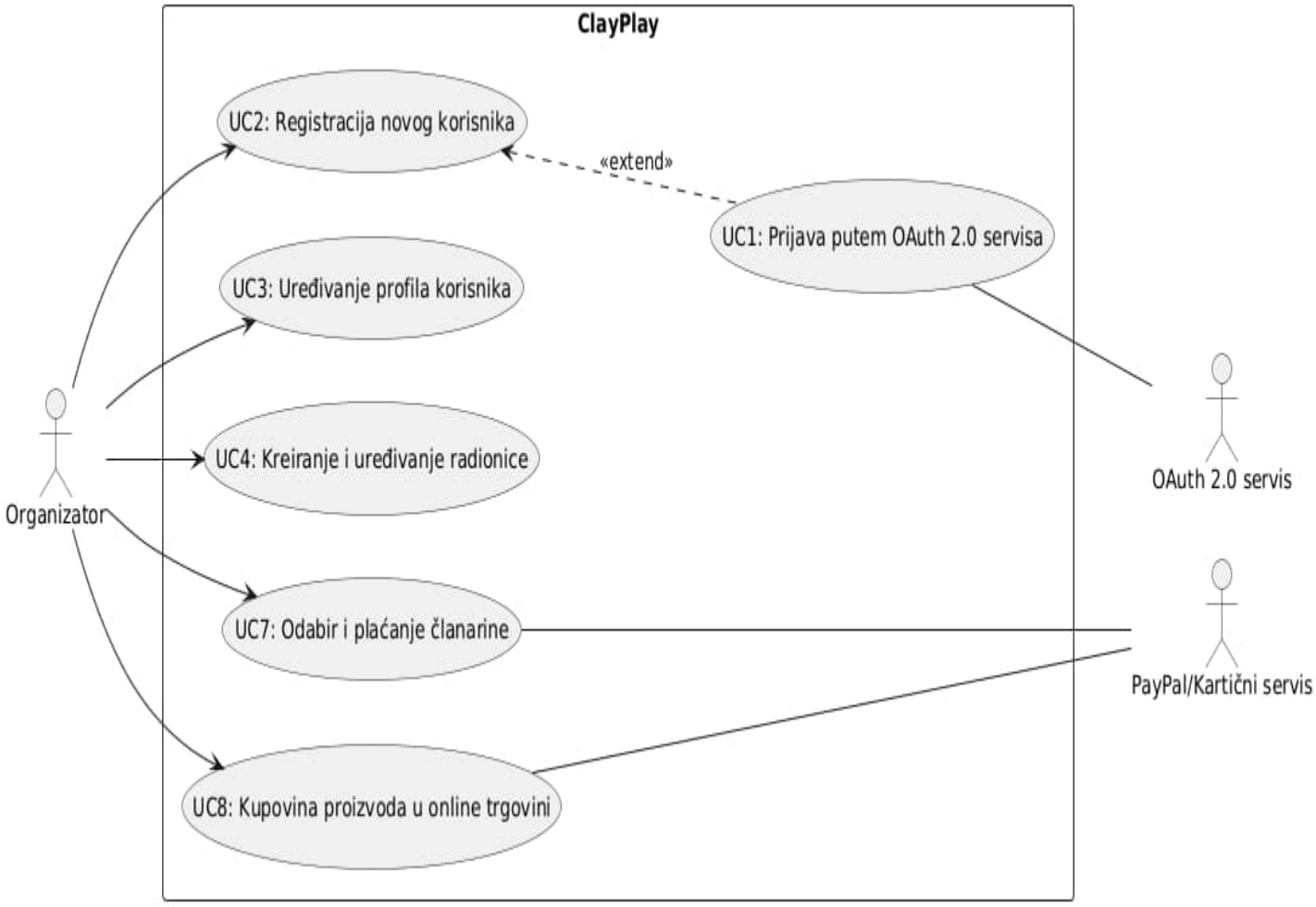
Prikazuje koje funkcionalnosti su dostupne pojedinim korisničkim ulogama (polaznik, organizator, administrator) te jasno razdvaja njihove odgovornosti i ovlasti u sustavu.

3.1. Polaznik



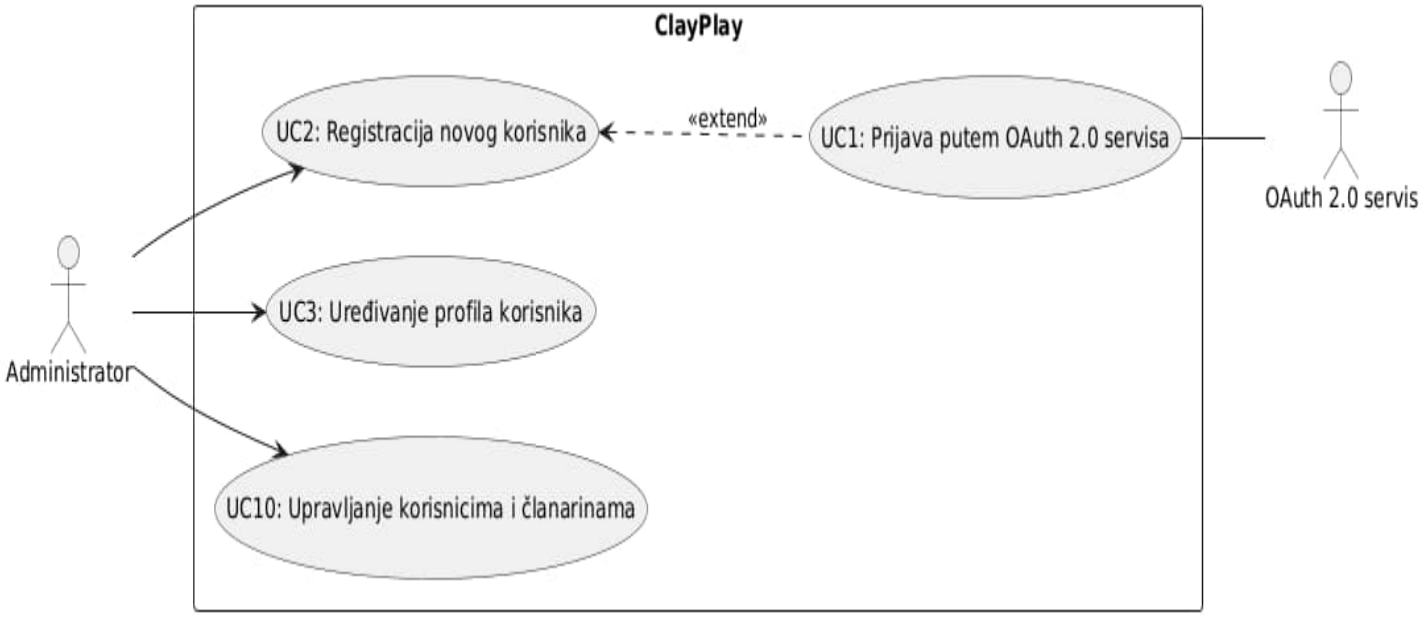
Prikazuje sve funkcionalnosti za polaznika.

3.2. Organizator



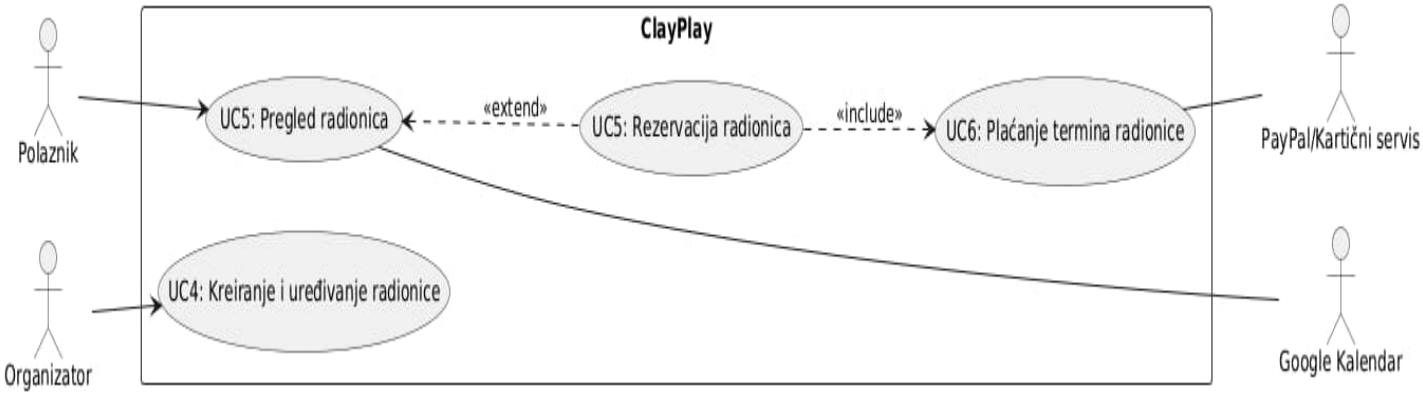
Prikazuje sve funkcionalnosti za organizatora.

3.3. Administrator



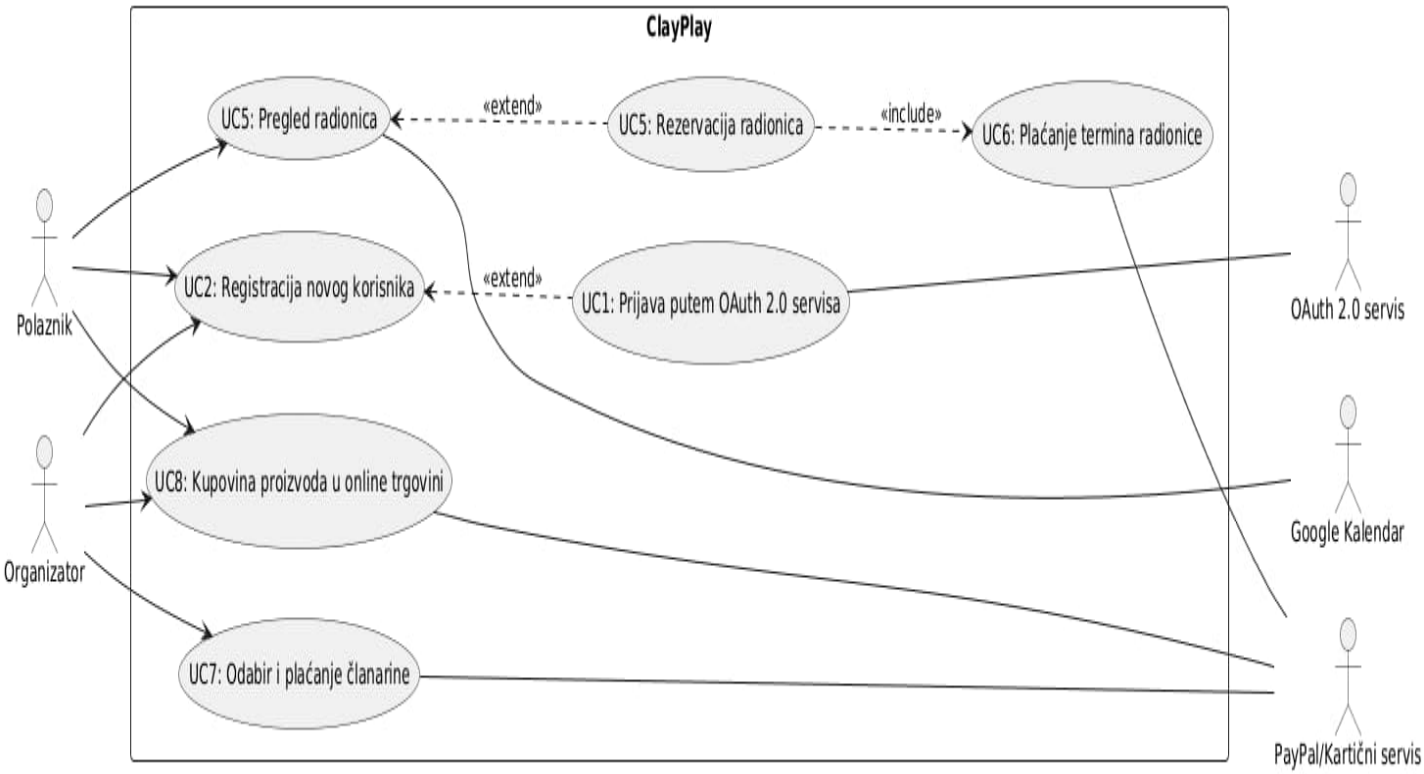
Prikazuje sve funkcionalnosti za administratora.

4. Dijagram obrazaca uporabe za osnovne poslovne procese



Opisuje kako sustav podržava glavne poslovne procese, poput organizacije radionica, rezervacije termina i plaćanja, te prikazuje međusobnu povezanost tih procesa.

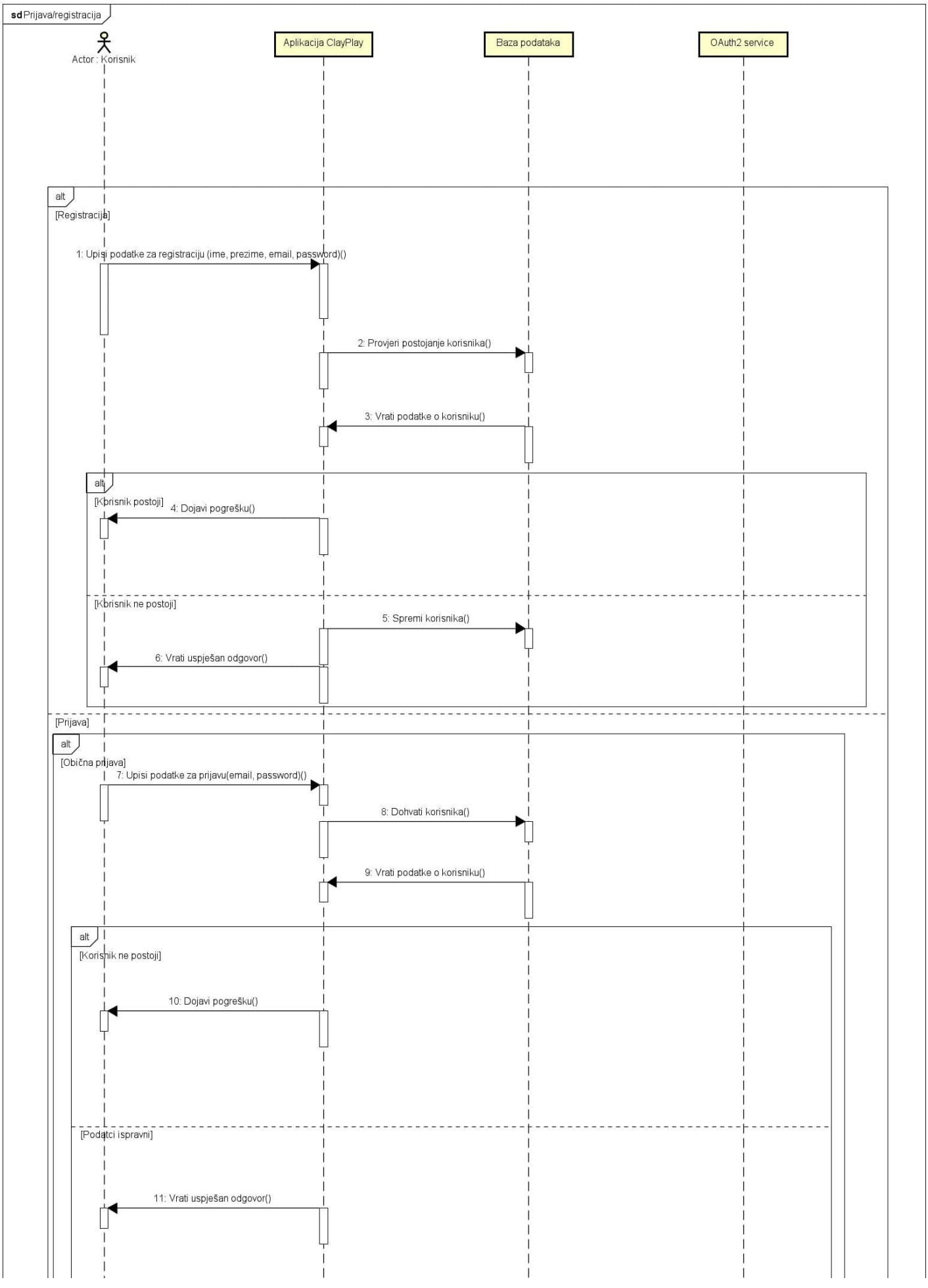
5. Dijagram obrazaca uporabe za kritične sustave i integracije

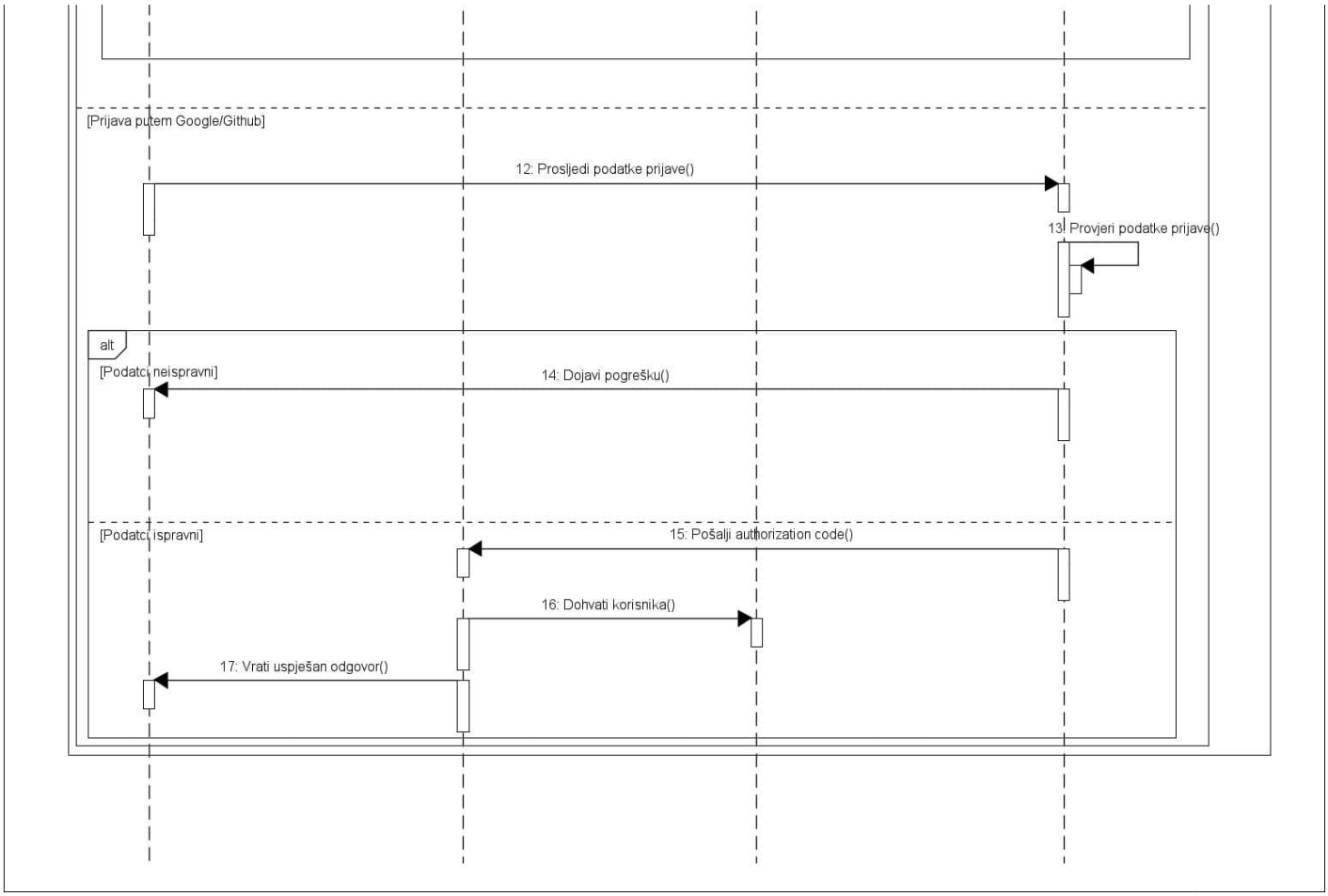


Prikazuje interakciju sustava s vanjskim servisima (OAuth 2.0, Google Kalendar, PayPal/kartični sustavi) koji su ključni za ispravno funkcioniranje autentikacije, sinkronizacije termina i plaćanja.

Sekvencijski dijagrami

SD1 - Prijava/registracija korisnika





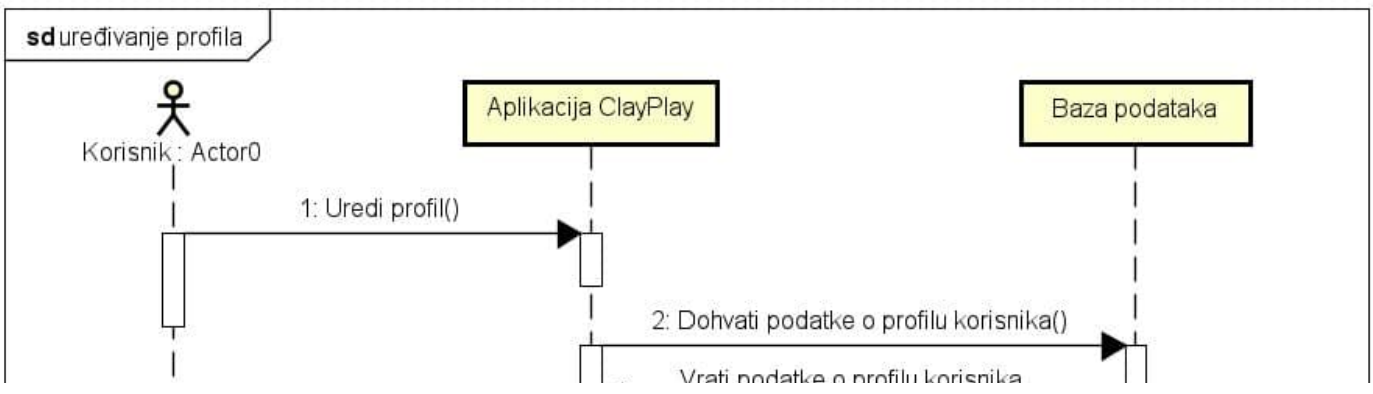
Sekvencijski dijagram prikazuje komunikaciju između aktera Korisnik, aplikacije ClayPlay, baze podataka i vanjskog OAuth2 servisa tijekom procesa registracije i prijave korisnika.

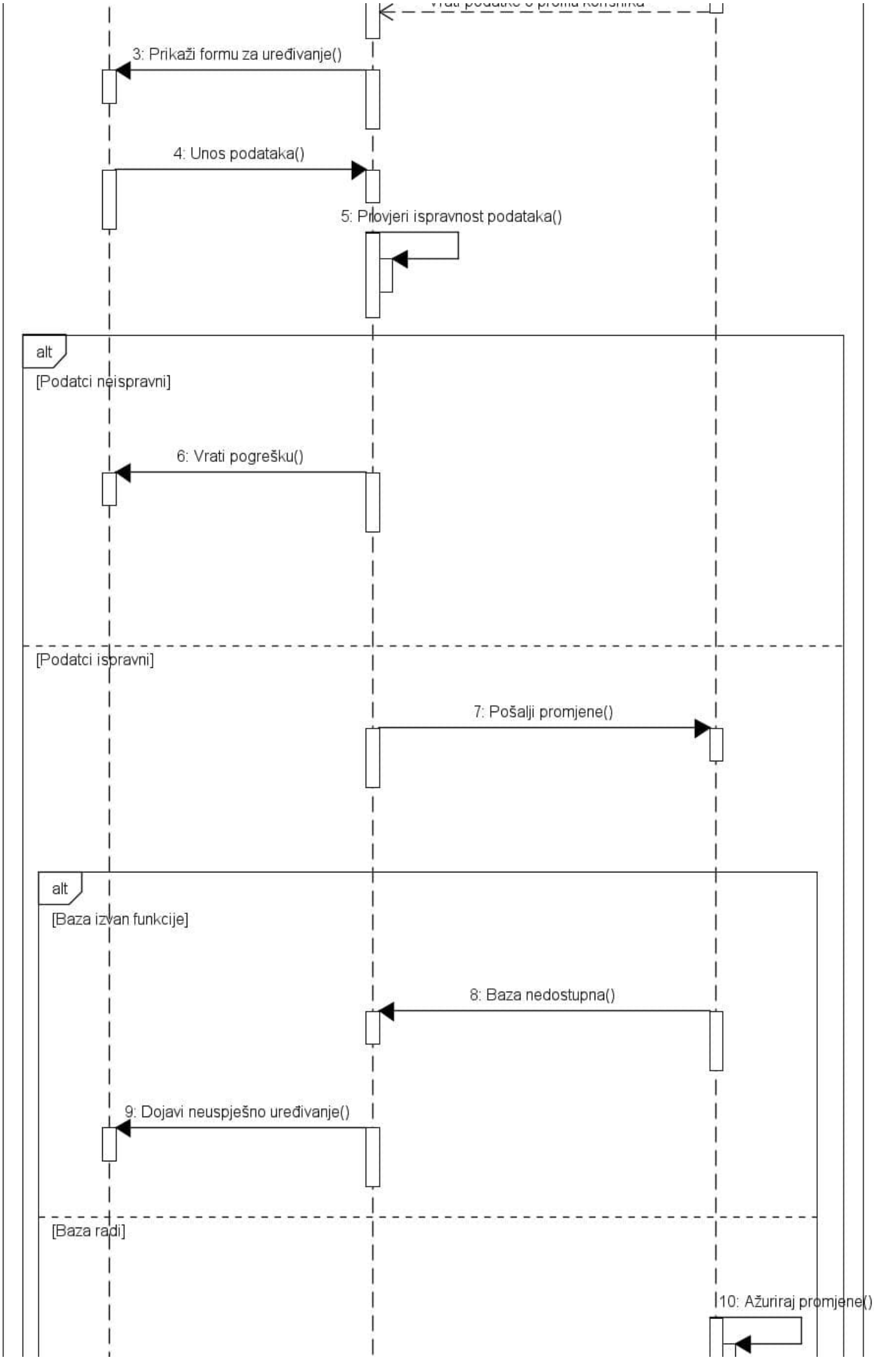
Kod registracije, korisnik unosi svoje podatke, aplikacija provjerava u bazi postoji li već korisnik s istim e-mailom te, ako ne postoji, sprema novog korisnika i vraća uspješan odgovor. Ako korisnik već postoji, vraća se poruka o pogrešci.

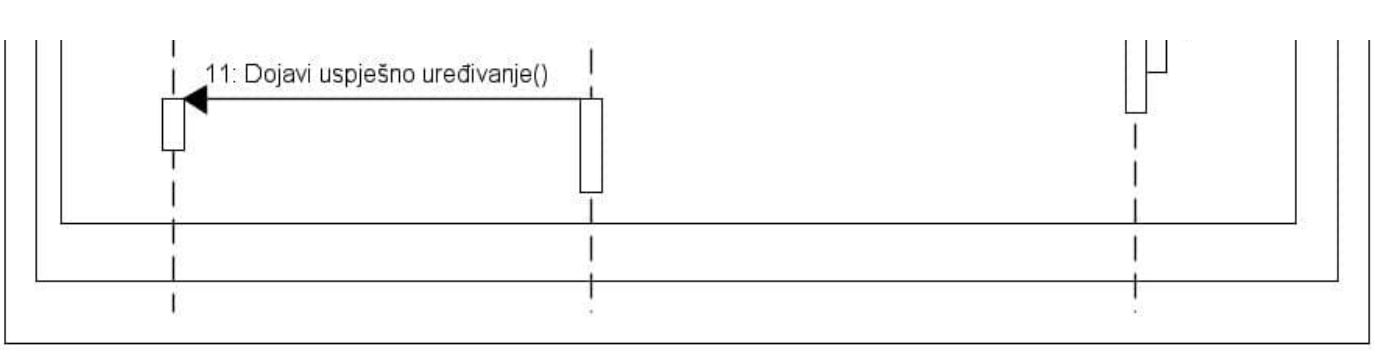
Kod klasične prijave, korisnik unosi e-mail i lozinku, aplikacija dohvaća podatke iz baze te, ovisno o ispravnosti, vraća uspješnu prijavu ili poruku o pogrešci.

Kod prijave putem vanjskog servisa (Google/GitHub), aplikacija komunicira s OAuth2 servisom koji provjerava podatke i vraća autorizacijski kod, nakon čega aplikacija dohvaća korisnika i potvrđuje prijavu.

SD2 – Uređivanje profila




3: Prikaži formu za uređivanje()
4: Unos podataka()
5: Provjeri ispravnost podataka()
alt
[Podatci neispravni]
6: Vrati pogrešku()
[Podatci ispravni]
7: Pošalji promjene()
alt
[Baza izvan funkcije]
8: Baza nedostupna()
9: Dojavi neuspješno uređivanje()
[Baza radi]
10: Ažuriraj promjene()

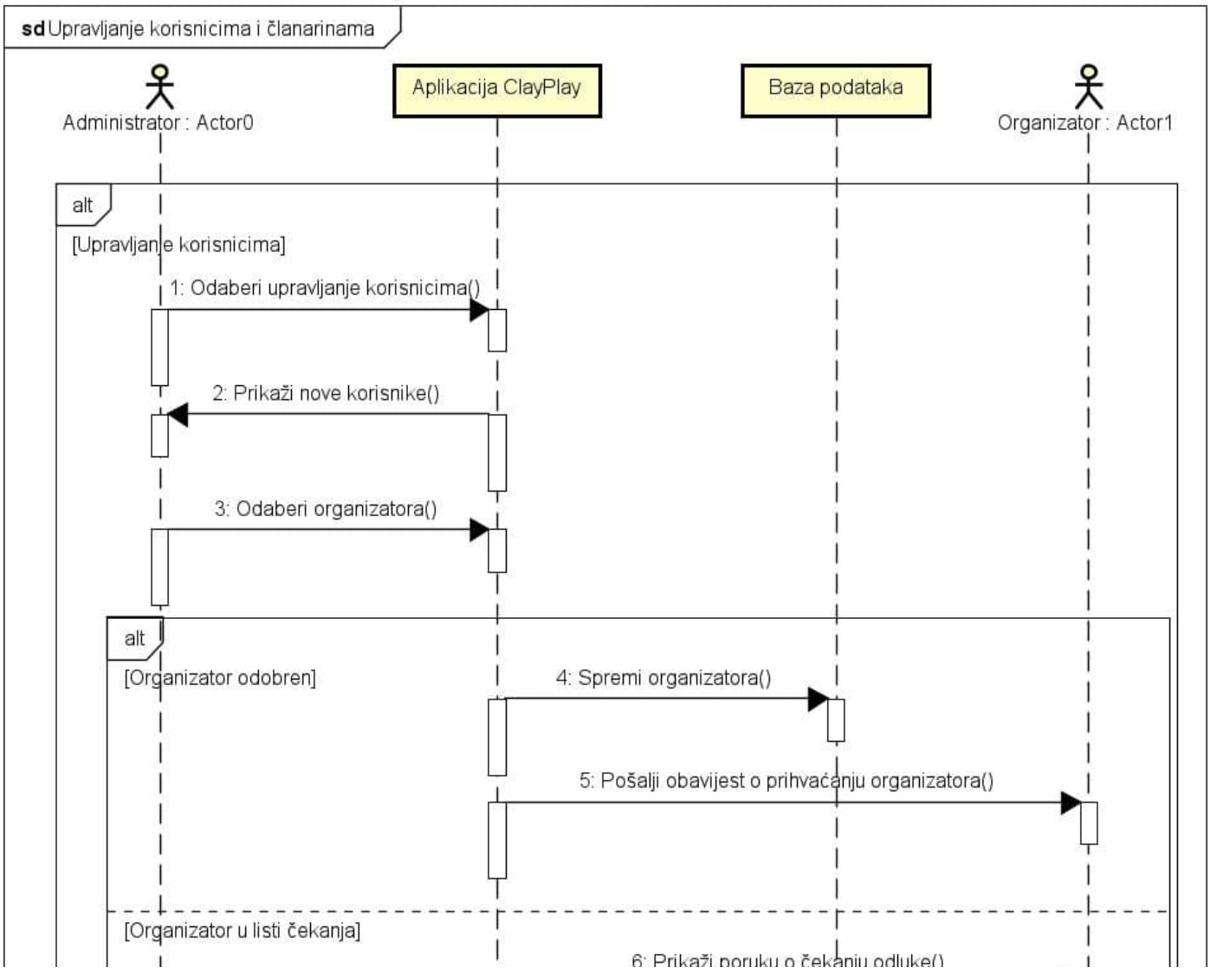


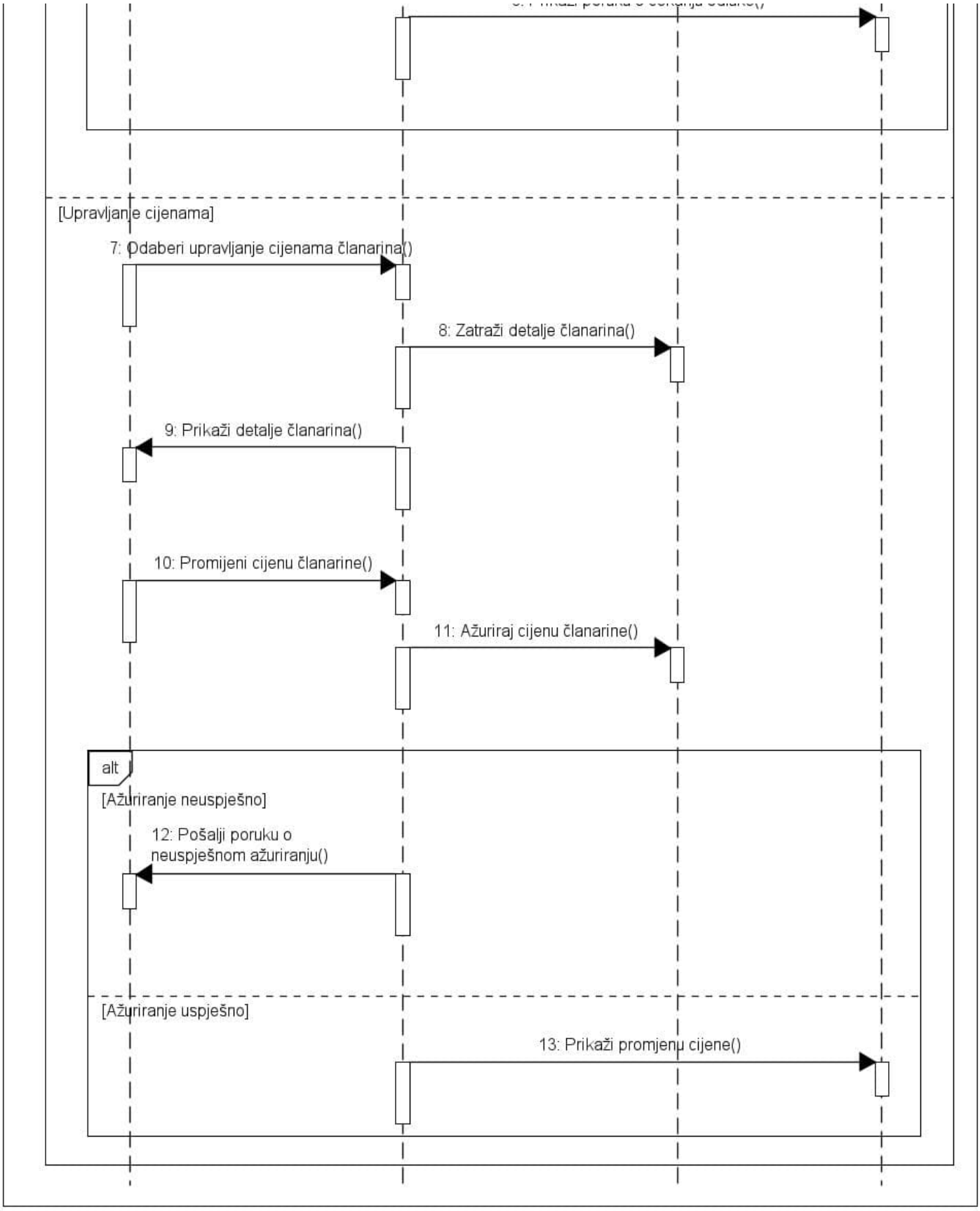
Sekvencijski dijagram prikazuje interakciju između aktera Korisnik (organizator ili polaznik), aplikacije ClayPlay i baze podataka tijekom procesa uređivanja profila.

Proces započinje kada korisnik odabere opciju za uređivanje profila, nakon čega aplikacija dohvaća postojeće podatke iz baze i prikazuje formu za uređivanje. Korisnik unosi nove podatke, a aplikacija provjerava njihovu ispravnost.

Ako su podaci neispravni, korisniku se vraća poruka o pogrešci. Ako su podaci ispravni, promjene se šalju u bazu podataka. U slučaju da baza nije dostupna, vraća se obavijest o neuspješnom uređivanju, dok se u suprotnom podaci ažuriraju i korisniku se vraća potvrda o uspješnom uređivanju profila.

SD3 - Upravljanje korisnicima i članarinama



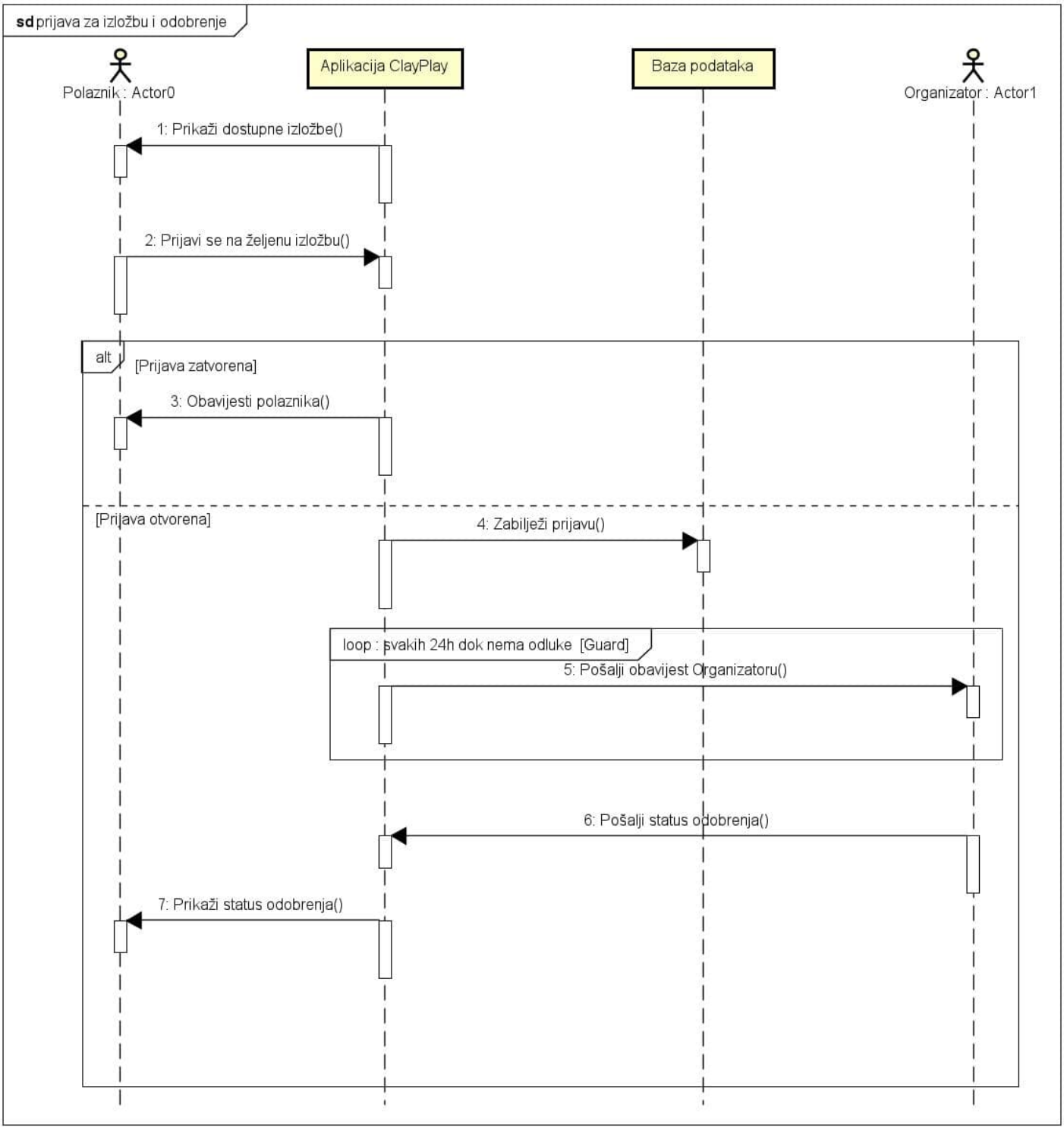


Sekvencijski dijagram prikazuje interakciju između aktera Administrator, Organizator, aplikacije ClayPlay i baze podataka tijekom procesa upravljanja korisnicima i članarinama.

U dijelu upravljanja korisnicima, administrator odabire opciju za upravljanje korisnicima te aplikacija prikazuje nove korisnike. Administrator zatim odabire organizatora, nakon čega se, ovisno o odluci, organizator ili odobrava i sprema u sustav uz slanje obavijesti, ili ostaje na listi čekanja uz prikaz poruke o čekanju odluke.

U dijelu upravljanja cijenama članarina, administrator dohvaća detalje članarina, mijenja cijenu te aplikacija ažurira podatke u bazi. Ako ažuriranje ne uspije, prikazuje se poruka o pogrešci, dok se u slučaju uspješnog ažuriranja organizatoru prikazuje nova cijena članarine.

SD4 - Prijava za izložbu i odobrenje



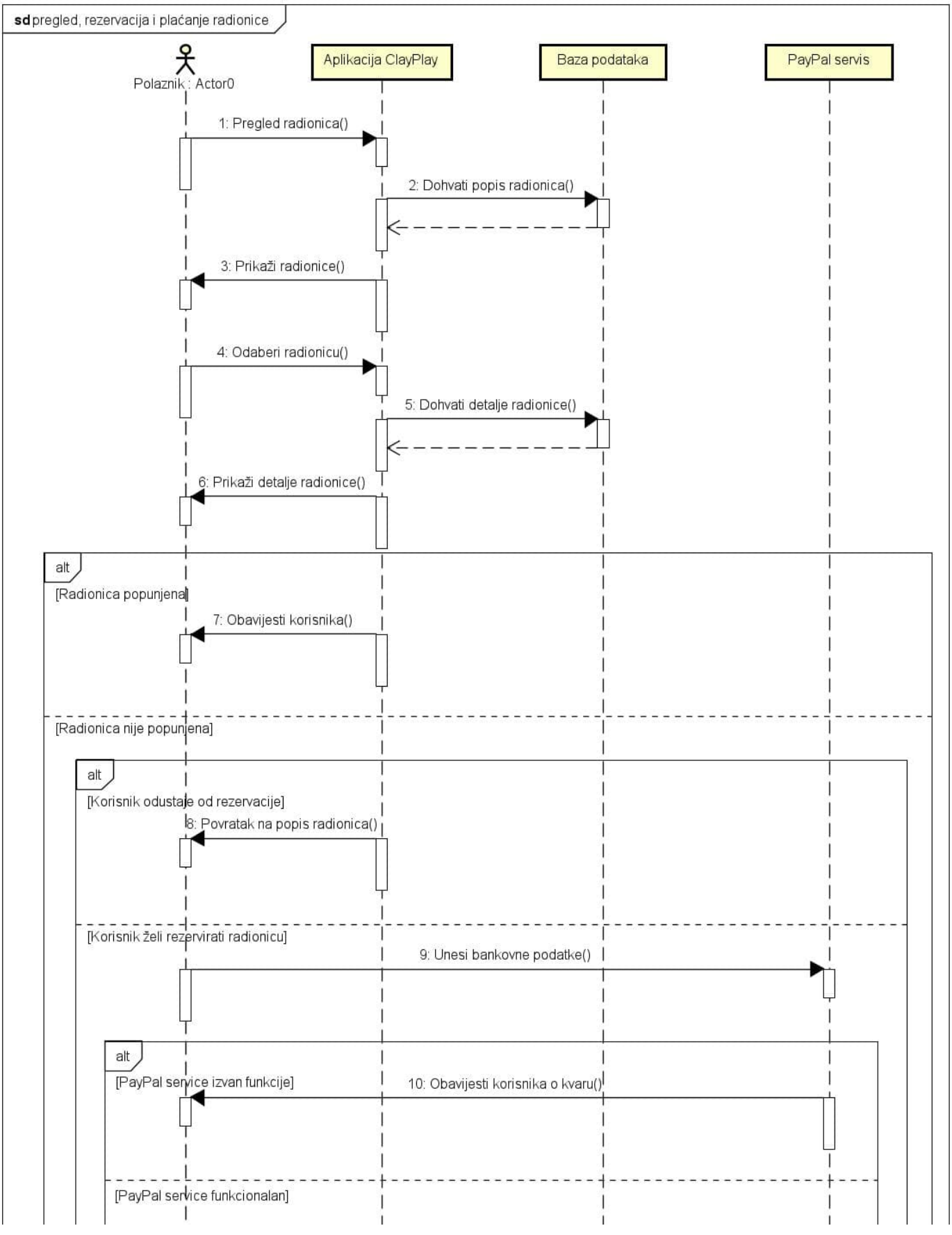
Sekvencijski dijagram prikazuje interakciju između aktera Polaznik, Organizator, aplikacije ClayPlay i baze podataka tijekom procesa prijave na izložbu i odobravanja prijave.

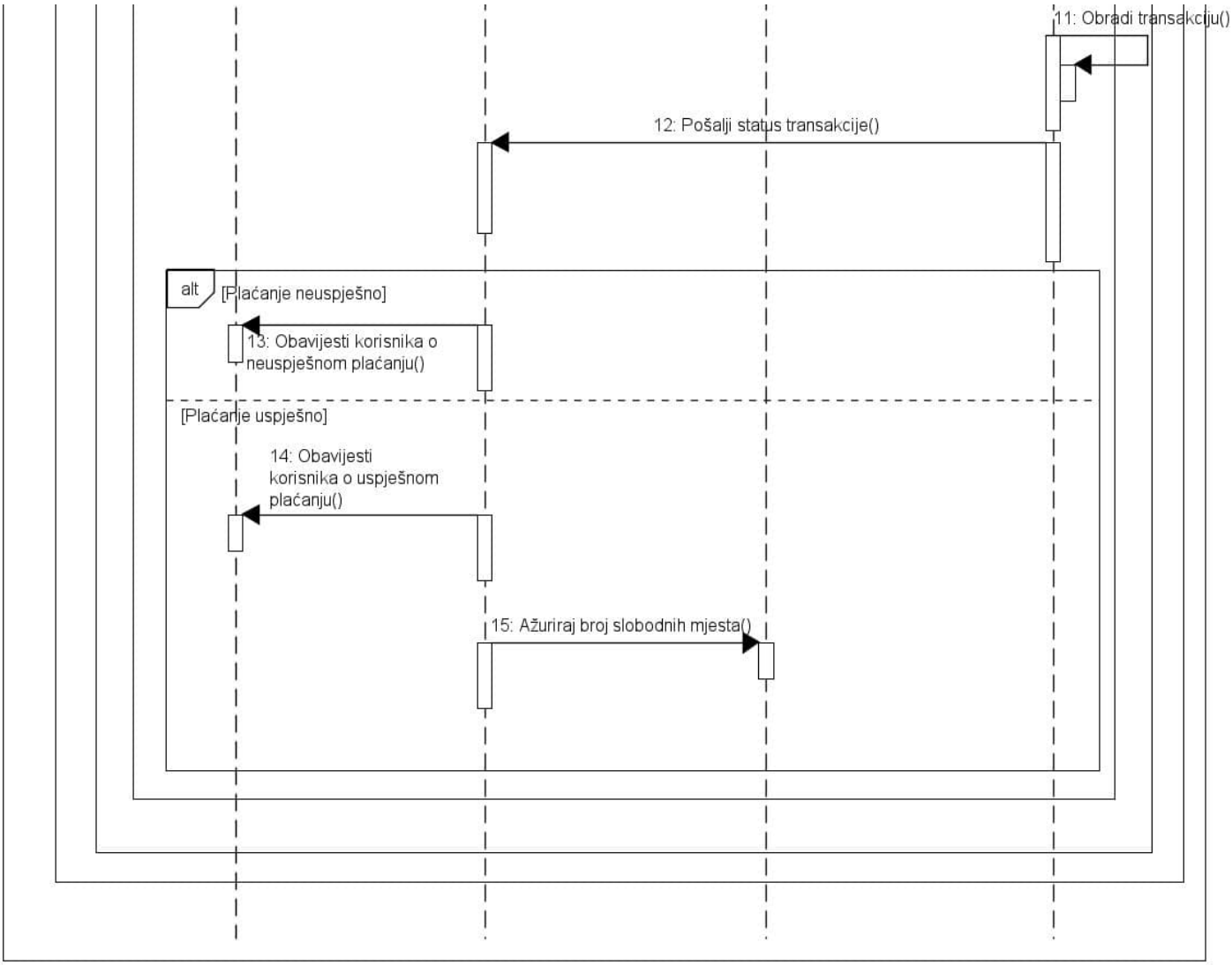
Proces započinje prikazom dostupnih izložbi polazniku, nakon čega se polaznik prijavljuje na željenu izložbu. Ako je prijava zatvorena, aplikacija obavještava polaznika o nemogućnosti prijave. Ako je prijava

otvorena, aplikacija bilježi prijavu u bazi podataka te periodički obavještava organizatora dok se ne donese odluka.

Nakon što organizator pošalje status odobrenja, aplikacija prikazuje polazniku konačni status prijave.

SD5 – Pregled, rezervacija i plaćanje radionice



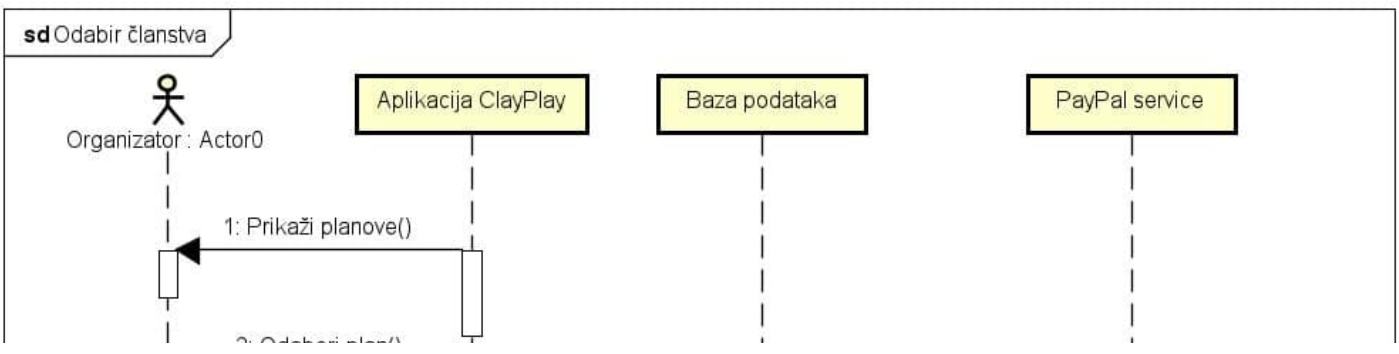


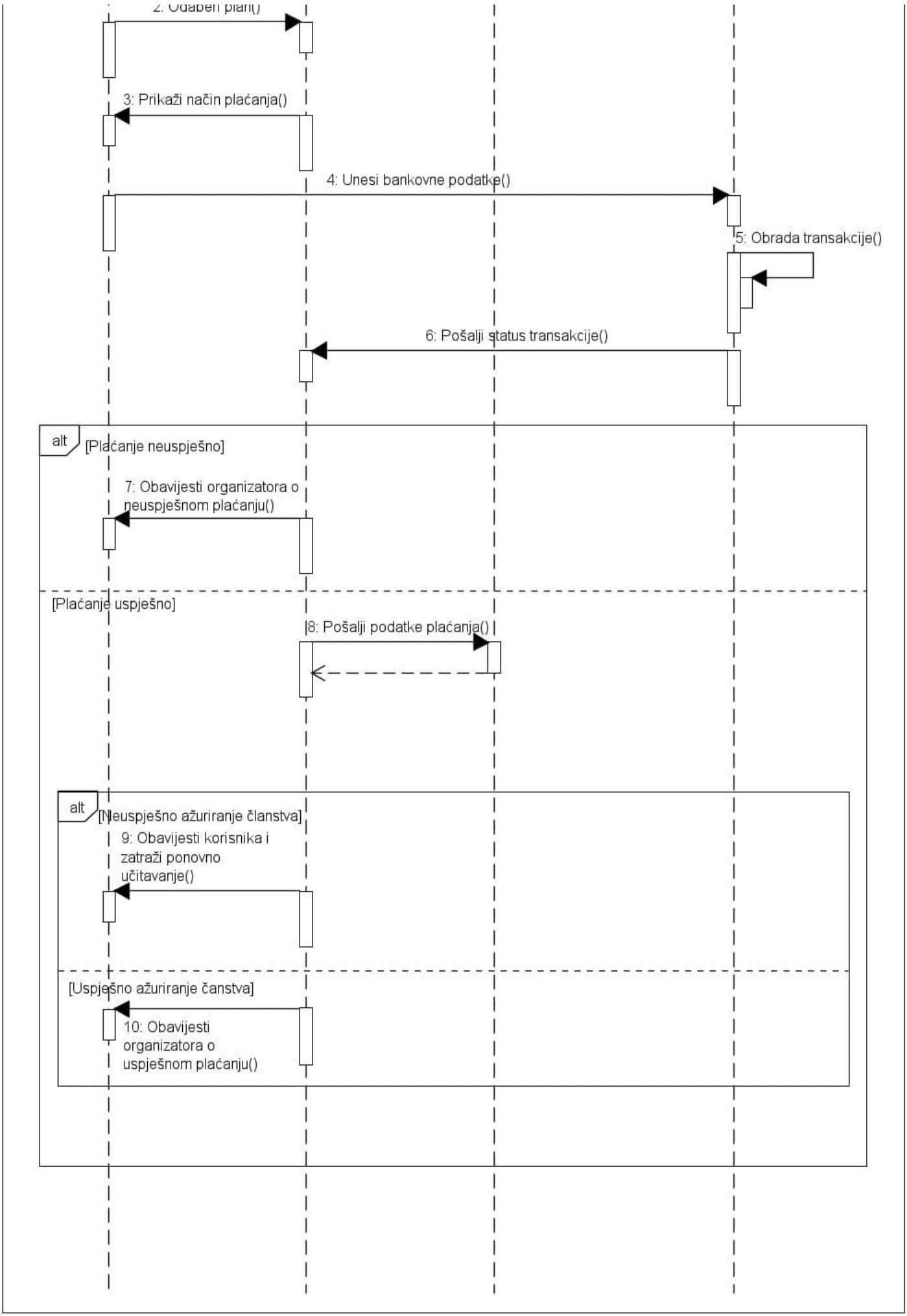
Sekvencijski dijagram prikazuje interakciju između aktera Polaznik, aplikacije ClayPlay, baze podataka i vanjskog PayPal servisa tijekom procesa pregleda, rezervacije i plaćanja radionice.

Proces započinje pregledom dostupnih radionica, pri čemu aplikacija dohvaća podatke iz baze i prikazuje ih polazniku. Nakon odabira radionice, prikazuju se njezini detalji.

Ako je radionica popunjena, korisnik se o tome obavještava. Ako radionica nije popunjena, korisnik može odustati od rezervacije ili nastaviti s unosom podataka za plaćanje. U slučaju nedostupnosti PayPal servisa, korisnik se obavještava o pogrešci. Ako je plaćanje uspješno, korisnik dobiva potvrdu, a aplikacija ažurira broj slobodnih mjesta u bazi podataka.

SD6 – Odabir članstva



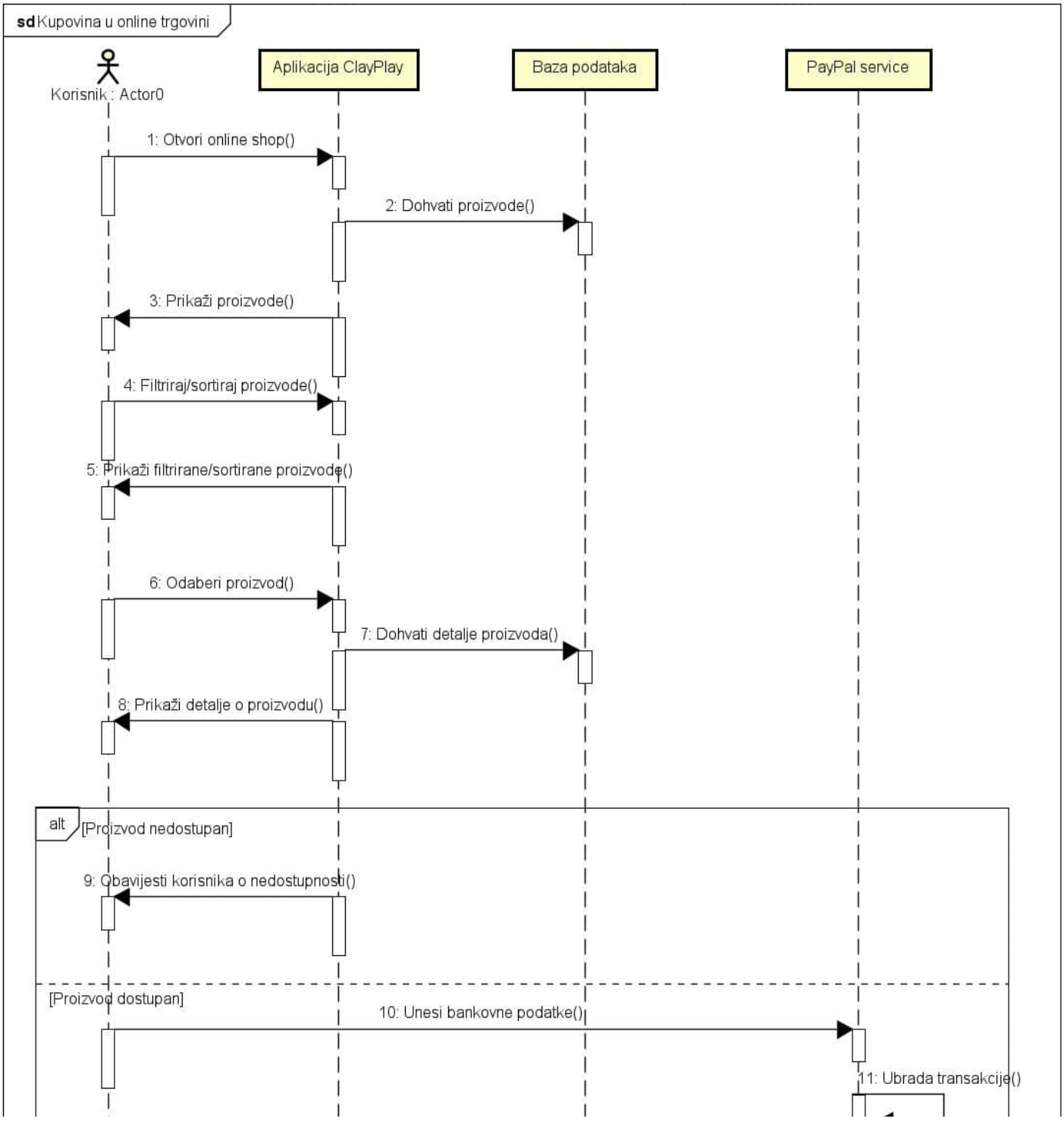

2: Odaberi plan()
3: Prikaži način plaćanja()
4: Unesi bankovne podatke()
5: Obrada transakcije()
6: Pošalji status transakcije()
alt
[Plaćanje neuspješno]
7: Obavijesti organizatora o neuspješnom plaćanju()
[Plaćanje uspješno]
8: Pošalji podatke plaćanja()
alt
[Neuspješno ažuriranje članstva]
9: Obavijesti korisnika i zatraži ponovno učitavanje()
[Uspješno ažuriranje članstva]
10: Obavijesti organizatora o uspješnom plaćanju()

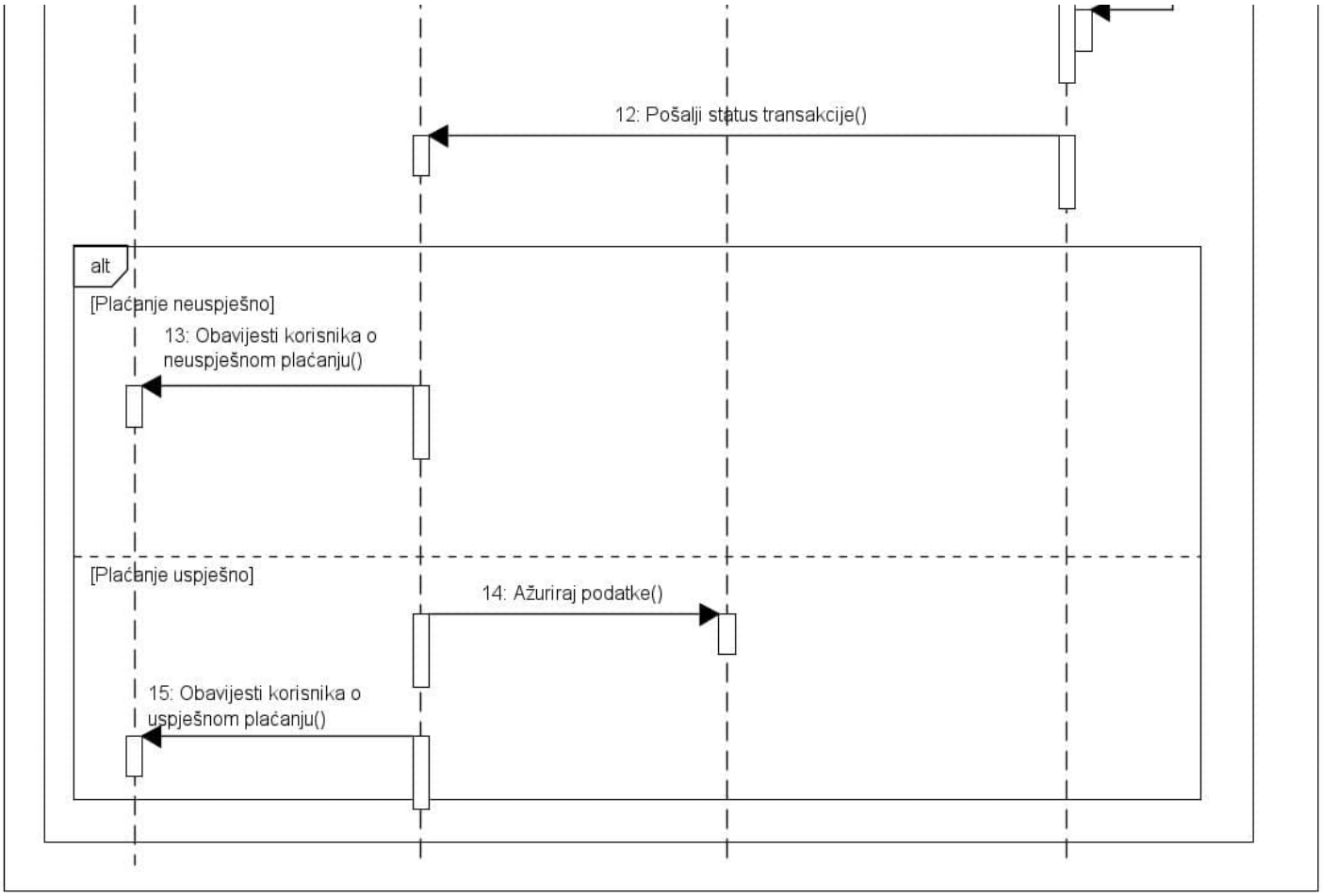
Sekvencijski dijagram prikazuje interakciju između aktera Organizator, aplikacije ClayPlay, baze podataka i vanjskog PayPal servisa tijekom procesa odabira i plaćanja članstva.

Proces započinje prikazom dostupnih planova članstva, nakon čega organizator odabire željeni plan i način plaćanja. Organizator zatim unosi podatke za plaćanje, koji se prosljeđuju PayPal servisu na obradu.

Ako plaćanje ne uspije, organizator se obavještava o neuspješnom plaćanju. Ako je plaćanje uspješno, podaci o plaćanju spremaju se u bazu podataka te se, ovisno o ishodu ažuriranja članstva, organizator obavještava o uspješnom ili neuspješnom ažuriranju članstva.

SD7 - Kupovina u online trgovini



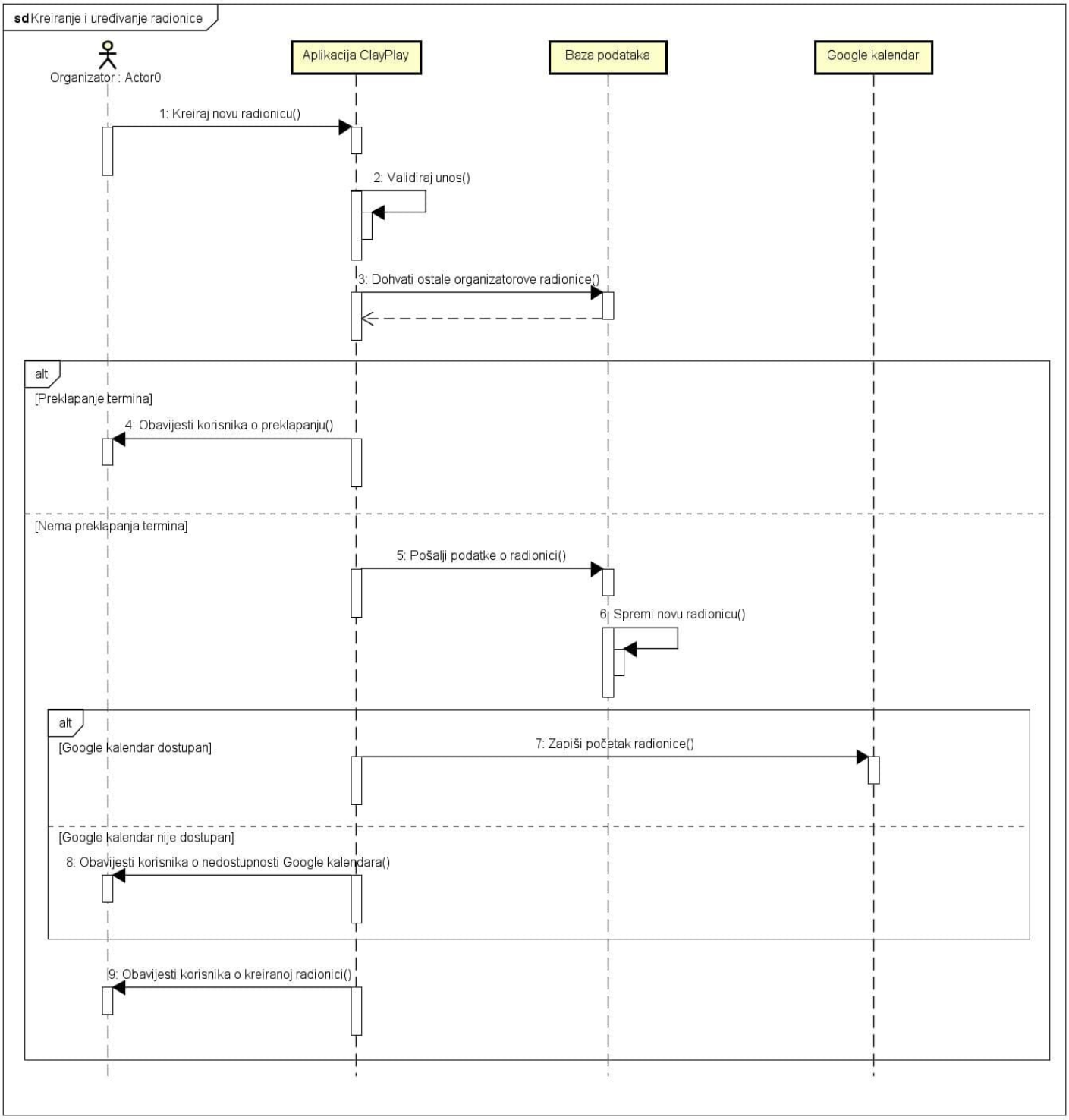


Sekvencijski dijagram prikazuje interakciju između aktera Korisnik, aplikacije ClayPlay, baze podataka i vanjskog PayPal servisa tijekom procesa kupovine u online trgovini.

Proces započinje otvaranjem online trgovine i prikazom proizvoda, pri čemu korisnik može filtrirati i sortirati proizvode. Nakon odabira proizvoda, aplikacija dohvaća i prikazuje njegove detalje.

Ako proizvod nije dostupan, korisnik se o tome obavještava. Ako je proizvod dostupan, korisnik unosi podatke za plaćanje, koji se šalju PayPal servisu na obradu. Ovisno o ishodu transakcije, korisnik se obavještava o neuspješnom ili uspješnom plaćanju, a u slučaju uspješnog plaćanja podaci se ažuriraju u bazi podataka.

SD8 – Kreiranje i uređivanje radionice



Sekvencijski dijagram prikazuje interakciju između aktera Organizator, aplikacije ClayPlay, baze podataka i vanjskog servisa Google kalendar tijekom procesa kreiranja i uređivanja radionice.

Proces započinje kada organizator unese podatke za novu radionicu, a aplikacija provjerava ispravnost unosa i dohvaća postojeće termine drugih radionica. Ako postoji preklapanje termina, korisnik se o tome obavještava.

Ako nema preklapanja, podaci o radionici spremaju se u bazu podataka. U slučaju dostupnosti Google kalendara, početak radionice se bilježi u kalendar, dok se u suprotnom korisnik obavještava o nedostupnosti servisa. Na kraju se organizatoru prikazuje potvrda o uspješnom kreiranju radionice.

Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

ID zahtjeva	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10
F-001	Da	Da	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
F-002	Da	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
F-003	Ne	Ne	Da	Ne	Ne	Ne	Ne	Ne	Ne	Ne
F-004	Ne	Ne	Ne	Da	Ne	Ne	Ne	Ne	Ne	Ne
F-005	Ne	Ne	Ne	Ne	Da	Ne	Ne	Ne	Ne	Ne
F-006	Ne	Ne	Ne	Da	Da	Ne	Ne	Ne	Ne	Ne
F-007	Ne	Ne	Ne	Ne	Ne	Da	Ne	Ne	Ne	Ne
F-008	Ne	Ne	Ne	Ne	Ne	Ne	Da	Ne	Ne	Ne
F-009	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
F-010	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Da	Ne	Ne
F-011	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Da	Ne	Ne
F-012	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
F-013	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Da	Ne
F-014	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Da	Ne
F-015	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
F-016	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne
F-017	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Ne	Da
F-018	Ne	Ne	Ne	Ne	Ne	Ne	Da	Ne	Ne	Da
F-019	Da	Da	Da	Da	Da	Da	Da	Da	Da	Da

1. Opis arhitekture

Stil arhitekture

Sustav ClayPlay koristi **klijent–poslužitelj** arhitektonski stil.

- **Klijent:** React web aplikacija služi kao korisničko sučelje koje omogućuje korisnicima pregled radionica, rezervacije, kupnje proizvoda, upravljanje profilom organizatora i administrativne funkcije (ovisno o ulozi).
- **Poslužitelj:** Backend implementiran u Flasku (Python) obrađuje poslovnu logiku, autentikaciju, autorizaciju, upravljanje radionicama, rezervacijama, proizvodima, izložbama, članstvima, transakcijama i obavijestima.
- **Baza podataka:** PostgreSQL pohranjuje sve podatke o korisnicima, organizatorima, administratorima, radionicama, rezervacijama, proizvodima, transakcijama, recenzijama, obavijestima i članstvima.

Ovaj stil odabran je jer omogućuje jasnu separaciju prezentacijskog sloja (React) od poslovne logike (Flask) i centraliziranu kontrolu podataka (PostgreSQL), što odgovara funkcionalnim i nefunkcionalnim zahtjevima sustava.

Podsustavi

- **Autentikacija i autorizacija:** upravljanje registracijom korisnika, prijavom, ulogama (korisnik, organizator, administrator) i ograničavanjem pristupa funkcionalnostima.
- **Radionice:** kreiranje, uređivanje i brisanje radionica (organizatori), pregled i pretraživanje radionica (korisnici).
- **Rezervacije:** korisnici mogu rezervirati i otkazati radionice; organizatori pregledavaju popis prijavljenih sudionika.
- **Proizvodi i izložbe:** organizatori kreiraju izložbe i povezuju proizvode; korisnici pregledavaju i kupuju proizvode.
- **Transakcije i članarine:** sustav vodi evidenciju uplata za radionice, proizvode i članstva organizatora.
- **Profili organizatora:** upravljanje podacima organizatora i prikaz njihovih radova.
- **Administracija:** administratori odobravaju nove organizatore, upravljaju korisnicima, članarinama i obavijestima.
- **Obavijesti:** slanje obavijesti korisnicima i praćenje pročitanih/nepročitanih poruka.

Preslikavanje na radnu platformu

Deployment se obavlja na **Render** platformi:

- **React frontend:** statički hostan na Renderu.
- **Flask backend:** deployan kao web service dostupan preko HTTPS.
- **PostgreSQL baza:** koristi se PostgreSQL instanca povezana s Flask aplikacijom.

Spremišta podataka

Koristi se **PostgreSQL** relacijska baza podataka koja pohranjuje:

- korisnike (obične, organizatore, administratore)
- radionice i rezervacije
- proizvode i izložbe
- recenzije
- transakcije i narudžbe
- planove članstva i aktivna članstva organizatora
- obavijesti i njihov status za svakog korisnika

Podaci su strukturirani u tablicama s definiranim relacijama (npr. korisnik–rezervacija–radionica).

Mrežni protokoli

Klijent komunicira s backendom putem **HTTP/HTTPS** zahtjeva.
Backend pristupa PostgreSQL bazi koristeći standardni PostgreSQL protokol.

Globalni upravljački tok podataka

Primjer toka za rezervaciju radionice:

1. Korisnik u React sučelju odabire radionicu.
2. Klijent šalje zahtjev prema Flask backendu.
3. Backend provjerava korisnika i dostupna mjesta.
4. Backend upisuje rezervaciju u PostgreSQL.
5. Korisniku se vraća potvrda, a organizatoru se prikazuje novi polaznik.

Slični tokovi postoje za kupovinu proizvoda, upravljanje profilima organizatora, administrativne funkcije i obavijesti.

Sklopovsko-programski zahtjevi

U sustavu se koriste:

- **React** za frontend
- **Flask (Python)** za backend
- **PostgreSQL** za bazu podataka
- **Render** za deployment

Nisu definirani dodatni specifični sklopovski zahtjevi u dostavljenim podacima.

---

2. Obrazloženje odabira arhitekture

Izbor arhitekture temeljen na principima oblikovanja

- **Visoka kohezija:** svaki podsustav ima jasno definirane odgovornosti (npr. rezervacije, proizvodi, članstva).
- **Niska povezanost:** frontend i backend su odvojeni, backend i baza su odvojeni slojevi.
- **Fleksibilnost:** nove funkcionalnosti moguće je dodavati bez utjecaja na postojeće.
- **Sigurnost:** backend centralizira autentikaciju, autorizaciju i obradu osjetljivih podataka.

Ovi principi omogućuju jednostavno održavanje, skalabilnost i nadogradivost sustava.

Razmatrane alternative

- **Monolit (samo Flask):** odbačen jer bi spajao logiku i prezentaciju te otežao razvoj.
- **Mikrousluge:** nepotrebno kompleksno rješenje za projekt ove veličine.
- **Modularni monolit:** moguć, ali klijent–poslužitelj bolje odgovara zahtjevu za mobilnim i web sučeljem.

---

3. Organizacija sustava na visokoj razini

Klijent–poslužitelj

- **Klijent:** React aplikacija omogućuje rad korisnicima, organizatorima i administratorima.

- **Poslužitelj:** Flask backend obrađuje sve poslovne funkcije i pristupa bazi.
- **Komunikacija:** HTTP/HTTPS zahtjevi.

Baza podataka

- **PostgreSQL** čuva sve podatke o korisnicima, radionicama, rezervacijama, proizvodima, izložbama, transakcijama, recenzijama, članstvima i obavijestima.

Datotečni sustav

U dostavljenim podacima nije specificirano korištenje datotečnog sustava, pa se ne opisuje dodatna pohrana.

Grafičko sučelje

- Dizajn izrađen u **Figmi**.
- Implementacija u **Reactu**, prilagođena mobilnim uređajima i jednostavna za korištenje.
- Klijent komunicira s backendom za sve funkcionalnosti sustava.

Organizacija aplikacije

1. Frontend i Backend slojevi

Aplikacija ClayPlay organizirana je u dva jasno odvojena sloja: **frontend** (React) i **backend** (Flask). Ova podjela odgovara klijent–poslužitelj arhitekturi i omogućuje jednostavniju nadogradnju, održavanje i modularnost sustava.

Frontend (React)

Frontend predstavlja prezentacijski sloj koji korisnicima omogućuje rad sa sustavom putem grafičkog sučelja.
Glavne odgovornosti frontend sloja su:

- prikaz korisničkog sučelja temeljenog na dizajnu iz Figme
- upravljanje navigacijom i prikazom ekrana ovisno o ulozi (korisnik, organizator, administrator)
- slanje zahtjeva prema backendu
- prikaz podataka dobivenih iz backend sloja

- obrada jednostavne logike prikaza (validacije unosa, lokalne interakcije, animacije)

Frontend se sastoji od sljedećih komponenti:

- **Komponente sučelja:** kartice radionica, liste proizvoda, modali za rezervacije, forme za uređivanje profila
- **Navigacijski modul:** rute za korisnike, organizatore i administratore
- **Servisi za komunikaciju:** funkcije koje šalju HTTP/HTTPS zahtjeve backendu
- **Stanje aplikacije:** lokalna i globalna pohrana stanja za uloge i podatke korisnika

Backend (Flask)

Backend predstavlja poslovni sloj aplikacije odgovoran za obradu podataka i poslovnu logiku.
Njegove ključne odgovornosti su:

- autentifikacija i autorizacija korisnika
- operacije nad radionicama, rezervacijama, proizvodima i izložbama
- upravljanje profilima organizatora
- upravljanje članstvima i transakcijama
- administrativne funkcije nad korisnicima
- spremanje i dohvat podataka iz PostgreSQL baze

Backend se sastoji od:

- **Modela podataka:** definicije poslovnih entiteta i veza
- **Servisa / poslovne logike:** logika rezervacija, kupnje proizvoda, validacije članstva
- **Kontrolera:** obrada HTTP zahtjeva i pozivanje odgovarajućih servisa
- **Pristupnog sloja bazi:** komunikacija s PostgreSQL-om
- **Sustava obavijesti:** slanje i pohrana obavijesti korisnicima

Frontend i backend međusobno komuniciraju putem **HTTP/HTTPS zahtjeva**, gdje svaki zahtjev frontend-a inicira rad backend funkcionalnosti, a backend vraća strukturirani odgovor.

---

2. MVC arhitektura u backendu

Backend aplikacije koristi **MVC (Model-View-Controller)** princip u pojednostavljenom obliku prilagođen Flask okruženju.

Model (M)

Modeli predstavljaju strukturu podataka i pravila ponašanja entiteta.
U aplikaciji ClayPlay modeli upravljaju:

- korisnicima i njihovim ulogama
- radionicama i rezervacijama
- proizvodima i izložbama
- transakcijama i članstvima
- obavijestima i interakcijama korisnika

Modeli su definirani kao Python klase koje odgovaraju tablicama u PostgreSQL bazi.

View (V)

U Flask kontekstu *View* predstavlja dio koji vraća strukturirani odgovor klijentu.
U ovom sustavu view komponente:

- vraćaju podatke u JSON formatu
- strukturiraju odgovore s porukama o uspjehu ili grešci
- ne sadrže poslovnu logiku

Controller (C)

Kontroleri predstavljaju glavne ulazne točke u backend logiku.
Oni:

- primaju zahtjeve od React frontenda
- pozivaju odgovarajuće modele i servise
- validiraju podatke
- vraćaju odgovore frontend sloju

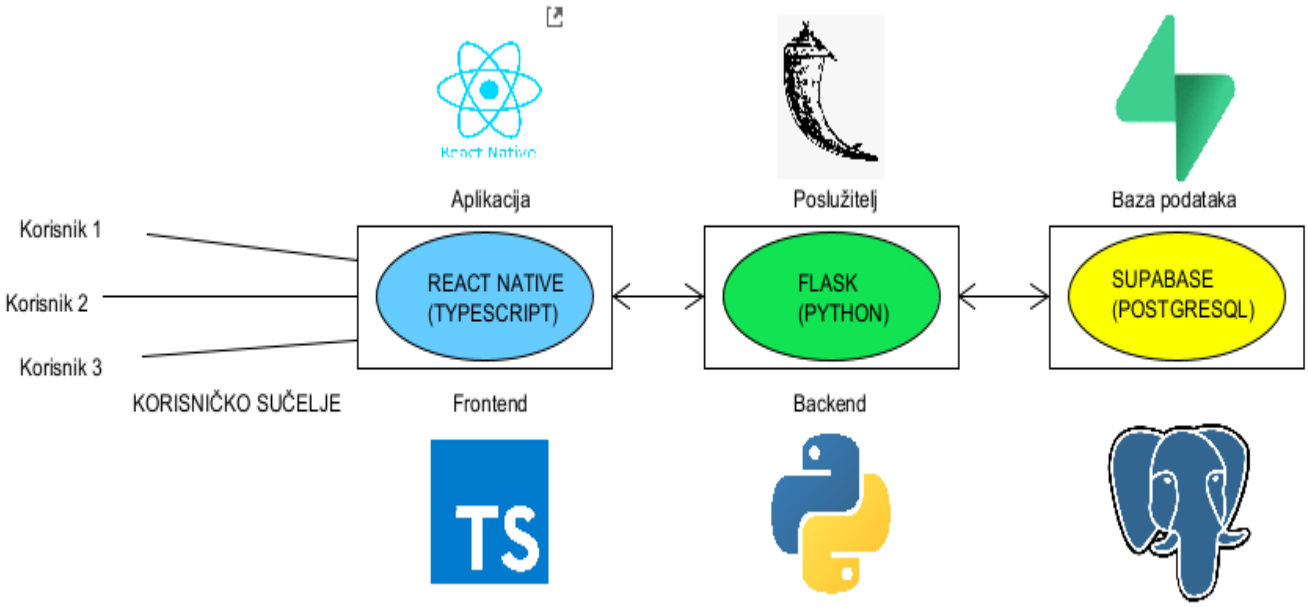
MVC osigurava:

- odvajanje sučelja od poslovne logike
- jednostavnije održavanje
- lakše testiranje pojedinačnih dijelova aplikacije

---

3. Dijagram visoke razine

Sljedeća skica prikazuje strukturu sustava i odnose među glavnim komponentama.
Dijagram je inicijalni i može se kasnije dodatno razraditi.



Baza podataka

admins

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
user_id	INT	Vanjski ključ tablice user
can_change_prices	BOOL	Oznaka smije li admin mijenjati cijene pretplata
can_send_notifications	BOOL	Oznaka smije li admin slati obavjesti
can_approve_users	BOOL	Oznaka smije li admin odobravati nove korisnike

comments

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
user_id	INT	Oznaka korisnika koji je napisao komentar
exhibition_id	INT	Oznaka izložbe na koju se odnosi komentar
content	TEXT	Sadržaj komentara
photo_id	INT	Oznaka slike priložene uz komentar
created_at	TIMESTAMP	Vrijeme objavljivanja komentara

exhibition_products

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
product_id	INT	Vanjski ključ tablice product
exhibition_id	INT	Oznaka za izložbu na kojoj se izlaže product

exhibition_registrations

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
participant_id	INT	Oznaka za polaznika koji je napravio registraciju
exhibition_id	INT	Oznaka za izložbu na koju se registrira polaznik
approved	BOOL	Oznaka je li polaznik odobren od strane organizatora

exhibitions

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
organizer_id	INT	Oznaka za organizatora izložbe
title	TEXT	Naziv izložbe
location	TEXT	Mjesto održavanja izložbe
date_time	TIMESTAMP	Vrijeme održavanja izložbe
description	TEXT	Opis izložbe

exhibition_registrations

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
participant_id	INT	Vanjski ključ tablice participant
exhibition_id	INT	Vanjski ključ tablice exhibition
approved	BOOL	Oznaka je li polaznik odobren od strane organizatora

membership_plans

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
name	TEXT	Naziv za vrstu pretplate
price	NUMERIC	Cijena za vrstu pretplate
duration_months	INT	Broj mjeseci koliko traje pretplata

membership_transactions

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
organizer_id	INT	Oznaka za organizatora koji je izvršio plaćanje
transaction_id	INT	Vanjski ključ iz tablice transactions

notification_subscriptions

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
user_id	INT	Oznaka za korisnika koji se pretplatio na obavjesti
type	TEXT	Vrsta pretplate za obavjesti

notifications

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
body	TEXT	Tijelo poruke iz obavjesti
type	TEXT	Vrsta pretplate za obavjesti
title	TEXT	Naslov obavjesti
subtitle	TEXT	Podnaslov obavjesti
subject	TEXT	Naslov mail obavjesti

orders

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
participant_id	ID	Oznaka za naručitelja
order_time	TIMESTAMP	Vrijeme slanja narudžbe
transaction_id	INT	Vanjski ključ iz tablice transactions

organizer_photos

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
photo_id	INT	Oznaka za sliku koja se nalazi na profilu organizatora
organizer_id	INT	Oznaka za organizatora na čiji se profil odnosi slika

organizers

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
user_id	INT	Vanjski ključ iz tablice users
membership_plan_id	INT	Oznaka za vrstu pretplate koja je zadnja bila aktivna prganizatoru
membership_expiry_date	DATE	Datum do kojeg vrijedi pretplata
profile_name	TEXT	Ime koje će se nalaziti na profilu organizatora
logo_photo_id	INT	Oznaka za sliku koja označava logo organizatora
banner_photo_id	INT	Oznaka za sliku koja označava banner na profilu organizatora
description	TEXT	Opis organizatora
approved_by_admin	BOOL	Oznaka je li organizator odobren od strane administratora

participants

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
user_id	INT	Vanjski ključ iz tablice user

photos

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
url	TEXT	Poveznica na sliku
description	TEXT	Opis slike

product_review

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
participant_id	ID	Oznaka za autora ocjene
product_id	ID	Oznaka za proizvod koji je ocijenjen
rating	INT	Ocjena koju je proizvod dobio
text	TEXT	Tekst uz ocjenu
created_at	TIMESTAMP	Vrijeme davanja ocjene

products

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
name	TEXT	Naziv za proizvod
description	TEXT	Opis proizvoda
price	NUMERIC	Cijena proizvoda
photo_id	INT	Oznaka za sliku proizvoda
quantity_left	INT	Broj koliko jedinki proizvoda je na stanju
exhibition_id	INT	Oznaka za izložbu na kojoj je proizvod bio

products_orders

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
product_id	INT	Oznaka za proizvod koji je u narudžbi
order_id	INT	Oznaka za narudžbu u kojoj je proizvod
quantity	INT	Broj jedinki vrste proizvoda u toj narudžbi

read_notifications

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
user_id	INT	Oznaka za korisnika koji je primio obavjest
notification_id	INT	Oznaka za poslanu obavjest
marked_as_read	INT	Oznaka je li korisnik pročitao obavjest

transactions

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
amount	NUMERIC	Količina novca uplaćena
method	TEXT	Vrsta plaćanja
date_time	TIMESTAMP	Vrijeme kad je obavljeno plaćanje

users

Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
first_name	TEXT	Ime korisnika
last_name	TEXT	Prezime korisnika
username	TEXT	Korisnićko ime korisnika
address	TEXT	Adresa korisnika

Atribut	Tip podatka	Opis varijable
mail	TEXT	Mail adresa korisnika
phone	TEXT	Broj mobitela korisnika
profile_photo_id	TEXT	Oznaka za profilnu sliku korisnika

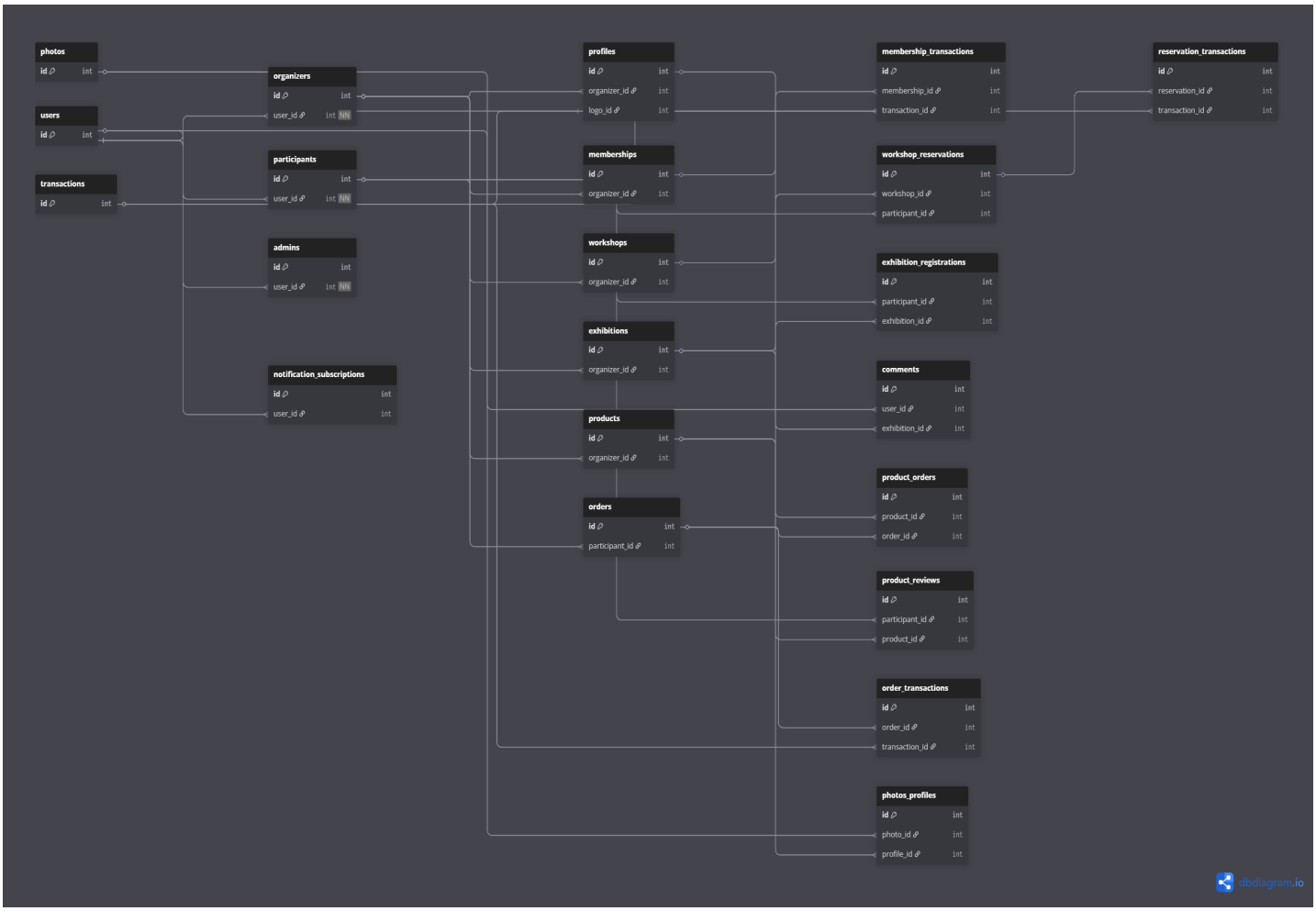
workshop_reservations

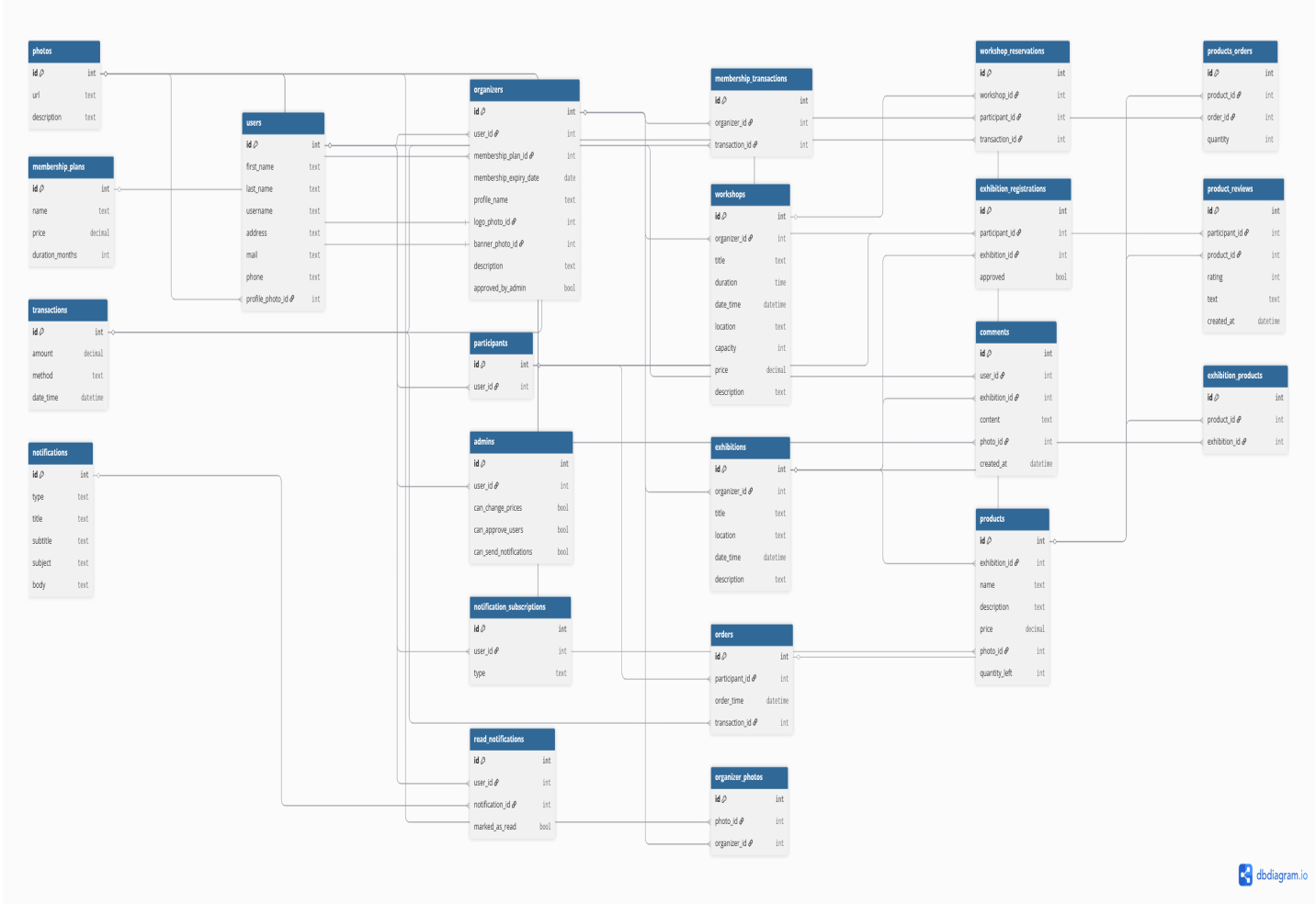
Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
workshop_id	INT	Oznaka za radionicu gdje se rezervira mjesto
transaction_id	INT	Vanjski ključ iz tablice transactions

workshops

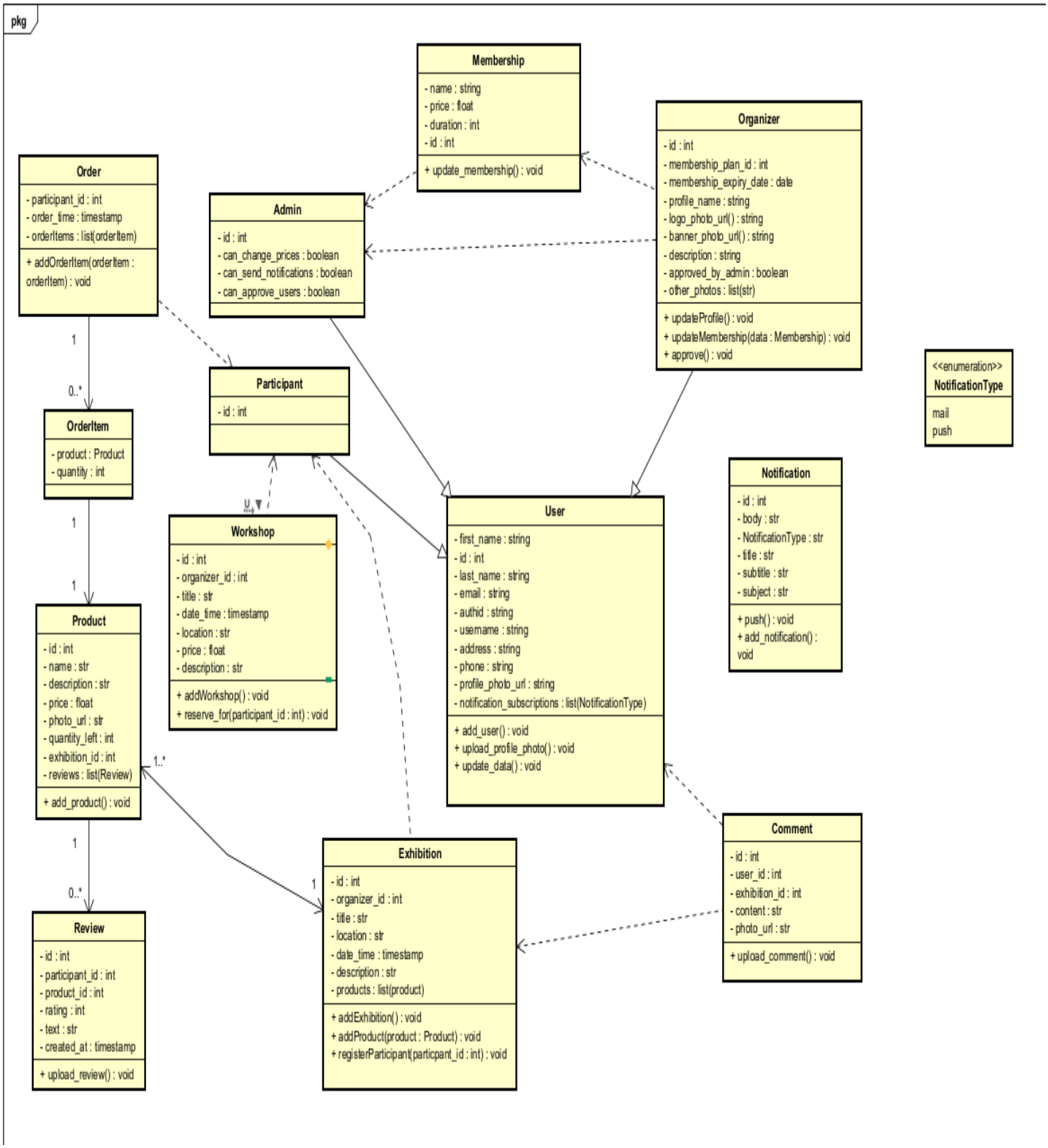
Atribut	Tip podatka	Opis varijable
id	INT	Jedinstveni identifikator
organizer_id	INT	Oznaka za organizatora koji organizira radionicu
title	TEXT	Naziv radionice
duration	TIME	Vrijeme trajanja radionice
date_time	TIMESTAMP	Vrijeme održavanja radionice
location	TEXT	Mjesto održavanja radionice
price	NUMERIC	Cijena za prisustvovanje na radionici
description	TEXT	Opis radionice

Dijagram baze podataka





Dijagram razreda



Slika 4.1 Dijagram klasa arhitekture backenda

Na slici 4.1 prikazana je arhitektura backenda pomoću dijagrama klasa. Većina klasa predstavlja tablicu iz baze podataka koja je opisana u prethodnim odlomcima. Metode na backendu uglavnom služe za komunikaciju s bazom podataka, točnije ubacivanjem novih podataka ili ažuriranjem starih. U daljnjem tekstu, izrazom 'Ažuriranje podataka' misli se na izmjenu podataka u bazi podataka.

Klasa `User` sadrži podatke o pojedinom korisniku. Te podatke nasljeđuju klase `Admin`, `Organizer` i `Participant`. Te iste podatke se može ažurirati u bazi podataka pozivom funkcije `update_data`, a

ako korisnik nije u bazi podataka potrebno je pozvati funkciju `add_user`. `upload_profile_photo` služi za promjenu slike profila korisnika.

`Organizer` ima mogućnost ažuriranja podataka o svom profilu pozivom metode `updateProfile`. Ažuriranje podataka o aktivnoj pretplati organizatora se radi se pozivom funkcije `updateMembership`. Pošto se za tu funkciju koristi klasa `Membership`, klase `Organizer` i `Membership` spojeni su odnosom 'uses'. `Organizer` također koristi `Admin` zbog funkcije `approve` u kojoj je potrebno provjeriti je li korisnik koji potvrđuje organizatora zapravo admin.

U klasi `Membership` moguće je promijeniti cijene i trajanja pretplate funkcijom `update_membership` koja također ovisi o klasi `Admin`.

Komentari se postavljaju u bazu funkcijom `upload_comment` u klasi `Comment`. Ta klasa ovisi o klasama `User` i `Exhibition` zbog toga što sadrži njihove identifikatore. `Exhibition` koristi klasu `Participant` kako bi polaznicima omogućila dolazak na izlože naredbom `registerParticipant`. `addExhibition` i `addProduct` služe za dodavanje izložbi i proizvoda u bazu podataka. Proizvodi koji se dodaju u bazu su povezani s točno tom izložbom. Klasa za izložbe sadrži listu proizvoda koji su izloženi.

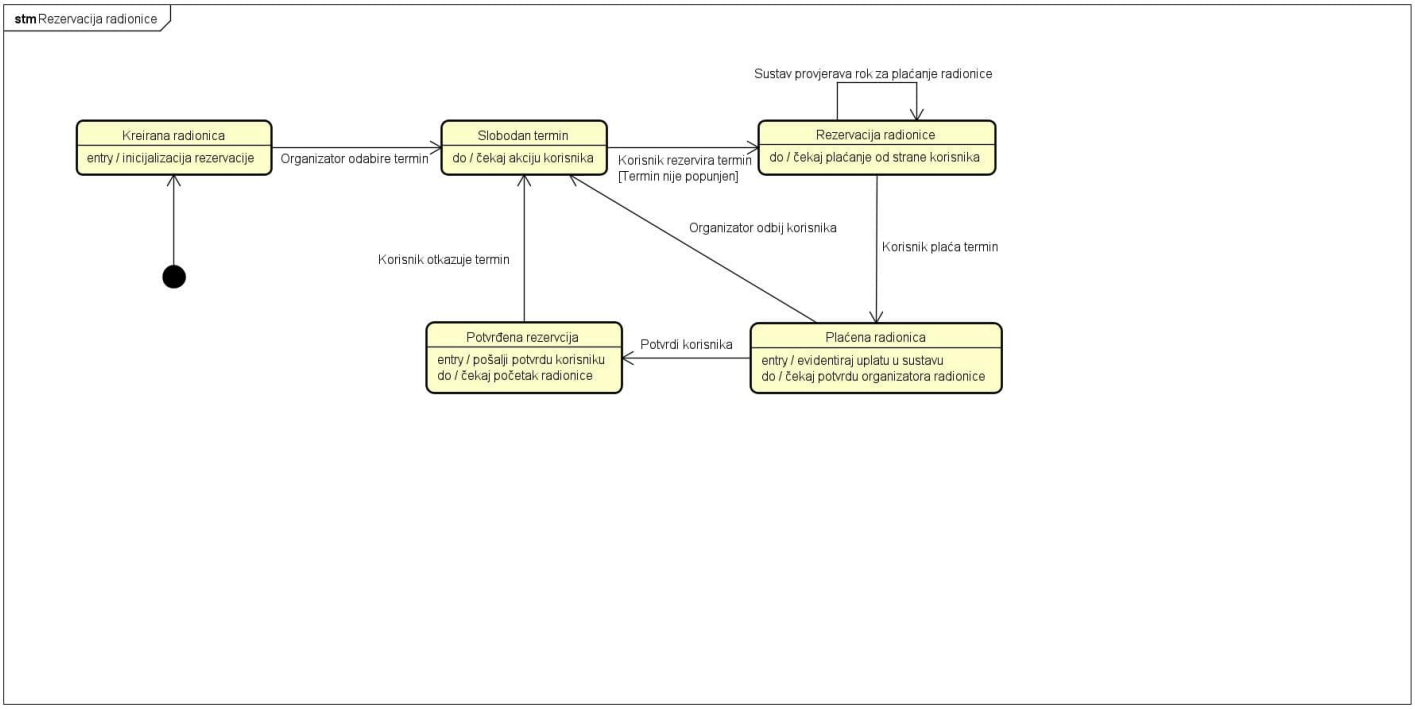
Proizvodi uz atribute što imaju u bazi podataka sadrže još listu ocjena koje su dobili. Ocjene se stavljaju u bazu podataka funkcijom `upload_review` u klasi `Product`. Slično, `add_product` dodaje instance klase `Product` u bazu podataka.

Klasa `Order` sadrži listu klasa `OrderItem`, koja nije u bazi, ali je usko povezana s tablicom orderProducts, jer sadrži informaciju o tome koliko je pojedinog proizvoda u košarici. Instance klase `OrderItem` se dodaju u košaricu funkcijom `addOrderItem`. Zbog informacije o naručitelju klasa `Order` ovisi o klasi `Participant`.

Napomena: Prilikom druge predaje projekta dijagram razreda i opisi moraju odgovarati stvarnom stanju implementacije

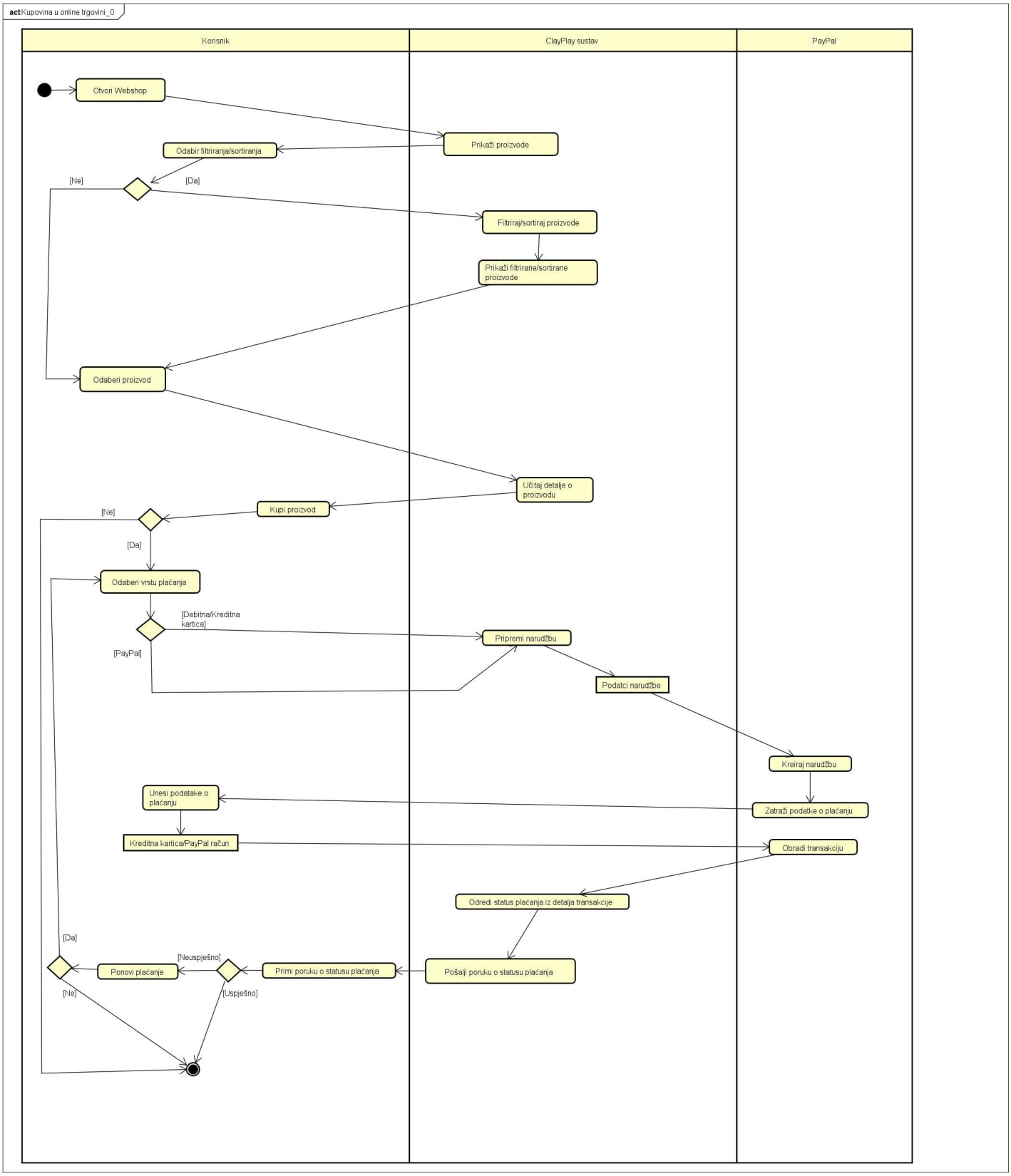
Dinamičko ponašanje aplikacije

UML dijagrami stanja



Dijagram stanja prikazuje životni ciklus objekta **Rezervacija** od trenutka kreiranja do završetka rada. Nakon ulaska u stanje **Kreirana radionica**, organizator odabire termin, čime sustav prelazi u stanje **Slobodan termin** i zaključava termin. U tom stanju korisnik može rezervirati termin, ako termin nije popunjen. Korisnik u stanju rezervirane radionice može ju i platiti te prelazi u stanje **Plaćena radionica**. Plaćena radionica prelazi u stanje **Potvrđena rezervacija** kada organizator potvrdi korisnika, gdje čeka početak radionice. U slučaju otkazivanja radionice od strane korisnika ili odbijanje korisnika od strane organizator sustav prelazi u stanje slobodan termin.

UML dijagrami aktivnosti



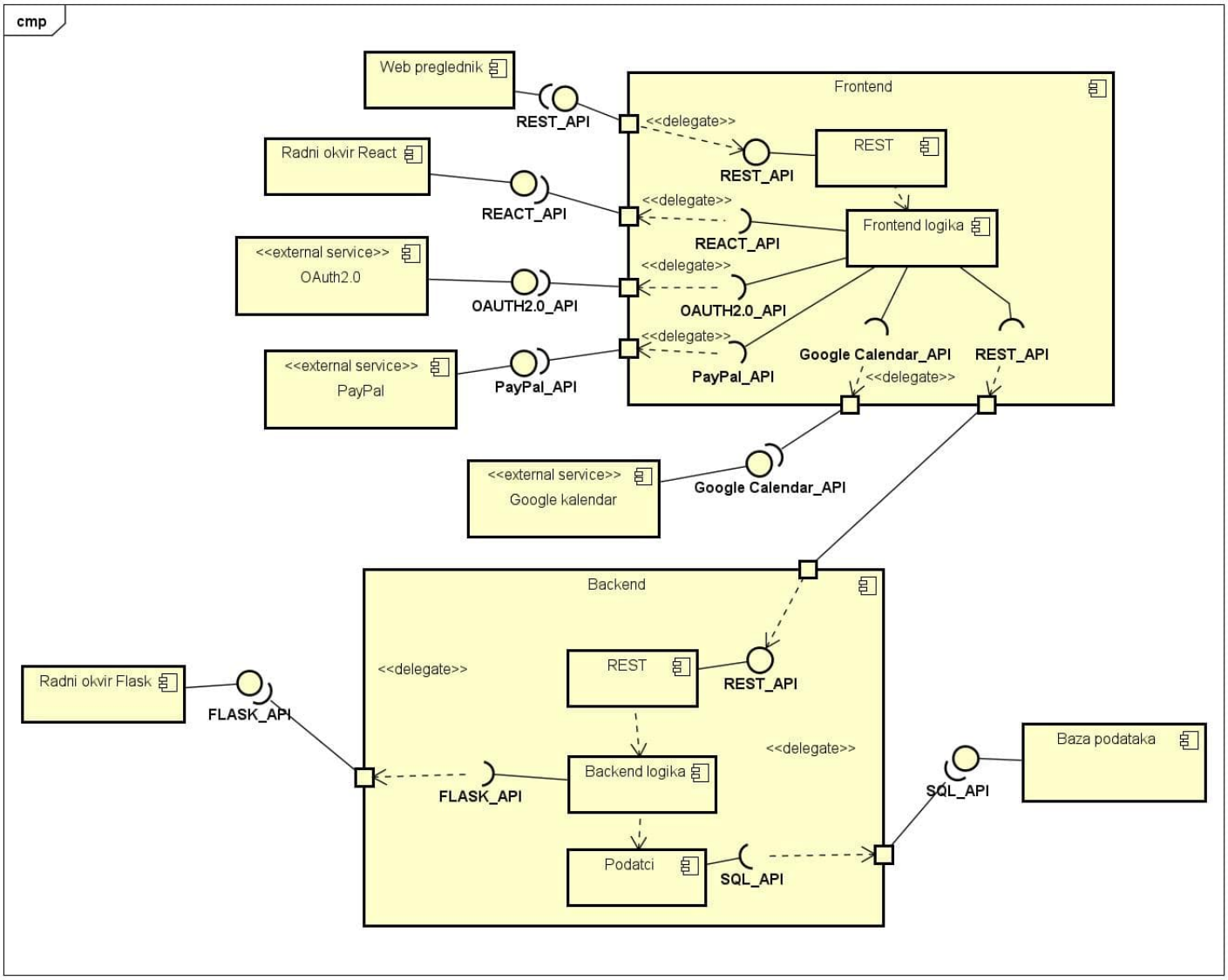
Kupovina u aplikaciji ClayPlay započinje otvaranjem webshopa, nakon čega aplikacija ih prikazuje korisniku. Korisnik može opcionalno primijeniti filtre i/ili sortiranje kako bi prilagodio prikaz proizvoda. Nakon što odabere proizvod, aplikacija prikazuje njegove detalje te korisnik odlučuje želi li nastaviti s kupnjom.

Ako korisnik odluči kupiti proizvod, započinje proces plaćanja. Korisnik najprije odabire način plaćanja (PayPal ili debitna/kreditna kartica), zatim aplikacija priprema narudžbu i nakon toga šalje platnom servisu (PayPal) detalje narudžbe. Platni servis potom kreira narudžbu te traži od korisnika unos

podataka o plaćanju. Korisnik zatim unosi potrebne podatke za izvršenje transakcije, nakon čega platni servis na temelju unesenih podataka obrađuje plaćanje.

Nakon obrade transakcije platni servis vraća aplikaciji detalje transakcije, na temelju kojih sustav određuje status plaćanja. Status plaćanja se potom šalje korisniku. Ako je plaćanje bilo neuspješno korisnik ima priliku izabrati želi li ponovno pokušati kupiti proizvod ili odustati od kupovine. Ako je plaćanje bilo uspješno kupovina je završena.

Dijagram komponenata

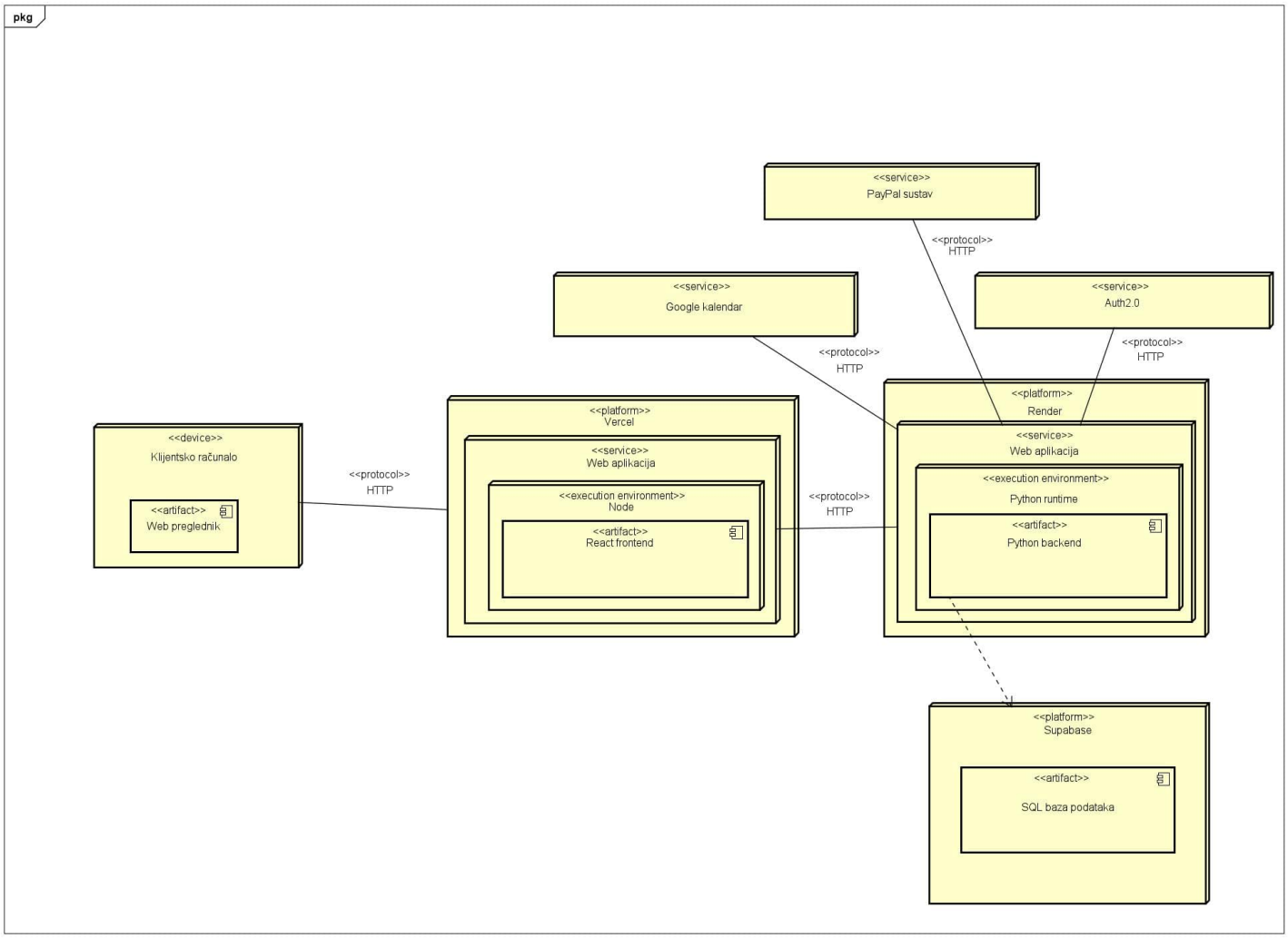


Na slici je prikazan UML dijagram komponenti sustava ClayPlay. Sustav je podijeljen u dvije glavne komponente: Frontend i Backend. Korisnik pristupa aplikaciji putem web preglednika te interakcija započinje u Frontendu.

Frontend je izgrađen korištenjem React radnog okvira te s ostatkom sustava komunicira putem REST_API sučelja. Putem istog sučelja Frontend ostvaruje komunikaciju s Backendom. Osim toga, Frontend koristi vanjske servise za autentifikaciju, plaćanje i upravljanje terminima, odnosno OAuth 2.0 za upravljanje autentifikacijom korisnika, PayPal API za obradu transakcija te Google Calendar API za prikaz i sinkronizaciju kalendarskih događaja.

Backend dio sustava implementiran je korištenjem Flask radnog okvira. Unutar Backenda nalaze se komponente za obradu poslovne logike te upravljanje podacima. Backend pristupa relacijskoj bazi podataka putem SQL_API sučelja, pri čemu se podaci pohranjuju u sustav.

Dijagram razmještaja



Na slici je prikazan dijagram razmještaja sustava ClayPlay koji opisuje fizički raspored aplikacijskih komponenti na različitim platformama te način njihove međusobne komunikacije. Korisnik pristupa sustavu putem web preglednika na klijentskom računalu, a frontend aplikacija je smještena na platformi Vercel, gdje se izvršava u Node.js okruženju kao React aplikacija. Frontend ostvaruje komunikaciju s backendom putem HTTP protokola.

Backend sustav nalazi se na platformi Render, gdje se izvršava u Python runtime okruženju i zadužen je za obradu poslovne logike te komunikaciju s bazom podataka i vanjskim servisima. Za autentifikaciju korisnika koristi se vanjski OAuth 2.0 sustav, dok je za obradu plaćanja integriran PayPal, a za upravljanje i sinkronizaciju termina koristi se Google Calendar.

Podatkovni sloj sustava realiziran je pomoću platforme Supabase, pri čemu se koristi SQL baza podataka za pohranu, dohvat i upravljanje podacima potrebnim za rad sustava.

Ispitivanje komponenti

Grupa 1: AuthController (autentikacija)

Opis funkcionalnosti

Upravljanje prijavom, registracijom i sesijama korisnika.

Postupak ispitivanja

- Svaki test koristi mockirane metode Supabase-a kako bi izolirao `AuthController` logiku.
- `beforeEach` resetira sve mockove.
- Poziv funkcija testira **resolves** ili **rejects** prema očekivanom rezultatu.

Ispitni slučajevi

#	Scenarij	Ulaz	Očekivano	Dobiveno	Status
1	Uspješan login	Email: `a@b.com`, Lozinka: `pass`	Funkcija resolves, vraća `undefined`	Funkcija resolves	Prolaz
2	Login s pogrešnom lozinkom	Email: `a@b.com`, Lozinka: `bad`	Funkcija rejecta, error `"Invalid"`	Error `"Invalid"`	Prolaz
3	Signup s postojećim email-om	Email: `x@y.com`, Lozinka: `pw`, Ime: `Full`	Funkcija rejecta, error `"Email exists"`	Error `"Email exists"`	Prolaz
4	Dohvat sesije kad nema sesije	Nema ulaza	Funkcija rejecta, error `"No active session."`	Error `"No active session."`	Prolaz
5	Poziv nepostojeće metode u AuthController	Nema ulaza	Funkcija baca grešku	Greška bačena	Prolaz
6	Login s praznim email/lozinkom	Email: `''`, Lozinka: `''`	Funkcija rejecta, error `"Email and password required"`	Error `"Email and password required"`	Prolaz

Objašnjenje ispitnih slučajeva

- **Slučaj 1:**
  Simulira se uspješan odgovor Supabase metode `signInWithPassword` bez greške (`error: null`).
  Test provjerava da metoda `loginWithEmailAndPassword` u tom slučaju ne baca grešku i da se Promise uspješno resolvea.

- **Slučaj 2:**
  Simulira se neuspješna prijava gdje Supabase vraća grešku s porukom `"Invalid"` .
  Očekuje se da `loginWithEmailAndPassword` prepozna grešku i rejecta Promise s odgovarajućom porukom.

- **Slučaj 3:**
  Simulira se pokušaj registracije korisnika s već postojećom email adresom.
  Metoda `signUpWithEmailAndPassword` mora proslijediti Supabase grešku i rejectati Promise s porukom `"Email exists"` .

- **Slučaj 4:**
  Simulira se situacija u kojoj Supabase vraća podatak da ne postoji aktivna korisnička sesija (`session: null`).
  Metoda `getCurrentSession` mora prepoznati da korisnik nije prijavljen i rejectati Promise s porukom `"No active session."` .

- **Slučaj 5:**
  Testira se ponašanje sustava pri pozivu metode koja ne postoji unutar `AuthController` objekta.
  Očekuje se da JavaScript baci grešku (`throws`), čime se potvrđuje ispravno rukovanje nevažećim pozivima funkcija.

- **Slučaj 6:**
  Simulira se rubni slučaj u kojem korisnik pokušava prijavu bez unosa emaila i lozinke.
  Metoda `loginWithEmailAndPassword` mora validirati ulazne podatke i rejectati Promise s jasnom porukom o grešci.

Kod

```
import { vi, describe, it, expect, beforeEach } from 'vitest';
import AuthController from '@/controllers/authController';

vi.mock('@/config/supabase', () => {
  return {
    supabase: {
      auth: {
```

```
        signInWithPassword: vi.fn(),
        signUp: vi.fn(),
        getSession: vi.fn(),
        signOut: vi.fn(),
        signInWithOAuth: vi.fn(),
      },
    },
  };
});

import { supabase } from '@/config/supabase';

describe('AuthController', () => {
  beforeEach(() => {
    vi.resetAllMocks();
  });

  it('resolves on successful login', async () => {
    (supabase.auth.signInWithPassword as any).mockResolvedValue({ error:
    await expect(AuthController.loginWithEmailAndPassword('a@b.com', 'pas
  });

  it('rejects on login error (wrong password)', async () => {
    (supabase.auth.signInWithPassword as any).mockResolvedValue({ error:
    await expect(AuthController.loginWithEmailAndPassword('a@b.com', 'bac
  });

  it('rejects on signup error (email exists)', async () => {
    (supabase.auth.signUp as any).mockResolvedValue({ error: { message: '
    await expect(AuthController.signUpWithEmailAndPassword('x@y.com', 'pw
  });

  it('rejects when getCurrentSession has no session', async () => {
    (supabase.auth.getSession as any).mockResolvedValue({ data: { sessior
    await expect(AuthController.getCurrentSession()).rejects.toThrow('No
  });

  it('throws when calling a non-existing AuthController function', () =>
    const callNonExisting = () => {(AuthController as any).nonExistingMet
    expect(callNonExisting).toThrow();
  });

  it('rejects login when email and password are empty (edge case)', async
    (supabase.auth.signInWithPassword as any).mockResolvedValue({error: {
```

```
    await expect(AuthController.loginWithEmailAndPassword('', '')).reject
  });

});
```

Grupa 2: fetchUtils (HTTP helper funkcije)

Opis funkcionalnosti

Funkcijski wrapperi `fetchGet` i `fetchPost` služe za standardizirani dohvat podataka preko `fetch` API-ja, uključujući obradu grešaka HTTP odgovora i parsiranja JSON-a.

Postupak ispitivanja

- Globalna `fetch` funkcija je mockirana pomoću `vi.fn()` kako bi se testovi izvodili bez stvarnih HTTP zahtjeva.
- `beforeEach` resetira sve mockove kako bi svaki test bio izoliran.
- Testovi provjeravaju ponašanje funkcija u slučaju:
  - neispravnog JSON odgovora
  - HTTP greške (`response.ok === false`)
- Očekivano ponašanje je da funkcije **rejectaju Promise** s odgovarajućom porukom greške.

Ispitni slučajevi

#	Scenarij	Ulaz	Očekivano	Dobiveno	Status
1	Neispravan JSON odgovor kod GET zahtjeva	URL: `'/x'`	Funkcija rejecta, error `"Invalid JSON response."`	Error `"Invalid JSON response."`	Prolaz
2	HTTP greška kod POST zahtjeva s porukom	URL: `'/x'` , body: `{}`	Funkcija rejecta, error `"Bad"`	Error `"Bad"`	Prolaz

Objašnjenje ispitnih slučajeva

- **Slučaj 1:**
  Simulira se uspješan HTTP odgovor (`ok: true`), ali `json()` baca grešku. Time se testira obrada situacije u kojoj backend vraća neispravan JSON.
  `fetchGet` mora prepoznati ovu grešku i rejectati s porukom `"Invalid JSON response."` .

- **Slučaj 2:**
  Simulira se HTTP greška (`ok: false`) pri POST zahtjevu, gdje backend vraća JSON s `message`

poljem.

`fetchPost` mora pročitati poruku greške iz odgovora i rejectati Promise s tom porukom.

Kod

```
import { describe, it, expect, vi, beforeEach } from 'vitest';
import { fetchGet, fetchPost } from '@/utils/fetchUtils';

globalThis.fetch = vi.fn();

describe('fetchUtils', () => {
  beforeEach(() => {
    vi.resetAllMocks();
  });

  it('fetchGet rejects on invalid JSON', async () => {
    (fetch as any).mockResolvedValue({
      ok: true,
      json: () => { throw new Error('bad json'); }
    });

    await expect(fetchGet('/x')).rejects.toThrow('Invalid JSON response.'
  });

  it('fetchPost rejects when response.ok is false and contains message',
    (fetch as any).mockResolvedValue({
      ok: false,
      json: async () => ({ message: 'Bad' })
    });

    await expect(fetchPost('/x', {})).rejects.toThrow('Bad');
  });
});
```

Stranica ProfileApproval (Admin – odobravanje profila)

Opis funkcionalnosti

Stranica `ProfileApproval` omogućava administratorima pregled novih profila koji čekaju odobrenje i njihovo odobravanje. Odobreni profili se uklanjaju s liste odmah nakon uspješnog zahtjeva prema backendu.

Postupak ispitivanja

- Mockirane su funkcije `fetchGet` i `fetchPost` kako bi se testovi izvodili bez stvarnih HTTP zahtjeva.
- Mockan je `supabase.auth.getSession` kako bi komponenta mogla dohvatiti sesiju.
- `beforeEach` resetira sve mockove radi izolacije testova.
- Test provjerava da:
  - Komponenta ispravno renderira dohvaćene profile.
  - Klik na gumb "Odobri Profil" šalje POST zahtjev i uklanja profil iz DOM-a.

Ispitni slučajevi

#	Scenarij	Ulaz	Očekivano	Dobiveno	Status
1	Render profila i odobravanje	Profil: `{ id: 1, profile name: 'Org', ... }`	Profil se prikazuje, nakon klika na "Odobri Profil" profil nestaje iz DOM-a	Profil se prikazuje, nakon klika nestaje	Prolaz

Objašnjenje ispitnog slučaja

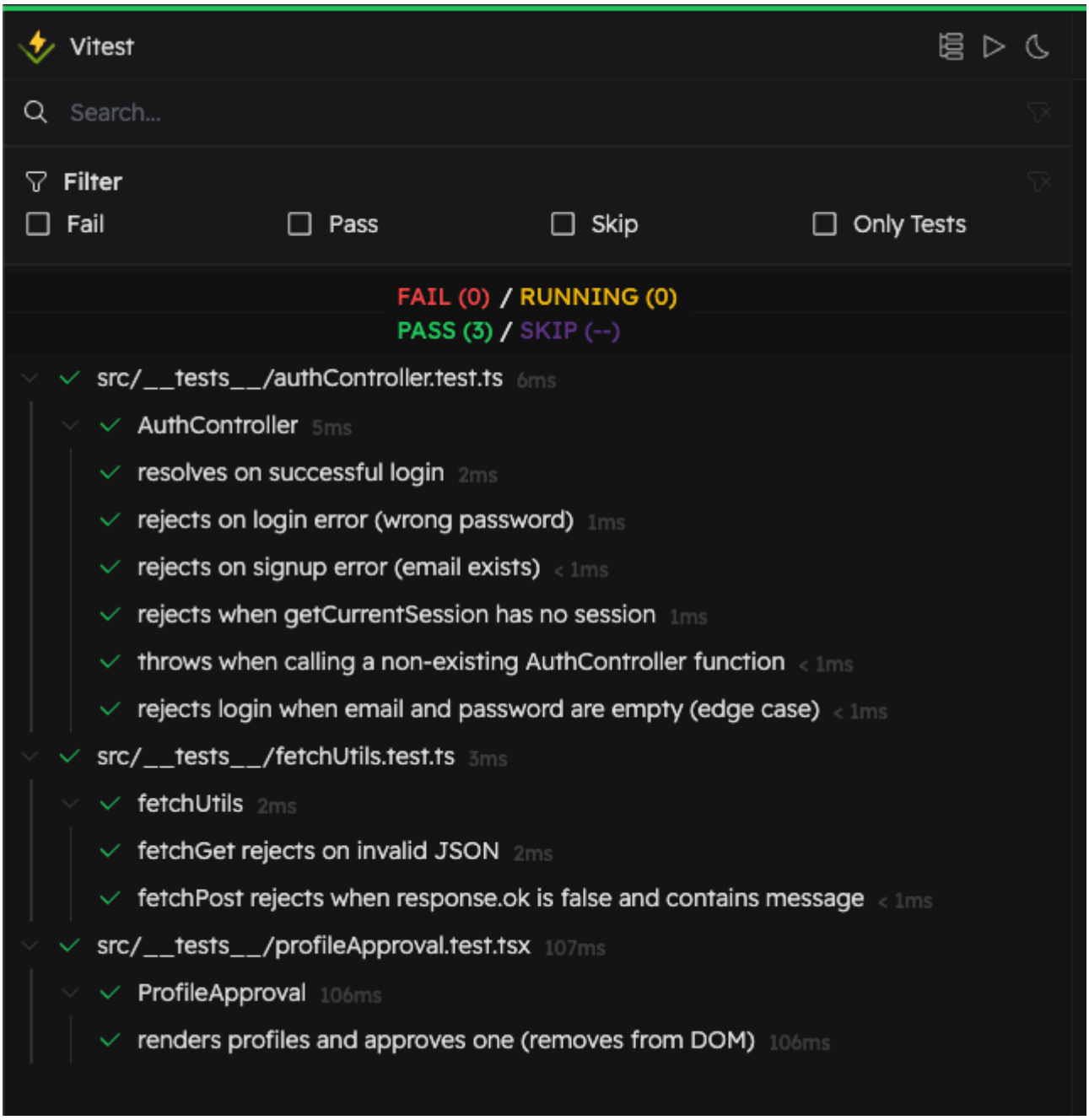
- **Render i odobravanje profila:**
  Mockirani `fetchGet` vraća listu profila. Komponenta ih prikazuje u DOM-u.
  Nakon što korisnik klikne na gumb "Odobri Profil", `fetchPost` šalje zahtjev za odobrenje, a profil se uklanja iz DOM-a (`queryByText` vraća `null`).
  Ovim testom se provjerava **integracija UI-a s mockiranim API-em** i reakcija komponenta na akciju korisnika.

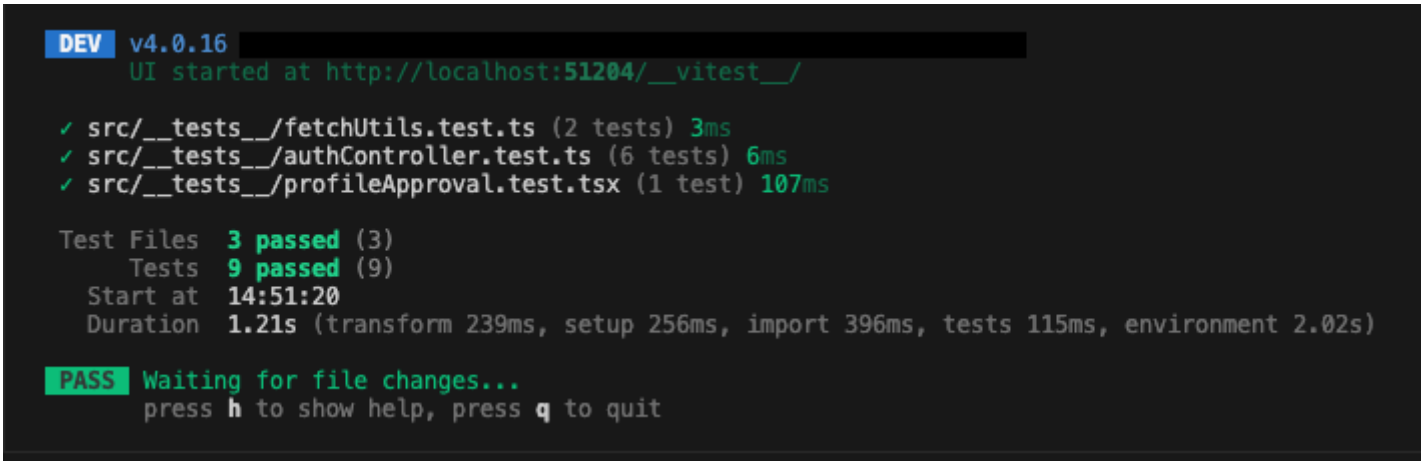
Kod

```
import { describe, it, expect, vi, beforeEach } from 'vitest';
import { render, screen, waitFor, fireEvent } from '@testing-library/reac
import { MemoryRouter } from 'react-router';
import ProfileApproval from '@/pages/admin/ProfileApproval';

vi.mock('@/config/supabase', () => ({
  supabase: {
    auth: {
      getSession: vi.fn(() => ({
        data: { session: { access_token: 'token' } }
      }))
    }
  }
}));

vi.mock('@/utils/fetchUtils', () => ({
```

```
  fetchGet: vi.fn(),
  fetchPost: vi.fn(),
}));

import { fetchGet, fetchPost } from '@/utils/fetchUtils';

describe('ProfileApproval', () => {
  beforeEach(() => {
    vi.resetAllMocks();
  });

  it('renders profiles and approves one (removes from DOM)', async () =>
    (fetchGet as any).mockResolvedValue({
      success: true,
      profiles: [
        {
          id: 1,
          profile_name: 'Org',
          logo_photo_url: '',
          banner_photo_url: '',
          description: 'D',
          address: '',
          mail: 'a@b',
          phone: ''
        }
      ]
    });
    (fetchPost as any).mockResolvedValue({ success: true });

    render(
      <MemoryRouter>
        <ProfileApproval />
      </MemoryRouter>
    );

    await waitFor(() => expect(screen.getByText('Org')).toBeInTheDocument
    const approveBtn = screen.getByRole('button', { name: /Odobri Profil/
    fireEvent.click(approveBtn);

    await waitFor(() => expect(screen.queryByText('Org')).not.toBeInTheDc
  });
});
```

VITEST REPORT - Unit testovi





Ispitivanje sustava

Registracija i prijava korisnika

1. Ulazi

Konkretni podaci koji se unose u sustav

- Ime: `registerXX` (dinamički generirano, XX = slučajni broj 10–99)
- Prezime: `registerXX`
- Email: `registerXX@test.com`
- Lozinka: `TestPassword123`

Simulacija korisničkih akcija

- Navigacija na stranicu za registraciju
- Unos podataka u registracijsku formu
- Klik na gumb **Register**
- Odabir korisničke uloge
- Odjava korisnika
- Navigacija na stranicu za prijavu
- Unos podataka za prijavu
- Klik na gumb **Login**

2. Koraci ispitivanja

1. Regisrirati novog korisnika i odabrati ulogu **Polaznik**
2. Provjeriti da je korisnik prijavljen (vidljiv **Logout** gumb)
3. Kliknuti na gumb **Logout** i pričekati preusmjerenje
4. Prijaviti se s istim korisnikom
5. Provjeriti da je korisnik uspješno prijavljen (vidljiv **Logout** gumb i puno ime korisnika)

3. Očekivani izlaz

- Registracija korisnika je uspješna.
- Korisnik je preusmjeren na odabir uloge nakon registracije.
- Nakon odabira uloge korisnik je automatski prijavljen.
- Nakon prijave korisnik vidi **Logout** gumb.
- Prikazano je ime i prezime prijavljenog korisnika.
- Nakon odjave i ponovne prijave, sustav ispravno autentificira korisnika.

4. Dobiveni izlaz

- Playwright E2E test prolazi bez grešaka.

Kod

```
import { test, expect } from '@playwright/test';

test('registracija i prijava', async ({ page }) => {
  const randomSuffix = Math.floor(10 + Math.random() * 90);
  const firstName = `register${randomSuffix}`;
  const lastName = `register${randomSuffix}`;
  const email = `register${randomSuffix}@test.com`;
  const password = 'TestPassword123';

  await page.goto('http://localhost:5173/auth/register');
  await expect(page).toHaveURL(/\/auth\/register/);

  const firstNameInput = page.locator('input[placeholder="First Name"]');
  const lastNameInput = page.locator('input[placeholder="Last Name"]');
  const emailInput = page.locator('input[placeholder="Email"]');
  const passwordInput = page.locator('input[placeholder="Password"]');
  const repeatPasswordInput = page.locator('input[placeholder="Repeat pas
  const registerButton = page.getByRole('button', { name: 'Register' });

  await firstNameInput.waitFor({ state: 'visible' });
  await firstNameInput.fill(firstName);

  await lastNameInput.waitFor({ state: 'visible' });
  await lastNameInput.fill(lastName);

  await emailInput.waitFor({ state: 'visible' });
  await emailInput.fill(email);

  await passwordInput.waitFor({ state: 'visible' });
  await passwordInput.fill(password);

  await repeatPasswordInput.waitFor({ state: 'visible' });
  await repeatPasswordInput.fill(password);

  await registerButton.click();

  await expect(page).toHaveURL(/\/rolechoose/);

  const polaznikCard = page.getByRole('heading', { name: 'Polaznik' });
  await polaznikCard.waitFor({ state: 'visible' });
  await polaznikCard.click();

  const continueButton = page.getByRole('button', { name: 'Nastavi' });
  await continueButton.waitFor({ state: 'visible' });
```

```
  await continueButton.click();

  await expect(page).toHaveURL(/\//);

  const logoutButton = page.getByRole('button', { name: 'Logout' });
  await logoutButton.waitFor({ state: 'visible', timeout: 10000 });

  await logoutButton.click();

  await page.waitForURL(/localhost:5173\/auth/, { timeout: 10000 });

  await page.goto('http://localhost:5173/auth/login');
  await expect(page).toHaveURL(/\/auth\/login/);

  const loginEmailInput = page.locator('input[placeholder="Email"]');
  const loginPasswordInput = page.locator('input[placeholder="Password"]'
  const loginButton = page.getByRole('button', { name: 'Login' });

  await loginEmailInput.waitFor({ state: 'visible' });
  await loginEmailInput.fill(email);

  await loginPasswordInput.waitFor({ state: 'visible' });
  await loginPasswordInput.fill(password);

  await loginButton.click();

  await expect(logoutButton).toBeVisible();
  await expect(page.getByText(`${firstName} ${lastName}`)).toBeVisible();
});
```

Odabir uloge prilikom registracije

1. Ulazi

Konkretni podaci koji se unose u sustav

- Ime: `roleXX` (dinamički generirano, XX = slučajni broj 10–99)
- Prezime: `roleXX`
- Email: `roleXX@test.com`
- Lozinka: `TestPassword123`

Simulacija korisničkih akcija

- Navigacija na stranicu za registraciju
- Unos podataka u registracijsku formu
- Klik na gumb **Register**
- Odabir uloge **Organizator**
- Klik na gumb **Nastavi**
- Otvaranje korisničkog izbornika

2. Koraci ispitivanja

1. Registrirati novog korisnika
2. Odabrati ulogu **Organizator**
3. Kliknuti gumb **Nastavi**
4. Provjeriti da je korisnik preusmjeren na početnu stranicu
5. Otvoriti korisnički izbornik i provjeriti prikazuje li se kartica **Članstvo**

3. Očekivani izlaz

- Registracija korisnika je uspješna.
- Korisnik je preusmjeren na stranicu za odabir uloge nakon registracije.
- Nakon odabira uloge **Organizator**, korisnik je uspješno prijavljen.
- Korisnik je preusmjeren na početnu stranicu aplikacije.
- U korisničkom izborniku dostupna je opcija **Članstvo**, što potvrđuje da korisnik ima ulogu organizatora.

4. Dobiveni izlaz

- Playwright E2E test prolazi bez grešaka.

Kod

```
import { test, expect } from '@playwright/test';

test('odabir uloge', async ({ page }) => {
  const randomSuffix = Math.floor(10 + Math.random() * 90);
  const firstName = `role${randomSuffix}`;
  const lastName = `role${randomSuffix}`;
  const email = `role${randomSuffix}@test.com`;
  const password = 'TestPassword123';

  await page.goto('http://localhost:5173/auth/register');
  await expect(page).toHaveURL(/\/auth\/register/);

  await page.locator('input[placeholder="First Name"]').fill(firstName);
  await page.locator('input[placeholder="Last Name"]').fill(lastName);
  await page.locator('input[placeholder="Email"]').fill(email);
```

```
  await page.locator('input[placeholder="Password"]').fill(password);
  await page.locator('input[placeholder="Repeat password"]').fill(passwor

  const registerButton = page.getByRole('button', { name: 'Register' });
  await expect(registerButton).toBeEnabled();
  await registerButton.click();

  await expect(page).toHaveURL(/\/rolechoose/);

  const organizatorCard = page.getByRole('heading', { name: 'Organizator'
  await organizatorCard.waitFor({ state: 'visible' });
  await organizatorCard.click();

  const continueButton = page.getByRole('button', { name: 'Nastavi' });
  await continueButton.waitFor({ state: 'visible' });
  await expect(continueButton).toBeEnabled();
  await continueButton.click();

  await expect(page).toHaveURL('http://localhost:5173/');

  const menuButton = page.getByRole('button').first();
  await menuButton.click();

  const membershipButton = page.getByRole('button', { name: 'Članstvo' })
  await expect(membershipButton).toBeVisible();
});
```

Pregled radionica i edge slučaj popunjenog kapaciteta

1. Ulazi

Konkretni podaci koji se unose u sustav

- Ime: `workshopXX` (dinamički generirano, XX = slučajni broj 10–99)
- Prezime: `workshopXX`
- Email: `workshopXX@test.com`
- Lozinka: `TestPassword123`

Simulacija korisničkih akcija

- Registracija novog korisnika
- Odabir uloge **Polaznik**

- Navigacija na stranicu **Radionice**
- Odabir radionice s punim kapacitetom

2. Koraci ispitivanja

1. Registrirati novog korisnika i odabrati ulogu **Polaznik**
2. Navigirati na stranicu **Radionice**
3. Odabrati radionicu **vrtne figure od gline** (popunjena radionica)
4. Provjeriti prikaz kapaciteta (`0 / 0 slobodnih mjesta`)
5. Provjeriti da je gumb **Popunjeno** onemogućen

3. Očekivani izlaz

- Registracija i odabir uloge su uspješni
- Korisnik vidi popis radionica
- Radionica s punim kapacitetom prikazuje ispravan tekst kapaciteta
- Gumb za rezervaciju je onemogućen i označen s **Popunjeno**

4. Dobiveni izlaz

- Playwright E2E test prolazi bez grešaka

Kod

```
import { test, expect } from '@playwright/test';

test('popunjena radionica', async ({ page }) => {
  const randomSuffix = Math.floor(10 + Math.random() * 90);
  const firstName = `workshop${randomSuffix}`;
  const lastName = `workshop${randomSuffix}`;
  const email = `workshop${randomSuffix}@test.com`;
  const password = 'TestPassword123';

  await page.goto('http://localhost:5173/auth/register');
  await expect(page).toHaveURL(/\/auth\/register/);

  await page.locator('input[placeholder="First Name"]').fill(firstName);
  await page.locator('input[placeholder="Last Name"]').fill(lastName);
  await page.locator('input[placeholder="Email"]').fill(email);
  await page.locator('input[placeholder="Password"]').fill(password);
  await page.locator('input[placeholder="Repeat password"]').fill(passwor

  const registerButton = page.getByRole('button', { name: 'Register' });
  await expect(registerButton).toBeEnabled();
  await registerButton.click();
```

```
  await expect(page).toHaveURL(/\/rolechoose/);

  const polaznikCard = page.getByRole('heading', { name: 'Polaznik' });
  await polaznikCard.waitFor({ state: 'visible' });
  await polaznikCard.click();

  const continueButton = page.getByRole('button', { name: 'Nastavi' });
  await expect(continueButton).toBeEnabled();
  await continueButton.click();

  await expect(page).toHaveURL('http://localhost:5173/');

  const menuButton = page.getByRole('button').first();
  await menuButton.click();

  const radioniceButton = page.getByRole('button', { name: 'Radionice' })
  await radioniceButton.click();

  await expect(page).toHaveURL(/\/workshops/);

  const fullWorkshopLink = page.getByRole('link', { name: /vrtne figure c
  await fullWorkshopLink.click();

  const capacityInfo = page.getByText('0 / 0 slobodnih mjesta');
  await expect(capacityInfo).toBeVisible();

  const popunjenoButton = page.getByRole('button', { name: 'Popunjeno' })
  await expect(popunjenoButton).toBeDisabled();
});
```

Nepostojeća stranica

1. Ulazi

- Navigacija na nepostojeći URL (`/ovo-ne-postoji-123`)
- Korištenje browser back funkcionalnosti (`<-`)

2. Koraci ispitivanja

1. Otvoriti početnu stranicu aplikacije
2. Navigirati na nepostojeći URL

3. Provjeriti da se prikazuje poruka **404 - Page Not Found**
4. Vratiti se pomoću browser back strelice (`<-`)
5. Provjeriti da se korisnik vraća na početnu stranicu (`/auth`)

3. Očekivani izlaz

- Nepostojeća stranica prikazuje točan 404 error
- Browser back (`<-`) ispravno vraća korisnika na prethodnu stranicu

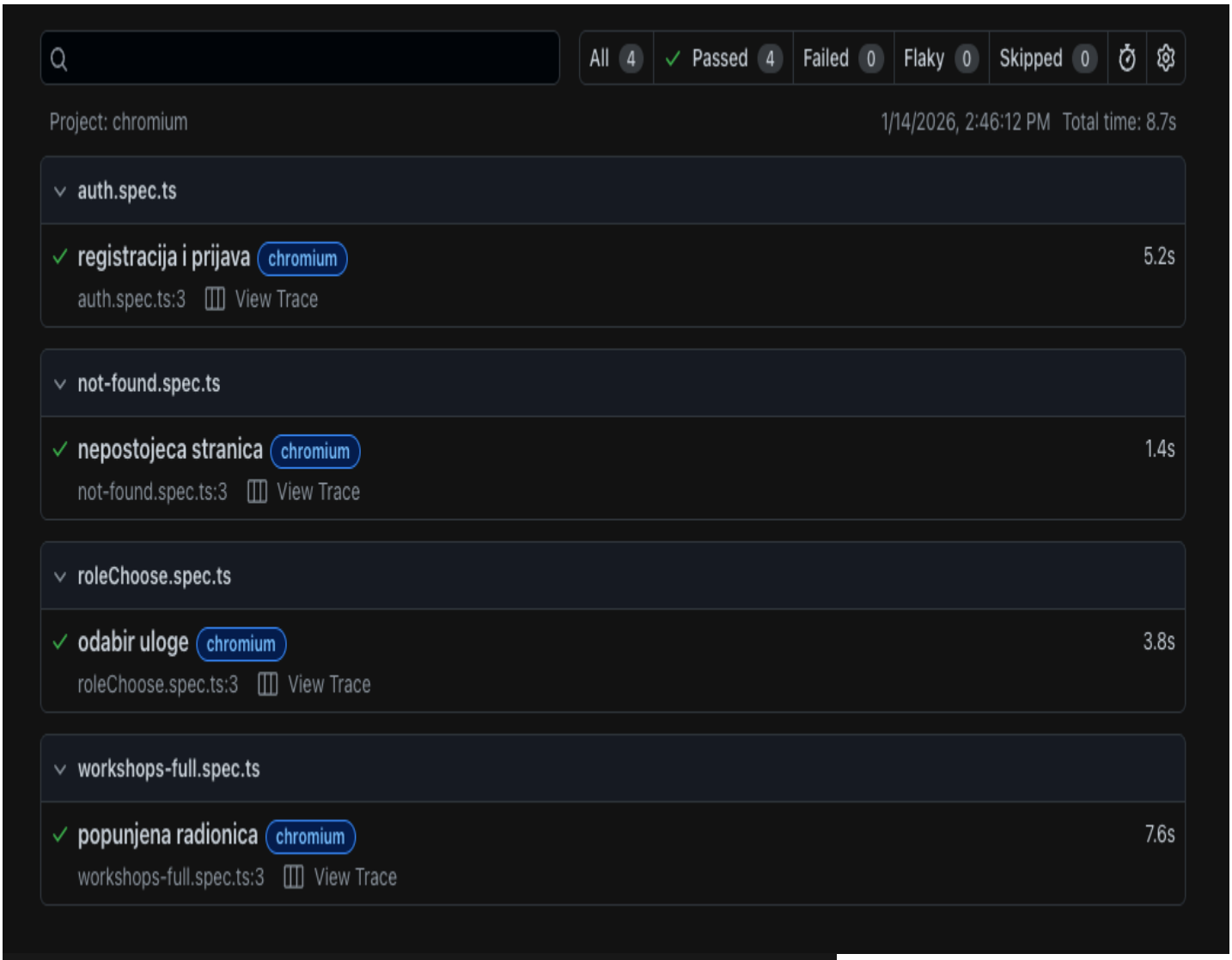
4. Dobiveni izlaz

- Playwright E2E test prolazi bez grešaka

Kod

```
import { test, expect } from '@playwright/test';

test('nepostojeca stranica', async ({ page }) => {
  await page.goto('http://localhost:5173/');
  await expect(page).toHaveURL('/auth');

  await page.goto('http://localhost:5173/ovo-ne-postoji-123');

  const errorHeading = page.getByText('404 - Page Not Found');
  await expect(errorHeading).toBeVisible();

  await page.goBack();

  await expect(page).toHaveURL('/auth');
});
```

PLAYWRIGHT REPORT - end-to-end testovi




```
> clayplay@0.0.0 e2e
> playwright test


Running 4 tests using 4 workers
  4 passed (8.7s)

To open last HTML report run:

  npx playwright show-report e2e/test-results/html-report
```

Korištene tehnologije i alati

1. Programski jezici:

`TypeScript 5.9.3`

> https://www.typescriptlang.org/

TypeScript (TS) je programski jezik visoke razine koji dodaje statičko tipiziranje s opcionalnim anotacijama tipova u JavaScript. Namijenjen je razvoju velikih aplikacija i prevodi se u JavaScript. Tipovi smanjuju broj grešaka tijekom razvoja i olakšavaju održavanje koda u timu.

`Python 3.13.5`

> https://www.python.org/

Python je programski jezik visoke razine opće namjene. Koristi dinamičku provjeru tipova i automatsko upravljanje memorijom te podržava više programskih paradigmi, uključujući strukturirano (posebno proceduralno), objektno orijentirano i funkcijsko programiranje. Omogućuje brzu izradu backend logike i skripti. Također, Python omogućuje brzu iteraciju i jednostavno testiranje.

`CSS 3`

> https://www.w3.org/Style/CSS/Overview.en.html

CSS je jezik koji se koristi za definiranje izgleda i oblikovanja dokumenata napisanih u jezicima poput HTML-a ili XML-a. Omogućuje dosljedno oblikovanje korisničkog sučelja, responzivni dizajn i odvajanje prezentacije od logike aplikacije. Koristi se uz HTML i JavaScript.

`HTML 5`

> https://html.com/

HTML je standardni označni jezik za dokumente namijenjene prikazu u web pregledniku. Definira sadržaj i strukturu web sadržaja te se često koristi uz CSS i JavaScript.

`JavaScript ES2023`

> https://www.javascript.com/

JavaScript je programski jezik visoke razine. Ima dinamičko tipiziranje, prototipski temeljenu objektno orijentiranu strukturu i funkcije prve klase. Podržava događajno, funkcijsko i imperativno programiranje. Omogućava implementaciju klijentske logike i korisničkog iskustva, poput dinamičkog prikaza sadržaja,

upravljanja stanjima i asinkronih zahtjeva prema serveru. Standardni je jezik izvršavanja u web pregledniku i osigurava kompatibilnost na svim modernim platformama. Koristi se u kombinaciji s HTML-om te CSS-om.

2. Radni okviri i biblioteke:

Frontend:

`React 19.2.0`

> https://react.dev/

Popularna JavaScript biblioteka za izgradnju interaktivnih korisničkih sučelja. React omogućuje stvaranje samostalnih komponenti koje se mogu ponovno koristiti i lako ažurirati, čime se poboljšava učinkovitost razvoja.

`Tailwind CSS 4.1.17`

> https://tailwindcss.com/

Tailwind CSS je CSS radni okvir temeljen na pomoćnim klasama koji omogućuje stiliziranje korisničkog sučelja izravno u HTML-u ili JSX-u bez potrebe za pisanjem prilagođenih CSS datoteka.

`Vite 7.2.4`

> https://vite.dev/

Vite je moderan razvojni alat za izradu web aplikacija koji omogućuje brzo pokretanje razvojnog poslužitelja i učinkovitu izradu produkcijskih verzija aplikacije.

`Shadcn UI 3.7.0`

> https://ui.shadcn.com/

ShadcnUI je biblioteka komponenata koja značajno ubrzava, olakšava i ujednačuje frontend razvoj, koristi RadixUI i TailwindCSS.

Backend:

`Flask 3.1.2`

> https://flask.palletsprojects.com/en/stable/

Flask je Python radni okvir za razvoj web aplikacija i API-ja, dizajniran s naglaskom na jednostavnost, fleksibilnost i proširivost.

3. Baza podataka:

`PostgreSQL 17.6.`

> https://www.postgresql.org/

PostgreSQL je sustav otvorenog izvora za upravljanje relacijskim bazama podataka poznat po svojoj pouzdanosti, robusnosti i bogatoj podršci za SQL standard. Često se koristi za web aplikacije, analitiku i skladištenje podataka zbog svoje skalabilnosti, performansi i jake zajednice korisnika.

`Supabase`

> https://supabase.com/

Supabase je platforma otvorenog izvora za izgradnju aplikacija koja pruža backend usluge poput baza podataka, autentikacije, pohrane datoteka i real-time funkcionalnosti. Temelji se na PostgreSQL-u i omogućuje brzu izradu skalabilnih web i mobilnih aplikacija bez potrebe za upravljanjem infrastrukturom. Supabase je poznat po jednostavnosti, integracijama i API-jevima za razvoj modernih aplikacija.

4. Razvojni alati: Popis korištenih IDE-ova, alata za verzioniranje

`VS Code`

> https://code.visualstudio.com/

VS Code nudi podršku za veliki broj programskih jezika kroz ekstenzije. Omogućuje učinkovito pisanje i održavanje koda zahvaljujući ugrađenom debugiranju te kontroli verzija (Git). Brz je, prilagodljiv i besplatan, što ga čini pogodnim za individualni i timski razvoj.

`Git 2.51.2.`

> https://git-scm.com/

Git je distribuirani sustav za kontrolu verzija koji omogućuje praćenje promjena u izvornom kodu, grananje (branching) i spajanje (merging) te jednostavnu suradnju više developera na istom projektu.

`GitHub`

> https://github.com/

GitHub je web platforma temeljena na Gitu koja omogućuje udaljeni smještaj repozitorija, upravljanje granama i pull requestovima te timsku suradnju kroz issue-e i code review.

5. Alati za ispitivanje:

`Vitest 4.0.16`

> https://vitest.dev/

Vitest je odabran za unit testiranje radi jednostavne integracije u Vite+TypeScript frontend projekta.

`Playwright 1.57.0`

> https://playwright.dev/

Playwright je odabran za e2e (system) testove radi mogućnosti testiranja u stvarnim preglednicima (Chromium/Firefox/WebKit) sa simuliranim korisničkim akcijama.

6. Cloud platforma:

Vercel (frontend):

Vercel je odabran za hosting klijentskog dijela aplikacije jer omogućuje jednostavno i brzo objavljivanje web aplikacije te pouzdanu dostupnost korisnicima. Platforma je bila pogodna za projekt jer ne zahtijeva složenu konfiguraciju i omogućuje lako ažuriranje aplikacije tijekom razvoja.

Render (backend):

Render je korišten za hosting poslužiteljskog dijela aplikacije jer pruža jednostavno rješenje za pokretanje serverskih aplikacija bez potrebe za upravljanjem vlastitim serverima. Odabran je zbog preglednosti, stabilnosti i jednostavnosti korištenja, što je bilo posebno važno za studentski tim i ograničeno vrijeme razvoja.

1. Instalacija

Frontend:

- **Node.js 22.14.0**
- **ES2023**
- **NPM 9.2.0**

Backend:

- **Flask 3.1.2**
- **Python 3.8+**

Baza podataka:

- **PostgreSQL 17.6.**

Pokretanje ClayPlay aplikacije lokalno

Prvo klonirati repozitorij: `git clone https://github.com/robinkov/progi-projekt.git`

Zatim se pozicionirati u mapu `/progi-projekt` naredbom `cd progi-projekt`

Backend

Treba imati instaliran **Python** s verzijom **3.8** ili novijom te Pythonov package manager `pip`

Napraviti `.env` datoteku u mapi `/backend` sa sljedećim ključevima:

```
DATABASE_URL = postgresql://postgres.sioqgofegshqcrmwbqtd:$$Progi1231@aws-1-eu-west-1.pooler.supabase.com:5432/p
SUPABASE_ANON_KEY = eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InNpb3Fnb2ZlZ3NocWNybXd
SUPABASE_SERVICE_KEY = eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InNpb3Fnb2ZlZ3NocWNy
SUPABASE_URL = https://sioqgofegshqcrmwbqtd.supabase.co
SECRET_KEY = $$Progi1231
GOOGLE_REDIRECT_URL = http://localhost:5000/auth/google/callback
GITHUB_REDIRECT_URL = http://localhost:5000/auth/github/callback
SUPABASE_VERIFY_URL = https://sioqgofegshqcrmwbqtd.supabase.co/auth/v1/callback
SUPABASE_JWT_SECRET = n8SwyxBbDmfduWrYFJRQ0ct/f30UM1I/x8lJ9vIwLEYqGUuWHFeSzDnR0b+bnu1yCz5T1YY2T5plOqyg7XvRtQ==
PAYPAL_CLIENT_ID = ARXyr_WfSF1KmFDFtp6FUNOJvCXnalaf9yBXHyouQFozXdmUHolBhU0iTIyf_N565XP08BX8G58aSOwF
PAYPAL_SECRET = EHfUI-XYyr7avXxKtlXJONU___S_dCwmv1ezJZx6djHONJlZjD9bOWhDlMu1pbnSPHKc3KcUHe-U1krQ
```

Za početak treba osigurati da su instalirani svi moduli:

```
cd backend
```

```
pip install -r requirements.txt
```

Ako su svi moduli uspješno instalirani onda se pokreće sami backend:

```
python run.py
```

Frontend

Nakon što je backend pokrenut, otvoriti novi shell i ovoga puta navigirati se u `frontend/` mapu:

```
cd frontend
```

Isto treba napraviti `.env` datoteku sa sljedećim ključevima:

```
VITE_APP_URL=http://localhost:5173
VITE_BACKEND_URL=http://localhost:9000
VITE_SUPABASE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InNpb3Fnb2ZlZ3NocWNybXdic
```

Treba pripaziti da je na računalo instaliran `npm` package manager i zatim instalirati sve module:

```
npm install
```

Ako su svi moduli uspješno instalirani, može se pokrenuti frontend:

```
npm run dev
```

Aplikaciji se zatim pristupa upisivanjem `localhost:5173` u internetski preglednik.

Upute za administratore

Administratorske ovlasti aktiviraju se i dobivaju prijavom (Login) sa sljedećim podacima:

Email: `admin@clayplay.com`

Password: `admin123`

Postupak postavljanja aplikacije u produkcijsko okruženje

Backend

Backend se postavlja na Render.
Nakon postavljanja profila, dodaje se novi projekt klikom na New -> Web Service.
Zatim kreće postavljanje sa sljedećim parametrima:

Odjeljak	Postupak
Source Code	Dodati https://github.com/robinkov/progi-projekt repozitorij.
Name	ClayPlay
Language	Odabrati 'Python 3' iz padajućeg izbornika
Branch	main
Region	Frankfurt (EU Central)
Root Directory	`backend`
Build Command	`pip install -r requirements.txt`
Start Command	`python run.py`

Nakon toga treba odabrati paket pretplate, Free opcija je sasvim dovoljna za ovaj projekt, te varijable okruženja.

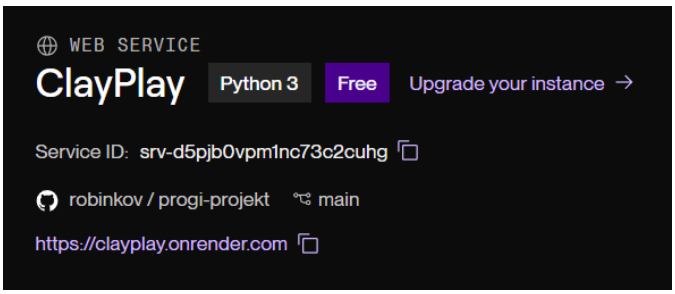
Mogu se uvesti kroz `.env` datoteku ili ručno dodati:

Environment variable	Value
DATABASE_URL	`postgresql://postgres.sioqgofegshqcrmwbqtd:$$Progi1231@aws-1-eu-west-1.pooler.supabase.com:5432/postgres`
SUPABASE_ANON_KEY	`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InNpb3Fnb2ZlZ3NocWNybXdicXRkIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NjE3NTkxNzksImV4cCI6MjA3NzMzNTE3OX0.Yzwssz1RUS93uwIUtpFJgc1GJHCCLp_dHV0Z5NxzMAo`
SUPABASE_SERVICE_KEY	`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InNpb3Fnb2ZlZ3NocWNybXdicXRkIiwicm9sZSI6InNlcnZpY2Vfcm9sZSIsImlhdCI6MTc2MTc1OTE3OSwiZXhwIjoyMDc3MzM1MTc5fQ.CHdcvo0mOz2mx4mHEbixbUitBEMdRLQph-jDsfKNRy8`

Environment variable	Value
SUPABASE_URL	`https://sioqgofegshqcrmwbqtd.supabase.co`
SECRET_KEY	`$$Progi1231`
GOOGLE_REDIRECT_URL	`http://localhost:5000/auth/google/callback`
GITHUB_REDIRECT_URL	`http://localhost:5000/auth/github/callback`
SUPABASE_VERIFY_URL	`https://sioqgofegshqcrmwbqtd.supabase.co/auth/v1/callback`
SUPABASE_JWT_SECRET	`n8SwyxBbDmfduWrYFJRQ0ct/f30UM1I/x8lJ9vIwLEYqGUuWHFeSzDnR0b+bnu1yCz5T1YY2T5plOqyg7XvRtQ==`
PAYPAL_CLIENT_ID	`ARXyr_WfSF1KmFDFtp6FUNOJvCXnalaf9yBXHyouQFozXdmUHolBhU0iTIyf_N565XP08BX8G58aSOwF`
PAYPAL_SECRET	`EHfUI-XYyr7avXxKtlXJONU___S_dCwmv1ezJZx6djHONJlZjD9bOWhDlMu1pbnSPHKc3KcUHe-U1krQ`

Konačno treba kliknuti na Deploy.

Provjera uspješnog postavljanja:



Kliknuti na link te bi se trebala otvoriti stranica sa sljedećim tekstom:

```
{
  "error": "Missing token",
  "success": false
}
```

Frontend

Za postavljanje frontend dijela aplikacije koristit će se Vercel. Nakon uspješnog postavljanja profila kreira se novi projekt Add New -> Project.

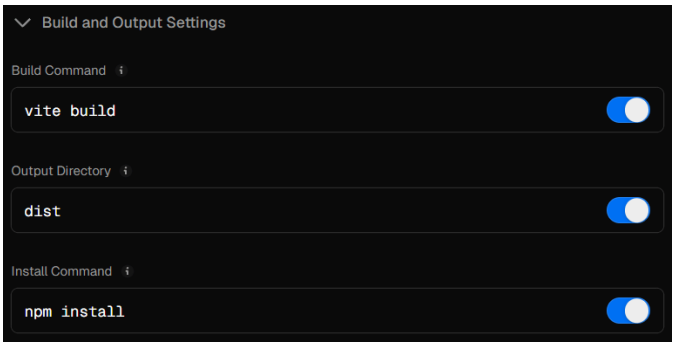
Dodati GitHub repozitorij https://github.com/robinkov/progi-projekt

Postaviti Project Name na 'clayplay'.

Kao Framework Preset iz padajućeg izbornika odabrati 'Vite'.

Root Directory postaviti na `frontend`.

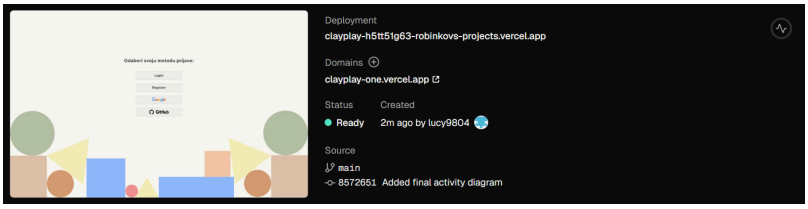
U Build and Output Settings staviti sljedeće vrijednosti:



Na kraju se postavljaju varijable okruženja.
Mogu se uvesti kroz `.env` datoteku ili ručno dodati:

Environment variable	Value
VITE_BACKEND_URL	**Uzeti URL od backenda koji se može pronaći na Renderu**
VITE_SUPABASE_KEY	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InNpb3Fnb2ZlZ3NocWNybXdicXRkIiwicm9sZSI6ImFub24iL

Kliknuti na Deploy.



Uzeti URL iz 'Domains' popisa te navigirati u Supabase projekte > ClayPlay projekt > Authentication > URL Configuration te u Site URL staviti URL iz 'Domains' popisa.

Pristup ClayPlay aplikaciji na javnom poslužitelju

Aplikaciji se pristupa putem sljedećeg URL-a: https://clayplay-one.vercel.app/

Prijava za glavnog administratora sustava:

E-mail: admin@clayplay.com

Password: admin123

Plaćanja se vrše preko lažnih PayPal kartica.

Iste su funkcionalnosti kao i kod lokalne verzije.

Zadatak projektnog tima bio je razvoj web aplikacije ClayPlay, čija je svrha digitalizirati organizaciju i promociju keramičkih radionica te omogućiti jednostavnu interakciju između organizatora i korisnika. Aplikacija služi kao centralna platforma na kojoj korisnici mogu pregledavati i rezervirati radionice keramike, dok organizatori mogu upravljati vlastitim radionicama, objavljivati nove termine te nuditi gotove keramičke proizvode. Sustav je osmišljen s ciljem objedinjavanja informacija, pojednostavljenja komunikacije i smanjenja administrativnog opterećenja organizatora radionica.

Tijekom semestra, kao i svaki tim, i naš je tim prošao kroz različite faze timskog rada. Prva faza bila je faza okupljanja tima (forming), tijekom koje smo definirali osnovnu ideju projekta, upoznali se sa zahtjevima zadatka te okvirno podijelili uloge i odgovornosti. U ovoj fazi naglasak je bio na planiranju i izradi početne dokumentacije, koja je poslužila kao temelj za daljnji razvoj sustava.

U drugoj fazi timskog rada (storming) pojavile su se poteškoće u procjeni opsega zadatka. Ova faza obilježena je prilagodbama inicijalne podjele poslova te dodatnim raspravama oko tehničkih rješenja. Unatoč tome, komunikacijom i zajedničkim dogovorima tim je uspio uspostaviti jasniji način rada.

Treću fazu (norming) obilježilo je stabiliziranje timske dinamike. U ovoj fazi članovi tima počeli su učinkovitije surađivati, a rad je postao organiziraniji i usmjereniji prema cilju. Implementirane su osnovne funkcionalnosti aplikacije, dok je dokumentacija dodatno prilagođena i ažurirana na temelju stvarnog stanja sustava i stečenog iskustva tijekom implementacije.

Završna faza timskog rada (performing) bila je usmjerena na dovršetak projekta. Svaki je član tima radio na svom dijelu sustava kako bi se osigurala funkcionalna i stabilna cjelina. U ovoj fazi implementirane su preostale funkcionalnosti, provedeno je testiranje sustava te dovršena cjelokupna dokumentacija. Na kraju su svi dijelovi aplikacije integrirani u jedinstveni sustav spreman za predaju.

Nakon završetka projekta slijedi faza raspuštanja tima (adjourning), koja dolazi po uspješnoj predaji i završetku projektnog zadatka.

Projekt ClayPlay bio je vrijedna prilika za stjecanje iskustva rada u timskom okruženju i prilagodbu različitim stilovima rada. Uz tehnička znanja, poput razvoja web aplikacija, izrade dokumentacije i modeliranja sustava, razvijene su i važne organizacijske i komunikacijske vještine.

Također, projekt je dodatno produbio razumijevanje rada s alatima za verzioniranje, dizajna softverske arhitekture te primjene teorijskih znanja u praktičnom kontekstu. Iako postoji prostor za poboljšanje, osobito u ranijem planiranju i raspodjeli zadataka, smatramo da je projekt postavio čvrste temelje za buduće timske i profesionalne projekte.

Kontinuirano osvježavanje

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

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Initial Home page	Marin Mikulčić	20.10.2025.
0.2	Prvi draft opisa projektnog zadatka	Marin Mikulčić	20.10.2025.
0.3	Dodani funkcionalni i nefunkcionalni zahtjevi	Marin Mikulčić	20.10.2025.
0.4	Reformatirani funkcionalni i nefunkcionalni zahtjevi	Robin Kovačić	24.10.2025.
0.5	Promijenjene neke klasifikacije i dokumentacijski zahtjevi spojeni u jednu točku	Marin Mikulčić	27.10.2025.
0.6	Dodano još dionika i aktora	Leon Krivski	11.11.2025.
0.7	Kloniran repozitorij profesora Sruk te dodan predlozak za opis arhitekture	Marin Mikulčić	13.11.2025.
0.8	Prepravljeni obrasci uporabe te opisi funkcionalnosti	Josip Bušelić, Leon Krivski	14.11.2025.
0.9	Napravljeni sekvencijski dijagrami te detaljno napisani opisi njihove funkcionalnosti	Lucija Kozić, Marin Mikulčić	14.11.2025.
1.0	Dodani dijagrami baze podataka te tablice koje opisuju bazu podataka, dodan dijagram razreda s detaljnim opisom	Lovre Jurjević	14.11.2025.
1.1	Ažuriran opis arhitekture, nadodana skica arhitekture i dijagram visoke razine	Marin Mikulčić	14.11.2025.
1.2	Ažuriran README.md, nadodani funkcijski i nefunkcijski zahtjevi	Lucija Kozić	3.1.2026.
1.3	Napravljen dijagram aktivnosti te njegov opis	Lucija Kozić	8.1.2026.
1.4	Napravljen dijagram satnja te njegov opis	Marin Mikulčić	9.1.2026.
1.5	Napravljen novi dijagram aktivnosti te njegov opis	Lucija Kozić	12.1.2026.
1.6	Dokumentirani ispitni slučajevi	Leon Krivski	14.1.2026.
1.7	Popravljen README.md	Lucija Kozić	19.1.2026.
1.8	Napisan zaključak i budući rad	Marin Mikulčić	19.1.2026.
1.9	Uređena i nadopunjena wiki stranica 7. Alati (Alati za razmještaj, Cloud platforma)	Leon Krivski	20.1.2026.
2.0	Napravljeni novi sekvencijski dijagrami i njihovi opisi	Lucija Kozić	20.1.2026.
2.1	Ispravljen dijagram komponenata i dijagram razmještaja	Lucija Kozić	21.1.2026.
2.2	završena sekcija 8. Upute za puštanje u pogon	Leon Krivski	22.1.2026.

Dnevnik sastajanja

1. sastanak

- Datum: 10. listopada 2025.
- Prisustvovali: R.Kovačić, L.Krivski, L.Kozić, M.Mikulčić, R.Matek, J.Bušelić, L.Jurjević
- Tema sastanka:
    - utvđenje teme Clayplay te opisivanje funkcijskih zahtjeva
    - izrada github projekta
    - dogovoren voditelj tima

2. sastanak

- Datum: 17. listopada 2025.
- Prisustvovali: R.Kovačić
- Teme sastanka:
    - detaljno opisani funkcijski i nefunkcijski zahtjevi
    - dogovoreno ime tima
    - objašnjen način individualnog ocjenjivanja

3. sastanak

- Datum: 31. listopada 2025.
- Prisustvovali: R.Kovačić, L.Krivski, L.Kozić, M.Mikulčić, R.Matek, J.Bušelić, L.Jurjević
- Teme sastanka:
    - prezentiran dosadasnji napredak aplikacije i dokumentacije
    - dogovoreno preformatiranje funkcionalnih i nefunkcionalnih zahtjeva

4. sastanak

- Datum: 14. studeni 2025.
- Prisustvovali: R.Kovačić, L.Krivski, L.Kozić, M.Mikulčić, R.Matek, J.Bušelić, L.Jurjević
- Teme sastanka:
    - prezentirane trenutne funkcionalnosti aplikacije te trenutni napredak sa dokumentacijom
    - dobivene povratne informacije sa područjima za poboljšanje i dodavanje

5. sastanak

- Datum: 12. prosinca 2025.
- Prisustvovali: R.Kovačić, L.Krivski
- Teme sastanka:
    - dobivene povratne informacije o drugom krugu ocjenjivanja i načinu testiranja aplikacije

6. sastanak

- Datum: 9. siječnja 2026.
- Prisustvovali: R.Kovačić, L.Kozić
- Teme sastanka:
  - dobivene povratne informacije o servisu za plaćanje (PayPal)
  - dobivene povratne informacije o dijagramu stanja i dijagramu aktivnosti

7. sastanak

- Datum: 16. siječnja 2026.
- Prisustvovali: R.Kovačić, L.Krivski, L.Kozić, M.Mikulčić, R.Matek, J.Bušelić, L.Jurjević
- Teme sastanka:
  - predstavljena alfa verzija
  - dobivene informacije o dijagramu razmještaja i dijagramu komponenti
  - definirane potrebne promjene u dokumentaciji

8. sastanak

- Datum: 19. siječnja 2026.
- Prisustvovali: R.Kovačić, L.Kozić, M.Mikulčić
- Teme sastanka:
  - ujednačavanje dokumentacije i implementacije

Plan rada

ClayPlay (16 tjedana, uključujući praznike)

Legenda: ● = aktivna aktivnost u tom tjednu
Članovi tima:

- **JB** – Josip Bušelić (backend)
- **LJ** – Lovre Jurjević (baza podataka, backend)
- **RK** – Robin Kovačić (frontend)
- **LK** – Lucija Kozić (UI/UX dizajn, dokumentacija)
- **KL** – Leon Krivski (testiranje, frontend)
- **RM** – Roko Matek (backend)
- **MM** – Marin Mikulčić (dokumentacija)

Aktivnost / Tjedan	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Članovi

Aktivnost / Tjedan	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Članovi
Inicijalizacija projekta (README)	●	●															RK, MM, RM
Dokumentacija		●	●			●	●						●	●	●	●	MM, LK, RM
Dizajn UI/UX		●	●														LK
Postavljanje baze i Supabase integracija			●	●													LJ
Autentikacija				●	●												RK, JB
Uloge korisnika (admin, organizator, korisnik)					●	●									●		RK
Upravljanje profilom korisnika					●	●	●								●		RK
Kreiranje i uređivanje radionica						●	●	●									JB, RM, LJ
Rezervacija radionica							●	●	●								RM, LJ
Pregled i odobravanje prijava (admin)								●	●								LJ
Frontend osnovne stranice i navigacija						●	●	●									RK, KL
Forum i diskusije									●	●							LJ
WebShop										●	●						KL
Filtriranje proizvoda											●	●					KL
PayPal integracija												●	●				LJ
Notifikacije													●	●			LJ
Membership sustav i planovi (admin panel)													●	●			RK
Kalendar i prikaz događaja													●	●			RK, JB
Testiranje (unit + e2e)													●	●			KL
Bug fixing													●	●	●	●	RK, LJ, KL
Deploy konfiguracija															●		RK, JB

Tablica aktivnosti

Kontinuirano osvježavanje

Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti. Potrebno je navesti koliko je sati koja osoba uložila u pojedinu komponentu, možete oblikovati tablicu ili ispisati za svaku osobu.

Stavka	Kovačić	Kozić	Krivski	Jurjević	Matek	Bušelić	Mikulčić
Upravljanje projektom	10	5	8	3	3	3	5
Opis projektnog zadatka	2	0	0	0	0	0	2
Funkcionalni zahtjevi	0	0	2	0	0	0	4
Opis pojedinih obrazaca	0	2	0	0	0	0	4
Dijagram obrazaca uporabe	0	0	5	0	0	0	3
Sekvencijski dijagrami	0	10	0	0	0	0	2
Opis ostalih zahtjeva	0	0	0	0	0	0	1
Arhitektura i dizajn sustava	0	3	0	0	0	0	3
Baza podataka	0	0	5	7	0	0	0
Dijagram razreda	0	0	0	0	0	0	0
Dijagram stanja	0	0	0	0	0	0	3
Dijagram aktivnosti	0	6	0	0	0	0	0
Dijagram komponenti	0	4	0	0	0	0	0
Korištene tehnologije i alati	0	0.5	1	0	0	0	2
Ispitivanje programskog rješenja	0	0	18	0	0	0	0
Dijagram razmještaja	0	3	0	0	0	0	0
Upute za puštanje u pogon	4	0.5	0,5	0	0	0	0
Dnevnik sastajanja	0	1	0	0	0	0	1
Zaključak i budući rad	0	1	0	0	0	0	0
Popis literature	0	0	0	0	0	0	0.5
Izrada početne stranice	8	0	0	0	0	0	1
Dizajn web stranica	0	6	9	0	0	0	0

Stavka	Kovačić	Kozić	Krivski	Jurjević	Matek	Bušelić	Mikulčić
Izrada baze podataka	0	0	0	2	0	0	0
Frontend	30	0	29	0	5	0	0
Spajanje s bazom podataka	0	0	0	1	1	1	0
Backend	0	0	4	0	20	20	0
Deploy	10	0	0	0	0	0	0
Izrada prezentacije projekta	0	3	0,5	0	0	0	0

Dijagram pregleda promjena

Kjučni izazovi i rješenja

Tijekom razvoja web aplikacije ClayPlay, tim se suočio s nekoliko ključnih izazova. Jedan od glavnih izazova bio je usklađivanje rasporeda i dostupnosti članova tima, što je u određenim fazama rezultiralo

sporijim napretkom i kašnjenjem u implementaciji pojedinih funkcionalnosti. Osim organizacijskih poteškoća, pojavili su se i tehnički izazovi vezani uz integraciju vanjskih servisa, posebice u području autentifikacije korisnika i povezivanja s bazom podataka, kao i prilagodbu frontenda i backenda zajedničkoj arhitekturi sustava.

Navedeni izazovi riješeni su boljom podjelom zadataka unutar tima, jasnijim definiranjem odgovornosti te kontinuiranom komunikacijom među članovima. Tehnički problemi rješavani su postupnim pristupom, testiranjem pojedinih modula te korištenjem službene dokumentacije korištenih tehnologija. Kroz proces rješavanja problema, tim je stekao vrijedna praktična znanja o razvoju web aplikacija, timskom radu i integraciji vanjskih servisa, što je značajno doprinijelo kvaliteti konačnog rješenja i profesionalnom razvoju svih članova tima.