

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

SportConnect

Grupa: TG12.3

Tim SportConnect:

Ime člana	Područje rada
Klara Katić	Dizajn, baza podataka, frontend
Hana Ćerić	Baza podataka, backend
Vid Knežević	Frontend
Luka Zuanović	Frontend
Luka Đuretić	Frontend, backend
Viktor Pijanec	Backend, baza podataka
Manuel Fijan	Backend, baza podataka

Nastavnik: Vlado Sruk

DETALJNI OPIS

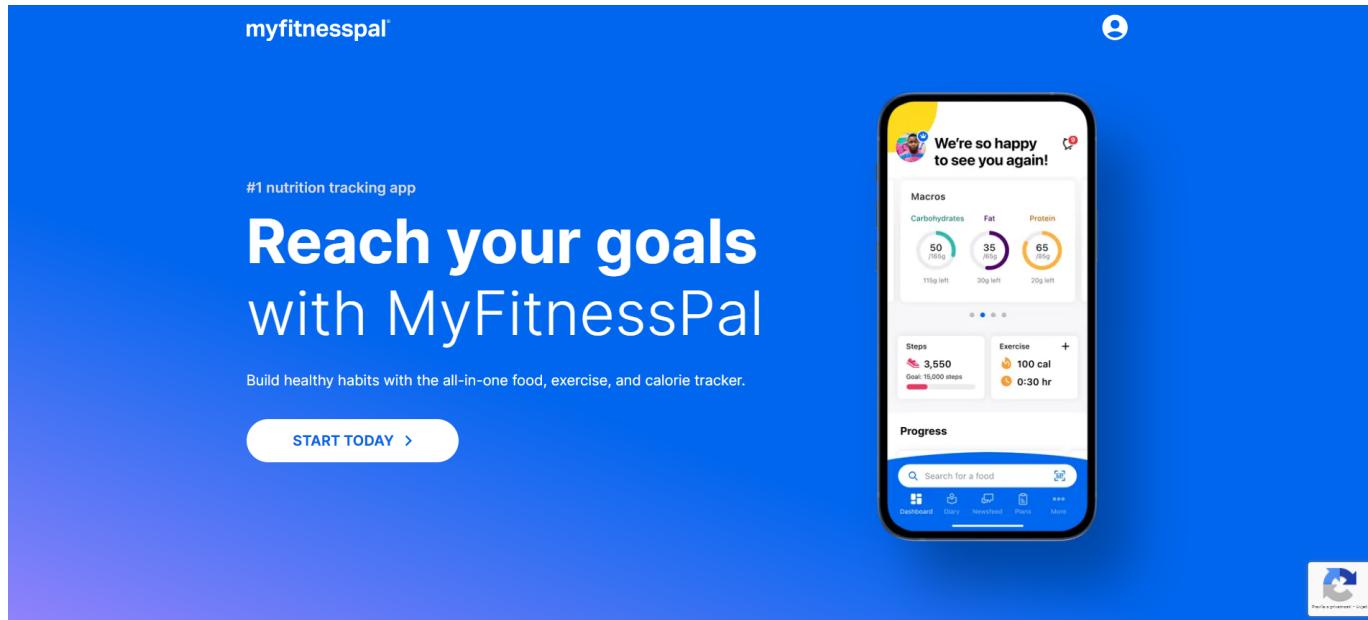
Aplikacija *SportConnect* predstavlja inovativno rješenje u svijetu sporta, wellnessa i zdravog načina života, gdje korisnici mogu pronaći inspiraciju, pristupiti stručnim savjetima te se povezati s profesionalcima i istomišljenicima. Cilj ovog projekta je stvoriti online zajednicu koja omogućava razmjenu znanja i resursa, omogućujući korisnicima (klijentima) da kroz pretplate pristupe kvalitetnom sadržaju, dok partneri (stručni korisnici) dobivaju priliku prezentirati vlastite savjete, planove i treninge, te povećati vlastiti ugled kroz sustav rangiranja (brončani, srebrni i zlatni status). Ova platforma korisnicima omogućava direktnu komunikaciju s partnerima, što predstavlja dodatnu vrijednost za svakoga tko želi unaprijeđivati zdravlje i fitness putem podrške i iskustva drugih. Osnovni cilj *SportConnecta* je omogućiti jednostavan pristup relevantnim informacijama koje će pomoći korisnicima u postizanju osobnih ciljeva vezanih uz zdrav način života.

SportConnect se fokusira na rješavanje problema povezanosti i jednostavnog pristupa relevantnim informacijama u području zdravog života. U današnjem vremenu postoji mnoštvo informacija dostupnih online, ali korisnici često nailaze na sadržaj neprovjerenog kvaliteta ili informacije koje nisu prilagođene njihovim specifičnim potrebama. *SportConnect* rješava ovaj problem tako što omogućava pristup certificiranim stručnjacima i profesionalcima kroz personalizirani sustav rangova. Klijenti imaju mogućnost

ocjenjivati sadržaj partnera, što doprinosi usponu partnera unutar sustava rangova, a time i povećanju njihove vidljivosti. Funkcionalnosti aplikacije uključuju sve potrebne korake za stvaranje korisničkog profila, pristup sadržaju, spremanje objava, komunikaciju među korisnicima, te sustav uplata i isplata putem sigurnosnih protokola kao što je OAuth 2.0. Time se osigurava sigurno i pouzdano korisničko iskustvo, uz neprekidnu mogućnost pristupa informacijama i relevantnim objavama.

Ovaj projekt ima veliki potencijal u društvenom i poslovnom smislu, jer podržava i popularizira zdrav način života, povezujući ljude kroz zajedničke ciljeve i vrijednosti. Klijentima pruža alat za jednostavno pronalaženje stručnih savjeta i inspiracije, dok partnerima omogućava stvaranje dodatnih izvora prihoda kroz monetizaciju svog sadržaja. Kroz personalizirane upute i pristup specifičnom sadržaju, *SportConnect* korisnicima pomaže u razvijanju zdravih navika, što može pozitivno utjecati na njihov svakodnevni život. Također, platforma omogućava stručnjacima da dosegnu širu publiku, povežu se s potencijalnim klijentima, te unovče svoje savjete i planove. Razmjena znanja među korisnicima s različitim razinama iskustva doprinosi stvaranju podržavajuće zajednice koja je posvećena unaprjeđivanju kvalitete života.

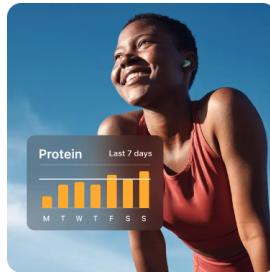
Postojeće aplikacije kao što su *MyFitnessPal* i *Strava* nude slične funkcionalnosti u smislu praćenja aktivnosti i dijeljenja napretka. Ipak, *SportConnect* se razlikuje po tome što uvodi jedinstven model rangiranih korisnika, u kojem stručnjaci dobivaju ocjene prema kvaliteti svog sadržaja i angažmanu korisnika. Dok *MyFitnessPal* nudi osnovne funkcionalnosti praćenja prehrane i treninga, a *Strava* omogućava praćenje sportskih aktivnosti, *SportConnect* pruža korisnicima mogućnost personaliziranog mentorstva. Ova aplikacija je dizajnirana tako da omogućuje korisnicima izravnu komunikaciju s partnerima, pristup sadržaju prema odabranom rangu, te osigurava jedinstvenu kombinaciju društvene mreže i platforme za mentorstvo. Također, *SportConnect* uključuje različite opcije za monetizaciju za partnere, što ga čini konkurentnim rješenjem koje nudi dodatne pogodnosti u usporedbi s postojećim aplikacijama.



MyFitnessPal početna stranica



Welcome to
myfitnesspal



Ready for some wins?
Start tracking, it's easy!



Discover the impact of
your food and fitness.



And make mindful eating
a habit for life

[CONTINUE](#)



MyFitnessPal prijava

[STRAVA](#)
 [Activities](#) ▾
 [Features](#)
 [Maps](#)
 [Challenges](#)
 [Subscription](#)

[Log In](#)



Community-Powered Motivation

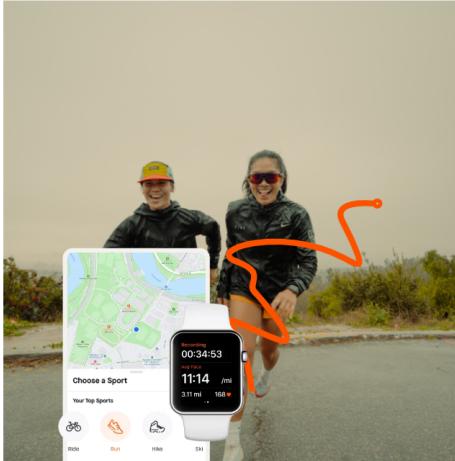
Track your progress and cheer each other on. Join over 100 million active people on Strava for free.

Already a Member? [Log In](#)

[Sign Up With Facebook](#)
 [Sign Up With Google](#)
 [Sign Up With Apple](#)

[Sign Up With Email](#)

By signing up for Strava, you agree to the [Terms of Service](#). View our [Privacy Policy](#).



STRAVA

[Features](#)

[Routes](#)

[What's New](#)

[Privacy](#)

[Subscription](#)

[Stories](#)

[Do Not Share My](#)

Strava početna stranica

[STRAVA](#)
 [Dashboard](#) ▾
 [Training](#) ▾
 [Maps](#)
 [Challenges](#)

 [Give a Gift](#)
[Start Trial](#)
 [1](#)



Davor
26 October 2024 at 13:26

Afternoon Swim

Distance: 1,599 m | Time: 34m 24s | Pace: 2:09 /100m

Latest Activity
Morning Activity • 30 Aug 2019

Your Training Log >

RELATIVE EFFORT

Disclaimer

This feature is not a replacement for a coach or health advice. The more we learn about you, the better information we can provide, but data is not the only indicator of health or fitness. Please pay attention to your body and stay safe. Consult with your doctor before beginning any exercise program.



8 kudos

Suggested Friends



Tomislav
Slavonski Brod, Brodsko-posavska županija, Croatia
You have mutual friends on Strava

[Follow](#)



Nenad
You have mutual friends on Strava

[Request to Follow](#)



Josip Fett
Bayerisch Gmain, Bavaria, Germany
You have mutual friends on Strava

[Request to Follow](#)

[Find and Invite Your Friends](#)

[Community Hub](#) [Support](#) [Subscription](#)
[Student Discount](#) [Terms and Conditions](#)
[Privacy Policy](#)
[Do Not Share My Personal Information](#)

Strava glavna stranica

SportConnect je aplikacija namijenjena širokom krugu korisnika koji su zainteresirani za zdravlje, sport i wellness. Primarna ciljna skupina su ljubitelji zdravog načina života koji traže savjete, informacije i inspiraciju za održavanje zdravlja. Aplikacija je također namijenjena profesionalnim trenerima, nutricionistima i influencerima koji žele širiti svoje znanje i dosegnuti šиру publiku. Osim toga, *SportConnect* je potencijalno zanimljiv i poslovnim partnerima te organizacijama koje žele promovirati proizvode i usluge usmjerenе na zdravlje i wellness. Ova platforma omogućava korisnicima da ostvare bolju povezanost s istomišljenicima i stručnjacima, dok istovremeno promovira zdrav i aktivan stil života.

Kao prilagodljiva platforma, *SportConnect* nudi brojne mogućnosti za personalizaciju i prilagodbu korisničkog iskustva. Na primjer, korisnici mogu odabrati rang sadržaja kojem žele pristupiti (brončani, srebrni ili zlatni), što im omogućava odabir relevantnih informacija prema vlastitim potrebama. Sustav rangova omogućava kontinuirano prilagođavanje, što partnerima pruža priliku da unaprijede svoj status prema korisničkoj ocjeni. Platforma je osmišljena na način da se može lako proširiti novim funkcionalnostima ili prilagoditi specifičnim zahtjevima korisnika i poslovnih partnera, čime dodatno doprinosi njenoj fleksibilnosti.

Projekt *SportConnect* obuhvaća širok spektar tehnoloških i funkcionalnih komponenti, od razvoja korisničkog sučelja u Reactu, do backend funkcionalnosti u Springu i upravljanja bazom podataka u PostgreSQL-u. Frontend će korisnicima omogućiti jednostavno i intuitivno korištenje svih funkcionalnosti aplikacije, dok će backend osigurati sigurnost i integritet podataka. Baza podataka bit će odgovorna za pohranu korisničkih podataka, transakcija i povijesti objava, dok će sustav sigurnosti uključivati autentifikaciju putem OAuth 2.0, osiguravajući da su podaci korisnika uvijek zaštićeni. Projektni zadatak također uključuje postavljanje i održavanje sigurnosnih protokola za sve finansijske transakcije, kako bi se korisnicima osigurala potpuna zaštita prilikom korištenja aplikacije.

U budućnosti se za *SportConnect* planira nekoliko nadogradnji koje bi dodatno poboljšale korisničko iskustvo i proširile funkcionalnost aplikacije. Jedna od mogućih nadogradnji uključuje integraciju umjetne inteligencije za preporuke sadržaja temeljenih na korisničkim interesima, čime bi se poboljšala personalizacija iskustva svakog korisnika. Također, razmatra se dodavanje sustava nagrađivanja, kroz koji bi se korisnici poticali na aktivnost i angažman unutar aplikacije. Osim toga, u planu je i razvoj mobilne aplikacije za iOS i Android, koja bi omogućila još lakši i pristupačniji način korištenja platforme. Implementacija automatskog chatbota za korisničku podršku također bi mogla poboljšati iskustvo korisnika pružajući im brzu i efikasnu podršku. Sve ove nadogradnje usmjerene su prema stvaranju cjelovitog i fleksibilnog rješenja koje zadovoljava različite potrebe korisnika, partnera i poslovnih suradnika, čime se osigurava održiv rast i kontinuirani razvoj *SportConnect* platforme.

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Sustav omogućuje pristup informacija same aplikacije.	Visok	Zahtjevdionika	Korisnik može pregledati informacije same aplikacije.
F-002	Sustav omogućuje stvaranje profila i prijavu korisnika.	Visok	Interni	Korisnik se može registrirati u sustav te stvoriti profil kojim se prijavljuje u aplikaciju.

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-003	Sustav omogućuje korisnicima spremanje i označavanje objava.	Srednji	Zahtjev dionika	Korisnik može spremati i označavati objave koje pregledava.
F-004	Sustav omogućuje korisnicima komuniciranja s drugim korisnicima.	Srednji	Zahtjev dionika	Korisnik može slati i primati poruke od drugih korisnika.
F-005	Sustav omogućuje korisnicima odabir ili otkazivanje ranga na plaćanje.	Visok	Interni	Korisnik može odabrati ili otkazati rang na plaćanje.
F-006	Sustav omogućuje uplate.	Visok	Interni	Korisnik može napraviti uplate putem aplikacije kako bi mogao pregledavati objave određenog ranga
F-007	Sustav omogućuje kreiranje i uređivanje besplatnih i plaćenih objava.	Visok	Zahtjev dionika	Partner može kreirati i uređivati objave.
F-008	Sustav omogućuje isplate novčanog iznosa partnerima.	Visok	Interni	Partneri mogu primati isplate od aplikacije.
F-009	Sustav omogućuje korisnicima odabiranje besplatnog ranga.	Srednji	Interni	Korisnik može odabrati besplatni rang.
F-010	Sustav omogućuje adminu brisanje sadržaja.	Visok	Interni	Admin može obrisati sadržaj ako ga smatra neprimjerenim.
F-011	Sustav omogućuje adminu filtriranje rangova.	Visok	Interni	Admin može dodjeljivati i mijenjati rang partnera.

Ostali zahtjevi

Zahtjevi za održavanje

ID zahtjeva	Opis	Prioritet
NF-3.1.1	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje.	Visok
NF-3.1.2	Sustav treba imati dovoljnu dokumentaciju.	Srednji
NF-3.1.3	Kôd sustava treba biti dokumentiran prema "Code Conventions for the Java Programming Language" dostupnim na Oracle.	Visok

ID zahtjeva	Opis	Prioritet
NF-3.1.4	Sustav treba biti opisan putem dokumenta oblikovanja (SRS).	Srednji
NF-3.1.5	Sustav treba biti popraćen "Priručnikom za rad" koji opisuje pravilnu upotrebu sustava.	Visok
NF-3.1.6	Sustav treba imati "Plan implementacije" za pravilno postavljanje sustava.	Srednji

Zahtjevi za sigurnost

ID zahtjeva	Opis	Prioritet
NF-3.2.1	Sustav treba biti oblikovan uz primjenu sigurnosnih protokola za zaštitu podataka korisnika.	Visok
NF-3.2.2	Svi osjetljivi podaci korisnika moraju biti kriptirani i zaštićeni tijekom pohrane i prijenosa.	Visok
NF-3.2.3	Sustav treba omogućiti autentifikaciju i autorizaciju za sve korisnike prema njihovojoj razini pristupa.	Srednji

Zahtjevi za performanse

ID zahtjeva	Opis	Prioritet
NF-3.3.1	Sustav treba osigurati optimalnu brzinu odziva na zahtjeve korisnika.	Visok
NF-3.3.2	Vrijeme odziva za učitavanje glavne stranice ne smije biti duže od 2 sekunde pri standardnom opterećenju.	Srednji

Zahtjevi za upotrebljivost

ID zahtjeva	Opis	Prioritet
NF-3.4.1	Sustav treba biti jednostavan za korištenje i razumljiv za korisnike svih razina.	Visok
NF-3.4.2	Svi važni dijelovi sučelja trebaju biti intuitivno postavljeni s jasnim oznakama.	Srednji
NF-3.4.3	Sustav treba imati pomoć unutar aplikacije koja objašnjava osnovne funkcionalnosti.	Niži

Dionici:

1. Klijenti

2. Partneri
3. Admini
4. Razvojni tim
5. Budući investitori/naručitelji

Aktori:

1. Aktor 1 (admin): Admini su odgovorni za nadzor i upravljanje cijelom aplikacijom kako bi osigurali sigurnost, poštivanje pravila i funkcionalnost. Može:

pristup informacija same aplikacije (UC3, UC5), adminske brisanje sadržaja (neprimjereno sadržaj i svaki koji se kosi s pravilima aplikacije) (UC15), adminske filtriranje rangova (kada se partner tek upisuje u aplikaciju admin će kroz njegove prve objave odlučiti kojem rangu pripada) (UC16)

nadzor komunikacija i interakcija između korisnika (UC13) upravljanje i kontrola sadržaja koji se objavljuje (UC13, UC14)

2. Aktor 2 (klijent) Klijenti su korisnici koji pristupaju sadržaju i komuniciraju s partnerima radi dobivanja savjeta ili inspiracije. Može:

pristup informacija same aplikacije (UC3, UC5, UC18), stvaranje profila/prijava (UC1, UC2), odabiranje besplatnog ranga (UC8), spremanje i označavanje objava (UC5, UC7, UC9), komunikacija s ostalim korisnicima (UC6), odabiranje i otkazivanje ranga za plaćanje (UC10), uplate (UC21)

primanje potvrda o plaćanju (UC21), primanje obavijesti (UC19)

3. Aktor 3 (partner) Partneri su stručnjaci koji pružaju sadržaj (npr. savjete, treninge) te zarađuju kroz objave. Može:

pristup informacija same aplikacije (UC3, UC5, UC18), stvaranje profila/prijava (UC1, UC2), spremanje i označavanje objava (UC5, UC7, UC9), komunikacija s ostalim korisnicima (UC6), odabiranje i otkazivanje ranga za plaćanje (ranga iznad sebe) (UC10), kreiranje i uređivanje objava (UC11, UC12), uplate (UC21), isplate (UC20)

primanje potvrda o plaćanju (UC21), primanje potvrda o uplatama (UC20), primanje obavijesti (UC19)

4. Aktor 4 (Baza podataka - pasivni) Baza podataka pohranjuje sve podatke o korisnicima, objavama, rangovima i povijesti transakcija. Može:

pohranja korisničkih podataka (UC4, UC8, UC10, UC17), pohranja objava, rangova i povijesti interakcija (UC6, UC7, UC9, UC11, UC12, UC14, UC15, UC16), evidencija povijesti transakcija (UC22)

Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija

- Glavni sudionik: Neregistrirani korisnik
- Cilj: Omogućiti korisniku stvaranje korisničkog računa kako bi mogao pristupiti dodatnim funkcionalnostima aplikacije.
- Sudionici: Korisnik, Baza podataka
- Preduvjet: Korisnik je na stranici za registraciju.
- Opis osnovnog tijeka:
 1. Korisnik unosi podatke potrebne za registraciju (npr. korisničko ime, e-mail adresu i lozinku).
 2. Sustav provjerava je li e-mail adresa već registrirana.
 3. Ako e-mail nije registriran, sustav pohranjuje korisničke podatke u bazu podataka i šalje potvrdu o registraciji na e-mail korisnika.
- Opis mogućih odstupanja:
 1. Korisnik unosi podatke u neispravnom formatu ili neka polja ostavlja praznima.
 - Sustav prikazuje poruku o pogrešci i traži od korisnika da ispravi podatke.
 2. E-mail adresa već postoji u sustavu.
 - Sustav obavještava korisnika da je e-mail adresa već registrirana i predlaže opciju za oporavak lozinke.

UC2 - Prijava u sustav

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisniku prijavu u sustav kako bi pristupio svom profilu i dodatnim funkcionalnostima aplikacije.
- Sudionici: Korisnik, Baza podataka
- Preduvjet: Korisnik ima registriran i potvrđen račun.
- Opis osnovnog tijeka:
 1. Korisnik unosi svoje vjerodajnice (e-mail i lozinku) u obrazac za prijavu.
 2. Sustav provjerava vjerodajnice u bazi podataka.
 3. Ako su vjerodajnice ispravne, sustav prijavljuje korisnika i omogućuje mu pristup aplikaciji.
- Opis mogućih odstupanja:
 1. Korisnik unosi neispravne podatke.
 - Sustav prikazuje poruku o pogrešci i traži ponovno unošenje podataka.
 2. Račun korisnika nije aktiviran.
 - Sustav obavještava korisnika da se mora registrirati prije prijave.

UC3 - Pregled osnovnih informacija o aplikaciji

- Glavni sudionik: Registrirani korisnik

- Cilj: Omogućiti korisnicima pregled osnovnih informacija o aplikaciji kako bi razumjeli njezinu svrhu, funkcionalnosti i prednosti.
- Sudionici: Korisnik
- Preduvjet: Korisnik je na glavnoj stranici aplikacije.
- Opis osnovnog tijeka:
 1. Korisnik otvara stranicu s informacijama o aplikaciji.
 2. Sustav prikazuje osnovne informacije o aplikaciji, njezinim funkcionalnostima, prednostima te načinima korištenja.
 3. Korisnik pregledava informacije i dobiva osnovno razumijevanje funkcionalnosti i svrhe aplikacije.

UC4 - Potvrda registracije

- Glavni sudionik: Baza podataka
- Cilj: Omogućiti aktivaciju korisničkog računa nakon registracije.
- Sudionici: Baza podataka
- Preduvjet: Korisnik je ispravno popunio obrazac za registraciju.
- Opis osnovnog tijeka:
 1. U bazu podataka se spremaju svi podatci korisnika iz obrasca za registraciju.
 2. Aktivacija korisničkog računa.
 3. Slanje potvrde o registraciji na e-mail korisnika.
 4. Korisnik sada može pristupiti aplikaciji.
- Opis mogućih odstupanja:
 1. Podatci se nisu uspješno spremili u bazu podataka.

Sustav obavještava korisnika o pogrešci te zahtjeva ponovno ispunjavanje obrasca za registraciju.

UC5 - Pregled sadržaja

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisniku pregled dostupnog sadržaja na platformi (npr. članci, videozapisi).
- Sudionici: Korisnik, Baza podataka
- Preduvjet: Korisnik je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik otvara stranicu sa sadržajem.
 2. Sustav dohvaća dostupni sadržaj iz baze podataka i prikazuje ga korisniku.
 3. Korisnik pregledava sadržaj, može odabrat određene članke ili videozapise za detaljnije gledanje.
- Opis mogućih odstupanja:
 1. Sadržaj nije dostupan (npr. zbog mrežne pogreške).

Sustav obavještava korisnika o pogrešci i preporučuje pokušaj kasnije.

UC6 - Komunikacija korisnika

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima da međusobno komuniciraju putem privatnih poruka.
- Sudionici: Korisnik, Baza podataka
- Preduvjet: Korisnik je prijavljen i ima pristup funkcionalnosti za komunikaciju.
- Opis osnovnog tijeka:
 1. Korisnik otvara komunikacijski kanal.
 2. Korisnik unosi poruku te ju šalje.
 3. Sustav pohranjuje poruku u bazu podataka i prikazuje ga primatelju.
- Opis mogućih odstupanja:
 1. Korisnik pokušava poslati poruku neadekvatnog sadržaja.
Sustav prikazuje upozorenje o neprimjerenom sadržaju i blokira slanje poruke.
 2. Mrežna pogreška onemogućuje slanje poruke.
Sustav obavještava korisnika i traži ponovni pokušaj.

UC7 - Ocjenjivanje sadržaja

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima da ocijene sadržaj i time izraze svoje mišljenje ili korisničko iskustvo.
- Sudionici: Korisnik, Baza podataka
- Preduvjet: Korisnik je prijavljen u sustav te je pregledao sadržaj koji želi ocijeniti.
- Opis osnovnog tijeka:
 1. Korisnik ocjenjuje sadržaj oznakom "koristan".
 2. Sustav bilježi ocjenu u bazi podataka.
 3. Sustav uvećava broj ocjena na odabranom sadržaju.
 4. Sustav prikazuje ažurirani broj ocjena kraj odabranog sadržaja.
- Opis mogućih odstupanja:
 1. Korisnik je već ocijenio sadržaj.
Sustav blokira višestruko ocjenjivanje i prikazuje obavijest da je ocjena već zabilježena.

UC8 - Odabir plana

- Glavni sudionik: Klijent
- Cilj: Omogućiti klijentu odabir besplatnog ili plaćenog plana za dodatne pogodnosti.
- Sudionici: Klijent, Baza podataka
- Preduvjet: Klijent je prijavljen i ima pristup stranici s planovima.
- Opis osnovnog tijeka:
 1. Klijent pregledava dostupne planove.
 2. Klijent odabire željeni plan i potvrđuje izbor.

3. Aktivira se sustav za naplatu.
4. U bazi podataka se bilježi odabrani plan kao i u korisničkom profilu.

UC9 - Ostavljanje komentara

- Glavni sudionik: Korisnik
- Cilj: Omogućiti korisnicima ostavljanje komentara na sadržaj.
- Sudionici: Korisnik, Baza podataka
- Preduvjet: Korisnik je prijavljen u sustav i ima pristup sadržaju koji omogućuje komentiranje.
- Opis osnovnog tijeka:
 1. Korisnik otvara objavu na kojoj želi ostaviti komentar.
 2. Sustav prikazuje polje za unos komentara.
 3. Korisnik unosi tekst komentara i potvrđuje unos.
 4. Sustav pohranjuje komentar u bazu podataka i prikazuje ga na stranici objave.
- Opis mogućih odstupanja:
 1. Korisnik odustaje od ostavljanja komentara.

Sustav ne pohranjuje unos i zatvara polje za unos komentara.

UC10 - Odabir plaćenog plana

- Glavni sudionik: Partner
- Cilj: Omogućiti partnerima odabir plaćenog plana za pristup dodatnim funkcionalnostima.
- Sudionici: Partner, Baza podataka
- Preduvjet: Partner je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Partner odabire opciju za odabir plaćenog plana.
 2. Sustav prikazuje različite dostupne planove.
 3. Partner odabire plan koji želi aktivirati.
 4. Sustav usmjerava partnera na stranicu za plaćanje.
 5. Sustav pohranjuje informacije o aktiviranom planu u bazu podataka i omogućava dodatne funkcionalnosti partneru.

UC11 - Kreiranje i dijeljenje objava

- Glavni sudionik: Partner
- Cilj: Omogućiti partnerima kreiranje i dijeljenje sadržaja s korisnicima.
- Sudionici: Partner, Baza podataka
- Preduvjet: Partner je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Partner odabire opciju za kreiranje nove objave.
 2. Sustav prikazuje formu za unos sadržaja objave.
 3. Partner unosi tekst, slike i/ili druge multimedijalne sadržaje.
 4. Partner potvrđuje objavu.
 5. Sustav pohranjuje objavu u bazu podataka i prikazuje je korisnicima koji imaju pristup.
- Opis mogućih odstupanja:
 1. Partner odustaje od kreiranja objave.

Sustav zatvara formu bez pohranjivanja sadržaja.

UC12 - Upravljanje vlastitim sadržajem

- Glavni sudionik: Partner
- Cilj: Omogućiti partnerima uređivanje, ažuriranje ili brisanje vlastitih objava.
- Sudionici: Partner, Baza podataka
- Preduvjet: Partner je prijavljen u sustav i ima objave koje može uređivati.
- Opis osnovnog tijeka:
 1. Partner otvara popis svojih objava.
 2. Partner odabire opciju za uređivanje ili brisanje objave.
 3. Ako partner odabere uređivanje, sustav prikazuje formu za izmjenu sadržaja.
 4. Partner unosi izmjene i potvrđuje ih.
 5. Sustav pohranjuje izmjene u bazu podataka.
- Opis mogućih odstupanja:
 1. Partner odustaje od izmjena.

Sustav ne pohranjuje promjene i zatvara formu za uređivanje.

UC13 - Pregled svakog sadržaja

- Glavni sudionik: Admin
- Cilj: Omogućiti administratorima pregled svih objava i interakcija na platformi.
- Sudionici: Admin, Baza podataka
- Preduvjet: Admin je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Admin odabire opciju za pregled sadržaja.
 2. Sustav prikazuje popis svih objava i korisničkih interakcija.
 3. Admin pregledava sadržaj prema potrebama.

UC14 - Uređivanje svakog sadržaja

- Glavni sudionik: Admin
- Cilj: Omogućiti administratorima uređivanje sadržaja.
- Sudionici: Admin, Baza podataka
- Preduvjet: Admin je prijavljen u sustav i pregledava sadržaj koji želi urediti.
- Opis osnovnog tijeka:
 1. Admin odabire sadržaj koji želi urediti.
 2. Sustav prikazuje opcije za uređivanje sadržaja.
 3. Admin unosi i potvrđuje izmjene.
 4. Sustav ažurira sadržaj u bazi podataka.
 5. Sustav šalje partneru obavijest o uređivanju njegovog sadržaja.
- Opis mogućih odstupanja:
 1. Admin odustaje od uređivanja.

Sustav prekida postupak uređivanja i zadržava postojeće stanje sadržaja.

UC15 - Brisanje svakog sadržaja

- Glavni sudionik: Admin
- Cilj: Omogućiti administratorima trajno brisanje neprimjerenog sadržaja.

- Sudionici: Admin, Baza podataka
- Preduvjet: Admin je prijavljen u sustav i pregledava sadržaj koji želi ukloniti.
- Opis osnovnog tijeka:
 1. Admin odabire sadržaj koji želi obrisati.
 2. Sustav prikazuje potvrdu o trajnom brisanju.
 3. Admin potvrđuje brisanje.
 4. Sustav uklanja sadržaj iz baze podataka.
 5. Sustav šalje partneru obavijest o brisanju njegovog sadržaja.
- Opis mogućih odstupanja:
 1. Admin odustaje od brisanja.
 2. Sustav ne briše sadržaj i zadržava ga u bazi podataka.

UC16 - Upravljanje sustavom rangiranja

- Glavni sudionik: Admin
- Cilj: Omogućiti administratorima kontrolu nad rangiranjem partnera u sustavu.
- Sudionici: Admin, Baza podataka
- Preduvjet: Admin je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Admin otvara stranicu za upravljanje rangovima.
 2. Sustav prikazuje trenutačne rangove i partnere kojima su dodijeljeni.
 3. Admin može dodati, urediti ili ukloniti rang.
 4. Sustav pohranjuje promjene ranga u bazu podataka.
 5. Sustav šalje partneru obavijest o promjeni njegovog ranga.
- Opis mogućih odstupanja:
 1. Admin odustaje od izmjene ranga.
 2. Sustav ne primjenjuje promjene i zadržava postojeće stanje.

UC17 - Izmjena osobnih podataka

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima ažuriranje osobnih informacija.
- Sudionici: Korisnik, Baza podataka
- Preduvjet: Korisnik je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik otvara profil i odabire opciju za izmjenu podataka.
 2. Sustav prikazuje formu s trenutačnim podacima.
 3. Korisnik unosi nove informacije i potvrđuje izmjene.
 4. Sustav pohranjuje izmjene u bazu podataka.
- Opis mogućih odstupanja:
 1. Korisnik odustaje od izmjena.
 2. Sustav zadržava stare podatke i ne primjenjuje promjene.

UC18 - Pristup podršci

- Glavni sudionik: Registrirani korisnik
- Cilj: Omogućiti korisnicima kontaktiranje podrške za pomoć.
- Sudionici: Korisnik, Admin
- Preduvjet: Korisnik je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Korisnik otvara opciju za kontakt s podrškom.
 2. Sustav prikazuje formu za unos pitanja ili problema.
 3. Korisnik unosi upit i šalje ga timu podrške.
 4. Admin prima upit i odgovara korisniku.
- Opis mogućih odstupanja:
 1. Korisnik odustaje od slanja upita.
Sustav zatvara formu bez slanja poruke.

UC19 - Primanje obavijesti

- Glavni sudionik: Partner
- Cilj: Omogućiti partneru primanje obavijesti vezanih uz aplikaciju, transakcije i druge relevantne događaje.
- Sudionici: Partner, Baza podataka
- Preduvjet: Partner je prijavljen u sustav.
- Opis osnovnog tijeka:
 1. Partner dobiva obavijest o novom događaju u aplikaciji.
 2. Sustav prikazuje sadržaj obavijesti partneru.
 3. Partner potvrđuje primetak obavijesti (npr. označava je kao pročitanu).
 4. Sustav ažurira status obavijesti u bazi podataka.
- Opis mogućih odstupanja:
 1. Obavijest nije dostupna.
Sustav prikazuje poruku o pogrešci. 2. Greška u dostavi obavijesti. Ž Sustav će pokušati ponovno poslati obavijest kasnije.

UC20 - Isplaćivanje od aplikacije

- Glavni sudionik: Partner
- Cilj: Omogućiti partneru da primi isplatu za pružene usluge ili sadržaj.
- Sudionici: Partner, Baza podataka
- Preduvjet: Partner ima ostvarenu zaradu u aplikaciji.
- Opis osnovnog tijeka:
 1. Partner zahtijeva isplatu zarađenih sredstava.
 2. Sustav provjerava raspoloživa sredstva partnera.
 3. Sustav inicira proces transakcije za isplatu.
 4. Isplata se pohranjuje u bazi podataka.
 5. Partner prima potvrdu o uspješnoj isplati.
- Opis mogućih odstupanja:
 1. Nedovoljna sredstva.

Ako partner nema dovoljno sredstava za isplatu, sustav prikazuje obavijest o nedostupnim sredstvima.

2. Greška u transakciji.

Ako dođe do greške tijekom transakcije, sustav će obavijestiti partnera i pokušati ponovno.

UC21 - Uplate za pristup rangu

- Glavni sudionik: Korisnik
- Cilj: Omogućiti korisnicima uplatu za pristup određenim rangovima sadržaja.
- Sudionici: Partner, Baza podataka
- Preduvjet: Korisnik ima aktivan profil i želi pristupiti sadržaju u određenom rangu.
- Opis osnovnog tijeka:
 1. Sustav prikazuje cijenu i uvjete pristupa odabranom rangu.
 2. Korisnik potvrđuje uplatu.
 3. Sustav inicira proces transakcije i pohranjuje je u bazi podataka.
 4. Korisnik prima potvrdu o uspješnoj uplati i dobiva pristup odabranom rangu.
 5. opis korak pet
- Opis mogućih odstupanja:
 1. Neuspjela uplata.

Sustav će prikazati poruku o grešci i ponuditi ponovno pokušavanje. Greška u transakciji. Korisnik će biti obaviješten i transakcija će se pokušati ponovno.

UC22 - Pohrana transakcije

- Glavni sudionik: Baza podataka
- Cilj: Osigurati evidenciju svih transakcija izvršenih u sustavu radi sigurnosti i praćenja finansijskih tokova.
- Sudionici: Korisnik, Baza podataka
- Preduvjet: Transakcija je izvršena (uplata, isplata ili druga finansijska aktivnost).
- Opis osnovnog tijeka:
 1. Sustav prima zahtjev za pohranu podataka o novoj transakciji.
 2. Podaci o transakciji (iznos, datum, sudionici) bilježe se u bazi podataka.
 3. Baza podataka potvrđuje uspješnu pohranu transakcije.
 4. Sudionici transakcije dobivaju obavijest o pohranjenoj transakciji.
- Opis mogućih odstupanja:
 1. Greška u pohrani podataka.

Sustav će pokušati ponovno i obavijestiti administratora ako problem potraje.

UC23 - Autentifikacija plaćanja

- Glavni sudionik: Banka
- Cilj: Osigurati da je plaćanje autorizirao ovlašteni korisnik.
- Sudionici: Korisnik, Banka, Sustav za autentifikaciju
- Preduvjet: Korisnik započinje proces plaćanja.
- Opis osnovnog tijeka:
 1. Korisnik unosi potrebne podatke za autentifikaciju (npr. PIN, lozinka, jednokratni kod, biometrija) putem korisničkog sučelja banke.
 2. Banka provjerava unesene podatke s postojećim zapisima i pravilima sigurnosti.
 3. Ako su podaci ispravni, banka odobrava autentifikaciju i obavještava sustav o uspjehu.
 4. Sustav bilježi uspješnu autentifikaciju i omogućuje nastavak plaćanja.
 5. Sustav bilježi uspješnu autentifikaciju i omogućuje nastavak plaćanja.
 6. Korisnik dobiva obavijest o uspješno autentificiranom plaćanju

- Opis mogućih odstupanja:

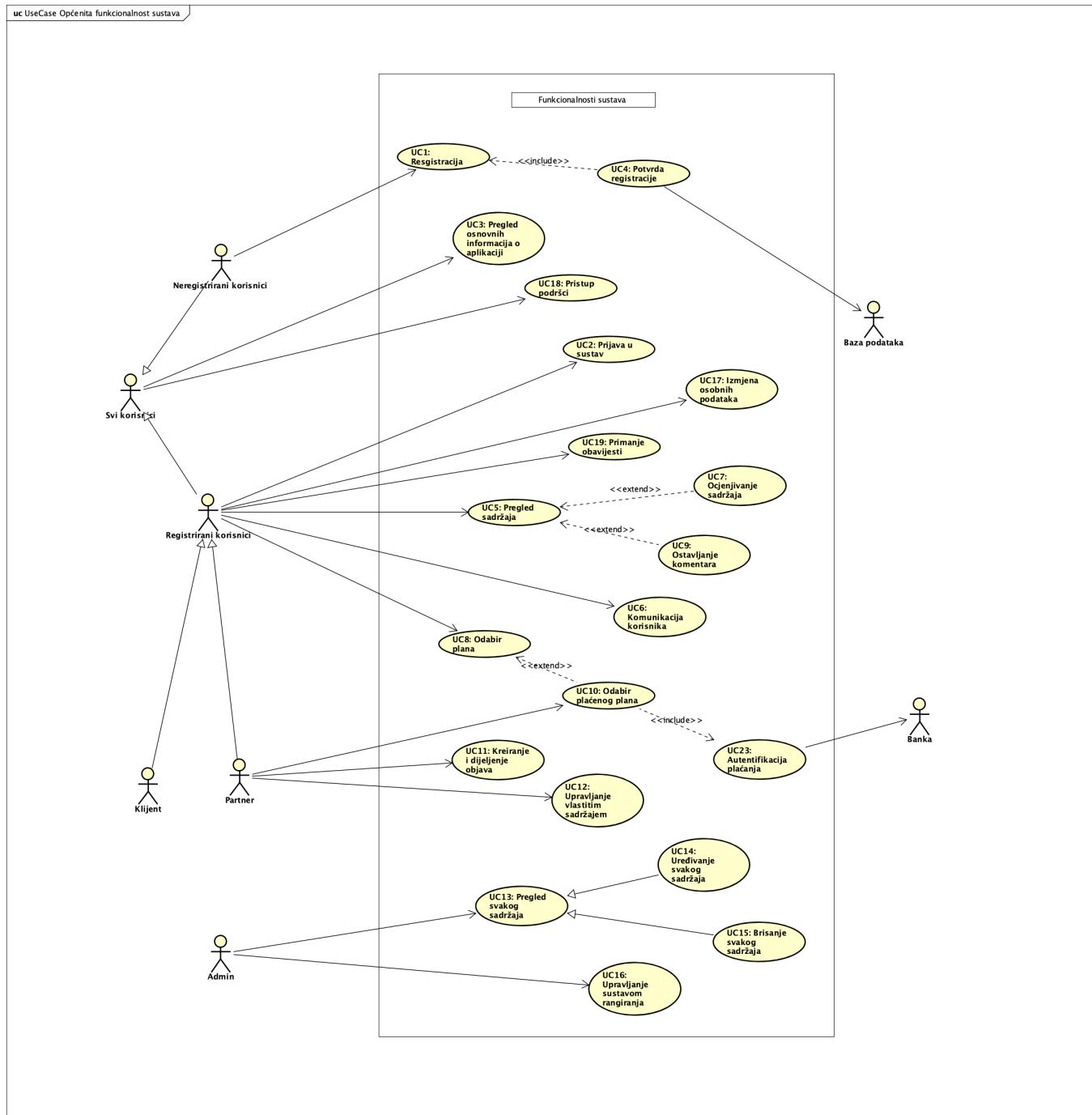
Neispravni podaci za autentifikaciju.

Banka obavještava sustav o neuspješnoj autentifikaciji, a sustav prenosi informaciju korisniku. Korisnik može ponovno unijeti podatke.

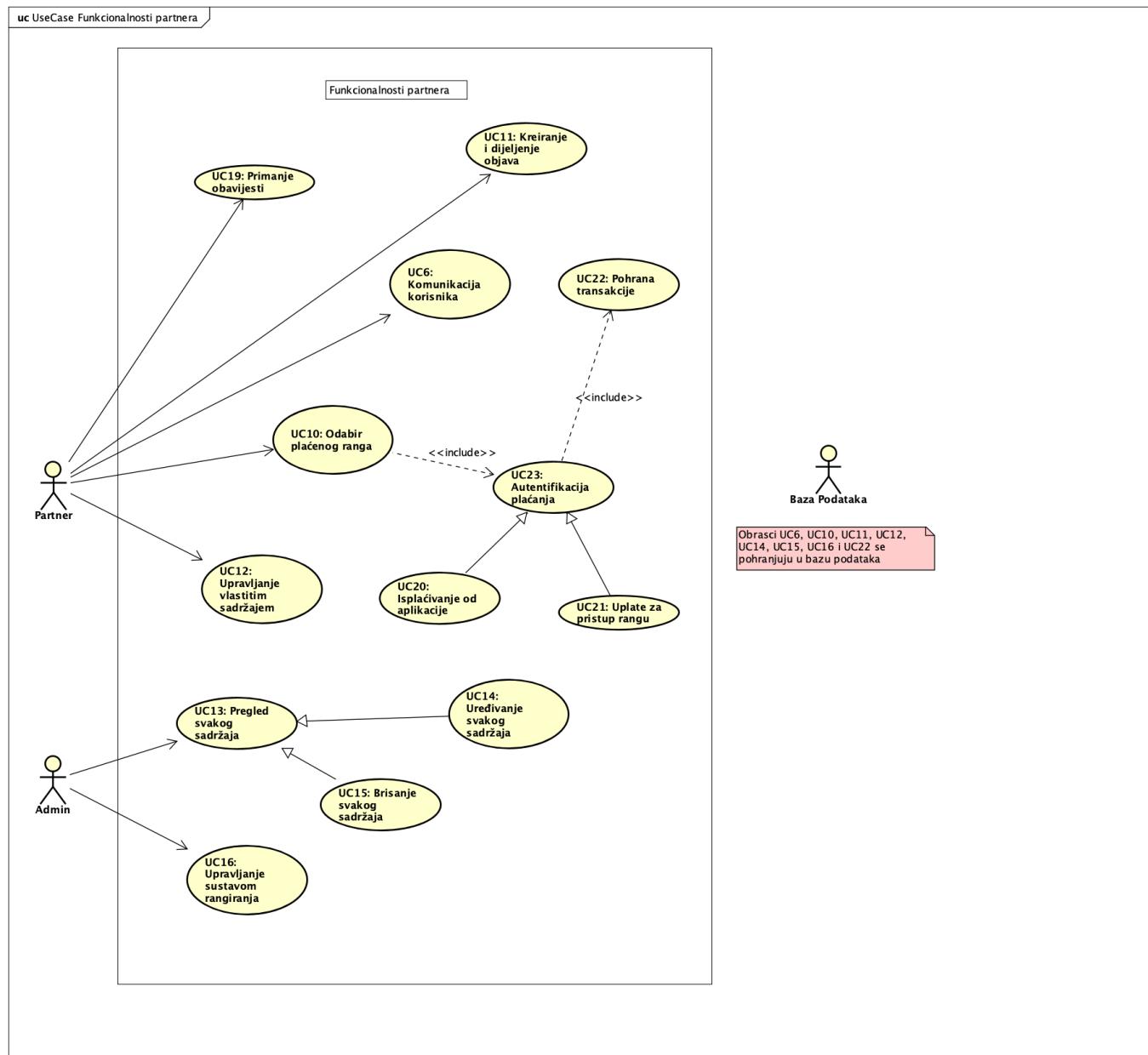
Tehnički problem s komunikacijom.

Sustav obavještava korisnika o nemogućnosti autentifikacije zbog tehničkih poteškoća. Sustav automatski ponavlja zahtjev ili omogućuje korisniku da pokuša ponovno kasnije.

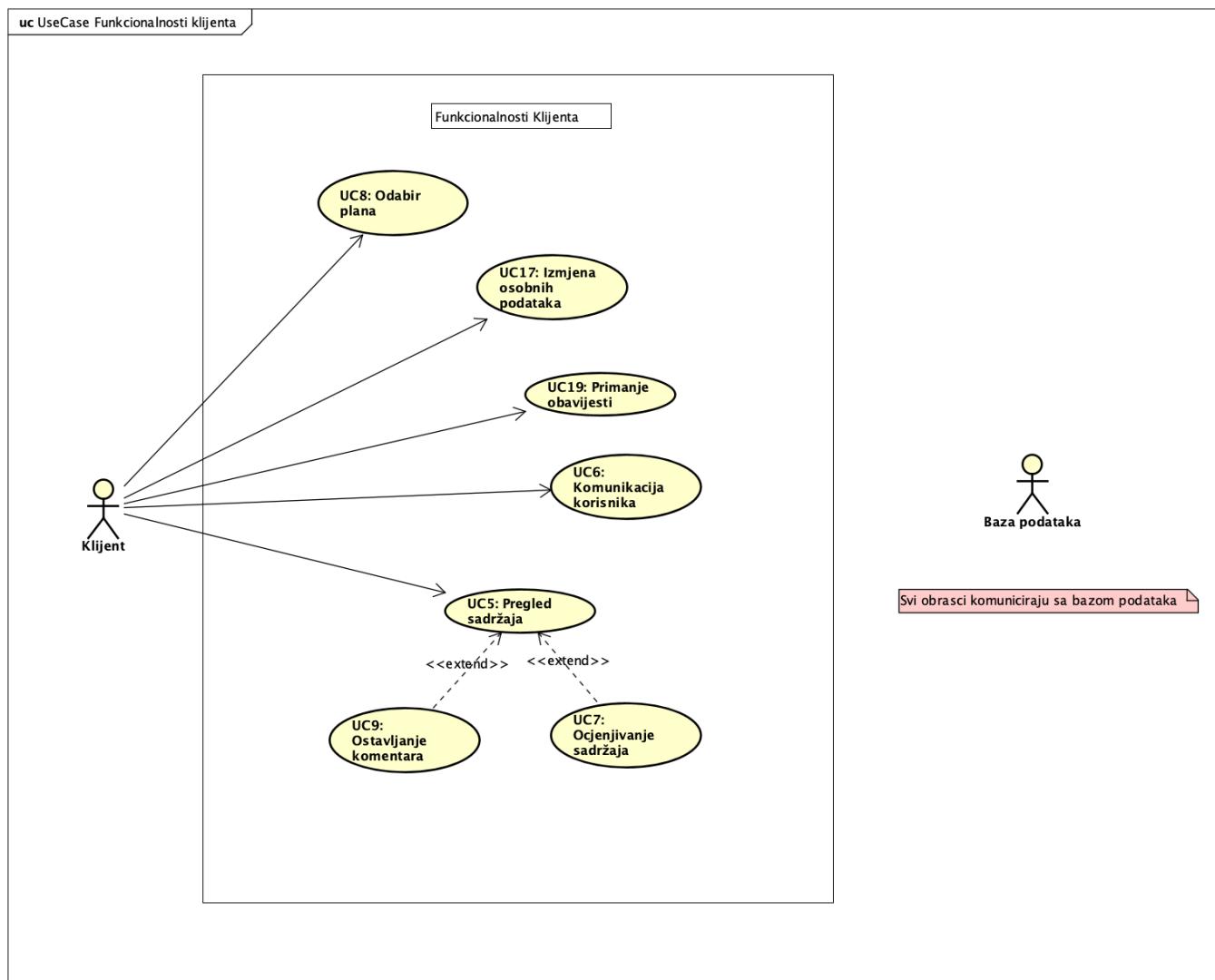
Dijagrami obrazaca uporabe



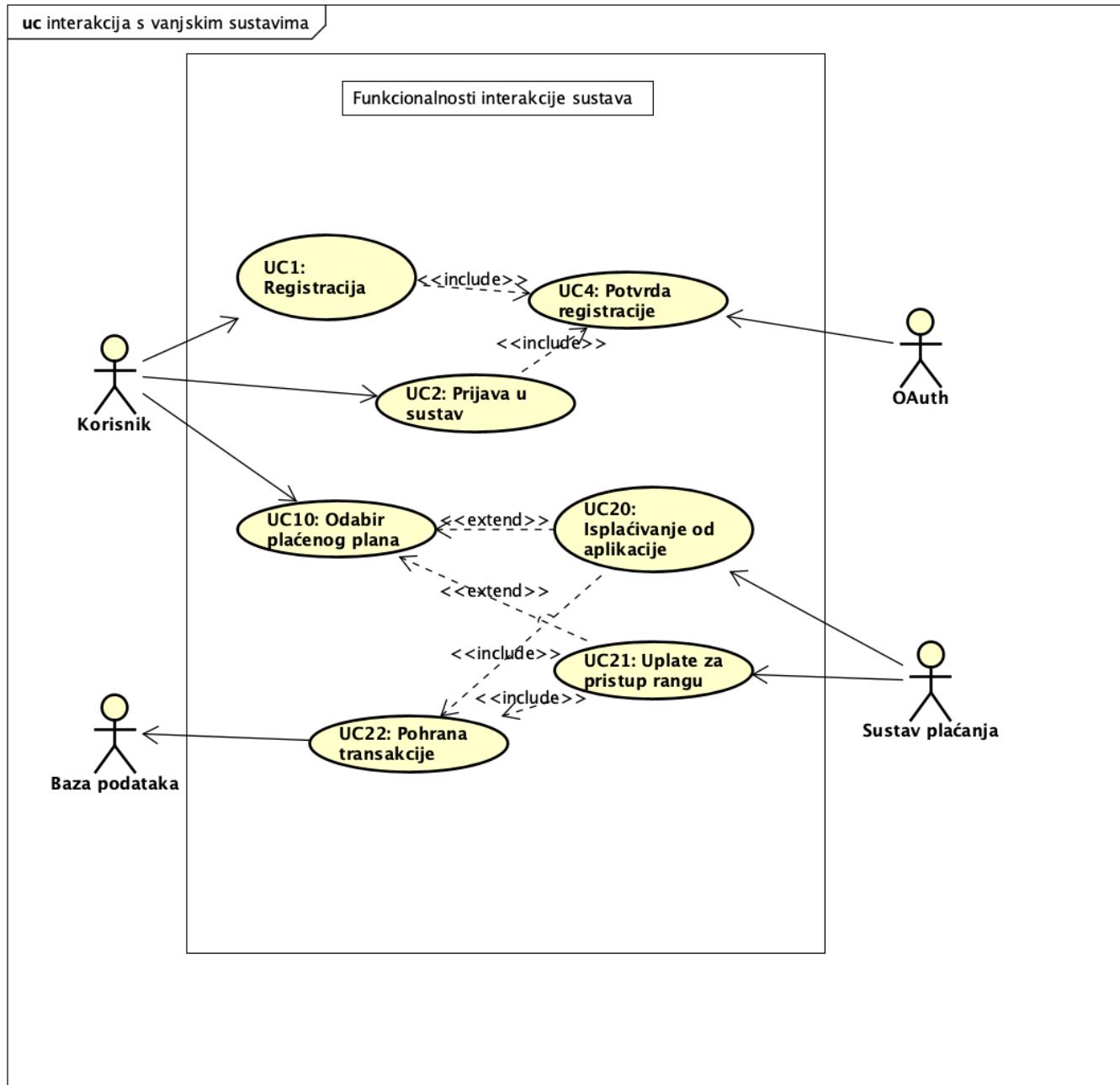
Dijagram obrasca uporabe, funkcionalnosti registriranih i neregistriranih korisnika, njihova podjela i funkcionalnosti admina.



Dijagram obrasca uporabe, funkcionalnosti partnera i njegova povezanost s bazom podataka te detaljan prikaz mogućnosti admina.

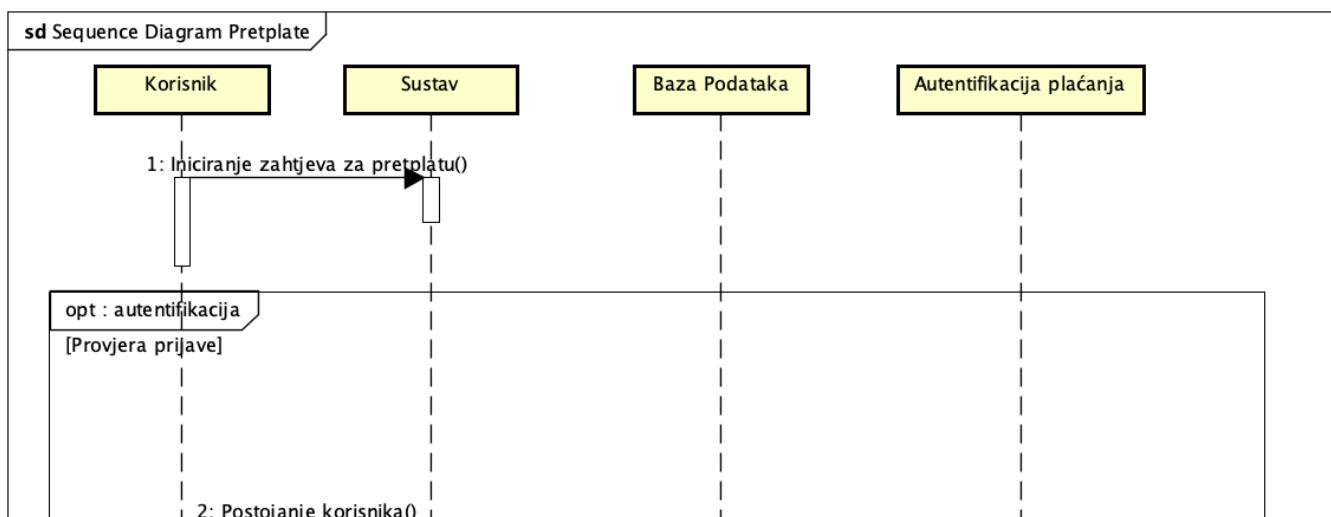


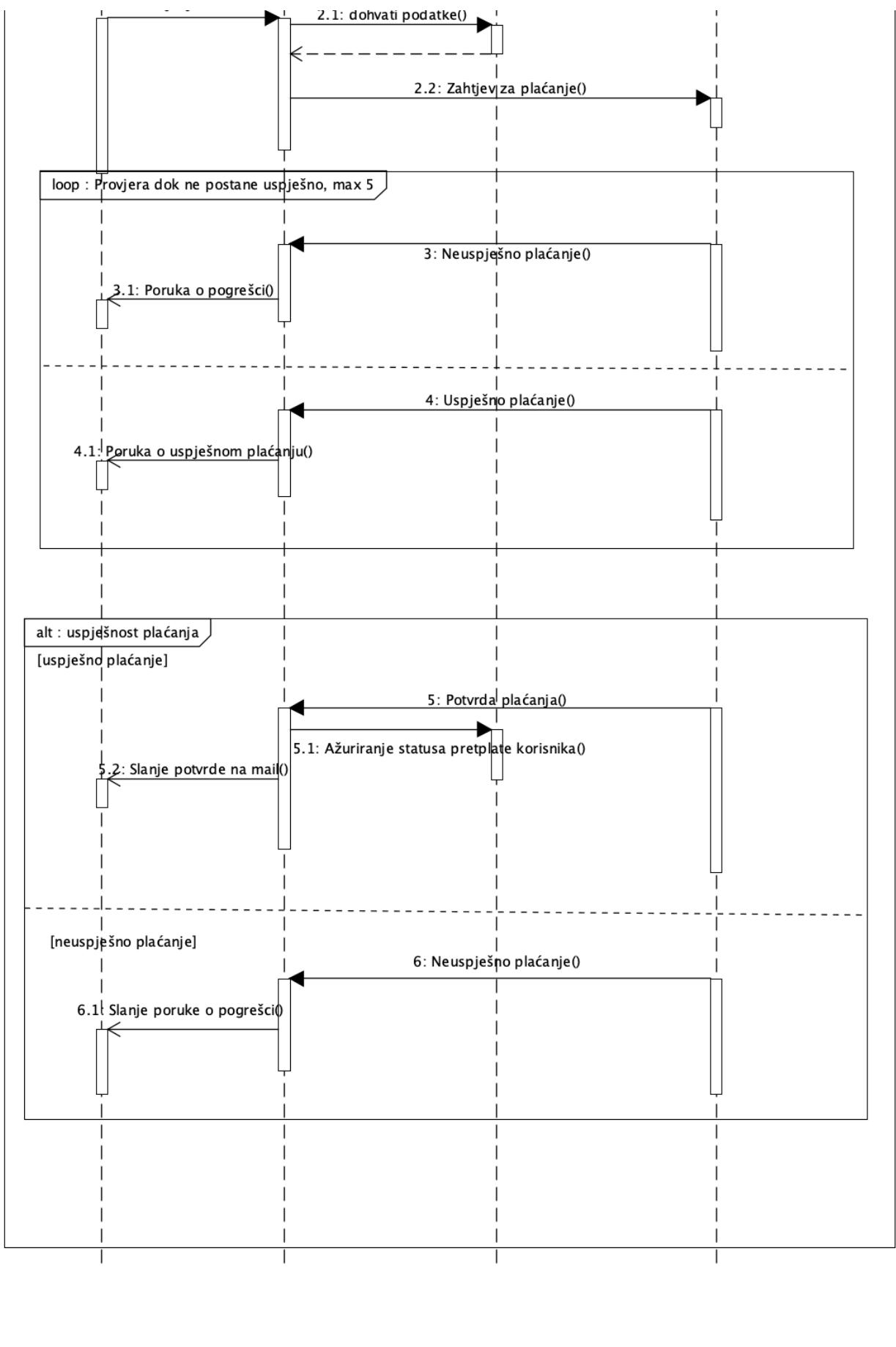
Dijagram obrasca uporabe, funkcionalnosti klijenta.



Dijagram obrazaca uporabe za kritične sustave i integracije

Sekvencijski dijagramи



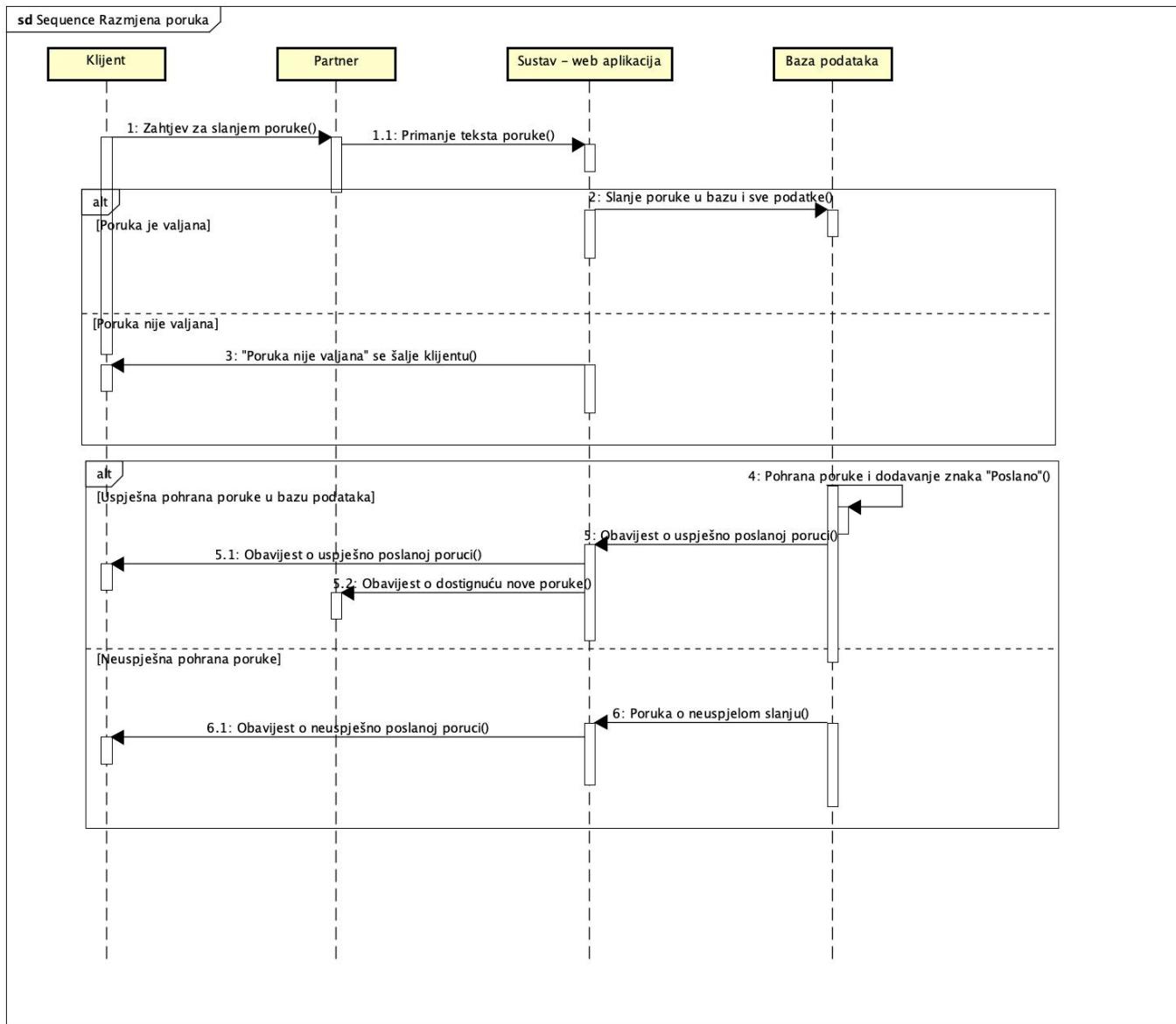


Sekvenčni dijagram, kako funkcioniра pretplata

Korišteni su obrasci uporabe UC10 - Odabir plaćenog plana, UC21 - Update za pristup rangu, i UC22 - Pohrana transakcije.

Korisnik pokreće zahtjev za pretplatom te sustav provjerava korisnikov status prijave. Ako je korisnik uspješno prijavljen, sustav pristupa bazi podataka kako bi provjerio postojanje korisničkih podataka i dohvata ih. Nakon potvrde postojanja korisnika, sustav inicira zahtjev za plaćanje prema usluzi za autentifikaciju plaćanja.

U slučaju uspješnog plaćanja, sustav ažurira status korisnikove pretplate u bazi podataka, pohranjuje transakciju te šalje korisniku potvrdu o uspješnoj preplati putem e-pošte. Međutim, ako plaćanje nije uspješno, sustav prikazuje korisniku poruku o pogrešci te omogućava ponovno pokretanje procesa plaćanja. Proces se ponavlja dok ne dođe do uspješnog plaćanja ili dok korisnik ne odluči odustati. Ako plaćanje opetovano ne uspije, sustav će korisniku prikazati konačnu obavijest o nemogućnosti ostvarivanja pretplate.



Sekvenčni dijagram, kako funkcioniра slanje poruka (u primjeru klijent šalje partneru poruku)

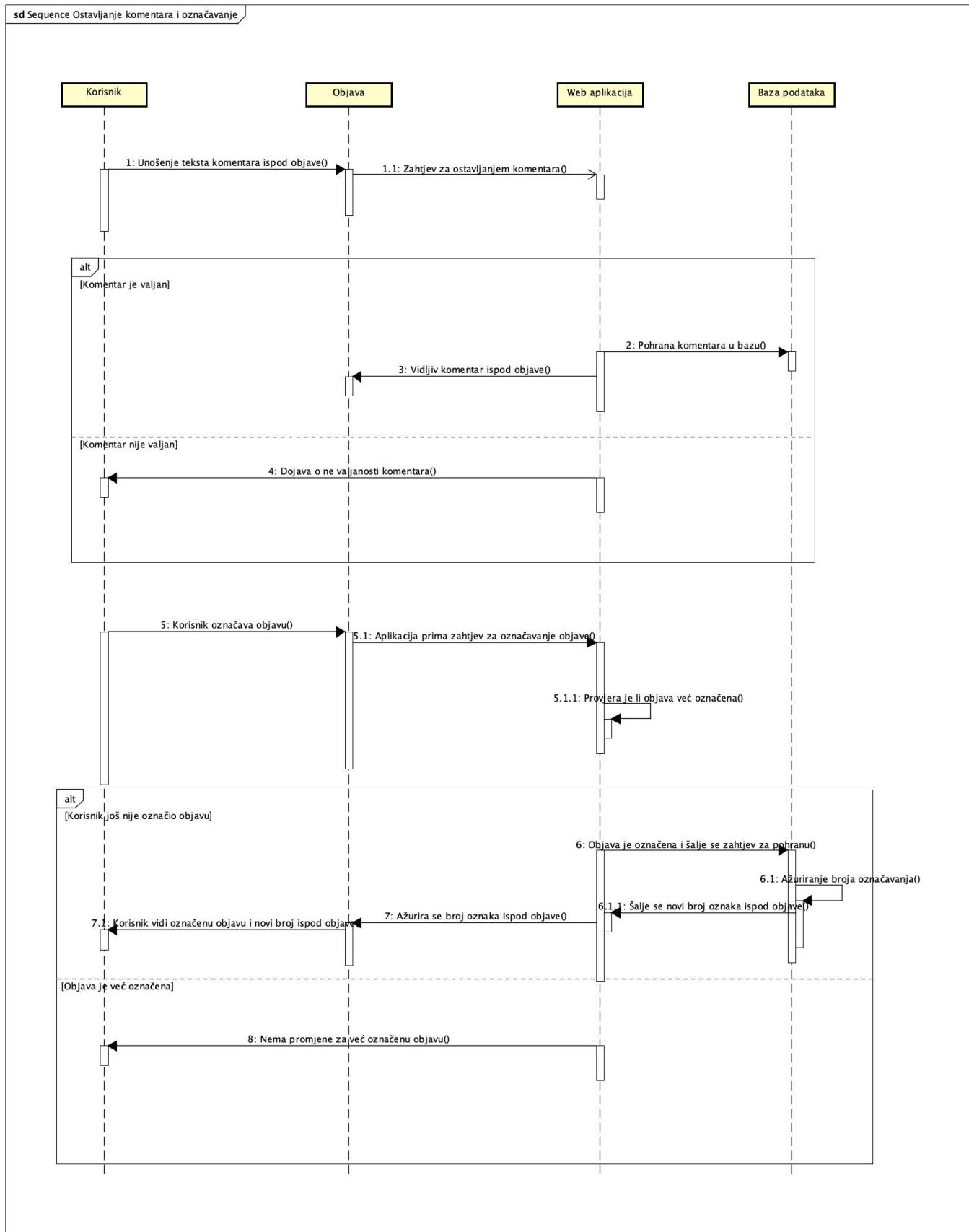
Korišteni su obrasci uporabe UC6 - Komunikacija korisnika, UC13 - Pregled svakog sadržaja.

Klijent započinje proces slanja poruke prema partneru putem web aplikacije. Web aplikacija prihvata tekst poruke i šalje ga na pohranu u bazu podataka, gdje se spremaju podaci poruke.

Sustav zatim provjerava ispravnost poruke. Ako je poruka valjana, vraća je u sustav s oznakom "Poslano" te šalje obavijest partneru o uspješno primljenoj poruci. Sustav tada obavještava klijenta o uspješnom slanju poruke, a partner dobiva obavijest o dolasku nove poruke.

Ako poruka nije valjana, sustav šalje poruku o pogrešci klijentu, navodeći da poruka nije uspješno poslana. U slučaju neuspješne pohrane poruke, klijent također prima obavijest o neuspjehu slanja.

Ovaj proces omogućava klijentu da komunicira s partnerom putem web aplikacije, a sve poruke se pohranjuju u bazu podataka kako bi bile dostupne za pregled i daljnje upravljanje.



Sekvenčni dijagram, funkcionalnost komentiranja i označavanja objava

Korišteni su obrasci uporabe UC9 - Ostavljanje komentara, UC13 - Pregled svakog sadržaja, i UC14 - Označavanje sadržaja.

Korisnik unosi tekst komentara ispod objave koju želi komentirati i šalje zahtjev za ostavljanjem komentara putem web aplikacije. Aplikacija zatim proslijeđuje zahtjev za pohranu komentara u bazu podataka.

Ako je komentar valjan, baza podataka vraća potvrdu te se komentar prikazuje ispod objave, vidljiv svim korisnicima. U slučaju nevaljanog komentara, korisnik dobiva obavijest o pogrešci te komentar nije objavljen.

Korisnik može zatražiti označavanje objave. Web aplikacija prima zahtjev za označavanje objave te provjerava je li objava već označena. Ako objava nije prethodno označena, sustav ažurira broj oznaka i šalje korisniku potvrdu o označavanju objave. U suprotnom se ne događa promjena.

Ovaj proces omogućava korisnicima interakciju s objavama kroz komentare i označavanje.

Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

Obrazac uporabe	Ključne funkcionalnosti
UC1 - Registracija	Stvaranje profila/prijava
UC2 - Prijava u sustav	Stvaranje profila/prijava
UC3 - Pregled osnovnih informacija o aplikaciji	Pristup informacija same aplikacije
UC4 - Potvrda registracije	Pohrana korisničkih podataka
UC5 - Pregled sadržaja	Spremanje i označavanje objava
UC6 - Komunikacija korisnika	Komunikacija s ostalim korisnicima
UC7 - Ocjenjivanje sadržaja	Spremanje i označavanje objava
UC8 - Odabir plana	Odabiranje i otkazivanje besplatnog ranga/ranga za plaćanje
UC9 - Ostavljanje komentara	Spremanje i označavanje objava
UC10 - Odabir plaćenog plana	Odabiranje i otkazivanje ranga za plaćanje
UC11 - Kreiranje i dijeljenje objava	kreiranje i uređivanje plaćenih/besplatnih objava
UC12 - Upravljanje vlastitim sadržajem	kreiranje i uređivanje plaćenih/besplatnih objava
UC13 - Pregled svakog sadržaja	Upravljanje i kontrola sadržaja koji se objavljuje
UC14 - Uređivanje svakog sadržaja	Upravljanje i kontrola sadržaja koji se objavljuje
UC15 - Brisanje svakog sadržaja	Adminsko brisanje sadržaja
UC16 - Upravljanje sustavom rangiranja	Adminsko filtriranje rangova
UC17 - Izmjena osobnih podataka	Pohrana korisničkih podataka

Obrazac uporabe	Ključne funkcionalnosti
UC18 - Pristup podršci	Pristup informacija same aplikacije
UC19 - Primanje obavijesti	Primanje obavijesti
UC20 - Isplaćivanje od aplikacije	Isplate
UC21 - Uplate za pristup rangu	Uplate
UC22 - Pohrana transakcije	Evidencija povijesti transakcija

Arhitektura sustava

Opis arhitekture

Stil arhitekture

Za SportConnect je odabran mikrouslužni stil arhitekture zbog nekoliko ključnih razloga:

- Fleksibilnost i skalabilnost: Mikrouslužna arhitektura omogućava razvoj sustava kao skupa manjih, neovisnih usluga koje mogu biti samostalno razvijene, testirane i implementirane. Svaka mikrousluga odgovara na specifičnu poslovnu funkciju (npr. upravljanje korisničkim profilima, sustav rangiranja, plaćanja itd.), što omogućava lakše skaliranje sustava prema potrebama korisnika i tržišta.
- Lakoća održavanja i nadogradnje: Mikrouslužna arhitektura omogućava razvoj i nadogradnje pojedinih komponenti sustava bez potrebe za ponovnim implementiranjem cijelog sustava, što smanjuje rizik i vrijeme potrebno za nove funkcionalnosti.
- Tehnološka neovisnost: Svaka mikrousluga može biti razvijena koristeći najprikladniju tehnologiju za specifičnu funkcionalnost (npr. React za frontend, Spring Boot za backend, PostgreSQL za bazu podataka), omogućujući veću fleksibilnost u odabiru tehnologije.

Podsistavi

SportConnect sustav može se podijeliti na nekoliko ključnih podsustava, svaki s jasno definiranim odgovornostima:

- Korisnički profil i autentifikacija: Ovaj podsustav upravlja registracijom, prijavom korisnika te autentifikacijom putem protokola OAuth 2.0 za sigurnost. Također omogućava korisnicima upravljanje svojim profilima, rangovima i postavkama.
- Sustav rangiranja: Ovaj podsustav odgovoran je za praćenje kvalitete sadržaja i angažmana korisnika te upravljanje statusima rangova (brončani, srebrni, zlatni). Temelji se na ocjenama korisnika i povratnim informacijama koje utječu na rangiranje partnera.
- Komunikacija i mentorstvo: Ovaj podsustav omogućava interakciju između klijenata i partnera. Omogućuje slanje poruka i dijeljenje sadržaja, uključujući planove i savjete.
- Plaćanje: Ovdje se upravlja svim transakcijama unutar aplikacije, uključujući pretplate i uplatu, s naglaskom na sigurnost finansijskih podataka.

Spremišta podataka

Za pohranu podataka bit će korištena relacijska baza podataka (PostgreSQL) zbog potrebe za kompleksnim upitima, referencijalnim integritetom i transakcijama. PostgreSQL omogućuje pouzdanu pohranu podataka o korisnicima, transakcijama, objavama, rangovima i povijesti interakcija.

Mrežni protokoli

Za komunikaciju između komponenti sustava i krajnjih korisnika odabran je HTTP/HTTPS za standardnu web komunikaciju između klijenta i poslužitelja.

Globalni upravljački tok

Podaci i informacije cirkuliraju kroz sustav kako slijedi:

1. Korisnik se prijavljuje putem aplikacije koristeći OAuth 2.0.
2. Korisnik pristupa sadržaju temeljenom na svom rangiranju.
3. Komunicira s partnerima, postavljajući pitanja ili reagiranjem na objave.
4. Sustav rangiranja ažurira status partnera na temelju korisničkih povratnih informacija.
5. Za financijske transakcije koristi se sustav za sigurnu uplatu i isplatu.
6. Podaci o korisnicima, ocjenama i transakcijama pohranjuju se u bazu podataka.

Sklopovskoprogramske zahtjevi:

- Operativni sustavi: Aplikacija će podržavati glavne operativne sustave (Linux, Windows, macOS) na poslužiteljima i uređajima korisnika.
- Procesori i memorija: Sustav treba biti optimiziran za rad na cloud platformama, s podrškom za visoke performanse i veliku količinu istovremenih korisnika (scale-out).
- Sigurnost: Sigurnosni protokoli kao što je OAuth 2.0, HTTPS i enkripcija osigurat će zaštitu podataka i transakcija.

Obrazloženje odabira arhitekture

SportConnect koristi mikrouslužnu arhitekturu zbog njenih ključnih prednosti, uključujući skalabilnost, fleksibilnost, sigurnost i modularnost. Ova arhitektura omogućava odvojeno upravljanje i skaliranje svakog dijela sustava, čime se poboljšava učinkovitost i održivost.

Izbor arhitekture temeljen na principima oblikovanja

Mikrouslužna arhitektura je odabrana zbog visoke kohezije, niske povezanosti, fleksibilnosti i sigurnosti. Visoka kohezija omogućava organiziranje sličnih funkcionalnosti u zasebne usluge, dok niska povezanost omogućava da usluge rade neovisno jedna o drugoj. Fleksibilnost omogućuje lakše prilagodbe zahtjevima, dok sigurnost osigurava zaštitu korisničkih podataka kroz specijalizirane sigurnosne protokole.

Ovaj pristup omogućava jednostavno upravljanje i kontinuirano proširenje sustava prema novim funkcionalnostima bez utjecaja na druge komponente.

Kompromisi i izazovi

Iako mikrouslužna arhitektura donosi značajne prednosti, postoji i određena kompleksnost u upravljanju više mikrousluga. S obzirom na to, koristit ćemo orkestracijske alate poput Kubernetes-a za jednostavnije upravljanje i skaliranje usluga. Također, za osiguranje konzistentnosti podataka koristit ćemo eventualnu konzistentnost i sinkronizaciju pomoću asinkronih poruka.

Unatoč početnim izazovima, mikrouslužna arhitektura pruža dugoročnu fleksibilnost i sigurnost koja odgovara zahtjevima SportConnecta.

Organizacija sustava na visokoj razini

Organizacija sustava na visokoj razini za SportConnect temelji se na klijent-poslužitelj arhitekturi, s bazom podataka, datotečnim sustavom i grafičkim sučeljem kao ključnim komponentama.

Klijent-poslužitelj arhitektura

SportConnect sustav koristi klijent-poslužitelj arhitekturu, u kojoj su korisnici (klijenti) povezani s poslužiteljem putem internetske mreže. Klijent je aplikacija koja omogućava korisnicima (klijentima i partnerima) da pristupe funkcijama platforme, poput pregleda sadržaja, stvaranja profila, razmjene poruka i obavljanja financijskih transakcija. Poslužitelj je centralni sustav koji upravlja poslovnom logikom aplikacije, obradom podataka, autentifikacijom i pohranom podataka u bazu podataka. Klijent šalje zahtjeve prema poslužitelju (putem HTTP/HTTPS protokola), a poslužitelj odgovara s potrebnim podacima ili izvršavanjem akcija.

Baza podataka

Baza podataka u sustavu SportConnect koristi relacijsku bazu podataka (PostgreSQL). Baza pohranjuje sve korisničke podatke, informacije o partnerima, transakcijama, povijesti objava, rangovima korisnika i ocjenama. Relacijska baza podataka omogućuje učinkovito upravljanje povezanim podacima kroz tablice, omogućujući brzo pretraživanje i ažuriranje podataka. Na primjer, kada korisnik postavi ocjenu za sadržaj partnera, ta se informacija spremi u bazu podataka, a te može utjecati na rang partnera. Baza također osigurava integritet podataka i podržava sigurnosne mehanizme, poput šifriranja i zaštite podataka.

Datotečni sustav

SportConnect sustav može koristiti datotečni sustav za pohranu medijskog sadržaja, kao što su slike profila, video materijali ili dokumenti koji su povezani s korisnicima i partnerima. Datotečni sustav omogućava pohranu i pristup velikim datotekama koje se ne bi moglo učinkovito pohranjivati u relacijskim bazama podataka. Datoteke se pohranjuju na poslužitelju ili u vanjskoj usluzi za pohranu, a korisnici imaju pristup datotekama putem odgovarajućih API-ja integriranih u aplikaciju.

Grafičko sučelje

Grafičko sučelje SportConnect aplikacije razvija se kao web aplikacija koristeći React za frontend. Web sučelje omogućava korisnicima jednostavan pristup svim funkcijama aplikacije, kao što su stvaranje profila, pregled sadržaja prema rangovima, komunikacija s partnerima, i obavljanje uplata i isplata. Grafičko sučelje je povezano s backend poslužiteljem putem RESTful API-ja koji omogućava komunikaciju između frontend-a i backend-a. API pozivi omogućuju klijentima da dobiju potrebne podatke (poput sadržaja, korisničkih podataka, transakcija) i izvrše akcije (kao što su ocjenjivanje sadržaja, uplata).

Organizacija aplikacije

Frontend i Backend slojevi

Frontend i backend slojevi aplikacije SportConnect organizirani su kako bi osigurali jasno odvajanje odgovornosti i efikasnu komunikaciju:

- **Frontend sloj:**

- Implementiran u Reactu koristeći TypeScript za bolje upravljanje tipovima i lakše otkrivanje pogrešaka u razvoju.
- Odgovoran za korisničko sučelje, upravljanje sesijama i lokalno spremanje tokena za autentifikaciju (JWT) u LocalStorage.
- Koristi REST API-je za dohvaćanje podataka i izvršavanje operacija na backendu.
- Stiliziran pomoću biblioteka poput Bootstrap, ChakraUI i Tailwind CSS, koje omogućavaju responzivni dizajn i ugodno korisničko iskustvo.
- Real-time funkcionalnost, poput chata, implementirana je pomoću Pusher-a koji koristi WebSocket tehnologiju.

- **Backend sloj:**

- Implementiran u Spring Boot frameworku koristeći slojevitu arhitekturu (Model-Repository-Service-Controller).
- Odgovoran za poslovnu logiku, validaciju podataka, autentifikaciju i autorizaciju korisnika te komunikaciju s bazom podataka.
- Koristi Spring Data JPA i Hibernate za rad s bazom podataka.
- Omogućuje sigurnu obradu plaćanja putem Stripe webhook-a.

MVC arhitektura

Aplikacija koristi Model-View-Controller (MVC) arhitekturu kako bi se postiglo odvajanje logike korisničkog sučelja, poslovne logike i podataka:

- **Model:**

- Predstavlja entitete baze podataka, poput korisnika, objava i transakcija.
- Mapiranje modela na relacijsku bazu podataka provodi se pomoću Hibernate ORM-a.

- **View:**

- Frontend u Reactu predstavlja View sloj koji komunicira s backendom putem REST API-ja.
- Koristi dinamične komponente za prikaz podataka koje dohvata s backend-a.

- **Controller:**

- Backend kontroleri upravljaju HTTP zahtjevima koji dolaze s frontend-a, pozivaju odgovarajuće servise i vraćaju odgovore u JSON formatu.

Ovaj pristup omogućava jednostavnije održavanje i lakše dodavanje novih funkcionalnosti u sustav.

Dijagram visoke razine

Dijagram visoke razine prikazuje glavne podsustave i njihove međusobne odnose:

- **Frontend:** Komunicira s backend-om putem REST API-ja i obrađuje zahtjeve korisnika.
- **Backend:** Implementira poslovnu logiku i povezuje se s bazom podataka.
- **Baza podataka (PostgreSQL):** Spremište za korisničke podatke, objave, transakcije i real-time poruke.

- **Pusher:** Koristi se za real-time komunikaciju između korisnika.
- **Stripe webhook:** Obrada plaćanja i praćenje statusa transakcija.

Reference

- Architect modern web applications with ASP.NET core and azure - .NET, Architect modern web applications with ASP.NET Core and Azure - .NET | Microsoft Learn. Dostupno: <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/> (7. studeni 2024.).
- Fowler, M., & Lewis, J. (2014). Microservices. Dostupno: <https://martinfowler.com/articles/microservices.html>
- Newman, S. (2015). Building Microservices. O'Reilly Media.

Baza podataka

Za naš sustav odabrali smo relacijsku bazu podataka, zbog pogodnosti za strukturiranje podataka i osiguravanje integriteta. Relacijska baza omogućuje jasnu definiciju entiteta kao tablica, uz podršku za transakcije potrebne za operacije update i isplate. Naša baza podataka sastoji se od jedanaest entiteta:

- USER - pohranjuje podatke o korisnicima
- CLIENT - pohranjuje podatke specifične za klijente
- PARTNER - pohranjuje podatke specifične za partnere
- POST - pohranjuje objave koje korisnici stvaraju
- COMMENT - pohranjuje komentare na objave
- POST LIKES - bilježi lajkove objave korisnika
- POST SAVES - bilježe spremljene objave korisnika
- CONVERSATIONS - bilježi id razgovora
- CONVERSATION PARTICIPANTS - bilježi sudionike razgovora
- CONVERSATION USER - pohranjuje podatke o korisniku i razgovoru u kojem sudjeluje
- CONVERSATION READ STATUS - bilježi vrijeme čitanja poruke u razgovoru
- MESSAGES - pohranjuje podatke o posланој poruci

Primarni i strani ključevi povezuju podatke između tablica. Koristimo indekse radi bržeg dohvaćanja, a transakcijski model osigurava dosljednost. Backup i replikacija čuvaju dostupnost podataka, a pristup je reguliran prema razinama autorizacije.

Opis tablica

User

Atribut	Tip podatka	Opis varijable
user_id (PK)	INT	Jedinstveni identifikator korisnika
username	VARCHAR(50)	Korisničko ime
email	VARCHAR(100)	Adresa elektroničke pošte
mobile_number	VARCHAR(20)	Broj mobitela
password_hash	VARCHAR(255)	Hash lozinke

Atribut	Tip podatka	Opis varijable
is_banned	BOOLEAN	Aktivan/Suspendiran korisnik
date_joined	DATE	Datum registracije
profile_picture	VARCHAR(255)	URL profilne slike
first_name	VARCHAR(50)	Ime
last_name	VARCHAR(50)	Prezime
subscription_plan	VARCHAR(50)	Vrsta pretplate

Client

Atribut	Tip podatka	Opis varijable
user_id (PK)(FK)	INT	Jedinstveni identifikator korisnika

Partner

Atribut	Tip podatka	Opis varijable
user_id (PK)(FK)	INT	Jedinstveni identifikator korisnika
rank	VARCHAR(50)	Rang (Bronze/Silver/Gold)
approved	BOOLEAN	(Nije)Odobrio admin
bio	VARCHAR(255)	Opis profila

Post

Atribut	Tip podatka	Opis varijable
post_id (PK)	INT	Jedinstveni identifikator objave
image_url	VARCHAR(255)	URL slike
text_content	VARCHAR(255)	Sadržaj
created_at	TIMESTAMP	Datum i vrijeme kreiranja objave
like_count	INT	Broj oznaka 'Useful'
post_tier	VARCHAR(50)	Rang objave
user_id (FK)	INT	Jedinstveni identifikator korisnika (partner) koji napravi objavu

Post Likes

Atribut	Tip podatka	Opis varijable
---------	-------------	----------------

Atribut	Tip podatka	Opis varijable
user_id (PK)(FK)	INT	Jedinstveni identifikator korisnika koji označava objavu
post_id (PK)(FK)	INT	Jedinstveni identifikator označene objave

Post Saves

Atribut	Tip podatka	Opis varijable
user_id (PK)(FK)	INT	Jedinstveni identifikator korisnika koji sprema objavu
post_id (PK)(FK)	INT	Jedinstveni identifikator spremljene objave

Comment

Atribut	Tip podatka	Opis varijable
comment_id	INT	Jedinstveni identifikator komentara
user_id (PK)(FK)	INT	Jedinstveni identifikator korisnika koji komentira objavu
post_id (PK)(FK)	INT	Jedinstveni identifikator komentirane objave
created_at	TIMESTAMP	Datum i vrijeme kreiranja komentara
text	VARCHAR(255)	Sadržaj komentara

Conversation

Atribut	Tip podatka	Opis varijable
id (PK)	INT	Jedinstveni identifikator razgovora

Conversation User

Atribut	Tip podatka	Opis varijable
id (PK)(FK)	INT	Jedinstveni identifikator razgovora
last_read	TIMESTAMP	Vremenski trenutak zadnjeg čitanja poruke
conversation_id	INT	Jedinstveni identifikator razgovora
user_id	INT	Jedinstveni identifikator korisnika

Conversation Read Status

Atribut	Tip podatka	Opis varijable
id (PK)(FK)	INT	Jedinstveni identifikator razgovora

Atribut	Tip podatka	Opis varijable
conversation_id	INT	Jedinstveni identifikator razgovora
user_id	INT	Jedinstveni identifikator korisnika
last_read_at	TIMESTAMP	Vrijeme zadnjeg čitanja poruke
last_read_timestamp	TIMESTAMP	Vremenski trenutak zadnjeg čitanja poruke

Conversation Participants

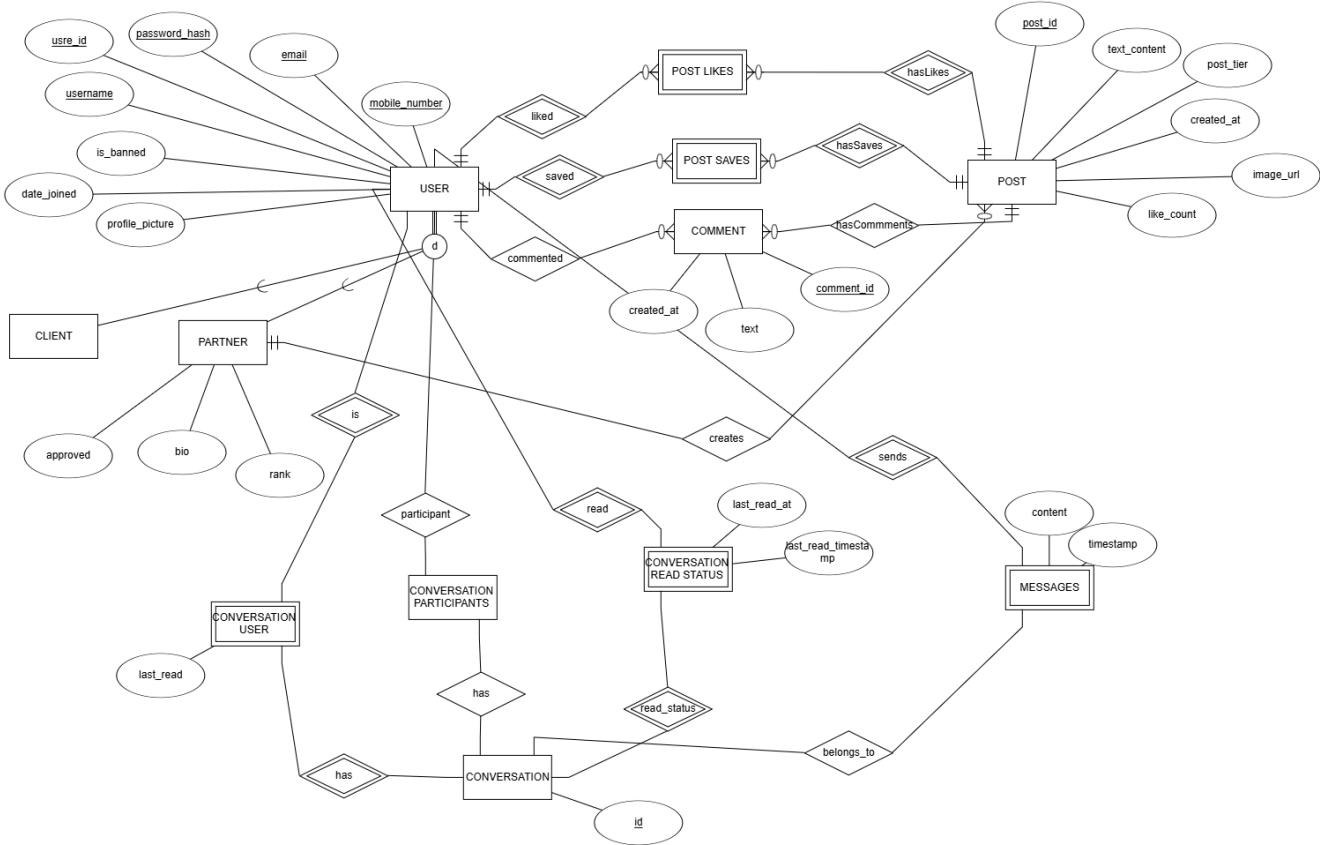
Atribut	Tip podatka	Opis varijable
conversation_id (PK)	INT	Jedinstveni identifikator razgovora
user_id	INT	Jedinstveni identifikator korisnika

Messages

Atribut	Tip podatka	Opis varijable
id (PK)(FK)	INT	Jedinstveni identifikator razgovora
content	VARCHAR(255)	Sadržaj poruke
timestamp	TIMESTAMP	Vrijeme slanja poruke
conversation_id	INT	Jedinstveni identifikator razgovora
sender_id	INT	Jedinstveni identifikator pošiljatelja

Dijagram baze podataka

ER model



Dijagram razreda

Ovaj dijagram predstavlja model baze podataka i servisne arhitekture sistema za upravljanje korisnicima, u kojem su prikazane različite klase i entiteti povezani s korisnicima, njihovim ulogama, autentifikacijom i autorizacijom, kao i interakcijom s vanjskim servisima. Slijedi opis ključnih komponenta i njihovih međusobnih veza.

- User** - Ovo je centralna klasa koja predstavlja osnovne informacije o korisniku, uključujući `id`, `email`, `username`, `password`, kao i podatke kao što su `firstName`, `lastName`, `phoneNumber`, i `profilePicture`.
- Client i Partner** - Specifični tipovi korisnika naslijeđeni od **User** klase. Ove klase omogućavaju dodatnu fleksibilnost za klijente i partnere, s različitim podacima specifičnim za svaki tip korisnika.
- Admin** - Ova klasa također nasljeđuje **User** i koristi se za kreiranje administratorskih korisnika s dodatnim privilegijama.
- UserRepository** - Repozitorij koji omogućava operacije s korisnicima u bazi podataka, kao što su pronađenje korisnika po `email` ili `username`. Implementira osnovne CRUD operacije za **User** entitet.
- UserDto i UserUpdateDto** - Data Transfer Object (DTO) klase, koje olakšavaju prijenos podataka o korisnicima između različitih slojeva aplikacije, uključujući informacije o kreiranju i ažuriranju korisnika.
- UserService** - Glavni servis za logiku vezanu za korisnike. Sadrži metode za registraciju, prijavu, i ažuriranje korisnika, kao i dodatnu logiku za validaciju i provjeru statusa korisnika.

7. **JwtTokenProvider** - Komponenta odgovorna za generiranje i validaciju JWT tokena, omogućava korisnicima da se autentificiraju koristeći tokene.
8. **SecurityConfig** - Konfiguracijska klasa za sigurnost koja implementira **AuthenticationFilter** i određuje pravila pristupa, što obuhvaća autorizaciju i autentifikaciju korisnika.
9. **CorsConfig** - Konfiguracijska klasa koja definira pravila za CORS (Cross-Origin Resource Sharing). Omogućava backendu kontrolu nad tim odakle dolaze zahtjevi, koje HTTP metode i zaglavlja su dopušteni, te omogućava rad s kolačićima i autorizacijskim tokenima.
10. **UserController** - Kontroler koji upravlja API pozivima za operacije vezane za korisnike, kao što su registracija, prijava, i ažuriranje podataka korisnika.
11. **PostController** - Kontroler koji omogućava upravljanje objavama, uključujući kreiranje, ažuriranje, likeanje, komentiranje i brisanje objava. Također omogućava pregled svih objava i filtriranje na temelju korisničke razine preplate.
12. **PaymentController** - Kontroler za upravljanje procesom plaćanja. Omogućava integraciju sa Stripe API-jem za upravljanje korisničkim pretplatama i provođenje plaćanja.
13. **ChatController** - Kontroler za upravljanje razgovorima i porukama između korisnika. Podržava stvaranje novih razgovora, dohvaćanje svih razgovora, slanje poruka i ažuriranje statusa pročitanih poruka.

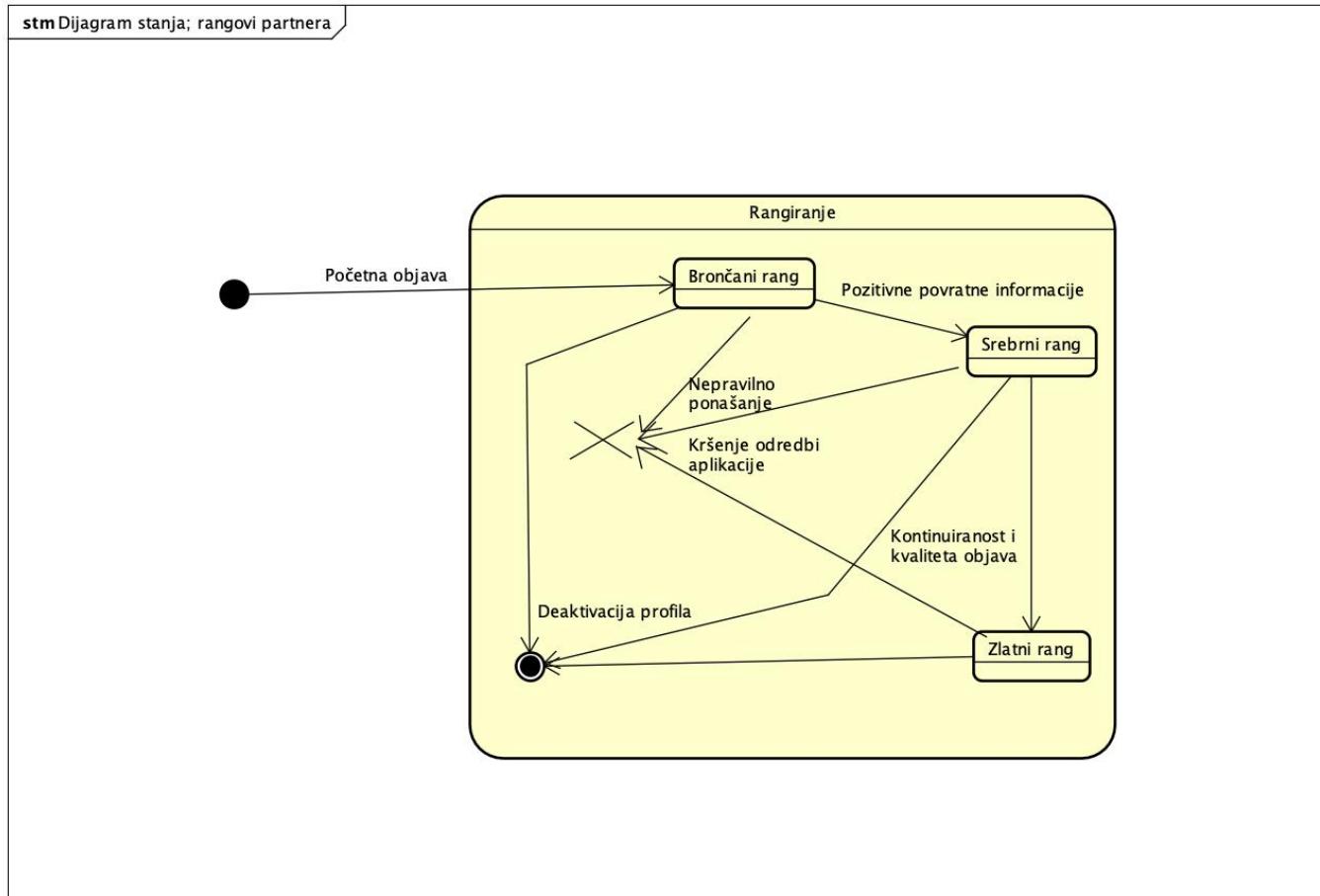


Dinamičko ponašanje aplikacije

Dinamičko ponašanje aplikacije odnosi se na način na koji objekti u sustavu evoluiraju kroz vrijeme, uključujući prijelaze između različitih stanja. To uključuje aktivnosti, događaje, odluke i interakcije unutar aplikacije.

UML dijagrami stanja

UML dijagrami stanja nužni su za razumijevanje dinamičkog ponašanja sustava. Oni jasno prikazuju promjene stanja objekata tijekom vremena ovisno o događajima i uvjetima.

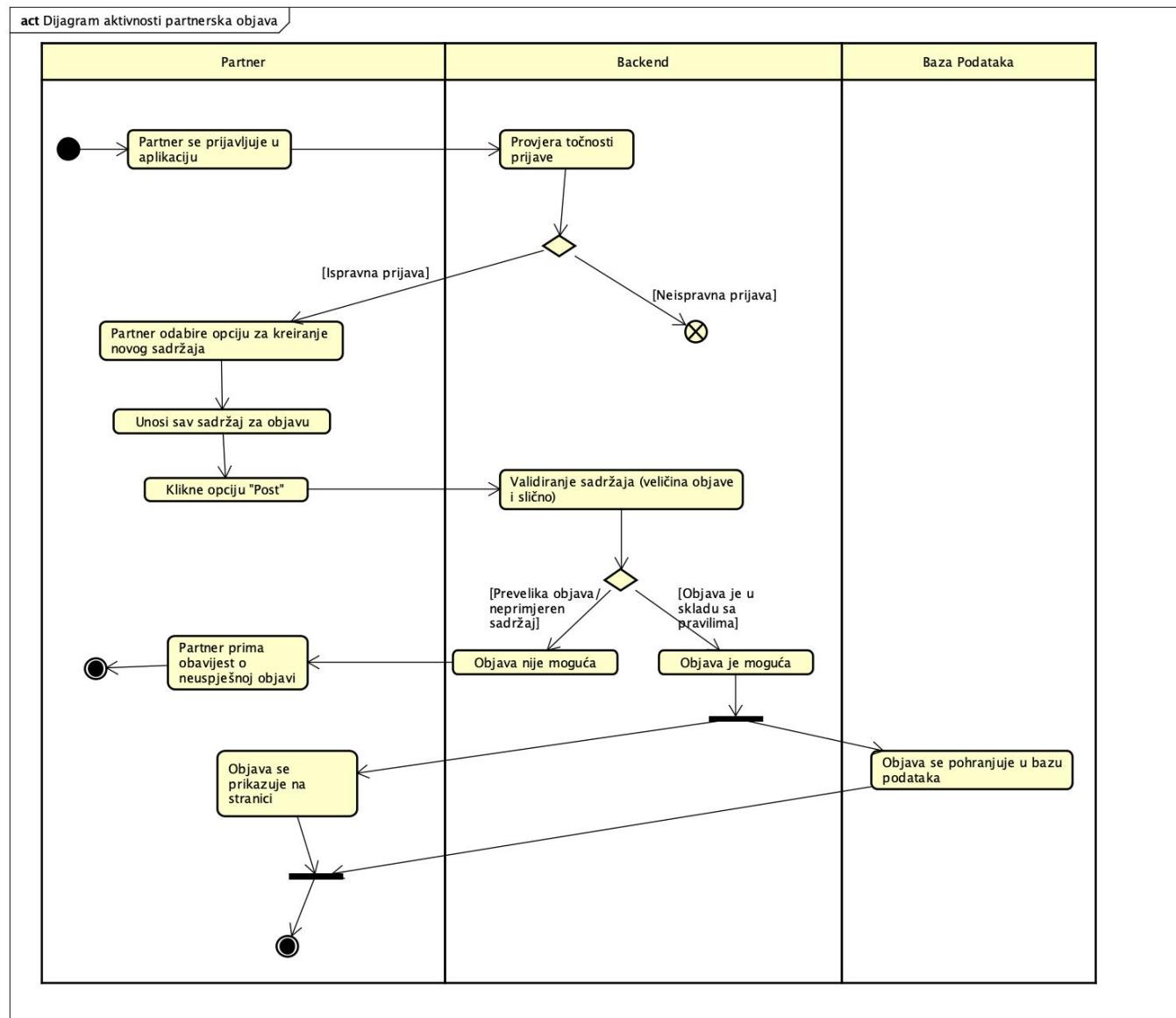


Dijagram stanja za rangiranje partnera

Svako stanje u dijagramu jasno označava trenutačni status partnera na platformi i reflektira njihovu razinu angažmana. Prijelazi su uvjetovani specifičnim kriterijima koje platforma definira, poput broja objava, kvalitete sadržaja i korisničkog angažmana. "Onemogućeno stanje" osigurava mehanizam kontrole kvalitete i poštivanja pravila platforme.

UML dijagrami aktivnosti

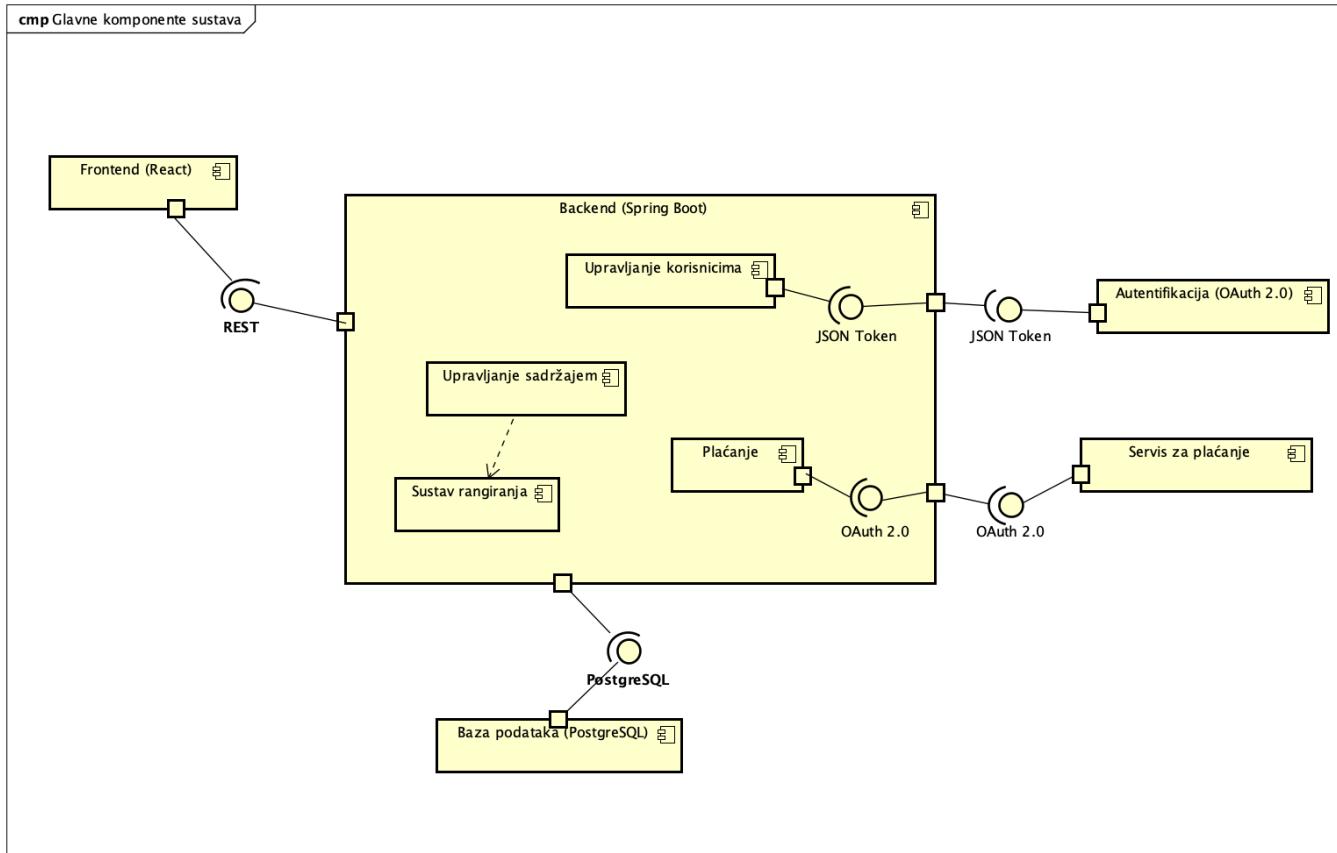
Dijagram aktivnosti prikazuje tijek izvršavanja određenog procesa. Osim za razumijevanje toka podataka unutar aplikacije, koristi se za analizu poslovnih procesa.



Dijagram aktivnosti za partnerske objave

Proces prikazuje korake koje partner prolazi pri kreiranju i dijeljenju sadržaja putem aplikacije. Uključuje provjeru podataka, validaciju sadržaja i prikazivanje objave.

Dijagram komponenata



Glavne komponente sustava i njihove uloge:

Frontend (React):

- Implementiran pomoću React biblioteke za razvoj korisničkog sučelja.
- Komunicira s backend-om preko REST API-ja.
- Prikazuje korisničke podatke, upravlja tokenima za autentifikaciju (JWT) i omogućava korisničku interakciju s aplikacijom.

Backend (Spring Boot):

- Implementira poslovnu logiku aplikacije kroz četiri sloja:

Upravljanje korisnicima: Upravlja procesima autentifikacije, autorizacije i pohrane korisničkih podataka koristeći JSON Web Tokene (JWT).

Upravljanje sadržajem: Omogućava kreiranje, uređivanje i brisanje sadržaja od strane korisnika.

Sustav rangiranja: Određuje rangove partnera na temelju angažmana i korisničkih ocjena.

Plaćanje: Upravljanje procesima transakcija i plaćanja putem Stripea.

- Koristi OAuth 2.0 protokol za sigurno rukovanje autentifikacijom i autorizacijom.

Baza podataka (PostgreSQL):

- Relacijska baza podataka koja pohranjuje podatke o korisnicima, sadržaju, transakcijama i sustavu rangiranja.

- Komunikacija s backendom ostvaruje se preko JDBC protokola.

Vanjski servisi:

- Autentifikacija (OAuth 2.0): Koristi se za upravljanje autentifikacijom korisnika.
- Servis za plaćanje (Stripe): Upravljanje transakcijama putem webhookova i sigurnog procesiranja plaćanja.

O komunikaciji:

- Frontend ↔ Backend:

Komunikacija putem REST API-ja preko HTTPS protokola.

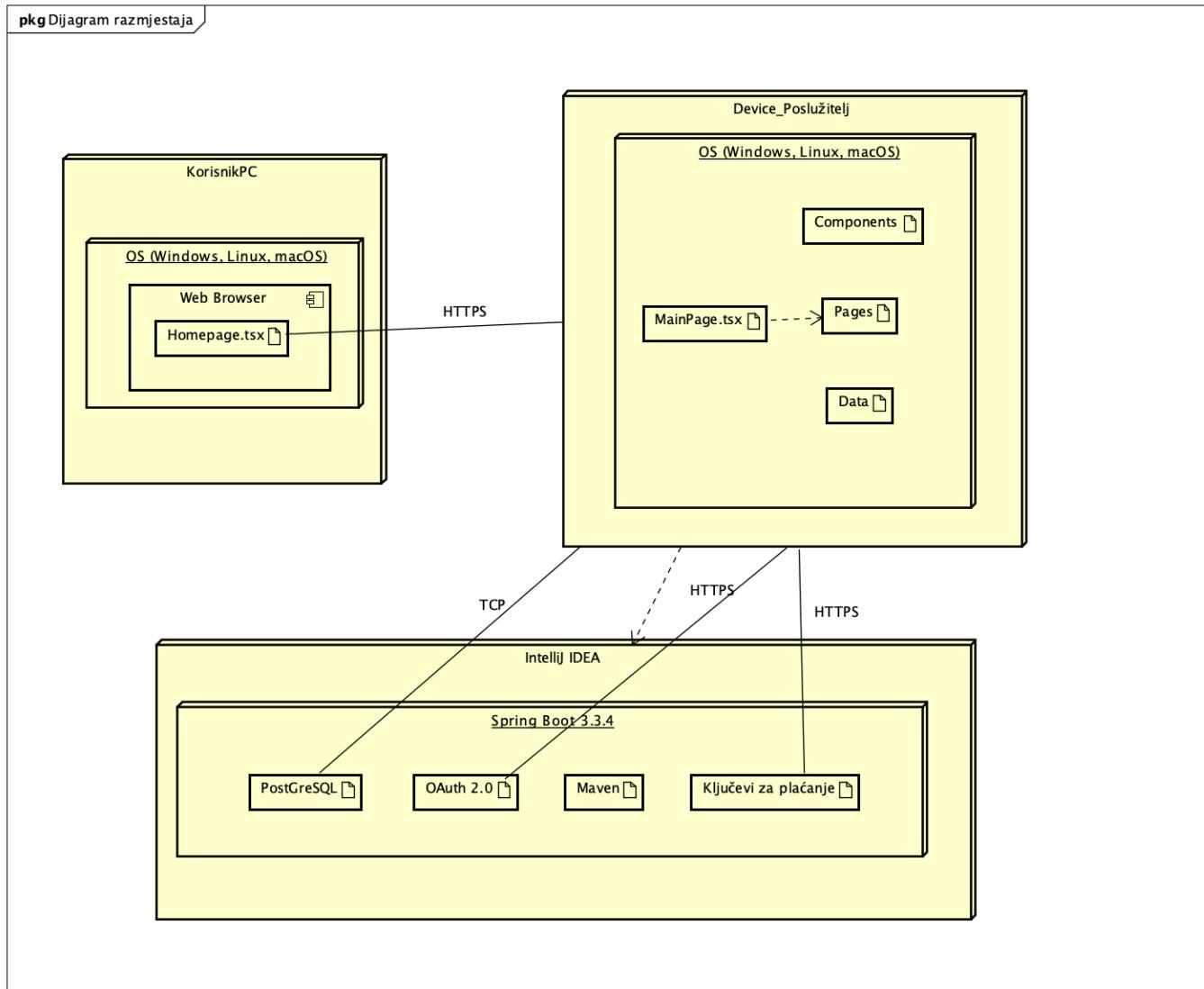
- Backend ↔ Vanjski servisi:

Autentifikacija i plaćanje koriste OAuth 2.0 za sigurnu razmjenu podataka.

- Backend ↔ Baza podataka:

Podaci se dohvaćaju i pohranjuju pomoću Spring Data JPA i Hibernate ORM-a.

Dijagram razmještaja



Čvorovi i njihova uloga:

- Korisnički uređaj (KorisnikPC):

Pokreće web preglednik (npr. Chrome, Firefox) za interakciju s aplikacijom.

Prikazuje korisničko sučelje implementirano u Reactu.

- Poslužitelj aplikacije (Device_Poslužitelj):

Hostira backend sustav implementiran u Spring Bootu.

Obrađuje zahtjeve korisnika i šalje odgovore prema frontendu.

Upravljanje poslovnom logikom aplikacije.

- Baza podataka (PostgreSQL):

Pohranjuje sve podatke potrebne za rad aplikacije (korisnici, sadržaj, transakcije).

Implementirana na dedicated serveru ili cloud platformi.

Povezanost čvorova:

- Korisnički uređaj ↔ Poslužitelj aplikacije:

Komunikacija se odvija preko HTTPS protokola.

REST API endpointi omogućuju sigurno slanje zahtjeva i dohvaćanje podataka.

- Poslužitelj aplikacije ↔ Baza podataka:

Komunikacija koristi JDBC protokol preko TCP veze za dohvaćanje i ažuriranje podataka.

Infrastruktura:

- IntelliJ IDEA: Koristi se kao razvojno okruženje za backend.
- Maven: Upravljanje ovisnosti u Spring Boot projektu.
- OAuth 2.0: Koristi se za autentifikaciju i autorizaciju korisnika.
- Ključevi za plaćanje: Upravljanje sigurnim transakcijama pomoću Stripea.

Ispitivanje komponenti

Ovaj skup testova za validaciju osnovnih funkcionalnosti chat servisa smo napisali koristeći JUnit i Mockito. JUnit omogućuje pisanje i izvođenje automatskih testova, osiguravajući da svaka komponenta sustava radi izolirano. Mockito služi za kreiranje lažnih (mock) objekata koji oponašaju ponašanje stvarnih ovisnosti čime se osigurava izoliranost testova i fokus na testiranje logike servisa. Cilj je osigurati da sustav ispravno reagira na različite scenarije, bilo da se radi o kreiranju novih razgovora, prepoznavanju postojećih, rukovanju pogreškama prilikom slanja poruka ili pokušajima korištenja neimplementiranih funkcionalnosti. Svi testovi se nalaze u backend folderu na putanji [src/test/java/hr/fer/sportconnect](#).

Test 1 - stvaranje razgovora s novim korisnicima

- Funkcionalnost: Stvaranje novog razgovora između dvaju korisnika koji prije nisu imali zajednički razgovor.
 - Ispitni slučaj:
 - Ulazni podaci:

```
User user1 = new User("ana@mail.com", ... )
```

```
User user2 = new User("marko@mail.com", ... )
```

- Očekivani rezultat:

Stvara se novi objekt tipa Conversation i spremi se u conversationRepository. Dva ConversationReadStatus zapisa (po jedan za svakog korisnika) čuvaju se u readStatusRepository. Metoda vraća spremljen razgovor s ispravnim ID-om i sudionicima.

- Dobiveni rezultati:

Novo stvoren razgovor s ID-om npr. 10, sadrži user1 i user2. "mockovi" potvrđuju da su conversationRepository.save(...) i readStatusRepository.saveAll(...) pozvani jednom.

- Postupak provođenja ispitivanja:

Stvorimo mock ConversationRepository i mock ConversationReadStatusRepository. Pozovemo findConversationBetweenUsers(user1, user2) koji vraća Optional.empty(). Kada se zove conversationRepository.save(...), vratimo Conversation s generiranim ID-om (npr. 10). Pozovemo chatService.createConversation(user1, user2) koji vraća prethodno stvoren razgovor. Provjerimo je li vraćen razgovor s dvama sudionicima i novim ID-om.

Test 2 - stvaranje razgovora s postojećim korisnicima

- Funkcionalnost: Stvaranje novog razgovora između dvaju korisnika koji već imaju zajednički razgovor.
 - Ispitni slučaj:
 - Ulazni podaci:

User user1 = new User("ana@mail.com", ...)

User user2 = new User("marko@mail.com", ...)

- Očekivani rezultat:

Metoda ne kreira novi razgovor nego vraća postojeći. Ne stvaraju se novi zapisi u ConversationReadStatus.

- Dobiveni rezultati:

Servis vraća postojeći razgovor, npr. s ID-om 5.

- Postupak provođenja ispitivanja:

Napravimo mock conversationRepository.findConversationBetweenUsers(user1, user2) koji vraća Optional.of(existingConversation). Pokušamo stvoriti novi razgovor chatService.createConversation(user1, user2). Očekujemo da dobijemo existingConversation natrag. Provjerimo da conversationRepository.save(...) i readStatusRepository.saveAll(...) nisu pozvani za novi razgovor.

Test 3 - slanje poruke u nepostojeći razgovor

- Funkcionalnost: Reakcija na grešku kad traženi razgovor ne postoji.
 - Ispitni slučaj:
 - Ulazni podaci:

Long conversationId = 99 (nepostojeći)

User sender = new User("marko@mail.com", ...)

content = "Test Message"

conversationRepository.findById(99) vraća Optional.empty().

- Očekivani rezultat:

Metoda baci RuntimeException("Conversation not found").

- Dobiveni rezultati:

Metoda baca RuntimeException.

- Postupak provođenja ispitivanja:

Napravimo mock conversationRepository.findById(99) koji vraća Optional.empty().

Pozovemo chatService.sendMessage(sender, 99, "Test Message"). Očekujemo

RuntimeException("Conversation not found"). Potvrđimo da

messageRepository.save(...) i pusher.trigger(...) nikada nisu pozvani.

Test 4 - prikačivanje razgovora na vrh liste

- Funkcionalnost: Pokušaj "pinConversation" i očekivano bacanje "NotImplementedException" iznimke.

- Ispitni slučaj:

- Ulazni podaci:

User user1 = new User("ana@mail.com", ...)

conversationId = 12

- Očekivani rezultat:

Baci "NotImplementedException("Pinning is not supported.")"

- Dobiveni rezultati:

Dogodila se iznimka "NotImplementedException("Pinning is not supported.")"

- Postupak provođenja ispitivanja:

Simuliramo poziv metode pinConversation budući da nije implementirana.

Test 5 - registracija novog korisnika

- Funkcionalnost: Registracija novog korisnika koji još ne postoji u sustavu.

- Ispitni slučaj:

- Ulazni podaci:

String email = "ana@mail.com"

String password = "password"

String firstName = "Ana"

String lastName = "Horvat"

String username = "anahorvat"

```
UserType userType = UserType.CLIENT  
SubscriptionPlan subscriptionPlan = SubscriptionPlan.FREE  
String mobileNumber = "+385 954123658"
```

- Očekivani rezultat:

Stvara se novi objekt tipa Client koji se sprema u clientRepository. Metoda vraća UserDto s ispravnim atributima korisnika. Sve provjere jedinstvenosti (email, korisničko ime, broj mobitela) uspješno prolaze, a lozinka se ispravno kodira.

- Dobiveni rezultati:

Generirani korisnik s emailom "ana@mail.com", korisničkim imenom "anahorvat", i kodiranom lozinkom "encodedPassword". "Mockovi" potvrđuju da su metode userRepository.existsByEmail(...), userRepository.existsByUserName(...), userRepository.existsByMobileNumber(...) i clientRepository.save(...) pozvane po jednom.

- Postupak provođenja ispitivanja:

Stvaramo mock objekte za userRepository, clientRepository, passwordEncoder, i userMapper.

Simuliramo povratne vrijednosti za metode userRepository.existsBy...(...), passwordEncoder.encode(...), i clientRepository.save(...).

Pozivamo userService.registerUser(userRegistrationDto).

Provjeravamo rezultat (UserDto) i njegove atribute.

Validiramo da su sve potrebne metode mockova pozvane točno onoliko puta koliko je očekivano.

Test 6 - kreiranje nove objave

- Funkcionalnost: Kreiranje nove objave za korisnika partnera.

- Ispitni slučaj:

- Ulagani podaci:

```
User user1 = new User("ana@mail.com", ... )  
  
String textContent = "Ovo je probni post."  
  
SubscriptionPlan subscriptionPlan = SubscriptionPlan.FREE
```

- Očekivani rezultat:

Stvara se novi objekt tipa Post koji sadrži korisnika (partnera), tekstualni sadržaj i odgovarajući tier (razinu pretplate). Objekt se sprema u postRepository, a metoda vraća spremljeni post (objavu).

- Dobiveni rezultati:

Generirani post s tekstualnim sadržajem "Ovo je probni post.", partnerom "Ana Horvat", i preplatom FREE. "Mockovi" potvrđuju da je metoda `postRepository.save(...)` pozvana jednom.

- Postupak provođenja ispitivanja:

Stvorimo mock za `postRepository`.

Simuliramo povratnu vrijednost za metodu `postRepository.save(...)`.

Pozovemo `postService.createPost(user, textContent, null, subscriptionPlan)`.

Provjerimo rezultat (Post) i njegove atribute.

Validiramo da je metoda mocka `postRepository.save(...)` pozvana točno jednom.

Ispitivanje sustava

Test 1 - **Kreiranje korisničkog računa** (CreateAccountTest)

Ulazi

- **Podaci koji se unose u sustav:**

- Ime: **Vernon**
- Prezime: **Schillinger**
- Korisničko ime: **schillinger**
- Kontakt: **+385 92508313**
- E-mail: **schillinger@gmail.com**
- Lozinka: **123456789**

- **Simulirane korisničke akcije:**

1. Navigacija na stranicu za kreiranje računa.
2. Unos podataka u formu.
3. Odabir uloge "Klijent".
4. Označavanje polja za prihvatanje uvjeta.
5. Klik na gumb "Kreiraj račun".

Koraci ispitivanja

1. Otvori aplikaciju u web pregledniku.
2. Navigiraj na stranicu za kreiranje računa pozivom metode `homePage.openCreateAccount()`.
3. Odaber ulogu "Klijent" pomoću metode `createAccountPage.selectClient()`.
4. Unesi podatke u formu koristeći sljedeće metode:
 - Ime: `createAccountPage.setName("Vernon")`
 - Prezime: `createAccountPage.setSurname("Schillinger")`
 - Korisničko ime: `createAccountPage.setUsername("schillinger")`
 - Kontakt: `createAccountPage.setContact("+385 92508313")`
 - E-mail: `createAccountPage.setEmail("schillinger@gmail.com")`

- Lozinka: `createAccountPage.setPassword("123456789")`
- 5. Označi polje za prihvatanje uvjeta: `createAccountPage.checkAgreementBox()`.
- 6. Klikni na gumb "Kreiraj račun" pomoću metode `createAccountPage.createAccount()`.
- 7. Verificiraj da je korisnik preusmjeren na glavnu stranicu aplikacije pozivom metode `Assert.assertTrue(mainPage.isThisMainPage())`.

Očekivani izlaz

- Korisnik je uspješno preusmjeren na glavnu stranicu aplikacije. Metoda `mainPage.isThisMainPage()` vraća `true`.

Dobiveni izlaz

```
=====
CreateAccountTest-Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====
Process finished with exit code 0
```

Test 2 - Objavljivanje tekstualne objave (PostTextTest)

Ulazi

- **Podaci koji se unose u sustav:**
 - Korisničko ime: `mcmanus`
 - Lozinka: `mcmanus`
 - Tekstualna objava: "`In Em City we workout too!`"
- **Simulirane korisničke akcije:**
 1. Prijava u sustav s korisničkim imenom i lozinkom.
 2. Unos tekstualne objave.
 3. Objavljivanje teksta.
 4. Prihvatanje obavijesti (alert).
 5. Provjera vidljivosti objavljenog teksta na stranici.

Koraci ispitivanja

1. Otvori aplikaciju u web pregledniku.
2. Navigiraj na stranicu za prijavu pomoću metode `homePage.openSignIn()`.
3. Unesi podatke za prijavu:
 - Korisničko ime: `signInPage.setUsername("mcmanus")`
 - Lozinka: `signInPage.setPassword("mcmanus")`
4. Klikni na gumb za prijavu pomoću metode `signInPage.signIn()`.
5. Unesi tekstualnu objavu:
 - Tekst: `mainPage.setTextPost("In Em City we workout too!")`

6. Klikni na gumb za objavljivanje teksta: `mainPage.postIt()`.
7. Prihvati obavijest (alert) pomoću metode `mainPage.acceptAlert()`.
8. Verificiraj da je objava vidljiva na stranici:
 - o `assertTrue(mainPage.isPostVisible("In Em City we workout too!"), "The post is not visible on the page.")`

Očekivani izlaz

- Objavljeni tekst "In Em City we workout too!" je vidljiv na stranici.

Dobiveni izlaz

```
=====
PostTextTest-Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====
Process finished with exit code 0
```

Test 3 - Pokušaj promjene korisničkog imena u krivom formatu (IncorrectUsernameChangeTest)

Ulazi

- **Podaci koji se unose u sustav:**
 - o Korisničko ime za prijavu: `busmalis`
 - o Lozinka za prijavu: `123456`
 - o Neispravno korisničko ime za promjenu: `bus`
- **Simulirane korisničke akcije:**
 1. Prijava na sustav s ispravnim korisničkim imenom i lozinkom.
 2. Navigacija na profil korisnika.
 3. Otvaranje stranice za uređivanje profila.
 4. Promjena korisničkog imena na neispravni format.
 5. Spremanje promjena.
 6. Provjera je li prikazana poruka o grešci.

Koraci ispitivanja

1. Otvori aplikaciju u web pregledniku.
2. Navigiraj na stranicu za prijavu pomoću metode `homePage.openSignIn()`.
3. Unesi podatke za prijavu:
 - o Korisničko ime: `signInPage.setUsername("busmalis")`
 - o Lozinka: `signInPage.setPassword("123456")`

4. Klikni na gumb za prijavu pomoću metode `signInPage.signIn()`.
5. Otvori stranicu "Moj profil" pomoću metode `mainPage.openMyProfile()`.
6. Otvori stranicu za uređivanje profila pomoću metode `myProfilePage.openEditProfile()`.
7. Promijeni korisničko ime na neispravni format:
 - Korisničko ime: `editProfilePage.changeUsername("bus")`
8. Spremi promjene pomoću metode `editProfilePage.saveChanges()`.
9. Verificiraj da je prikazana poruka o neispravnom formatu korisničkog imena:
 - `Assert.assertTrue(editProfilePage.isMessagePresent())`

Očekivani izlaz

- Test uspijeva ako je vidljiva poruka koja ukazuje na **neispravan format korisničkog imena**.

Dobiveni izlaz

```
=====
IncorrectUsernameChangeTest-Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====

Process finished with exit code 0
```

Test 4 - **Pokušaj pretplate na višu razinu s najviše razine** (UpgradeFromGoldTest)

Ulazi

- **Podaci koji se unose u sustav:**
 - Korisničko ime za prijavu: `busmalis`
 - Lozinka za prijavu: `123456`
- **Simulirane korisničke akcije:**
 1. Prijava na sustav s ispravnim korisničkim imenom i lozinkom.
 2. Navigacija na profil korisnika.
 3. Provjera dostupnosti gumba za nadogradnju na viši nivo pretplate.

Koraci ispitivanja

1. Otvori aplikaciju u web pregledniku.
2. Navigiraj na stranicu za prijavu pomoću metode `homePage.openSignIn()`.
3. Unesi podatke za prijavu:
 - Korisničko ime: `signInPage.setUsername("busmalis")`
 - Lozinka: `signInPage.setPassword("123456")`
4. Klikni na gumb za prijavu pomoću metode `signInPage.signIn()`.

5. Otvori stranicu "Moj profil" pomoću metode `mainPage.openMyProfile()`.
6. Pokušaj pronaći i kliknuti na gumb za nadogradnju pretplate pomoću metode `myProfilePage.upgradeNow()`.
7. Ako gumb nije dostupan (baca `NoSuchElementException`):
 - Verificiraj da element za nadogradnju pretplate ne postoji pomoću metode `Assert.assertThat(myProfilePage.isElementPresent(), is(false))`.

Očekivani izlaz

- Test uspijeva ako:
 - Gumb za nadogradnju pretplate nije dostupan za korisnike koji su već preplaćeni na **Gold** pretplatu.
 - Baca se iznimka `NoSuchElementException`, koja potvrđuje da element ne postoji.

Dobiveni izlaz

```
=====
UpgradeFromGoldTest-Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
=====
Process finished with exit code 0
```

Korištene tehnologije i alati

Za razvoj aplikacije korištene su sljedeće tehnologije:

Programski jezici

Aplikacija je implementirana u Java (verzija 17), suvremenom programskom jeziku poznatom po svojoj stabilnosti, sigurnosti i širokoj podršci u razvoju enterprise aplikacija.

Radni okviri i biblioteke

Backend

Za poslužiteljsku stranu aplikacije korišten je **Spring Boot** (verzija 3.3.4), popularni okvir za izgradnju modernih Java aplikacija. Spring Boot olakšava razvoj i upravljanje aplikacijama pomoću unaprijed konfiguriranih postavki i bogatog ekosustava integracija. Korišteni Spring Boot starteri uključuju:

- **spring-boot-starter-web**: Za razvoj REST API-ja i web aplikacija.
- **spring-boot-starter-data-jpa**: Za jednostavnu integraciju s bazama podataka i rad s JPA (Java Persistence API).

- **spring-boot-starter-security**: Za implementaciju sigurnosnih značajki kao što su autentifikacija i autorizacija.
- **spring-boot-starter-validation**: Za jednostavnu validaciju ulaznih podataka.
- **spring-boot-starter-oauth2-client**: Za integraciju s OAuth2 pružateljima identiteta.
- **spring-boot-starter-webflux**: Omogućuje reaktivno programiranje za visokoperformansne aplikacije.

Frontend

Za klijentski dio aplikacije korišten je **React** (verzija 18), popularna biblioteka za izgradnju interaktivnih korisničkih sučelja. React omogućuje razvoj komponenata koje su ponovo iskoristive i lako održive, čime se poboljšava učinkovitost razvoja.

Za upravljanje stilovima korišten je **styled-components**, što omogućuje primjenu stilova unutar React komponenti koristeći CSS-in-JS pristup.

Dodatne biblioteke i alati

- **PostgreSQL JDBC Driver (verzija 42.2.18)**: Povezivanje aplikacije s PostgreSQL bazom podataka.
- **MapStruct (verzija 1.5.5.Final)**: Za automatsko mapiranje između DTO-ova i entiteta, čime se pojednostavljuje rad s objektima u aplikaciji.
- **JSON Web Token (io.jsonwebtoken, verzija 0.11.5)**: Za upravljanje sigurnim tokenima za autentifikaciju.
- **AWS SDK (verzija 2.20.23)**: Za integraciju s Amazon S3 i druge AWS usluge.
- **Pusher HTTP Java (verzija 1.3.4)**: Za real-time komunikaciju putem Pusher usluge.
- **Stripe Java SDK (verzija 28.2.0)**: Za integraciju Stripeom za obradu plaćanja.

Alati za testiranje

- **spring-boot-starter-test**: Uključuje JUnit i druge alate za testiranje.
- **Maven Surefire Plugin (verzija 3.5.2)**: Konfiguriran za pokretanje testova s mogućnošću ignoriranja neuspjelih slučajeva ako je potrebno.

Kombinacija ovih tehnologija omogućuje brzu implementaciju, visoku sigurnost, skalabilnost te lakoću održavanja i proširenja aplikacije.

Baza podataka

Aplikacija koristi PostgreSQL (verzija 42.2.18), pouzdanu i skalabilnu relacijsku bazu podataka. PostgreSQL je odabran zbog svoje podrške za složene SQL operacije i naprednih značajki poput indeksiranja, čime se osigurava visoka izvedba i pouzdanost.

Razvojni alati

Za razvoj koda koristi se IntelliJ IDEA, moćno integrirano razvojno okruženje s naprednim značajkama za refaktoriranje i analizu koda. Verzioniranje koda upravlja se pomoću Git-a (verzija 2.44.0), čime je omogućeno praćenje promjena i učinkovita timska suradnja.

Alati za ispitivanje

Za ispitivanje aplikacije korišten je **Selenium**, alat za automatizirano testiranje sustava.

Selenium omogućuje testiranje korisničkog sučelja kako bi se osiguralo da aplikacija ispravno reagira na interakcije korisnika u internetskom pregledniku. Ovaj alat osigurava da aplikacija radi prema očekivanjima iz perspektive krajnjeg korisnika.

Alati za razmještaj

Za razmještaj aplikacije koristi se Docker (verzija 4.35.1), koji omogućuje standardizaciju okruženja za razvoj, testiranje i produkciju. Docker kontejneri osiguravaju dosljedne uvjete rada bez obzira na razlike u operacijskim sustavima.

Cloud platforma

Aplikacija je hostana na Google Cloud platformi, poznatoj po svojoj visokoj dostupnosti, skalabilnosti i bogatom setu integriranih alata za upravljanje aplikacija u oblaku. Google Cloud osigurava stabilnu infrastrukturu i omogućuje lako proširenje aplikacije prema potrebama korisnika.

1. Instalacija

- **Preduvjeti:**

- IntelliJ IDEA, Visual Studio Code ili sličan IDE (ove upute su napisane za IntelliJ i Visual Studio Code)
- Java 17, Spring Boot 3.3.4, React 18.3.1, TypeScript 5.5.3
- Docker 4.35.1
- Google Cloud, Render

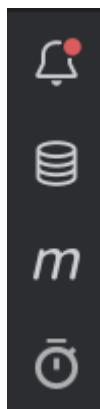
- **Preuzimanje backenda:** Upute za preuzimanje izvornog koda.

Potrebno je preuzeti kod s GitHuba te ga učitati unutar IntelliJa kao Maven projekt.

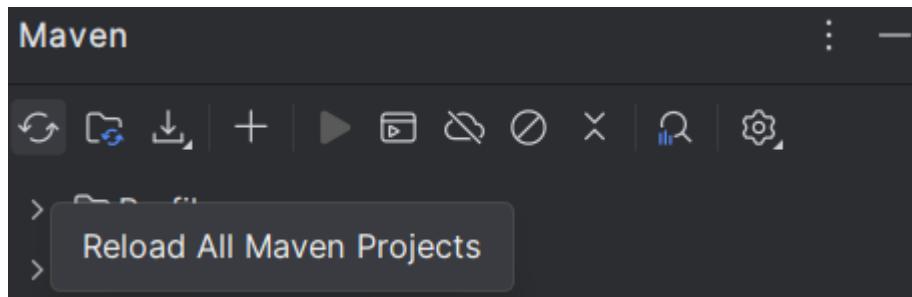
```
git clone --branch backend https://github.com/ManuelFijan/SportConnect.git
```

Instalacija ovisnosti: Upute za instaliranje ovisnosti.

Nakon što je projekt učitan unutar IntelliJa, sve ovisnosti bi se automatski trebale instalirati iz pom.xml datoteke. Ako se zbog bilo kojeg razloga to ne desi, potrebno je na desnoj strani kliknuti na Maven ikonu (slovo m, *slika 1*). U novo otvorenom okviru treba kliknuti na tipku "Reload all Maven projects" (*slika 2*).



Slika 1



Slika 2

- **Preuzimanje frontenda:** Upute za preuzimanje izvornog koda.

Potrebno je preuzeti kod s GitHuba te ga učitati unutar Visual Studio Codea.

```
git clone --branch frontend https://github.com/ManuelFijan/SportConnect.git
```

Instalacija ovisnosti: Upute za instaliranje ovisnosti.

U terminalu unutar VSC-a je potrebno upisati `npm install` (slika 3) kako bi se instalirale sve ovisnosti.

```
PS C:\Users\Manuel\Desktop\react\FRONTEND\IzvorniKod\front> npm install
```

Slika 3

2. Postavke

Detaljne upute za konfiguraciju aplikacije:

- Konfiguracijska datoteka `application.properties` se nalazi u direktoriju `src/main/resources`. Služi kao pohrana za API ključeve ili mijenjanje konfiguracije aplikacije. Moguće je mijenjati ime aplikacije, port na kojem se pokreće ili omogućiti ispis dodatnih logova prilikom izbacivanja iznimki. API ključevi i slične stvari se pohranjuju najčešće u ovom obliku:
`ovo.je.ime.varijable.u.app.properties=${ovo.je.ime.varijable.u.environment.variable s}`. Varijable se koriste u kodu s pomoću `@Value` anotacije, npr.
`@Value("${ime.varijable.u.app.properties}")`. Referenciramo se na varijablu u `application.properties` datoteke koja svoju vrijednost vuče iz `environment` varijabla koja je spremljena

samo na našem računalu lokalno. Tako nitko ne može ukrasti naše API ključeve te napraviti štetu.

```

spring.application.name=SportConnect
server.servlet.session.cookie.same-site=none
server.servlet.session.cookie.secure=true

server.port=8080

#baza
spring.datasource.url=${SPRING_DATASOURCE_URL}
spring.datasource.username=${SPRING_DATASOURCE_USERNAME}
spring.datasource.password=${SPRING_DATASOURCE_PASSWORD}
spring.datasource.driver-class-name=org.postgresql.Driver

```

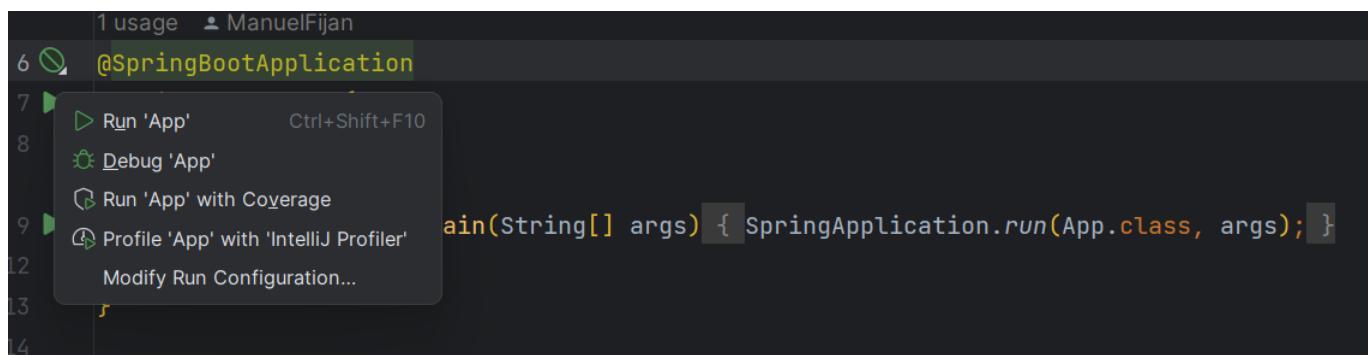
- Frontend kod također ima jednu konfiguracijsku datoteku `.env` u kojoj se nalazi URL lokalno pokrenutog ili deployanog backenda. Tako URL nije hardkodiran te ga je potrebno promijeniti na samo jednom mjestu ako želimo testirati različitu verziju. Datoteka se nalazi u korijenskom direktoriju projekta.

`VITE_BACKEND_API=https://sportconnect-531910440961.europe-west3.run.app`

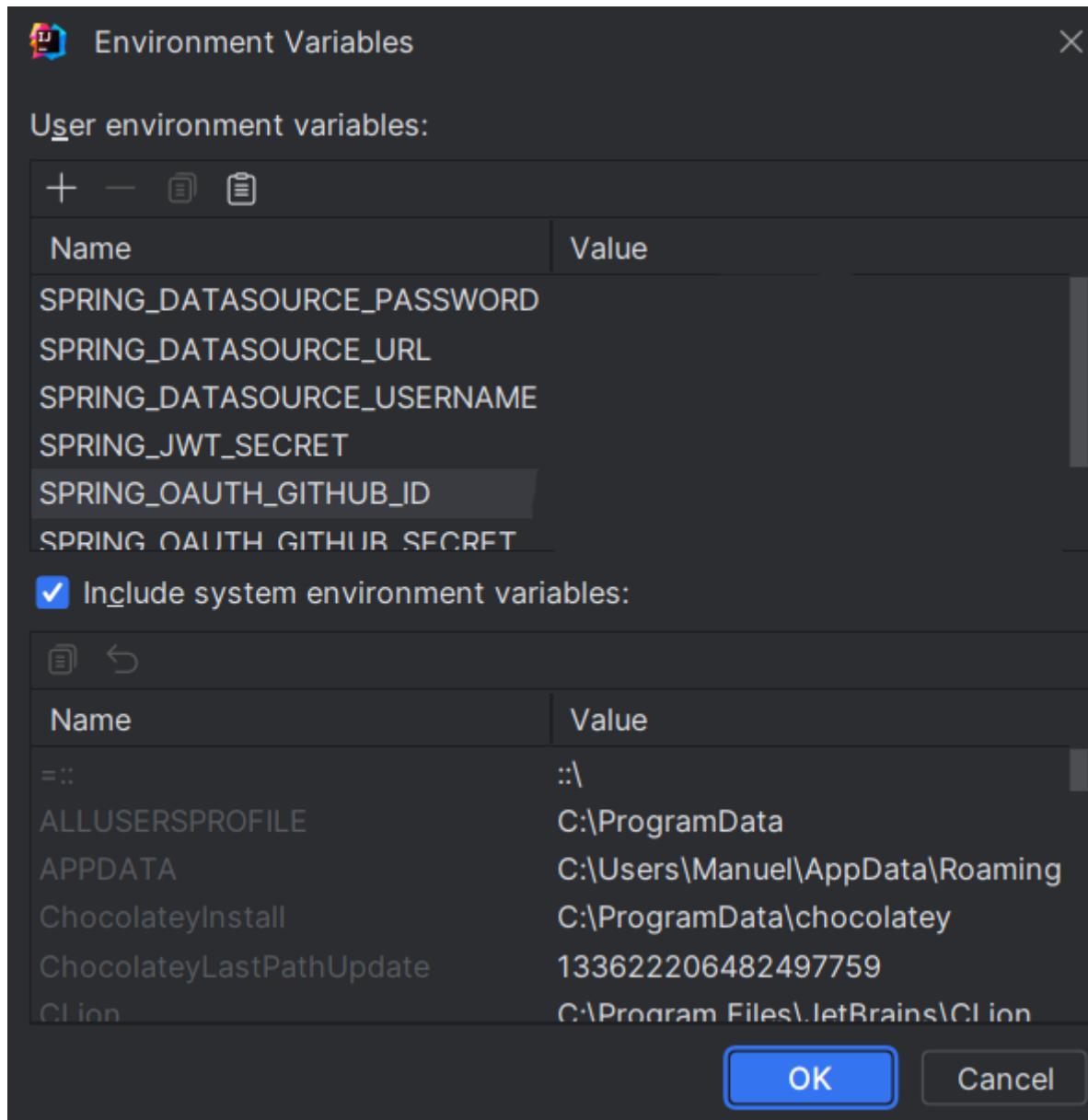
3. Pokretanje aplikacije lokalno

Backend

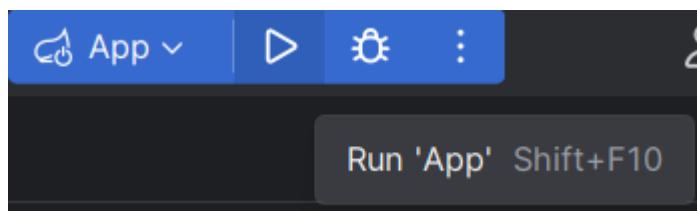
- Unutar direktorija `src/main/java/hr/fer/sportconnect` treba pronaći App datoteku te kliknuti na zeleni trokut kraj deklaracije klase i odabratи opciju "Modify Run Configuration" (*slika 4*). U novom okviru pod "Environment variables" potrebno je navesti sve parove naziv varijable - vrijednost varijable koje su korištene unutar navedene datoteke `application.properties` (*slika 5*). Na taj način zaštićuju povjerljive i rizične informacije kao što su API ključevi vanjskih servisa. Kada su sve vrijednosti unesene, potrebno je stisnuti "Apply" i "OK". Backend je sada moguće pokrenuti na vrhu navigacijske trake (*slika 6*).



Slika 4



Slika 5



Slika 6

- Provjera rada: Backend se po defaultu pokreće na portu 8080 te je stoga u tražilicu najlakše upisati `http://localhost:8080` uz poziv nekog endpointa, npr. `http://localhost:8080/users/get-information/manuel.fijan12@gmail.com`. Ako dobijemo odgovor (slika 7), backend radi. Također, IntelliJ ispisuje prikladnu poruku:

Tomcat started on port 8080 (http) with context path '/'

Started App in 16.04 seconds

```
{"userId":109,"email":"manuel.fijan12@gmail.com","firstName":"Manuel","lastName":"Fijan","userName":"manuelfijan","userType":"CLIENT","subscriptionPlan":"BRONZE","mobileNumber":"+3853643345","profilePicture":"https://lh3.googleusercontent.com/a/ACg8ocLeuD3hL5UW-KHSU5p5o2QnDCDapz3QJQC6nhMedNn_Mr3pvVc=s96-c","banned":false}
```

Slika 7

Frontend

- U terminalnu VSC-a je potrebno upisati `npm start` nakon čega se pojavljuje odgovarajuća poruka (slika 8).

```
VITE v5.4.9 ready in 1364 ms

→ Local: http://localhost:3000/
→ Network: use --host to expose
→ press h + enter to show help
```

Slika 8

- Provjera rada: U tražilicu je potrebno upisati `http://localhost:3000/` te se nakon toga otvara početna stranica aplikacije.

4. Puštanje backenda u pogon

1. Priprema Spring aplikacije

1. Izgradnja .jar datoteke:

- U root direktoriju aplikacije pokrenite:

```
./mvnw clean package
```

- Ovo će generirati .jar datoteku u direktoriju `target/`.

2. Stvaranje Dockerfilea:

- U root direktoriju aplikacije stvorite datoteku `Dockerfile`:

```
FROM openjdk:17-jdk-slim
COPY target/your-application.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

- Zamijenite `your-application.jar` nazivom stvarne .jar datoteke iz prethodnog koraka.

2. Postavljanje Google Cloud okruženja

1. Stvaranje Google Cloud projekta:

- Prijavite se na Google Cloud Console i stvorite novi projekt.

2. Omogućavanje potrebnih API-ja:

- Pokrenite sljedeće naredbe kako biste omogućili potrebne API-je:

```
gcloud services enable containerregistry.googleapis.com
gcloud services enable compute.googleapis.com
gcloud services enable run.googleapis.com
```

3. Kreiranje servisnog računa:

- Kreirajte servisni račun:

```
gcloud iam service-accounts create github-actions-sa \
--display-name="GitHub Actions Service Account"
```

- Dodijelite mu potrebne uloge:

```
gcloud projects add-iam-policy-binding <PROJECT_ID> \
--member="serviceAccount:github-actions-
sa@<PROJECT_ID>.iam.gserviceaccount.com" \
--role="roles/editor"
```

- Preuzmite JSON ključ:

```
gcloud iam service-accounts keys create key.json \
--iam-account=github-actions-sa@<PROJECT_ID>.iam.gserviceaccount.com
```

4. Dodavanje ključeva u GitHub Secrets:

- Otvorite GitHub repozitorij, idite na **Settings > Secrets and variables > Actions > Secrets**, i dodajte:
 - **GCLOUD_KEY**: Sadržaj preuzetog **key.json** ključa.
 - **GCLOUD_PROJECT**: ID vašeg Google Cloud projekta.
 - **GCLOUD_REGION**: Regiju u kojoj želite deplovati aplikaciju, npr. **eu-central1**.

3. Kreiranje GitHub Actions Workflowa

1. Kreiranje YAML datoteke:

- U rootu repozitorija kreirajte direktorij **.github/workflows**.
- Unutar tog direktorija kreirajte datoteku **deploy.yml**:

```

name: Deploy to Google Cloud

on:
  push:
    branches:
      - backend

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      # Checkout source code
      - name: Checkout code
        uses: actions/checkout@v3

      # Set up Google Cloud CLI
      - name: Set up Google Cloud SDK
        uses: google-github-actions/setup-gcloud@v1
        with:
          service_account_key: ${{ secrets.GCLOUD_KEY }}
          project_id: ${{ secrets.GCLOUD_PROJECT }}

      # Authenticate Docker with GCR
      - name: Authenticate Docker with GCR
        run: gcloud auth configure-docker

      # Build and push Docker image
      - name: Build and Push Docker image
        run: |
          docker build -t gcr.io/${{ secrets.GCLOUD_PROJECT }}}/spring-backend:latest .
          docker push gcr.io/${{ secrets.GCLOUD_PROJECT }}}/spring-backend:latest

      # Deploy to Cloud Run
      - name: Deploy to Cloud Run
        run: |
          gcloud run deploy spring-backend \
            --image=gcr.io/${{ secrets.GCLOUD_PROJECT }}}/spring-backend:latest \
            --region=${{ secrets.GCLOUD_REGION }} \
            --allow-unauthenticated

```

2. Što radi ovaj workflow:

- Na svaki **push** u backend granu, workflow:
 - Preuzima izvorni kod.
 - Postavlja Google Cloud CLI koristeći servisni račun.
 - Autentificira Docker za korištenje Google Container Registry (GCR).
 - Izgrađuje Docker image aplikacije i šalje ga u GCR.

- Deploya aplikaciju na Google Cloud Run.

4. Pokretanje workflowa

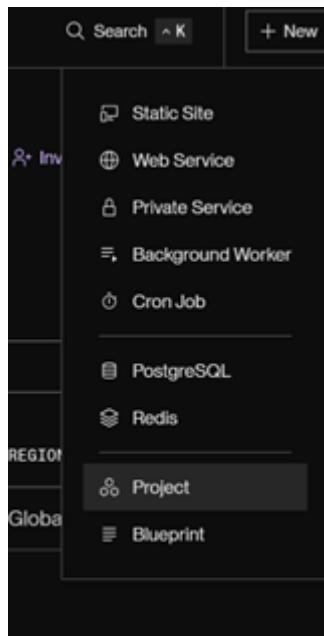
1. Kada izvršite promjenu na backend grani, workflow će se automatski pokrenuti.
2. Nakon uspješnog deploya, aplikacija će biti dostupna na URL-u koji generira Cloud Run. Ovaj URL možete pronaći u Google Cloud Console pod Cloud Run uslugama.
5. Puštanje frontenda u pogon

1. Izrada Render.com profila

- Registrirajte se na [Render](#) (preporučeno putem GitHuba).

2. Kreiranje novog projekta

1. Kliknite **New** i odaberite **Static Page** za deploy frontenda.
2. Povežite Render sa svojim GitHub računom i odaberite projekt ili direktorij koji želite deployati.



3. Konfiguracija projekta

1. Ispunite potrebne podatke:

- **Naziv projekta:** Odaberite ime za svoj deployani projekt.
- **Branch:** Odredite branch na GitHubu s kojeg želite deployati projekt.
- **Root direktorij:** Postavite putanju do glavnog direktorija projekta.
- **Build naredba:** Unesite naredbu koja će se izvršiti pri buildanju projekta, npr.:

```
npm install; npm run build
```

- **Putanja do build datoteke:** Navedite putanju do izlazne datoteke (npr. `dist/`).

You are deploying a Static Site

Source Code

ManuelFjäll / SportConnect · 1d ago Edit

Name
A unique name for your static site.

SportConnect Name is already in use

Project Optional
Add this static site to a project once it's created.

Create a new project to add this to?
You don't have any projects in this workspace. Projects allow you to group resources into environments so you can better manage related resources.

[+ Create a project](#)

Branch
The Git branch to build and deploy.

master

Root Directory Optional
If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a monorepo.

e.g. src

Build Command
Render runs this command to build your app before each deploy.

\$

Publish Directory
The relative path of the directory containing built assets to publish. Examples: ./build, dist and frontend/build.

e.g. build Required

2. Dodajte varijable okruženja (ako ih projekt koristi). Na primjer: VITE_BACKEND_API=https://example-backend.com/api

Environment

[+ Create environment group](#)

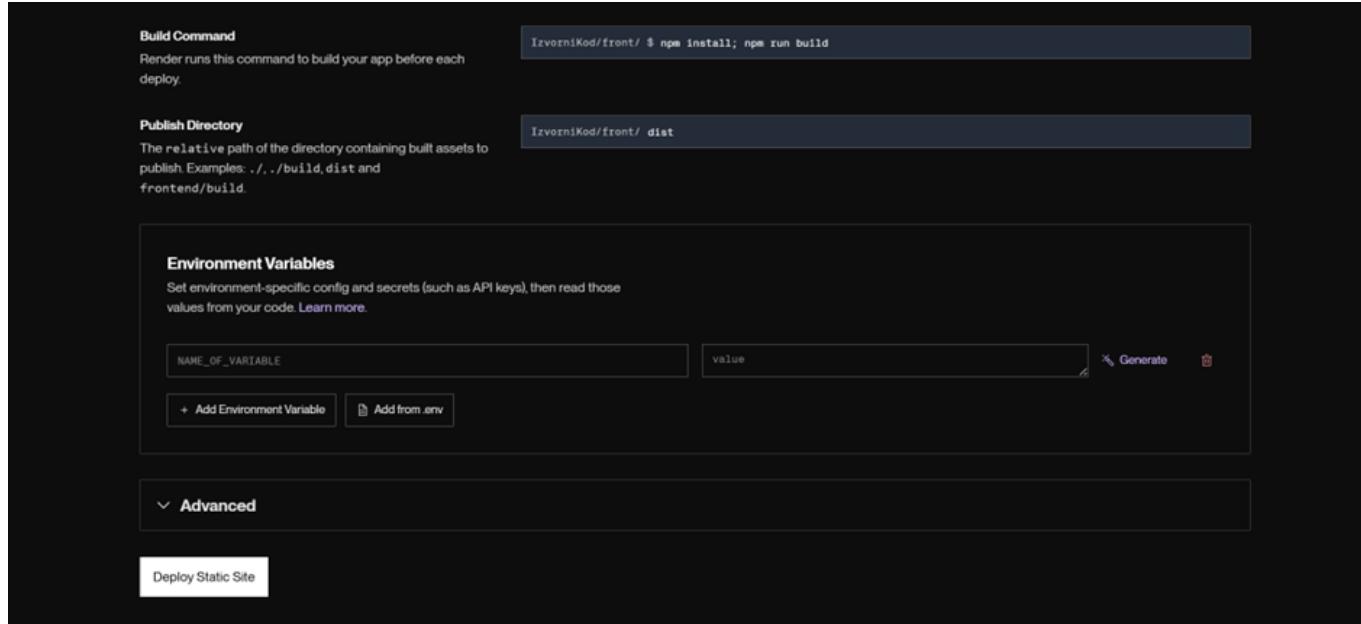
Environment Variables
Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

[Edit](#)

Key	Value
VITE_BACKEND_API

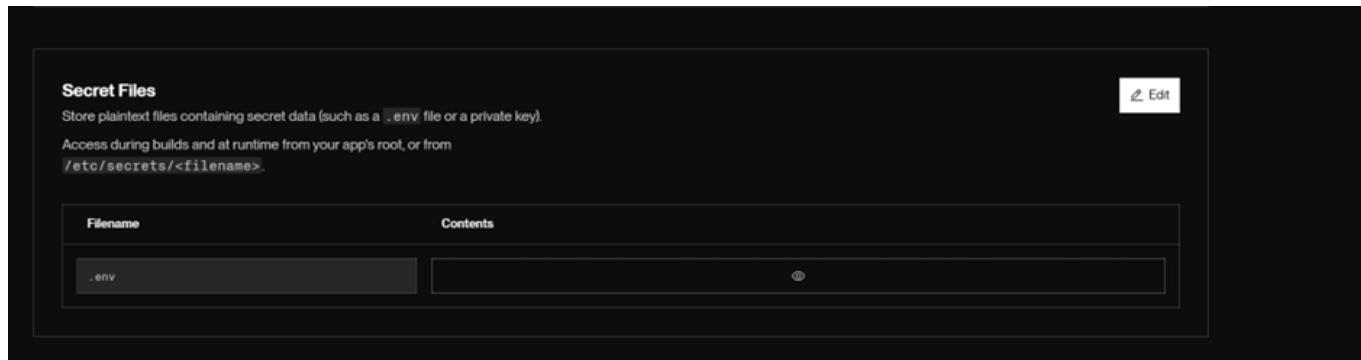
4. Deploy statične stranice

- Kliknite **Deploy Static Site**.

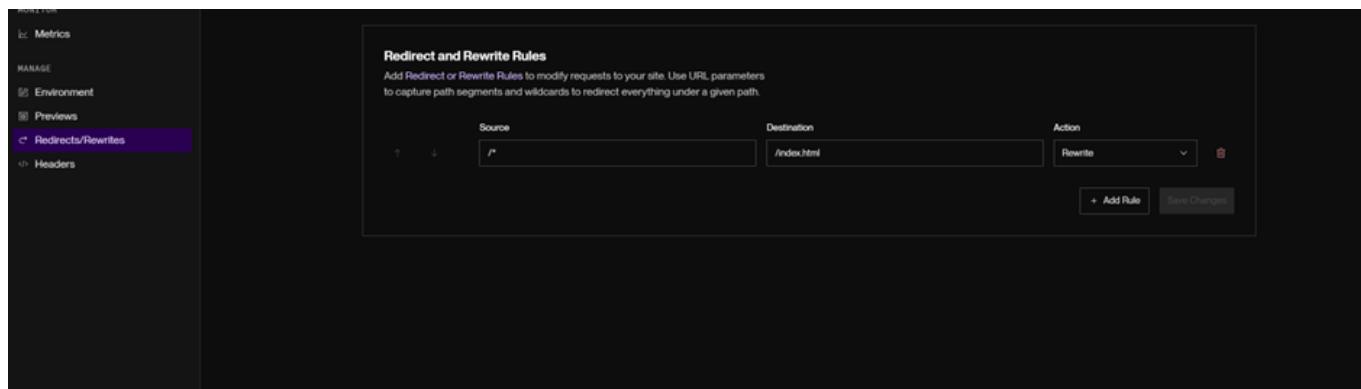


5. Naknadne konfiguracije

1. Ako projekt koristi `.env` datoteku koja je označena kao `gitignore`, dodajte je ručno u **Secret Files** u postavkama projekta na Renderu (*slika 9*).
2. Za projekte s React Routerom, navedite koji `.html` file se treba ponovno učitavati za omogućavanje pravilnog redirecta (*slika 10*).



Slika 9



Slika 10

6. Upravljanje projektom

- Za privremeno pauziranje, koristite **Suspend**.

- Za ponovno pokretanje, koristite **Resume**.
- Obje opcije dostupne su pod **Settings** u vašem projektu na Renderu.

7. Pristup aplikaciji

- Nakon uspješnog deploja, aplikaciji možete pristupiti putem URL-a koji generira Render (npr. <https://sportconnect-8b70.onrender.com>).

6. Upute za administratore

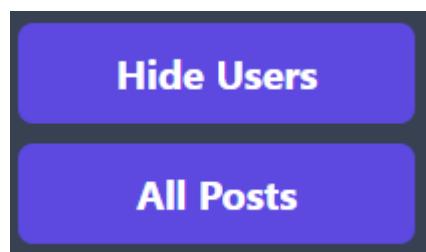
Smjernice za administratore aplikacije nakon puštanja u pogon:

- Pristup administratorskom sučelju:

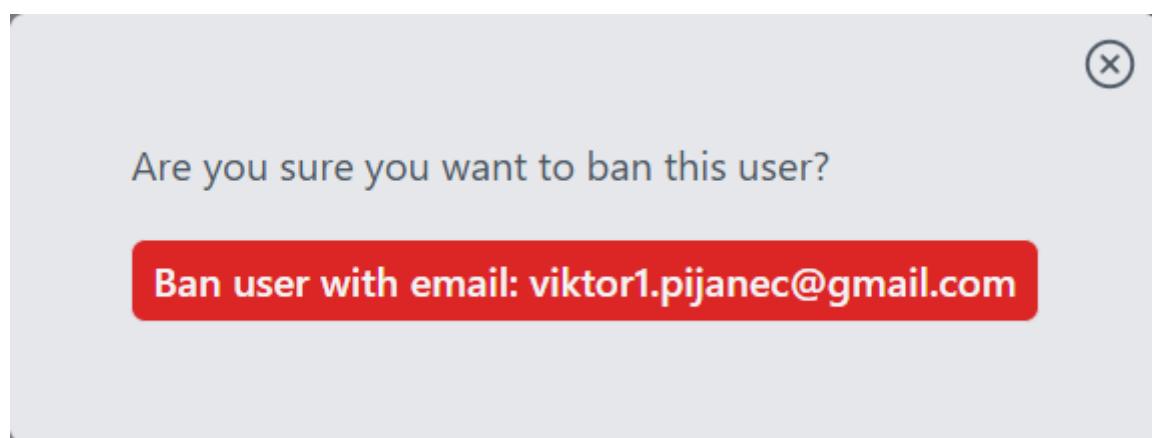
- Admin se u sustav prijavljuje sa svojim posebnim podacima (email i lozinka) preko stranice za prijavu koju koriste i ostali korisnici. Sustav prepoznaje njegove posebne podatke te ga odvodi na glavnu stranicu koja na navigacijskoj traci ima poseban odjeljak "Admin panel" (*slika 11*) gdje može obavljati sve svoje obaveze (brisanje objava, blokiranje korisnika...) (*slika 12, 13, 14*).



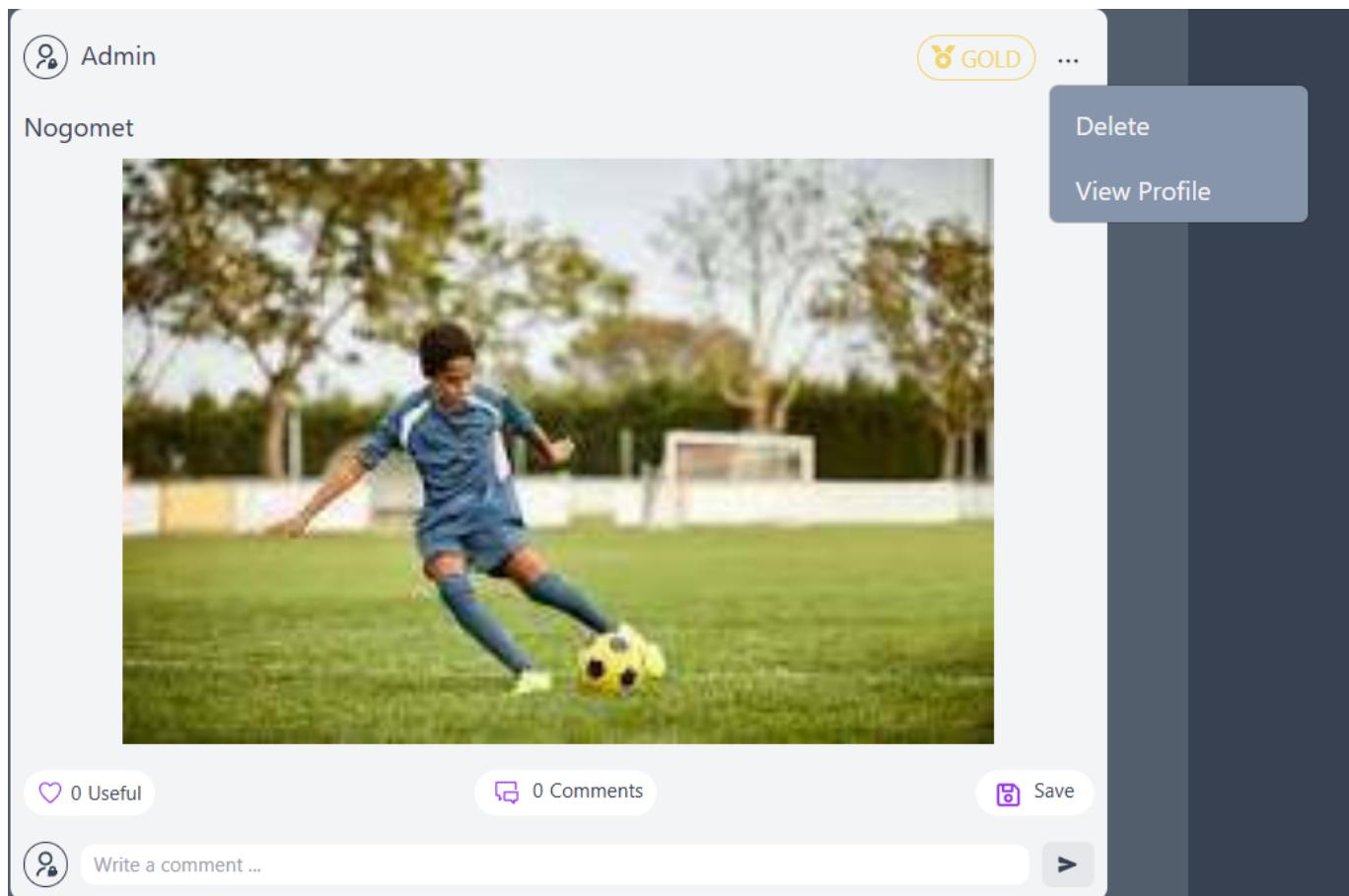
Slika 11



Slika 12



Slika 13



Slika 14

- Redovito održavanje:
 - Arhiviranje baze podataka.
 - Moguće preko Supabasea gdje se i nalazi baza podataka:
<https://supabase.com/dashboard/projects>.
 - Pregled logova.
 - Logovi se mogu provjeravati putem Google Cloud konzole do koje se dolazi putem ovog linka: <https://console.cloud.google.com/>. Potrebno je u gornjem lijevom kutu kliknuti na Navigation menu (3 crtice), odabrati Cloud Run te konačno ime aplikacije za koju želi vidjeti logove. Ovdje se nalaze sve potrebne informacije koje mu mogu pomoći u identificiraju i rješavanju problema kao i sve informacije o stanju sustava.
- Ažuriranje aplikacije (primjer: povlačenje novih verzija iz Git repozitorija i ponovno pokretanje aplikacije).
 - Sve promjene se automatski puštaju u pogon nakon što su napravljeni svi potrebni testovi te se admin ne mora brinuti o tome.

Opis pristupa aplikaciji na javnom poslužitelju

Kako pristupiti aplikaciji

Korisnici mogu pristupiti aplikaciji putem internetskog preglednika na sljedećoj adresi:
<https://sportconnect-8b7o.onrender.com/>

Aplikacija zahtijeva aktivnu internetsku vezu za pravilno funkcioniranje.

Pristup administratorskom sučelju

Pristup administratorskom sučelju moguć je samo putem prijave s valjanim administratorskim korisničkim imenom i lozinkom.

Administratori trebaju unijeti svoje vjerodajnice na stranici za prijavu kako bi pristupili naprednim funkcionalnostima aplikacije.

Ograničenja

- **Pristup internetu:** Aplikacija zahtijeva stabilnu internetsku vezu za rad jer se nalazi na javnom poslužitelju.
 - **Administratorske ovlasti:** Pristup administratorskim funkcionalnostima je ograničen samo na ovlaštene korisnike s ispravnim vjerodajnicama.
-

Pristup aplikaciji Dokumentirajte postupak i pružite jasne smjernice za korištenje aplikacije na javnom poslužitelju.

- Navedite ograničenja!
- U uputama obuhvatite kako korisnici mogu pristupiti aplikaciji putem internetskog preglednika.
- Priložite korake za pristup administratorskom sučelju ako je primjenjivo.

Osvrt na izradu projektnog zadatka

Izrada projektnog zadatka trajala je ukupno približno 13 tjedana. Tijekom tog razdoblja, neki članovi tima su već u prvom tjednu rada ostvarili oko 60 sati rada, što je odraz predanosti i intenzivnog angažmana na projektu. Ovaj trud rezultirao je izradom kvalitetne i funkcionalne aplikacije, unatoč izazovima i ograničenjima.

Jedan od najvećih tehničkih izazova bio je nedostatak prethodnog iskustva svih članova tima u razvoju aplikacija. Korištene tehnologije, uključujući React (TypeScript) za frontend, Spring Boot za backend, PostgreSQL za bazu podataka, kao i alate poput Stripe-a za plaćanja i Pusher-a za real-time komunikaciju, bile su potpuno nove za tim. Rješavanje tehničkih problema zahtijevalo je konzultacije s kolegama, upotrebu online resursa poput YouTube tutorijala i umjetne inteligencije, te konzultacije s asistentima i demonstratorima. Upravo su ove metode omogućile uspješno svladavanje svih izazova.

Tijekom izrade projekta, tim je stekao brojna nova znanja:

- Razumijevanje razvoja aplikacija koristeći slojevitu arhitekturu (Model-Repository-Service-Controller).
- Implementacija real-time komunikacije pomoću Pusher-a.
- Korištenje REST API-ja za povezivanje frontenda i backenda.
- Upravljanje bazama podataka pomoću PostgreSQL-a i integracija s Hibernate ORM-om.

- Sigurno procesiranje plaćanja koristeći Stripe i OAuth 2.0 protokol za autentifikaciju.

Za bržu i kvalitetniju realizaciju projekta, bilo bi korisno unaprijed imati osnovno znanje o korištenim tehnologijama, posebno u područjima integracije plaćanja i real-time komunikacije. Također, iskustvo s arhitektonskim pristupima poput mikrouslužne arhitekture moglo bi olakšati planiranje i implementaciju sustava.

Timska dinamika bila je iznimno pozitivna, zahvaljujući složnosti samog tima i preuzetoj inicijativi vođe tima. Članovi tima razvili su snažan osjećaj zajedništva i suradnje, što je značajno pridonijelo uspješnoj realizaciji projekta.

Perspektiva za nastavak rada u projektnoj grupi uključuje:

- Optimizaciju postojećih modula za još veću skalabilnost i učinkovitost.
- Proširenje aplikacije novim značajkama, poput sustava preporuka sadržaja temeljenog na umjetnoj inteligenciji.

Ukupno gledajući, projektni zadatak pokazao se kao izvrstan primjer praktične primjene teorijskog znanja, a tim je uspješno stekao vrijedne vještine i iskustvo u razvoju kompleksnih sustava.

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napravljen predložak	Manuel Fijan	21.10.2024
0.2	Dodan novi sastanak	Manuel Fijan	21.10.2024
0.3	Dodani novi sastanci	Manuel Fijan	24.10.2024
0.4	Dodan opis projektnog zadatka	Klara Katić	26.10.2024
0.5	Dodan novi sastanak	Manuel Fijan	31.10.2024
0.6	Napravljena analiza zahtjeva	Hana Ćerić	15.11.2024
0.7	Napravljena specifikacija zahtjeva sustava	Hana Ćerić	15.11.2024
0.8	Dodani novi sastanci	Manuel Fijan	14.11.2024
0.9	Dodani novi sastanci	Manuel Fijan	17.12.2024
1.0	Izmjene u Analizi zahtjeva	Klara Katić	02.01.2025
1.1	Dodani novi sastanci	Manuel Fijan	16.01.2025
1.2	Dodan dio testova komponenti	Manuel Fijan	19.01.2025
1.3	Dovršeno ispitivanje komponenata	Manuel Fijan	19.01.2025
1.4	Dovršeno ispitivanje sustava	Hana Ćerić	19.01.2025
1.5	Izmjene u specifikacijama zahtjeva sustava	Klara Katić	20.01.2025
1.6	Napravljen odjeljak puštanje u pogon	Manuel Fijan, Luka Đuretić	20.01.2025
1.7	Opisane tehnologije za implementaciju aplikacije	Hana Ćerić	20.01.2025

Rev.	Opis promjene/dodataka	Autori	Datum
1.8	Napisan opis pristupa aplikaciji na javnom poslužitelju	Hana Ćerić	21.01.2025
1.9	Opisana arhitektura i dizajn sustava	Klara Katić, Hana Ćerić	22.01.2025
2.0	Opisana arhitektura komponenata i razmještaja	Klara Katić	22.01.2025
2.1	Napisan zaključak i budući rad	Klara Katić	22.01.2025
2.2	Popisana korištena literatura	Hana Ćerić	22.01.2025

Popis svih referenci i literature koja je pomogla pri ostvarivanju projekta:

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/> books/SE
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>
7. Spring Boot GitHub, <https://github.com/spring-projects/spring-boot>
8. Spring Boot, <https://spring.io/guides/gs/spring-boot>
9. Spring Boot Tutorial, <https://www.geeksforgeeks.org/spring-boot/>
10. Spring Boot Tutorial, YouTube, <https://youtu.be/9SGDpanrc8U?si=BUfhjmy2SyxhHT3S>
11. Learn React, React, <https://react.dev/learn>
12. React Tutorial, W3Schools, <https://www.w3schools.com/REACT/DEFAULT.ASP>
13. React Tutorial, YouTube, <https://youtu.be/SqcY0GIETPk?si=ID2WJu9vJ8U0hwrg>
14. Selenium, https://www.selenium.dev/documentation/test_practices/overview/
15. Java Testing with Selenium Course, YouTube, <https://youtu.be/QQliGCtqD2w?si=NQ8jL8mv0qRob30y>

Dnevnik sastajanja Kontinuirano osvježavanje

X. sastanak

- Datum: 17. listopada 2024.
- Prisustvovali:
- Teme sastanka:
 - opis prve teme
 - LOŠE dogovor načina komunikacije
 - BOLJE uspostavljena Discord grupa svih članova i demonstratora
 - opis druge teme
 - poveznica na issue (kratki link npr. #22 vidi [link](#))

1. sastanak

- Datum: 17. listopada 2024.
- Prisustvovali: V.Pijanec, M.Fijan, L.Đuretić, L.Zuanović, V.Knežević, K.Katić, H.Ćerić
- Teme sastanka:
 - upoznavanje s radom na projektu
 - rasprava o temi aplikacije

- BOLJE: napravljena grupa za razmjenu poruka, detaljno smo razradili plan rada i ideju aplikacije

- issue

2. sastanak

- Datum: 22. listopada 2024.
- Prisustvovali: V.Pijanec, M.Fijan, L.Đuretić, L.Zuanović, V.Knežević, K.Katić, H.Ćerić
- Teme sastanka:
 - rasprava o funkcijskim i nefunkcijskim zahtjevima

3. sastanak

- Datum: 24. listopada 2024.
- Prisustvovali: V.Pijanec, M.Fijan, L.Đuretić, L.Zuanović, V.Knežević, K.Katić, H.Ćerić
- Teme sastanka:
 - prezentiranje funkcijskih i nefunkcijskih zahtjeva
 - BOLJE: napisano dovoljno zahtjeva
 - LOŠE: potrebno je eksplicitno napisati što koji sudionik može raditi

4. sastanak

- Datum: 24. listopada 2024.
- Prisustvovali: V.Pijanec, M.Fijan
- Teme sastanka:
 - rasprava o tome kako ćemo oblikovati backend i početak rada
 - BOLJE: utrvdili smo da ćemo backend oblikovati na temelju Controller - Service - Repository arhitekture
 - issue

5. sastanak

- Datum: 31. listopada 2024.
- Prisustvovali: L.Zuanović, M.Fijan
- Teme sastanka:
 - prezentiranje uml dijagrama i dijelova aplikacije koje smo napravili do sada
 - BOLJE: login i registracija su već bili u funkciji
 - LOŠE: potrebno napraviti više dijagrama i detaljnije ih opisati
 - issue

6. sastanak

- Datum: 7. studenog 2024.
- Prisustvovali: M.Fijan, L.Đuretić, L.Zuanović, K.Katić, H.Ćerić
- Teme sastanka:
 - pokazivanje rada povezanog frontenda i backenda
 - BOLJE: zaključili da frontend i backend uspješno komuniciraju

7. sastanak

- Datum: 14. studenog 2024.

- Prisustvovali: V.Pijanec, M.Fijan, L.Đuretić, L.Zuanović, V.Knežević, K.Katić, H.Ćerić
- Teme sastanka:
 - prezentiranje funkcionalnosti aplikacije za prvu predaju
 - nedostaci u dokumentaciji
 - BOLJE: pokazivanje deployanog frontenda i backenda
 - LOŠE: potrebno izbrisati komentare na wikiju i neke obrasce preoblikovati
 - issue

8. sastanak

- Datum: 12. prosinca 2024.
- Prisustvovali: V.Pijanec, M.Fijan, L.Zuanović, H.Ćerić
- Teme sastanka:
 - diskusija o greškama na prvoj predaji i načinu bodovanja
 - BOLJE: velika aktivnost tijekom prvog dijela rada na projektu, funkcionalnosti aplikacije
 - LOŠE: greške u dokumentaciji

9. sastanak

- Datum: 17. prosinca 2024.
- Prisustvovali: V.Pijanec, M.Fijan, L.Đuretić, L.Zuanović, K.Katić,
- Teme sastanka:
 - dogovor oko nastavka rada na projektu
 - BOLJE: postavljeni ciljevi koje moramo dovršiti za drugu predaju

10. sastanak

- Datum: 9. siječnja 2025.
- Prisustvovali: V.Pijanec, L.Đuretić, L.Zuanović
- Teme sastanka:
 - pokazivanje funkcionalnosti napravljenih do sada demosu
 - BOLJE: napravljen vrlo veliki dio funkcionalnosti

11. sastanak

- Datum: 9. siječnja 2025.
- Prisustvovali: V.Pijanec, M.Fijan, L.Đuretić, L.Zuanović, V.Knežević, H.Ćerić
- Teme sastanka:
 - dogovor oko završetka projekta
 - BOLJE: predstavili smo što smo napravili tijekom praznika i raspodijelili posao koji je potreban za završetak projekta

12. sastanak

- Datum: 16. siječnja 2025.
- Prisustvovali: V.Pijanec, M.Fijan, L.Đuretić, V.Knežević, K.Katić, H.Ćerić
- Teme sastanka:
 - pokazivanje alfa verzije aplikacije
 - BOLJE: gotovo cijela aplikacija je već bila gotova

13. sastanak

- Datum: 23. siječnja 2025.
- Prisustvovali: V.Pijanec, M.Fijan, L.Đuretić, L.Zuanović, K.Katić, H.Ćerić
- Teme sastanka:
 - završna pitanja o dovršavanju projekta

Plan rada

Tr	Zadatak	Status	Odgovorne osobe	Tjedan 1-2	Tjedan 3-4	Tjedan 5-10	Tjedan 11-12	Tjedan 13
	Planiranje	Završeno	Cijeli tim	Određena tema, raspodjela poslova	Komunikacija na sastancima	Komunikacija na sastancima	Komunikacija na sastancima	
	Dizajn	Završeno	Klara Katić	Napravljen u Canvi	Napravljen dizajn i logo	Dovršeno	Dovršeno	
	Dokumentiranje projekta	Završeno	Hana Ćerić, Klara Katić	Napravljen opis projekta	Napravljene točke 1/2/3/4	Dovršeni dijagrami do 4. točke	Napravljeni ostali dijagrami i ostatak dokumentacije	Dovršeno
	Frontend razvoj	Završeno	Luka Đuretić, Luka Zuanović, Vid Knežević	Proces učenja	Napravljen sign in i početni djelovi	Napravljen glavni dio za prvu predaju	Vježba aplikacije dovršena	Usavršavanje
	Baza podataka	Završeno	Hana Ćerić, Manuel Fijan	/	Definiran izgled baze u erdplus-u	Izrađena baza podataka	Punjjenje baze	Dovršeno
	Backend razvoj	Završeno	Manuel Fijan, Viktor Pijanec	Proces učenja	Rad na funkcionalnosti sign in-a	Provjera tokena pri sign in-u	Povezivanje sa vanjskim	Usavršavanje
	Deploy	Završeno	Manuel Fijan, Viktor Pijanec	/	Proces učenja	Prvi deploy napravljen	Dovršen Deploy	Dovršeno
	Testiranje	Završeno	Hana Ćerić	/	/	Proces učenja	Odradeno testiranje	Dovršeno

Gantogram za prikaz rada na projektu

Tablica aktivnosti

Kontinuirano osvježavanje

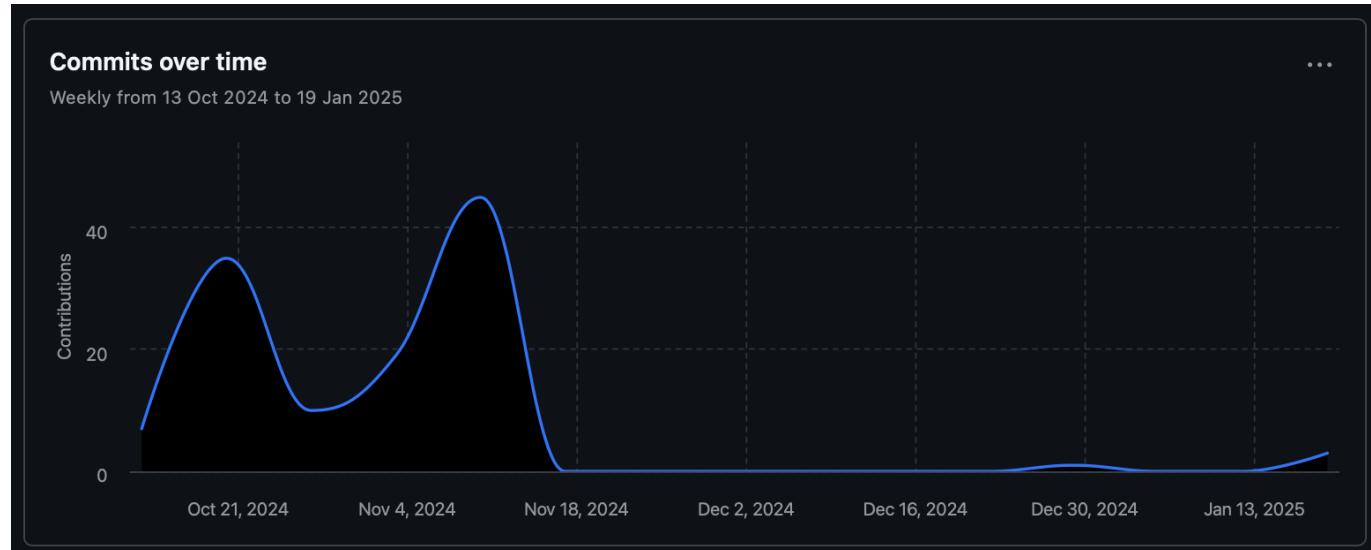
Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti. Potrebno je navesti koliko je sati koja osoba uložila u pojedinu komponentu, možete oblikovati tablicu ili ispisati za svaku osobu.

Aktivnost	Hana Ćerić	Luka Đuretić	Manuel Fijan	Klara Katić	Vid Knežević	Viktor Pijanec	Luka Zuanović
Upravljanje projektom	0	0	0	0	0	0	5
Opis projektnog zadatka	0	0	0	1	0	0	0
Funkcionalni zahtjevi	1	0.5	0	2	0	0	0
Opis pojedinih obrazaca	10	0	0	0	0	0	0
Dijagram obrazaca	0	1	0	4	0	0	0
Sekvensijski dijagrami	0	0	0	4	0	0	0
Opis ostalih zahtjeva	4	0	0	1	0	0	0
Arhitektura i dizajn	7	0	2	12	0	0	0
Baza podataka	20	0	6	0	0	2	0
Dijagram razreda	0	0	0	0	0	0	0
Dijagram stanja	0	0	0	6	0	0	0

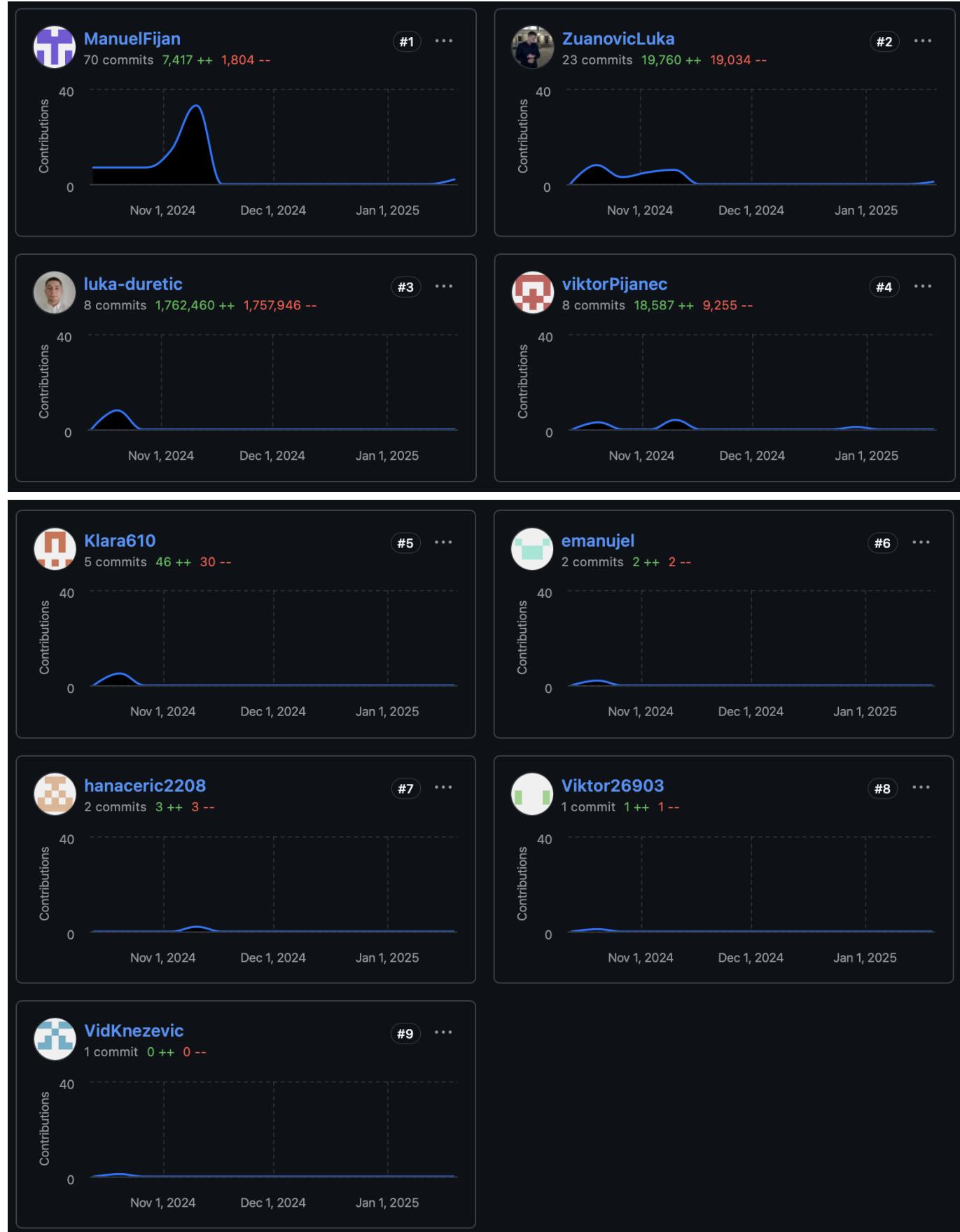
Aktivnost	Hana Ćerić	Luka Đuretić	Manuel Fijan	Klara Katić	Vid Knežević	Viktor Pijanec	Luka Zuanović
Dijagram aktivnosti	0	0	0	6	0	0	0
Dijagram komponenti	0	0	0	5	0	0	0
Tehnologije i alati	3	0	0	7	0	0	0
Ispitivanje programa	12	0	4	0	0	3	0
Dijagram razmještaja	0	0	0	5	0	0	0
Upute puštanja u pogon	0.5	1.5	3	0	0	0	0
Dnevnik sastajanja	0	0	1	0	0	0	0
Zaključak i budući rad	0	0	0	1	0	0	0
Popis literature	0.5	0	0	0	0	0	0
Izrada baze podataka	0	0	2	0	0	1	0
Izrada prezentacije	0	0	0	3	0	0	0
Izrada komponenata na frontend-u	0	87	0	0	82	1	97
Responzivnost frontend-a	0	42	0	0	18	0	55
Povezivanje frontend-a i backend-a	0	39	2	0	40	4	45
Povezivanje sa vanjskim servisima	0	0	2	0	0	5	5
Testiranje ispravnosti endpoint-a	0	0	4	0	0	5	0
Izrada modela, servisa i kontrolera	0	0	154	0	0	121	0
Ukupno sati	58	168	180	57	140	142	207

Dijagram pregleda promjena

Dijagram cjelokupnog tima:



Ostali dijagrami svakog pojedinaca;



Dijagram prikazuje distribuciju promjena u repozitoriju projekta tijekom razdoblja od 13+ tjedana. Svaki član tima doprinosio je različitim dijelovima projekta, a dijagram jasno pokazuje raspodjelu commitova i promjena. Ključne faze projekta (razvoj, testiranje) vidljive su kroz intenzivnije aktivnosti.

Ključni izazovi i rješenja

Izrada projektnog zadatka trajala je približno 13 tjedana, tijekom kojih su članovi tima intenzivno radili na razvoju aplikacije. Ovaj period uključivao je planiranje, razvoj, testiranje i dokumentiranje.

Tijekom razvoja projekta, prepoznati su sljedeći tehnički izazovi:

- Nedostatak iskustva s novim tehnologijama:

Članovi tima nisu imali prethodnog iskustva s tehnologijama poput Reacta, Spring Boota, Stripea i Pushera. Rješenje: Korišteni su online resursi poput YouTube tutorijala, konzultacije s asistentima i kolegama te primjena umjetne inteligencije za generiranje rješenja. Praktičnim radom tim je postupno usvojio potrebne vještine.

- Integracija real-time komunikacije (Pusher):

Implementacija real-time značajki za chat sustav zahtjevala je razumijevanje WebSocket komunikacije i korištenje Pusher servisa. Rješenje: Dokumentacija Pushera proučavana je detaljno, a tim je testirao različite scenarije kako bi osigurao stabilnost i pouzdanost sustava.

- Usklađivanje timskog rada:

Rad na različitim modulima uzrokovao je povremene konflikte pri spajanju koda. Rješenje: Uvedena je praksa redovitih sastanaka tima i korištenje Git grananja kako bi se smanjile greške prilikom integracije koda.

Tijekom projekta, tim je naučio:

- Implementirati slojevitu arhitekturu aplikacije (Model-Repository-Service-Controller).
- Upravljati bazama podataka i koristiti ORM alate poput Hibernatea.
- Razvijati real-time funkcionalnosti i sigurne sustave za autentifikaciju.
- Raditi u timu koristeći alate za verzioniranje koda poput Git-a.

PROGI PREZENTIRANJE;

AKTORI;

aktivni; korisnici (klijenti, partneri), admin pasivni; baza podataka FUNKCIJSKI ZAHTJEVI;

pristup informacija same aplikacije -> svi aktivni korisnici (klijenti, partneri i admini) stvaranje profila/prijava -> klijenti i partneri spremanje i označavanje objava -> klijenti i partneri komunikacija s ostalim korisnicima -> klijenti i partneri spremanje i označavanje drugih objava -> klijenti i partneri odabiranje i otkazivanje ranga za plaćanje -> klijenti i partneri (brončanog i srebrnog ranga) update -> klijenti i partneri (brončanog i srebrnog ranga) kreiranje i uređivanje plaćenih objava -> partneri kreiranje i uređivanje besplatnih objava -> partneri isplate -> partneri odabiranje besplatnog ranga -> klijenti adminske brisanje sadržaja -> admin adminske filtriranje rangova -> admin NEFUNKCIJSKI ZAHTJEVI;

autentifikacija plaćanja prihvatanje odredbi korištenja aplikacije provjera licence dostupnost (automatski emailovi) sigurnost (OAuth 2.0) Naša ideja je napraviti aplikaciju koja se zove "SportConnect". Već smo izgenerirali sami logo aplikacije sa sloganom SC - Nourish & Thrive. Sama aplikacija je principa društvene mreže. Aplikacija se sastoji od korisnika takozvanih klijenata i partnera.

Aplikacija će izgledati tako da u prvoj stranici postoji dio "About" - gdje se može detaljnije vidjeti svrha aplikacije te će postojati naravno "Sign up" za nove korisnike i "Log in" za postojeće. Front end će se raditi u Reactu, backend u Springu, a dizajn u Figmi. Kada novi korisnik ulazi u sign up dio, to jest u kreiranje svog profila može birati opciju - partner/klijent. Sama podjela će jasno biti napravljena u bazi podataka PostGreSQL.

Kada novi korisnik odluči biti partner prvo mora pristati na sva pravila aplikacije, a zatim će sami admini procjenjivati njegov rang u aplikaciji (zlato, srebro, bronca). Procjenjivat će sve što na početku partner objavljuje; slike, planove, savjete. Klijenti i drugi partneri označavaju njegove objave s "koristan", također mogu i spremati dijelove blogova, komentirati ih što pomaže samom rangu partnera (raste)... Nakon što je novi korisnik otvorio svoj profil, dobiva notifikaciju potvrde na svoj email. Ta funkcionalnost će se napraviti preko Firebase Cloud Messaginga ili preko FreeChata. Također će se tako i obavljati komunikacija između partnera i klijenta. Također sve uspješno obavljene uplate i isplate partnera dolaze uz potvrdu na emailu.

Kada novi korisnik odluči biti klijent nakon što pristane na sva pravila aplikacije može birati rang aplikacije kojem želi prisustovati. Pristup svim brončanim dijelovima aplikacije je 15€, pristup srebrnim i brončanim blogovima je 25€, a pristup svim blogovima je 35€ (cijene su fiktivne i plaćaju se na mjesecnoj bazi). Postojat će i besplatni sadržaj od partnera koji još nisu odobreni i od partnera koji biraju da im je određeni dio sadržaja besplatan. Klijent obavlja plaćanje preko aplikacija PayPal ili Stripe. Autentifikacije za plaćanje će se ostvariti preko OAuth2.0 ili preko Firebase Authentication, a notifikacije potvrde plaćanja (i ostale nužne notifikacije) će dobivati klijent preko svog upisanog emaila, a ta se funkcionalnost naravno obavlja preko istih aplikacija kao u partnerskim korisnicima. Također je identična funkcionalnost slanja poruka i komuniciranja kao kod partnera. Osim toga klijent ima priliku otkazati plaćeni plan kada god to odluči, a 2 dana prije isteka plana i ponovnog automatskog plaćanja dobiva obavijest o navedenom.

Sami admini kontroliraju cijelu aplikaciju koristeći sve od navedenih tehnologija za sigurno i korisno okruženje.

S ovom aplikacijom potičemo razmjenjivanje znanja među ljudima sličnih rangova, ali još bitnije stvaramo pristupačniju zajednicu za ljudе koji žele stvoriti zdrave navike pomoću inspiracija, savjeta i umrežavanja.