Programsko inzenjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Gym Progress Tracking App

Tim: <TG04.2>

CodeMonkeys

Članovi tima:

- Matija Martinović
- Petar Babić
- Leonardo Cigula
- Luka Kordić
- Ema Adamec
- Sonja Ćiković
- Gabrijela Klepec

1. Opis projektnog zadatka

Nakon analize postojećih rješenja za praćenje napretka u fitnessu, primijetili smo da trenutno ne postoji besplatna, ali ni kvalitetna plaćena verzija koja bi kvalitetno zadovoljila potrebe korisnika, zbog čega smo osmislili ideju za GYM Progress Tracker.

Omogućili bi praćenje napretka u teretani te pružali korisnicima intuitivne i personalizirane funkcionalnosti, s posebnim fokusom na progressive overload i RPE skalu. Progressive overload je ključan princip u teretani jer osigurava kontinuiran napredak u snazi i mišićnoj masi. Povećavanjem težine, broja ponavljanja ili intenziteta vježbi, tijelo se stalno prilagođava većim opterećenjima, što vodi do boljih rezultata. Praćenjem RPE skale te progressive overloada kroz našu aplikaciju, korisnici bi mogli precizno vidjeti gdje napreduju i identificirati područja koja trebaju dodatno razmatranje u vezi plana treninga. Ovakav uvid omogućava planiranje budućih treninga s ciljem optimizacije rezultata. Korisnicima bi bio omogućen upload slike barkoda pomoću vanjskog servisa dohvatiti informacije o makronutrijentima i kalorijama neke namirnice kako bi lakše bilježili dnevni unos.

Postojale bi tri razine korisnika (za svaku je potrebna registracija i autorizacija putem OAuth 2.0 standarda): obični korisnik, privatni trener i administrator.

2. Analiza zahtjeva

Funkcionalni zahtjevi

Obični korisnik

ID zahtjeva	Opis	Prioritet	Izvor	Kriterij prihvaćanja

FO-001	Korisnik unosi osobne podatke (težina, visina) i ciljeve pri prijavi	Visok	Zahtjev dionika	Korisnik može uspješno unijeti podatke o težini, visini i ciljevima prilikom registracije
FO-002	Postavljanje trening plana: korisnik kreira plan s brojem serija, ponavljanjima i RPE za svaku vježbu	Visok	Zahtjev dionika	Korisnik može kreirati trening plan s definiranim brojem serija, ponavljanjima i RPE za svaku vježbu
FO-003	Praćenje napretka (progressive overload): pregled tjednog napretka i povećanja težina po vježbama	Visok	Povratne informacije korisnika	Korisnik može vidjeti tjedni napredak i povećanja težina kroz vrijeme za pojedine vježbe
FO-004	Tutorijali za vježbe s detaljnim opisima izvođenja	Srednji	Dokument zahtjeva	Korisnik može pregledati tutorijale s opisima izvođenja vježbi
FO-005	Praćenje treninga uživo: unos rezultata (težina, ponavljanja, RPE) uz timer za odmor	Visok	Zahtjev dionika	Korisnik može unositi podatke o težini, ponavljanjima i RPE tijekom treninga te koristiti timer za odmor
FO-006	Praćenje kalorija i makronutrijenata: unos obroka i praćenje kalorija i makronutrijenata	Srednji	Povratne informacije korisnika	Korisnik može unositi dnevne obroke te pratiti unos kalorija i makronutrijenata
FO-007	Postavke korisničkog računa: uređivanje osobnih podataka (težina, visina) i praćenje napretka	Srednji	Zahtjev dionika	Korisnik može uređivati osobne podatke i vidjeti statistike napretka
FO-008	Analitika osobnog napretka: prikaz grafova	Srednji	Dokument zahtjeva	Korisnik može vidjeti grafove

	i statistika o napretku			i statistike o napretku u odnosu na postavljene ciljeve
FO-009	Mogućnost angažiranja privatnog trenera	Srednji	Zahtjev dionika	Korisnik može odabrati i angažirati privatnog trenera unutar aplikacije
FO-010	Ostavljanje recenzije o privatnom treneru nakon suradnje	Nizak	Povratne informacije korisnika	Korisnik može ostaviti recenziju o treneru nakon završene suradnje
FO-011	Feedback tijekom suradnje s trenerom: unosi dojmove o treningu, prehrani i osjećaju	Srednji	Zahtjev dionika	Korisnik može unositi dojmove o svakodnevnim aspektima treninga i prehrane tijekom suradnje s trenerom
FO-012	Ostavljanje poruke treneru nakon treninga	Nizak	Povratne informacije korisnika	Korisnik može poslati poruku treneru nakon završenog treninga

---

Privatni trener

ID zahtjeva	Opis	Prioritet	Izvor	Kriterij prihvaćanja
FPT-001	Trener uređuje individualne planove treninga za klijente	Visok	Dokument zahtjeva	Trener može kreirati i prilagođavati planove treninga za svakog klijenta
FPT-002	Trener uređuje individualne prehrambene planove za klijente	Visok	Zahtjev dionika	Trener može dodijeliti prilagođene prehrambene planove s preporučenim obrocima i unosom kalorija

FPT-003	Praćenje napretka klijenata: pregled napretka po treninzima i prehrani	Srednji	Dokument zahtjeva	Trener ima uvid u klijentov napredak i može ga pratiti po treninzima i prehrani
FPT-004	Postavljanje ciljeva za klijente i prilagođavanje planova prema ciljevima	Visok	Zahtjev dionika	Trener može postaviti ciljeve za klijente i prilagoditi planove kako bi im pomogao u postizanju tih ciljeva
FPT-005	Trener prima summary o treningu i dojmovima klijenata	Srednji	Povratne informacije korisnika	Trener prima dnevni summary o klijentovom treningu, prehrani i općem dojmu
FPT-006	Trener promovira svoje usluge putem aplikacije	Nizak	Dokument zahtjeva	Trener može objavljivati promocijske sadržaje ili profile u aplikaciji

---

Administrator

ID zahtjeva	Opis	Prioritet	Izvor	Kriterij prihvaćanja
FA-001	Admin upravlja vježbama: dodavanje novih vježbi ili odobravanje korisničkih prijedloga	Srednji	Zahtjev dionika	Admin može dodavati ili odobravati nove vježbe koje korisnici predlažu
FA-002	Admin kreira predefinirane planove treninga dostupne korisnicima	Srednji	Dokument zahtjeva	Admin može dodavati unaprijed definirane planove koje korisnici mogu koristiti
FA-003	Moderacija sadržaja: odobravanje ili odbijanje novih vježbi i sadržaja	Visok	Dokument zahtjeva	Admin može moderirati sadržaj, uključujući odobravanje ili odbijanje novih

				vježbi i drugih prijedloga

Ostali zahtjevi

ID zahtjeva	Opis	Prioritet
NF-1	Performanse - vrijeme odziva na upite unutar aplikacije treba biti optimalno kako bi se osigurala brza i ugodna interakcija korisnika	Visok
NF-2	Dostupnost - Aplikacija treba omogućiti deploy s ažuriranjima putem CI/CD pipelinea (GitHub Actions) kako bi se osigurala kontinuirana isporuka	Visok
NF-3	Enkripcija podataka - Osigurati enkripciju podataka u prijenosu korištenjem HTTPS-a i SSL/TLS protokola	Visok
NF-4	Autentifikacija - Korištenje OAuth2.0 standarda za provjeru identiteta korisnika	Visok
NF-5	Autorizacija - Osigurati da korisnici imaju pristup samo podacima i funkcionalnostima za koje su ovlašteni	Visok
NF-6	Responzivni dizajn - Aplikacija treba biti optimizirana za različite veličine ekrana (desktop, tablet, mobitel)	Srednji
NF-7	Internacionalizacija - Pravilno rukovanje različitim formatima datuma, vremena i brojeva za podršku međunarodnim korisnicima	Nizak
NF-8	API kompatibilnost - Pravilna kompatibilnost API-ja za dohvaćanje makronutrijenata, kalorija i ostalih bitnih informacija o namirnicama	Srednji
NF-9	GDPR - Osigurati usklađenost s GDPR regulativama, uključujući pravo na zaborav i prenosivost podataka	Visok
NF-10	Zaštita osobnih podataka - Implementirati stroge kontrole pristupa osobnim podacima korisnika	Visok
NF-11	Centralizirani logging - Implementirati sustav za centralizirano prikupljanje i analizu logova radi boljeg praćenja performansi i problema	Srednji
NF-12	Vrijeme učitavanja stranice - Početna stranica treba se učitati unutar 3 sekunde na prosječnoj mobilnoj vezi (3G)	Visok
NF-13	Optimizacija resursa - Implementirati lazy loading za slike i druge teške resurse radi optimizacije učitavanja	Srednji
NF-14	File Upload - Omogućiti siguran upload datoteka putem Amazon S3	Srednji
NF-15	Slanje emaila - Omogućiti slanje emailova korisnicima za	Nizak

	različite notifikacije	
NF-16	Potvrda točnosti unesenog e-maila - Slanje potvrde korisnicima radi verifikacije e-mail adrese	Visok

Dionici

Obični korisnici

- Korisnici aplikacije koji traže personalizirane planove treninga, prate napredak i prehranu te koriste aplikaciju za postizanje svojih ciljeva u fitnessu.
- Relevantni zahtjevi: FO-001 do FO-012.

Privatni treneri

- Treneri koji kreiraju i prilagođavaju treninge i prehrambene planove za svoje klijente, prate njihov napredak te komuniciraju i pružaju podršku klijentima.
- Relevantni zahtjevi: FPT-001 do FPT-006.

Administratori aplikacije

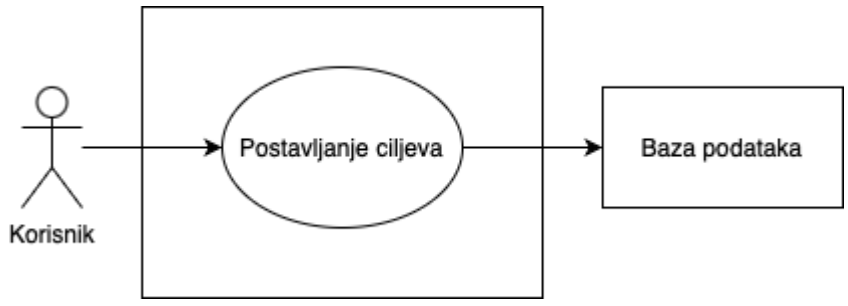
- Osobe zadužene za upravljanje aplikacijom i moderaciju sadržaja, uključujući upravljanje vježbama i predefiniranim planovima treninga te osiguravanje kvalitete sadržaja.
- Relevantni zahtjevi: FA-001 do FA-003.

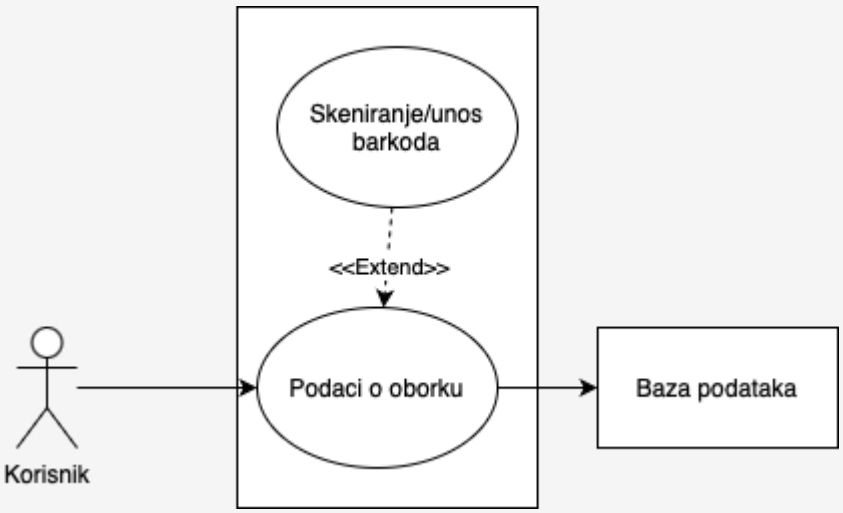
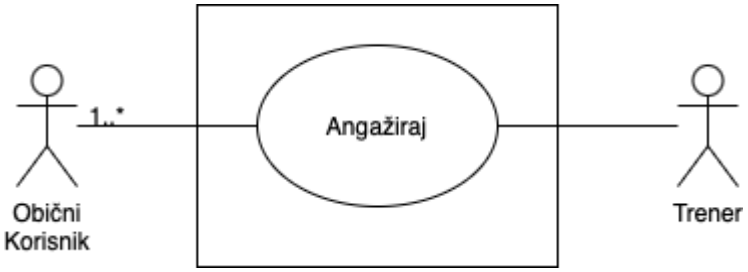
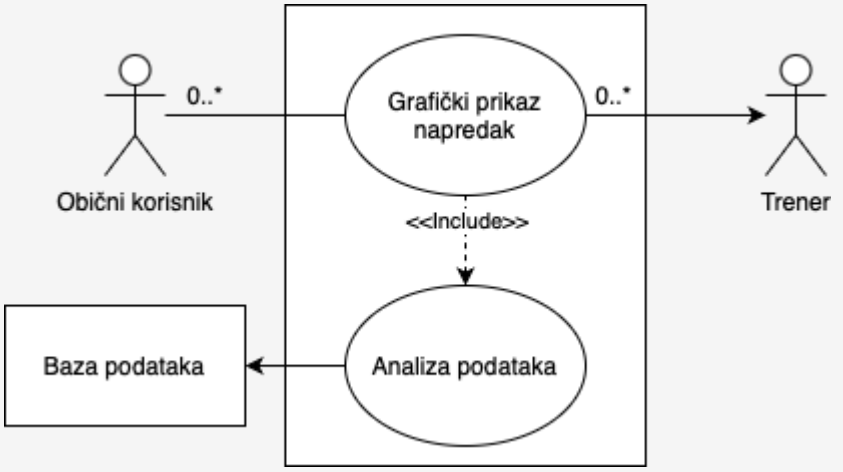
Razvojni tim

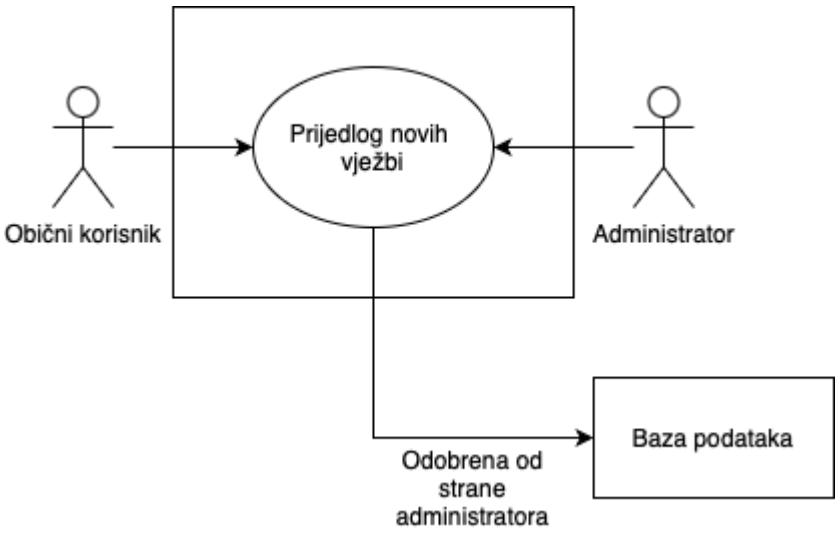
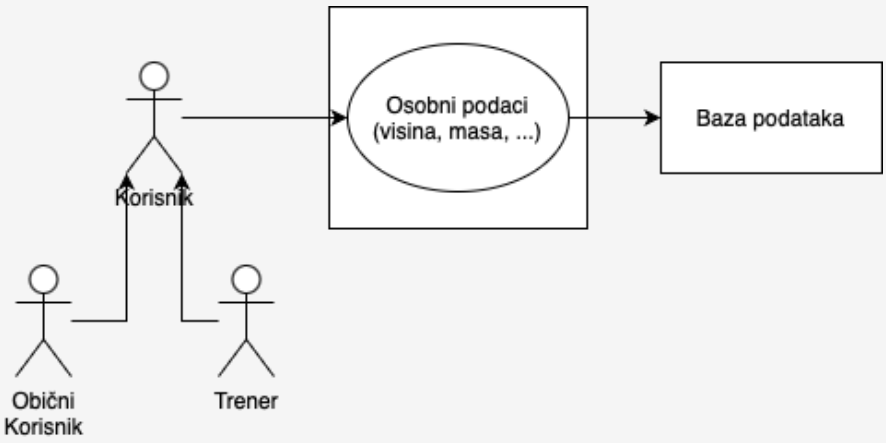
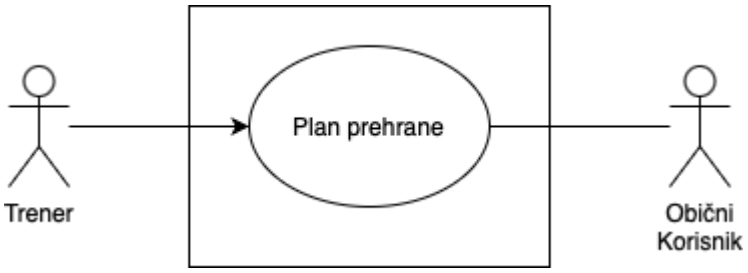
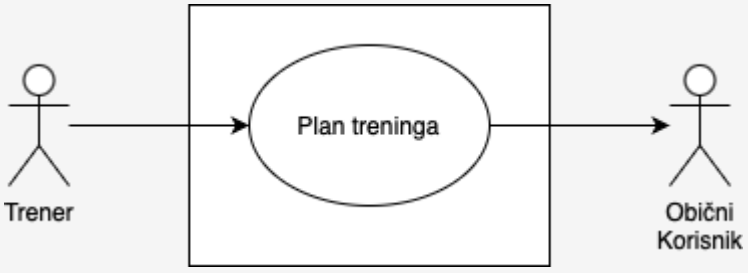
- Izravno uključeni u definiranje i održavanje dokumentacije zahtjeva te praćenje povratnih informacija korisnika.

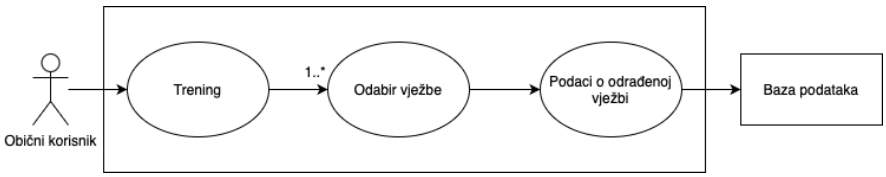
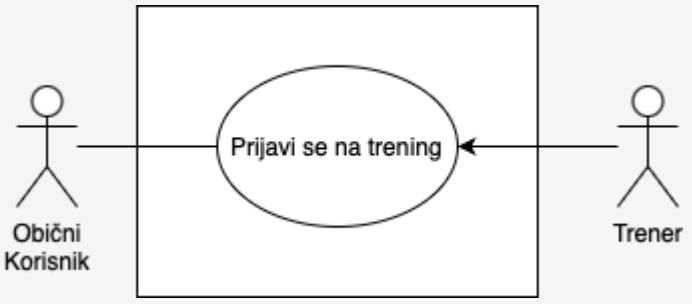
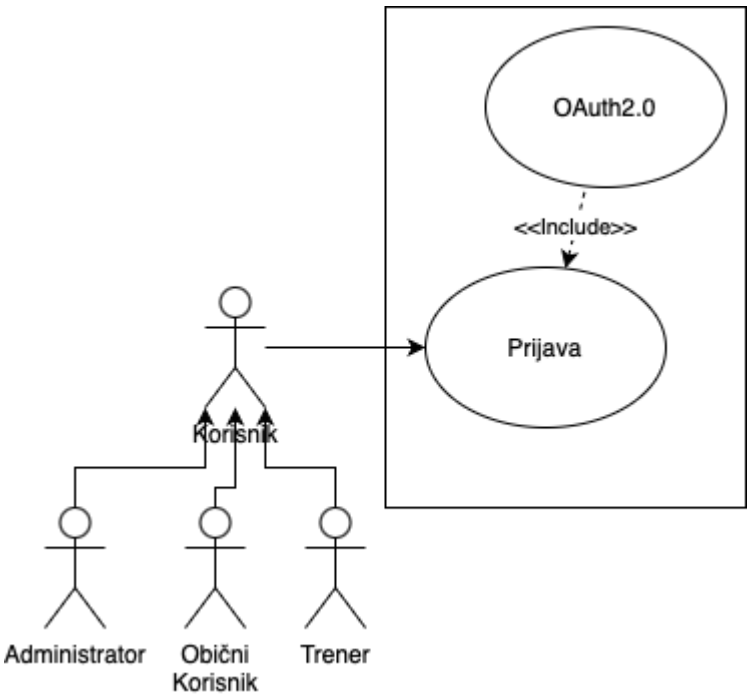
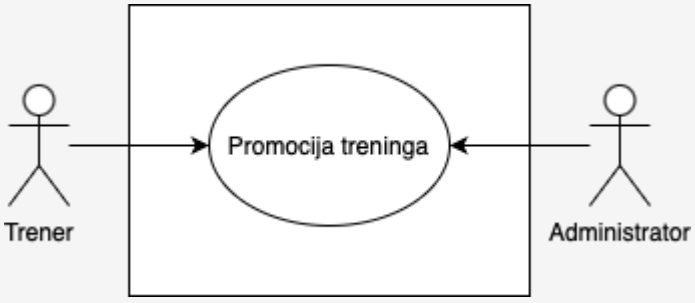
3. Specifikacija zahtjeva sustava

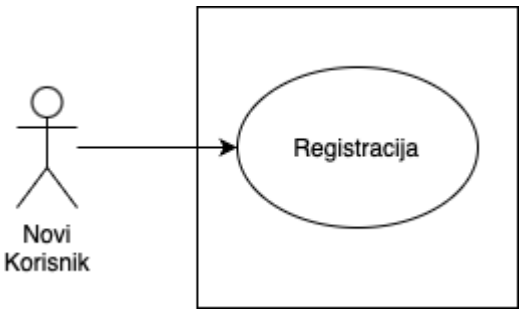
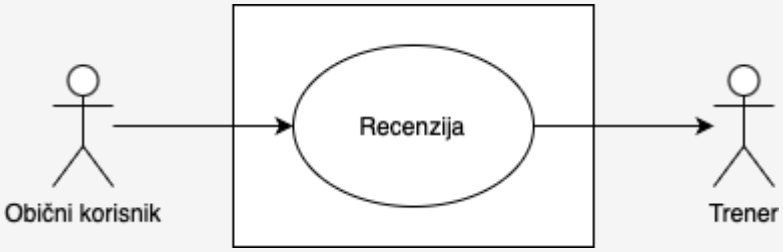
Dijagrami obrazaca uporabe

Opis	Dijagram
Ciljevi	
Hrana	

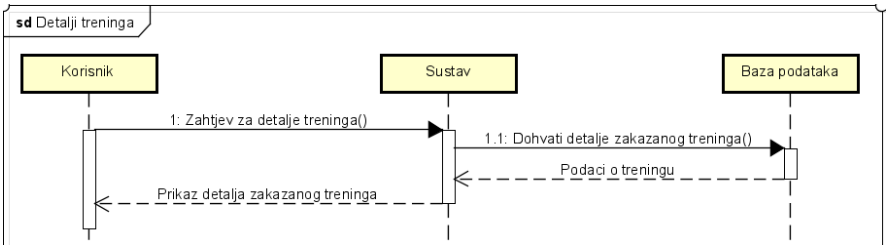
	
Korisnik-Trener prijava	
Napredak	
Nove vježbe	

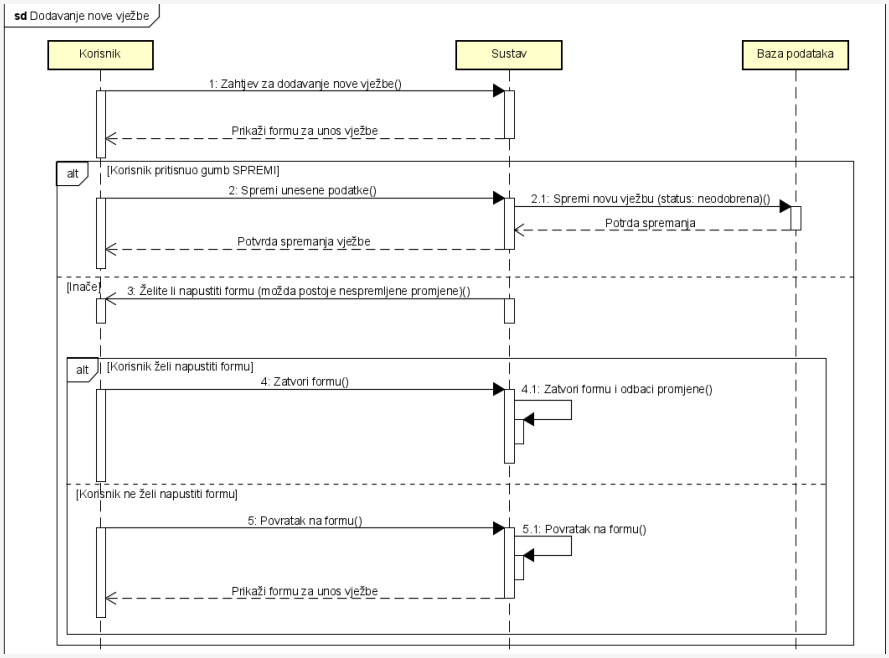
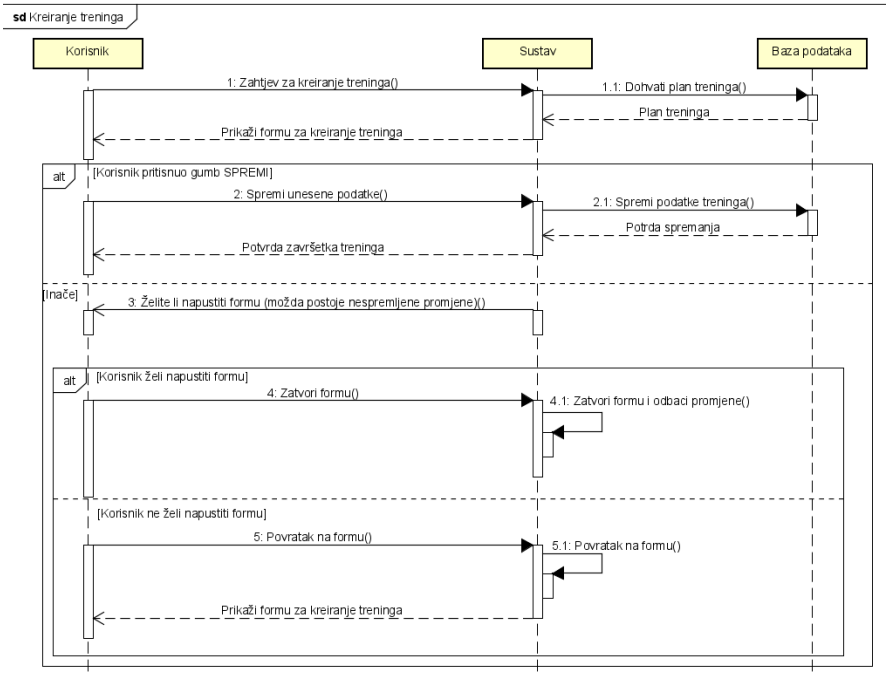
	
Osobni podaci	
Plan prehrane	
Plan treninga	

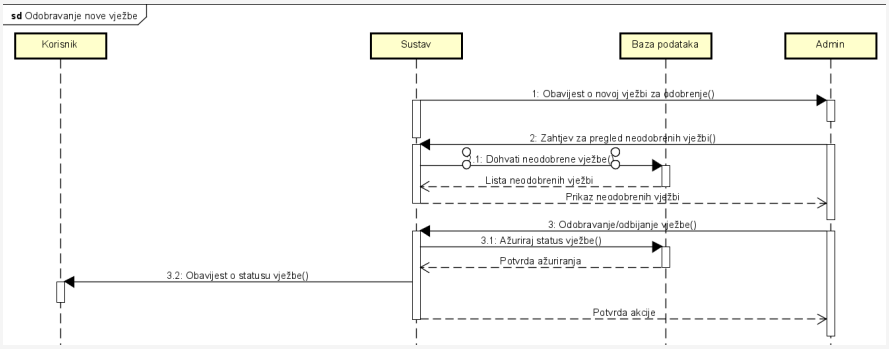
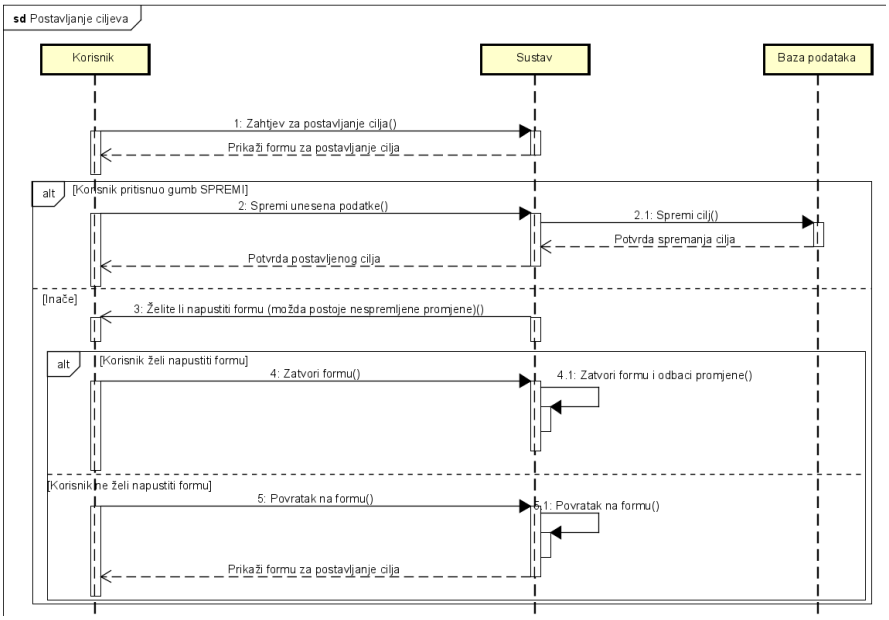
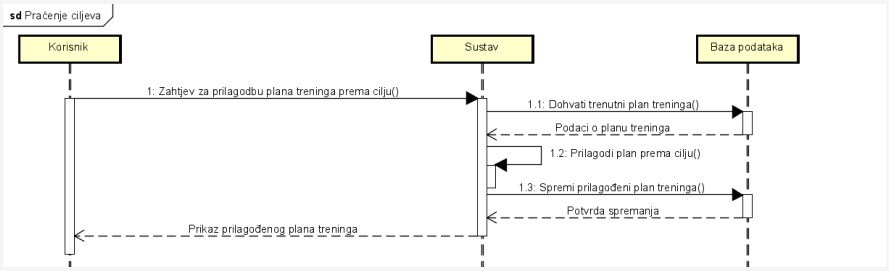
Praćenje treninga	
Prijava na trening	
Prijava	
Promocija	
Registracija	

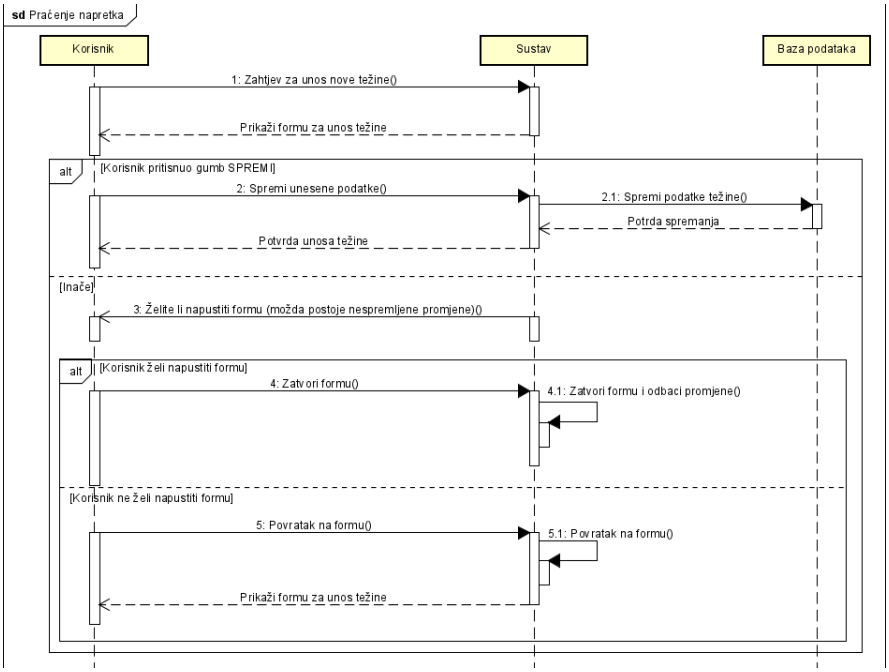
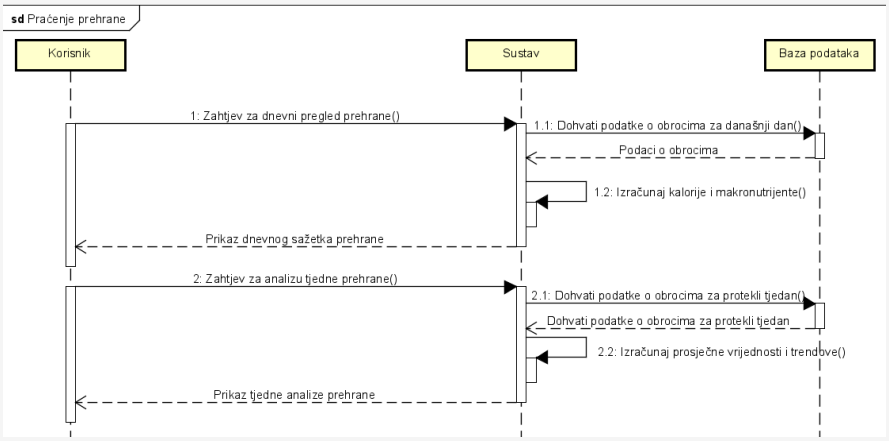
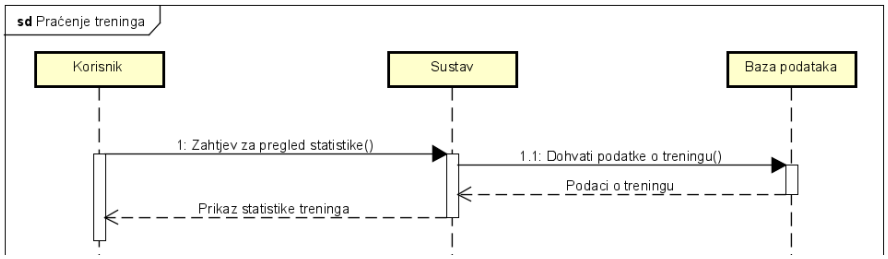
	
[Trener recenzija]	

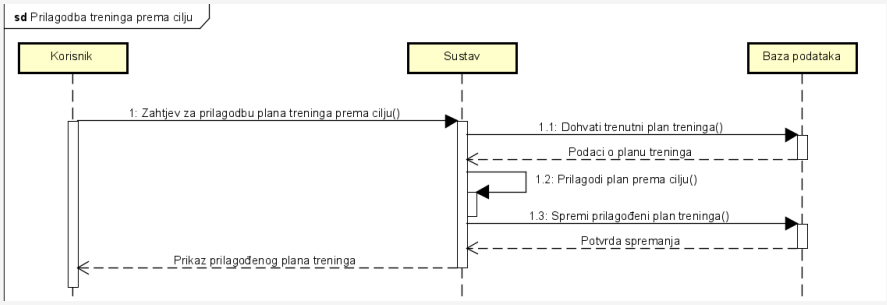
Sekvencijski dijagrami

Diagram	Link
Detalji treninga	
Dodavanje nove vježbe	

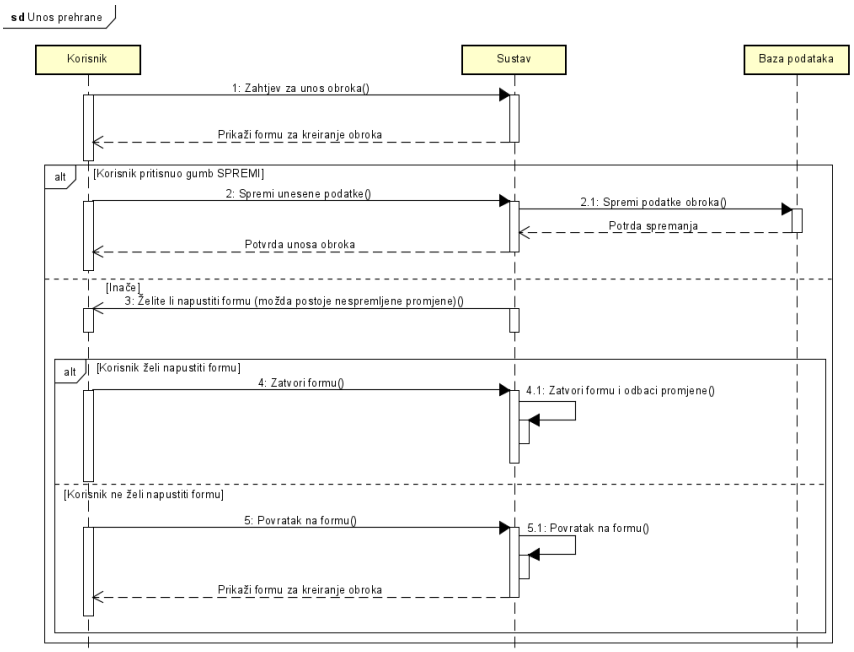
	
Kreiranje treninga	
Odobravanje nove vježbe	

	
Postavljanje ciljeva	
Praćenje ciljeva	
Praćenje napretka	

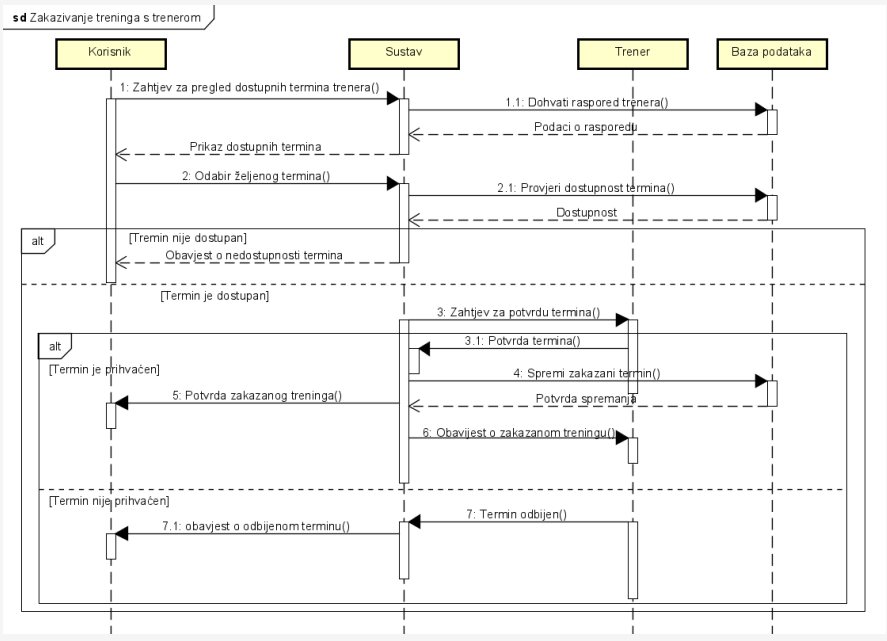
	
Praćenje prehrane	
Praćenje treninga	
Prilagodba treninga prema cilju	



Unos prehrane



Zakazivanje treninga s trenerom



Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

---

Postavljanje ciljeva

Element	Opis
Naziv	Postavljanje ciljeva
Redni broj	uc1
Opis	Korisnik postavlja ciljeve koje će pratiti putem aplikacije
Glavni aktor	Registrirani korisnik
Preduvjeti	N/A
Osnovni tijek	1. Korisnik unosi ciljeve putem forme 2. Aplikacija šalje podatke u bazu podataka 3. Baza pohranjuje ciljeve u tablicu
Moguća odstupanja	2.a) Ako korisnik ne upiše ispravne vrijednosti, aplikacija traži ispravak

---

Unos obroka

Element	Opis
Naziv	Unos obroka
Redni broj	uc2
Opis	Način na koji korisnik unosi podatke o obroku za praćenje makronutrijenata
Glavni aktor	Registrirani korisnik
Preduvjeti	N/A
Osnovni tijek	1. Korisnik otvara formu za unos podataka o svom obroku 2. Korisnik unosi dio po dio svog obroka 3. Korisnik mijenja količinu unesenog svakog artikla u obroku 4. Korisnik šalje formu prema aplikaciji 5. Aplikacija prosljeđuje formu prema bazi podataka 6. Baza podataka zapisuje podatke u tablicu za daljnju obradu
Moguća odstupanja	2.a) Korisnik može skenirati barkod sa proizvoda 4.a) Ako je forma prazna, aplikacija traži potvrdu korisnika za unos; ako korisnik potvrdi prazan unos, aplikacija odbacuje formu

---

Interakcija korisnika i trenera

Element	Opis
Naziv	Interakcija korisnik - trener
Redni broj	uc3
Opis	Vizualni prikaz interakcije korisnika s trenerom za angažman u vezi treninga ili prehrane
Glavni aktor	Registrirani korisnik / korisnici
Preduvjeti	Korisnik ima osobnog trenera unutar aplikacije
Osnovni tijek	1. Korisnik šalje zahtjev za angažman trenera 2. Korisnik bira opciju angažmana (plan treninga, plan prehrane ili oba) 3. Aplikacija obavještava trenera 4. Trener može zatražiti podatke o korisniku 5. Baza šalje podatke aplikaciji, koja ih prikazuje treneru 6. Trener može urediti podatke i poslati ih natrag u bazu 7. Aplikacija obavještava korisnika o ažuriranim podacima
Moguća odstupanja	N/A

Prikaz napretka

Element	Opis
Naziv	Prikaz napretka
Redni broj	uc4
Opis	Prikaz napretka korisnika ili grupe korisnika za trenera
Glavni aktor	Registrirani korisnik, Trener
Preduvjeti	Trener ima korisnike koje trenira
Osnovni tijek	1. Korisnik zatraži prikaz napretka 2. Aplikacija zatraži podatke iz baze 3. Aplikacija analizira podatke 4. Podaci se prikazuju korisniku u grafičkom obliku
Moguća odstupanja	Trener može vidjeti napredak za svakog korisnika kojeg trenira

---

Prijedlog novih vježbi

Element	Opis
Naziv	Prijedlog novih vježbi
Redni broj	uc5
Opis	Korisnik predlaže nove vježbe koje bi se dodale u bazu
Glavni aktor	Registrirani korisnik, Admin

Preduvjeti	N/A
Osnovni tijek	1. Korisnik predlaže novu vježbu ispunjavajući formu 2. Aplikacija sprema prijedlog u privremenu tablicu 3. Admin vidi vizualni indikator novih prijedloga 4. Admin može pregledati predložene vježbe 5. Admin potvrđuje novu vježbu 6. Aplikacija dodaje vježbu u bazu podataka, a privremeni zapis se briše
Moguća odstupanja	1.a) Ako je admin, ide izravno na korak 6 5.a) Ako admin odbije vježbu, ona se ne dodaje u bazu

---

Unos osobnih podataka

Element	Opis
Naziv	Unos osobnih podataka
Redni broj	uc6
Opis	Način unosa osobnih podataka korisnika
Glavni aktor	Registrirani korisnik
Preduvjeti	N/A
Osnovni tijek	1. Korisnik unosi podatke putem forme 2. Aplikacija šalje podatke u bazu
Moguća odstupanja	Aplikacija provjerava unos i traži ispravak ako je format pogrešan

---

Unos plana prehrane

Element	Opis
Naziv	Unos plana prehrane
Redni broj	uc7
Opis	Način interakcije korisnika ili trenera putem plana prehrane
Glavni aktor	Registrirani korisnik
Preduvjeti	N/A
Osnovni tijek	1. Korisnik kreira i unosi plan prehrane 2. Plan se sprema u bazu
Moguća odstupanja	Trener može kreirati plan prehrane za svoje korisnike koji ga također mogu uređivati

---

Unos plana treninga

Element	Opis

Naziv	Unos plana treninga
Redni broj	uc8
Opis	Način interakcije korisnika ili trenera putem plana treninga
Glavni aktor	Registrirani korisnik
Preduvjeti	N/A
Osnovni tijek	1. Korisnik kreira i unosi plan treninga 2. Plan se sprema u bazu
Moguća odstupanja	Trener može kreirati plan za svoje korisnike koji ga također mogu uređivati

---

Odrađivanje treninga

Element	Opis
Naziv	Odrađivanje treninga
Redni broj	uc9
Opis	Način pokretanja i bilježenja vježbi tijekom treninga
Glavni aktor	Registrirani korisnik
Preduvjeti	N/A
Osnovni tijek	1. Korisnik pokreće trening 2. Otvara se forma za unos vježbi 3. Korisnik bilježi podatke o vježbi 4. Nakon završetka, podaci se šalju u bazu 5. Baza pohranjuje podatke o treningu
Moguća odstupanja	N/A

Prijava korisnika na grupni trening

Element	Opis
Naziv	Prijava korisnika na grupni trening
Redni broj	uc10
Opis	Proces prijave korisnika na grupni trening
Glavni aktor	Registrirani korisnik, Trener
Preduvjeti	Trener vodi grupu za trening
Osnovni tijek	1. Korisnik se prijavljuje na grupni termin
Moguća odstupanja	Trener može prijaviti korisnika na termin po dogovoru

---

Prijava korisnika

Element	Opis
Naziv	Prijava korisnika
Redni broj	uc11
Opis	Proces prijave korisnika
Glavni aktor	Registrirani korisnik
Preduvjeti	Korisnik je registriran
Osnovni tijek	1. Korisnik započinje prijavu 2. Korisnik se prijavljuje putem autentifikacije (OAuth)
Moguća odstupanja	Ako unos nije ispravan, aplikacija obavještava korisnika

---

Promocija vježbi

Element	Opis
Naziv	Promocija vježbi
Redni broj	uc12
Opis	Način promoviranja vježbi putem platforme
Glavni aktor	Trener / Admin
Preduvjeti	Baza vježbi nije prazna
Osnovni tijek	1. Trener ili admin bira vježbu za promociju 2. Aplikacija obavještava korisnike ili prikazuje indikator nove vježbe
Moguća odstupanja	N/A

Registracija korisnika

Element	Opis
Naziv	Registracija korisnika
Redni broj	uc13
Opis	Proces registracije korisnika
Glavni aktor	Novi korisnik
Preduvjeti	N/A
Osnovni tijek	1. Korisnik otvara formu za registraciju 2. Korisnik ispunjava formu 3. Nakon ispunjavanja, korisnik postaje registrirani korisnik 4. Aplikacija šalje podatke u bazu
Moguća	2.a) Ako korisnik ne ispunjava ispravno formu, aplikacija traži

odstupanja	ispravak

Recenzija korisnika

Element	Opis
Naziv	Recenzija korisnika
Redni broj	uc14
Opis	Način davanja recenzije treneru
Glavni aktor	Registrirani korisnik
Preduvjeti	Trener ima korisnike koje trenira
Osnovni tijek	1. Korisnik dolazi na stranicu trenera 2. Otvara se forma za ocjenu i komentar 3. Recenzija se šalje i pohranjuje u bazu 4. Prilikom otvaranja profila, aplikacija dohvaća recenzije 5. Aplikacija prikazuje recenzije i izračunava prosječnu ocjenu
Moguća odstupanja	N/A

4. Arhitektura i dizajn sustava

Arhitektura sustava

Opis arhitekture

Stil arhitekture:

Klijent-poslužitelj

Podsustavi:

- **Frontend**
  - Next.js i TailwindCSS za responzivni i prilagodljivi dizajn
  - Integracija s backendom ostvaruje se putem REST API-ja
- **Backend**
  - Implementiran u Java Spring Frameworku koristeći Spring Boot za brzo kreiranje REST API-ja.
  - Spring Security implementira autentifikaciju i autorizaciju (OAuth 2.0).
- **Pohrana podataka**
  - PostgreSQL relacijska baza podataka za pohranu korisničkih podataka.
  - Amazon S3 koristi se za spremanje datoteka
- **Integracija s vanjskim sustavima**

    - Vanjski API za dohvat makronutrijenata i kalorija namirnica preko barkoda

- **Administracija**
    - Sučelje za moderaciju sadržaja, dodavanje vježbi i predefiniranih planova te pregled korisničkih aktivnosti.

Preslikavanje na radnu platformu:

- **Frontend**
    - Hosting putem Vercel za optimalne performanse i brzu isporuku sadržaja
    - Responzivnost osigurana za rad na mobilnim i desktop uređajima.
- **Backend**
    - Deployment na AWS EC2 instancu s CI/CD pipelineom kroz GitHub Actions
    - Backend je osiguran HTTPS protokolima za siguran prijenos podataka
- **Pohrana podataka**
    - Korisnički podaci pohranjenu su u PostgreSQL bazi, dok slike idu na Amazon S3.

Spremišta podataka:

- **Relacijska baza podataka**
    - PostgreSQL: Koristi se za pohranu korisničkih podataka, trening plana, napretka i unosa kalorija.
    - Ebean ORM omogućava jednostavno mapiranje između relacijske baze i Java objekata
- **Datotečni sustavi**
    - Amazon S3 za pohranu slika

Mrežni protokoli:

- **HTTPS**
    - Svi zahtjevi između klijenta i servera osigurani su putem HTTPS protokola.
    - Podaci se enkriptiraju koristeći SSL/TLS certifikate.
- **REST API**
    - Koristi se za komunikaciju između frontenda i backenda.
- **OAuth 2.0**
    - Standard za autentifikaciju i autorizaciju korisnika.

Globalni upravljački tok:

- **Registracija i prijava**

    - Korisnici prolaze OAuth 2.0 autentifikaciju za pristup funkcionalnostima aplikacije.
    - Nakon prijave, korisnik može unijeti svoje ciljeve i trenutačne osobne podatke.
- **Praćenje i ažuriranje podataka**
    - Korisnici unose podatke o treningu, prehrani i napretku u stvarnom vremenu.
    - Backend obrađuje i sprema podatke u PostgreSQL bazu podataka.
- **Personalizirane funkcionalnosti**
    - Backend procesira korisničke podatke i pruža prilagođene preporuke za trening i prehranu.
- **Moderacija sadržaja**
    - Administrator pregledava prijedloge novih vježbi i planova, te odobrava ili odbija sadržaj.

Sklopovskoprogramski zahtjevi:

- **Serveri**
    - Backend server: 2 vCPU, 1 GB RAM (za optimalan rad sa Spring Boot i PostgreSQL).
    - Frontend server: 4 vCPU, 8 GB RAM
    - CDN hosting za frontend (Vercel).
- **Sigurnost**
    - SSL certifikati za enkripciju podataka.
    - Implementacija stroge kontrole pristupa kroz autorizaciju korisnika.
- **Podrška za mobilne uređaje**
    - Optimizacija za rad na uređajima različitih rezolucija i brzina interneta (3G i novije).
- **Skalabilnost**
    - Mikroservisna arhitektura omogućava horizontalno skaliranje komponenti prema potrebi.
    - Automatsko skaliranje kroz VPS infrastrukturu i cloud resurse (Amazon S3).

Obrazloženje odabira arhitekture

Odabrana arhitektura sustava koristi model klijent-poslužitelj s podrškom za integraciju različitih podsustava. Ovo rješenje osigurava jednostavnu razmjenu podataka između komponenti i omogućava modularnost, skalabilnost i održivost. Korištenjem klijent-poslužitelj arhitekture na početku razvoja projekta osigurava manju složenost, te ako se kasnije odlučimo (ili aplikacija postane previše kompleksna) olakšava prijelaz na mikroservisnu arhitekturu

Ključni faktori pri odabiru:

- Skalabilnost: Sustav je osmišljen s podrškom za horizontalno skaliranje kroz optimizaciju frontend i backend slojeva te korištenje VPS resursa.
- Sigurnost: Uključeni su OAuth 2.0 standard za autentifikaciju i HTTPS za enkripciju komunikacije.
- Jednostavnost implementacije: Kombinacija provjerenih tehnologija (Spring Boot, NextJS) omogućava brzo postavljanje osnovne funkcionalnosti.
- Modularnost: Jasno definirani slojevi frontend i backend aplikacije olakšavaju održavanje i dodavanje novih funkcionalnosti.

Principi oblikovanja arhitekture:

- Visoka kohezija: Svaka komponenta aplikacije fokusirana je na određenu funkcionalnost (npr. upravljanje korisnicima, trening planovima ili unosom podataka).
- Niska povezanost: Komponente komuniciraju preko jasno definiranih API sučelja, što omogućava jednostavno testiranje i zamjenu pojedinih modula.
- Fleksibilnost: Koriste se industrijski standardni protokoli i tehnologije, što omogućava laku integraciju s vanjskim servisima i sustavima.
- Sigurnost: Podaci su zaštićeni višeslojnim pristupom, uključujući SSL/TLS enkripciju i sigurno pohranjivanje korisničkih podataka.

Razmatrane alternative:

- Mikrouslužna arhitektura: Razmatrali smo korištenje mikrousluga za organizaciju sustava, ali smo je odbacili zbog početne složenosti postavljanja, većih troškova održavanja i nedostatka potrebe za takvom razinom skalabilnosti u početnim fazama projekta.
- Monolitna arhitektura: Iako jednostavnija za implementaciju, monolitna arhitektura može otežati skaliranje i budući razvoj, pa smo se odlučili za modularnu monolitnu arhitekturu.

Organizacija sustava na visokoj razini

Klijent-poslužitelj:

- Sustav koristi klijent-poslužitelj arhitekturu. Frontend (NextJS) šalje zahtjeve backendu (Spring Boot), koji obrađuje poslovnu logiku i dohvaća podatke iz PostgreSQL baze ili Amazon S3.

Baza podataka:

- PostgreSQL: Relacijska baza podataka koristi se za pohranu strukturiranih podataka, uključujući korisničke profile, treninge, prehrambene planove i druge entitete.

Datotečni sustav:

- Amazon S3: Koristi se za pohranu slika i drugih datoteka (npr. barkodova), pritom osiguravajući visoku dostupnost i performanse

Grafičko sučelje:

- Frontend aplikacija izrađena u NextJS-u pruža responzivno i intuitivno korisničko iskustvo, povezano s backendom putem REST API-ja.

Organizacija aplikacije

Frontend i Backend slojevi:

- **Frontend** (NextJS): Fokusira se na prezentaciju podataka i korisničko iskustvo. NextJS pruža brzu izvedbu kroz server-side rendering (SSR) i optimizacije za različite uređaje.
- **Backend** (Spring Boot): Implementira poslovnu logiku, sigurnosne mehanizme i interakciju s bazom podataka. Backend koristi Ebean ORM za upravljanje podacima.

Baza podataka

Opis tablica

User

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator korisnika
name	String	Ime korisnika
email	String	Email korisnika
emailVerified	Boolean	Oznaka je li email korisnika verificiran
image	String	URL slike profila korisnika
role	Enum	Uloga korisnika (npr. `obični`, `trener`, `administrator`)
goal_weight	Number	Ciljana težina korisnika
password	String (opcionalno)	Enkriptirana lozinka korisnika
currentNutritionalPlan	UUID (FK)	Referenca na trenutačni plan prehrane
bodyMeasurement	UUID (FK)	Referenca na trenutačne tjelesne mjere korisnika
goalBodyMeasurement	UUID (FK)	Referenca na ciljne tjelesne mjere korisnika
trainerId	UUID (FK)	Referenca na osobnog trenera korisnika

Meal

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator obroka
isSuggestion	Boolean	Oznaka je li obrok prijedlog
userId	UUID (FK)	Referenca na korisnika koji je konzumirao obrok

createdById	UUID (FK)	Referenca na korisnika ili trenera koji je kreirao obrok
nutritionPlan	UUID (FK)	Referenca na povezani plan prehrane
time	Timestamp	Vrijeme kada je obrok konzumiran
suggestedId	UUID (FK)	Referenca na prijedlog obroka

Food

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator hrane
calories	Number	Broj kalorija po porciji
unit	Enum	Jedinica mjere (npr. grami, komadi)
defaultNumber	Number	Zadana veličina porcije
fats	Number	Količina masti po porciji
carbs	Number	Količina ugljikohidrata po porciji
protein	Number	Količina proteina po porciji

---

foodMeal

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator zapisa
quantity	Number	Količina hrane u obroku
foodId	UUID (FK)	Referenca na hranu
mealId	UUID (FK)	Referenca na obrok

---

nutritionPlan

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator plana prehrane
userId	UUID (FK)	Referenca na korisnika
name	String	Naziv plana prehrane
calories	Number	Ciljana dnevna kalorijska vrijednost
protein	Number	Ciljani dnevni unos proteina
carbs	Number	Ciljani dnevni unos ugljikohidrata
fat	Number	Ciljani dnevni unos masti
startDate	DateTime	Datum početka plana

endDate	DateTime	Datum završetka plana

---

performedSet

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator izvedene serije
performedExercises	UUID (FK)	Referenca na izvedenu vježbu
reps	Number	Broj ponavljanja u seriji
weight	Number	Korištena težina u seriji
rpe	Number	Skala percepcije napora (RPE) za seriju

---

performedExercises

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator izvedene vježbe
workoutSession	UUID (FK)	Referenca na trening sesiju
plannedExercise	UUID (FK)	Referenca na planiranu vježbu

---

workoutSession

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator sesije treninga
workoutId	UUID (FK)	Referenca na planirani trening
userId	UUID (FK)	Referenca na korisnika koji je sudjelovao u sesiji
date	DateTime	Datum trening sesije
userReviewId	UUID (FK)	Referenca na korisnički pregled sesije
trainerReviewId	UUID (FK)	Referenca na trenerski pregled sesije

---

plannedExercise

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator planirane vježbe
workoutId	UUID (FK)	Referenca na planirani trening
exerciseId	UUID (FK)	Referenca na vježbu
sets	Number	Broj serija planiranih za vježbu
reps	Number	Broj ponavljanja planiranih za vježbu

rpe	Number	Ciljani RPE za vježbu
order	Number	Redoslijed vježbe u treningu

---

workoutPlan

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator plana treninga
name	String	Naziv plana treninga
description	String	Opis plana treninga
image	String	Slika koja predstavlja plan treninga
createdById	UUID (FK)	Referenca na korisnika koji je kreirao plan
userId	UUID (FK, opcionalno)	Referenca na korisnika koji slijedi plan
originalWorkoutPlanId	UUID (FK)	Referenca na originalni plan treninga

---

workout

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator treninga
name	String	Naziv treninga
description	String	Opis treninga
workoutPlanId	UUID (FK)	Referenca na plan treninga
order	Number	Redoslijed treninga u planu

---

exercise

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator vježbe
name	String	Naziv vježbe
description	String	Opis vježbe
gif	String	Animacija ili video koji prikazuje vježbu
createdById	UUID (FK)	Referenca na korisnika koji je kreirao vježbu
isApproved	Boolean	Oznaka je li vježba odobrena

primaryMuscleGroup	UUID (FK)	Referenca na primarnu mišićnu grupu
secondaryMuscleGroup	UUID (FK)	Referenca na sekundarnu mišićnu grupu

---

review

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator recenzije
rating	Number	Ocjena recenzije
comment	String	Komentar recenzije

---

goals

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator cilja
userId	UUID (FK)	Referenca na korisnika koji ima cilj
gender	Enum	Spol korisnika
protein	Number	Ciljani unos proteina (g)
height	Number	Visina korisnika (cm)
timeline_weeks	Number	Trajanje cilja u tjednima
createdAt	DateTime	Datum kada je cilj kreiran
updatedAt	DateTime	Datum kada je cilj posljednji puta ažuriran

---

verificationToken

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator tokena
token	UUID	Jedinstveni token
userId	UUID (FK)	Referenca na korisnika koji koristi token
expires	DateTime	Datum kada token ističe

---

sleepLog

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator loga spavanja
userId	UUID (FK)	Referenca na korisnika
date	DateTime	Datum loga spavanja
duration	Number	Trajanje spavanja u satima

quality	Number	Kvaliteta spavanja (ocjena 1-10)
notes	String	Dodatne napomene vezane uz spavanje
createdAt	DateTime	Datum kada je log spavanja kreiran
updatedAt	DateTime	Datum kada je log spavanja posljednji puta ažuriran

---

muscleGroup

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator mišićne grupe
name	String	Naziv mišićne grupe
description	String	Opis mišićne grupe
image	String	Slika koja predstavlja mišićnu grupu

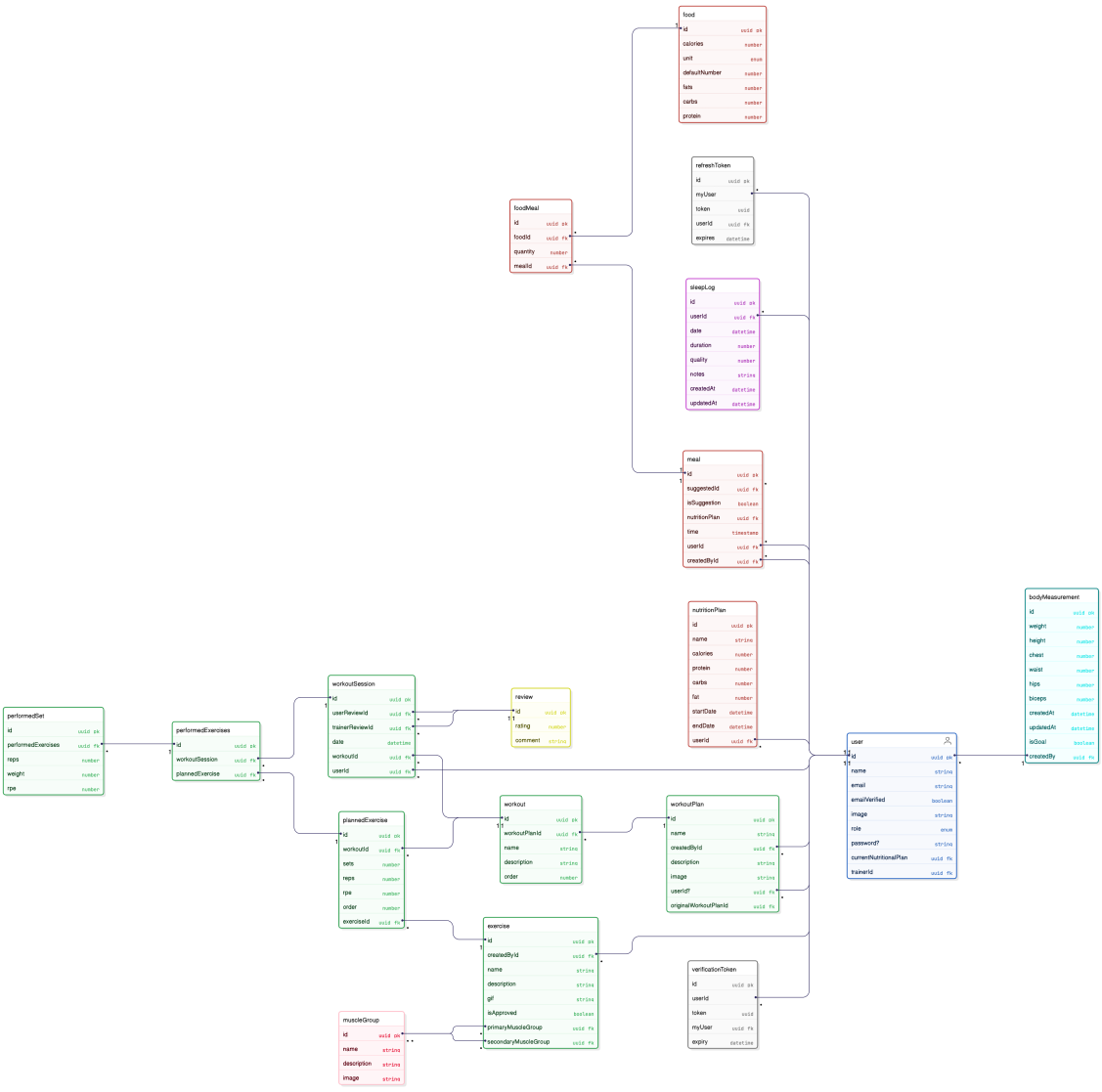
---

bodyMeasurement

Atribut	Tip podatka	Opis varijable
id	UUID	Jedinstveni identifikator tjelesnih mjera
weight	Number	Težina korisnika (kg)
height	Number	Visina korisnika (cm)
chest	Number	Opseg grudi korisnika (cm)
waist	Number	Opseg struka korisnika (cm)
hips	Number	Opseg bokova korisnika (cm)
biceps	Number	Opseg bicepsa korisnika (cm)
createdAt	DateTime	Datum kada su mjere kreirane
updatedAt	DateTime	Datum kada su mjere posljednji puta ažurirane

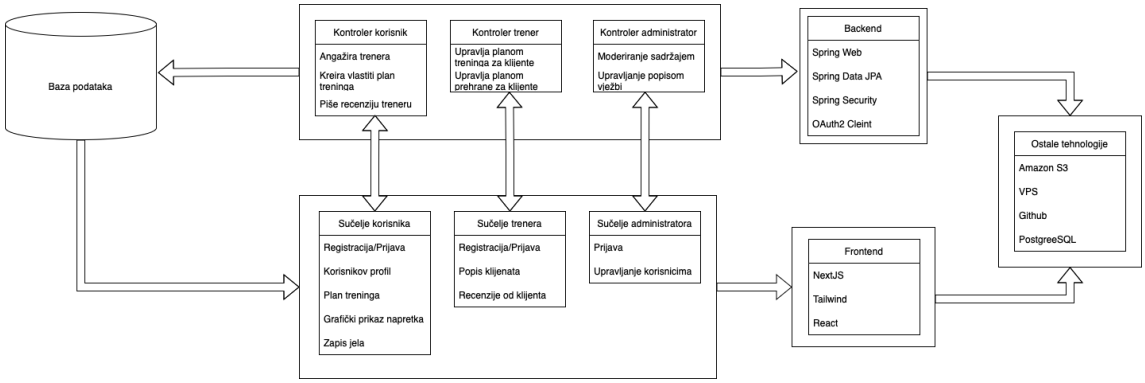
---

Dijagram baze podataka

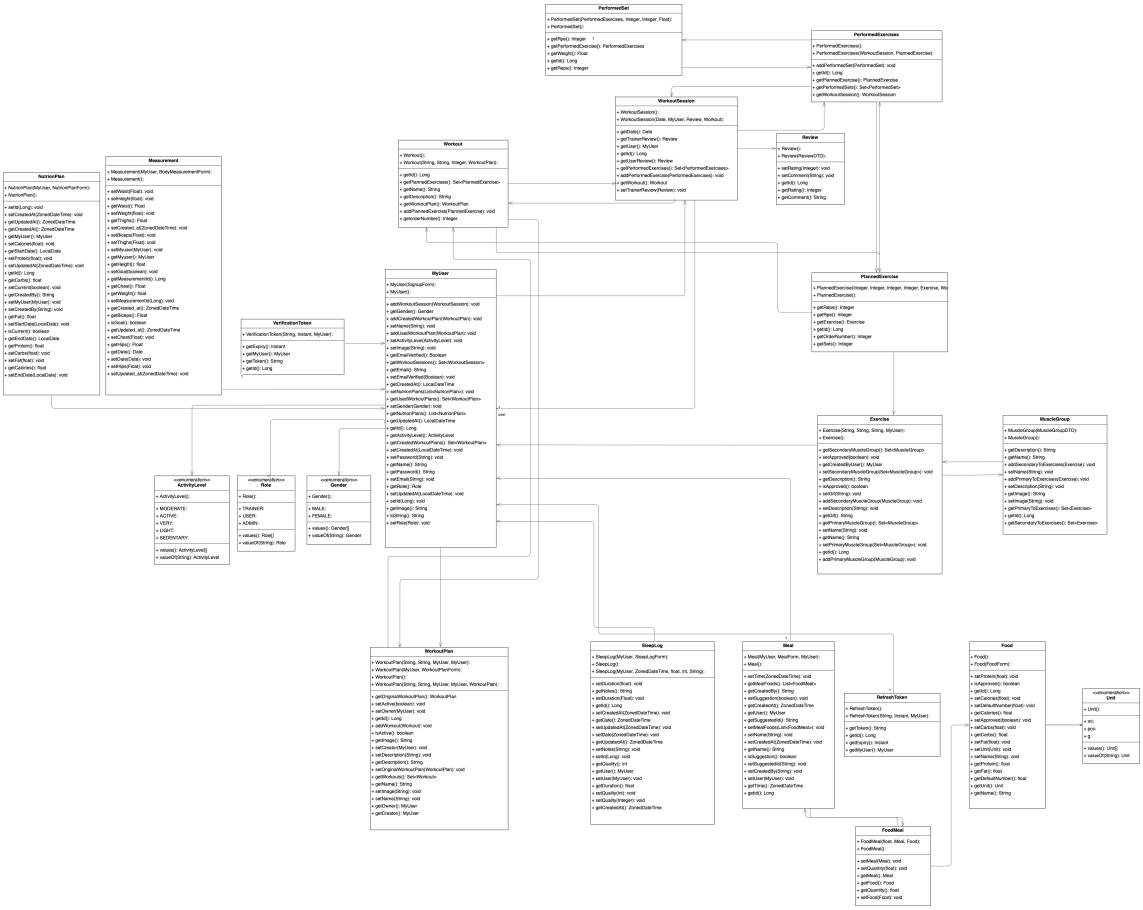


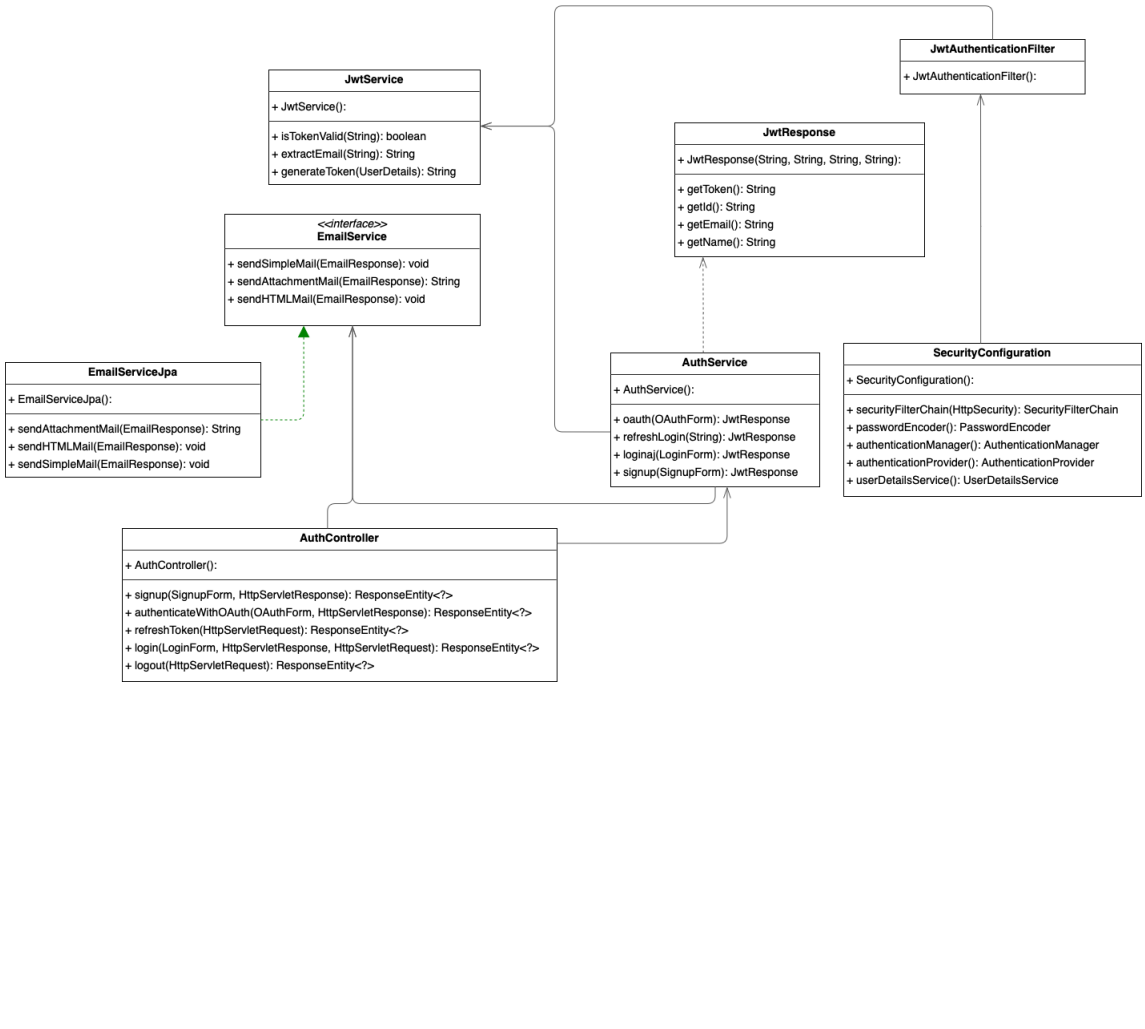
Opći pregled arhitekture sustava

MVC dijagram



Dijagram razreda


PerformedSet
+ PerformedSet(PerformedExercises, Integer, Integer, Float):
+ PerformedSet():
+ getReps(): Integer
+ getPerformedExercise(): PerformedExercises
+ getWeight(): Float
+ getId(): Long
+ getReps(): Integer
PerformedExercises
+ PerformedExercises():
+ PerformedExercises(WorkoutSession, PlannedExercise):
+ addPerformedSet(PerformedSet): void
+ getId(): Long
+ getPlannedExercise(): PlannedExercise
+ getPerformedSets(): Set<PerformedSet>
+ getWorkoutSession(): WorkoutSession
WorkoutSession
+ WorkoutSession():
+ WorkoutSession(Date, MyUser, Review, Workout):
+ getDate(): Date
+ getTrainerReview(): Review
+ getUser(): MyUser
+ getId(): Long
+ getUserReview(): Review
+ getPerformedExercises(): Set<PerformedExercises>
+ addPerformedExercise(PerformedExercises): void
+ getWorkout(): Workout
+ setTrainerReview(Review): void
Review
+ Review():
+ Review(ReviewDTO):
+ setRating(Integer): void
+ setComment(String): void
+ getId(): Long
+ getRating(): Integer
+ getComment(): String
Workout
+ Workout():
+ Workout(String, String, Integer, WorkoutPlan):
+ getId(): Long
+ getPlannedExercises(): Set<PlannedExercise>
+ getName(): String
+ getDescription(): String
+ getWorkoutPlan(): WorkoutPlan
+ addPlannedExercise(PlannedExercise): void
+ getorderNumber(): Integer
Measurement
+ Measurement(MyUser, BodyMeasurementForm):
+ Measurement():
+ setWaist(Float): void
+ setHeight(Float): void
+ getWaist(): Float
+ setWeight(Float): void
+ getThighs(): Float
+ setCreated_at(ZonedDateTime): void
+ setBiceps(Float): void
+ setThighs(Float): void
+ setMyUser(MyUser): void
+ getMyUser(): MyUser
+ getHeight(): float
+ setIsabl(boolean): void
+ getMeasurementId(): Long
+ getChest(): Float
+ setWeight(): float
+ setMeasurement(id):Long): void
+ getCreated_at(): ZonedDateTime
+ getBiceps(): Float
+ isVoid(): boolean
+ getUpdated_at(): ZonedDateTime
+ setChest(Float): void
+ getHips(): Float
+ getDate(): Date
+ setDateDate(): void
+ setHips(Float): void
+ setUpdated_at(ZonedDateTime): void
NutricePlan
+ NutricePlan(MyUser, NutricePlanForm):
+ NutricePlan():
+ setId(Long): void
+ setCreatedAt(ZonedDateTime): void
+ getUpdatedAt(): ZonedDateTime
+ getCreatedAt(): ZonedDateTime
+ getMyUser(): MyUser
+ setCalories(float): void
+ getStartDate(): LocalDate
+ setProtein(float): void
+ setUpdatedAt(ZonedDateTime): void
+ getId(): Long
+ getCarbs(): float
+ setCurrent(boolean): void
+ getCreatedBy(): String
+ setMyUser(MyUser): void
+ setCreatedBy(String): void
+ getFat(): float
+ setStartDate(LocalDate): void
+ isCurrent(): boolean
+ getEndDate(): LocalDate
+ getProtein(): float
+ setCarbs(float): void
+ setFat(float): void
+ getCalories(): float
+ setEndDate(LocalDate): void
VerificationToken
+ VerificationToken(String, Instant, MyUser):
+ getExpiry(): Instant
+ getMyUser(): MyUser
+ getToken(): String
+ getId(): Long
MyUser
+ MyUser(SignupForm):
+ MyUser():
+ addWorkoutSession(WorkoutSession): void
+ getGender(): Gender
+ addCreatedWorkoutPlan(WorkoutPlan): void
+ setName(String): void
+ addJoadWorkoutRun(WorkoutRun): void
+ setActivityLevel(ActivityLevel): void
+ setImage(String): void
+ getEmailVerified(): Boolean
+ getWorkoutSessions(): Set<WorkoutSession>
+ getEmail(): String
+ setEmailVerified(Boolean): void
+ getCreatedAt(): LocalDateTime
+ setNutionPlans(List<NutlcePlan>): void
+ getUserWorkoutPlans(): Set<WorkoutPlan>
+ setGender(Gender): void
+ getNutionPlans(): List<NutlcePlan>
+ getUpdatedAt(): LocalDateTime
+ getId(): Long
+ getActivityLevel(): ActivityLevel
+ getCreatedWorkoutPlans(): Set<WorkoutPlan>
+ setCreatedAt(LocalDateTime): void
+ setPassword(String): void
+ getName(): String
+ getPassword(): String
+ setEmail(String): void
+ getRole(): Role
+ setUpdatedAt(LocalDateTime): void
+ setId(Long): void
+ getImage(): String
+ toString(): String
+ setHeal(Note): void
«enumeration»
ActivityLevel
+ ActivityLevel():
+ MODERATE:
+ ACTIVE:
+ VERY:
+ LIGHT:
+ SEDENTARY:
+ values(): ActivityLevel[]
+ valueOf(String): ActivityLevel
«enumeration»
Role
+ Role():
+ TRAINER:
+ USER:
+ ADMIN:
+ values(): Role[]
+ valueOf(String): Role
«enumeration»
Gender
+ Gender():
+ MALE:
+ FEMALE:
+ values(): Gender[]
+ valueOf(String): Gender
PlannedExercise
+ PlannedExercise(Integer, Integer, Integer, Integer, Exercise, Wo
+ PlannedExercise():
+ getReps(): Integer
+ getPlan(): Integer
+ getExercise(): Exercise
+ getId(): Long
+ getOrderNumber(): Integer
+ setSets(): Integer
Exercise
+ Exercise(String, String, String, MyUser):
+ Exercise():
+ getSecondaryMuscleGroup(): Set<MuscleGroup>
+ setApproved(boolean): void
+ getCreatedBy(User(): MyUser
+ setSecondaryMuscleGroup(Set<MuscleGroup>): void
+ getDescription(): String
+ isApproved(): boolean
+ setId(String): void
+ addSecondaryMuscleGroup(MuscleGroup): void
+ setDescription(String): void
+ getId(): Long
+ getPrimaryMuscleGroup(): Set<MuscleGroup>
+ setName(String): void
+ getName(): String
+ setPrimaryMuscleGroup(Set<MuscleGroup>): void
+ getId(): Long
+ addPrimaryMuscleGroup(MuscleGroup): void
MuscleGroup
+ MuscleGroup(MuscleGroupDTO):
+ MuscleGroup():
+ getDescription(): String
+ getName(): String
+ addSecondaryToExercises(Exercise): void
+ setName(String): void
+ addPrimaryToExercises(Exercise): void
+ setDescription(String): void
+ getImage(): String
+ setImage(String): void
+ getPrimaryToExercises(): Set<Exercise>
+ getId(): Long
+ getSecondaryToExercises(): Set<Exercise>
WorkoutPlan
+ WorkoutPlan(String, String, MyUser, MyUser):
+ WorkoutPlan(MyUser, WorkoutPlanForm):
+ WorkoutPlan():
+ WorkoutPlan(String, String, MyUser, MyUser, WorkoutPlan):
+ getOriginalWorkoutPlan(): WorkoutPlan
+ setActive(boolean): void
+ setOwner(MyUser): void
+ getId(): Long
+ addWorkout(Workout): void
+ isActive(): boolean
+ getImage(): String
+ setCreator(MyUser): void
+ setDescription(String): void
+ getDescription(): String
+ setOriginalWorkoutPlan(WorkoutPlan): void
+ getWorkouts(): Set<Workout>
+ getName(): String
+ setImage(String): void
+ setName(String): void
+ getOwner(): MyUser
+ getCreator(): MyUser
SleepLog
+ SleepLog(MyUser, SleepLogForm):
+ SleepLog():
+ SleepLog(MyUser, ZonedDateTime, float, int, String):
+ setDuration(float): void
+ getNotes(): String
+ setDuration(Float): void
+ getId(): Long
+ setCreatedAt(ZonedDateTime): void
+ getDate(): ZonedDateTime
+ setUpdatedAt(ZonedDateTime): void
+ setDate(ZonedDateTime): void
+ getUpdatedAt(): ZonedDateTime
+ setNotes(String): void
+ setId(Long): void
+ getCountry(): int
+ getUser(): MyUser
+ setUser(MyUser): void
+ getDuration(): float
+ setQuality(int): void
+ setQuality(Integer): void
+ getCreatedAt(): ZonedDateTime
Meal
+ Meal(MyUser, MealForm, MyUser):
+ Meal():
+ setTime(ZonedDateTime): void
+ getMealFoods(): List<FoodMeal>
+ getCreatedBy(): String
+ setSuggestion(boolean): void
+ getCreatedAt(): ZonedDateTime
+ getUser(): MyUser
+ getSuggested(): String
+ setMealFoods(List<FoodMeal>): void
+ setName(String): void
+ setCreatedAt(ZonedDateTime): void
+ getName(): String
+ isSuggestion(): boolean
+ setSuggested(String): void
+ setCreatedBy(String): void
+ setUser(MyUser): void
+ getTime(): ZonedDateTime
+ getId(): Long
RefreshToken
+ RefreshToken():
+ RefreshToken(String, Instant, MyUser):
+ getToken(): String
+ getId(): Long
+ getExpiry(): Instant
+ getMyUser(): MyUser
Food
+ Food():
+ Food(FoodForm):
+ setProtein(float): void
+ isApproved(): boolean
+ getId(): Long
+ setCalories(float): void
+ setDefaultNumber(float): void
+ getCalories(): float
+ setApproved(boolean): void
+ setCarbs(float): void
+ getCarbs(): float
+ setFat(float): void
+ setUnit(int): void
+ setName(String): void
+ getProtein(): float
+ getFat(): float
+ getDefaultNumber(): float
+ getUnit(): Unit
+ getName(): String
«enumeration»
Unit
+ Unit():
+ KG:
+ pcs:
+ g:
+ values(): Unit[]
+ valueOf(String): Unit
FoodMeal
+ FoodMeal(float, Meal, Food):
+ FoodMeal():
+ setMeal(Meal): void
+ setQuantity(float): void
+ getMeal(): Meal
+ getFood(): Food
+ getQuantity(): float
+ setFood(Food): void


JwtAuthenticationFilter
+ JwtAuthenticationFilter():
JwtService
+ JwtService():
+ isTokenValid(String): boolean
+ extractEmail(String): String
+ generateToken(UserDetails): String
JwtResponse
+ JwtResponse(String, String, String, String):
+ getToken(): String
+ getId(): String
+ getEmail(): String
+ getName(): String
<<interface>>
EmailService
+ sendSimpleMail(EmailResponse): void
+ sendAttachmentMail(EmailResponse): String
+ sendHTMLMail(EmailResponse): void
EmailServiceJpa
+ EmailServiceJpa():
+ sendAttachmentMail(EmailResponse): String
+ sendHTMLMail(EmailResponse): void
+ sendSimpleMail(EmailResponse): void
AuthService
+ AuthService():
+ oauth(OAuthForm): JwtResponse
+ refreshLogin(String): JwtResponse
+ loginaj(LoginForm): JwtResponse
+ signup(SignupForm): JwtResponse
SecurityConfiguration
+ SecurityConfiguration():
+ securityFilterChain(HttpSecurity): SecurityFilterChain
+ passwordEncoder(): PasswordEncoder
+ authenticationManager(): AuthenticationManager
+ authenticationProvider(): AuthenticationProvider
+ userDetailsService(): UserDetailsService
AuthController
+ AuthController():
+ signup(SignupForm, HttpServletResponse): ResponseEntity<?>
+ authenticateWithOAuth(OAuthForm, HttpServletResponse): ResponseEntity<?>
+ refreshToken(HttpServletRequest): ResponseEntity<?>
+ login(LoginForm, HttpServletResponse, HttpServletRequest): ResponseEntity<?>
+ logout(HttpServletRequest): ResponseEntity<?>

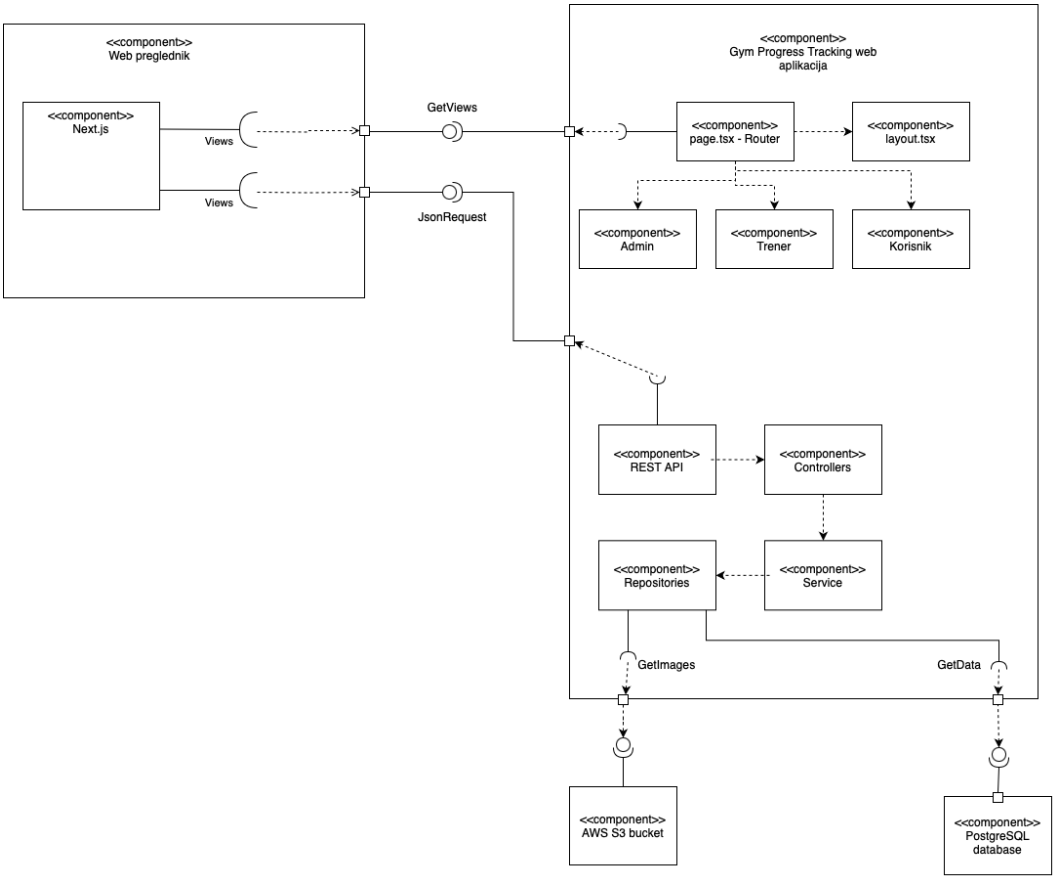
Dinamičko ponašanje aplikacije

UML dijagrami stanja

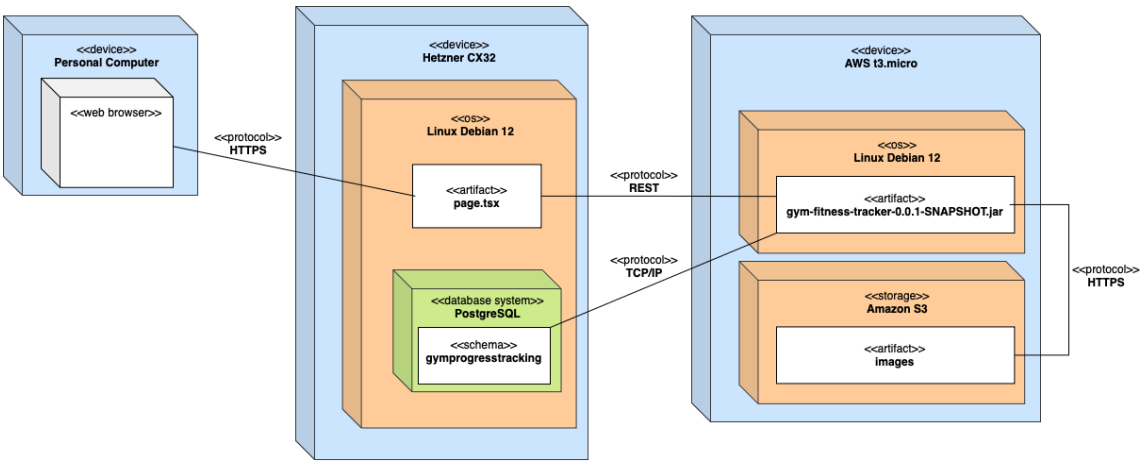
UML dijagrami aktivnosti

5. Arhitektura komponenata i razmještaja

Dijagram komponenata



Dijagram razmještaja



6. Ispitivanje programskog rješenja

Ispitivanje komponenti

Sanity check

Prvobitna provjera je li se Spring aplikacijski kontekst ispravno učitao. U većini slučajeva ako padne ovaj test, past će i sve ostale.

```
@Test
void contextLoads() {
}

- Očekivani izlaz: "Prolaz (Spring aplikacijski kontekst se ispravno učitao)"
- Dobiveni izlaz: "Prolaz"
```

FoodController

Početne postavke:

```
@Mock
private JwtService jwtService;

@Mock
private MyUserService myUserService;

@Mock
private FoodService foodService;

@InjectMocks
private FoodController foodController;

@BeforeEach
void setup() {
```

```
    MockitoAnnotations.openMocks(this);
}
```

1. Uspješno pretraživanje hrane (redovni slučaj)

```
@Test
void testSearchFood_Success() {
    // Arrange
    String token = "Bearer valid.jwt.token";
    String email = "user@example.com";
    String barcode = "123456789012";

    // Mock user data
    MyUser user = new MyUser();
    user.setEmail(email);

    // Create the Food object
    Food food = new Food(new FoodForm(
            "Test Food",
            200.0f,
            "g",
            100,
            5.0f,
            30.0f,
            10.0f,
            true
    ));

    // Spy on the Food object to mock only specific methods
    Food foodSpy = spy(food);

    // Mock the getId() method to return a non-null value
    when(foodSpy.getId()).thenReturn(1L);  // Mock the ID to return 1L

    // Mock the service calls
    when(jwtService.extractEmail("valid.jwt.token")).thenReturn(email);
    when(myUserService.getMyUser(email)).thenReturn(user);
    when(foodService.createFoodFromBarcode(user, barcode)).thenReturn(foodSpy);

    // Act
    ResponseEntity<?> response = foodController.searchFood(token, barcode);

    // Assert
    assertEquals(200, response.getStatusCodeValue());
    assertNotNull(response.getBody());
    assertTrue(response.getBody() instanceof FoodResponse);

    FoodResponse foodResponse = (FoodResponse) response.getBody();
    assertEquals(food.getName(), foodResponse.getName());
    assertEquals(food.getCalories(), foodResponse.getCalories());
    assertEquals(food.getCarbs(), foodResponse.getCarbs());
    assertEquals(food.getFat(), foodResponse.getFats());
```

```
    assertEquals(food.getProtein(), foodResponse.getProtein());
    assertEquals(food.getUnit(), foodResponse.getUnit());
    assertEquals(food.getDefaultNumber(), foodResponse.getDefaultNumber());

    // Verify the mock interactions
    verify(jwtService, times(1)).extractEmail("valid.jwt.token");
    verify(myUserService, times(1)).getMyUser(email);
    verify(foodService, times(1)).createFoodFromBarcode(user, barcode);
}
```

2. Spremanje hrane (redovni slučaj)

```
@Test
void testSaveFood() {
    // Arrange
    String token = "Bearer valid.jwt.token";
    String email = "test@example.com";
    FoodForm form = new FoodForm();
    MyUser user = new MyUser();
    Food food = new Food();
    food.setName("Pizza");
    food.setCalories(300);
    food.setCarbs(40);
    food.setFat(10);
    food.setProtein(12);
    food.setDefaultNumber(1);
    food.setApproved(true);

    // Spy on the Food object to mock specific methods (e.g., getId())
    Food foodSpy = spy(food);

    // Mock the getId() method to return a non-null value (1L)
    when(foodSpy.getId()).thenReturn(1L);

    // Prepare the expected response
    FoodResponse expectedResponse = new FoodResponse(foodSpy);

    // Mock the service calls
    when(jwtService.extractEmail("valid.jwt.token")).thenReturn(email);
    when(myUserService.getMyUser(email)).thenReturn(user);
    when(foodService.createFoodFromForm(user, form)).thenReturn(foodSpy);

    // Act
    ResponseEntity<?> response = foodController.saveFood(form, token);

    // Assert
    assertEquals(200, response.getStatusCodeValue());
    assertTrue(response.getBody() instanceof FoodResponse);

    // Extract the actual response body
    FoodResponse actualResponse = (FoodResponse) response.getBody();
```

```
    // Manually assert that the fields match
    assertEquals(expectedResponse.getName(), actualResponse.getName());
    assertEquals(expectedResponse.getCalories(), actualResponse.getCalories());
    assertEquals(expectedResponse.getCarbs(), actualResponse.getCarbs());
    assertEquals(expectedResponse.getFats(), actualResponse.getFats());
    assertEquals(expectedResponse.getProtein(), actualResponse.getProtein());
    assertEquals(expectedResponse.getDefaultNumber(),
actualResponse.getDefaultNumber());
    assertEquals(expectedResponse.isApproved(), actualResponse.isApproved());

    // Verify the mock interactions
    verify(jwtService, times(1)).extractEmail("valid.jwt.token");
    verify(myUserService, times(1)).getMyUser(email);
    verify(foodService, times(1)).createFoodFromForm(user, form);
}
```

NutrionController

Početne postavke:

```
@Mock
private JwtService jwtService;

@Mock
private MyUserService myUserService;

@Mock
private NutrionService myNutrionService;

@InjectMocks
private NutrionController nutritionPlanController;

@BeforeEach
void setup() {
    MockitoAnnotations.openMocks(this);
}
```

1. Dohvat postojećeg nutricionističkog plana (redovni slučaj)

```
@Test
void testGetNutritionPlan_Success() {
    // Arrange
    String token = "Bearer valid.jwt.token";
    String email = "test@example.com";

    MyUser user = new MyUser();
    user.setId(1L); // Ensure user ID is set
    user.setEmail(email);
    user.setRole(Role.USER); // Set the Role to avoid null issues

    NutrionPlan plan = new NutrionPlan();
    plan.setId(1L); // Set ID to avoid null issues
```

```
    plan.setCalories(2000);
    plan.setMyUser(user); // Ensure the NutrionPlan is linked to MyUser

    NutrionPlanResponse expectedResponse = new NutrionPlanResponse(plan);

    when(jwtService.extractEmail("valid.jwt.token")).thenReturn(email);
    when(myUserService.getMyUser(email)).thenReturn(user);
    when(myNutrionService.getMyPlan(user)).thenReturn(plan);

    // Act
    ResponseEntity<?> response = nutritionPlanController.getNutrionPlan(token);

    // Assert
    assertEquals(200, response.getStatusCodeValue());
    assertTrue(response.getBody() instanceof NutrionPlanResponse);

    NutrionPlanResponse actualResponse = (NutrionPlanResponse) response.getBody();
    assertEquals(expectedResponse.getId(), actualResponse.getId());
    assertEquals(expectedResponse.getCalories(), actualResponse.getCalories());

    verify(jwtService, times(1)).extractEmail("valid.jwt.token");
    verify(myUserService, times(1)).getMyUser(email);
    verify(myNutrionService, times(1)).getMyPlan(user);
}

- Očekivani izlaz:
Authorization header: Bearer valid.jwt.token
Extracted email: test@example.com
Found user: test@example.com
Retrieved nutrition plan:
GymFitnessTrackerApplication.model.domain.NutrionPlan@298d9a05
- Dobiveni izlaz:
Authorization header: Bearer valid.jwt.token
Extracted email: test@example.com
Found user: test@example.com
Retrieved nutrition plan:
GymFitnessTrackerApplication.model.domain.NutrionPlan@298d9a05
```

2. Pogrešan token (rubni slučaj)

```
@Test
void testGetNutritionPlan_InvalidToken() {
    // Arrange
    String token = "Bearer invalid.jwt.token";

    when(jwtService.extractEmail("invalid.jwt.token")).thenThrow(new
RuntimeException("Invalid token"));

    // Act & Assert
    RuntimeException exception = assertThrows(RuntimeException.class, () -> {
        nutritionPlanController.getNutrionPlan(token);
    });
```

```
    assertEquals("Invalid token", exception.getMessage());

    verify(jwtService, times(1)).extractEmail("invalid.jwt.token");
    verifyNoInteractions(myUserService);
    verifyNoInteractions(myNutrionService);
}

- Očekivani izlaz:
Received GET request to /api/nutrition-plan
Authorization header: Bearer invalid.jwt.token
Error in getNutrionPlan: Invalid token
- Dobiveni izlaz:
Received GET request to /api/nutrition-plan
Authorization header: Bearer invalid.jwt.token
Error in getNutrionPlan: Invalid token
```

3. Korisnik ažurira nutricionistički plani te dobiva error jer nije trener (rubni slučaj)

```
@Test
    void testUpdateNutritionPlan_UserNotTrainer_ThrowsAdminRestrictedException() {
        // Arrange
        String token = "Bearer valid.jwt.token";
        String email = "test@example.com";
        String planId = "1";

        // Mock a NutrionPlanForm with values to update
        NutrionPlanForm form = new NutrionPlanForm(2500f, 150f, 300f, 80f, "2025-01-
01", "2025-12-31");

        // Create a mock MyUser object with a non-trainer role
        MyUser user = new MyUser();
        user.setId(2L);  // User ID does not match "createdBy"
        user.setEmail(email);
        user.setRole(Role.USER);  // Role is not TRAINER

        MyUser user2 = new MyUser();
        user2.setId(3L);  // User ID does not match "createdBy"
        user2.setEmail(email+"n");
        user2.setRole(Role.USER);  // Role is not TRAINER

        // Create a mock NutrionPlan object
        NutrionPlan existingPlan = new NutrionPlan();
        existingPlan.setId(1L);  // Plan ID
        existingPlan.setMyUser(user2);
        existingPlan.setCreatedBy("2");  // "createdBy" does not match user ID

        // Mock service methods
        when(jwtService.extractEmail("valid.jwt.token")).thenReturn(email);
        when(myUserService.getMyUser(email)).thenReturn(user);
        when(myNutrionService.getPlanFromId(planId)).thenReturn(existingPlan);
```

```
        // Act & Assert
        AdminRestrictedException exception = assertThrows(
                AdminRestrictedException.class,
                () -> nutritionPlanController.updateNutrionPlan(form, token, planId)
        );

        // Verify the exception message
        assertEquals("USER nije trener", exception.getMessage());

        // Verify method invocations
        verify(jwtService, times(1)).extractEmail("valid.jwt.token");
        verify(myUserService, times(1)).getMyUser(email);
        verify(myNutrionService, times(1)).getPlanFromId(planId);
        verify(myNutrionService, never()).updateNutrionPlan(any(), any());  // Ensure
update was not called
    }

- Očekivani izlaz:
Received POST request to /api/nutrition-plan/{id}
Authorization header: Bearer invalid.jwt.token
Error in getNutrionPlan: Invalid token
- Dobiveni izlaz:
Received POST request to /api/nutrition-plan/{id}
Authorization header: Bearer invalid.jwt.token
Error in getNutrionPlan: Invalid token
```

4. Korisnik ažurira nutricionistički plani te dobiva error jer nije nađen plan (rubni slučaj)

```
@Test
    void testUpdateNutritionPlan_PlanNotFound() {
        // Arrange
        String token = "Bearer valid.jwt.token";
        String email = "test@example.com";
        String planId = "1";

        // Create a NutrionPlanForm with values to update
        NutrionPlanForm form = new NutrionPlanForm(2500f, 150f, 300f, 80f, "2025-01-
01", "2025-12-31");

        // Mock services
        MyUser user = new MyUser();
        user.setId(1L);
        user.setEmail(email);

        when(jwtService.extractEmail("valid.jwt.token")).thenReturn(email);
        when(myUserService.getMyUser(email)).thenReturn(user);
        when(myNutrionService.getPlanFromId(planId))
                .thenThrow(new NoNutrionPlanException("NO existing plan to update"));
// Throw exception here
```

```
        // Act & Assert
        NoNutrionPlanException exception = assertThrows(NoNutrionPlanException.class,
() -> {
            nutritionPlanController.updateNutrionPlan(form, token, planId);
        });

        assertEquals("NO existing plan to update", exception.getMessage()); // Ensure
correct exception message is thrown

        // Verify interactions
        verify(jwtService, times(1)).extractEmail("valid.jwt.token");
        verify(myUserService, times(1)).getMyUser(email);
        verify(myNutrionService, times(1)).getPlanFromId(planId);
        verify(myNutrionService, times(0)).updateNutrionPlan(any(), any()); // Ensure
update was NOT called
    }

- Očekivani izlaz:
Received POST request to /api/nutrition-plan/{id}
Authorization header: Bearer invalid.jwt.token
Error in getNutrionPlan: Invalid token
- Dobiveni izlaz:
Received POST request to /api/nutrition-plan/{id}
Authorization header: Bearer invalid.jwt.token
Error in getNutrionPlan: Invalid token
```

5. Korisnik stvara nutricionistički plani te dobiva error jer nije postojeći korisnik (rubni slučaj)

```
@Test
    void testPostNutritionPlan_UserNotFound() {
        // Arrange
        String token = "Bearer valid.jwt.token";
        String email = "test@example.com";

        // Create a NutrionPlanForm with values to post
        NutrionPlanForm form = new NutrionPlanForm(2500f, 150f, 300f, 80f, "2025-01-
01", "2025-12-31");

        // Mock services
        MyUser user = new MyUser();
        user.setId(1L);
        user.setEmail(email);

        // Mock the jwtService and myUserService
        when(jwtService.extractEmail("valid.jwt.token")).thenReturn(email);
        when(myUserService.getMyUser(email)).thenReturn(user);  // Simulate user found

        // Simulate the case where the plan is not created
        when(myNutrionService.createNutrionPlan(user, form)).thenThrow(new
RuntimeException("Error creating nutrition plan"));
```

```
        // Act & Assert
        Exception exception = assertThrows(RuntimeException.class, () -> {
            nutritionPlanController.postNutrionPlan(form, token);
        });

        assertTrue(exception.getMessage().contains("Error creating nutrition plan"));

        // Verify interactions
        verify(jwtService, times(1)).extractEmail("valid.jwt.token");
        verify(myUserService, times(1)).getMyUser(email);
        verify(myNutrionService, times(1)).createNutrionPlan(user, form);  // Ensure
plan creation was attempted
    }


- Očekivani izlaz:
Received POST request to /api/nutrition-plan/
Authorization header: Bearer invalid.jwt.token
Error in getNutrionPlan: Invalid token
- Dobiveni izlaz:
Received POST request to /api/nutrition-plan/
Authorization header: Bearer invalid.jwt.token
Error in getNutrionPlan: Invalid token
```

6. Korisnik stvara nutricionistički plani te je uspješan (redovni slučaj)

```
@Test
    void testPostNutritionPlan_Success() {
        // Arrange
        String token = "Bearer valid.jwt.token";
        String email = "test@example.com";

        // Create a NutrionPlanForm with values to post
        NutrionPlanForm form = new NutrionPlanForm(2500f, 150f, 300f, 80f, "2025-01-
01", "2025-12-31");

        // Mock services
        MyUser user = new MyUser();
        user.setId(1L);
        user.setEmail(email);
        user.setRole(Role.USER);  // Set the role to USER

        // Create a mock NutrionPlan object and populate it with values from the form
        NutrionPlan createdPlan = new NutrionPlan();
        createdPlan.setId(1L);  // Set the plan ID after creation
        createdPlan.setMyUser(user);  // Set the user to avoid null user error
        createdPlan.setCalories(form.getCalories());  // Set calories from the form
        createdPlan.setProtein(form.getProtein());  // Set protein from the form
        createdPlan.setCarbs(form.getCarbs());  // Set carbs from the form
        createdPlan.setFat(form.getFat());  // Set fat from the form
```

```
        createdPlan.setStartDate(LocalDate.parse(form.getStartDate()));  // Set start
date from the form
        createdPlan.setEndDate(LocalDate.parse(form.getEndDate()));  // Set end date
from the form

        // Mock service methods
        when(jwtService.extractEmail("valid.jwt.token")).thenReturn(email);
        when(myUserService.getMyUser(email)).thenReturn(user);
        when(myNutrionService.createNutrionPlan(user, form)).thenReturn(createdPlan);

        // Mock the removeCurrentPlan method to return a boolean (e.g., true)
       // when(myNutrionService.removeCurrentPlan(user)).thenReturn(true);

        // Act
        ResponseEntity<?> response = nutritionPlanController.postNutrionPlan(form,
token);

        // Assert
        assertEquals(200, response.getStatusCodeValue());  // Status should be OK
(200)
        assertNotNull(response.getBody());  // Response body should not be null
        assertTrue(response.getBody() instanceof NutrionPlanResponse);  // Response
should be of type NutrionPlanResponse

        NutrionPlanResponse responseBody = (NutrionPlanResponse) response.getBody();
        assertEquals("1", responseBody.getId());  // Check that the ID is set
correctly
        assertEquals("1", responseBody.getUserId());  // Check the user ID
        assertEquals(2500f, responseBody.getCalories(), 0.01f);  // Check calories
        assertEquals(150f, responseBody.getProtein(), 0.01f);  // Check protein
        assertEquals(300f, responseBody.getCarbs(), 0.01f);  // Check carbs
        assertEquals(80f, responseBody.getFat(), 0.01f);  // Check fat
        assertEquals("2025-01-01", responseBody.getStartDate().toString());  // Check
start date
        assertEquals("2025-12-31", responseBody.getEndDate().toString());  // Check
end date

        // Verify the service methods were called correctly
        verify(jwtService, times(1)).extractEmail("valid.jwt.token");
        verify(myUserService, times(1)).getMyUser(email);
        verify(myNutrionService, times(1)).createNutrionPlan(user, form);
        //verify(myNutrionService, times(1)).removeCurrentPlan(user);  // Verify
removeCurrentPlan was called
    }


- Očekivani izlaz:
Received POST request to /api/nutrition-plan/
Authorization header: Bearer invalid.jwt.token
Succesful creation of nutrion plan
- Dobiveni izlaz:
Received POST request to /api/nutrition-plan/
```

```
Authorization header: Bearer invalid.jwt.token
Succesful creation of nutrion plan
```

Ispitivanje sustava

Formular za prijavu:

1. Ispravno korisničko ime i lozinka (redovni slučaj)

```
- Koraci:
  1.  Otvoriti aplikaciju
  2.  Unijeti ispravno korisničko ime i lozinku (kcfakduswvbjevmoqz@hthlm.com,
kcfakduswvbjevmoqz@hthlm.com)
  3.  Kliknuti na "Sign-in"
  4.  Provjeriti je li korisnik preusmjeren na početnu stranicu aplikacije

- Očekivani izlaz: "Prijava uspješna."
- Dobiveni izlaz: "Korisnik je preusmjeren na početnu stranicu"
```

2. Ispravno korisničko ime, ali pogrešna lozinka (rubni slučaj)

```
- Koraci:
  1.  Otvoriti aplikaciju
  2.  Unijeti ispravno korisničko ime, ali pogrešnu lozinku
(kcfakduswvbjevmoqz@hthlm.com, 1234)
  3.  Kliknuti na "Sign-in"
  4.  Provjeriti je li korisnik preusmjeren na početnu stranicu aplikacije

- Očekivani izlaz: "Wrong username and/or password"
- Dobiveni izlaz: "Wrong username and/or password"
```

3. Neispravno korisničko ime, ali postojeća lozinka (rubni slučaj)

```
- Koraci:
  1.  Otvoriti aplikaciju
  2.  Unijeti pogrešno korisničko ime, ali postojeću lozinku (qedgncccshgttwgf,
kcfakduswvbjevmoqz@hthlm.com)
  3.  Kliknuti na "Sign-in"
  4.  Provjeriti je li korisnik preusmjeren na početnu stranicu aplikacije

- Očekivani izlaz: "Wrong username and/or password."
- Dobiveni izlaz: "Wrong username and/or password."
```

4. Nepostojeće korisničko ime i lozinka (rubni slučaj)

```
- Koraci:
  1.  Otvoriti aplikaciju
  2.  Unijeti pogrešno korisničko ime i lozinku (l;aksd;awlkd, wa;lkdsa)
  3.  Kliknuti na "Sign-in"
```

```
  4.  Provjeriti je li korisnik preusmjeren na početnu stranicu aplikacije

- Očekivani izlaz: "Wrong username and/or password."
- Dobiveni izlaz: "Wrong username and/or password."
```

5. SQL injection (rubni slučaj)

```
- Koraci:
  1.  Otvoriti aplikaciju
  2.  Unijeti SQL upit ('OR '1'='1' --, 'OR '1'='1' --)
  3.  Kliknuti na "Sign-in"
  4.  Provjeriti je li korisnik preusmjeren na početnu stranicu aplikacije

- Očekivani izlaz: "Wrong username and/or password."
- Dobiveni izlaz: "Wrong username and/or password."
```

6. Provera cookiesa (redovni slučaj)

```
- Koraci:
  1.  Otvoriti aplikaciju
  2.  Unijeti ispravno korisničko ime i lozinku (kcfakduswvbjevmoqz@hthlm.com,
kcfakduswvbjevmoqz@hthlm.com)
  3.  Kliknuti na "Sign-in"
  4.  Provjeriti je li korisnik preusmjeren na početnu stranicu aplikacije
  5.  Otvoriti novi tab i zatvoriti trenutni
  6.  U novom tabu opet otići na našu stranicu
  7.  Provjeriti jesmo li prijavljeni

- Očekivani izlaz: "Prijava uspješna."
- Dobiveni izlaz: "Presumjereni na početnu stranicu."
```

Trening

1. Pokretanje treninga (redovni slučaj)

```
  1.  Otvoriti aplikaciju
  2.  Unijeti ispravno korisničko ime i lozinku (kcfakduswvbjevmoqz@hthlm.com,
kcfakduswvbjevmoqz@hthlm.com)
  3.  Kliknuti na "Sign-in"
  4.  U lijevom izborniku odabrati srednji gumb (Workouts)
  5.  Odabrati neki trening i kliknuti na "Start workout"

- Očekivani izlaz: "Popis dostupnih vježbi"
- Dobiveni izlaz: "Popis dostupnih vježbi"
```

2. Dodavanje vježbe (redovni slučaj)

```
  1.  Otvoriti aplikaciju
  2.  Unijeti ispravno korisničko ime i lozinku (kcfakduswvbjevmoqz@hthlm.com,
kcfakduswvbjevmoqz@hthlm.com)
```

```
  3.  Kliknuti na "Sign-in"
  4.  U lijevom izborniku odabrati najgornji gumb (Workout Plans)
  5.  Kliknuti na "Edit your workout plan"
  6.  Kliknuti na "Add workout"
  7.  Pod Workout name upišemo naziv (npr. "test")
  8.  Kliknemo na "Create Workout"

- Očekivani izlaz: "Dodana je vježba unutar treninga"
- Dobiveni izlaz: "Dodana je vježba unutar treninga"
```

Prehrana

1. Dodavanje jela (redovni slučaj)

```
  1.  Otvoriti aplikaciju
  2.  Unijeti ispravno korisničko ime i lozinku (kcfakduswvbjevmoqz@hthlm.com,
kcfakduswvbjevmoqz@hthlm.com)
  3.  Kliknuti na "Sign-in"
  4.  U lijevom izborniku odabrati predzadnji gumb (Nutrition)
  5.  Kliknuti na "Add new meal"
  6.  Unijeti naziv jela
  7.  Kliknuti na "Add new food"
  8.  Po potrebi popuniti željena polja
  9.  Kliknuti na "Save food"
  10. Kliknuti na "Save Meal"

- Očekivani izlaz: "Dodano je jelo u graf"
- Dobiveni izlaz: "Dodano je jelo u graf"
```

Admin

1. Dodavanje plana treninga (redovni slučaj)

```
  1.  Otvoriti aplikaciju (dodati na kraj linka /admin)
  2.  U lijevom izborniku odabrati drugi gumb ("Workout Plans")
  3.  Kliknuti na "Dodaj novi trening plan"
  4.  Unjeti potrebne podatke
  5.  Kliknuti na "Create Plan"

- Očekivani izlaz: "Vježba je dodana u popis vježbi"
- Dobiveni izlaz: "Vježba nije dodana u popis vježbi"
```

2. Dodavanje jela (redovni slučaj)

```
  1.  Otvoriti aplikaciju (dodati na kraj linka /admin)
  2.  U lijevom izborniku odabrati srednji gumb ("Food")
  3.  Kliknuti na "Dodaj novu hranu"
  4.  Unjeti potrebne podatke
  5.  Kliknuti na "Dodaj"
```

```
- Očekivani izlaz: "Jelo je dodano u popis jela"
- Dobiveni izlaz: "Jelo je dodano u popis jela"
```

7. Tehnologije za implementaciju aplikacije

Korištene tehnologije i alati

Frontend

- **NextJS**: Next.js je napredni JavaScript okvir za React aplikacije koji omogućava server-side rendering (SSR) i statičko generiranje stranica. Osigurava brzinu, SEO optimizaciju i fleksibilnost, te dolazi s ugrađenom podrškom za rute, API-je i mnoge druge značajke koje olakšavaju razvoj kompleksnih web aplikacija.
- **Tailwind**: Tailwind CSS je moderni CSS framework koji koristi utility-first pristup, pružajući razne klase koje omogućuju brz i efikasan dizajn bez potrebe za pisanjem prilagođenog CSS-a. Daje programerima veliku fleksibilnost u stilizaciji, uz značajke za prilagodbu i optimizaciju.
- **React**: omogućava brzu izradu dinamičkih korisničkih sučelja putem komponentnog pristupa i virtualnog DOM-a
- **Visual Studio Code 1.96.3**: besplatan, open-source editor koda koji nudi bogat skup alata za razvoj aplikacija uz podršku za mnoge programske jezike i ekstenzije. Koristi se zbog svoje fleksibilnosti, brzine i moćnih funkcionalnosti kao što su debagiranje, kontrola verzija i integracija sa različitim alatima.

Backend

- **Spring Web**: Koristi se za razvoj REST API-ja i upravljanje HTTP zahtjevima, omogućujući izradu kontrolera za organizirano rukovanje zahtjevima u aplikaciji.
- **Spring Data JPA**: Omogućuje jednostavniji rad s bazom podataka kroz JPA repository sloj, automatizirajući upite i pristup podacima.
- **Spring Security**: Koristi se za autentifikaciju i zaštitu korisničkih podataka unutar aplikacije, osiguravajući pristup samo ovlaštenim korisnicima.
- **OAuth2 Client**: Pruža mogućnost korisnicima da se prijave putem vanjskih računa (Google, Apple, GitHub), pojednostavljujući proces autentifikacije.
- **IntelliJ 2023.3.2 - Ultimate**: IDEA se koristi kao moćno integrirano razvojno okruženje (IDE) koje programerima omogućava brzo i efikasno pisanje, pregledanje i debugging koda, posebno za Javu i povezane jezike. Pruža napredne funkcije kao što su automatsko dovršavanje koda, integracija alata za verzioniranje i analizu koda, čime se ubrzava razvojni proces.

Ostalo

- **PostgreSQL 16.6**: moćan, open-source sustav za upravljanje relacionim bazama podataka koji omogućava skladištenje, pretragu i manipulaciju velikom količinama podataka sa visokim performansama. Koristi se zbog svoje skalabilnosti, pouzdanosti i bogatog skupa funkcionalnosti kao što su podrška za složene upite, transakcije i ekstenzibilnost.

- **Amazon S3**: Amazon Simple Storage Service (S3) je skalabilna usluga za pohranu podataka u oblaku koja omogućuje spremanje i preuzimanje bilo koje količine podataka, bilo kada i bilo gdje. Koristi se za pohranu velikih količina statičkih podataka, kao što su slike, videozapisi, i sigurnosne kopije.
- **VPS**: Virtual Private Server (VPS) je virtualizirani poslužitelj koji korisnicima pruža privatne resurse unutar dijeljenog poslužitelja. Omogućuje veću kontrolu, prilagodljivost i sigurnost nego dijeljeni hosting, te se često koristi za hostanje aplikacija, web stranica i baza podataka.
- **Github Actions**: GitHub Actions je CI/CD platforma koja omogućuje automatizaciju tijeka rada na GitHub repozitorijima. Omogućuje programerima da automatski testiraju, grade i distribuiraju svoju aplikaciju nakon svakog unosa promjena u kod, čime se ubrzava razvoj i osigurava konzistentnost.
- **Git**: Git je distribuirani sustav za kontrolu verzija koji omogućava programerima praćenje promjena u kodu, suradnju u timovima i lako upravljanje različitim verzijama projekta.
- **Selenium 4.27**: open-source alat za automatizaciju web aplikacija koji omogućava testiranje interfejsa u različitim web preglednicima. Koristi se za pisanje testova koji simuliraju korisničke interakcije, čime omogućava efikasno testiranje i osiguranje kvaliteta web aplikacija
- **Postnam 11.27**: alat za testiranje i razvoj API-ja koji omogućava slanje zahtjeva, analizu odgovora i automatizaciju testova. Koristi se zbog svoje jednostavnosti, moćnih funkcionalnosti za debugging i podrške za kolaboraciju među timovima

8. Upute za puštanje u pogon

1. Instalacija

```
git clone https://github.com/Petar-Babic/CodeMonkeys.git
cd CodeMonkeys/IzvorniKod/frontend
npm install
```

2. Postavke

Baza podataka

Potrebno je kreirati bazu podataka (PostgreSQL) na VPS serveru ili korištenjem DigitalOcean, Supabase i sličnih platformi.

Upute za kreiranje na VPS-u (Linux Ubuntu):

```
sudo apt update
```

Naredba za instalaciju:

```
apt install postgresql postgresql-contrib
```

Provjera statusa:

```
service postgresql status
```

Pokrenuti CLI tool:

```
sudo -u postgres psql
```

Kreiranje korisnika

Unutar PostgreSQL CLI alata:

1. Kreirati korisnika s odabranim korisničkim imenom i lozinkom:

   ```
   CREATE USER korisnicko_ime WITH PASSWORD 'lozinka';
   ```

2. Dodijeliti korisniku privilegije (po potrebi):
   - Superuser privilegije:

     ```
     ALTER USER korisnicko_ime WITH SUPERUSER;
     ```

   - Ili samo kreiranje baza:

     ```
     ALTER USER korisnicko_ime CREATEDB;
     ```

Kreiranje baze podataka

1. Kreirati bazu podataka i dodijeliti vlasnika:

   ```
   CREATE DATABASE ime_baze OWNER korisnicko_ime;
   ```

2. Provjeriti postojeće baze:

   ```
   \l
   ```

3. Provjeriti korisnike:

   ```
   \du
   ```

Konfiguracija za pristup s drugih IP adresa

1. Uređivanje `postgresql.conf` filea za omogućavanje vanjskog pristupa:

   ```
   sudo nano /etc/postgresql/16/main/postgresql.conf
   ```

   Urediti redak:

   ```
   listen_addresses = '*'
   ```

2. Uređivanje `pg_hba.conf` filea za definiranje pravila pristupa:

   ```
   sudo vim /etc/postgresql/16/main/pg_hba.conf
   ```

   Dodati pravilo za određeni IP raspon (ili sve IP adrese):

   ```
   host    all             all             0.0.0.0/0               md5
   ```

Restart PostgreSQL servisa

Primijeniti promjene:

```
systemctl restart postgresql
```

Prikupljanje podataka za pristup bazi

Nakon toga imamo sve najbitnije informacije za pristup bazi:

- **IP adresa**: [ip address]
- **Port**: [port] (obično 5432)
- **Ime baze**: [database]
- **Korisničko ime**: [username]
- **Lozinka**: [password]

Backend

Potrebno je namjestiti application.properties datoteku koja se nalazi: `CodeMonkeys/IzvorniKod/backend/src/main/resources/`:

```
spring.datasource.url="vaš URL od baze podataka"
spring.datasource.username="vaše korisničko ime od baze podataka"
spring.datasource.password="vaša lozinka od baze podataka"
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update

# Email configuration
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username="vaše korisničko ime od maila"
spring.mail.password="vaša lozinka od maila"
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

#AWS Configuration
cloud.aws.credentials.accessKey="vaš AWS access key"
cloud.aws.credentials.secretKey="vaš AWS secret key"
cloud.aws.region.static="regija vašeg AWS servera"
cloud.aws.s3.bucket="vaš AWS S3 bucket"
```

Frontend

```
# GOOGLE
GOOGLE_CLIENT_ID="vaš google client id"
# project_id="gym-progress-tracking-app"
# auth_uri="https://accounts.google.com/o/oauth2/auth"
# token_uri="https://oauth2.googleapis.com/token"
# auth_provider_x509_cert_url="https://www.googleapis.com/oauth2/v1/certs"
GOOGLE_CLIENT_SECRET="vaš google secret"
# redirect_uris_0="http://localhost:3000/api/auth/callback/google"
```

```
# javascript_origins_0="http://localhost:3000"

# NEXTAUTH
NEXTAUTH_SECRET="vaš nextauth secret"
NEXTAUTH_URL=http://localhost:3000


# FACEBOOK
FACEBOOK_CLIENT_ID="vaš client ID od facebook-a"
FACEBOOK_CLIENT_SECRET="vaš facebook secret"

AWS_ACCESS_KEY_ID="vaš access key ID"
AWS_SECRET_ACCESS_KEY="vaš access key"
AWS_REGION=us-east-1
AWS_BUCKET_NAME=gym-progress-app
PORT=3001
```

3. Pokretanje aplikacije

Instalacija potrebnih modula:

```
npm install
```

Razvojno okruženje:

```
npm run dev
```

Produkcijsko okruženje:

Prevođenje aplikacije:

```
npm run build
```

Pokretanje poslužitelja:

```
npm start
```

Provjera rada: http://localhost:3000/

4. Upute za administratore

`Smjernice za administratore aplikacije nakon puštanja u pogon:`

`Pristup administratorskom sučelju:`

`URL za admin panel (npr. /admin).`

`Početni podaci za prijavu (ako postoje).`

`Redovito održavanje:`

`Arhiviranje baze podataka.`

`Pregled logova.`

`Ažuriranje aplikacije (primjer: povlačenje novih verzija iz Git repozitorija i ponovno pokretanje aplikacije).`

`git pull origin main`

`npm install`

`npm run build`

`npm start` `Rješavanje problema: Kako pristupiti logovima i dijagnosticirati greške (npr. logs/error.log ili docker logs).`

5. AWS i Hetzner deploy (Cloud Deploy)

AWS (backend)

maven.yml

Osigurajte da vaš projekt koristi github CI/CD pipeline i izmjenite datoteku prema navedenom primjeru kako bi se ispravno napravila konfiguracija deploya:

```
name: Java CI with Maven

on:
  workflow_dispatch:
  push:
    branches: ["dev"]
    paths:
      - "IzvorniKod/backend/**"

jobs:
  build:
    runs-on: self-hosted

    steps:
      - uses: actions/checkout@v4
        with:
          ref: dev

      - name: Generate application.properties
        run: |
            echo "spring.application.name=gym-fitness-tracker
            spring.datasource.url=${{ secrets.DB_URL }}
            spring.datasource.username=${{ secrets.DB_USERNAME }}
            spring.datasource.password=${{ secrets.DB_PASSWORD }}
            spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
            spring.jpa.hibernate.ddl-auto=update
```

```
            # Email configuration
            spring.mail.host=smtp.gmail.com
            spring.mail.port=587
            spring.mail.username=${{ secrets.MAIL_USERNAME }}
            spring.mail.password=${{ secrets.MAIL_PWD }}
            spring.mail.properties.mail.smtp.auth=true
            spring.mail.properties.mail.smtp.starttls.enable=true

            #AWS Configuration
            cloud.aws.credentials.accessKey=${{ secrets.AWS_ACCESS_KEY }}
            cloud.aws.credentials.secretKey=${{ secrets.AWS_SECRECT_KEY }}
            cloud.aws.region.static=${{ secrets.AWS_REGION }}
            cloud.aws.s3.bucket=${{ secrets.AWS_S3_BUCKET }}
            " > IzvorniKod/backend/src/main/resources/application.properties

      - name: Set up JDK 17
        uses: actions/setup-java@v4
        with:
          java-version: "17"
          distribution: "temurin"

      - name: Build with Maven
        run: mvn -B package --file IzvorniKod/backend/pom.xml

      - name: Execute Jar File
        run: sudo kill $(sudo lsof -t -i:8080) & sudo java -jar /home/admin/actions-runner/_work/CodeMonkeys/CodeMonkeys/IzvorniKod/backend/target/gym-fitness-tracker-0.0.1-SNAPSHOT.jar &

      - name: Edit permissions
        run: sudo chmod -R 777 /home/admin/actions-runner/_work/CodeMonkeys
```

Postavljanje na AWS

- Prijavite se na AWS
- **Kreirajte VPC**:
  - Odaberite 'VPC and more'
  - Upišite naziv vašeg VPC-a, (npr. vpc-backend)
  - Ostale postavke ostavite kakve su bile zadana
  - Kliknete na 'Create VPC'
- **Kreirajte EC2 instancu**:
  - Prvo je potrebno izgenerirati SSH ključeve kako bi se mogli spojiti na server
    - U lijevom izborniku odaberite: Network & Security -> Key Pairs
    - Kliknite na 'Create key pair'
    - Upišite vlastiti naziv para ključa (npr. ssh-backend)
    - Ostale postavke ostavimo na njihovim početnim vrijednostima
    - Kliknemo na 'Create key pair'
    - **BITNO** pohranite negdje sigurno .pem datoteku, ako ju izgubite više se nemožete prijaviti na EC2 instancu
  - Sada je potrebno kreirati sigurnosnu grupu

        - U lijevom izborniku odaberite: Network & Security -> Security Groups
        - Kliknite na 'Create security group'
        - Upišite vlastiti naziv sigurnosne grupe (npr. security-group-backend)
        - Za VPC odaberete ranije kreirani VPC
        - Inbound rules (potrebno je omogućiti SSH pristup, kao i pristup samom backendu):
            - Type: SSH, Source: Anywhere-IPv4
            - Type: Custom TCP, Port: 8080, Source: Anywhere-IPv4
            - Type: HTTPS, Port: 443, Source: Anywhere-IPv4
            - Type: HTTP, Port: 80, Source: Anywhere-IPv4
        - Ostale postavke ne diramo
        - Kliknemo na 'Create security group'
    - Napokon možemo kreirati EC2 instancu:
        - U lijevom izborniku odaberite: Instances -> Instances
        - Kliknite na 'Launch instances'
        - Upišite vlastiti naziv (npr. EC2-backend)
        - Unutar 'Application and OS Images (Amazon Machine Image)' odaberite Quick Start -> Debian
        - Za AMI odaberite 'Free tier'
        - Unutar 'Key pair (login)' odaberite ranije kreirani SSH ključ
        - Network settings:
            - Kliknite na 'Edit'
            - VPC: odaberite ranije kreirani VPC
            - Auto-assign public IP: Enable (Limit za free tier je 750h unutar prvih 12 mjeseci)
            - Firewall (security groups): Select existing security group
            - Common security groups: odaberite ranije kreiranu sigurnosnu grupu
            - Kliknite na 'Launch instance'

- **Alocirajte statičku IP adresu za pristup serveru**:
    - U seach upišite 'Elastic IP'
    - Kliknite na 'Elastic IPs - EC2 feature'
    - Kliknite na 'Allocate Elsatic IP address
        - Postavke ostavimo na početnim vrijednostima
        - Kliknite na 'Allocate'
    - Odaberite kreiranu Elastic IP instancu
        - Actions -> Associate Elastic IP address
        - Instance -> Odaberite ranije kreiranu EC2 instancu

Spajanje s EC2 instancom putem SSH (na MacOs-u):

- otvorite terminal i upišite: `sudo ssh -i "putanja do .pem datoteke" admin@"Elastic IP adresa"`

Povezivanje github-a i EC2 instance kako bi se omogućio automatski Deployment:

- Otvorite vaš github repozitorij i kliknite na 'Settings'

- U lijevom izborniku: Code and automation -> Actions -> Runners
- Kliknite na: 'New self-hosted runner'
  - Odaberite Linux
  - Spojite se na server putem SSH i copy/paste 'Download' i 'Configure' naredbe

---

Hetzner (frontend)

1. Kreiranje VPS servera

2. Instalacija potrebnih alata na serveru

Ažuriranje paketa

```
sudo apt update && sudo apt upgrade -y
```

Instalacija Node.js i npm-a

1. Dodajte Node.js repozitorij:

   ```
   curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
   ```

2. Instalirajte Node.js i npm:

   ```
   sudo apt install -y nodejs
   ```

3. Provjerite verzije:

   ```
   node -v
   npm -v
   ```

Instalacija PM2

PM2 je alat za pokretanje i upravljanje Node.js aplikacijama.

```
sudo npm install -g pm2
```

3. Kupovina domene

1. Kupite domenu putem platformi poput Namecheap, GoDaddy ili sličnih.
2. Povežite domenu s VPS-om putem DNS postavki (postavite A zapis na IP adresu VPS-a).

4. Instalacija i konfiguracija Nginxa

Instalacija Nginxa

```
sudo apt install nginx -y
```

Konfiguracija Nginxa za Next.js

1. Kreirajte novu konfiguracijsku datoteku:

   ```
   sudo nano /etc/nginx/sites-available/frontend
   ```

2. Dodajte sljedeću konfiguraciju:

```
server {
    server_name yourdomain.com www.yourdomain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    listen 80;
}
```

3. Aktivirajte konfiguraciju:

```
sudo ln -s /etc/nginx/sites-available/frontend /etc/nginx/sites-enabled/
```

4. Testirajte konfiguraciju:

```
sudo nginx -t
```

5. Ponovno pokrenite Nginx:

```
sudo systemctl restart nginx
```

5. Instalacija SSL certifikata (Certbot)

1. Instalirajte Certbot:

```
sudo apt install certbot python3-certbot-nginx -y
```

2. Kreirajte SSL certifikat:

```
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com
```

3. Testirajte automatsko obnavljanje certifikata:

```
sudo certbot renew --dry-run
```

Opis prisutpa aplikaciji na javnom poslužitelju

`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.`

Odlaskom direktno na domenu gymprogress.lukakordic.me .

`Priložite korake za pristup administratorskom sučelju ako je primjenjivo.`

Odlaskom na podstranicu gymprogress.lukakordic.me/admin nakon prvotne prijave u sustav.

9. Zaključak i budući rad

Izrada projekta trajala je nekoliko mjeseci, pri čemu je tim intenzivno surađivao i održavao redovite sastanke kako bi se osiguralo praćenje svih faza razvoja. No naravno da nije sve bilo tako lijepo i bajno, za vrijeme izrade projekta susreli smo se s mnogim problemima i rupama u našem znanju, kao: prvo doticaj s Next.jsom i Java Springom, autentifikacijom putem OAuth 2.0 standarda, usklađivanjem frontend i backend... Unatoč tim problemima uspjeli smo kao tim izaći jači i pametniji, ključ svega toga bio je timski rad i naravno Googleanje. Smatramo da smo posadili čvrste temelje za daljnji razvoj web aplikacije.

Tehnički izazovi:

- Integracija OAuth 2.0 autentifikacije i autorizacije: Ovo je bio jedan od složenijih dijelova implementacije, no uspješno je riješen kroz istraživanje i postavljanje odgovarajuće konfiguracije na frontendu i backendu.
- JWT autentifikacija i autorizacija: Implementacija ovih funkcionalnosti zahtijevala je napredno razumijevanje sigurnosnih protokola te povezivanje s bazom podataka, što je uspješno realizirano.
- Postavljanje servera i deployment: Proces konfiguracije AWS EC2 instance te automatizacije deploya putem GitHub Actionsa bio je tehnički zahtjevan. Ovaj izazov je riješen kroz temeljitu pripremu i primjenu alata poput Nginxa i Certbota za SSL certifikate.
- Komunikacija između frontenda i backenda: Poteškoće su se pojavljivale zbog nesinkroniziranog razvoja, ali redoviti sastanci i razmjena informacija omogućili su rješavanje problema.

Stečena znanja:

- Kroz izradu projekta stečeno je iskustvo u radu s tehnologijama poput Java Spring Boota, Next.js-a, AWS-a, te u korištenju alata za upravljanje projektima kao što su GitHub Actions i Swagger.io.
- Članovi tima usvojili su dublje razumijevanje arhitekture web aplikacija, dizajna baza podataka, te korištenja sigurnosnih protokola za autentifikaciju.
- Poboljšane su vještine timskog rada, planiranja resursa i upravljanja projektima.

Potrebna znanja za brži i kvalitetniji rad:

- Naprednije poznavanje sigurnosnih standarda za autentifikaciju i autorizaciju moglo je ubrzati implementaciju tih funkcionalnosti.
- Više iskustva s deployment alatima i server konfiguracijama (npr. Docker) omogućilo bi brži i jednostavniji postupak postavljanja aplikacije na produkciju.

- Bolje razumijevanje dizajna korisničkog sučelja (UI/UX) doprinijelo bi intuitivnijem korisničkom iskustvu.

Funkcionalnosti koje nisu implementirane:

- večinski dio nutricionističkog plana
- ne može se ostaviti recenzija kada korisnik napravi trening

A. Dnevnik promjena dokumentacije

Opis promjene/dodatka	Autori	Datum
Updated B. Prikaz aktivnosti grupe (markdown)	Petar Babić	Jan 14, 2024
Updated B. Prikaz aktivnosti grupe (markdown)	Petar Babić	Jan 13, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 13, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 13, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 13, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 13, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 13, 2024
Updated 6. Ispitivanje programskog rješenja (markdown)	Petar Babić	Jan 13, 2024
Updated 5. Arhitektura komponenata i razmještaja (markdown)	Petar Babić	Jan 13, 2024
Updated B. Prikaz aktivnosti grupe (markdown)	Petar Babić	Jan 12, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 11, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 11, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 7, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 7, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 7, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 7,

		2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 7, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 7, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 7, 2024
Updated 5. Implementacija i korisničko sučelje (markdown)	Petar Babić	Jan 7, 2024
Created 6. Ispitivanje programskog rješenja (markdown)	Petar Babić	Jan 7, 2024
Updated 7. Tehnologije za implementaciju aplikacije (markdown)	Petar Babić	Jan 7, 2024
Updated 7. Tehnologije za implementaciju aplikacije (markdown)	Petar Babić	Jan 7, 2024
Created 8. Upute za puštanje u pogon(markdown)	Petar Babić	Jan 7, 2024
Updated 6. Zaključak i budući rad (markdown)	Petar Babić	Jan 7, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Jan 7, 2024
Updated 7. Popis literature (markdown)	Petar Babić	Jan 7, 2024
Created 7. Tehnologije za implementaciju aplikacije (markdown)	Petar Babić	Jan 7, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Jan 7, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Leonardo Cigula	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Leonardo Cigula	Nov 15, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Leonardo Cigula	Nov 15, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 15, 2024
Updated 7. Popis literature (markdown)	Petar Babić	Nov 15, 2024

Updated 9. Prikaz aktivnosti grupe (markdown)	Ema Adamec	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Ema Adamec	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Ema Adamec	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Sonja Cikovic	Nov 15, 2024
updated tablica	Gabrijela Klepec	Nov 15, 2024
updated Tablica	Gabrijela Klepec	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 15, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Matija Martinović	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Kordic Luka	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Matija Martinović	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Matija Martinović	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Matija Martinović	Nov 15, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 14, 2024

Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 14, 2024
Updated 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 14, 2024
Created A. Dnevnik promjena dokumentacije (markdown)	Petar Babić	Nov 14, 2024
Created 9. Prikaz aktivnosti grupe (markdown)	Petar Babić	Nov 14, 2024
Updated 7. Popis literature (markdown)	Matija Martinović	Nov 14, 2024
Updated 7. Popis literature (markdown)	Matija Martinović	Nov 14, 2024
Updated 7. Popis literature (markdown)	Petar Babić	Nov 14, 2024
Updated 7. Popis literature (markdown)	Petar Babić	Nov 13, 2024
Updated 7. Popis literature (markdown)	Petar Babić	Nov 13, 2024
Created 7. Popis literature (markdown)	Petar Babić	Nov 12, 2024
Updated 6. Implementacija i korisničko sučelje (markdown)	Petar Babić	Nov 12, 2024
Updated Implementacija i korisničko sučelje (markdown)	Petar Babić	Nov 12, 2024
Created Implementacija i korisničko sučelje (markdown)	Petar Babić	Nov 12, 2024
Updated 7. Zaključak i budući rad (markdown)	Petar Babić	Nov 12, 2024
Created 7. Zaključak i budući rad (markdown)	Petar Babić	Nov 12, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 11, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 9, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Petar Babić	Nov 9, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 4, 2024
Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 4, 2024

Updated 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 4, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Petar Babić	Nov 4, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Petar Babić	Nov 4, 2024
Updated Home (markdown)	Petar Babić	Nov 4, 2024
Updated Home (markdown)	Petar Babić	Nov 4, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Leonardo Cigula	Nov 4, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Leonardo Cigula	Nov 4, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Leonardo Cigula	Nov 4, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Leonardo Cigula	Nov 4, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Leonardo Cigula	Nov 4, 2024
Created 4. Arhitektura i dizajn sustava (markdown)	Petar Babić	Nov 3, 2024
Updated 3. Specifikacija zahtjeva sustava (markdown)	Petar Babić	Nov 3, 2024
Created 3. Specifikacija zahtjeva sustava (markdown)	Petar Babić	Nov 3, 2024
Updated 2. Analiza zahtjeva (markdown)	Petar Babić	Nov 3, 2024
Updated 2. Analiza zahtjeva (markdown)	Petar Babić	Nov 3, 2024
Created _Footer (markdown)	Petar Babić	Nov 3, 2024
Created 2. Analiza zahtjeva (markdown)	Petar Babić	Nov 3, 2024
Created 1. Opis projektnog zadatka (markdown)	Petar Babić	Nov 3, 2024
Created Home (markdown)	Petar Babić	Nov 3, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Petar Babić	Oct 29, 2024

Updated GYM PROGRESS TRACKING APP (markdown)	Petar Babić	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Kordic Luka	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Kordic Luka	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Kordic Luka	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Kordic Luka	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Kordic Luka	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Petar Babić	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Petar Babić	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Petar Babić	Oct 28, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Kordic Luka	Oct 21, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Kordic Luka	Oct 21, 2024
Updated GYM PROGRESS TRACKING APP (markdown)	Kordic Luka	Oct 21, 2024
Updated Home (markdown)	Kordic Luka	Oct 21, 2024
Updated Home (markdown)	Kordic Luka	Oct 21, 2024
Updated Home (markdown)	Kordic Luka	Oct 21, 2024
Updated Home (markdown)	Kordic Luka	Oct 21, 2024
Updated Home (markdown)	Kordic Luka	Oct 21, 2024
Updated Home (markdown)	Kordic Luka	Oct 21, 2024
Initial Home page	Petar Babić	Oct 20, 2024

A. Popis literature

Frontend

1. Next.js docs
2. NextAuth.js docs

Backend

1. CROZ predavanja
2. Spring Boot Security - configuration and role based authorization
3. Spring Boot Security - JWT authentication and authorization
4. Testing the Web Layer
5. Java Unit Testing with JUnit - Tutorial

Deployment

1. How to Easily Deploy a Spring Boot Application to AWS EC2
2. Auto Deploy Spring Boot Project Using GitHub Actions to AWS EC2
3. How To Set Up Let's Encrypt with Nginx Server
4. How to inject github secrets in application.properties file of Spring boot application?

Dokumentacija

1. Draw.io
2. Eraser

B. Prikaz aktivnosti grupe

Dnevnik

---

1. sastanak

- Datum: 14. listopada 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula, Luka Kordić, Ema Adamec, Sonja Ćiković, Gabrijela Klepec
- Teme sastanka: upoznavanje članova i uspostava osnovnog komunikacijskog kanala

> *opis prve teme:*
>
> > *članovi su se međusobno upoznali*
> >
> > *opis druge teme uspostavljena je Whatsapp grupa*

---

2. sastanak

- Datum: 16. listopada 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula, Luka Kordić, Ema Adamec, Sonja Ćiković, Gabrijela Klepec
- Teme sastanka: odabir vlastite teme, uspostava novih komunikacijskih kanala

> *opis prve teme:*
>
> > *Rasprava koju temu je najbolje izabrati za web stranicu (Gym Progress Tracking ili LetterBoxd x Last.fm ili već predloženu)*
> >
> > *opis druge teme Uspostava Discord kanala*

---

3. sastanak

- Datum: 21. listopada 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula, Luka Kordić, Ema Adamec, Sonja Ćiković, Gabrijela Klepec
- Teme sastanka: prezentacija projekta

> *opis prve teme:*
>
> > *prezentirana je ideja aplikacije Gym Fitness Tracking*

---

4. sastanak

- Datum: 22. listopada 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula, Luka Kordić, Ema Adamec, Sonja Ćiković
- Teme sastanka: odabir tehnologija, podjela poslova, dogovor oko API-ja

> *opis prve teme:*
>
> > *za frontend je odabran next.js za backend je odabrana Java Spring*
> >
> > *opis druge teme: frontend: Luka Kordić, Ema Adamec, Sonja Ćiković, Gabrijela Klepec*
> >
> > *backend: Matija Martinović, Petar Babić, Leonardo Cigula*
> >
> > *deployment: Petar Babić (backend), Luka Kordić (frontend)*
> >
> > *dokumentacija: Petar Babić*
> >
> > *opis treće teme: za api dokumentaciju je dogovoreno da će se koristiti swagger.io*

---

5. sastanak

- Datum: 28. listopada 2024.
- Prisustvovali: Petar Babić, Leonardo Cigula, Luka Kordić, Ema Adamec
- Teme sastanka: UML dijagrami

> *opis prve teme:*
>
> > *asistentu su prezentirani UML dijagrami*

---

6. sastanak

- Datum: 29. listopada 2024.
- Prisustvovali: Luka Kordić, Ema Adamec, Sonja Ćiković, Gabrijela Klepec
- Teme sastanka: opis frontend-a

> *opis prve teme:*
>
> > *poslovi su detaljnije podijeljeni po članovima objašnjeno je u kojem ćemo smjeru ići*

---

7. sastanak

- Datum: 31. listopada 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula
- Teme sastanka: opis backend-a

> *opis prve teme:*
>
> > *poslovi su detaljnije podijeljeni po članovima objašnjeno je u kojem ćemo smjeru ići*

---

8. sastanak

- Datum: 4. studenog 2024.
- Prisustvovali: Petar Babić, Leonardo Cigula
- Teme sastanka: popravljeni UML dijagrami i novi ER dijagram i plan baze podataka

> *opis prve teme:*
>
> > *prezentirani su dorađeni UML dijagrami*
> >
> > *opis druge teme: prezentirana je zamišljena baza podataka*

---

9. sastanak

- Datum: 9. studenog 2024.
- Prisustvovali: Luka Kordić, Gabrijela Klepec
- Teme sastanka: pomoć oko namještanja frontend-a

> *opis prve teme:*
>
> > *objašnjeno je kako krenuti s frontend-om*

---

10. sastanak

- Datum: 11. studenog 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula, Luka Kordić, Ema Adamec, Sonja Ćiković, Gabrijela Klepec
- Teme sastanka: prezentacija aplikacije

> *opis prve teme:*
>
> > *prezentirana je početna verzija aplikacije, kao i njena osnovna funkcionalnost*

---

11. sastanak

- Datum: 12. siječnja 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula, Luka Kordić

- Teme sastanka: izmjena baze podataka

> *opis prve teme:*
>
> > *baza podataka je izmjenjena*

---

12. sastanak

- Datum: 13. siječnja 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula, Luka Kordić, Ema Adamec, Sonja Ćiković, Gabrijela Klepec
- Teme sastanka: prezentacija alfa verzije aplikacije

> *opis prve teme:*
>
> > *prezentirana je alfa verzija aplikacije*

---

13. sastanak

- Datum: 13. siječnja 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula, Luka Kordić
- Teme sastanka: poboljšanja API-ija

> *opis prve teme:*
>
> > *pronađene su i identificirane glavne pogreške u API-ju i popravljene*

---

14. sastanak

- Datum: 20. siječnja 2024.
- Prisustvovali: Matija Martinović, Petar Babić, Leonardo Cigula
- Teme sastanka: zadnja laboratorijska vježba prije kolokviranja

> *opis prve teme:*
>
> > *pronađene su i identificirane glavne pogreške u dijagramu stanja*

Tablica aktivnosti

Matija Martinović

1. ciklus

Komponenta	Opis rada	Uloženo sati
Backend	dodavanje početnih funkcionalnosti servera putem Controllera i klase/entiteta User (dodavanje u bazu, pronalazak iz baze, posebni exception...)	2
Backend	konfiguracija sigurnosti web aplikacije, dodavanje	4

	autentifikacije i autorizacije na razini korisnika	
Backend	dodavanje autentifikacije i autorizacije putem JWT-a	6
Backend	postavljanje mogućnosti autentifikacije putem OAuth2-a	14

2. ciklus

Komponenta	Opis rada	Uloženo sati
Backend	funkcionalnosti dodavanja, ažuriranja, brisanja i svih pregleda entiteta workout plan, workout, planned exercise	15
Backend	funkcionalnost dodavanja, ažuriranja, brisanja i svih pregleda workout session-a, performed exercise-a i svakog seta	10
Backend	funkcionalnosti vezane za exercise i muscle group	6
Backend	pregled i pokusaj implementacije AWS S3-a radi file uploada	4
Baza podataka	refaktoriranje baze radi boljih povezanosti potrebnih entiteta	3

Petar Babić

1. ciklus

Komponenta	Opis rada	Uloženo sati
Ostalo	Namještanje i održavanje GitHub repozitorija	4
Dokumentacija	Pisanje, održavanje i uljepšavanje wiki stranice GitHub-a	5
Dokumentacija	Nacrtani i dorađeni dijagrami obrazaca uporabe	2
Deployment	Postavljen backend na AWS EC2 instancu	9
Deployment	Povezana EC2 instanca s domenom (putem nginx) i namješten SSL (pomoću certbot-a)	2
Deployment	Namješteni github actions-i kako bi se backend automatski deployao na AWS server nakon pushanja na dev branch i korištenje GitHub sercets-a kako bi se sakrile bitne informacije (username i password od baze, secrect-i od Facebook-a i Google-a)	3

2. ciklus

Komponenta	Opis rada	Uloženo sati
Testovi	Smišljeni i napravljeni početni JUnit testovi	8

Testovi	Smišljeni i napravljeni selenium testovi	10
Dokumentacija	Dodavanje zadanih stavki na GitHub wiki	5
Dijagrama	Izrada: gantogram, komponenata, razmještaja, razreda, stanja i aktivnosti	10
Deployment	Održavanje backenda na AWSu	2

Leonardo Cigula

1. ciklus

Komponenta	Opis rada	Uloženo sati
Tablica oblikovnih obrazaca	Izrada navedene kompomente te popravak navedene komponente nakon prve prezentacije; svi use caseovi popisani po tablicama	5
Dijagram razreda	Izrada konceptualnog i trenutnog dijagrama razreda za našu aplikaciju	2
Inicijaliziranje backend projekta	Inicijalizirao Spring boot projekt	1
Login i register	Napravio registraciju i login prema dogovorenom API	5
Restruktiranje	Popravio strukturu backend projekta	0.5
Workout i Nutrition	Napravio kontolere za workout i nutrition	1
UserDataController	Implementirao sve komponente da bi se moglo upisivati podatke o korisniku u bazu te modele,servise te sve ostalo za taj dio	10

2. ciklus

Komponenta	Opis rada	Uloženo sati
Refresh Token	Napravljen refresh token za apk	5
Email Servis	Napravio simple mail servis	2
Testovi	MyUnit testovi	2
Refaktoriranje baze	Refaktirao bazu za vlastite potrebe	2
BACKEND	Java Spring Backend za nutrition, user data , food i meals etc	30

Luka Kordić

1. ciklus

Komponenta	Opis rada	Uloženo sati
Frontend	Implementacija početne stranice, uvjeta korištenja i politike privatnosti	1
Frontend	Razvoj stranica za prijavu i registraciju, integracija OAuth autentifikacije i povezivanje s pozadinskim sustavom	5
Ostalo	Konfiguracija tajnih ključeva (secrets) na developers.facebook.com i console.cloud.google.com za OAuth funkcionalnost	1
Frontend	Izrada obrasca za unos osnovnih informacija o korisniku (visina, težina, dnevna aktivnost, ciljevi). Napomena: trenutno nije prikazano na stranici zbog nedovršenog pozadinskog sustava i problema s preusmjeravanjem	5
Deployment	Postavljanje frontend aplikacije na VPS (Virtual Private Server) poveznica	1
Frontend	Razvoj funkcionalnosti za kreiranje, uređivanje i primjenu postojećih (predefiniranih od strane administratora aplikacije) planova vježbanja	13

2. ciklus

Komponenta	Opis rada	Uloženo sati
Baza podataka	Izrada novog relacijskog dijagrama	2
Backend	Dodavanje upload slika, gif-va, ... funckionalnosti na AWS S3	2
Frontend	Izrada admin dijela-a za upravljanje vježbama, hranom, planovima treninga te mišićnim skupinama u aplikaciji	25
Backend	Mjenajnje JWT payloada kod prijave trener-a, popravljanje CORS errora na backendu, dodavanje 4-5 endpointa	5
Postman	Planiranje endpointa-a za backend	3
Frontend	Dodavanje funkcionalnosti za trener-a, dodavanje opcije da se izabere trener te postane trener	10
Frontend	Izrada hook-ova, svih tip-ova za ts, spajanje frontend-a sa backendom (koliko sam uspio)	15

Ema Adamec

1. ciklus

Komponenta	Opis rada	Uloženo sati
Frontend	Izrada stranice za praćenje prehrane	6
Dokumentacija	Izrada sekvencijskih dijagrama	1
Frontend	Implementacija dodavanja novih i uređivanja postojećih obroka	3

2. ciklus

Komponenta	Opis rada	Uloženo sati
Frontend	Dodavanje funkcionalnosti na stranici za praćenje prehrane	18
Frontend	Povazivanje stranice za praćenje prehrane s backendom	4

Sonja Ćiković

1. ciklus

Komponenta	Opis rada	Uloženo sati
Frontend	Izrada stranice sa prikazom kategorija vježbi(exercises)	3
Frontend	Izrada stranice za prikaz pojedine kategorije sa listom pripadajućih vježbi	8
Frontend	Implementacija pretrage i funkcionalnosti dodavanja novih vježbi	3
Frontend	Dodavanje funkcionalnosti za pregled detalja o vježbama	4

2. ciklus

Komponenta	Opis rada	Uloženo sati
Frontend	Ažurirana funkcionalnost za pregled detalja o vježbi(povijest)	7
Frontend	Ažurirana funkcionalnost za dodavanje vježbi	9
Dokumentacija	Izrada prezentacije	2

Gabrijela Klepec

1. ciklus

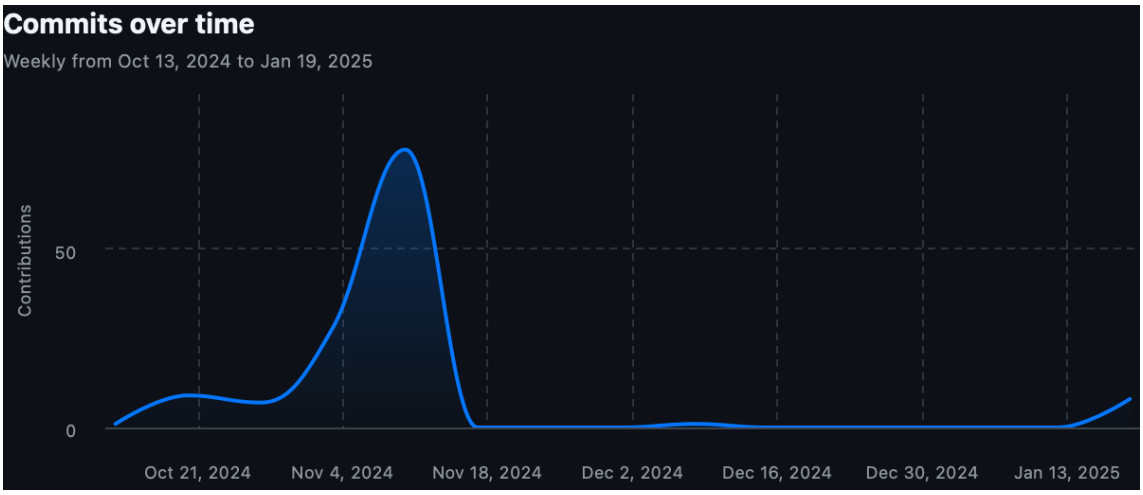
Komponenta	Opis rada	Uloženo sati

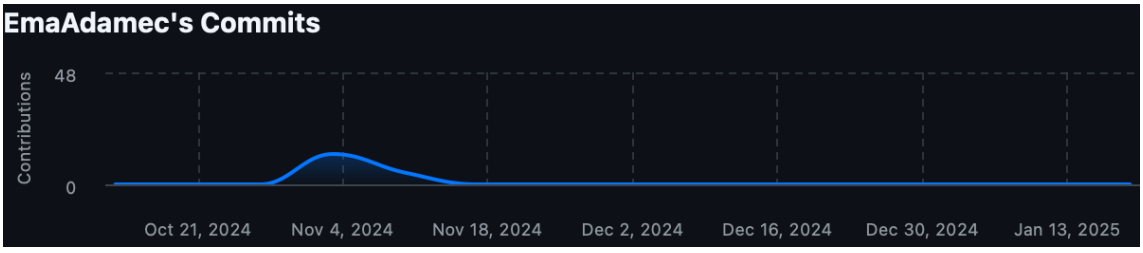
Frontend	Postavljanje stranice profila i stranice za pokretanje treninga	3
Frontend	Implementacija ciljeva (goals) na stranici profila i odgovarajućih grafova s chart.js	6
Frontend	Postavljanje izgleda stranice za workout (prikaz treninga iz rutine i završenih prošlih treninga)	7
Frontend	Izrada stranice workout-now	1

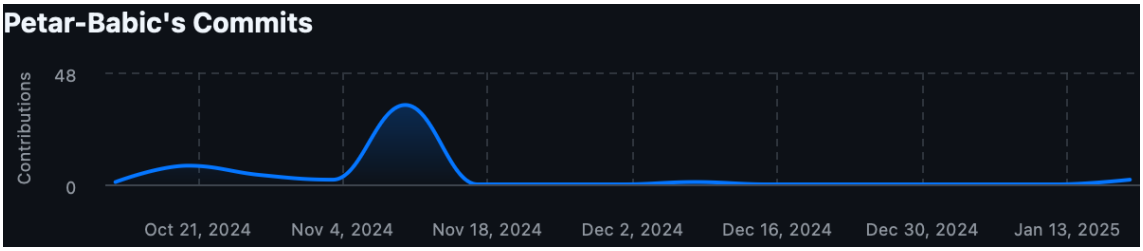
2. ciklus

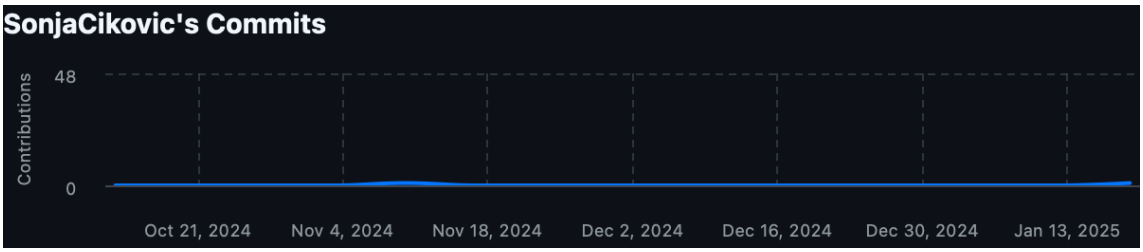
Komponenta	Opis rada	Uloženo sati
Frontend	Popravljanje komponente za pokrenuti trening ExerciseInProgress	3
Frontend	Prikaz odabranog treninga	6

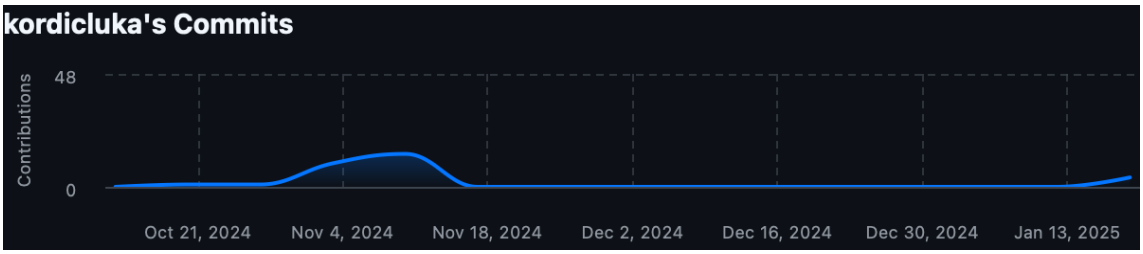
Dijagram pregleda promjena

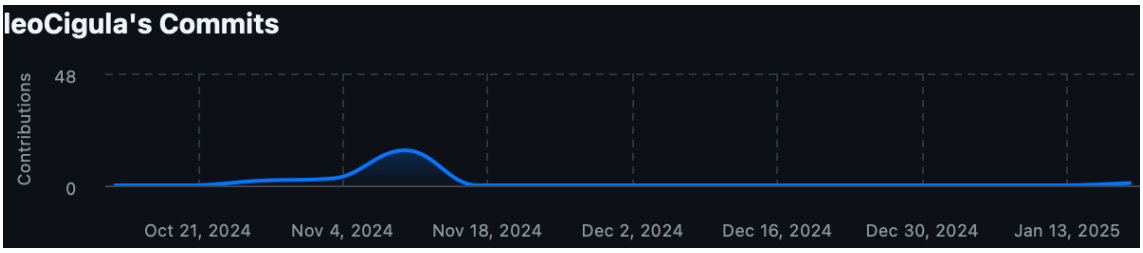


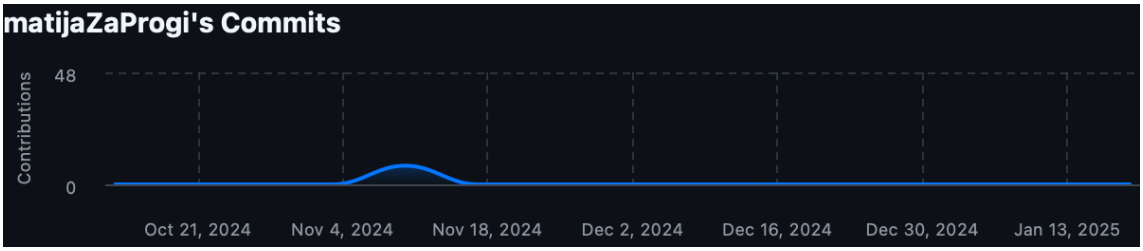


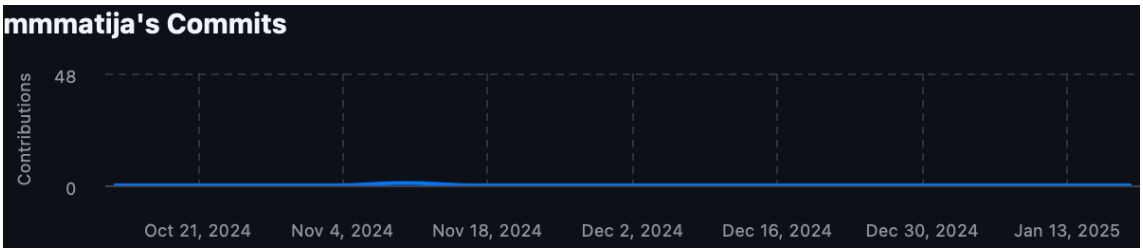

Petar-Babic's Commits
Contributions
48
0
Oct 21, 2024
Nov 4, 2024
Nov 18, 2024
Dec 2, 2024
Dec 16, 2024
Dec 30, 2024
Jan 13, 2025


SonjaCikovic's Commits
Contributions
48
0
Oct 21, 2024
Nov 4, 2024
Nov 18, 2024
Dec 2, 2024
Dec 16, 2024
Dec 30, 2024
Jan 13, 2025


kordicluka's Commits
Contributions
48
0
Oct 21, 2024
Nov 4, 2024
Nov 18, 2024
Dec 2, 2024
Dec 16, 2024
Dec 30, 2024
Jan 13, 2025


leoCigula's Commits
Contributions
48
0
Oct 21, 2024
Nov 4, 2024
Nov 18, 2024
Dec 2, 2024
Dec 16, 2024
Dec 30, 2024
Jan 13, 2025


matijaZaProgi's Commits
Contributions
48
0
Oct 21, 2024
Nov 4, 2024
Nov 18, 2024
Dec 2, 2024
Dec 16, 2024
Dec 30, 2024
Jan 13, 2025


mmmatija's Commits
Contributions
48
0
Oct 21, 2024
Nov 4, 2024
Nov 18, 2024
Dec 2, 2024
Dec 16, 2024
Dec 30, 2024
Jan 13, 2025

Kjučni izazovi i rješenja

Tehnički izazovi:

- Integracija OAuth 2.0 autentifikacije i autorizacije: Ovo je bio jedan od složenijih dijelova implementacije, no uspješno je riješen kroz istraživanje i postavljanje odgovarajuće konfiguracije na frontendu i backendu.
- JWT autentifikacija i autorizacija: Implementacija ovih funkcionalnosti zahtijevala je napredno razumijevanje sigurnosnih protokola te povezivanje s bazom podataka, što je uspješno realizirano.
- Postavljanje servera i deployment: Proces konfiguracije AWS EC2 instance te automatizacije deploya putem GitHub Actionsa bio je tehnički zahtjevan. Ovaj izazov je riješen kroz temeljitu pripremu i primjenu alata poput Nginxa i Certbota za SSL certifikate.
- Komunikacija između frontenda i backenda: Poteškoće su se pojavljivale zbog nesinkroniziranog razvoja, ali redoviti sastanci i razmjena informacija omogućili su rješavanje problema.

Stečena znanja:

- Kroz izradu projekta stečeno je iskustvo u radu s tehnologijama poput Java Spring Boota, Next.js-a, AWS-a, te u korištenju alata za upravljanje projektima kao što su GitHub Actions i Swagger.io.
- Članovi tima usvojili su dublje razumijevanje arhitekture web aplikacija, dizajna baza podataka, te korištenja sigurnosnih protokola za autentifikaciju.
- Poboljšane su vještine timskog rada, planiranja resursa i upravljanja projektima.

Potrebna znanja za brži i kvalitetniji rad:

- Naprednije poznavanje sigurnosnih standarda za autentifikaciju i autorizaciju moglo je ubrzati implementaciju tih funkcionalnosti.
- Više iskustva s deployment alatima i server konfiguracijama (npr. Docker) omogućilo bi brži i jednostavniji postupak postavljanja aplikacije na produkciju.
- Bolje razumijevanje dizajna korisničkog sučelja (UI/UX) doprinijelo bi intuitivnijem korisničkom iskustvu.