Aplikacija za nogometne trenere

Pernar, Domagoj

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:618728

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-20**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1391

APLIKACIJA ZA NOGOMETNE TRENERE

Domagoj Pernar

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1391

APLIKACIJA ZA NOGOMETNE TRENERE

Domagoj Pernar

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1391

Pristupnik:	**Domagoj Pernar (0036543915)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Krešimir Fertalj

Zadatak: **Aplikacija za nogometne trenere**

Opis zadatka:

Cilj završnog rada je izrada programske podrške za planiranje, praćenje i analizu treninga i utakmica koja će povećati efikasnost rada trenerskog osoblja, procjenu napretka pojedinaca i tima kao cjeline te optimizaciju strategije i taktike nogometne igre. Aplikacija treba omogućiti praćenje prisustva igrača na treningu i utakmicama, evidenciju zdravstvenog stanja i ozljeda igrača, planiranje i praćenje treninga i utakmica te primjene taktika i strategija na utakmicama. Implementirati generiranje izvještaja i analiza za evaluaciju timskih i individualnih performansi.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

Uvod

U današnjem dinamičnom svijetu sporta, tehnologija igra ključnu ulogu u unapređenju performansi i optimizaciji trenažnih procesa. Ovaj završni rad bavi se razvojem web aplikacije namijenjene nogometnim trenerima, s ciljem povećanja efikasnosti njihovog rada te poboljšanja performansi tima i pojedinaca.

Trenersko osoblje često nema sustav za praćenje prisustva igrača na treningu i utakmicama, evidenciju zdravstvenog stanja i ozljeda igrača, te planiranje i praćenje treninga i utakmica. Ovaj nedostatak može otežati procjenu napretka pojedinaca i tima, kao i optimizaciju strategije i taktike nogometne igre. Bez centraliziranog sustava za praćenje i analizu, podaci se često gube ili su nedostupni, što dodatno otežava trenerskom osoblju donošenje informiranih odluka.

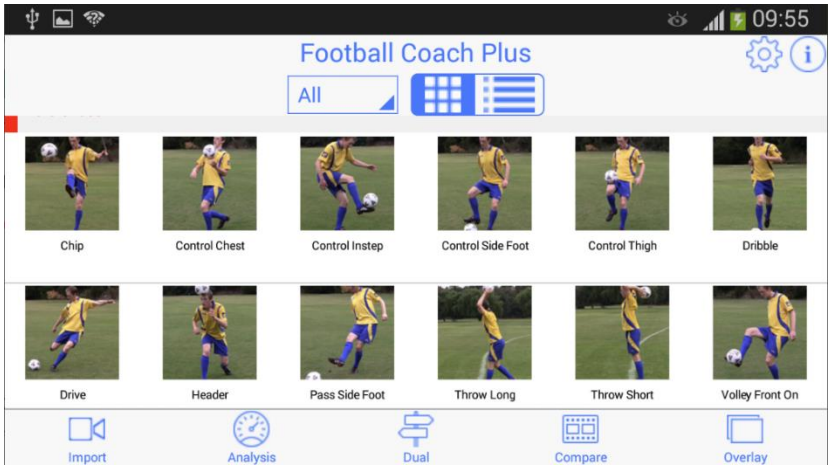
Aplikacija razvijena u sklopu ovog rada omogućit će sveobuhvatno planiranje, praćenje i analizu treninga i utakmica, pružajući trenerima alat za preciznu procjenu napretka svojih igrača. Ključne funkcionalnosti uključuju evidenciju prisustva na treninzima i utakmicama, praćenje zdravstvenog stanja i ozljeda, kao i detaljno planiranje trenažnih i taktičkih aktivnosti. Također, aplikacija će omogućiti vođenje izvještaja i analiza, koje će trenerima pomoći u donošenju informiranih odluka o strategiji i taktici igre.

Kroz integraciju ovih funkcionalnosti, cilj je stvoriti alat koji će unaprijediti sve aspekte trenerskog posla, od administrativnih zadataka do analize sportskih performansi. Centralizirani sustav za praćenje i analizu podataka ne samo da će povećati efikasnost rada trenerskog osoblja, već će i omogućiti bolje planiranje i optimizaciju trenažnih procesa, što je ključno za postizanje vrhunskih rezultata u nogometnoj igri.

1. Analiza reprezentativnih aplikacija

1.1. Football Coach Plus

Football Coach Plus [1] je napredni alat za razvoj performansi nogometaša, osmišljen za trenere kako bi pružili vizualne povratne informacije igračima tijekom treninga i natjecanja. Omogućuje snimanje videa, analizu tehnike u stvarnom vremenu, usporenom snimku i kadar po kadar. Igrači mogu uspoređivati svoje snimke s referentnim videozapisima, što pomaže u prepoznavanju tehničkih problema i praćenju napretka. Unutar aplikacije može se snimati, obrezivati, dijeliti i usporedno prikazivati videozapise, također može se dijeliti datoteke putem iTunesa.

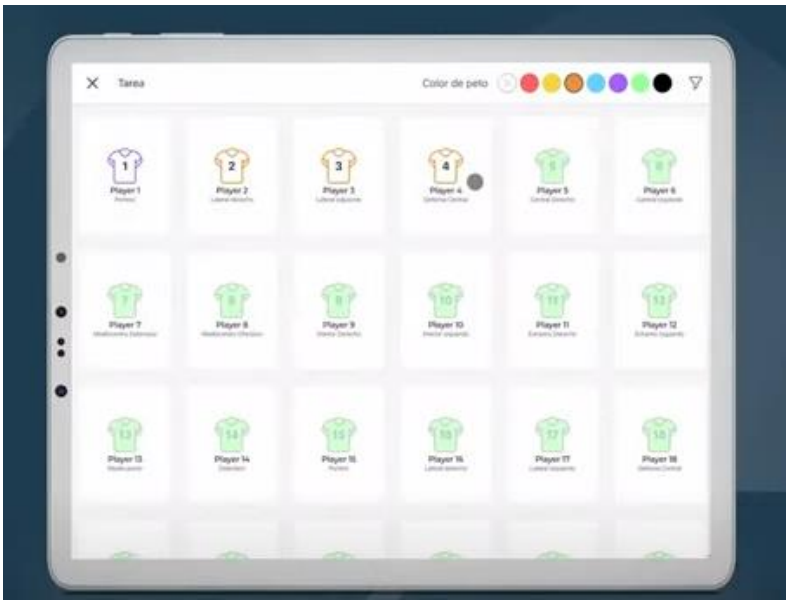


Slika 1.1 Početni ekran aplikacije Football Coach Plus [1]

Slika 1.2 Ekran gledanja videozapisa [1]

1.2. Bcoach

Bcoach [2] je aplikacija namijenjena trenerima nogometa i futsala. Futsal [12] je kratica portugalskih futebol de salão, odnosno španjolskih riječi fútbol sala, fútbol de salón, koje znače dvoranski nogomet. Igra se na tvrdom terenu manjem od nogometnog. "Futsal" je i u Hrvatskoj službeni naziv za dvoranski nogomet koji se igra po pravilima FIFA-e. Omogućuje dizajniranje treninga, detaljno opisivanje taktičkih poteza na virtualnoj ploči i prikupljanje statistike utakmica. Treneri mogu animirati, spremati i dijeliti zadatke, analizirati poteze koristeći različite boje, igrače i terene, te dobiti detaljnu statistiku utakmica za lakšu analizu.



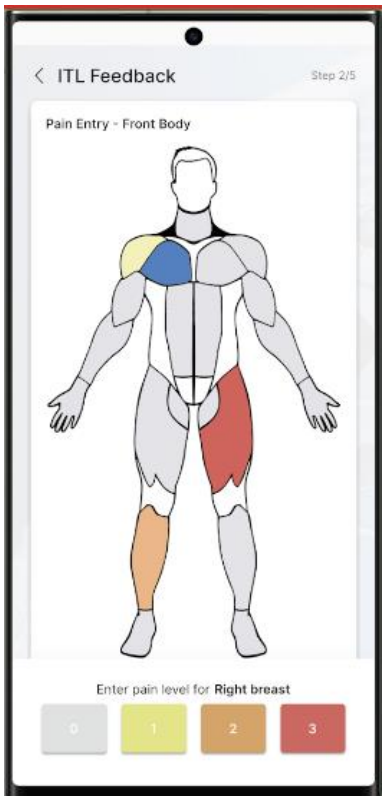
Slika 1.3 Ekran popisa igrača [2]

Slika 1.4 Ekran postavljanja treninga [2]

1.3. Ultrax

Ultrax [3] je inovativna sportska znanstvena poslovna inteligencijska platforma dizajnirana da revolucionira način na koji sportske organizacije djeluju. Obuhvaća sveobuhvatan niz značajki pažljivo osmišljenih kako bi unaprijedili komunikaciju, pojednostavili upravljanje podacima, olakšali napore *skautinga* te produbili razumijevanje u svim sportovima i klubovima, bez obzira na njihovu razinu. Cilj platforme Ultrax je isporučiti vrhunsku, troškovno učinkovitu sportsku analitiku putem besprijekorno integriranog hardvera i softverske usluge temeljene na oblaku.

Ultrax aplikacija pomaže klubovima poput HNK Rijeka, ŽNK Dinamo Zagreb i NK Lokomotiva Zagreb u praćenju napora igrača tako što igrači nakon treninga odnosno utakmica šalju izvještaje o svome zdravstvenom stanju. Treneri tako dolaze do informacija direktno od svojih igrača što mogu iskoristiti za prilagođavanje napora na treninzima i doziranjem igrača na utakmicama.



Slika 1.5 Ekran povratne informacije igrača o boli [3]

Slika 1.6 Ekran postavljanja taktike [3]

2. Specifikacija zahtjeva

2.1. Funkcionalni zahtjevi

- Omogućiti registraciju i prijavu korisnika.
- Omogućiti pregled i uređivanje timova prijavljenog korisnika.
- Omogućiti unos igrača i dodavanje u određene timove.
- Omogućiti vođenje evidencije o prisutnosti igrača na treningu i utakmicama.
- Pružiti mogućnost vođenja evidencije o zdravstvenom stanju i ozljedama igrača.
- Omogućiti unos naziva i opisa taktike pojedinog tima.
- Omogućiti planiranje treninga s detaljnim opisima i rasporedima.
- Omogućiti pregled popisa utakmica s primjenom taktike i pregledom strijelaca i asistenata.
- Omogućiti zapisivanje izvještaja o timskim i individualnim performansama radi procjene napretka i optimizacije strategija.

2.2. Nefunkcionalni zahtjevi

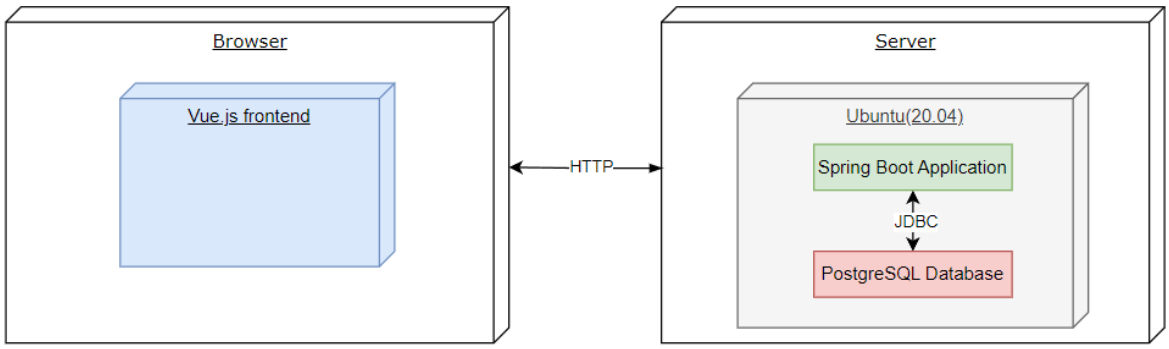
- Sustav treba biti brz i responzivan.
- Aplikacija treba biti napisana korištenjem programskog okvira Spring u jeziku Java.
- Aplikacija će biti izgrađena za Web.
- Aplikacija treba biti optimizirana kako bi pružala brze odazive i minimalno vrijeme učitavanja.
- Aplikacija treba imati intuitivno korisničko sučelje i jednostavnu navigaciju.

- Osigurati visoku dostupnost sustava i pouzdanost u radu kako bi se spriječilo gubitak podataka.
- Sustav treba biti skalabilan kako bi mogao rasti s brojem korisnika i količinom podataka.
- Osigurati sigurnost podataka kroz mehanizme kao što su enkripcija lozinki i ograničavanje pristupa podacima.

3. Arhitektura sustava

3.1. Opis arhitekture sustava

U završnom radu sustav koristi klijent-poslužitelj arhitekturu gdje je arhitektura poslužitelja implementirana u Javi pomoću radnog okvira Spring, a arhitektura korisničkog sučelja u Vue.js. Baza podataka PostgreSQL pohranjuje podatke o tablicama. Klijentska aplikacija, izgrađena u Vue.js, komunicira s poslužiteljem putem HTTP protokola koristeći metode GET, POST, PUT i DELETE za dohvaćanje, kreiranje, ažuriranje i brisanje podataka. Poslužitelj obrađuje ove zahtjeve, izvršava potrebne operacije na bazi podataka i vraća odgovore klijentskoj aplikaciji, omogućujući glatku i efikasnu interakciju korisnika s aplikacijom.



Slika 3.1 Dijagram ugradnje rješenja

3.2. Baza podataka

Baza podataka sadrži 11 međusobno povezanih tablica. Tablice su povezane putem različitih tipova ključeva kako bi omogućile integrirano upravljanje podacima. Glavne tablice uključuju users, teams, players, matches, training, attendance, healthStatus, tactics, tacticsApplication, reports, i playerMatchStats.

Tablica users sadrži osnovne informacije o korisnicima, uključujući trenersko osoblje, a povezana je s tablicom reports preko authorId. Tablica teams definira timove i povezana je s tablicama players, matches, i training preko teamId. Tablica players pohranjuje podatke o igračima, uključujući njihove zdravstvene statuse u tablici healthStatus, prisustvo na treninzima i utakmicama u tablici attendance, te statistike mečeva u tablici playerMatchStats.

Tablica matches prati pojedine utakmice, uključujući podatke o protivnicima, lokacijama i rezultatima. Trening aktivnosti su pohranjene u tablici training, dok tablica tactics sadrži različite taktičke pristupe koji se primjenjuju na utakmicama, evidentirano u tablici tacticsApplication.

Korišten je PostgreSQL [9] kao sustav za upravljanje bazama podataka te Hibernate JPA iz Spring okvira [5] za modeliranje ovih podataka, što omogućuje jednostavnu i efikasnu manipulaciju podacima unutar Java aplikacije.

Slika 3.2 Prikaz tablica baze podataka

Tablica 1 Users

Naziv atributa	Tip atributa	Opis atributa
userId (PK)	LONG	Jedinstveni identifikator korisnika
username	VARCHAR(255)	Korisničko ime
firstName	VARCHAR(255)	Ime korisnika
lastName	VARCHAR(255)	Prezime korisnika
email (U) (NN)	VARCHAR(255)	Email adresa
Lozinka (NN)	VARCHAR(255)	Lozinka korisnika
role	VARCHAR(255)	Uloga korisnika

Tablica 2 Reports

Naziv atributa	Tip atributa	Opis atributa
reportId (PK)	LONG	Jedinstveni identifikator izvještaja
date (NN)	DATE	Datum izvještaja
description	VARCHAR(255)	Opis izvještaja
authorId (FK)	LONG	Jedinstveni identifikator autora izvještaja

Tablica 3 Teams

Naziv atributa	Tip atributa	Opis atributa
teamId (PK)	LONG	Jedinstveni identifikator tima
teamName (NN)	VARCHAR(255)	Naziv tima
season	VARCHAR(255)	Sezona u kojoj tim igra
coachId (FK)	LONG	Jedinstveni identifikator trenera tima

Tablica 4 Players

Naziv atributa	Tip atributa	Opis atributa
playerId (PK)	LONG	Jedinstveni identifikator igrača
firstName (NN)	VARCHAR(255)	Ime igrača
lastName (NN)	VARCHAR(255)	Prezime igrača
dateOfBirth	DATE	Datum rođenja
position	VARCHAR(255)	Pozicija igrača
height	INT	Visina igrača
weight	INT	Težina igrača
teamId (FK)	LONG	Jedinstveni identifikator tima kojem igrač pripada

Tablica 5 Matches

Naziv atributa	Tip atributa	Opis atributa
matchId (PK)	LONG	Jedinstveni identifikator utakmice
date (NN)	DATE	Datum utakmice
opponent	VARCHAR(255)	Protivnički tim
location	VARCHAR(255)	Lokacija utakmice
result	VARCHAR(255)	Rezultat utakmice
teamId (FK)	LONG	Jedinstveni identifikator tima kojem utakmica pripada

Tablica 6 Training

Naziv atributa	Tip atributa	Opis atributa
trainingId (PK)	LONG	Jedinstveni identifikator treninga
date (NN)	DATE	Datum treninga
type	VARCHAR(255)	Tip treninga
description	VARCHAR(255)	Opis treninga
teamId (FK)	LONG	Jedinstveni identifikator tima kojem trening pripada

Tablica 7 Attendance

Naziv atributa	Tip atributa	Opis atributa
attendanceId (PK)	LONG	Jedinstveni identifikator prisustva
date (NN)	DATE	Datum prisustva
present	VARCHAR(255)	Označava prisutnost igrača
trainingOrMatch	VARCHAR(255)	Označava je li prisustvo na treningu ili utakmici
playerId (FK)	LONG	Jedinstveni identifikator igrača kojem prisustvo pripada

Tablica 8 HealthStatus

Naziv atributa	Tip atributa	Opis atributa
healthStatusId (PK)	LONG	Jedinstveni identifikator zdravstvenog stanja
date (NN)	VARCHAR(255)	Datum zapisa
statusNote	VARCHAR(255)	Bilješka o zdravstvenom stanju
playerId (FK)	LONG	Jedinstveni identifikator igrača kojem status pripada

Tablica 9 PlayerMatchStats

Naziv atributa	Tip atributa	Opis atributa
playerMatchStatId (PK)	LONG	Jedinstveni identifikator prisustva
goals	INT	Broj postignutih golova
assists	INT	Broj asistencija
matchId (FK)	LONG	Jedinstveni identifikator utakmice
playerId (FK)	LONG	Jedinstveni identifikator igrača

Tablica 10 Tactics

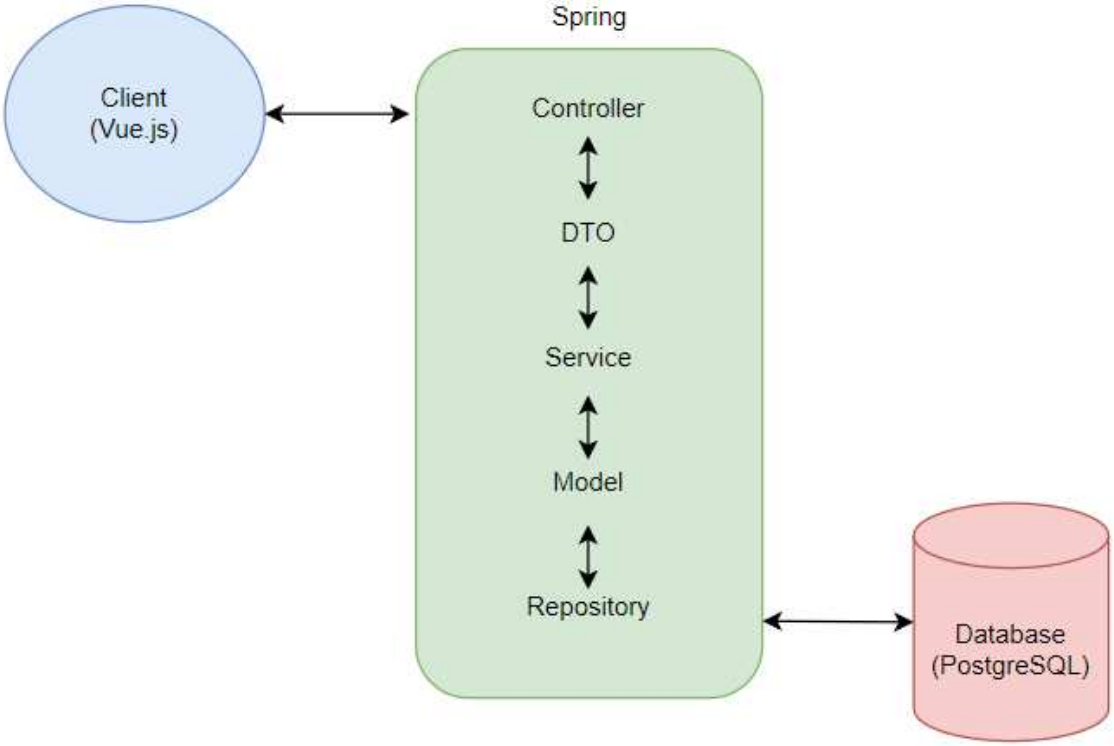
Naziv atributa	Tip atributa	Opis atributa
tacticId (PK)	INT	Jedinstveni identifikator taktike
name (NN)	VARCHAR(255)	Naziv taktike
description	VARCHAR(255)	Opis taktike
teamId (FK)	INT	Jedinstveni identifikator tima kojem taktika pripada

Tablica 11 TacticsApplication

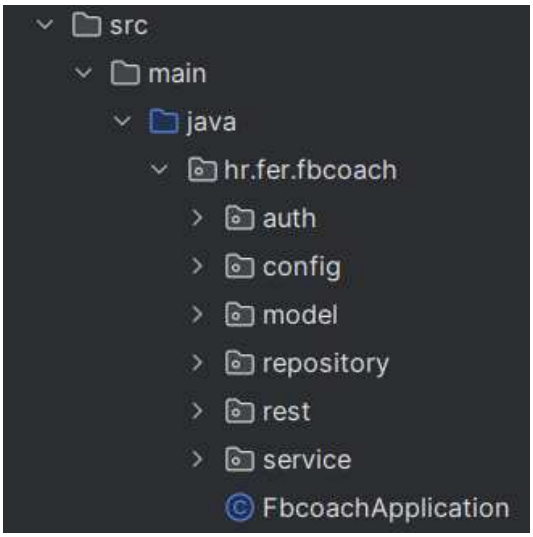
Naziv atributa	**Tip atributa**	**Opis atributa**
tacticsApplicationId (PK)	LONG	Jedinstveni identifikator primjene taktike
description	VARCHAR(255)	Opis primjene taktike
matchId (FK)	LONG	Jedinstveni identifikator utakmice kojoj primjena taktika pripada
tacticId (FK)	LONG	Jedinstveni identifikator tima kojem primjena taktika pripada

4. Arhitektura poslužitelja

Aplikacija koristi monolitnu arhitekturu, što znači da su sve komponente integrirane u jednu jedinstvenu aplikaciju. Unutar ove arhitekture, aplikacija koristi kontroler (upravljač) za upravljanje HTTP zahtjevima (GET, POST, PUT, DELETE), servis za poslovnu logiku, repozitorij za interakciju s bazom podataka, DTO (Data Transfer Object) za prijenos podataka između slojeva aplikacije, te model za reprezentaciju entiteta baze podataka. Kontroleri obrađuju zahtjeve klijenata, servisi izvršavaju poslovne operacije, dok repozitoriji komuniciraju s PostgreSQL bazom koristeći JPA i Hibernate.



Slika 4.1 Arhitektura monolitne aplikacije



Slika 4.2 Prikaz arhitekture poslužitelja

U projektu koristi se organizirana arhitektura paketa koja omogućava čitljivost i održivost koda. Projekt je organiziran unutar direktorija src/main/java/hr/fer/fbcoach, koji je podijeljen na nekoliko ključnih paketa:

1. auth - Paket za autentifikaciju, koji sadrži klase i konfiguracije povezane s prijavom korisnika, autorizacijom i sigurnošću.

2. config - Paket za konfiguracijske klase koje su potrebne za ispravan rad aplikacije.

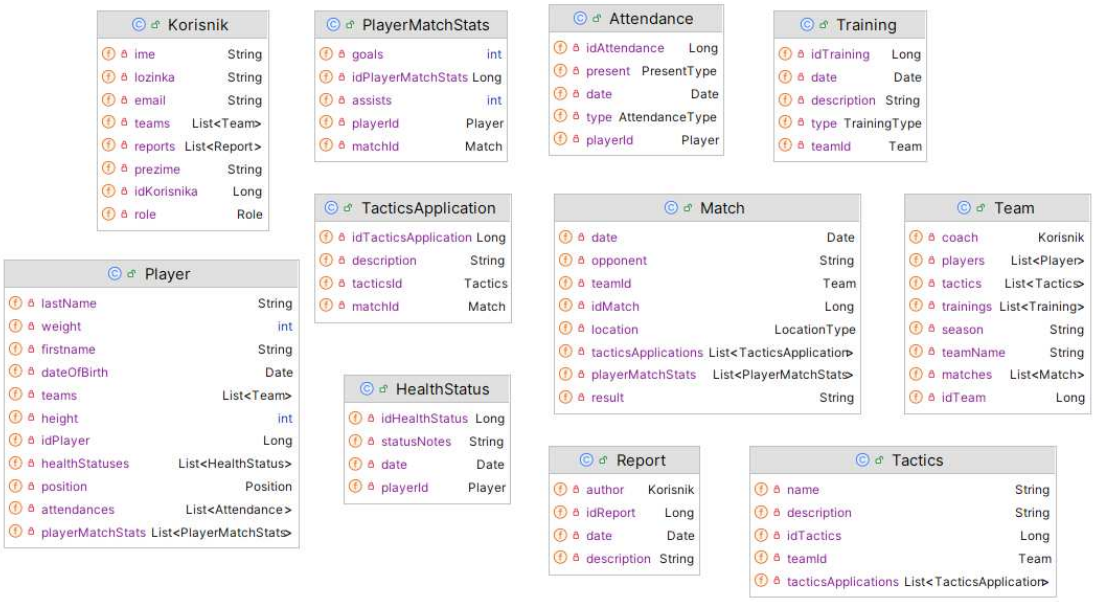
3. model - Paket za modele podataka. Ovdje su definirane entitet klase koje predstavljaju strukture podataka pohranjenih u bazi podataka, često koristeći JPA anotacije.

4. repository - Paket za repozitorije. Ovdje se nalaze sučelja koja pružaju metode za interakciju s bazom podataka. Ovi repozitoriji koriste Spring Data JPA kako bi omogućili jednostavne operacije nad bazom podataka.

5. rest - Paket za REST kontrolere. Ovdje su smještene klase koje definiraju krajnje točke API-ja (endpoints) i upravljaju HTTP zahtjevima (GET, POST, PUT, DELETE).

6. service - Paket za servisne klase. Ovaj sloj sadrži poslovnu logiku aplikacije. Servisi komuniciraju s repozitorijima kako bi izvršavali operacije nad podacima i pružali potrebne informacije kontrolerima.

4.1. Model



Slika 4.3 Dijagram klasa modela

U radnom okviru Spring, modeli predstavljaju entitete koji se koriste za mapiranje podataka između aplikacije i baze podataka. Ovi entiteti koriste JPA (Java Persistence API) anotacije kako bi definirali strukturu tablica u bazi podataka. Primjera modela Training iz projekta:

Slika 4.4 Prikaz klase Training

JPA (Java Persistence API) anotacije su specifične anotacije koje se koriste u Java programiranju za mapiranje Java objekata na relacijske tablice u bazi podataka. Ove anotacije olakšavaju rad s bazama podataka unutar Java aplikacija bez potrebe za pisanjem SQL koda.

1. Anotacije:
    - @Entity: Označava da je klasa JPA entitet.
    - @Table(name = "training"): Definira naziv tablice u bazi podataka na koju se entitet mapira.
    - @Id: Označava primarni ključ entiteta.
    - @GeneratedValue(strategy = GenerationType.IDENTITY): Definira strategiju generiranja vrijednosti za primarni ključ.
    - @Column(name = "id_training"): Mapira atribut na kolonu u bazi podataka.
    - @ManyToOne: Označava mnogostruku vezu prema entitetu Team.

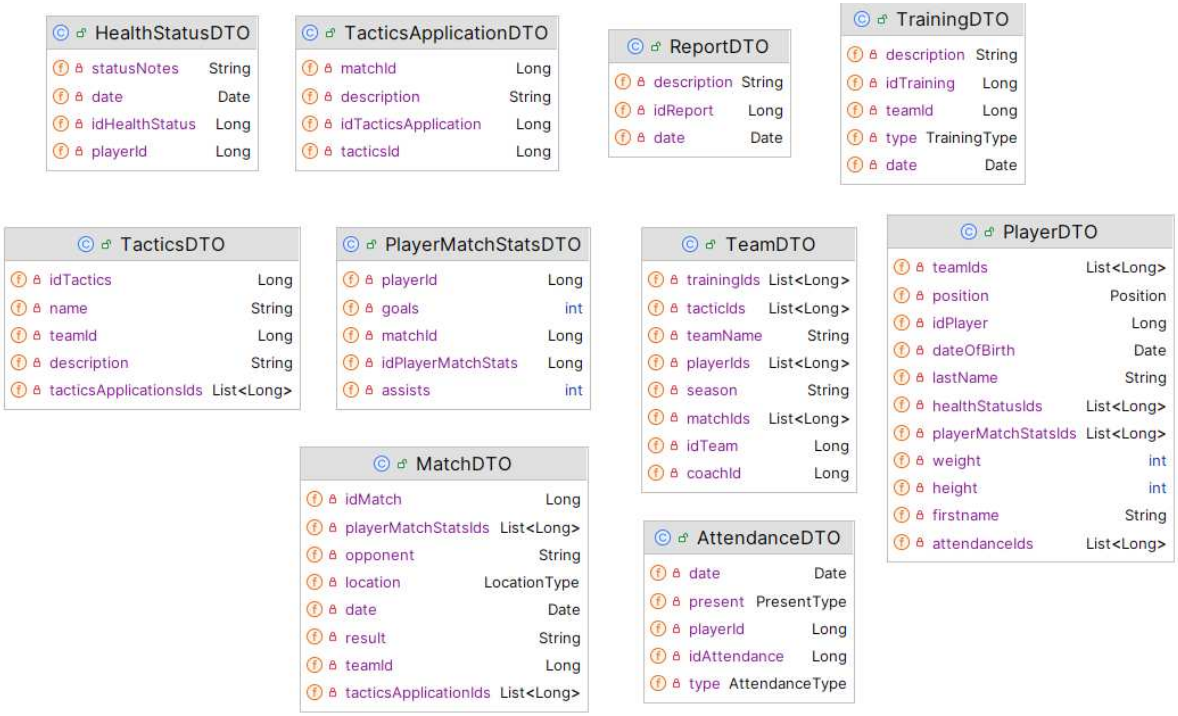
- @JoinColumn(name = "id_team"): Definira naziv strane kolone koja povezuje ovaj entitet s Team entitetom.
- @JsonBackReference: Koristi se za rješavanje problema cikličke zavisnosti u JSON serijalizaciji

2. Lombok Anotacije:
    - @Data: Generira *gettere*, *settere*, *toString*, *equals*, i *hashCode* metode.
    - @Builder: Omogućava graditeljski obrazac za kreiranje instanci klase.
    - @NoArgsConstructor: Generira konstruktor bez argumenata.
    - @AllArgsConstructor: Generira konstruktor sa svim argumentima.

3. Polja:
    - idTraining: Primarni ključ za entitet.
    - date: Datum treninga.
    - description: Opis treninga.
    - type: Tip treninga (koristi TrainingType).
    - teamId: Referenca na tim kojem trening pripada.

Ovaj model definira strukturu podataka za treninge, uključujući atribute i njihove veze s drugim entitetima, omogućujući efikasno mapiranje između objekata u aplikaciji i zapisa u bazi podataka.

4.2. DTO



Slika 4.5 Dijagram klasa DTO

U radnom okviru Spring, DTO (Data Transfer Object) je objekt koji se koristi za prijenos podataka između različitih slojeva aplikacije, posebno između klijenta i servera. DTO-ovi se često koriste za enkapsulaciju podataka koje treba prenijeti, čime se smanjuje broj poziva metode i omogućava fleksibilnije mapiranje podataka. Primjer DTO iz projekta:

```
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class TrainingDTO {

    private Long idTraining;
    private Date date;
    private String description;
    private TrainingType type;
    private Long teamId;
}
```

Slika 4.6 Prikaz DTO Training

1. Definicija DTO-a:
    - TrainingDTO je jednostavna Java klasa koja sadrži atribute potrebne za prijenos podataka o treningu između slojeva aplikacije.

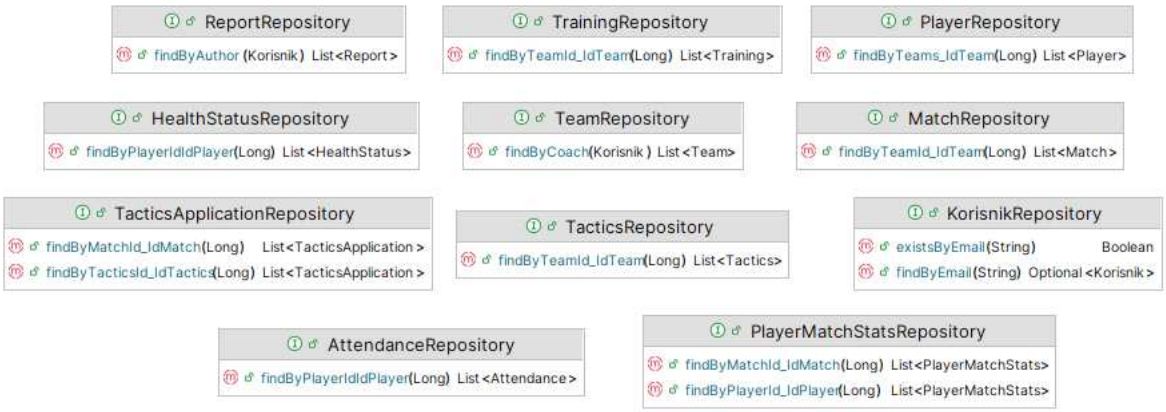
2. Lombok Anotacije:
    - @Data: Generira *gettere*, *settere*, *toString*, *equals*, i *hashCode* metode.
    - @Builder: Omogućava graditeljski obrazac za kreiranje instanci klase.
    - @AllArgsConstructor: Generira konstruktor sa svim argumentima.
    - @NoArgsConstructor: Generira konstruktor bez argumenata.

3. Polja:
    - idTraining: Identifikator treninga.
    - date: Datum treninga.
    - description: Opis treninga.
    - type: Tip treninga, koristi TrainingType enum.
    - teamId: Identifikator tima kojem trening pripada.

Korištenje DTO-a (Data Transfer Object) donosi brojne prednosti u razvoju aplikacija, ponajviše kroz separaciju podataka. DTO omogućava razdvajanje entitetskih modela od podataka koji se prenose između različitih slojeva aplikacije. Time se smanjuje zatezanje između baza podataka i klijentskog sučelja, čime se omogućuje efikasniji i lakši razvoj aplikacije. Fleksibilnost koju pruža DTO ogleda se u mogućnosti da sadrži samo ona polja koja su potrebna za određeni zahtjev ili odgovor. Na taj način smanjuje se količina prenesenih podataka, što direktno doprinosi poboljšanju performansi aplikacije.

Još jedna važna prednost korištenja DTO-ova je sigurnost. Korištenjem DTO-ova izbjegava se izlaganje unutarnjih entitetskih modela direktno klijentima, što može značajno povećati sigurnost aplikacije. Ovaj pristup sprječava potencijalne zloupotrebe i neželjene pristupe podacima. Sve ove prednosti čine DTO-ove izuzetno korisnima za pojednostavljenje procesa prijenosa podataka, poboljšanje performansi aplikacije i povećanje sigurnosti, što u konačnici olakšava razvoj i održavanje aplikacije.

4.3. Repozitorij



Slika 4.7 Dijagram sučelja Repository

U radnom okviru Spring, repozitorij (repository) je sloj koji upravlja trajnošću podataka, omogućujući jednostavne CRUD (Create, Read, Update, Delete) operacije nad bazom podataka. Repozitoriji koriste JPA (Java Persistence API) za interakciju s bazom podataka i omogućuju transparentno mapiranje između Java objekata i relacijskih tablica. Primjer repozitorija iz projekta:

```
public interface TrainingRepository extends JpaRepository<Training,Long> {
    1 usage  new *
    List<Training> findByTeamId_IdTeam(Long teamId);
}
```

Slika 4.8 Prikaz sučelja TrainingRepository

1. Proširivanje JpaRepository:
    - TrainingRepository proširuje JpaRepository, pružajući osnovne metode za rad s entitetom Training, kao što su save, findById, findAll, deleteById, i mnoge druge.
    - JpaRepository<Training, Long>: Prvi parametar je tip entiteta s kojim repozitorij radi, dok je drugi parametar tip primarnog ključa tog entiteta.

2. Prilagođene metode upita:
    - List<Training> findByTeamId_IdTeam(Long teamId): Ova metoda koristi Spring Data JPA sposobnost generiranja upita temeljenih na imenovanju metoda. Metoda pronalazi sve treninge koji pripadaju timu s danim teamId.

3. Anotacije i jednostavnost:
    - Nema potrebe za dodatnim anotacijama jer JpaRepository već pruža potrebne CRUD operacije.
    - Ova metoda omogućava jednostavno i čitljivo dohvaćanje podataka bez potrebe za pisanjem složenih SQL upita.

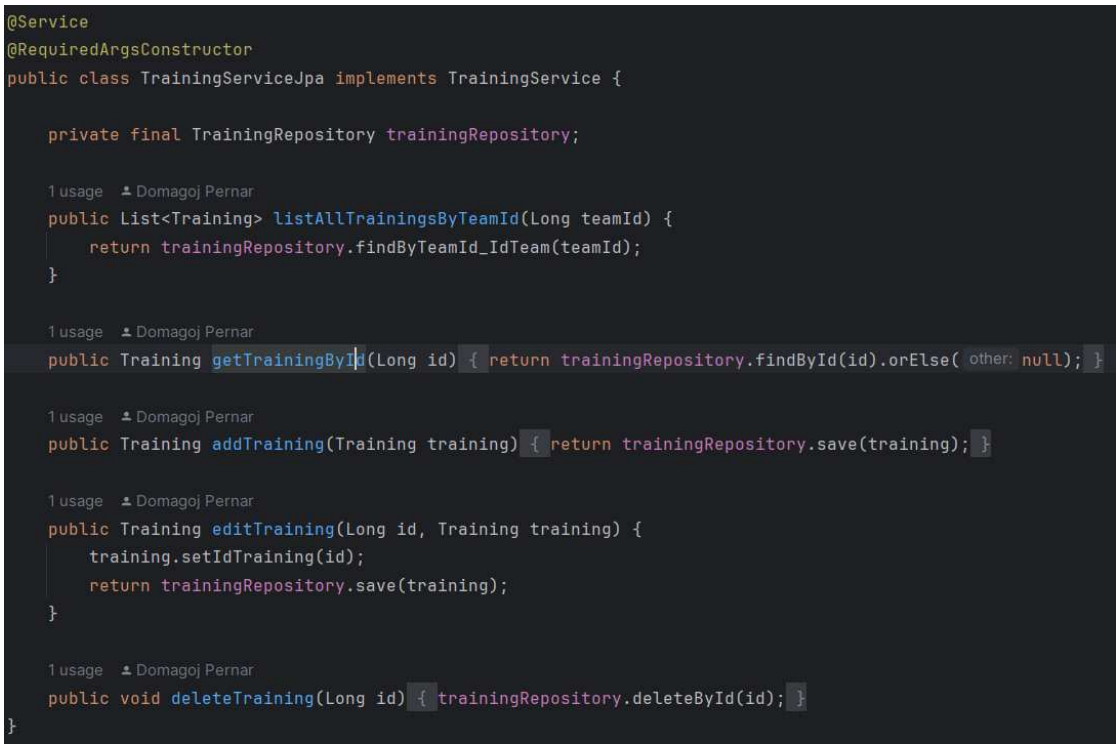
Spring Data JPA automatizira mnoge uobičajene zadatke u radu s bazama podataka, omogućujući programerima da se fokusiraju na poslovnu logiku aplikacije umjesto na niske razine operacija pristupa podacima.

4.4. Servis

U radnom okviru Spring, servisni sloj (service layer) odgovoran je za implementaciju poslovne logike aplikacije. Servisi posreduju između kontrolera (koji primaju HTTP zahtjeve) i repozitorija (koji upravljaju pristupom bazi podataka), osiguravajući čisto odvajanje odgovornosti i bolju organizaciju koda. Primjer sučelja servisa iz projekta i njegove implementacije:

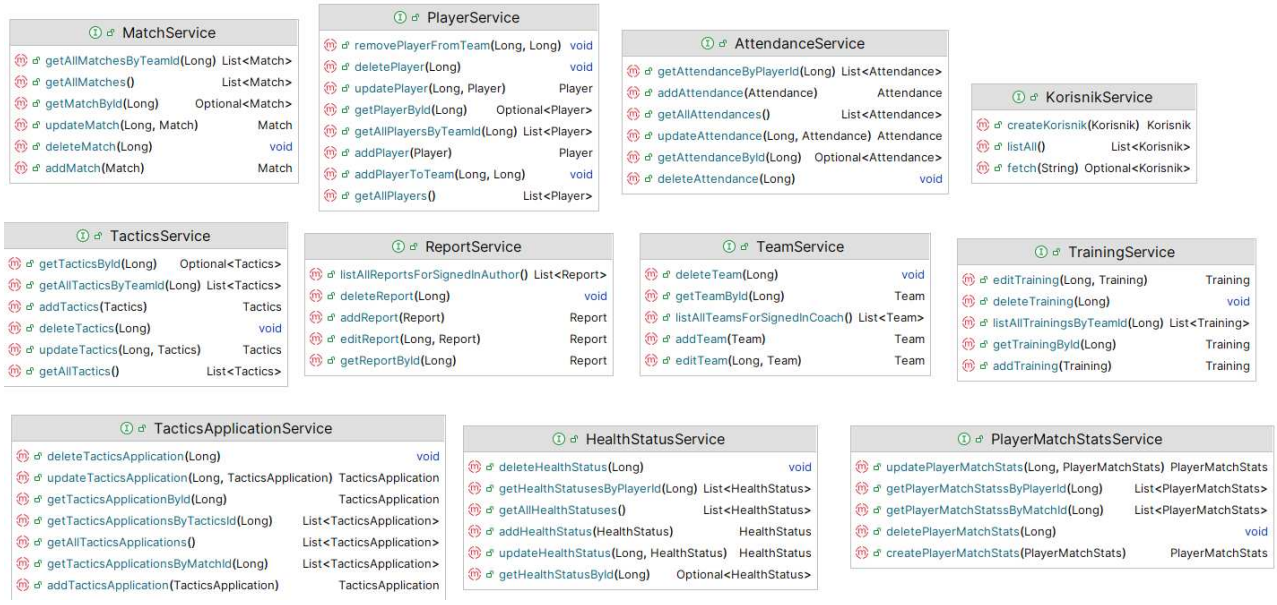
```
public interface TrainingService {
    1 usage  1 implementation  Domagoj Pernar
    List<Training> listAllTrainingsByTeamId(Long teamId);

    1 usage  1 implementation  Domagoj Pernar
    Training addTraining(Training training);

    1 usage  1 implementation  Domagoj Pernar
    Training editTraining(Long id, Training training);

    1 usage  1 implementation  Domagoj Pernar
    void deleteTraining(Long id);

    1 usage  1 implementation  Domagoj Pernar
    Training getTrainingById(Long id);
}
```

Slika 4.9 Prikaz sučelja TrainingService



Slika 4.10 Prikaz klase TrainingServiceJpa

Sučelje TrainingService deklarira metode koje svaki servis mora implementirati, omogućujući tako jasnu strukturu i pravila za upravljanje treninzima unutar aplikacije. Ovo sučelje uključuje operacije poput dohvaćanja treninga po ID-u tima, dodavanja novog treninga, uređivanja postojećeg treninga, brisanja treninga te dohvaćanja treninga po ID-u. Ove metode osiguravaju standardizirani pristup podacima o treninzima i njihovu manipulaciju.



Slika 4.11 Dijagram sučelja Service

Implementacija TrainingServiceJpa je označena s @Service, što znači da je ova klasa Springov servisni *bean*. Spring *beanovi* su objekti koje upravlja Spring IoC (Inversion of Control) kontejner. Oni predstavljaju osnovne gradivne blokove Spring aplikacije. *Beanovi* se definiraju u Spring konfiguraciji koristeći anotacije ili XML konfiguracijske datoteke. Spring kontejner upravlja životnim ciklusom *beanova*, uključujući njihovo instanciranje, inicijalizaciju i uništavanje. *Beanovi* se mogu automatski injektirati u druge *beanove*, omogućujući jednostavno povezivanje i upravljanje ovisnostima unutar aplikacije.

Slika 4.12 Dijagram klasa ServiceJpa

Nadalje, koristi @RequiredArgsConstructor iz Lomboka za automatsko generiranje konstruktora koji injektira TrainingRepository. Ova implementacija sadrži sve metode deklarirane u TrainingService sučelju, koristeći TrainingRepository za interakciju s bazom podataka. Metode poput listAllTrainingsByTeamId, getTrainingById, addTraining, editTraining i deleteTraining sadrže poslovnu logiku i upravljaju CRUD operacijama nad entitetom Training. Ovakav pristup omogućuje efikasno upravljanje treninzima te osigurava dosljednost i integritet podataka u aplikaciji.

4.5. Kontroler (Upravljač)

U radnom okviru Spring, kontroler (controller) je sloj koji upravlja dolaznim HTTP zahtjevima, obrađuje ih i vraća odgovarajuće odgovore klijentu. Kontroleri koriste anotacije za mapiranje HTTP zahtjeva na metode koje obrađuju te zahtjeve. Primjer kontrolera iz projekta:

```
@RestController
@RequestMapping("/training")
@RequiredArgsConstructor
@CrossOrigin(origins = "http://localhost:5173")
public class TrainingController {

    private final TrainingService trainingService;
    private final ModelMapper modelMapper;
```

Slika 4.13 Prikaz klase TrainingController

```
no usages  Domagoj Pernar
@GetMapping("/{id}")
public ResponseEntity<TrainingDTO> getTrainingById(@PathVariable Long id) {...}

no usages  Domagoj Pernar
@PostMapping("/add")
public ResponseEntity<TrainingDTO> addTraining(@RequestBody TrainingDTO trainingDTO) {
    Training training = convertToEntity(trainingDTO);
    Training createdTraining = trainingService.addTraining(training);
    return ResponseEntity.status(HttpStatus.CREATED).body(convertToDto(createdTraining));
}

no usages  Domagoj Pernar
@PutMapping("/edit/{id}")
public ResponseEntity<TrainingDTO> editTraining(@PathVariable Long id, @RequestBody TrainingDTO trainingDTO) {
    Training training = convertToEntity(trainingDTO);
    Training updatedTraining = trainingService.editTraining(id, training);
    return ResponseEntity.ok(convertToDto(updatedTraining));
}

no usages  Domagoj Pernar
@DeleteMapping("/delete/{id}")
public ResponseEntity<Void> deleteTraining(@PathVariable Long id) {
    trainingService.deleteTraining(id);
    return ResponseEntity.noContent().build();
}
```

Slika 4.14 Prikaz krajnjih točaka u klasi TrainingController

```
4 usages  Domagoj Pernar *
private TrainingDTO convertToDto(Training training) {
    TrainingDTO trainingDTO = new TrainingDTO();
    trainingDTO.setIdTraining(training.getIdTraining());
    trainingDTO.setDate(training.getDate());
    trainingDTO.setDescription(training.getDescription());
    trainingDTO.setType(training.getType());

    if (training.getTeamId() != null) {
        trainingDTO.setTeamId(training.getTeamId().getIdTeam());
    }

    return trainingDTO;
}

2 usages  Domagoj Pernar *
private Training convertToEntity(TrainingDTO trainingDTO) {
    return modelMapper.map(trainingDTO, Training.class);
}
```

Slika 4.15 Pretvaranje objekata u klasi TrainingController

Anotacije:

- @RestController: Označava da je klasa REST kontroler koji će obraditi HTTP zahtjeve i vratiti odgovore u JSON formatu.
- @RequestMapping("/training"): Definira osnovni URL put za sve metode u kontroleru.
- @RequiredArgsConstructor: Anotacija iz Lombok biblioteke koja generira konstruktor s potrebnim ovisnostima.
- @CrossOrigin(origins = "http://localhost:5173"): Dozvoljava CORS zahtjeve s određenog izvora, u ovom slučaju *front end* aplikacije koja radi na lokalnom *hostu*.

Metode za obradu HTTP zahtjeva:

- @GetMapping(""): Obrađuje GET zahtjev za dobivanje svih treninga za određeni tim.
- @GetMapping("/{id}"): Obrađuje GET zahtjev za dobivanje treninga prema ID-u.
- @PostMapping("/add"): Obrađuje POST zahtjev za dodavanje novog treninga.
- @PutMapping("/edit/{id}"): Obrađuje PUT zahtjev za ažuriranje postojećeg treninga prema ID-u.
- @DeleteMapping("/delete/{id}"): Obrađuje DELETE zahtjev za brisanje treninga prema ID-u.

ResponseEntity:

- Koristi se za vraćanje HTTP odgovora s odgovarajućim statusnim kodom.

DTO Konverzija:

- convertToDto(Training training): Metoda koja konvertira entitet Training u DTO TrainingDTO.
- convertToEntity(TrainingDTO trainingDTO): Metoda koja konvertira DTO TrainingDTO u entitet Training.

Prednosti korištenja kontrolera u Springu su višestruke i značajno doprinose strukturi i održivosti aplikacije. Prva i najvažnija prednost je Separation of Concerns (odvajanje odgovornosti). Kontroleri u Springu odvajaju logiku obrade zahtjeva od poslovne logike, koja je smještena u servisima, te od upravljanja podacima, koje obavljaju repozitoriji. Ovaj pristup omogućuje jasnu podjelu zadataka unutar aplikacije, čime se povećava modularnost i olakšava buduće promjene i proširenja koda.

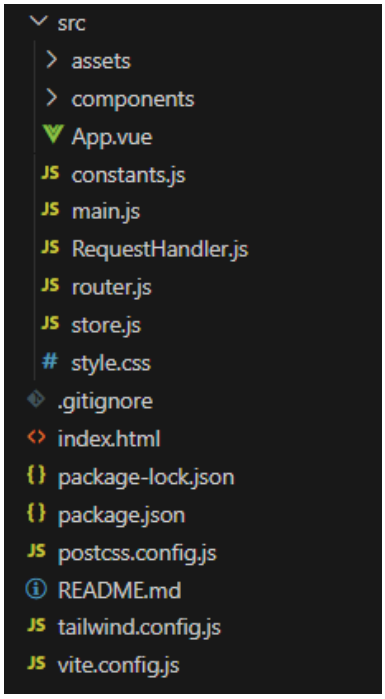
Druga prednost je poboljšana čitljivost i održivost koda. Korištenje kontrolera čini kod čitljivijim i lakšim za održavanje jer jasno definira koji dio koda upravlja HTTP zahtjevima, a koji dio se bavi poslovnom logikom i upravljanjem podacima. Ovaj jasni razgraničenje odgovornosti olakšava programerima razumijevanje strukture aplikacije i pojednostavljuje proces ispravljanja pogrešaka i dodavanja novih značajki.

Osim toga, Spring pruža standardizirani način za definiranje i obradu različitih tipova HTTP zahtjeva, što olakšava razvoj RESTful API-ja. Standardizacija smanjuje mogućnost grešaka i omogućuje dosljednu implementaciju API-ja, što je posebno korisno u timskom okruženju gdje više programera radi na istoj aplikaciji.

Kao primjer, kontroler TrainingController učinkovito upravlja HTTP zahtjevima povezanim s treninzima, omogućujući jednostavnu interakciju između klijentske aplikacije i poslužiteljske logike. Ovakav kontroler omogućuje lako dohvaćanje, dodavanje, uređivanje i brisanje treninga, čime se olakšava razvoj i održavanje aplikacija koje koriste treninge kao ključni dio svojih funkcionalnosti.

5. Arhitektura korisničkog sučelja

Moja aplikacija je izrađena koristeći Vue.js, progresivni JavaScript radni okvir za izradu korisničkih sučelja. Vue.js omogućava razvoj komponentno-orijentiranih aplikacija gdje se sučelje dijeli na male, višekratno upotrebljive komponente. Svaka komponenta sadrži svoj vlastiti HTML, CSS i JavaScript kod, što olakšava upravljanje i održavanje koda. Vue.js pruža reaktivnost, omogućujući automatsko ažuriranje sučelja kada se podaci promijene, te integraciju s vanjskim API-ima putem HTTP zahtjeva za dinamičko dohvaćanje i prikaz podataka.



Slika 5.1 Prikaz arhitekture korisničkog sučelja

Struktura Vue.js projekta organizirana je na način koji olakšava upravljanje i održavanje koda. Ovo su značajke:

- **assets**: Direktorij za statičke datoteke kao što su slike i stilovi.
- **components**: Direktorij koji sadrži Vue komponente, ključne dijelove sučelja.
- **App.vue**: Glavna Vue komponenta koja služi kao ulazna točka aplikacije.
- **constants.js**: Datoteka za definiranje konstantnih vrijednosti koje se koriste u aplikaciji.
- **main.js**: Ulazna JavaScript datoteka koja inicijalizira Vue aplikaciju.

- **RequestHandler.js**: Modul za rukovanje HTTP zahtjevima.
- **router.js**: Konfiguracija za Vue Router, upravljanje navigacijom unutar aplikacije.
- **store.js**: Konfiguracija za Vuex, centralizirano skladište podataka za upravljanje stanjem aplikacije.
- **style.css**: Datoteka za globalne stilove.

Ostale konfiguracijske datoteke kao što su postcss.config.js, tailwind.config.js, i vite.config.js koriste se za postavljanje alata i prilagođavanje okruženja za razvoj.

5.1. Rute

U Vue.js aplikaciji, rute definiraju navigaciju između različitih komponenti i stranica. U nastavku je prikazana struktura ruta:

Osnovne rute:

- /login vodi na komponentu Login.
- /register vodi na komponentu Register.
- /reports vodi na komponentu Reports.

Dinamične rute:

- /edit-report/:id za uređivanje izvještaja prema ID-u.
- /list-training/:teamId za prikaz treninga određenog tima.
- /player-details/:id za prikaz detalja igrača prema ID-u.

```
import ListPlayers from './components/ListPlayers.vue'
import PlayerDetails from './components/PlayerDetails.vue'
import EditPlayer from './components/EditPlayer.vue'
```

Slika 5.2 Prikaz umetanja datoteka

Slika 5.3 Postavljanje ruta za umetnute datoteke

Slika 5.4 Provjera autentifikacije korisnika

Konfiguracija *rutera* unutar Vue.js aplikacije omogućuje jednostavnu i učinkovitu navigaciju kroz različite dijelove aplikacije. Korištenjem createRouter s poviješću weba (createWebHistory), aplikacija koristi HTML5 način rada povijesti, što omogućuje čist i čitljiv URL bez *hash* znakova. Također, korištenje beforeEach metode za provjeru korisničke autentifikacije prije pristupa određenim rutama osigurava da samo ovlašteni korisnici mogu pristupiti zaštićenim dijelovima aplikacije.

Prednosti ovakve konfiguracije su višestruke. Prvo, omogućuje jednostavnu navigaciju unutar aplikacije, gdje korisnici mogu lako pristupiti različitim dijelovima aplikacije putem definiranih URL-ova. Drugo, centralizirano upravljanje rutama omogućuje da su sve rute definirane na jednom mjestu, što olakšava upravljanje i održavanje aplikacije. Konačno,

zaštita ruta omogućuje dodavanje sigurnosnih mjera, kao što je kontrola pristupa temeljenog na stanju aplikacije (npr. autentifikacija), čime se osigurava da samo autorizirani korisnici mogu pristupiti određenim dijelovima aplikacije.

Router.js file pruža robusnu strukturu za navigaciju unutar Vue.js aplikacije, omogućujući jednostavan prijelaz između različitih komponenti i stranica. Ova konfiguracija ne samo da poboljšava korisničko iskustvo, već i osigurava da aplikacija ostane sigurna i lako održiva, što je ključno za dugoročni razvoj i skaliranje aplikacije.

5.2. Komponente

U Vue.js, komponente su osnovni gradivni blokovi aplikacije, koji omogućuju modularno i višekratno korištenje koda. Komponenta se sastoji od tri osnovna dijela: template, script i style. Template definira HTML strukturu, script sadrži poslovnu logiku, dok style definira izgled komponente.

```
export default {
  data() {
    return {
      treninzi: [],
      isDodajVisible: false,
      noviTrening: {
        date: "",
        description: "",
        type: "",
        teamId: this.$route.params.teamId
      },
      trainingTypes: ["STRENGTH", "EXPLOSIVE", "RUNNING", "TACTICAL"],
    };
  },
```

Slika 5.5 Varijable unutar komponente ListTraining.vue

```
async mounted() {
  if (this.$store.state.user) {
    try {
      const response = await RequestHandler.getRequest(
        SPRING_URL.concat(`/training?teamId=${this.$route.params.teamId}`)
      );
      console.log("Fetched trainings:", response);

      if (Array.isArray(response)) {
        this.treninzi = response;
        this.treninzi.sort((a, b) => (a.idTraining > b.idTraining ? 1 : -1));
      } else {
        console.warn("Unexpected data format:", response);
      }
    } catch (error) {
      console.error("Error fetching trainings:", error);
    }
  }
},
```

Slika 5.6 Dohvaćanje podataka i spremanje u varijable unutar komponente Training.vue

```
navigateToEditPage(id) {
  this.$router.push({ name: 'EditTraining', params: { id: id } });
}
```

Slika 5.7 Navigacija na drugu lokaciju

Primjer komponente ListTraining.vue u Vue.js aplikaciji uključuje definiranje strukture prikaza i logike komponente. U dijelu templatea definira se struktura prikaza, kao što su forme za unos podataka i prikaz liste treninga. Ovdje se koriste Vue direktive poput v-if i v-for za upravljanje prikazom elemenata temeljem stanja aplikacije, omogućujući dinamičan prikaz podataka i formi.

Script dio komponente sadrži logiku koja upravlja komponentom, uključujući podatke (data), metode (methods) i životni ciklus komponente (mounted). U data objektu definirani su podaci o treninzima, vidljivost forme te novi trening koji se dodaje. Metode uključuju toggleDodaj za promjenu vidljivosti forme, dodajNoviTrening za dodavanje novog treninga, obrisiTrening za brisanje treninga i navigateToEditPage za navigaciju na stranicu za uređivanje. Ove metode omogućuju interakciju korisnika s komponentom i manipulaciju podacima unutar aplikacije.

Korištenje komponente ListTraining.vue moguće je unutar drugih komponenti ili direktno u glavnoj aplikaciji (App.vue) putem oznake <ListTraining />. Podaci i metode su dostupni putem this ključne riječi, a komunikacija s backendom se odvija putem RequestHandler modula. Komponente u Vue.js omogućuju razvoj aplikacija koje su jednostavne za

održavanje i proširivanje, omogućujući razdvajanje poslovne logike i vizualnog prikaza, čime se postiže veća modularnost i lakše upravljanje kodom.

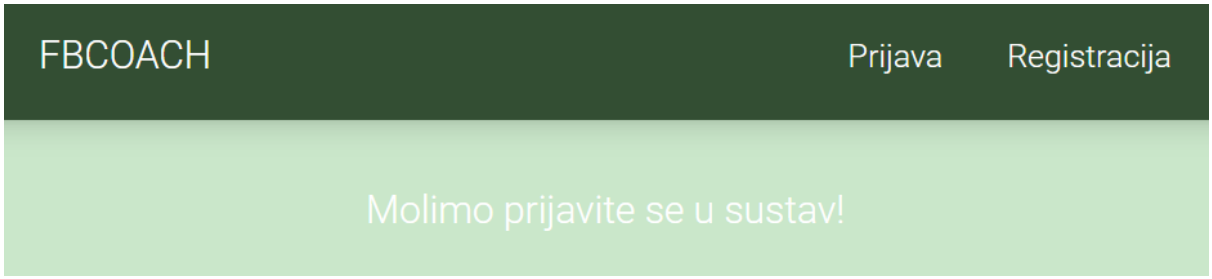
Ovaj pristup komponentama omogućuje razvoj složenih aplikacija na način koji je organiziran, učinkovit i jednostavan za razumijevanje, što je ključno za dugoročno održavanje i proširenje aplikacije.

6. Funkcionalnosti aplikacije

Aplikacija je zamišljena kao jednostavni alat za nogometne trenere koji pomaže u vođenju podataka nogometnih ekipa. Rješenje je osmišljeno kako bi korisnik na jednom mjestu brzo i efikasno mogao raspolagati podacima i iskoristio ih za daljnju analizu ekipe. Svi podaci su stvoreni od strane korisnika i dostupni su samo njemu.

6.1. Autorizacija

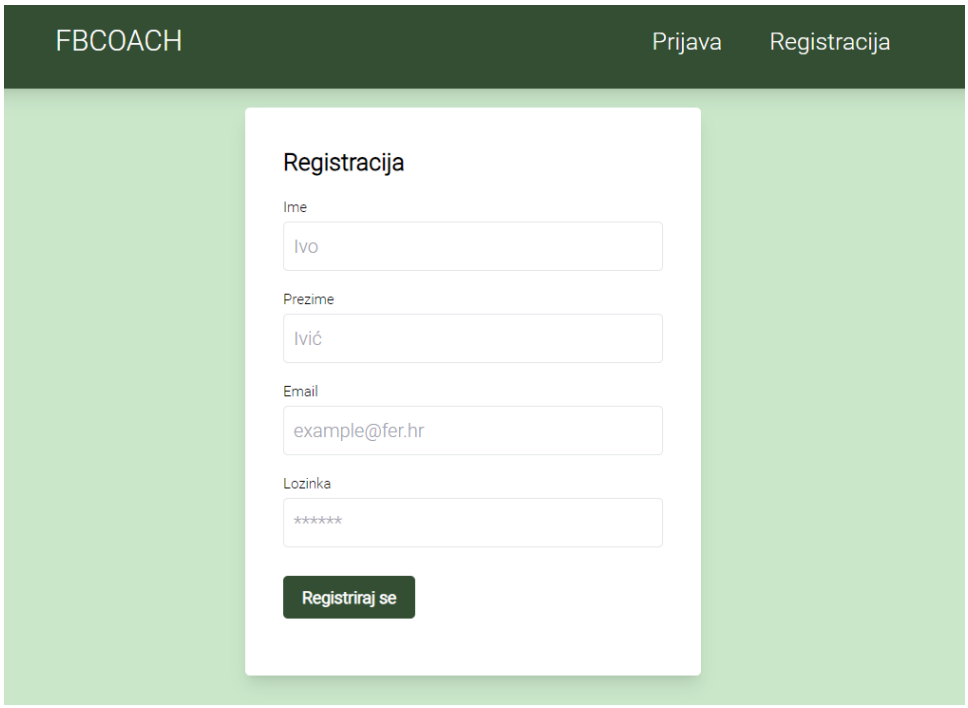
Ovo je prvi ekran koji se prikaže nakon dolaska na početnu stranicu. U navigacijskoj traci dostupni su gumbovi Prijava i Registracija. Dok se ne stisne jedan od njih na ekranu je prikazan poruka „Molimo prijavite se u sustav!“.



Slika 6.1 Početni ekran neautoriziranog korisnika

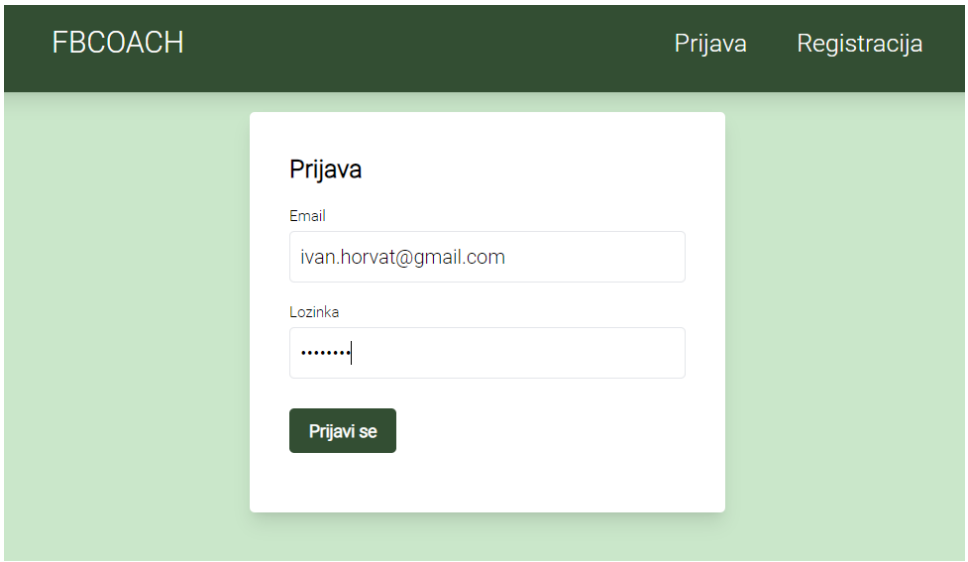
Nakon klika na gumb Registracija prikazana je forma za registraciju.

U prikazu forme za registraciju dostupna polja za unos su ime, prezime, email te lozinka. Prikazani su primjeri unutar pripadajućih polja.



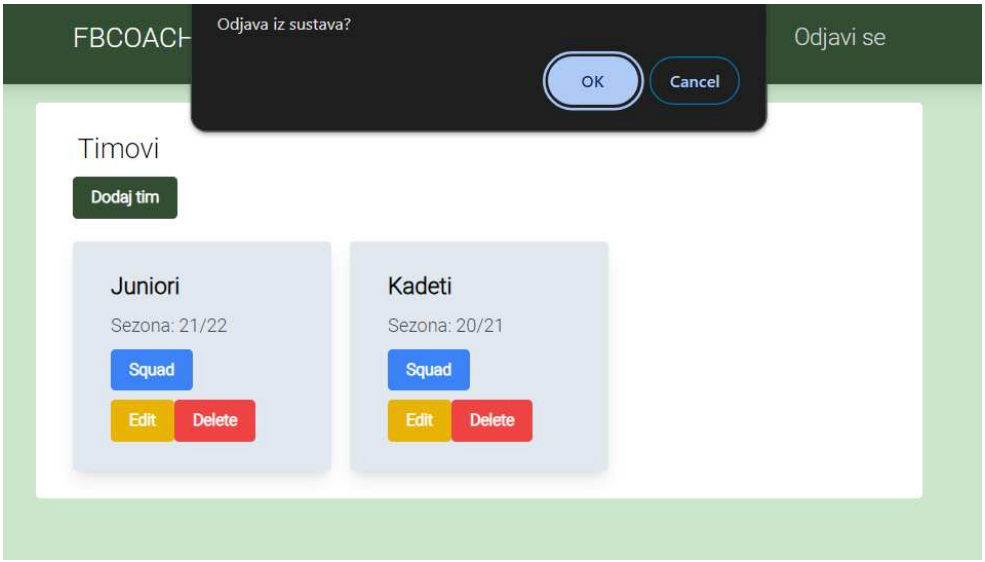
Slika 6.2 Forma za registraciju novog korisnika

U prikazu forme za prijavu dostupna polja za unos su email i lozinka. Prikaz lozinke se ne prikazuje te je prikazan točkama radi veće sigurnosti.



Slika 6.3 Forma za prijavu novog korisnika

Odjava iz aplikacije odvija se klikom na gumb „Odjavi se“ nakon čega se prikazuje skočni prozor sa pitanjem za ponovnu potvrdu odjave.

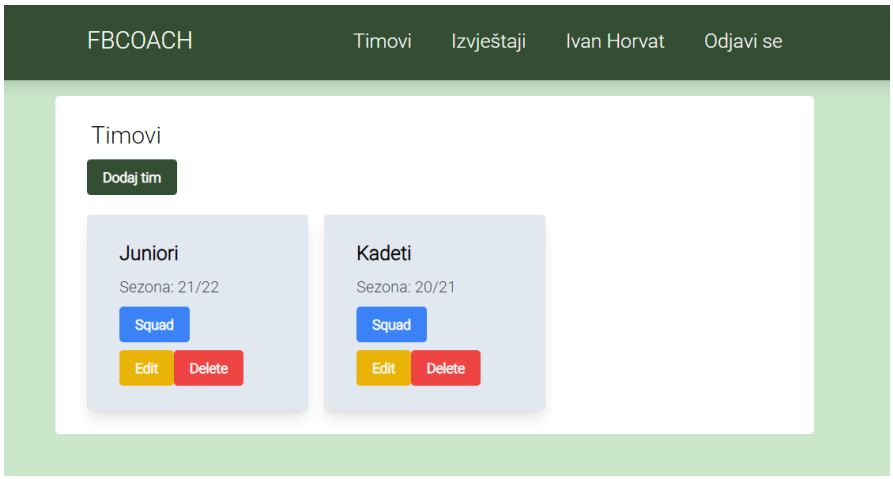


Slika 6.4 Skočni prozor za odjavu korisnika

6.2. Timovi

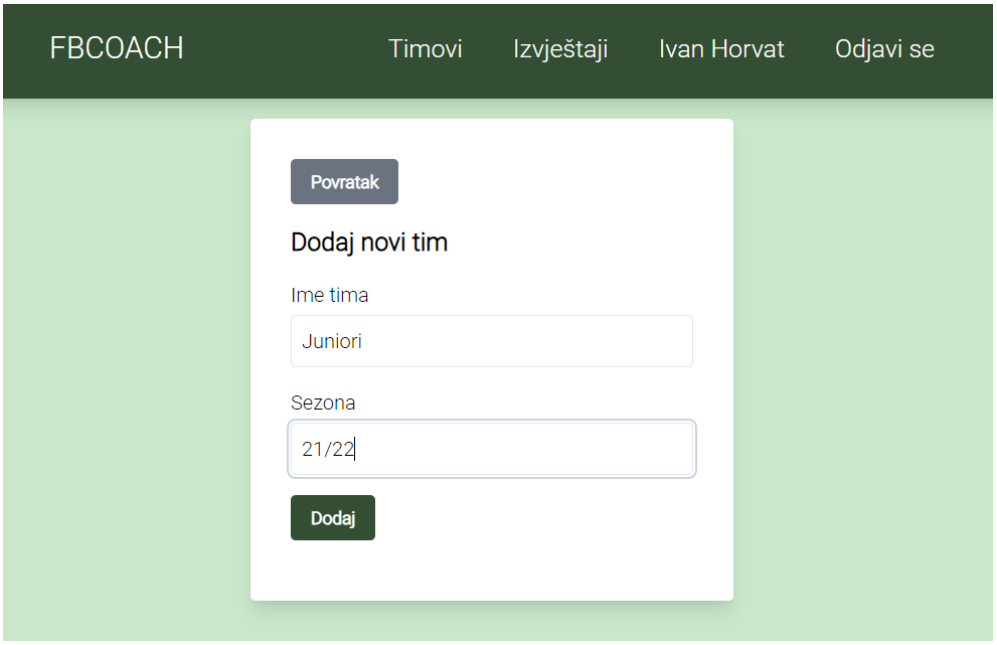
Prijavom u aplikaciju prvi prikaz je prikaz timova korisnika. Također se prikazuju gumb „Timovi“ i „Izvještaji“ koji su uvijek u navigacijskoj traci lijevo od imena i prezimena trenera te vode do pripadajućih prikaza.

U prikazu timova nudi se dodavanje uređivanje i brisanje pojedinog tima. Plavi gumb „Squad“ nudi prikaz detalja tog tima.



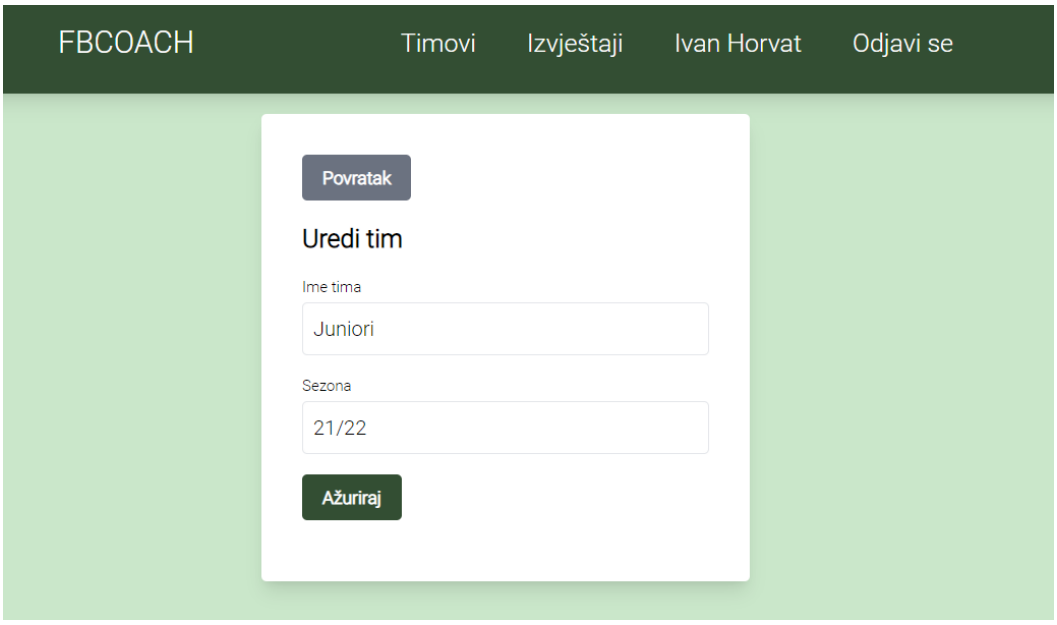
Slika 6.5 Početni prikaz nakon prijave korisnika

Na idućoj slici može se vidjeti popunjena forma za dodavanje novog tima. Pri vrhu nalazi se gumb „Povratak“ koji vodi natrag na početni prikaz timova i poništava formu. Klikom na gumb „Dodaj“ dolazi do kreiranja novog tima.



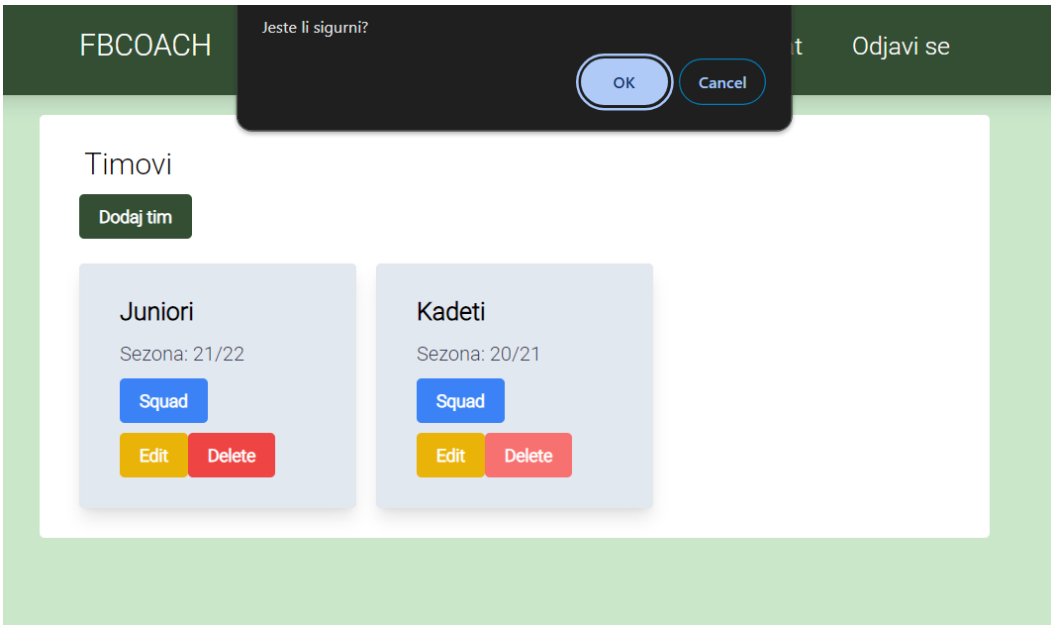
Slika 6.6 Forma za dodavanje novog tima

Iduća forma vrlo je slična prošloj, uz razliku na gumb „Ažuriraj“ koji ažurira ime ili sezonu tima. Do ove forme dolazi se klikom na žuti gumb „Edit“ u glavnom prikazu timova.



Slika 6.7 Forma za uređivanje tima

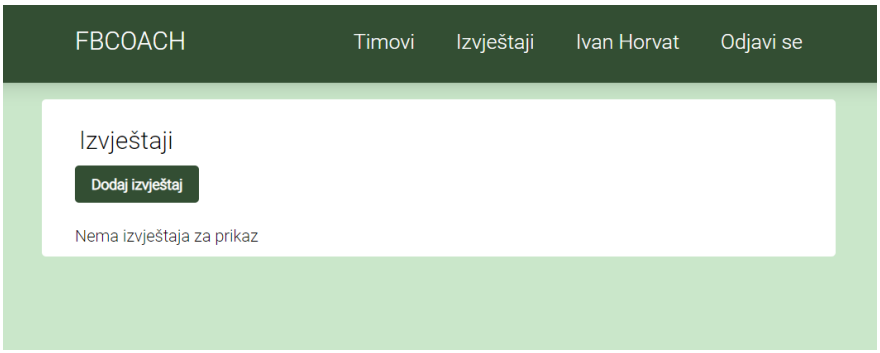
Brisanje pojedinog tima obavlja se klikom na gumb „OK“ u prikazu skočnog prozora.



Slika 6.8 Skočni prozor za potvrdu brisanja tima

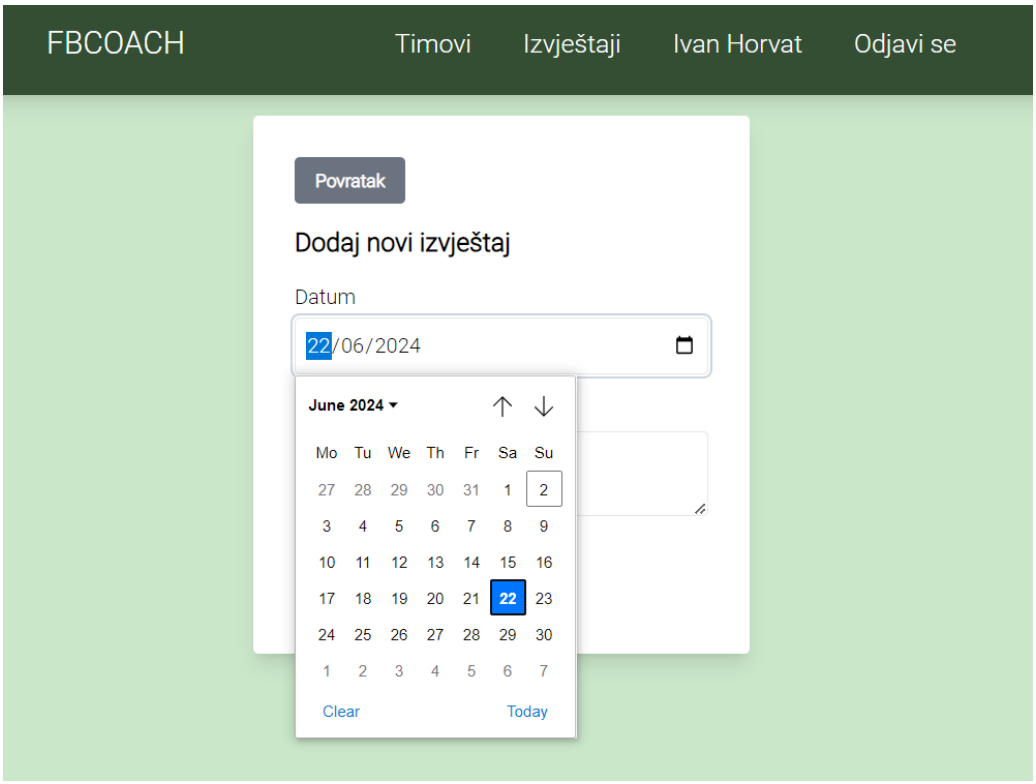
6.3. Izvještaji

Klikom na gumb „Izvještaji“ koji se nalazi na navigacijskoj traci dolazimo do prikaza svih izvještaja. Također možemo dodati nove klikom na gumb „Dodaj izvještaj“ ili urediti i izbrisati postojeće izvještaje.



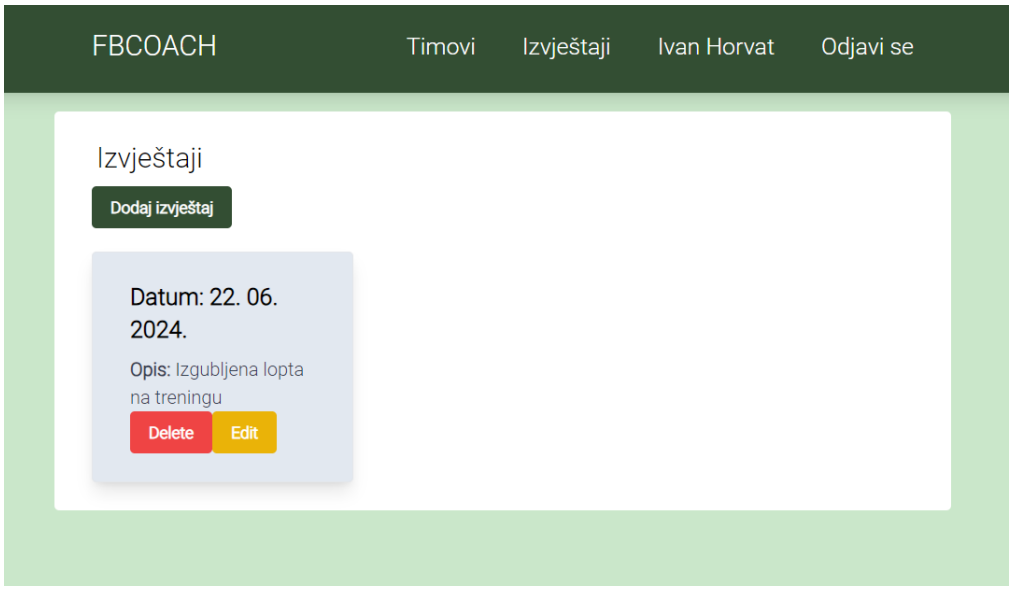
Slika 6.9 Prikaz bez izvještaja

Forma za dodavanje novih izvještaja uključuje kalendar u kojem lagano možemo izabrati željeni datum. Ispod polja za unos datuma nalazi se polje za unos samog izvještaja.



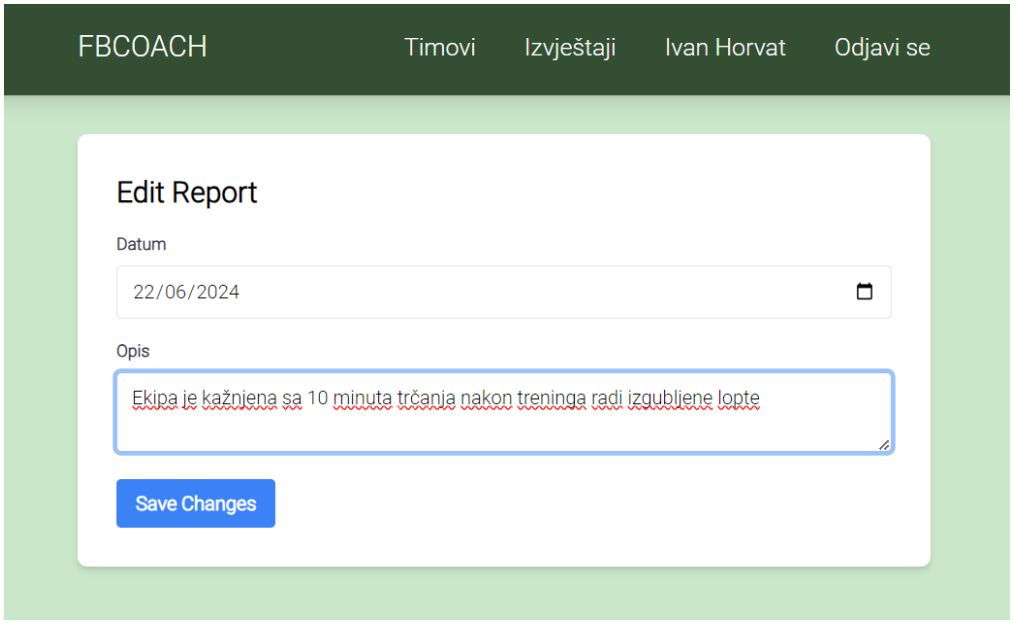
Slika 6.10 Forma za dodavanje novog izvještaja

Prikaz nakon dodaje novog izvještaja možemo vidjeti u nastavku. Omogućen je uređivanje i brisanje pojedinog izvještaja.



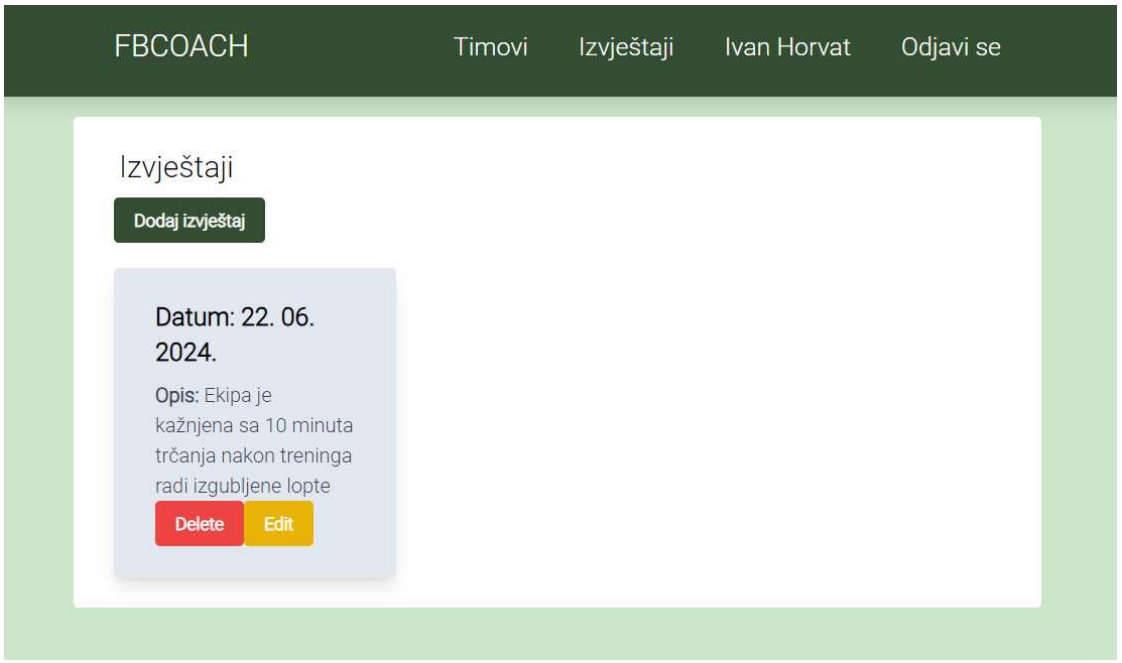
Slika 6.11 Popis izvještaja

Klik na gumb „Edit“ u prijašnjem prikazu vodi nas do forme za uređivanje. U ovoj formi uređen je tekst izvještaja te smo ga odlučili spremiti klikom na gumb „Save changes“.



Slika 6.12 Forma za uređivanje izvještaja

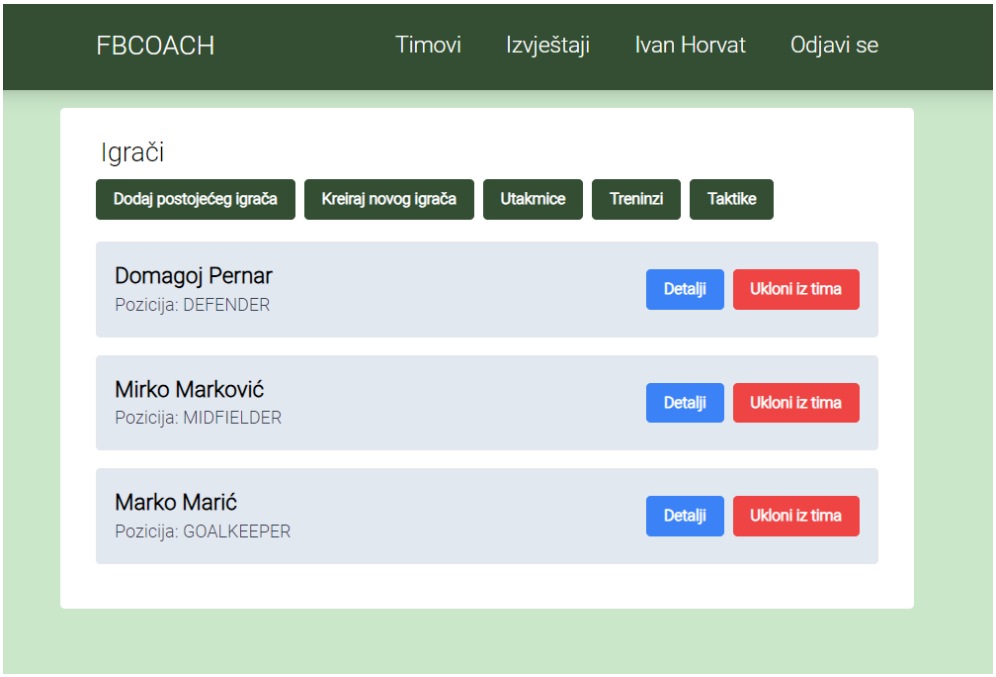
Nakon povratka na prikaz izvještaja novo uređeni tekst je prikazan odmah.



Slika 6.13 Prikaz uređenog izvještaja

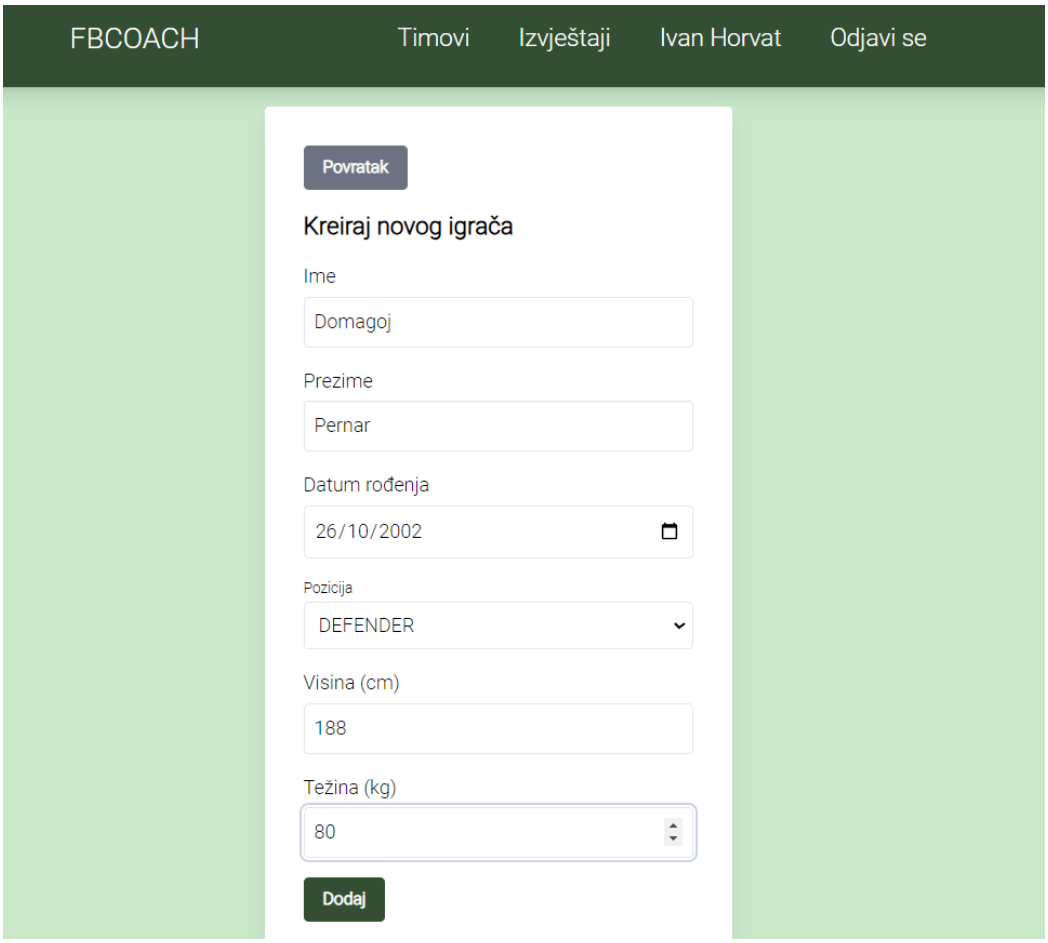
6.4. Igrači

Klikom na gumb „Squad“ kod pojedinog tima dolazimo do prikaza sa popisom igrača. Ovaj složeni prikaz nam nudi brojne opcije. Ukoliko želimo dodati već kreirane igrače koji nisu već vezani za naš tim do možemo klikom na gumb „Dodaj postojećeg igrača“. Ukoliko želimo stvoriti igrača koji nije u bazi podataka našeg korisnika, to možemo učiniti klikom na gumb „Kreiraj novog igrača“. Gumbovima „Utakmice“, „Treninzi“ i „Taktike“ odlazimo na pripadne stranice za prikaz utakmica, treninga odnosno taktika. Za svakog igrača dostupni su gumb „Detalji“ i „Ukloni iz tima“. Klikom na gumb „Ukloni iz tima“ igrača ne brišemo, već ga uklanjamo iz tog tima.



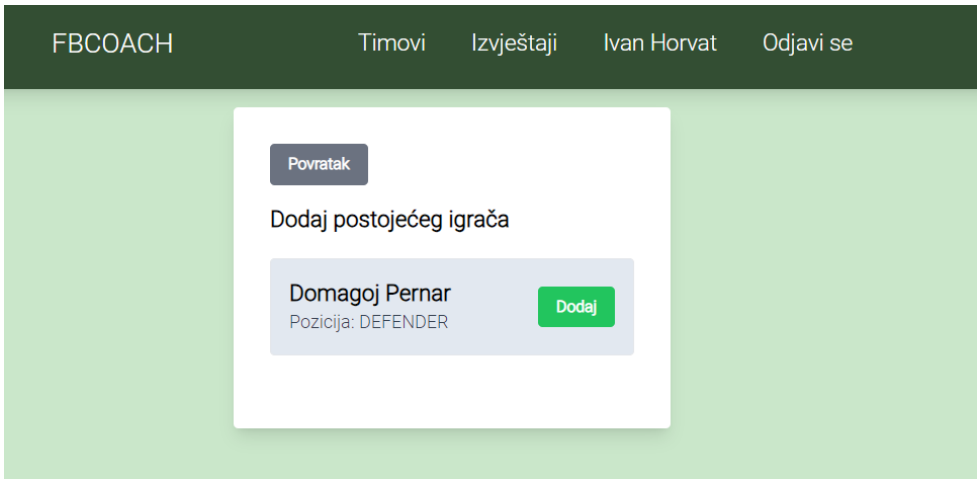
Slika 6.14 Složeni prikaz popisa igrača

Ukoliko smo se odlučili za kreiranje novog igrača dolazimo do forme za kreiranje novog igrača. Moramo popuniti sva polja i potvrditi kreaciju klikom na gumb „Dodaj“.



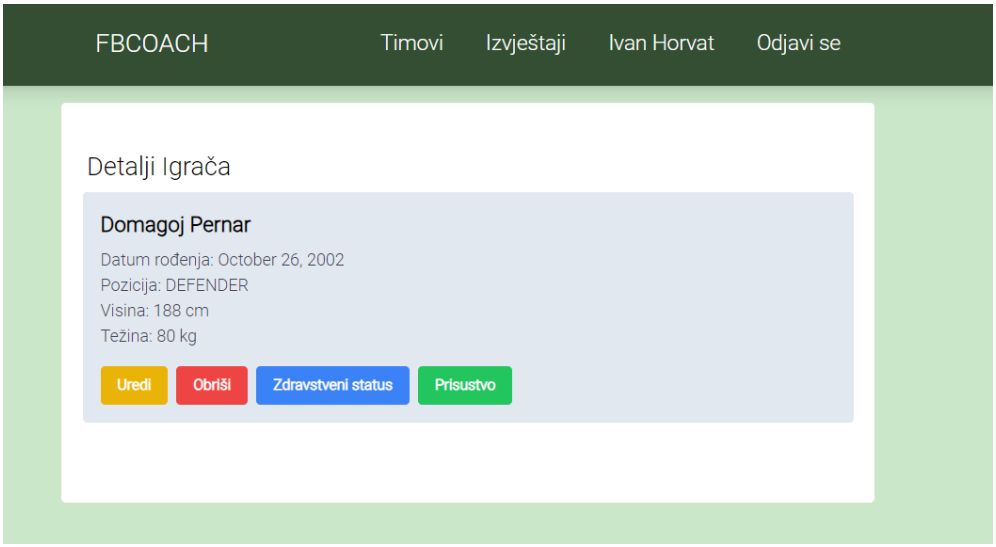
Slika 6.15 Forma za dodavanje novog igrača

Nakon što kreiramo novog igrača moramo ga dodati u tim. Igrač je kreiran u bazi no nije pridijeljen nijednom timu. Klikom na gumb „Dodaj“ taj igrač postaje dio tog tima. Ukoliko se odlučimo da ne želimo dodati nijednog igrača, gumb „Povratak“ nam omogućuje vraćanje na prikaz igrača.



Slika 6.16 Prikaz igrača za dodavanje u tim

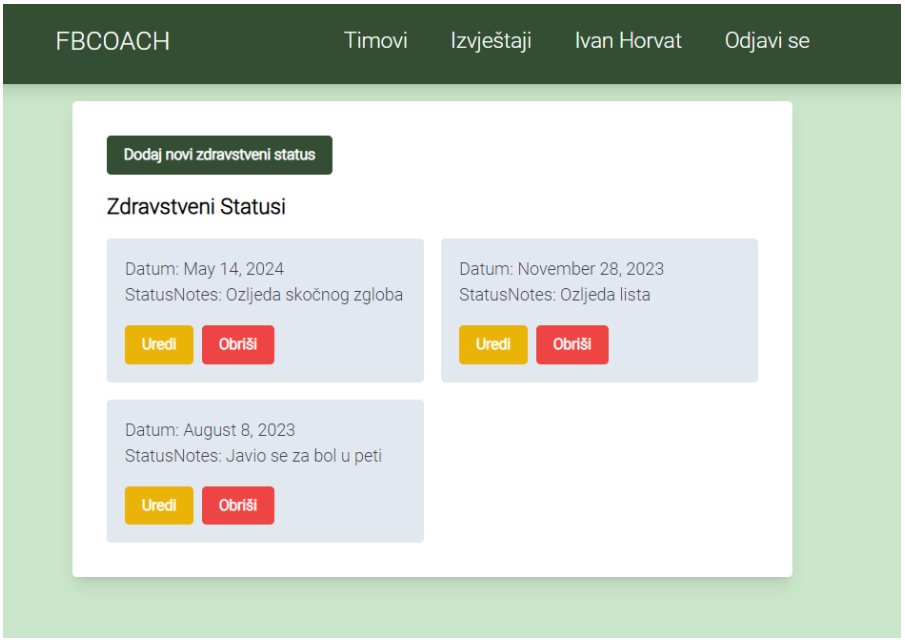
Klikom na gumb „Detalji“ dolazimo do detaljnog prikaza igrača. Prikazani su podaci poput datuma rođenja, pozicije, visine, težine. U ovom prikazu možemo urediti i izbrisati igrača te pogledati njegov zdravstveni status i prisustvo.



Slika 6.17 Detalji igrača

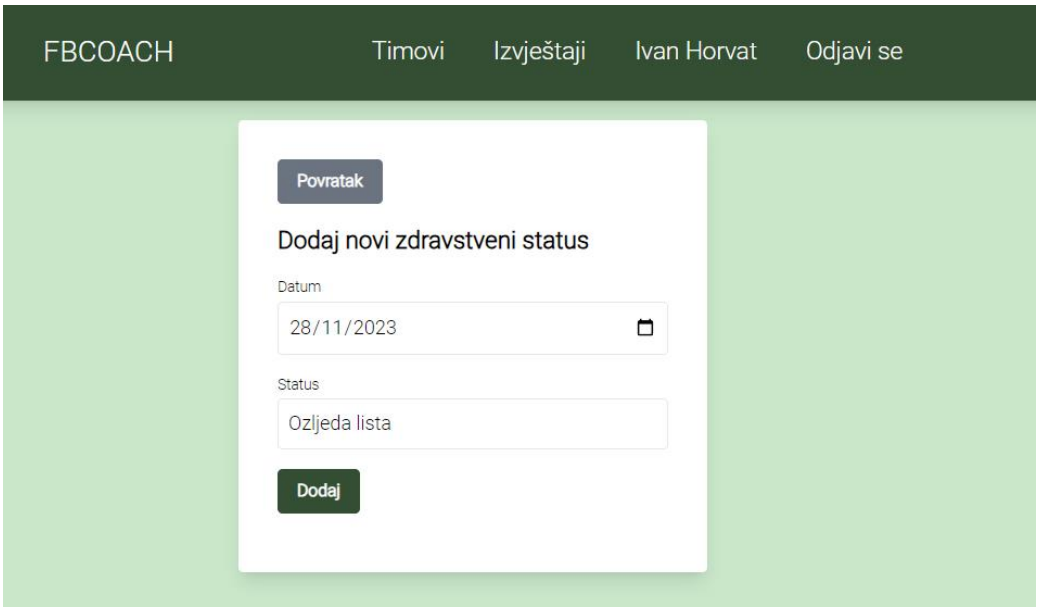
6.4.1. Zdravstveni status

Ukoliko smo se odlučili kliknuti na „Zdravstveni status“ izlistani su svi zdravstveni statusi sortirani po datumu. Jedan status sadrži datum i opis zdravstvenog stanja igrača.



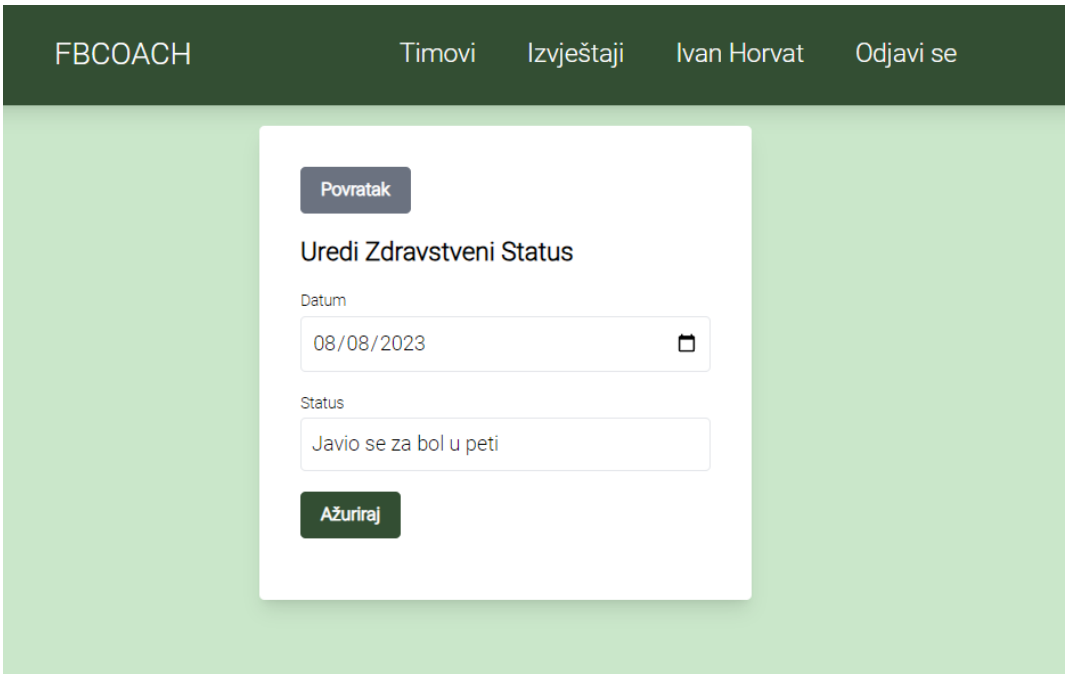
Slika 6.18 Popis zdravstvenih statusa igrača

Forma za dodavanje slična je i ostalima za dodavanje.



Slika 6.19 Forma za dodavanje igrača

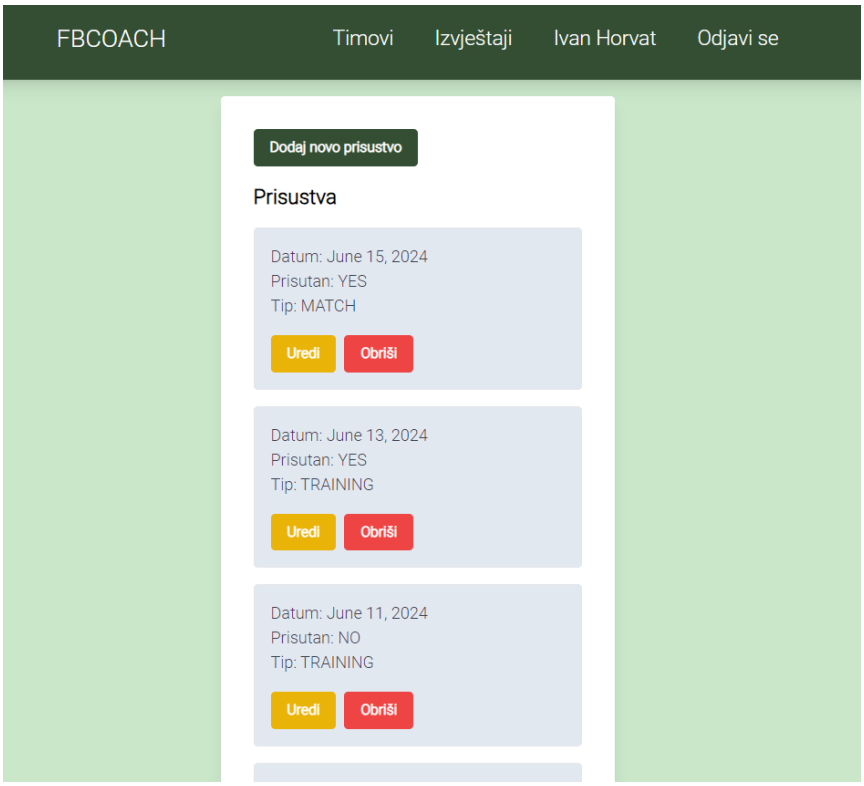
Forma za uređivanje također je slična i ostalima za uređivanje te sadrži postojeći sadržaj jednog statusa.



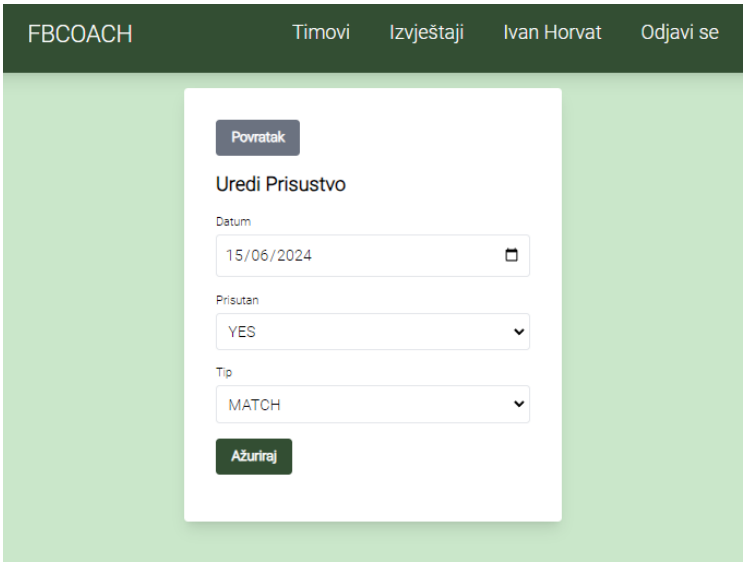
Slika 6.20 Forma za uređivanje igrača

6.4.2. Prisustva

Osim zdravstvenog statusa svaki igrač ima i svoj gumb za prisustva. Klikom na gumb „Prisustva“ dolazimo do popisa igračevih dolaznosti na treninge ili utakmice sortirane po datumu. Prisustva se također mogu urediti, obrisati te kreirati nova.



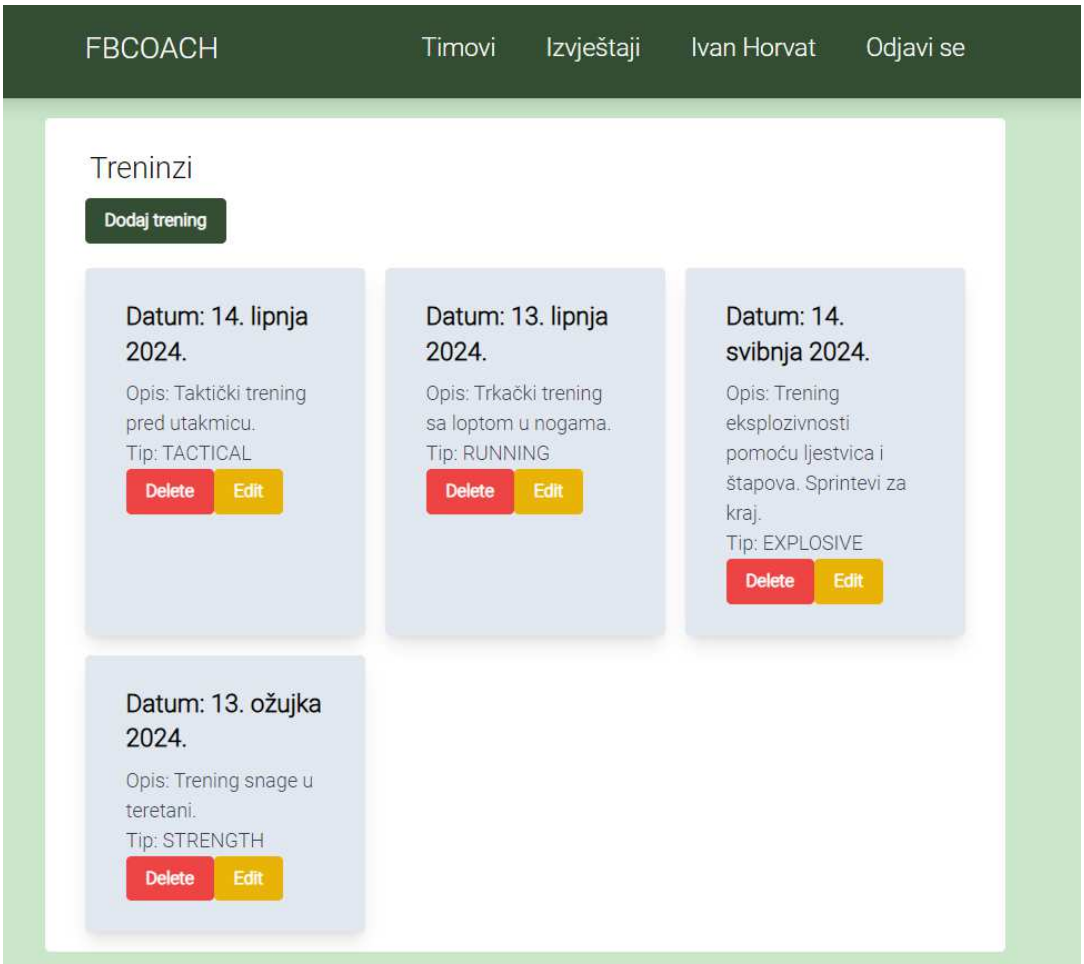
Slika 6.21 Popis prisustava igrača



Slika 6.22 Forma za uređivanje prisustva igrača

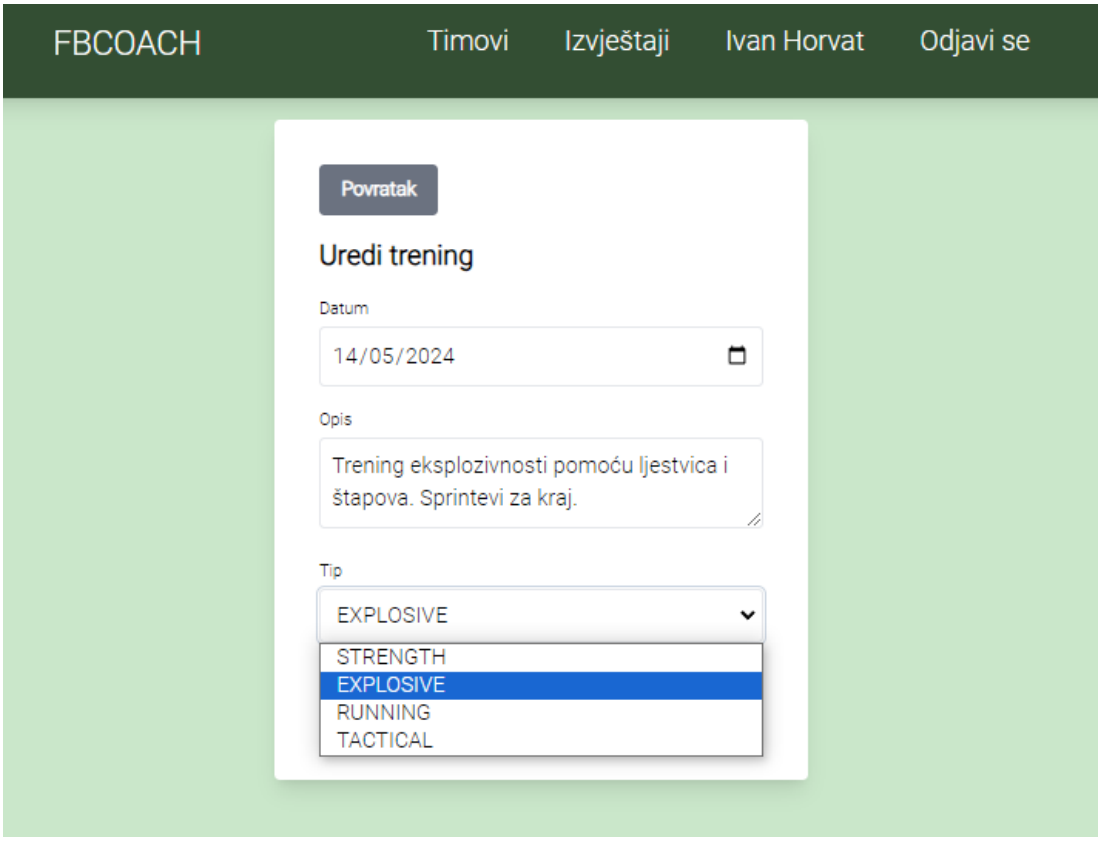
6.5. Treninzi

Ukoliko se sjetimo, na ekranu sa igračima nalaze se gumbovi „Utakmice“ , „Treninzi“ i „Taktike“. Gumb „Treninzi“ vodi nas do ekrana sa popisom treninga sortiranih po datumu. Također, kao i ostali entiteti ovaj je također podložan promjenama, moguće ga je obrisati, urediti i kreirati nove. Ovime trener može unaprijed napraviti treninge, pripremiti se i imati plan i biti organiziran što je i cilj same aplikacije.



Slika 6.23 Popis treninga pojedinog tima

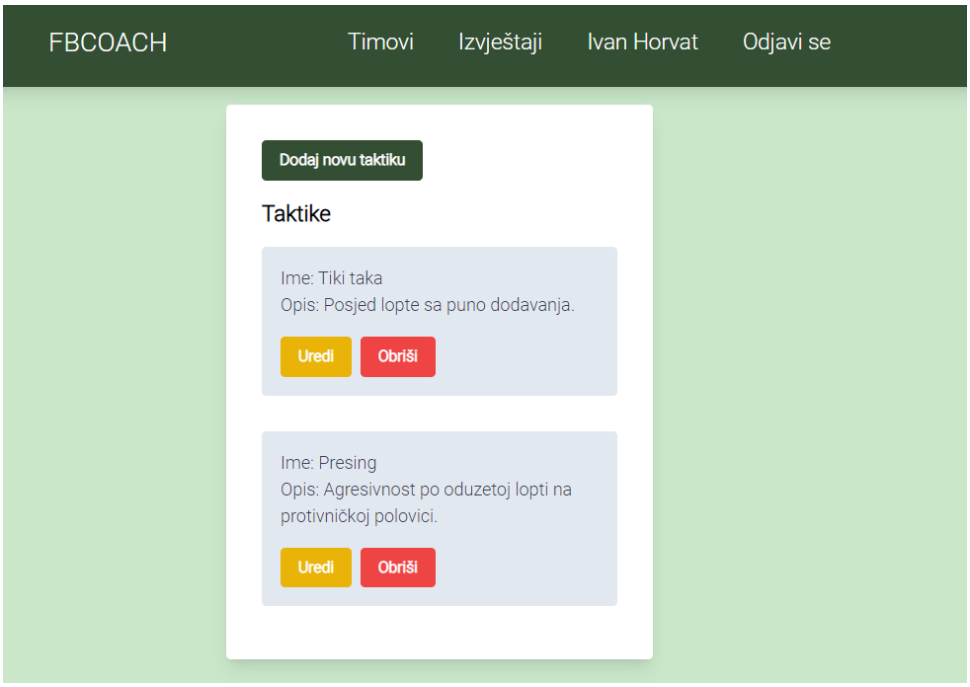
U prikazu za uređivanje imamo padajuću listu za tip treninga.



Slika 6.24 Forma za uređivanje treninga

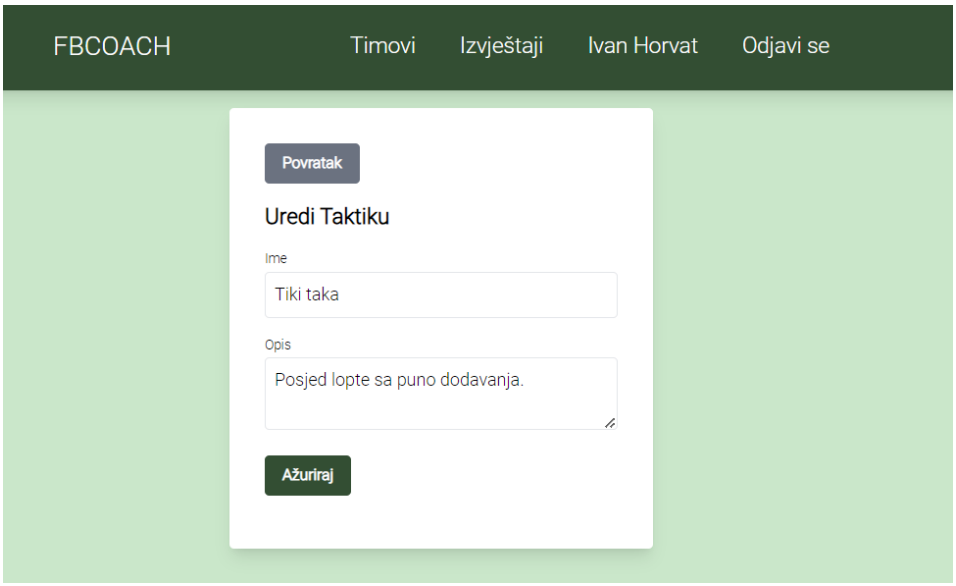
6.6. Taktike

Kao što je navedeno i ranije, sa prikaza igrača moguće je doći i do prikaza za taktike. U ovom ekranu trener dodaje taktike koje će koristiti za taj tim. Cilj je da trener na jednom mjestu ima svaku taktiku sa pripadajućim opisom. Koju kasnije može primijeniti na utakmicu.

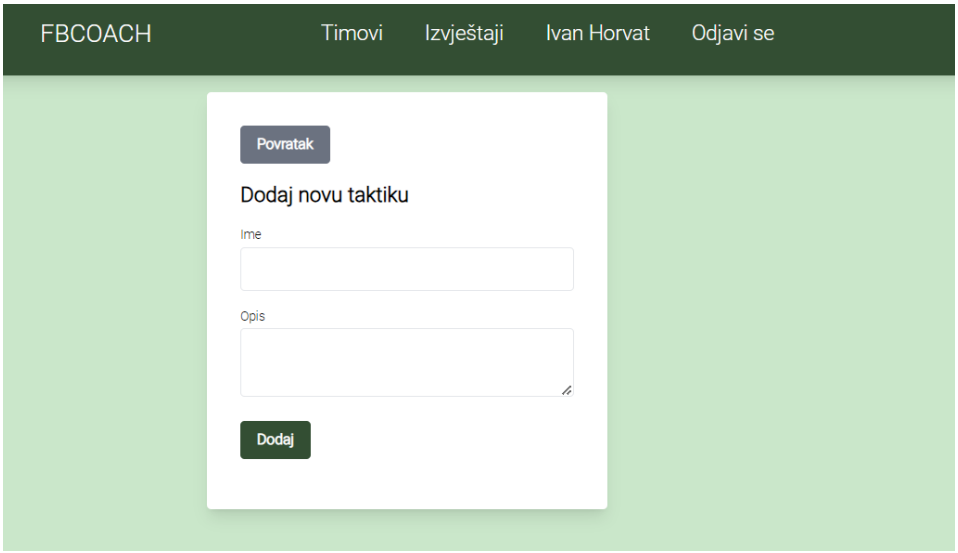


Slika 6.25 Prikaz taktika pojedinog tima

U nastavku su priložene slike koje prikazuju dodavanje i uređivanje pojedine taktike. Također je dostupno i brisanje taktike.



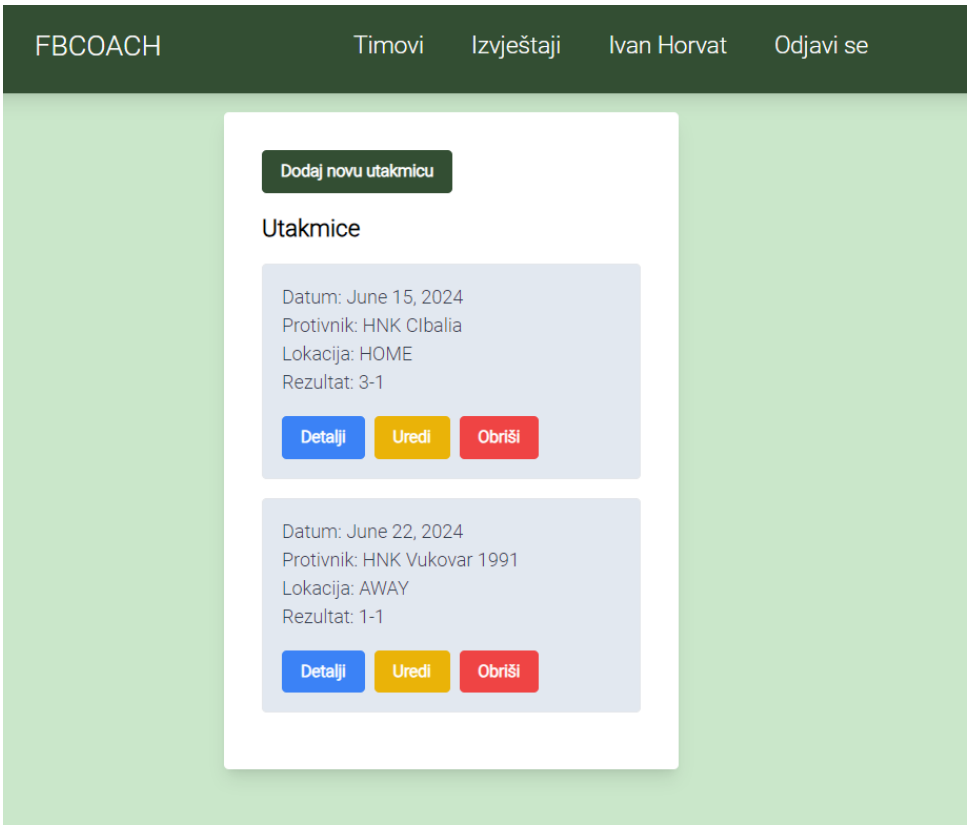
Slika 6.26 Forma za uređivanje taktike



Slika 6.27 Forma za dodavanje nove taktike

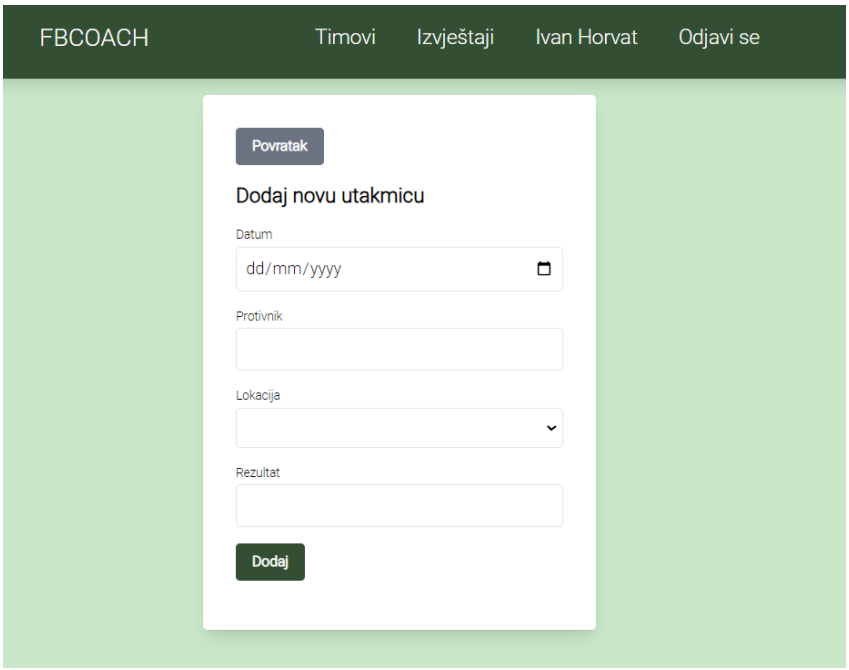
6.7. Utakmice

Zadnji prikaz je prikaz utakmica koji nam nudi informacije o datumu igranja, protivniku, lokaciji koja može biti „HOME“ što predstavlja domaću utakmicu i „AWAY“ što predstavlja gostujuću utakmicu. Osim ta tri atributa sam rezultat utakmice je prikazan.



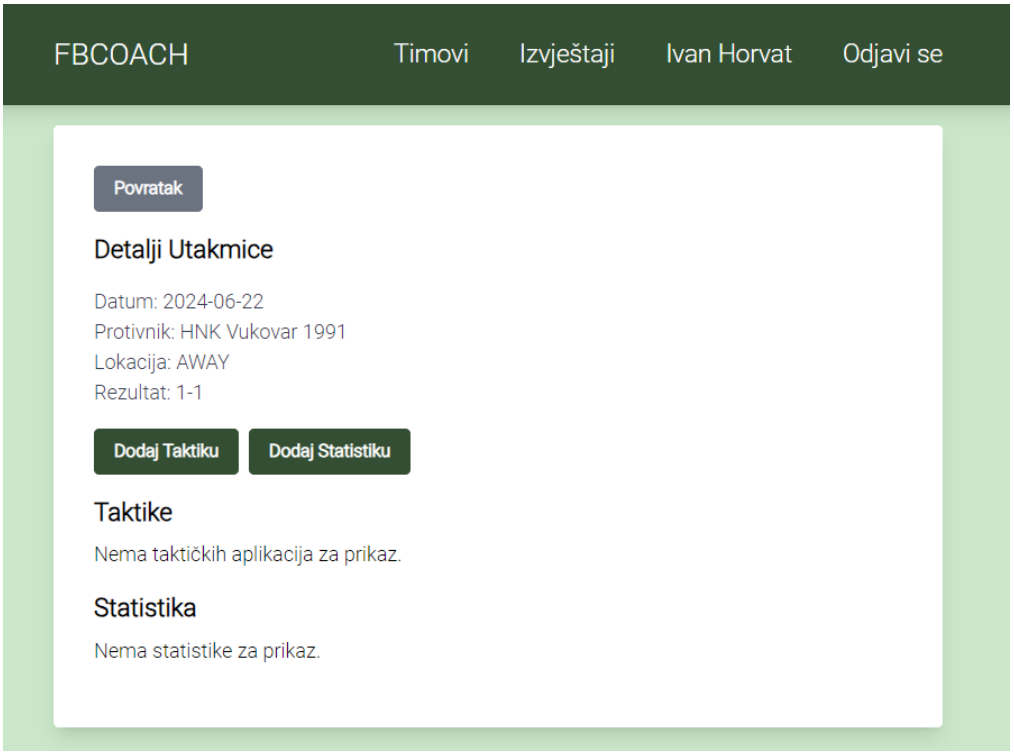
Slika 6.28 Popis utakmica pojedinog tima

Forma za dodavanje nove utakmice sadrži datum, protivnika, lokaciju te rezultat.



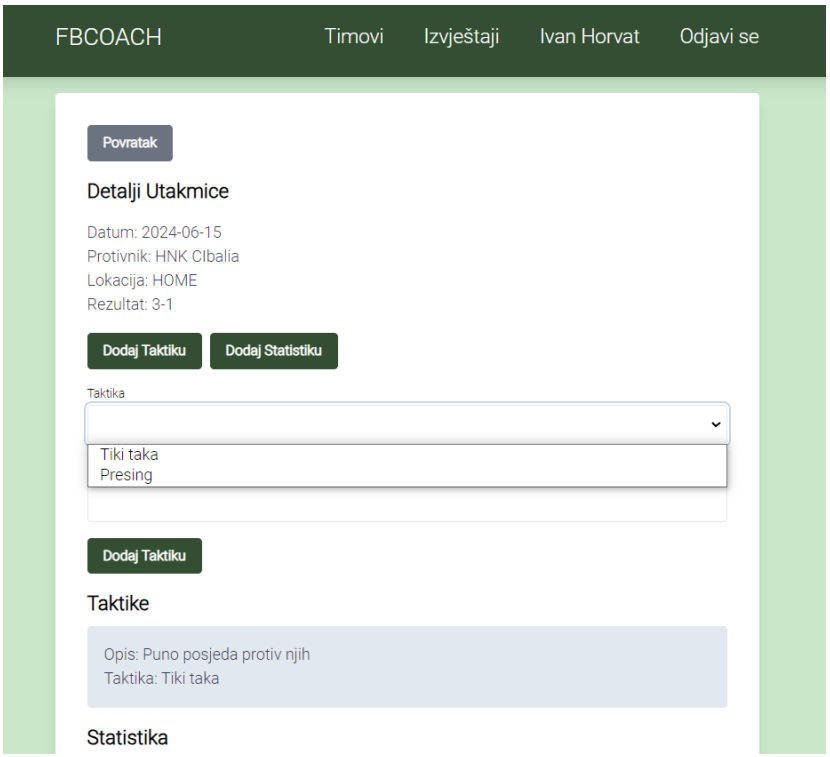
Slika 6.29 Forma za dodavanje nove utakmice

Klikom na gumb „Detalji“ otvara se najsloženiji prikaz u aplikaciji. U prvom dijelu prikazani su detalji utakmice kao i što je ranije navedeno. Gumb „Dodaj Taktiku“ dodaje taktiku koja je vezana za ekipu i utakmicu. To znači da je svaka taktika iskoristiva za više od jedne utakmice i opis se može prilagoditi samoj utakmici. Gumb „Dodaj statistiku“ dodaje broj golova i asistencija za igrača iz trenutnog tima.



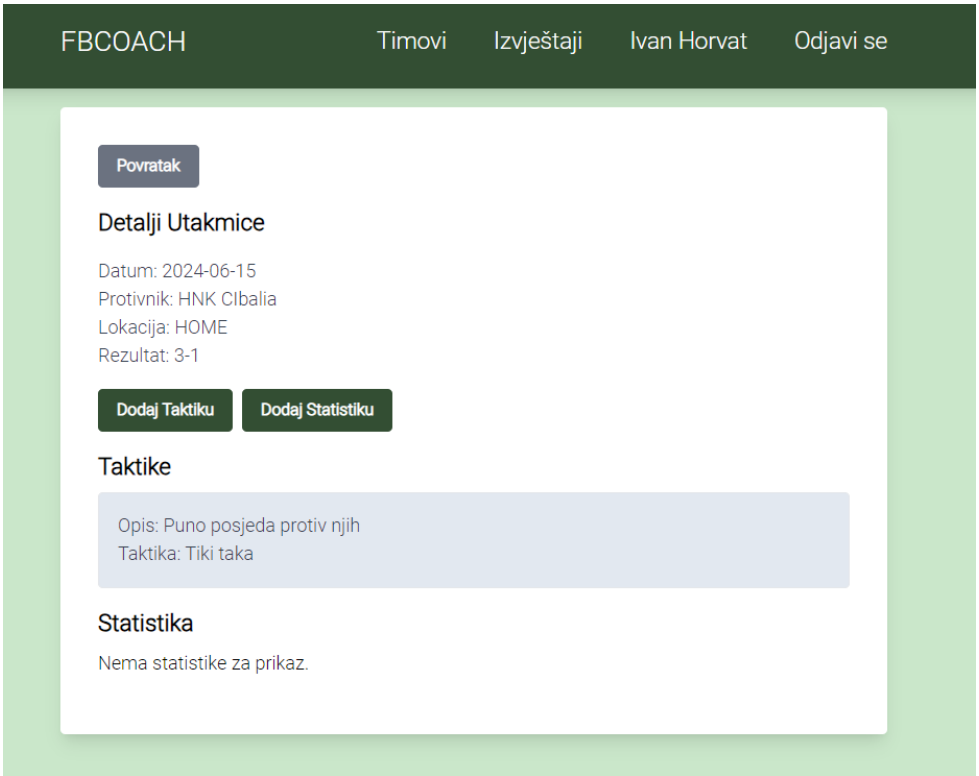
Slika 6.30 Detalji pojedine utakmice

Dodavanje primjene taktike na utakmicu dinamički je unutar složenog prikaza i sklanja se odnosno prikazuje klikom na gumb „Dodaj Taktiku“. Nudi se padajuća lista sa taktikom napravljenom u trenutno pozicioniranom timu.



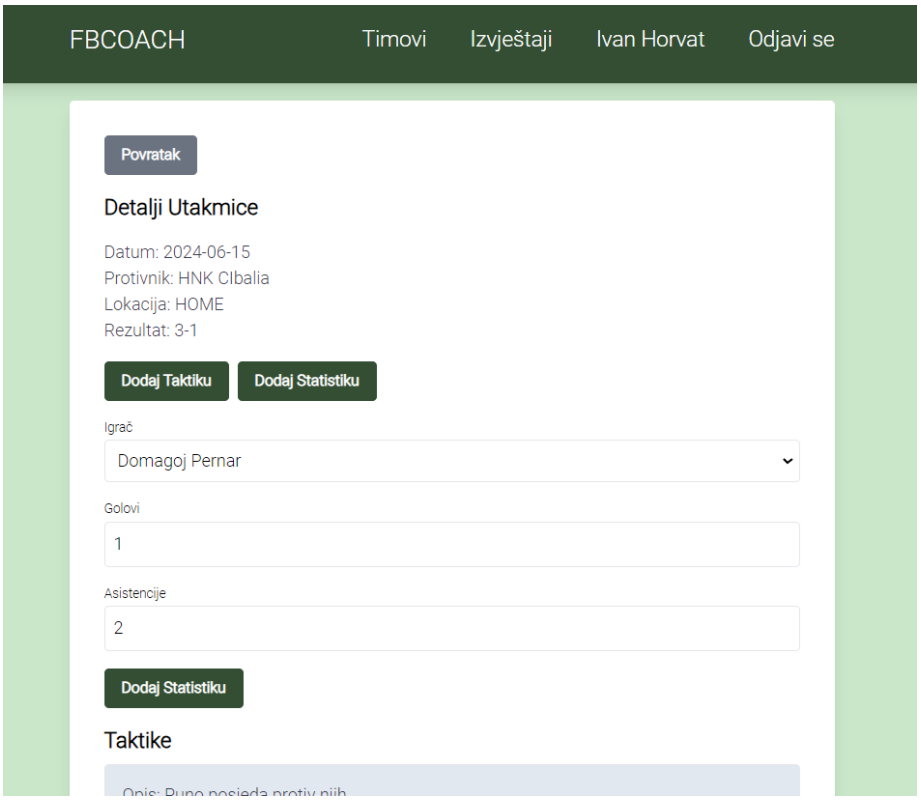
Slika 6.31 Dodavanje taktike za pojedinu utakmicu

Nakon što je primjena taktike na utakmicu uspješno dodana sa svojim opisom prikaže se u jednostavnom prikazu.



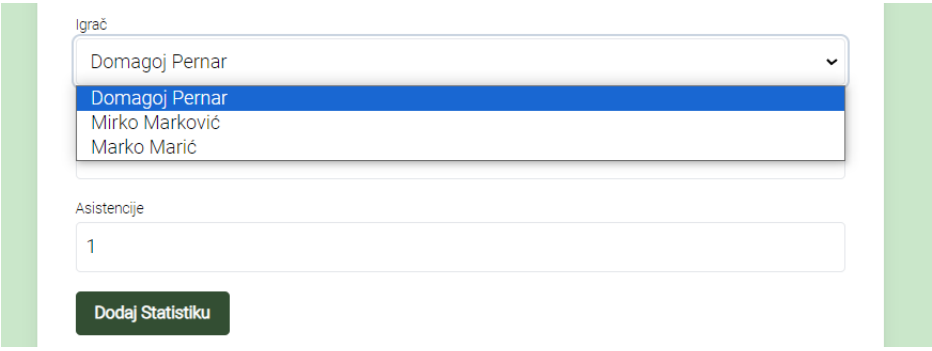
Slika 6.32 Prikaz dodane taktike za pojedinu utakmicu

Osim taktike moguće je dodati i statistiku. Dodavanje statistike igrača na utakmicu dinamički je unutar složenog prikaza i sklanja se odnosno prikazuje klikom na gumb „Dodaj Statistiku“. Nudi se padajuća lista sa statistikom igrača napravljenom u trenutno pozicioniranom timu.



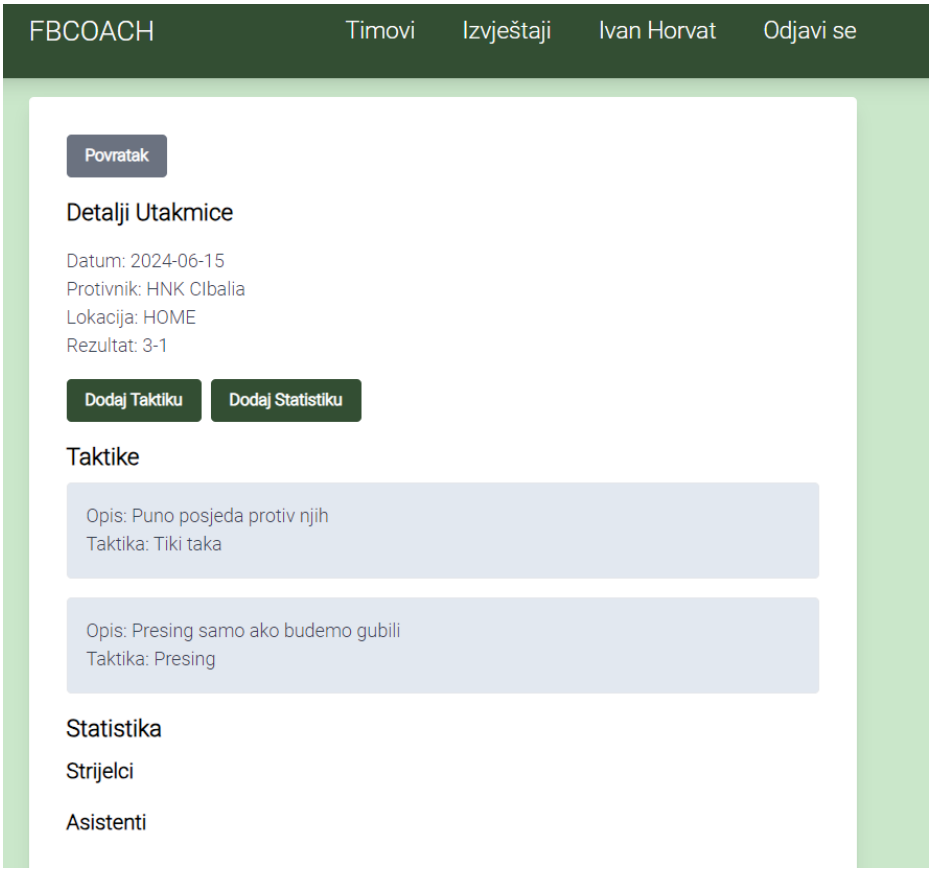
Slika 6.33 Prikaz dodavanja statistike za pojedinu utakmicu

Padajuća lista sa popisom igrača koji se nalazi u timu.



Slika 6.34 Padajuća lista sa igračima trenutnog tima

Nakon što smo dodali i statistiku igrača i primjenu taktike na utakmicu konačni složeni izgled prikazan je na slici u nastavku.



Slika 6.35 Prikaz detalja utakmice, primjene taktike i statistika igrača

U prethodnih nekoliko stranica prikazani su ekrani aplikacije kako bi se detaljno objasnili njihov dizajn i funkcionalnost. Svaki ekran je opisan sa svojim ključnim elementima i načinom interakcije s korisnikom. Dano je objašnjenje kako se koristi svaka funkcija unutar aplikacije, od dodavanja i uređivanja podataka do navigacije između različitih dijelova sustava. Kroz ove prikaze i opise, jasno je demonstrirano kako aplikacija pomaže korisnicima odnosno trenerima u upravljanju trenažnih procesa i praćenja performansi nogometnog tima, naglašavajući korisničku pristupačnost i efikasnost.

7. Korištene tehnologije i alati

U idućih nekoliko odlomaka ukratko su objašnjene korištene tehnologije te alati pomoću kojih je aplikacija razvijana od temelja.

7.1. PostgreSQL

PostgreSQL [9] je besplatan sustav za upravljanje relacijskim bazama podataka poznat po svojoj stabilnosti, fleksibilnosti i skalabilnosti. Razvijen je s naglaskom na usklađenost s SQL standardima i bogatom podrškom za napredne tipove podataka. PostgreSQL nudi širok spektar značajki poput podrške za složene upite, integriteta transakcija, replikacije i sigurnosti. Zahvaljujući svojoj robusnosti i ekstenzibilnosti, koristi se u raznim aplikacijama, od malih projekata do velikih globalnih sustava.

7.2. PgAdmin

PgAdmin [8] je popularan alat za upravljanje PostgreSQL bazama podataka, dizajniran za jednostavno upravljanje i administraciju. Pruža grafičko korisničko sučelje koje olakšava stvaranje i uređivanje baza podataka, tablica, indeksa, i drugih objekata unutar PostgreSQLa. PgAdmin također nudi alate za izvršavanje SQL upita, vizualizaciju podataka i upravljanje korisnicima i njihovim dozvolama. Podržava rad s udaljenim bazama podataka, što ga čini fleksibilnim rješenjem za administratore i developere.

7.3. Java

Java [4] je visoko popularan programski jezik poznat po svojoj platformi nezavisnosti, što znači da se programi napisani u Javi mogu pokretati na bilo kojem uređaju koji ima Java Virtual Machine (JVM). Razvio ga je Sun Microsystems 1995. godine, a danas ga održava Oracle Corporation. Java je objektno-orijentirana, što olakšava modularnost koda i ponovnu upotrebu. Koristi se u razvoju raznih aplikacija, od web aplikacija i mobilnih

aplikacija do kompleksnih softverskih rješenja. Njegova robusnost, sigurnost i skalabilnost čine ga jednim od najčešće korištenih jezika u industriji.

7.4. Spring

Radni okvir Spring [5] je popularan besplatni okvir za izradu aplikacija u Javi, razvijen od strane Rod Johnsona. Pruža infrastrukturu za upravljanje konfiguracijom, transakcijama i sigurnošću, što pojednostavljuje razvoj robusnih aplikacija. Jedna od ključnih značajki Springa je Dependency Injection (DI), koja omogućava lakše upravljanje objektima i njihovim zavisnostima unutar aplikacije. Korištenjem DI, okvir Spring omogućava kreiranje fleksibilnih i testabilnih aplikacija. Osim toga, Spring podržava Aspect-Oriented Programming (AOP) za razdvajanje poslovne logike od sistemskih usluga, te integraciju s različitim tehnologijama i okvirima.

7.5. Vue

Vue.js [6] je progresivni JavaScript okvir za izgradnju korisničkih sučelja. Dizajniran je da bude lako integriran u projekte, bilo da se radi o malim komponentama ili velikim aplikacijama. Vue koristi deklarativno renderiranje i komponentni model za jednostavniju organizaciju koda. Također podržava reaktivnost, što omogućuje automatsko ažuriranje sučelja kada se podaci promijene. Popularan je zbog svoje jednostavnosti i fleksibilnosti, te nudi opsežnu dokumentaciju koja olakšava učenje i implementaciju.

7.6. IntelliJ IDEA

IntelliJ IDEA [7] je integrirano razvojno okruženje (IDE) koje je razvila tvrtka JetBrains. Dizajnirano je za razvoj aplikacija u različitim programskim jezicima, s posebnim naglaskom na Javu. IntelliJ IDEA nudi napredne alate za kodiranje, uključujući automatsko dovršavanje koda, pametnu navigaciju i robusne alate za refaktoring. Također podržava širok spektar okvira kao što su Spring, Hibernate, i JavaScript, te integraciju s alatima za izgradnju i kontrolu verzija. Njegova intuitivna korisnička sučelja i opsežne mogućnosti čine ga popularnim izborom među razvojnim inženjerima.

7.7. Visual Studio Code

Visual Studio Code [11] (VS Code) je besplatan uređivač koda razvijen od strane Microsofta. Dizajniran je za razvoj aplikacija u raznim programskim jezicima, uključujući JavaScript, Python, C++, i mnoge druge. VS Code nudi značajke poput naprednog automatskog dovršavanja koda, integriranog terminala, alata za ispravljanje pogrešaka(debugging) i podrške za Git. Korisnicima je dostupna široka paleta proširenja koja dodatno povećavaju funkcionalnost uređivača, čineći ga prilagodljivim i moćnim alatom za programere.

7.8. Github

GitHub [10] je platforma u oblaku za pohranu, dijeljenje i suradnju pri pisanju programskog koda. Omogućuje spremanje koda u repozitorije, praćenje i upravljanje promjenama, pregledavanje koda od strane drugih te zajednički rad na projektima. Git, na kojem je GitHub izgrađen, je sustav za kontrolu verzija koji inteligentno prati promjene u datotekama i olakšava suradnju više ljudi na istim datotekama. GitHub omogućuje rad s Git repozitorijima putem preglednika, ali i lokalno, s kontinuiranim sinkroniziranjem promjena.

Zaključak

U ovom završnom radu predstavljeni su rezultati razvoja web aplikacije namijenjene nogometnim trenerima, koja olakšava upravljanje treninzima, utakmicama i igračima. Kroz analizu postojećih aplikacija, definiranje funkcionalnih i nefunkcionalnih zahtjeva te arhitektonsko osmišljavanje sustava, stvorena je robusna aplikacija koja uključuje sve ključne aspekte potrebne trenerima za efikasno vođenje tima.

Kroz ovaj rad pokazano je kako tehnologija može značajno unaprijediti sportske procese, pružajući trenerima alate za precizno planiranje, praćenje i analizu, što u konačnici doprinosi boljim rezultatima tima. Aplikacija ne samo da centralizira podatke, već i olakšava donošenje informiranih odluka, optimizaciju strategija i poboljšanje performansi igrača.

Ovaj projekt predstavlja sveobuhvatan alat za nogometne trenere koji žele iskoristiti prednosti moderne tehnologije u svom radu. Kroz jasnu strukturu, integraciju naprednih funkcionalnosti i korisničku pristupačnost, aplikacija postavlja temelje za daljnji razvoj i unapređenje sportskih aplikacija.

Literatura

[1] „Football coach plus“ , Poveznica: https://www.c4e1.com.au/products/Football-Coach-Plus.html pristupljeno 12. ožujka 2024.

[2] „Bcoach“ , Poveznica: https://bcoach.app/en/home/ pristupljeno 12. ožujka 2024.

[3] „Ultrax“ , Poveznica: https://www.ultrax.ai/ pristupljeno 12. ožujka 2024.

[4] „Java“ , Poveznica: https://www.java.com/en/download/help/whatis_java.html pristupljeno 20. ožujka 2024.

[5] „Spring“ , Poveznica: https://spring.io/why-spring pristupljeno 20. ožujka 2024.

[6] „Vue“ , Poveznica: https://vuejs.org/ pristupljeno 28. ožujka 2024.

[7] „IntelliJ IDEA“ , Poveznica: https://www.jetbrains.com/idea/ pristupljeno 20. ožujka 2024.

[8] „PgAdmin“ , Poveznica: https://www.pgadmin.org/ pristupljeno 22. ožujka 2024.

[9] „PostgreSQL“ , Poveznica: https://www.postgresql.org/about/ pristupljeno 22. ožujka 2024.

[10] „Github“ , Poveznica: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git pristupljeno 20. ožujka 2024.

[11] „Visual Studio Code“ , Poveznica: https://code.visualstudio.com/ pristupljeno 28. ožujka 2024.

[12] „Futsal“, Poveznica: https://hr.wikipedia.org/wiki/Futsal pristupljeno 12. ožujka 2024.

Sažetak

Naslov: Aplikacija za nogometne trenere

Ovaj rad opisuje razvoj web aplikacije za nogometne trenere. Nakon opisa postojećih rješenja i alata za nogometne trenere, izdvojeni su funkcionalni i nefunkcionalni zahtjevi moje aplikacije. Poštujući njih napravljena je web aplikacija za nogometne trenere koja podržava pregled timova, igrača, utakmica, treninga i taktika te detalje igrača poput evidencije prisutnosti, zdravstvenog stanja igrača i vođenje izvještaja. Opisana je arhitektura sustava koja sadrži opis baze podataka, pozadinske i vidljive strane aplikacije. Na samom kraju prikazane su korištene tehnologije.

Ključne riječi: nogomet, trener, web aplikacija, Java, Vue, PostgreSQL

Summary

Title: Football coach application

This thesis describes the development of a web application for soccer coaches. After describing the existing solutions and tools for football coaches, the functional and non-functional requirements of my application were displayed. Respecting them, a web application for football coaches was created that supports the overview of teams, players, matches, training and tactics, as well as player details such as attendance records, player health status and keeping reports. The system architecture is described, which contains a description of the database, background and visible sides of the application. At the very end, the technologies used are presented.

Key words: football, coach, web application, Java, Vue, PostgreSQL