Internet aplikacija za vođenje statistike o treninzima

Duvnjak, Predrag

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Internet aplikacija za vođenje statistike o treninzima

Diplomski rad

Predrag Duvnjak

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 29.11.2021.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Predrag Duvnjak
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1049R, 06.10.2019.
OIB studenta:	02446163766
Mentor:	Doc.dr.sc. Ivan Vidović
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Tomislav Matić
Član Povjerenstva 1:	Doc.dr.sc. Ivan Vidović
Član Povjerenstva 2:	Izv. prof. dr. sc. Ivan Aleksi
Naslov diplomskog rada:	Internet aplikacija za vođenje statistike o treninzima
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U ovom radu potrebno je istražiti postojeća rješenja za vođenje statističkih podataka o treninzima sportaša. Nakon istraživanja potrebno je razviti vlastitu internet aplikaciju koja će korisniku omogućiti jednostavan unos podataka o treningu za više sportaša, uređivanje svih podataka, prikaz podataka u obliku grafova te izvod izvještaja za jednog ili više odabranih sportaša. Tehnologiju za izradu je moguće proizvoljno odabrati, ali je potrebno osigurati responzivnost aplikacije.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 1 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	29.11.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 19.12.2021.

Ime i prezime studenta:	Predrag Duvnjak
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1049R, 06.10.2019.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Internet aplikacija za vođenje statistike o treninzima**

izrađen pod vodstvom mentora Doc.dr.sc. Ivan Vidović

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1

2. POSTOJEĆA RJEŠENJA 2

3. KORIŠTENE TEHNOLOGIJE I ALATI 3

3.1. ASP.NET Core 3

3.2. Entity Framework Core 3

3.3. PostgreSQL i pgAdmin 4 3

3.4. React 3

3.5. JavaScript 4

3.6. CSS 4

3.7. HTML 4

3.8. React-Bootstrap 4

3.9. Postman 4

4. IZRADA APLIKACIJE 5

4.1 Baza podataka 5

4.1.1. Izgled baze podataka 5

4.1.2. Izrada baze podataka 6

4.2 Web API 9

4.2.1. Kontroleri i sloj servisa 9

4.2.2. Autentifikacija i tokeni 11

4.2.3. Testiranje web API-a 12

4.3 Korisničko sučelje 13

4.3.1. Komponente i hook-ovi 13

4.3.2. Navigiranje unutar aplikacije 14

4.3.3. Grafovi 14

4.3.4. Dodatne biblioteke 15

5. IZGLED APLIKACIJE 17

6. ZAKLJUČAK.....24

LITERATURA.....25

SAŽETAK.....27

ABSTRACT.....28

ŽIVOTOPIS.....29

1. UVOD

Tema diplomskog rada izrada je internet aplikacije koja bi olakšala trenerima vođenje različitih grupa sportaša, te im pružila mogućnost praćenja statistike. Ovakav oblik aplikacija koristan je osobama koje su zadužene za vodstvo više sportaša jer im omogućuje lakše vođenje bilješki o ostvarenim rezultatima. Trener pristupom internetskoj stranici ima mogućnost prijave na postojeći profil ili registriranje novoga profila. Nakon uspješne prijave, neke od pruženih funkcionalnosti su stvaranje različitih grupa sportaša, dodavanje treninga te pregled statistike pojedinih osoba u svrhu efikasnijeg praćenja napretka.

Internet aplikacija sastoji se od više dijelova potrebnih kako bi navedena usluga mogla biti pružena krajnjim korisnicima, a tehnologije se mogu podijeliti na *front end, back end* i bazu podataka. *Front end* dio predstavlja izradu sučelja s kojim korisnik ima interakciju, te pomoću kojega može unositi informacije, uređivati ih i uklanjati. *Back end* dio sadržava web API koji isporučuje zahtjeve koji pristižu od korisnika, te omogućuje komunikaciju s bazom podataka.

Nakon uvodnog poglavlja, u drugom poglavlju opisana su postojeća rješenja te usporedbe s rješenjem danim u diplomskom radu. Treće poglavlje sadrži tehnologije koje su korištene prilikom izrade i testiranja aplikacije. Četvrto poglavlje opisuje izradu aplikacije te se sastoji od dijelova koji opisuju oblik i način izrade baze podataka, izradu web API-a na kojega pristižu zahtjevi od korisnika te korisničkog sučelja s kojim osobe imaju interakciju prilikom korištenja aplikacije. Unutar petog poglavlja detaljno je prikazan izgled korisničkoga sučelja te je pojašnjen način rukovanja s funkcionalnostima kojima korisnici mogu rukovati. Kao zadnje poglavlje naveden je zaključak s osvrtom na rad i postignute rezultate.

2. POSTOJEĆA RJEŠENJA

U ovome poglavlju napravljen je pregled postojećih rješenja za vođenje sportske statistike. U literaturi je navedeno više postojećih rješenja, a prema platformi za koju su namijenjene se mogu podijeliti na: desktop aplikacije, internetske aplikacije i mobilne aplikacije.

Završni rad „Aplikacija za praćenje fitness treninga“ [1] opisuje izradu aplikacije namijenjenu trenerima pomoću koje se unose podatci o osobama koje dolaze u teretanu te količina njihovih tjednih dolazaka. Također, pružena je funkcionalnost prijedloga vježbi na osnovu informacija unesenih o odabranoj osobi. Neke od značajnijih razlika navedenoga rješenja u odnosu na rješenje opisano u ovome radu je to što u aplikaciji opisanoj u ovome radu postoji mogućnost zapisivanja ostvarenih rezultata tijekom treninga, prikaz statistike u obliku grafova te korištenje aplikacije pristupom internetskoj stranici.

Diplomski rad „Izrada i testiranje Angular aplikacije za nogometnu statistiku“ [2] opisuje stvaranje aplikacije koja trenerima omogućuje praćenje nogometne statistike. Jedna od većih razlika rada je to što je namijenjen za praćenje statistike nogometnih klubova i nogometaša, dok je u ovome diplomskom radu aplikacija zamišljena tako da može biti od koristi trenerima koji su zaduženi za raznolike oblike treninga te bilježenje rezultata, praćenje statistike i napretka.

Diplomski rad „Web aplikacija za upravljanje teretanom“ [3] opisuje izradu internetske aplikacije koja služi za upravljanje radom teretane. Postoji više mogućih uloga koje su ponuđene osobama, kao što su admin, trener i korisnik. Ovisno o tome kojemu tipu uloge osoba pripada, dolazi do podjele mogućih funkcionalnosti. To predstavlja jednu od značajnih razlika u odnosu na aplikaciju opisanu u ovome radu koja je namijenjena specifično za trenere koji samostalno unose grupe i sportaše, te podatke dobivene tijekom odrađenih treninga.

Sportlyzer [4] predstavlja softver za upravljanje timovima koji obuhvaćaju područja kao što su tim menadžment, razvoj igrača te komunikacija članova. Neke od pruženih funkcionalnosti su: rukovanje članovima, praćenje prisutnosti sportaša, notifikacije SMS-om i elektroničkom poštom, detaljno planiranje treninga te analiza napretka sportaša. Sportlyzer pruža brojne funkcionalnosti za više tipova korisnika kao što su roditelji, sportaši i treneri dok je u sklopu ovoga diplomskog rada pruženo znatno jednostavnije rješenje namijenjeno samo trenerima kako bi na jednostavan način mogli lakše upravljati različitim grupama sportaša.

3. KORIŠTENE TEHNOLOGIJE I ALATI

Tehnologije korištene prilikom izrade ovog diplomskog rada su: ASP.NET Core, Entity Framework Core, PostgreSQL, React, JavaScript, CSS, HTML, React-Bootstrap. Tijekom upravljanja bazom podataka korišten je alat pgAdmin4, a prilikom testiranja web API-a korišten je Postman. Opis nabrojanih tehnologija i alata dan je u nastavku.

3.1. ASP.NET Core

ASP.NET Core [5] višeplatformski je okvir otvorenoga koda koji se može koristiti prilikom izrade internetskih aplikacija i servisa, IoT aplikacija. Prilikom izrade internetske aplikacije u sklopu ovog diplomskog rada okvir je korišten za izradu web API-a namijenjenog za obradu zahtjeva koji pristižu s internetske stranice. Neke od funkcionalnosti koje omogućuje ASP.NET Core su: ugrađeno ubrizgivanje ovisnosti, modularan i lagan cjevovod zahtjeva, mogućnost stvaranja korisničkog sučelja zajedno uz web API.

3.2. Entity Framework Core

Entity Framework Core [6] korišten je za objektno-relacijsko mapiranje koje omogućuje rukovanje bazom podataka koristeći objekte, te smanjuje potrebu za pisanjem koda za pristup podatcima. Korištenjem Entity Framework Core-a moguće je generirati modele iz postojeće baze, stvoriti modele koji odgovaraju modelima baze te stvoriti bazu prema definiranim modelima. Izmjene nad oblikom baze podataka moguće je izvršavati korištenjem migracija koje omogućavaju primjenjivanje izmjena izvršenih nad modelima podataka na samu bazu podataka.

3.3. PostgreSQL i pgAdmin 4

PostgreSQL [7] je besplatni menadžmentski sustav relacijskih baza podataka otvorenoga koda koji pruža mnoge mogućnosti kao što su automatski ažurirani pogledi, okidači baze podataka, strani ključevi. Također postoje i raznoliki alati koji mogu pomoći prilikom administracije PostreSQL-a kao što je pgAdmin 4 [7] koji predstavlja besplatno korisničko sučelje otvorenoga koda preko kojega je moguće izvršavati upite nad bazom podataka te dohvaćati informacije.

3.4. React

React [8] predstavlja JavaScript biblioteku korištenu prilikom izrade korisničkog sučelja. Radi se o besplatnoj biblioteci otvorenog koda kojoj je glavna zadaća brinuti se o stanjima te prikazivati to stanje pa se prilikom izrada aplikacija obično koriste i dodatne biblioteke kako bi se

postigle željene funkcionalnosti. Prilikom izrade React komponenti, korišten je JSX koji je ekstenzija JavaScript sintakse koja prema strukturi sliči HTML-u.

3.5. JavaScript

JavaScript [9] je programski jezik visoke razine koji predstavlja jednu od temeljnih tehnologija koje se koriste prilikom izrade internetskih stranica. Njime je moguće definirati dodatne funkcionalnosti koje će se dogoditi tijekom interakcije korisnika s internetskom stranicom. Većina internetskih preglednika sadržava JavaScript engine koji se koristi za izvršavanje koda na uređajima korisnika.

3.6. CSS

CSS [10] predstavlja popularnu tehnologiju koja se koristi tijekom izrada internetskih stranica te se koristi za opisivanje načina prezentiranja dokumenta. Pomoću CSS-a moguće je odvojiti način prikaza i sadržaja što može pridonijeti kontroli i fleksibilnosti specifikacije karakteristika prikaza te je moguće korištenje istih stilova na više stranica stvaranjem datoteka u kojima su stilovi opisani.

3.7. HTML

HTML [11] je opisni jezik za stvaranje dokumenata koji se mogu prikazati u internetskom pregledniku. Sastoji se od elemenata koji čine gradivne blokove internetske stranice te kojima je definirana struktura stranice. HTML oznake koje se pišu prilikom dodavanja sadržaja na stranicu koriste šiljaste zagrade te mogu sadržavati druge HTML oznake kao podelemente.

3.8. React-Bootstrap

React-Bootstrap [12] pruža React implementacije Bootstrap komponenti koje su korištene prilikom izrade korisničkoga sučelja. Pošto React-Bootstrap dolazi bez *stylesheet-ova,* dodan je i Bootstrap. Korištenjem React-Bootstrapa olakšana je izrada responzivne aplikacije, te uz dostupne stilove lakše je postići ljepši izgled sučelja.

3.9. Postman

Postman [13] predstavlja alat korišten za testiranje web API-a, koji omogućuje lagano testiranje krajnjih točaka korištenjem različitih oblika HTTP zahtjeva. Moguće je lako izmijeniti sadržaj zaglavlja i tijela zahtjeva, te je prema dobivenim odgovorima olakšan pronalazak i ispravljanje potencijalnih pogrešaka.

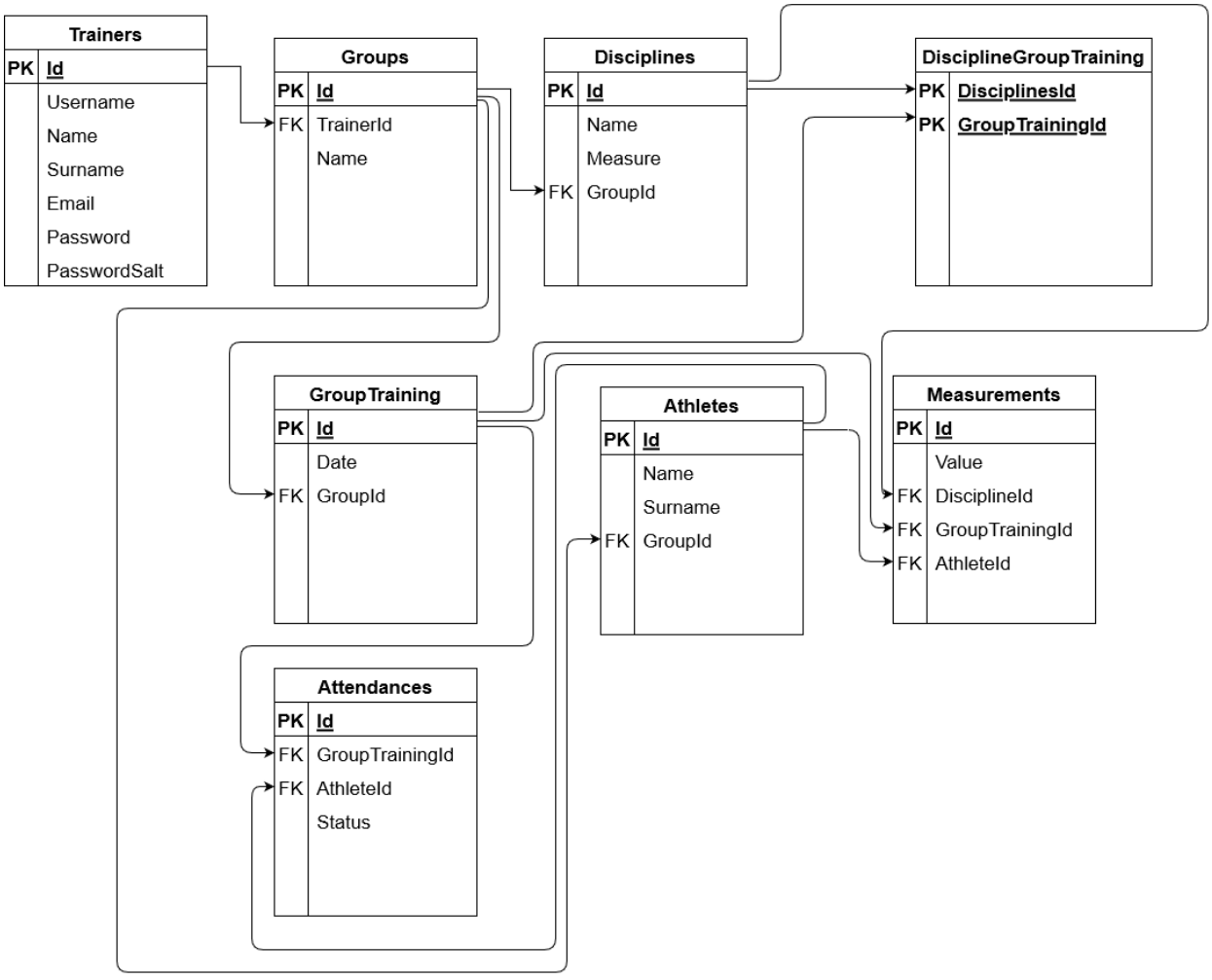
4. IZRADA APLIKACIJE

Unutar ovoga poglavlja opisan je proces izrade internetske aplikacije koji se sastoji od izrade baze podataka i web API-a te korisničkog sučelja.

4.1 Baza podataka

Baza podataka korištena je za spremanje podataka s kojima korisnici rade tokom interakcije s internetskom aplikacijom. Za izradu baze podataka korišten je PostgreSQL te pgAdmin 4 za provjere i pregled stanja baze podataka.

4.1.1. Izgled baze podataka



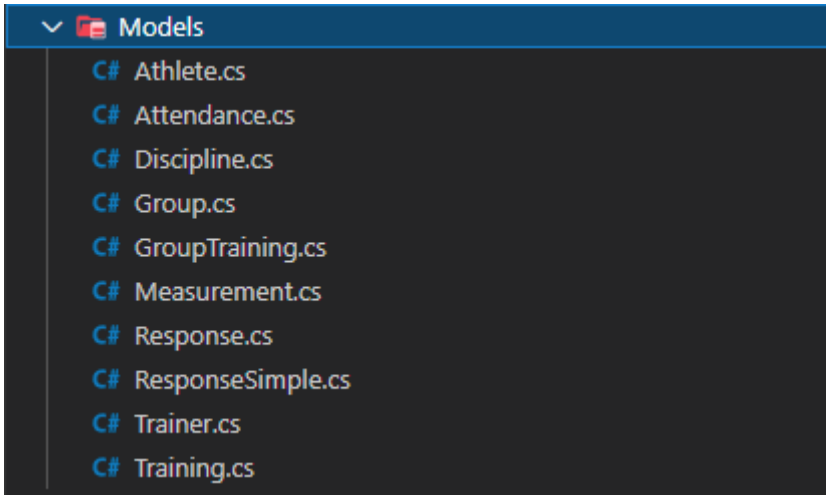
Slika 4.1. Dijagram baze podataka.

Na slici 4.1. vidljiv je dijagram baze podataka te veze kojima su tablice povezane. Tablica *Trainers* sadrži osnovne informacije o treneru, te je vidljivo da postoji i stupac s nazivom *PasswordSalt* koji predstavlja dodatnu mjeru sigurnosti pri skladištenju lozinke trenera. *Groups* sadrži naziv grupe,

te strani ključ koji ga povezuje s trenerom koji stvara grupu. *Disciplines* predstavlja discipline za koje je moguće stvarati mjerenja tijekom treninga, te sadrži naziv discipline, mjernu jedinicu koja se koristi pri zapisivanju vrijednosti te strani ključ koji ju veže uz grupu koja sadrži tu disciplinu. *GroupTraining* se odnosi na trening koji će grupa odraditi u neko određeno vrijeme te sadrži i strani ključ grupe za koju će trening biti održan. *Athletes* sadrži podatke o osobi te strani ključ koji osobu veže uz grupu kojoj pripada. *Measurements* se koristi kako bi se spremila vrijednost postignutoga rezultata jedne discipline tijekom treninga, što znači da je svakome sportašu tijekom treninga stvoren jedan unos unutar *Measurements* tablice za svaku disciplinu koja se mjeri tokom treninga. Tablica sadrži tri strana ključa koji se odnose na disciplinu koja se mjeri, sportaša za kojega se unosi vrijednost i trening, te ostvarenu vrijednost za određenu disciplinu tokom treninga. Unutar tablice *Attendances* bilježi se prisutnost sportaša na određenom treningu, te sadrži strani ključ na sportaša i trening za kojega se prisutnost sportaša unosi. *DisciplineGroupTraining* služi kako bi za svaki trening bilo određeno koje će se discipline mjeriti tokom treninga te se primarni ključ, za razliku od svih ostalih tablica koje imaju zaseban primarni ključ, sastoji od dva strana ključa koji se odnose na disciplinu koja se mjeri te trening tijekom kojega se mjerenje izvodi.

4.1.2. Izrada baze podataka

Baza podataka opisana je unutar web API-a te je izrađena *code first* pristupom. Time se stvaranje tablica i izmjene nad shemom baze izvode izmjenjivanjem koda koji opisuje modele baze podataka te stvaranjem migracija i ažuriranjem stanja baze.



Slika 4.2. Prikaz modela.

Modeli sa slike 4.2. definiraju modele podataka koji se koriste pri izradi baze podataka i rukovanju informacijama prilikom komuniciranja s bazom podataka, te sadržavaju opis podataka koji se skladište kao i veze i restrikcije vezane uz pojedini podatak. Direktorij također sadrži i dodatne

datoteke poput *Response.cs* koja sadrži klasu koja predstavlja standardizirani oblik odgovora kojega korisnici primaju.

```
using System.ComponentModel.DataAnnotations;

namespace application.Models
{
    9 references
    public class Attendance
    {
        0 references
        public int Id { get; set; }
        [Required]
        6 references
        public int AthleteId { get; set; }
        3 references
        public Athlete Athlete { get; set; }
        [Required]
        3 references
        public int GroupTrainingId { get; set; }
        2 references
        public GroupTraining GroupTraining { get; set; }
        [Required]
        5 references
        public bool Status { get; set; }
    }
}
```

Slika 4.3. Model prisutnosti.

Na slici 4.3. klasom *Attendance* predstavljen je model pomoću kojega se stvara tablica koja skladišti podatke o prisutnosti sportaša. Klasa se sastoji od svojstva različitih tipova podataka, te ta svojstva predstavljaju stupce unutar tablice. Korištene su i anotacije poput *Required* koje naglašavaju da su navedena svojstva obavezna, te se pri stvaranju baze podataka takvi stupci označavaju kao stupci koji ne mogu ostati prazni pri novome unosu. Svojstva unutar klase s tipovima podataka *Athlete* i *GroupTraining* su dodana kako bi Entity Framework Core razumio o kakvome tipu veze se radi između klasa, te se pripadajući ključevi automatski smatraju stranim ključevima unutar tablice. Ovakav oblik stvaranja baze podataka se naziva *code first* pristupom gdje se na osnovu modela kao što je model sa slike 4.3. omogućuje stvaranje baze podataka. Na sličan način kao na slici 4.3. stvoreni su i ostali modeli. Entity Framework Core omogućuje objektno relacijsko mapiranje te značajno doprinosi lakšem rukovanju s podatcima.

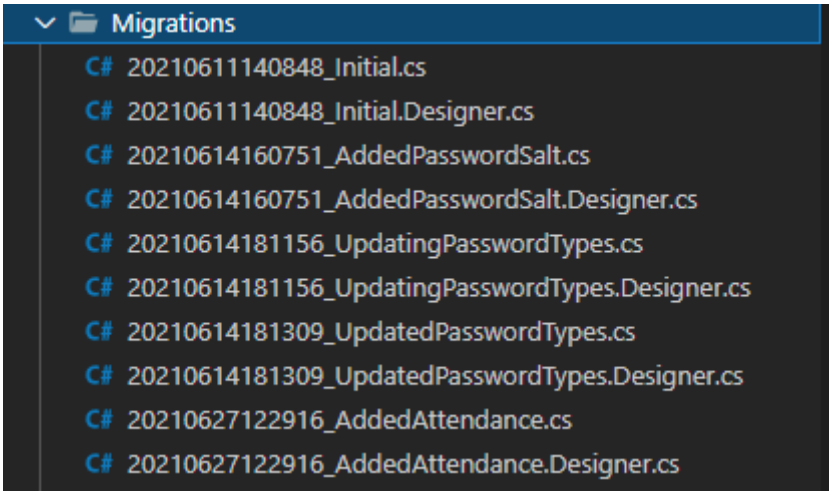
```
using application.Models;
using Microsoft.EntityFrameworkCore;

namespace application.Data
{
    public class DatabaseContext : DbContext
    {
        public DatabaseContext(DbContextOptions<DatabaseContext> options) : base(options)
        {

        }
        public DbSet<Trainer> Trainers { get; set; }
        public DbSet<Group> Groups { get; set; }
        public DbSet<Discipline> Disciplines { get; set; }
        public DbSet<Athlete> Athletes { get; set; }
        public DbSet<Measurement> Measurements { get; set; }
        public DbSet<GroupTraining> GroupTraining { get; set; }
        public DbSet<Attendance> Attendances { get; set;}
    }
}
```

Slika 4.4. Kontekst baze podataka.

Definiranje komunikacije s bazom podataka ostvareno je pomoću *DatabaseContext* klase, vidljive na slici 4.4., koja nasljeđuje *DbContext* klasu, čija instanca predstavlja vezu s bazom te se koristi za slanje upita. Za svaku tablicu postoji *DbSet* svojstvo s odgovarajućim modelom, preko kojeg je moguće izvoditi upite nad željenom tablicom.



Slika 4.5. Primjer migracija.

Jednu od važnih funkcionalnosti pruženih od Entity Famework Core-a predstavljaju migracije, koje omogućavaju laganu izmjenu sheme baze podataka. Na slici 4.5. vidljive su neke od migracija stvorene tijekom razvoja, kojima je postupno izmijenjena i baza podataka. Nakon stvaranja migracije korištenjem „*dotnet ef migrations add naziv_migracije*“ stvorene su datoteke unutar

kojih se nalazi kod zaslužan za izmjene te je moguće primijeniti izmjene nad bazom korištenjem „*dotnet ef database update*“. Povijest migracija se zapisuje i u bazu podataka, te je automatski stvorena tablica s nazivom *__EFMigrationsHistory*.

4.2 Web API

Web API stvoren je korištenjem naredbe „*dotnet new webapi“* koji predstavlja ASP.NET Core web API template. Njime su pružene krajnje točke na koje pristižu zahtjevi klijenata koji rukuju podatcima s dostupnoga sučelja.

4.2.1. Kontroleri i sloj servisa

Kontroler prikazan na slici 4.6. unutar kojega su opisane krajnje točke vezane uz trenere sastoji se od klase *TrainerController* koja nasljeđuje *ControllerBase* klasu koja predstavlja jednu od osnovnih klasa za kontrolere, te su joj dodijeljeni atributi *ApiController* te *Route* kojima je određeno kako će se pristupati krajnjim točkama opisanim unutar kontrolera. Korištenjem *HttpPost* atributa, definiran je tip zahtjeva koji se očekuje uz određenu rutu.

```
namespace application.Controllers
{
    [ApiController]
    [Route("[controller]")]
    0 references
    public class TrainerController : ControllerBase
    {
        3 references
        private readonly ITrainerService _trainerService;
        0 references
        public TrainerController(ITrainerService trainerService)
        {
            _trainerService = trainerService;
        }
        [HttpPost("login")]
        0 references
        public ActionResult<Response<TrainerDto>> Login(LoginDto loginDto)
        {
            Response<TrainerDto> response =_trainerService.Login(loginDto.Username, loginDto.Password);
            if (response.Success)
            {
                return Ok(response);
            }
            return BadRequest(response);
        }
```

Slika 4.6. Trainer kontroler.

Kroz konstruktor je također primljen i objekt preko kojega se izvodi obrada zaprimljenog zahtjeva te komuniciranje s bazom. Pozivanjem metode *Login* navedenog objekta, prosljeđuju se korisničko ime i lozinka, te se u ovisnosti o vraćenom rezultatu korisnicima vraća odgovarajući oblik odgovora.

Tablica 4.1. Prikaz krajnjih točki.

Kontroler	Krajnja točka	Tip zahtjeva
Athlete	/Athlete	POST
	/Athlete/{id}	DELETE
	/Athlete/{id}	GET
	/Athlete/statistics/{id}	GET
Discipline	/Discipline	POST
	/Discipline/{id}	DELETE
	/Discipline/{id}	GET
	/Discipline/training/{id}	GET
Group	/Group	GET
	/Group	POST
	/Group/{id}	DELETE
GroupTraining	/GroupTraining/{id}	GET
	/GroupTraining/{id}	POST
	/GroupTraining/{id}	DELETE
	/GroupTraining/list/{id}	GET
	/GroupTraining	POST
Trainer	/Trainer/login	POST
	/Trainer/register	POST
	/Trainer	GET

Krajnje točke web API-a prikazane su u tablici 4.1., uz koje su vidljivi i tipovi HTTP zahtjeva koji se koriste uz određeni zahtjev prema krajnjim točkama. Svaki kontroler prikazan unutar tablice sadržava više krajnjih točki, te su vitičastim zagradama naznačene vrijednosti unutar ruta koje će biti ispunjene informacijama vezanim uz zahtjev. Također je vidljivo da krajnje točke mogu imati iste rute, no razlikuju se prema tipu zahtjeva.

Objekt koji kontroler prima kroz konstruktor predstavlja dodatni sloj aplikacije unutar kojega je definirana logika rukovanja zahtjevima. Korištenje takve strukture prebacuje dio odgovornosti iz kontrolera te pomaže pri održavanju kontrolera jednostavnijima.

```
public Response<List<GetGroupDto>> GetGroups()
{
    var id = GetTrainerId();
    Response<List<GetGroupDto>> response = new Response<List<GetGroupDto>>();
    try
    {
        List<Group> groups = _databaseContext.Groups
            .Where(group => group.TrainerId == id).ToList();
        response.Body = groups.Select(group => _autoMapper.Map<GetGroupDto>(group)).ToList();
        response.Success = true;
    }
    catch
    {
        response.Text = "Request failed.";
        response.Success = false;
    }
    return response;
}
```

Slika 4.7. Prikaz metode iz GroupServices klase.

Metoda *GetGroups* (Slika 4.7.) sadrži logiku vezanu uz dohvaćanje grupa za koje je zadužen trener koji šalje zahtjev. Korištenjem LINQ-a, kojim je omogućeno pisanje upita unutar C#, dohvaćaju se odgovarajuće grupe iz baze podataka. Također, korišten je i AutoMapper [14] koji omogućuje mapiranje objekata u drugi oblik. U prikazanom kodu objekt tipa *Group* mapiran je u *GetGroupDto* kako bi samo potreban dio informacija izvornog objekta bio vraćen korisniku.

4.2.2. Autentifikacija i tokeni

Svaki profil koji predstavlja trenera sadržava lozinku koja je potrebna za pristup profilu, te se lozinke skladište unutar baze podataka i provjeravaju prilikom pokušaja prijave. Pošto je poželjno izbjeći skladištenje lozinki u njihovome izvornom obliku radi sigurnosti, nad lozinkom je potrebno izvršiti *hash* funkciju kojom se izmjenjuje izvorni oblik lozinke prema primijenjenom algoritmu prikazanom na slici 4.8. [15].

```
private byte[] HashPassword(string password, byte[] passwordSalt)
{
    byte[] hashedPassword = KeyDerivation.Pbkdf2(
        password: password,
        prf: KeyDerivationPrf.HMACSHA256,
        iterationCount: 5000,
        numBytesRequested: 64,
        salt: passwordSalt
    );
    return hashedPassword;
}
```

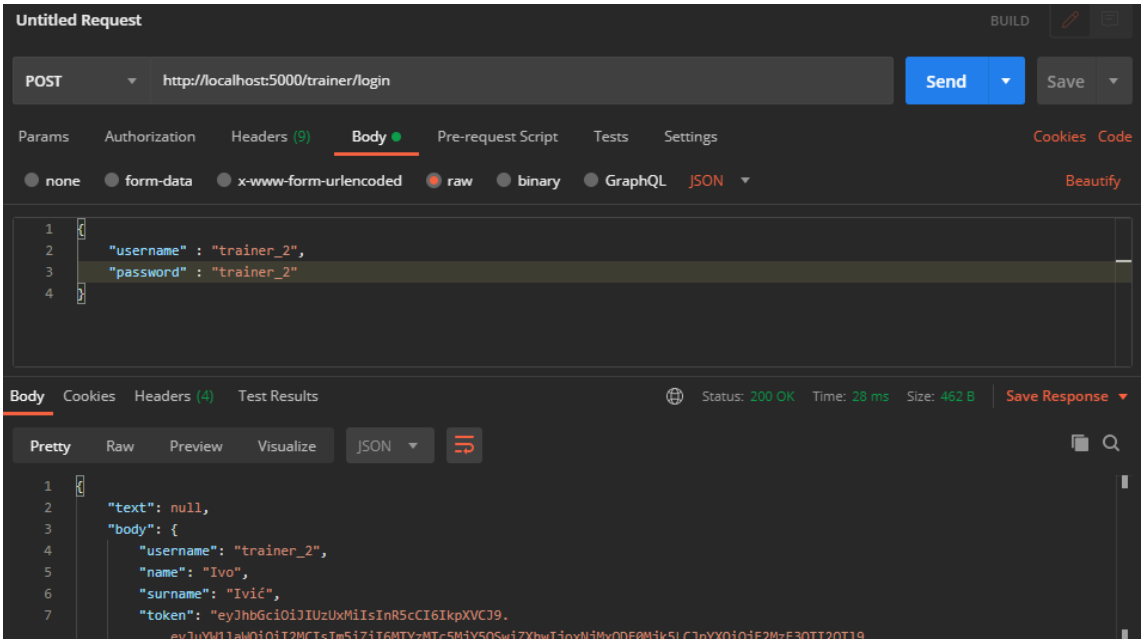
Slika 4.8. Metoda za hashiranje lozinke.

Metoda za *hashiranje* lozinke nalazi se unutar *TrainerService* klase te je korištena pri registraciji trenera, kao i pri prijavi. Kao argumente prima lozinku u obliku *stringa* i sol pod nazivom *passwordSalt* koja se nasumično generira prilikom registracije trenera te je predstavljena u obliku 16 byte-nog polja. Radi se o dodatnoj zaštiti koja omogućava da jednake lozinke zbog različitih vrijednosti soli imaju različite zapise unutar baze. Lozinka je *hashirana* koristeći *KeyDerivation.Pbkdf2* metodu, kojoj je određeno korištenje *HMACSHA256* kao *pseudorandom* funkcije.

Nakon što je registracija uspješno obavljena te se korisnik uspješno ulogirao, korisniku se uz podatke vezane uz profil vraća i token koji se šalje prilikom slanja svih zahtjeva kako bi se omogućila autorizacija zahtjeva.

4.2.3. Testiranje web API-a

Stvoreni web API pruža krajnje točke korisnicima, te ih je korisno testirati kako bi se sa sigurnošću utvrdilo da vraćaju željene oblike odgovora.



Slika 4.9. Testiranje pomoću Postman-a.

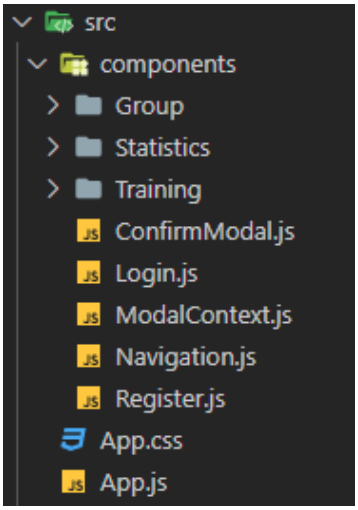
Testiranje prijave na račun prikazano je na slici 4.9. gdje su unutar tijela zahtjeva navedeni korisničko ime i lozinka. Odgovor vraćen od strane web API-a ima status 200 što označava da je zahtjev uspješno izvršen te se na donjem dijelu slike nalaze vraćene vrijednosti. Primljene vrijednosti sadržavaju osnovne podatke o treneru koji se mogu koristiti za prikaz na stranici te token koji se šalje prilikom dohvaćanja informacija vezanih uz određenog korisnika.

4.3 Korisničko sučelje

Korisničko sučelje predstavlja sučelje pomoću kojega korisnici imaju mogućnost interakcije s aplikacijom. Prilikom početka izrade aplikacije, korištenjem naredbe „*npx create-react-app*“ stvoreno je okruženje za razvoj aplikacije koje unutar sebe sadrži React.

4.3.1. Komponente i hook-ovi

React omogućuje raspodjelu korisničkog sučelja na niz komponenti što omogućuje efikasnu distribuciju odgovornosti komponenti te višestruku uporabu. Pri izradi aplikacije komponente su pisane u obliku funkcijskih komponenti koje prema obliku odgovaraju JavaScript funkcijama. Struktura direktorija komponenti stvorenih tijekom razvoja vidljiva je na slici 4.10., te služi kao pomoć pri organizaciji datoteka te lakšem rukovanju tijekom razvoja aplikacije.



Slika 4.10. Prikaz funkcijskih komponenti.

Jednu od važnih funkcionalnosti pruženih od React-a predstavljaju hook-ovi kao što su *useState* koji omogućuje dodavanje stanja funkcijskoj komponenti, *useEffect* kojim je moguće izvoditi određene poslove pri pojavi promjena koje se prate te useContext koji smanjuje potrebu za prosljeđivanjem argumenata kroz niz komponenti.

```
const [input, setInput] = useState({
    name: "",
    measure: "",
    groupId: props.groupId
});
```

Slika 4.11. Primjer useState hook-a.

Primjer *useState* hook-a sa slike 4.11. sadrži stanje informacija o disciplini koja će biti dodana, te se *input* koristi za pristup vrijednostima, dok se za postavljanje stanja poziva funkcija *setInput*.

4.3.2. Navigiranje unutar aplikacije

Navigiranje unutar aplikacije podržano je korištenjem biblioteke React Router [16] pošto je React namijenjen prvobitno za izradu jednostraničnih aplikacija.

```
if (!user.loggedIn)
{
  return (
    <>
      <Switch>
        <Route path = "/register">
          <Register/>
        </Route>
        <Route path = "/">
          <Login handleLogin={handleLogin}/>
        </Route>
      </Switch>
      <ToastContainer/>
    </>
  )
}
```

Slika 4.12. Usmjeravanje korisnika.

Ukoliko trener nije ulogiran, na slici 4.12. unutar *Switch* komponente nalaze se *Route* komponente koje unutar sebe sadržavaju komponente koje predstavljaju forme za registraciju i prijavu. Ovisno o adresi na kojoj se korisnik nalazi, odgovarajući *Route* prikazuje komponentu s formom.

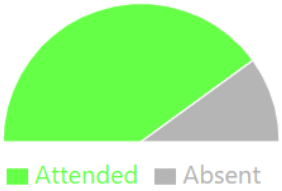
4.3.3. Grafovi

Grafički prikaz prisutnosti sportaša te rezultati postignuti tijekom treninga dostupni su za pregled u obliku grafova koji su izrađeni pomoću biblioteke Recharts [17].

```
<ResponsiveContainer
    minHeight="200px"
    minWidth="120px">
    <PieChart>
        <Pie
            data={attendance}
            dataKey="value"
            startAngle={180}
            endAngle={0}
        >
            {attendance.map((attendance, index) => (
                <Cell key={`cell_${index}`} fill={colors[index]} />
            ))}
        </Pie>
        <Legend wrapperStyle={{marginTop:20}} align='center' verticalAlign='middle'/>
        <Tooltip formatter={(value) => formatTooltip(value)}/>
    </PieChart>
</ResponsiveContainer>
```

Slika 4.13. Kod grafa za prikazivanje prisutnosti sportaša.

Kod prikazan na slici 4.13. predstavlja kod za iscrtavanje grafa prisutnosti, te se sastoji od komponenti iz navedene biblioteke za grafove. Korištenjem *ResponseContainer* komponente omogućena je skalabilnost grafova, te *PieChart* predstavlja graf. Također, vidljivo je da graf sadrži *Legend* i *Tooltip* komponente pomoću kojih se prelaskom preko grafa prikazuje postotak prisutnost ili odsutnosti. Ispod grafa nalazi se legenda koja prikazuje koja boja predstavlja pojedinu vrijednost. Primjer grafa stvorenog korištenjem prikazanog koda vidljiv je na slici 4.14..



Slika 4.14. Graf prisutnosti sportaša.

4.3.4. Dodatne biblioteke

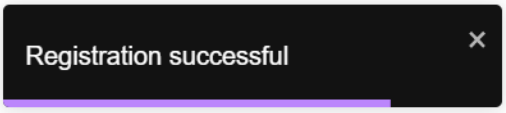
Tijekom izrade korisničkog sučelja, osim do sada navedenih biblioteka dodane su i dodatne biblioteke koje su olakšale proces implementacije željenih funkcionalnosti i izgleda sučelja.

Axios [18] služi kao pomoć pri rukovanju HTTP zahtjevima, te omogućuje funkcionalnosti kao što su presretanje zahtjeva i odgovora. Na slici 4.15. prikazan je kod koji služi za dohvaćanje discipline za određenu grupu.

```
axios.get(`http://localhost:5000/discipline/${param.groupId}`).then(response => {
    setDisciplines(response.data.body);
})
```

Slika 4.15. Dohvaćanje liste disciplina.

React-Toastify [19] korišten je tijekom prikazivanja obavijesti te pruža lagan način za prikaz uspješnosti zahtjeva. Slika 4.16. prikazuje obavijest o uspješnoj registraciji novoga trenera.



Slika 4.16. Obavijest o registraciji korisnika.

Biblioteka date-fns [20] korištena je pri rukovanju datumima, te omogućuje korisne funkcionalnosti kao što su lakše rukovanje postavljanjem i formatiranjem datuma.

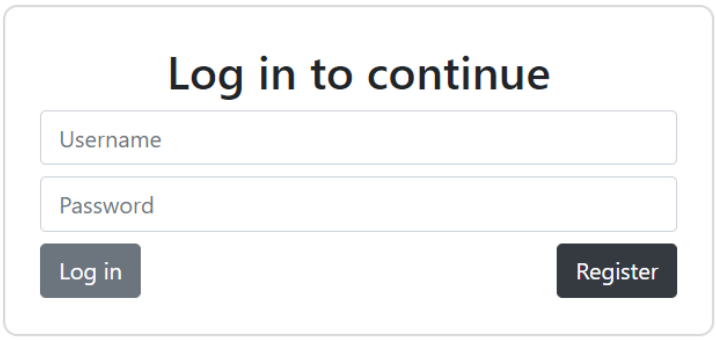
```
    startDate : format(addMonths(Date.now(),-3), 'yyyy-LL-dd'),
    endDate : format(Date.now(), 'yyyy-LL-dd')
```

Slika 4.17. Početni i krajnji datum.

Slika 4.17. prikazuje uporabu navedene biblioteke te su postavljeni i formatirani početni i krajnji datumi koji su postavljeni na razliku od 3 mjeseca.

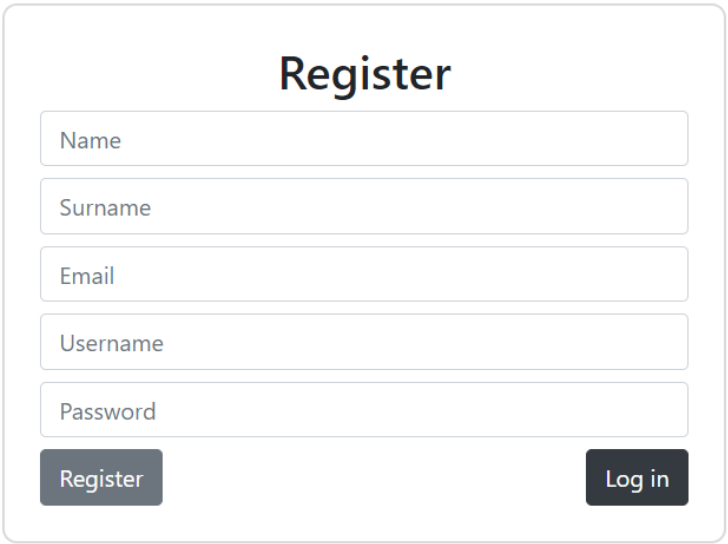
5. IZGLED APLIKACIJE

Prilikom pristupa internetskoj stranici prikazuje se forma za prijavu (Slika 5.1.) pomoću koje se korisnik mora prijaviti kako bi mogao nastaviti dalje.



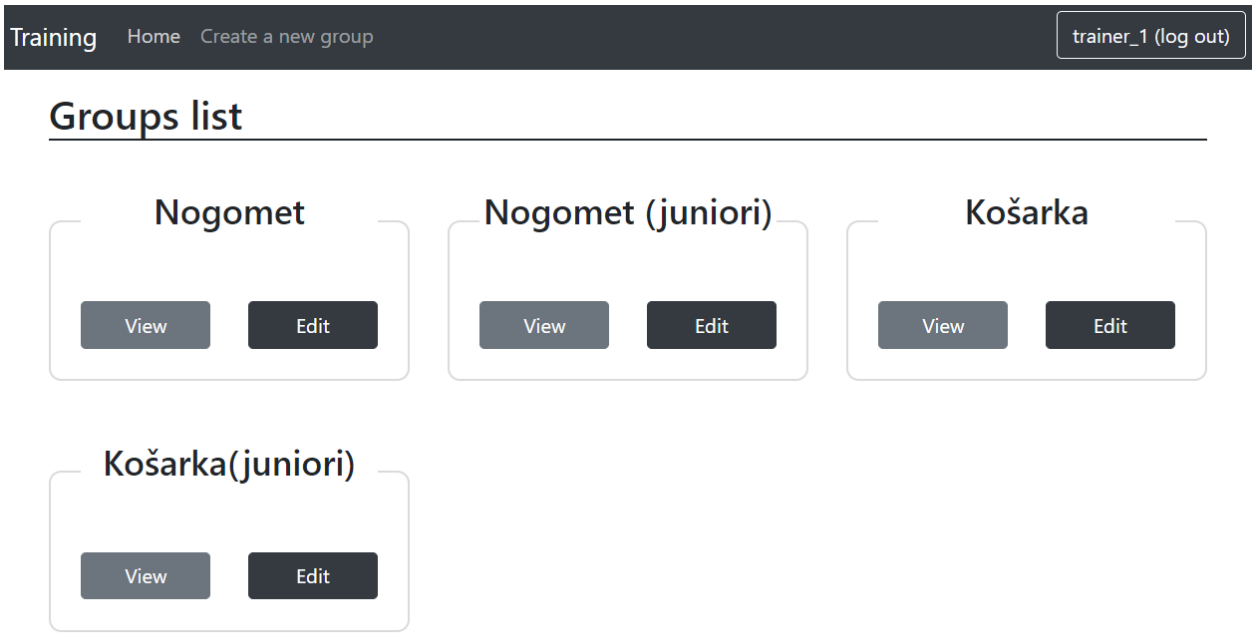
Slika 5.1. Prijava trenera.

Prilikom prijave potrebno je unijeti korisničko ime i lozinku, no ukoliko se radi o novome korisniku s desne strane na slici 5.1. vidljiv je gumb kojim se pristupa registracijskoj formi.



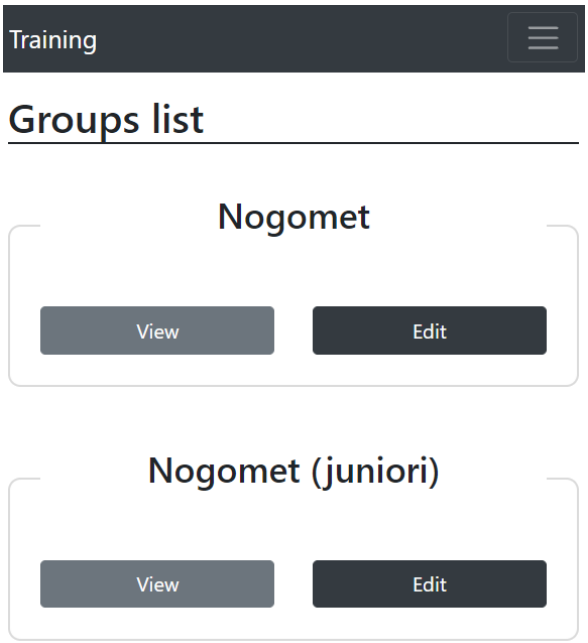
Slika 5.2. Registracija novoga trenera.

Registracijska forma sa slike 5.2. zahtijeva unos osnovnih podataka o treneru, kao i podataka koji će biti korišteni tijekom pristupa računu trenera. Prilikom registracije, sva polja moraju biti ispunjena te ukoliko je forma ispunjena i zahtjev uspješno obrađen, uspješnost zahtjeva se prikazuje u obliku obavijesti. Obavijesti su stvorene pomoću funkcionalnosti dobivene iz React-Toastify biblioteke.



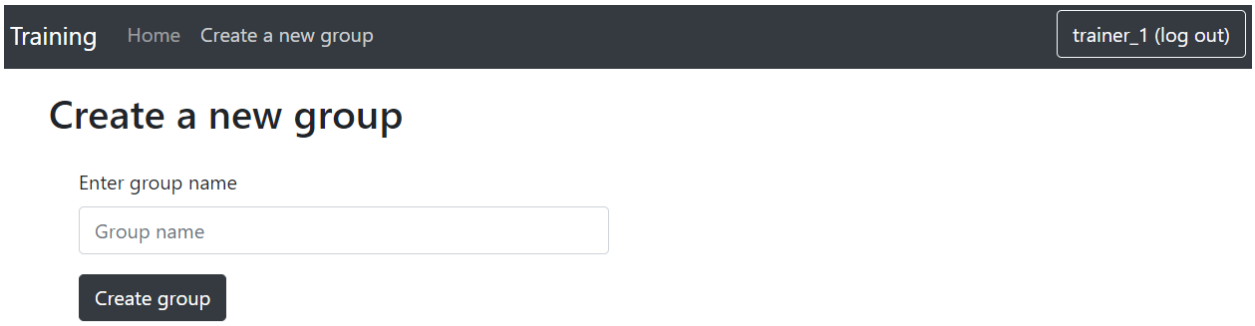
Slika 5.3. Prikaz liste grupa.

Nakon prijave na račun, treneru je prikazana lista grupa koje vodi, te kao što je prema slici 5.3. vidljivo svaka grupa ima dva pripadajuća gumba pomoću kojih je moguće otići na pregled treninga grupe i uređivanje grupe. Također, na vrhu se nalazi navigacijska traka koja omogućuje povratak na početnu stranicu, stvaranje nove grupe te odjavu s računa.



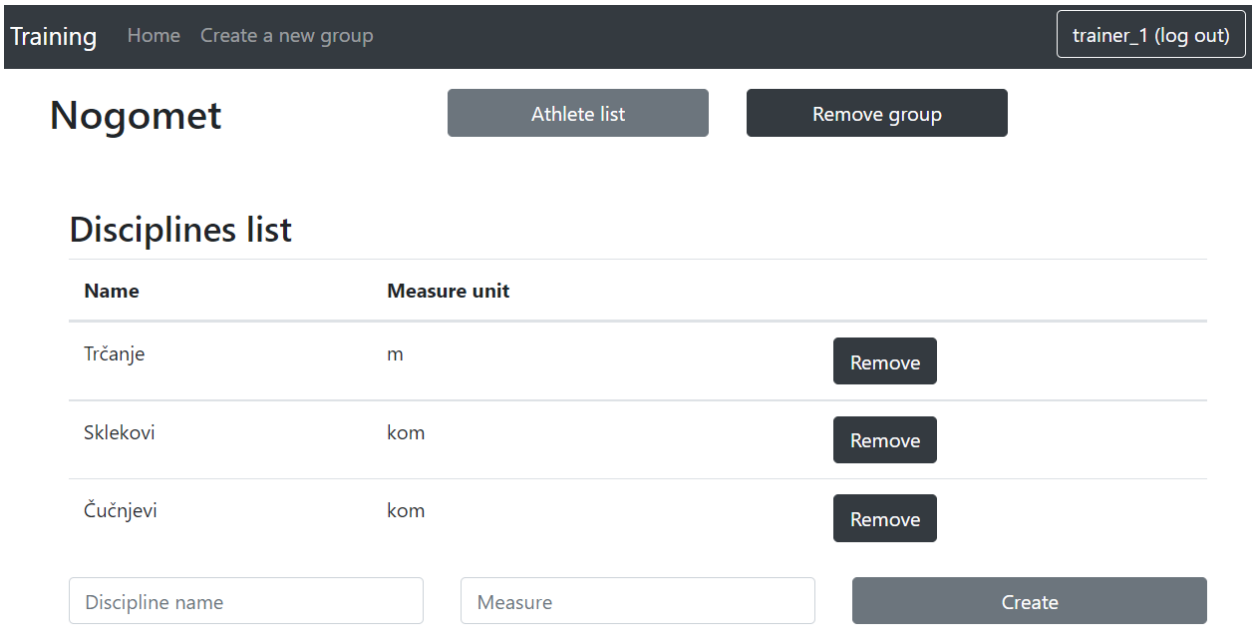
Slika 5.4. Umanjeni prikaz liste grupa.

Sužavanjem preglednika, mijenja se način prikaza elemenata stranice (Slika 5.4.) što čini responzivni dizajn.



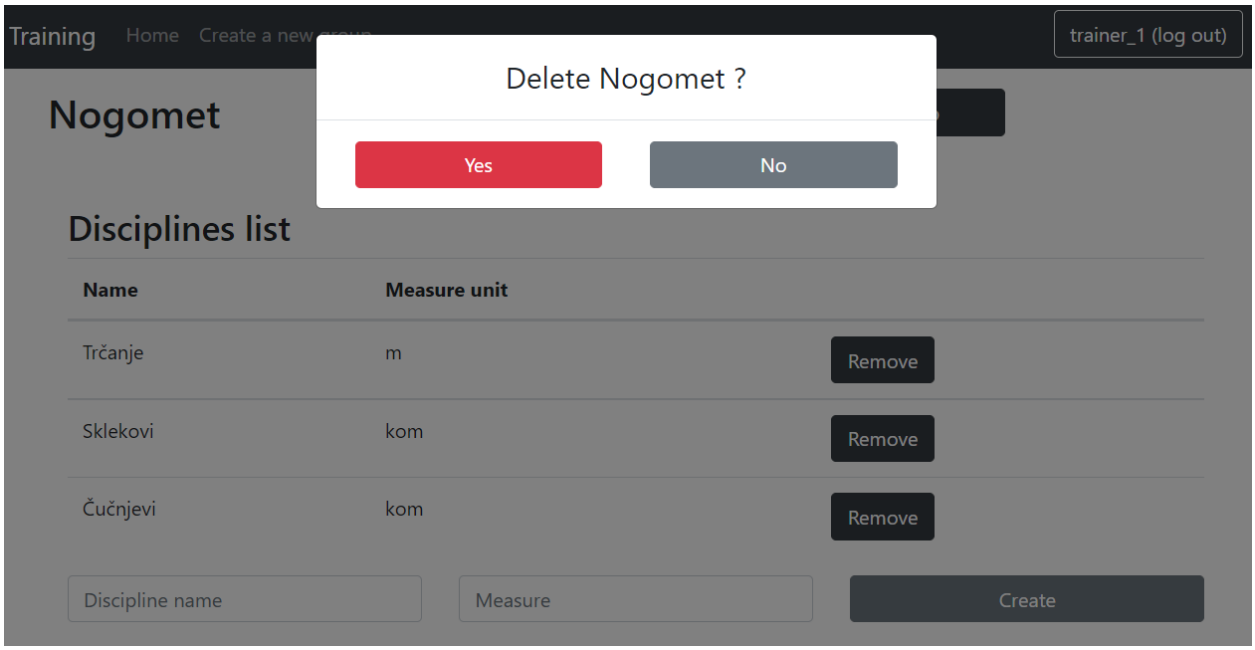
Slika 5.5. Prikaz stvaranja nove grupe.

Klikom na „*Create a new group*“ koji se nalazi na navigacijskoj traci, prikazuje se forma sa slike 5.5. unutar koje se unosi naziv nove grupe koju trener želi stvoriti. Naziv nove grupe mora biti jedinstven u odnosu na druge grupe.



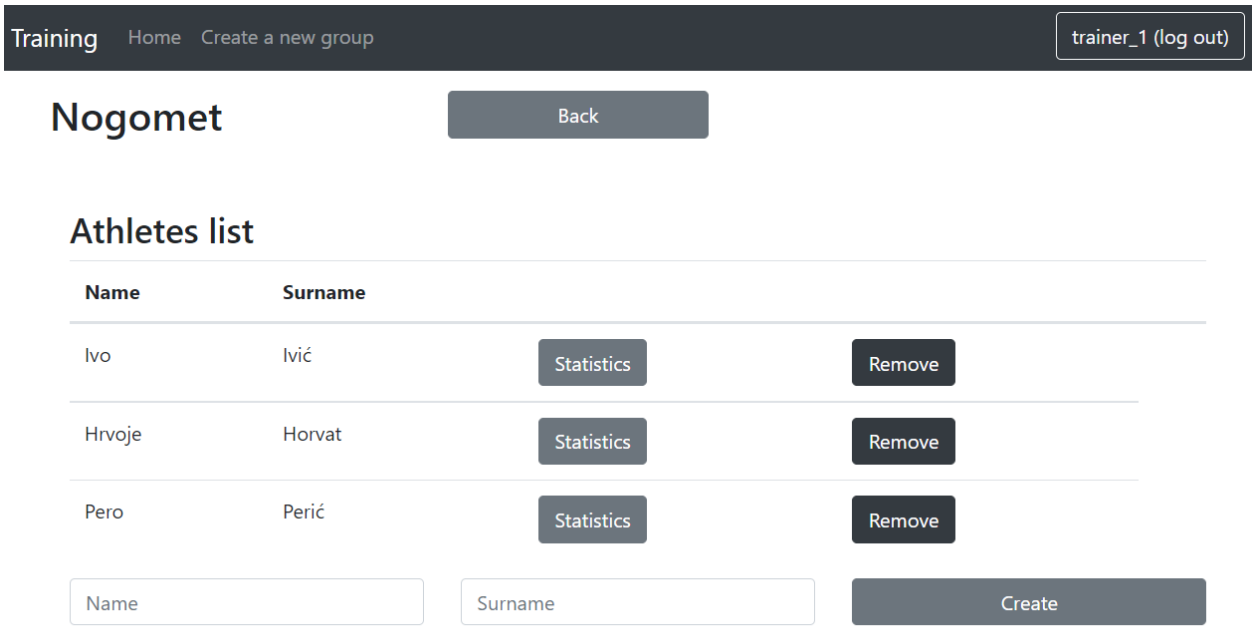
Slika 5.6. Lista disciplina grupe.

Lista disciplina (Slika 5.6.) koje se koriste prilikom zapisivanja rezultata tijekom održavanja treninga prikazuje se klikom na „Edit“ gumb određene grupe (Slika 5.3.). Za svaku disciplinu je naveden naziv i mjerna jedinica, kao i gumb za uklanjanje discipline. Ispod liste disciplina nalazi se forma za dodavanje novih disciplina, te se pri dodavanju novih disciplina naziv discipline treba razlikovati od postojećih naziva. Iznad liste disciplina, dostupni su gumbovi koji vode na listu sportaša koji se nalaze u grupi, te gumb za brisanje grupe.



Slika 5.7. Prozor za potvrdu brisanja.

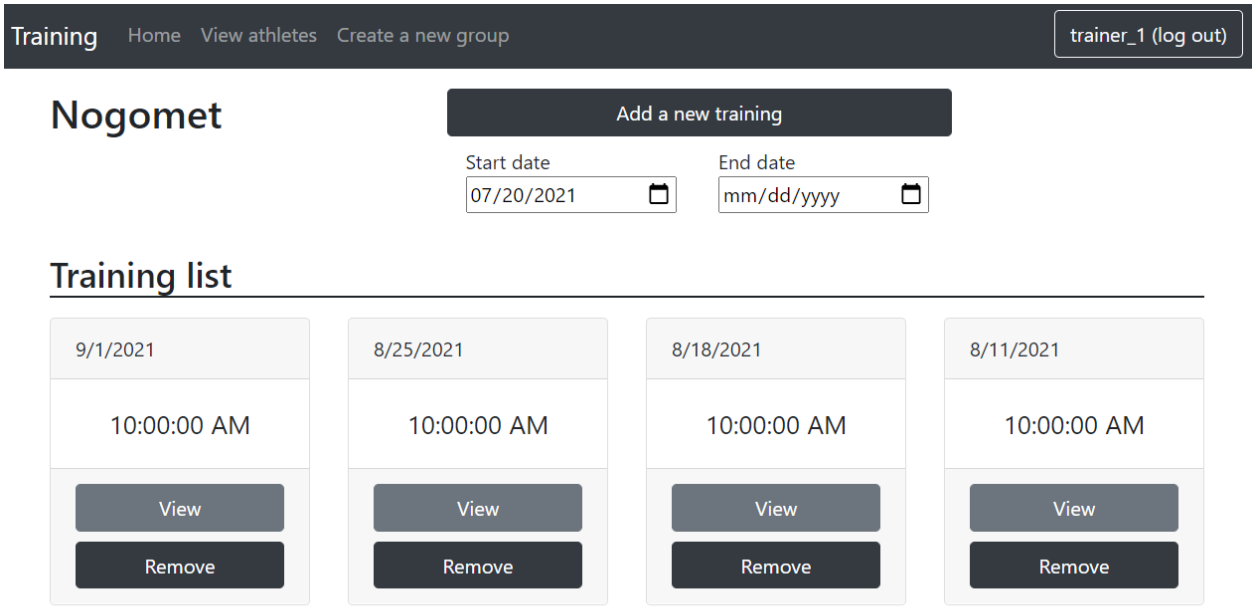
Kod svakog uklanjanja, korisnici moraju potvrditi svoj odabir (Slika 5.7.) kako bi se spriječilo slučajno uklanjanje koje može uzrokovati gubitak podataka.



Slika 5.8. Lista sportaša grupe.

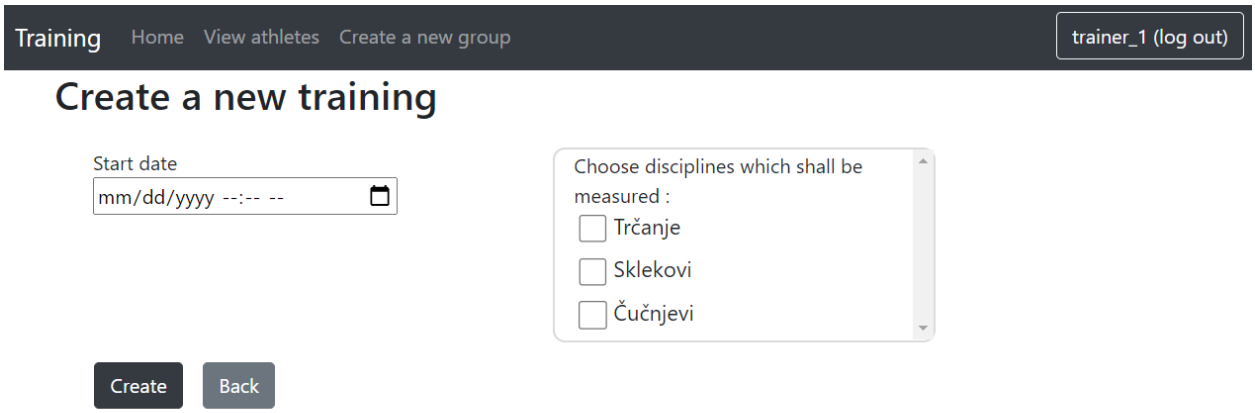
Prilikom pristupa listi sportaša koji se nalaze u određenoj grupi, prikazana je lista sa slike 5.8.. Za svaku osobu na listi prikazani su gumbovi koji vode na statistiku te osobe, te na uklanjanje osobe

s liste. Ispod liste sportaša nalazi se forma kojom je moguće dodati nove osobe, te je pri unosu potrebno unijeti ime i prezime.



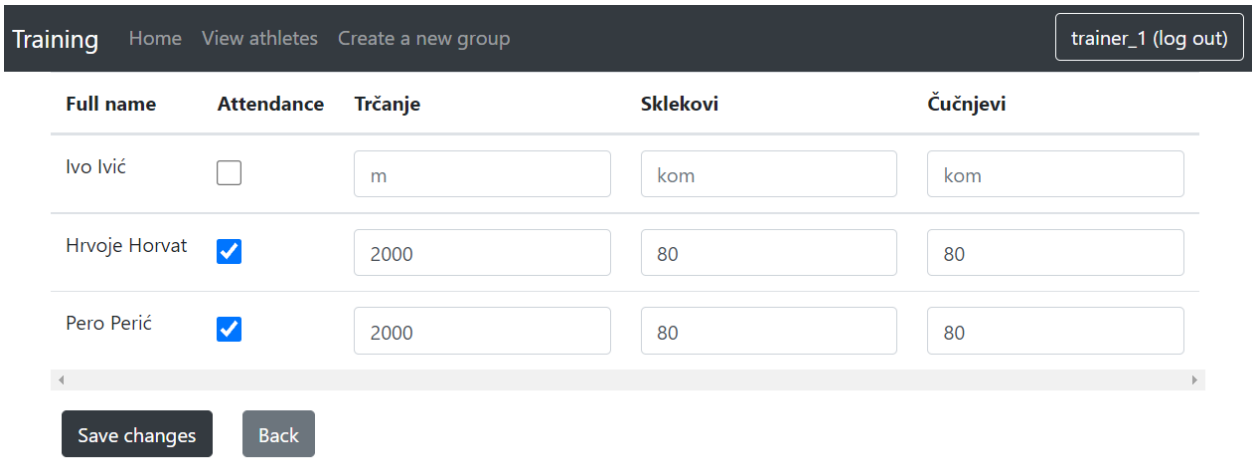
Slika 5.9. Lista treninga grupe.

Lista treninga grupe vidljiva na slici 5.9. prikazuje se klikom na gumb „View“ određene grupe (Slika 5.3.). Za svaki trening naveden je datum i vrijeme treninga, te gumbovi za pregled i uklanjanje treninga. Kako bi bilo moguće filtrirati treninge koji su prikazani, iznad liste treninga nalaze se forme za unos početnog i krajnjeg datuma kojima je moguće urediti raspon prikazanih treninga na listi. Pri stvaranju novog treninga, iznad unosa datuma nalazi se gumb za stvaranje treninga s natpisom „Add a new training“. Također je vidljivo da se pristupom treninzima grupe u navigacijskoj traci pojavljuje poveznica „View athletes“ radi lakšeg pristupa sportašima i statistici.



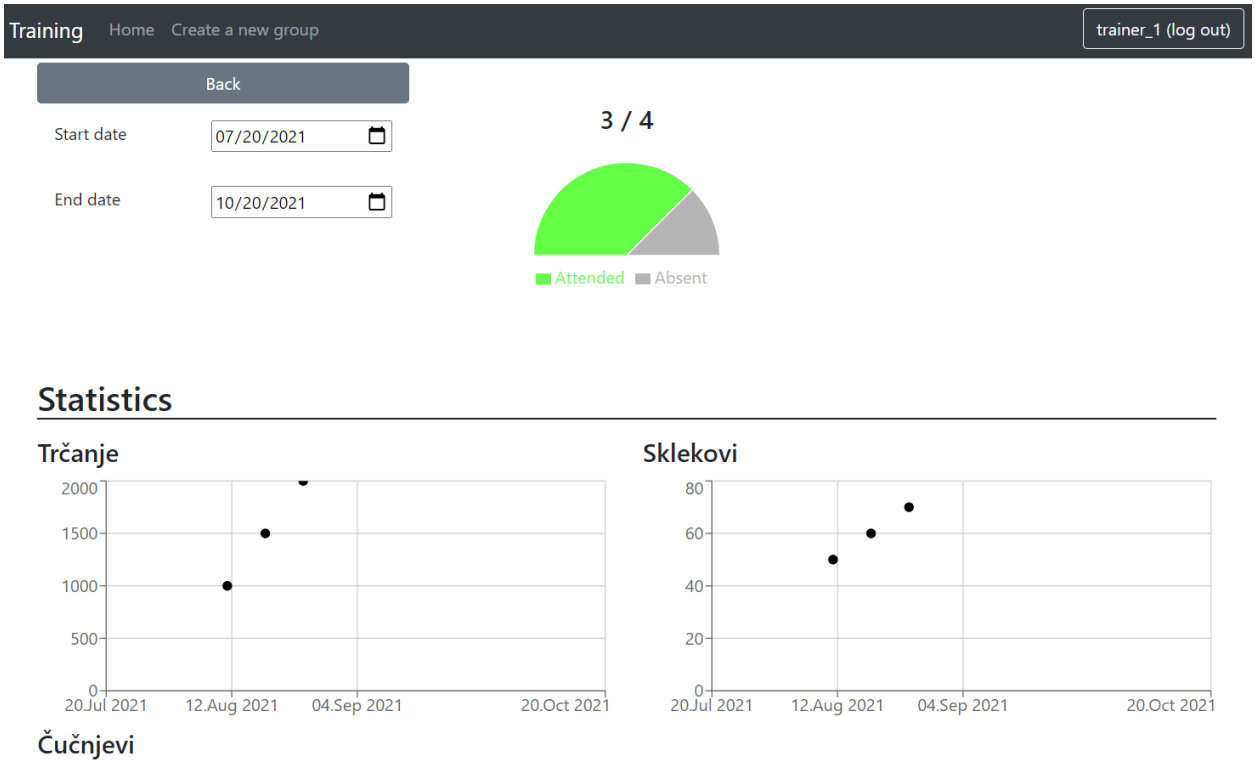
Slika 5.10. Stvaranje novog treninga.

Prilikom stvaranja novog treninga, potrebno je odabrati datum i vrijeme kada će trening biti održan, te kao što je vidljivo na slici 5.10. moguće je odabrati i discipline koje će se mjeriti tijekom treninga. U popisu disciplina navedene su sve discipline koje je korisnik dodao za trenutno odabranu grupu.



Slika 5.11. Uređivanje određenog treninga.

Klikom na pregled treninga unutar neke grupe, svakoj osobi je moguće urediti vrijednosti ostvarene tokom treninga na disciplinama koje su nabrojane tijekom stvaranja treninga (Slika 5.11.). Za svaku osobu je potrebno označiti i prisutnost kako bi bilo moguće spremiti ostvarene vrijednosti.



Slika 5.12. Statistike sportaša.

Statistika ostvarenih rezultata sportaša dostupna je u obliku grafova prikazanih na slici 5.12., na kojoj je za svaku disciplinu prikazan graf s pripadajućim vrijednostima. Raspon vrijednosti prikazanih na grafu moguće je izmjenjivati korištenjem formi za unos početnog i krajnjeg datuma vidljivih u gornjem lijevom kutu. Prisutnost sportaša također je prikazana u obliku grafa gdje je zelenom bojom prikazan postotak prisutnosti, dok siva predstavlja odsutnost. Iznad grafa nalazi se tekst koji prikazuje omjer prisutnosti i ukupnog broja treninga, te je prijelazom preko grafa moguće vidjeti točan postotak prisutnosti.

6. ZAKLJUČAK

U sklopu ovog diplomskog rada izrađena je internetska aplikacija za praćenje statistike sportaša tijekom treninga koja omogućuje trenerima stvaranje korisničkog računa na internetskoj stranici te pristup, praćenje i unos podataka ostvarenih tijekom treninga. Ovakav oblik aplikacije koristan je prilikom vođenja više grupa sportaša gdje nastaje veća količina podataka koji su tijekom vremena kompleksniji za pregled.

Unutar diplomskog rada opisan je postupak izrade aplikacije koji se sastoji od izrade korisničkog sučelja, web API-a i baze podataka. Izgled baze podataka prikazan je u obliku dijagrama, te je programski opisan unutar web API dijela kojim je *code first* pristupom stvorena baza podataka. Web API zaslužan za rukovanje zahtjevima korisnika stvoren je pomoću ASP.NET Core-a te je tijekom razvoja korišten C# programski jezik. Prilikom testiranja web API-a korišten je Postman za slanje zahtjeva na krajnje točke definirane unutar kontrolera. Pri stvaranju korisničkog sučelja korišteni su JavaScript, React te preostale biblioteke i tehnologije koje su olakšale implementaciju željenih funkcionalnosti.

Tijekom izrade rada, kao jedan od ponavljajućih problema predstavljao je neispravan odgovor s web API-a. Zbog takvih slučajeva, bilo je potrebno uložiti dodatan trud tijekom testiranja aplikacije. Kao potencijalne nadogradnje aplikaciji moguće je povećati količinu dostupnih funkcionalnosti na korisničkome sučelju što bi omogućilo pojedinim korisnicima bolju kontrolu i pregled grupa.

LITERATURA

[1] E. Wittendorfer, "Aplikacija za praćenje fitness treninga", Završni rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Osijek, 2017., dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:135605 [21.10.2021.]

[2] J. Faletar, "Izrada i testiranje Angular aplikacije za nogometnu statistiku", Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Osijek, 2020., dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:064142 [21.10.2021.]

[3] I. Šitina, „Web aplikacija za upravljanje teretanom“, Diplomski rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Osijek, 2019., dostupno na: https://urn.nsk.hr/urn:nbn:hr:200:396621 [21.10.2021.]

[4] Sportlyzer, dostupno na: https://www.sportlyzer.com/ [21.10.2021.]

[5] ASP.NET Core, dostupno na: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0 [21.10.2021.]

[6] Entity Framework Core, dostupno na: https://docs.microsoft.com/en-us/ef/core/ [21.10.2021.]

[7] PostgreSQL, dostupno na: https://en.wikipedia.org/wiki/PostgreSQL [21.10.2021.]

[8] React, dostupno na: https://en.wikipedia.org/wiki/React_(JavaScript_library) [21.10.2021.]

[9] JavaScript, dostupno na: https://en.wikipedia.org/wiki/JavaScript [21.10.2021.]

[10] CSS, dostupno na: https://en.wikipedia.org/wiki/CSS [21.10.2021.]

[11] HTML, dostupno na: https://en.wikipedia.org/wiki/HTML [21.10.2021.]

[12] React-Bootstrap, dostupno na: https://react-bootstrap-v4.netlify.app/ [21.10.2021.]

[13] Postman, dostupno na: https://www.postman.com/ [21.10.2021.]

[14] AutoMapper, dostupno na: https://automapper.org/ [21.10.2021.]

[15] ASP.NET Core dokumentacija, dostupno na: https://docs.microsoft.com/en-us/aspnet/core/security/data-protection/consumer-apis/password-hashing?view=aspnetcore-5.0 [21.10.2021.]

[16] React Router, dostupno na: https://reactrouter.com/web/guides/quick-start [25.10.2021.]

[17] Recharts, dostupno na: https://recharts.org/ [21.10.2021.]

[18] Axios, dostupno na: https://axios-http.com/docs/intro [25.10.2021.]

[19] React-Toastify, dostupno na: https://fkhadra.github.io/react-toastify/introduction [25.10.2021.]

[20] date-fns, dostupno na: https://date-fns.org/docs/Getting-Started [25.10.2021.]

SAŽETAK

Tema diplomskog rada izrada je internetske aplikacije koja služi trenerima kao pomoć prilikom vođenja različitih grupa sportaša. Unošenjem rezultata ostvarenih tijekom treninga, statistika je dostupna u obliku grafova što omogućuje lakše praćenje progresa osoba. Prilikom izrade aplikacije stvoren je web API korištenjem ASP.NET Core-a koji obrađuje zahtjeve korisnika te komunicira s bazom podataka. Korisničko sučelje temelji se na JavaScript biblioteci React koja je trenutno jedna od najpopularnijih tehnologija za izradu korisničkih sučelja. Uz React, korištene su i dodatne biblioteke kojima su omogućene funkcionalnosti kao što su responzivnost stranice te grafovi.

Ključne riječi: ASP.NET Core, React, statistika sportskog treninga, JavaScript

ABSTRACT

Internet application for training statistics tracking

Master's thesis theme is about the development of an internet application which serves trainers as an aid in leading different groups of athletes. By entering results obtained during the training, statistics are available in the form of graphs which makes it easier to track progress of a person. During the application development, web API was created using ASP.NET Core which processes client requests and communicates with the database. User interface is based on a JavaScript library React which is currently one of the most popular technologies for creating user interfaces. Alongside React, additional libraries were used that enabled functionalities such as page responsiveness and graphs.

Key words: ASP.NET Core, React, sports training statistics, JavaScript

ŽIVOTOPIS

Predrag Duvnjak rođen je 23.7.1997. u Našicama. Nakon završetka osnovne škole upisuje prirodoslovnu matematičku gimnaziju u Srednjoj školi Isidora Kršnjavoga u Našicama. Nakon toga 2016. upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Preddiplomski studij uspješno dovršava 2019. te dobiva titulu sveučilišni prvostupnik inženjer računarstva. Iste godine upisuje diplomski studij računarstva smjera Informacijske i podatkovne znanosti. Tijekom prve godine diplomskog studija pridružuje se IAESTE-u.

Vlastoručni potpis

_____