# RAZVOJ WEB APLIKACIJE ZA PRAĆENJE NAPRETKA U FITNESSU IZMEĐU TRENERA I KLIJENTA

**Pintarić, Leon**

**VISOKO UČILIŠTE ALGEBRA**

ZAVRŠNI RAD

# Razvoj web aplikacije za praćenje napretka u fitnessu između trenera i klijenta

Leon Pintarić

Zagreb, veljača 2023.

# Predgovor

Ovaj rad posvećujem mojoj baki čija je želja bila da završim fakultet jednoga dana i koja mi je bila motivacija u trenutcima kad je bilo teško. Također ovim putem bih se želio zahvaliti svojoj obitelji, prijateljima i profesorima koji su me podržavali tijekom pisanja ovog rada. Posebno se želim zahvaliti svojem mentoru Aleksanderu Radovanu koji me vodio kroz ovaj rad te uvijek bio dostupan i odgovarao na svaki moj upit.

**Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi**

# Sažetak

Motivacija za odabir ove teme je povezanost programskog inženjerstva i trenutnog hobija. Također jedan od motiva je potreba za takvim načinom rada koji se pojavio tijekom pandemije. Glavni cilj je izrada web aplikacije koja će optimizirati i omogućiti online treniranje s fitness trenerom. Izrađena je MVC aplikacija koja omogućuje treneru registraciju svojih novih korisnika. U aplikaciju se ulazi prijavom. Trener će vidjeti sve svoje klijente te će za svakoga moći izraditi i pregledati personalizirane treninge, vježbe s nazivima, serijama, ponavljanjima, kilažama i izvedbom na treningu. Uz trening, trener će imati mogućnost izrade i pregled plana prehrane. Web aplikacija također omogućuje klijentu pregled treninga i vježbi koje mu je trener zadao za naredni tjedan te po kojima će trenirati. Klijent ima mogućnost uvida u svoj plan prehrane, koji uključuje složene jelovnike za sva tri obroka u danu. Uz plan prehrane klijent vidi izračunate dnevne kalorije s dnevnim makronutrijentima. Na svom računu klijent može pregledati izvedbu svake vježbe kroz video materijal te pratiti napredak u obliku grafičkog prikaza. Fizioterapeut kroz istu aplikaciju može određenom klijentu pružati podršku u obliku savjeta koje klijent može vidjeti na svojem računu. Komunikacija između baze podataka i web aplikacije odvijat će se preko ASP.NET Core Web programskog sučelja – API.

**Ključne riječi:** web aplikacija, fitness, praćenje treninga, MVC, Programsko sučelje API, baza podataka.

# Summary

The motivation for choosing this topic is the connection between software engineering and the current hobby. Also, one of the motives is the need for such a way of working, which appeared during the pandemic. The main goal is to create a web application that will optimize and enable online training with a fitness trainer. An MVC application will be built that allows the trainer to register new users. The application is accessed by logging in. The trainer will see all his clients and will be able to create and review personalized workouts for each, exercises with names, sets, repetitions, weight and performance in the training. In addition to the training, the trainer will have the opportunity to create and review the diet plan. The web application also allows the client to review his training and exercises that the trainer has compiled for him for the following week and which the client trains according to. The client has the possibility to view his diet plan in a way to see what he should eat for breakfast, lunch and dinner. With the diet plan, the client sees the calculated daily calories with daily macronutrients. On his account, the client can review the performance of each exercise through video material and monitor the progress in the form of a graphic display. Through the same application, a physiotherapist can provide support to a certain client in the form of advice that the client sees on his account. The communication between the database and the web application will take place through the ASP.NET Core Web Programming Interface - API.

**Ključne riječi:** web application, fitness, training tracking, MVC, Programming Interface API, database.

# Sadržaj

# 1. Uvod

Glavna ideja ovog rada je upoznavanje s najnovijim tehnologijama koje se koriste u razvoju web aplikacija te prikaz dobivenih znanja u obliku fitness web aplikacije za suradnju između trenera i klijenta uz podršku fizioterapeuta.

Web aplikacija je izrađena u C# programskom jeziku i temeljena je na MVC (Model-View-Controller) arhitekturi. Tijekom izrade rada predstavit će se MVC i ASP.NET Core Web programsko sučelje – API arhitektura te ostale biblioteke koje su korištene u aplikaciji. Također kroz rad bit će prikazano međudjelovanje navedenih tehnologija.

Početak rada bit će usmjeren na pregled postojećih rješenja na tržištu te usporedbu navedenih s izrađenom web aplikacijom. Iza toga slijedi pregled potreba osobnih fitness trenera u današnje doba. U nastavku će biti opisane arhitekture i model baze podataka te komunikacija web aplikacije s programskim sučeljem – API i bazom podataka. Naposljetku, bit će opisane funkcionalnosti koje su implementirane unutar aplikacije, a odnose se na dodavanje osobnog plana prehrane te pregled, kreiranje i praćenje treninga trenera i klijenta, pregled vježbi pomoću video materijala, praćenje napretka te podrške fizioterapeuta.

# 2. Pregled postojećih rješenja na tržištu

Na internetu trenutno postoji velika baza različitih aplikativnih rješenja, pa tako postoje i aplikacije koje daju slična rješenja. Neke od trenutno popularnih aplikacija za komunikaciju i praćenje fitness napretka biti će opisane u nastavku. Po završetku, biti će prikazana usporedba aplikativnih rješenja te prednosti i mane u usporedbi s ostalima.

## 2.1. Trainerize aplikacija

Trainerize je trenutno jedna od najpopularnijih aplikacija za online komunikaciju i praćenje treninga između trenera i klijenta. Aplikacija Trainerize omogućuje izradu prilagođenih planova za online klijente. Treneru unutar svoje web aplikacije se pruža opcija da svakom klijentu unutar unaprijed kreiranih biblioteka dodijeli trening, vježbu i cjelokupan program. Također, trener ima mogućnost dodavanja bilješke i uputa s vježbama koje klijenti mogu vidjeti i samostalno pratiti. Nakon izrade plana treninga i vježbi, trener ima mogućnost isporučiti planove obroka (u PDF formatu) ili ponuditi propisane planove obroka. Klijenti mogu pratiti vježbe, prehranu, ciljeve fitnessa, fotografije napretka i više s mobilnom aplikacijom Trainerize na Android ili iOS operacijskom sustavu. Aplikacija radi na engleskom jeziku. Treneru je omogućen broj klijenata kroz pakete, gdje svaki od njih ima određen broj klijenata, a maksimalan broj klijenata iznosi dvjesto. Paketi se plaćaju na mjesečnoj ili godišnjoj bazi, ovisno o paketu. Maksimalni broj klijenata je 200. Pristup klijentima može imati samo jedan trener. Aplikacija je prilagođena za male, srednje i velike kupce. [1]

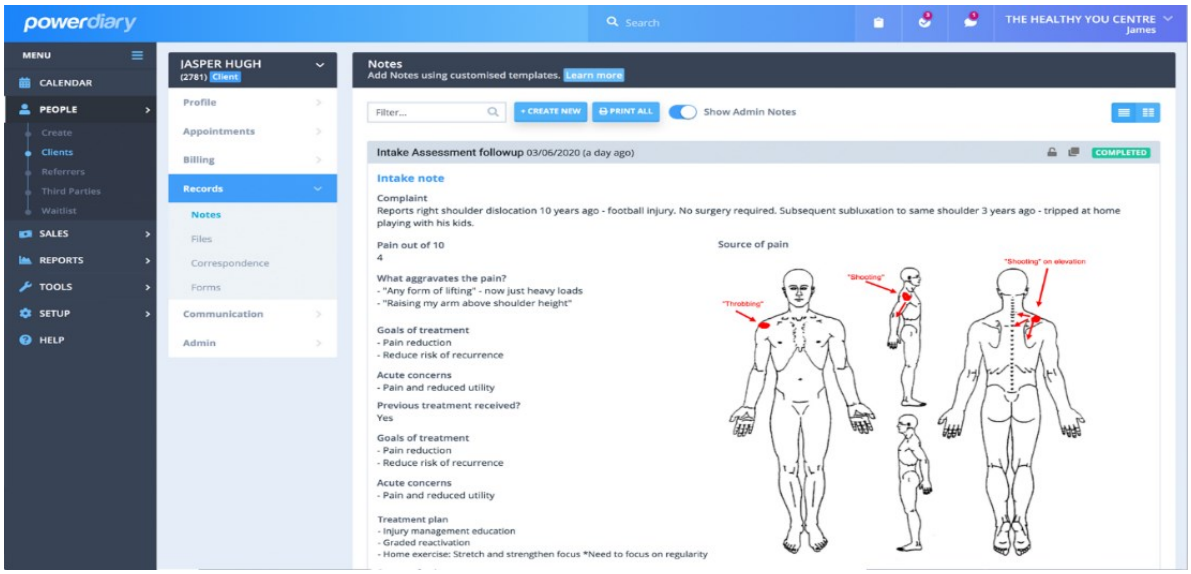
## 2.2. TrueCoach aplikacija

TrueCoach je također aplikacija za online komunikaciju i praćenje treninga između trenera i klijenta. Za razliku od Trainerize aplikacije, ona ima podršku za timski način rada, odnosno u aplikaciji je dostupna opcija da postoji više trenera koji treniraju jednu osobu. TrueCoach nudi dodavanje novih klijenata, isporuku personaliziranih planova vježbanja, praćenje napretka i održavanje komunikacije. Unutar aplikacije moguće je pratiti svoju povijest i napredak treniranja. Pomoću grafičkog prikaza, prikazan je napredak prema određenim ciljevima. Trener je u mogućnosti dodijeliti makronutritivne ciljeve te možete pratiti unos

makronutrijenata svojih klijenata i prikazati njihov napredak. Klijent unutar svoje aplikacije može pregledavati videosnimke svake vježbe. Aplikacija može raditi jedino na engleskom jeziku. TrueCoach je prilagođena za manje i srednje klijente, gdje je trenerima također broj klijenta omogućen kroz pakete. Paketi se plaćaju na mjesečnoj ili godišnjoj bazi, a najveći paket nudi do 50 klijenta. [2]

## 2.3. Power Diary aplikacija

Power Diary je drugačija aplikacija od prethodno navedenih, no potrebna je za analizu radi funkcionalnosti koju će imati nova aplikacija. Ova aplikacija ne služi trenerima za kreiranje i praćenje klijenata, već fizioterapeutima za online komunikaciju s klijentima. Unutar aplikacije, fizioterapeut može dodavati klijente i raspored za online razgovore. Pored online razgovora, fizioterapeut može komunicirati s korisnicima putem chat opcije. Za svakog klijenta, fizioterapeut je u mogućnosti ostaviti bilješku ili komentar kao rješenje njegovog problema (prikazano na Slika 2.1). Aplikacija je također prilagođena za male i srednje korisnike. Fizioterapeut može odabrati paket, ovisno količini klijenata, koji se plaćaju na tjednoj bazi. Najveći broj klijenata iznosi 100 ljudi. Aplikaciju može koristi samo jedan fizioterapeut te je sama aplikacija dostupna na engleskom jeziku. [3]




Slika 2.1 Opcija bilješke unutar Power Diary aplikacije [4]

# 3. Potrebe osobnih trenera

Unazad nekoliko godina postalo je sve popularnije online mentorstvo u fitnessu između trenera i klijenta. Cilj ovog rada jest uvođenje automatizacije, što bi omogućilo uštedu vremena i novca za mlade trenere te bolje i lakše korisničko iskustvo za klijente. S obzirom na veliki porast novih fitness trenera, veća je i potreba za primjenjivim i jednostavnijim rješenjima, međutim ona su vrlo skupa. Zbog nedostupnih, već gotovih automatiziranih aplikacija, većina trenera na ovim prostorima koristi e-mail adrese kao kompenzaciju navedenome, kako bi bili u mogućnosti slati treninge, vježbe i plan prehrane preko Excelovih tablica.

## 3.1. Opis web aplikacije

Web aplikacija za osobne trenere je nastala kao potreba tržišta, prema iskustvu autora te analizom postojećih aplikacija na tržištu s ciljem da se automatiziraju i ujedine različite funkcionalnosti u jedinstvenu aplikaciju. Aplikacija je napravljena na jednostavan i intuitivan način sa svim potrebnim opcijama za trenera i klijenta bez nadoplata za sve opcije. Uz programiranje personaliziranih treninga i vježbi, moguće je dodavanje primjera plana prehrane za svaki dan i svaki obrok. Uz primjere plana prehrane ispisane su nutritivne vrijednosti i cjelodnevne kalorije s makronutijentima. Osim osobnog treninga, vježbi i plana prehrane, moguć je i pregled savjeta od fizioterapeuta. Fizioterapeut je u mogućnosti poslati bilješku s odgovorom na problem klijenta.

Na početnoj stranici trenera, treneru je omogućen uvid u sve klijente, gdje su prikazane informacije poput imena, e-mail adrese i datuma kreiranja klijenta. Na alatnoj traci omogućeno je dodavanje novog klijenta, upravljanje s podacima, a klikom na pojedinog klijenta može vidjeti njegove treninge s informacijama, dodati novi trening, pregledati plan prehrane ili dodati novi plan prehrane. Svaki trening sadrži vježbe s nazivom, serijama, ponavljanjima, kilažom te izvedbom na treningu. Klijent prijavom na svoj račun može vidjeti svoje treninge te unutar svakog treninga vježbe. Pritiskom na neku od vježbi klijent vidi graf napretka po datumu i kilaži. Na alatnoj traci klijentu je dostupan pregled plana prehrane, savjeti fizioterapeuta i video materijal s pravilnom izvedbom vježbe. Fizioterapeut putem korisničkog računa može slati savjete klijentima.

## 3.2. Aplikativna podrška

Aplikativna podrška u kategoriji fitness online treniranja nije najveća, no neke od dostupnih aplikacija opisane su u prijašnjoj cjelini. Dvije aplikacije su slične, dok je treća jedinstvena. Aplikacija Trainerize usmjerena je isključivo na online treniranje i prehranu. Trenutno je najpopularnija na tržištu za tu uslugu, a to se očituje brojem registriranih klijenata i trenera. TrueCoach je slična aplikacija, kao i aplikacija Trenerize. Uz standardne funkcionalnosti online treniranja i plana prehrane, TrueCoach posjeduje mogućnost dodavanja još trenera te je tako moguće ostvariti da više trenera trenira jednu osobu. Trenerima je omogućen rad u paru, što rezultira boljim korisničkim iskustvom i efikasnijim radom, kako bi klijent što lakše i brže došao do ciljeva.

U treniranju vrlo često dolazi do ozljeda ili istegnuća, gdje fitness trener katkad nije potpuno stručan za davati savjete i znati ispravno riješiti problem, pa dolazi do potrebe za mišljenjem fizioterapeuta. Trenutno na tržištu su te dvije usluge vrlo povezane. Uz aplikacije za online treniranje i plan prehrane, nešto je drugačija aplikacija Power Diary koja nudi mogućnost fizioterapeutsko savjetovanje i online razgovore s klijentima. Klijent dobiva razne savjete u obliku bilješki. Trenutni nedostaci svake od aplikacija su količina klijenata, cijena, nepovezanost s fizioterapeutom ili nemogućnost treniranja u paru.

Nova aplikacija za online treniranje i komunikaciju uključuje većinu gore navedenih funkcionalnosti i zbog toga što objedinjuje više aplikacija čini je najsadržajnijom što se tiče ponude. Uz opcije za kreiranje i praćenje treninga i prehrane, nudi mogućnost klijentu da u jednoj aplikaciji ima podršku fizioterapeuta i fitness trenera. Dodatno, aplikacija nudi neograničen broj klijenta te više trenera i fizioterapeuta unutar aplikacije. Također uz plan prehrane dolazi primjer prehrane za svaki obrok uz nutritivne vrijednosti za taj obrok i dan.

# 4. Opis arhitekture rješenja

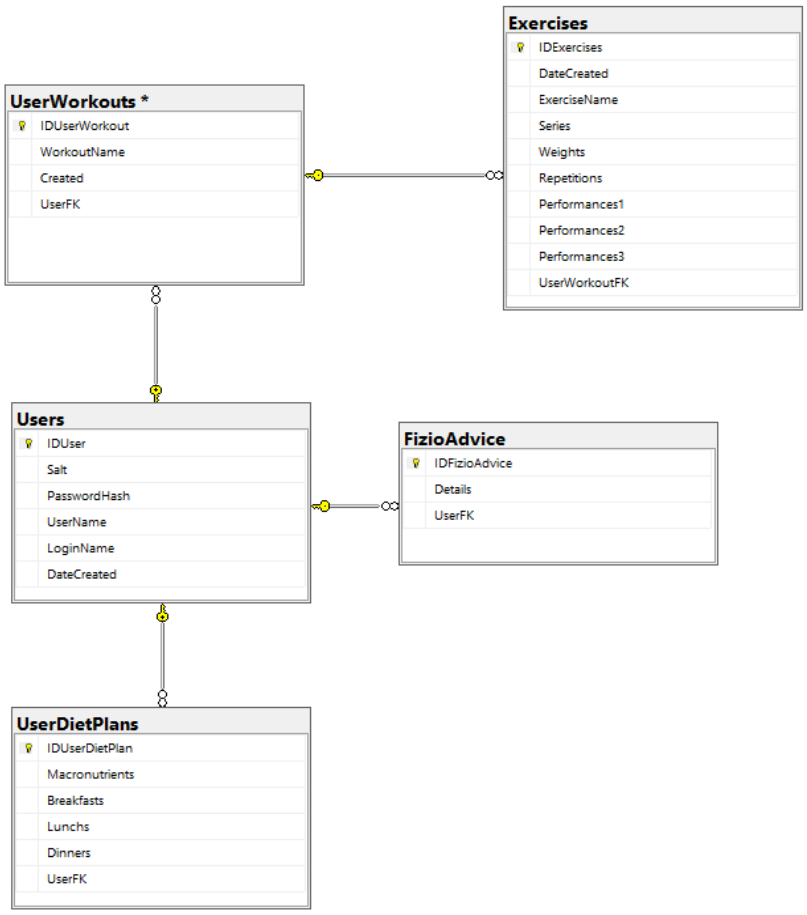
## 4.1. Baza podataka

U cjelini baze podataka bit će definirano što je baza podataka i na primjerima pokazano korištenje baze podataka i njenih funkcija unutar aplikacije, kao glavni izvor podataka koje dohvaća web aplikacija preko programskog sučelja API. Baza podataka je organizirani skup ili kolekcija koja u sebi sadrži podatak ili skup podataka. Tablice su ključni objekt u bazi podataka jer sadrže sve informacije ili podatke unutar stupaca i redaka te mogu i ne moraju biti međusobno povezane. Također, postoje i drugi oblici baza podataka koje ne koriste tablični sustav. Danas u gotovo svim poslovnim područjima su prisutne baze podataka jer je to jedan od najjednostavnijih načina pohrane podataka. Ovisno o bazi podataka u kojoj se radi, koriste se različiti jezici za upravljanje s podacima. Operacije koje se najčešće koriste nad podacima su CRUD operacije, a odnose se na stvaranje (engl. *create*), čitanje (engl. *read*), ažuriranje (engl. *update*) i brisanje (engl. *delete*), no one su neke od operacija i mogućnosti koje se izvršavaju uz pomoć SQL jezika. (Alvaro, 2016) Baza podataka može biti pohranjena lokalno na računalu gdje je kreirana ili na serveru kako bi bila dostupna i ostalim korisnicima.
MVC fitness aplikacija koristit će bazu podataka pohranjenu na internetu te za potrebe testiranja koristit će i lokalnu bazu na računalu. Komunikacija između baze podataka i web aplikacije odvijati će se preko *ASP.NET Core Web* programskog sučelja – API o čemu će biti rečeno u narednoj cjelini rada. [6]

### 4.1.1. Model baze podataka

Za prikazivanje povezanosti između podataka u bazi podataka korišten je ER Model (engl. *Entity Relationship Model*). ER Model pomaže pri sustavnom analiziranju zahtjeva za podacima, kako bi se proizvela dobro dizajnirana baza podataka. ER model predstavlja entitete koji označavaju objekte i odnose između njih. Svrha veza u bazi jest povezivanje entiteta s točnim podacima koji im pripadaju ili se odnose na njih. Primjer: trening i klijent. Prilikom izrade treninga važno je da izrađeni trening bude povezan sa željenim korisnikom. Povezivanje tablice klijenta s tablicom treninga, omogućava povezivanje treninga i osobe, jedan zapis u tablici korisnika bit će povezan s kreiranim treningom. Zbog riječi o

individualnom pristupu trenera i klijenta, jedna osoba može imati jedan ili više treninga, dok jedan trening ne može imati više osoba. Takvu vezu moguće je dobiti povezivanjem entiteta preko ključeva koji mogu biti primarni ili strani. Ovisno o tipu entiteta i tome vezujemo ili ne vezujemo nešto na taj entitet, dodjeljuju se primarni i strani ključevi. Na slici 2. prikazan je ER Dijagram baze podataka koja je korištena za web aplikaciju. Trener prilikom registracije novog klijenta unosi korisničko ime, e-mail adresu, koja je jedinstvena, te lozinku koju klijent unosi prilikom prijave. Unutar web aplikacije postoji više korisnika, pa tako trener ima dodijeljeno ovlaštenje „Trainer“, fizioterapeut ovlaštenje „Physiotherapist“ i klijent ne posjeduje ovlaštenje. Unutar tablice *Users* nalaze se svi navedeni korisnici aplikacije s njihovim korisničkim podacima i ovlaštenjima. Treneri unutar aplikacije imaju mogućnost kreiranja treninga, vježbi i plana prehrane. Tablica *Users* je povezana s tablicom *UserWorkout* s vezom „jedan na više“. Kreirani trening za određenog klijenta sprema se u tablicu *UserWorkout* gdje su upisani ime treninga i datum kad je trening kreiran kako bi se mogao pratiti napredak klijenta iz tjedna u tjedan. U tablicu *Exercises* spremaju se vježbe svakog treninga koje je trener kreirao. Unutar tablice *Exercises* sprema se naziv vježbe, datum kada je vježba kreirana kako bi se mogao pokazati napredak u obliku grafičkog prikaza, broj serija koji je potrebno izvesti, radna kilaža s kojom klijent izvodi vježbu, opseg broja ponavljanja te izvedba na treningu. Jedan trening je također povezan vezom „jedan na više“ jer se jedan trening sastoji od više vježbi. Unutar tablice *UserDietPlans* pohranjeni su makronutrijeti i primjeri prehrane i svaki zapis je povezan s njegovim korisnikom. Također tablica *FizioAdvice* pohranjuje savjete fizioterapeuta i povezana je s klijentom. Opisana baza podataka vidljiva je na Slika 4.1. [7]




Slika 4.1 Prikaz dijagrama baze podataka

### 4.1.2. Model baze podataka na web serveru

Pored lokalne baze podataka, bit će dostupna baza na web serveru (engl. *hosting*). Usluga pohrane podataka na web serveru je pohrana na udaljene poslužitelje kojima se pristupa putem interneta. Prednost takve baze u odnosu na lokalnu jest široka dostupnost, jednostavnost korištenja i dostupna tehnička podrška. Na taj način baza podataka uvijek je dostupna aplikaciji pod uvjetom da aplikacija ima pristup internetu. Komunikacija između baze i web aplikacije odvija se preko programskog sučelja – API.

### 4.1.3. Praktična primjena baze podataka

Baza podataka i programsko sučelje – API povezani su pomoću poveznice *connection string* kako bi API mogao slati ili primati podatke. Za rad s podacima unutar baze podataka zadužene su procedure pomoću kojih koji se provode CRUD operacije. Procedura će primati podatke od programskog sučelja - API, te ovisno o potrebi, stvarati nove vrijednosti, dohvaćati određene vrijednosti ili vršiti izmjene nad postojećim vrijednostima.

```
create proc getWeights
@id int,
@name varchar(50)
AS
begin
     SELECT Exercises.Weights, Exercises.DateCreated
     FROM Exercises
     join UserWorkouts ON Exercises.UserWorkoutFK =
     UserWorkouts.IDUserWorkout
     JOIN Users ON Users.IDUser = UserWorkouts.UserFK
     where Users.IDUser = @id and Exercises.ExerciseName =
     @name
end
```

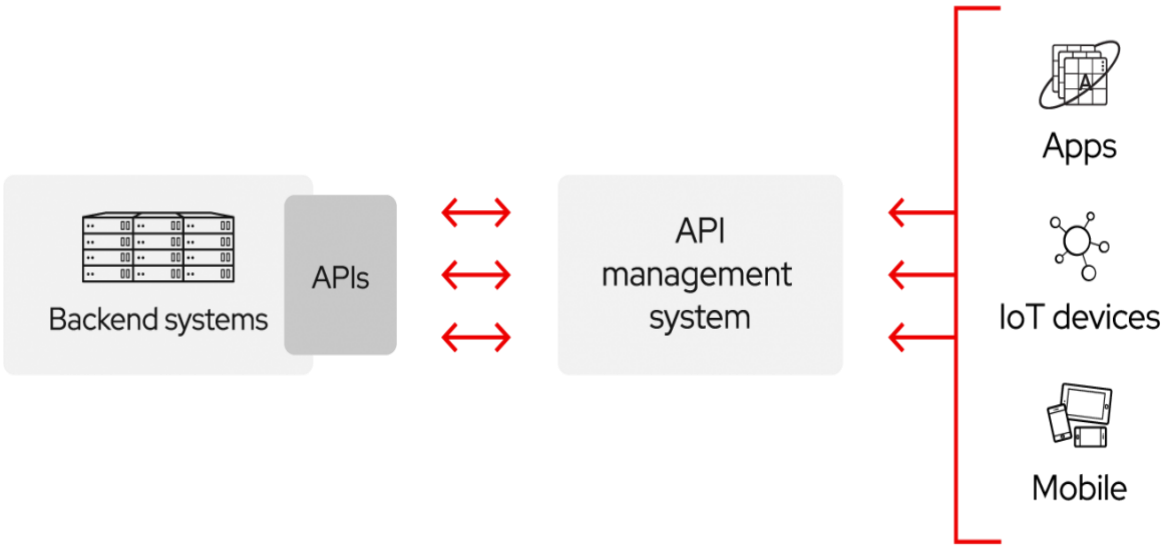
Kôd 4.1 Primjer procedure za dohvat podataka

Kôd 4.1 prikazuje jednu od procedura iz baze podataka. Prikazana procedura zadužena je za dohvat podataka u svrhu grafa napretka. Procedura prima parametre ID korisnika i ime vježbe te vraća kilažu i datum kada je trening odrađen.

## 4.2. Programsko sučelje - API

Programsko sučelje - API (engl. *application programming interface*) omogućuje komunikaciju dva softverska programa. Posrednički sloj koji obrađuje prijenose podataka između sustava. Primjer, softverski sustav meteorološkog zavoda sadrži dnevne vremenske

podatke. Aplikacija za vremensku prognozu na mobilnom uređaju "razgovara" sa sustavom meteorološkog zavoda putem programskog sučelja – API i prikazuje dnevna ažuriranja vremenske prognoze na mobilnom uređaju. Komunikacija se odvija koristeći zahtjeve i odgovore. Arhitektura API-a obično se objašnjava terminima klijenta i poslužitelja. Aplikacija koja šalje zahtjev naziva se klijent, a aplikacija koja šalje odgovor naziva se poslužitelj. Dakle, u primjeru vremenske prognoze, baza podataka meteorološkog zavoda je poslužitelj, a mobilna aplikacija je klijent. [8] Na Slika 4.2 prikazana komunikacija između klijenta i poslužitelja. Klijenti traže zahtjeve koje API zaprimi te API prosljeđuje upit nad bazom podataka, a zaprimljene podatke od baze podataka vraća određenom klijentu.

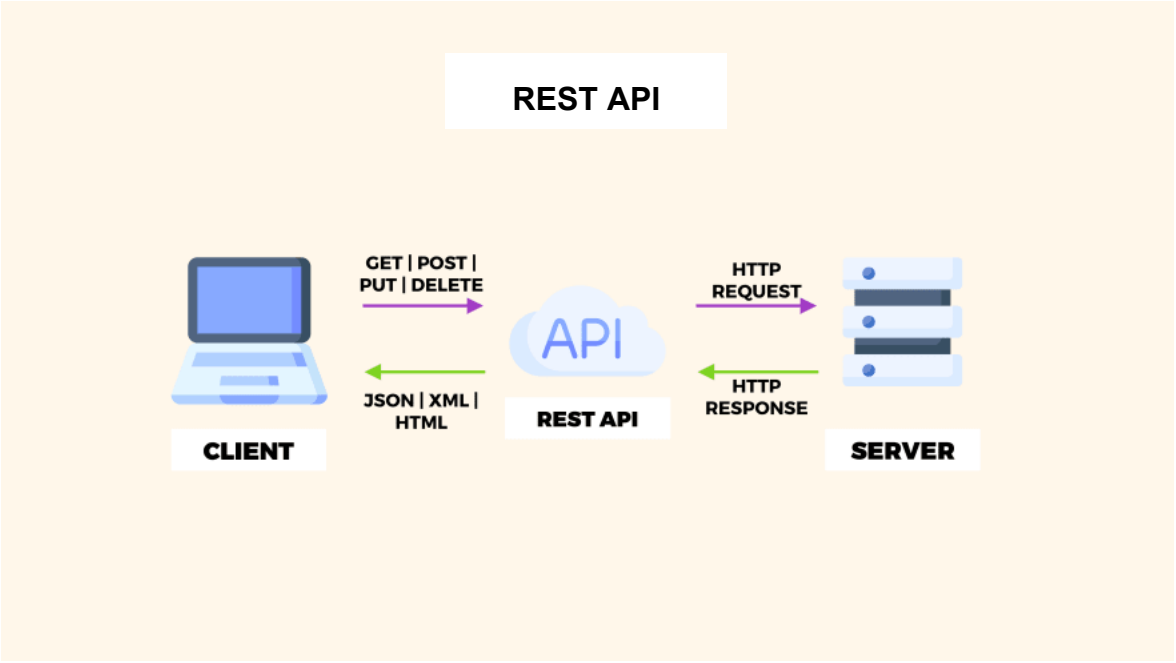



Slika 4.2 Komunikacija klijenta i poslužitelja s API servisom [9]

### 4.2.1. Korištenje REST oblika

Reprezentacijski prijenos stanja (engl. *Represetational State Transfer*) je arhitektura web servisa koja koristi HTTP protokol za komunikaciju između klijenta i servera. REST se koristi za stvaranje skalabilnih i fleksibilnih web servisa i aplikacija. Informacije ili prikaz isporučuju se u nekoliko formata putem HTTP-a: JSON (engl. *Javascript Object Notation*), HTML, XLT, Python, PHP ili čisti tekst, nešto više u nastavku rada. JSON se često koristi jer je čitljiv i za ljude i za strojeve. U REST arhitektonskom stilu, klijent i poslužitelj su dvije neovisne komponente koje mogu raditi neovisno jedna o drugoj, ali mogu međusobno razmjenjivati podatke u istom formatu. To omogućuje da se klijent i poslužitelj razvijaju neovisno jedan od drugog i da budu fleksibilni u promjeni. Svi API zahtjevi za isti resurs trebaju izgledati isto, bez obzira odakle zahtjev dolazi. REST API osigurava da isti dio

podataka, kao što je ime ili adresa e-pošte korisnika, pripada samo jednom jedinstvenom identifikatoru izvora – URI (engl. *Uniform Resource Identifier*). [10]

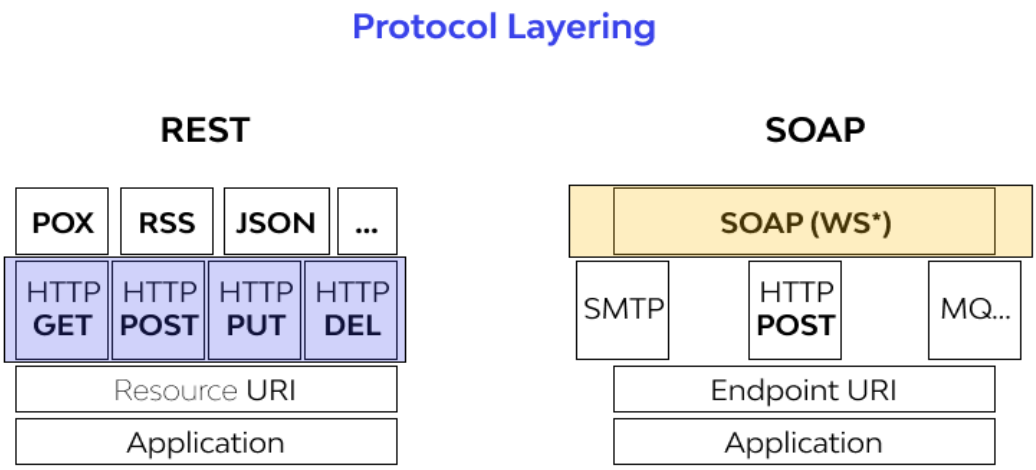



Slika 4.3 Prikaz REST API arhitekture [11]

Slika 4.3 prikazuje objašnjeni dio REST API oblika. Slanje zahtjeva HTTP metodom i primanje podataka preko određenog formata.

### 4.2.2. Korištenje SOAP oblika

SOAP (Simple Object Access Protocol) je protokol koji predstavlja osnovni komunikacijski protokol namijenjen razmjeni tekstualnih poruka između računalnih sustava. SOAP protokolom opisuje se način na koji će poruka biti oblikovana prilikom prijenosa nekim od transportnih protokola i način na koji će ta poruka biti razmijenjena i obrađena među aplikacijama. Poruka se naziva SOAP XML dokument ili SOAP omotnica (engl. *envelope*), a može sadržavati tekst poruke s informacijom (engl. *payload*), poruke o pogrešci, uputstva o obradi poruke, sigurnosna proširenja, prava pristupa poruci. SOAP koristi XML jezik za slanje i primanje odgovora. Najčešće se koristi za koordinaciju, sigurnost, u komunikaciji te u svrhu izmjene transakcija. Sam protokol omogućava jednosmjernu i dvosmjernu komunikaciju. Kod jednosmjerne komunikacije karakteristično je što nema povratnog odgovora, dok u dvosmjernoj komunikaciji slijedi odgovor na zahtjev. Mnoge organizacije koriste fleksibilniji REST API oblik komunikacije, dok velika poduzeća češće koriste SOAP oblik komunikacije. [12]




Slika 4.4 Prikaz protokola [13]

Na Slika 4.4 prikazan je SOAP web servis i njegovi zahtjevi i protokoli. Radi usporedbe uz SOAP nalazi se REST web servis s njegovim zahtjevima i protokolima. REST se često smatra bržom i fleksibilnijom alternativom za implementaciju u scenarijima temeljenim na webu, dok je SOAP protokol sa specifičnim zahtjevima kao što je razmjena XML poruka. REST API-i ne koristi puno memorije, što ih čini idealnim za novije kontekste poput internet stvari (IoT), razvoja mobilnih aplikacija i drugo. SOAP web servis nudi ugrađenu sigurnost i usklađenost transakcija koje su u skladu s potrebama tvrtki, ali zato koristi više memorije. Javni API-i, poput API Google mapa, koriste REST servis. Za potrebe ovog rada komunikacija je omogućena putem REST web servisa zbog jednostavnosti izvedbe, fleksibilnosti i lakše implementacije komunikacije za buduću platformu. (Kanjilal, 2013)

### 4.2.3. JSON format za razmjenu podataka

JSON format (engl. *JavaScript Object Notation*) je jednostavniji tekstualni standard za razmjenu podataka. Format kao takav, vrlo je čitljiv i njegova primjena je vrlo jednostavna u odnosu na druge formate. JSON je programski neovisan s time se može koristiti u bilo kojem programskom jeziku. Također JSON je u podskupu JavaScript-a te se time obrada podatka dobro uklapa u JavaScript kod. Njegov format bazira se na uređenim parovima imena i vrijednosti te podržava podatkovne strukture poput nizova-lista ili vektora. S obzirom da je temeljen na tekstu, JSON koristi malo memorije pa to omogućava brzu

strukturiranu razmjenu podataka, što se u konačnici odnosi na jednostavnu razmjenu potrebnih podataka. U zapisu potrebno je znatno manje informacija nego što je potrebno u XML zapisu jer se koriste samo informacije s objektima. [15]

```
{
   "weight":111,
   "date":"30.1.2023"
},
{
   "weight":120,
   "date":"2.2.2023"
},
{
   "weight":125,
   "date":"5.2.2023"
}
```

Kôd 4.2 Primjer JSON formata

Kôd 4.2 prikazuje dohvaćene podatke iz baze podataka u JSON formatu. Dohvaćene podatke vratila je procedura s Kôd 4.1. Vidljivo je da JSON format lako čitljiv za čovjeka i stroj.

### 4.2.4. Praktična primjena programskog sučelja - API

Programsko sučelje - API će se nalaziti na internetu i bit će zaduženo za svaki oblik komunikacije s bazom podataka i trenutno s web aplikacijom. Radi moguće buduće nadogradnje na platformu za mobilne uređaje, izabran je vanjski web servis kako bi bilo koja platforma mogla pristupiti podacima. Web aplikacija i programsko sučelje – API povezani su pomoću URL-a te tako web aplikacija dohvaća podatke koje mu API prosljeđuje od baze podataka. Unutar API-a nalazi se više upravljača gdje se u svakom nalaze logički definirane metode. Pomoću REST poziva odvija se sva komunikacija tj. upravljač i metode unutar njega su zaduženi za razmjenu podataka kao stvaranje, ažuriranje, brisanje ili čitanje podataka.

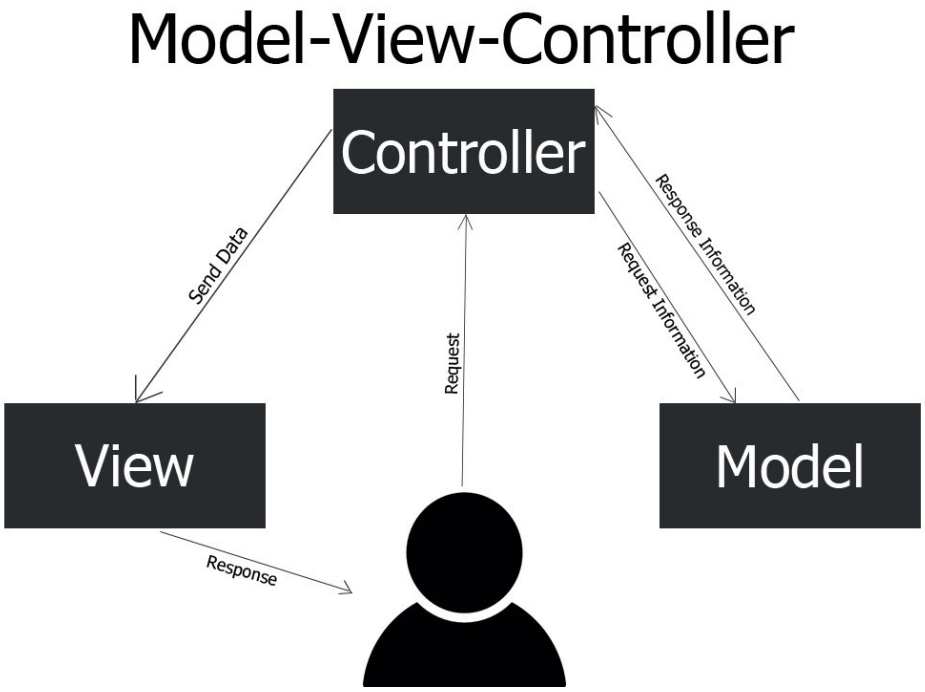
```
[HttpGet("id={id}/name={name}")]
public List<Graf> GetWeightChartInfo(int id, string name)
{
 List<Graf> a = new List<Graf>();
 cmd = new SqlCommand("getWeights", Con);
 cmd.CommandType=System.Data.CommandType.StoredProcedure;
 cmd.Parameters.Add("@id",
 System.Data.SqlDbType.Int).Value= id;
 cmd.Parameters.Add("@name",
 System.Data.SqlDbType.VarChar).Value = name;
 Con.Open();
 cmd.Connection = Con;
 cmd.ExecuteNonQuery();
 dr = cmd.ExecuteReader();
 while (dr.Read())
 {
   a.Add(new Graf
   {
     Date = dr["DateCreated"].ToString(),
     Weight = (double)dr["Weights"]
   });
 }
 Con.Close();
 return a;
}
```

Kôd 4.3 Primjer metode poziva unutar programskog sučelja

Postavljeni primjer Kôd 4.3 prikazuje dohvaćanje podataka iz baze podatka gdje se povezuje s procedurom `getWeights` koja je prikazana na primjeru Kôd 4.1. Metoda `GetWeightChartInfo` vraća podatke kilaže i datuma web aplikaciji.

## 4.3. MVC oblikovni obrazac

Arhitektonski obrazac Model-View-Controller (MVC) dijeli aplikaciju u tri glavne skupine komponenti, model, pogled (engl. *view*) i upravljač (engl. *controller*). Svaka skupina obrazaca odrađuje određeni dio unutar MVC aplikacije. Takav način raspodjele pomaže u razdavanju briga (engl. *Separation of concerns*) te se takvim načinom razvoja aplikacije omogućuje bolja preglednost koda, ispravak pogrešaka i testiranja. Također, takav način uveliko pojednostavljuje rad na istom projektu s više razvojnih inženjera. Korištenje obrasca prikazano je na Slika 4.5. Dolazni zahtjevi korisnika usmjeravaju se do upravljača koji je odgovoran za rad s modelom, za izvođenje radnji korisnika i dohvaćanje rezultata upita. Upravljač prihvaća dohvaćene podatke modela te odabire pogled koji će predstaviti korisniku.[16]




Slika 4.5 Prikaz MVC arhitekture [17]

### 4.3.1. Upravljač

Upravljač je komponenta koja upravlja interakcijom korisnika i rada s modelom, kako bi u konačnici odabrao pogled za renderiranje. U MVC aplikaciji, pogled prikazuje samo dane informacije dok upravljač obrađuje i odgovara na unos i interakciju korisnika. Kako i samo ime govori, on upravlja kako aplikacija odgovara na određeni zahtjev, odnosno upravljač je početna ulazna točka i odgovoran je za odabir s kojim tipovima modela će raditi i koji će pogled renderirati. (Freeman, 2013)

### 4.3.2. Pogled

Pogledi su odgovorni za prikaz sadržaja kroz korisničko sučelje. Pomoću Razor preglednika ugrađuju C# kod unutar HTML-a. Koristi se za svu logiku korisničkog sučelja aplikacije tako što generira korisničko sučelje za korisnika. Prikazi se renderiraju pomoću podataka koje prikuplja model. Budući da pogled jedino komunicira s upravljačem, podaci modela se ne uzimaju izravno iz modela, nego putem upravljača. (Freeman, 2013)

### 4.3.3. Model

Model odgovara cijeloj logici podataka s kojom korisnik radi. To mogu biti podaci koji se prenose između komponenti upravljač i pregled ili bilo koji podaci povezani s poslovnom logikom. Model sudjeluje u dodavanju ili dohvatu podatka. Odgovara na zahtjev upravljača jer upravljač ne može sam komunicirati s programskim sučeljem – API ili bazom podataka. (Freeman, 2013)

### 4.3.4. Razor

Pogledi ASP.NET Web Application MVC koriste mehanizam Razor za renderiranje pogleda. Razor je kompaktan jezik za definiranje pogleda pomoću ugrađenog C# koda, odnosno sintaksa dopušta dodavanje poslužiteljskog koda klijentskom sadržaju. Pomoću Razor-a, pogledi u MVC-u mogu se strogo tipizirati na temelju modela. Upravljač može proslijediti strogo tipizirani model pogledima omogućujući im provjeru tipa. Korištenjem Razor mehanizma moguće je definirati izglede, zamjenjive dijelove ili djelomične prikaze. [19]

### 4.3.5. Praktična primjena MVC arhitekture

Web aplikacija MVC je prezentacijski sloj odnosno sloj s kojim komunicira korisnik. Aplikacija je zadužena za prikazivanje potrebnih podataka primljenih od programskog sloja – API. Komunikacija web aplikacije i programskog sučelja – API odvija se preko URL-a (o čemu je govoreno u primjeni API-a) i JSON formata, a na taj način web aplikacija šalje i dohvaća podatke koje API prihvaća ili prosljeđuje od baze podataka. Sama web aplikacija nema direktnih dodirnih točaka s bazom podataka i obrnuto.

```
public static List<Graf> GetWeightData(WeightChartModel info)
 {
  using (var client = new HttpClient())
  {
   var endpoint = new Uri(url + "User/id=" + info.UserID +
   "/name=" + info.ExerciseName);
   var result = client.GetAsync(endpoint).Result;
   var json = result.Content.ReadAsStringAsync().Result;
   return JsonConvert.DeserializeObject<List<Graf>>(json);
  }
 }
```

Kôd 4.4 Primjer metode za komunikaciju s API servisom

Kôd 4.4 je primjer metode koja komunicira s programskim sučeljem – API. Pomoću metode i JSON formata web aplikacija šalje i prima podatke od API-a. Također pomoću ove metode web aplikacija će poslati i dohvatiti podatke iz primjera Kôd 4.3. Dohvaćeni podaci uz pomoć upravljača i modela biti će prikazani korisniku kao pogled.

# 5. Opis funkcionalnosti implementirane aplikacije

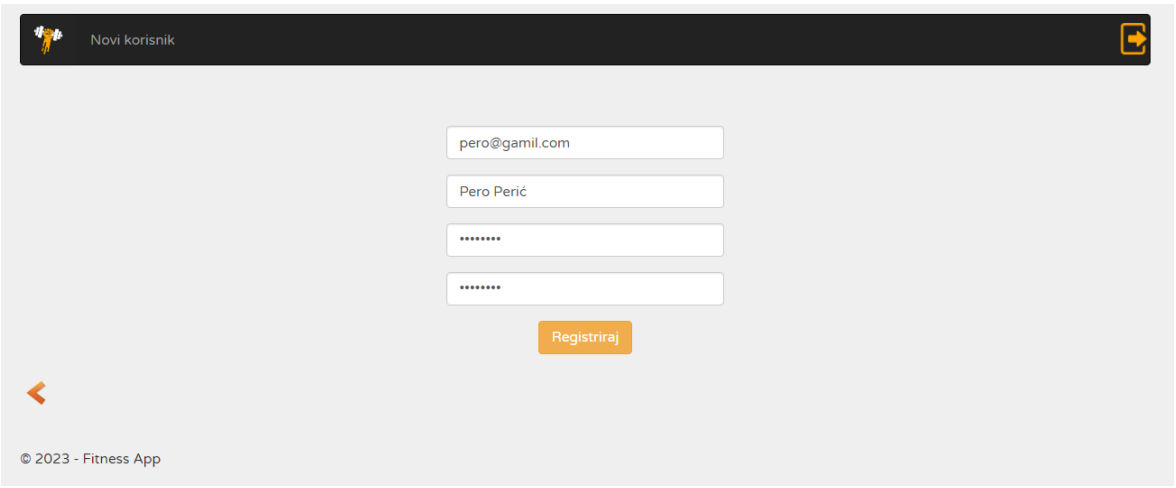
U ovom dijelu bit će opisane funkcionalnosti web aplikacije i pobliže objašnjen način na koji rade. Kroz cjeline bit će opisan proces prijave i registracije, praćenje prehrane, praćenje treninga, grafički prikaz napretka i podrška fizioterapeuta.

## 5.1. Prijava i registracija

U nastavku rada bit će opisana registracija klijenta i prijava trenera, fizioterapeuta i klijenta. Kroz primjere iz aplikacije opisat će se koraci koje korisnik prolazi te kako se ti podaci dohvaćaju i zapisuju.

### 5.1.1. Registracija u web aplikaciju

Prilikom postavljanja aplikacije u rad, korisnički računi trenera i fizioterapeuta unaprijed su postavljeni. Jedina registracija koja se događa na prezentacijskom sloju jest registracija novih klijenata. Prilikom ulaska klijenta u trenerov program, trener ga registrira kao novog klijenta. Kod registracije klijenta, trener upisuje ime, prezime, e-mail adresu i nasumičnu šifru što se vidi na Slika 5.1.




Slika 5.1 Registracija klijenta

Nakon što su podaci uneseni i pritisnuta je tipka „Registriraj“, aktivira se akcijska metoda Kôd 5.1 koja primljene podatke prosljeđuje metodi Kôd 5.2 .

```
[HttpPost]
public ActionResult TrenerRegistracija(UsersRegister user)
{
     if (user.Pwd != user.PwdChecker)
     {
          return View();
     }
     else
     {
          Users user2 = new Users
          {
          Pwd = user.Pwd,
          Id = user.Id,
          DateCreated = user.DateCreated,
          Email = user.Email,
          Username = user.Username,
          Roll = ""

          };
     int a = Repository.CreateUser(user2);
     ViewBag.TrenerRegistracija
     = "Korisnik je uspješno dodan!";
     return View();
     }
}
```

Kôd 5.1 Akcijska metoda za registraciju klijenta

```
internal static void CreateUser(Users a)
{
     using (var client = new HttpClient())
     {
          var endpoint = new Uri(url + "User/AddUser");
          var newUserJson = JsonConvert.SerializeObject(a);
          var payload = new StringContent(newUserJson
          ,Encoding.UTF8, "application/json");
          var result = client.PostAsync(endpoint
          ,payload).Result;
          var json =
          result.Content.ReadAsStringAsync().Result;
     }
}
```

Kôd 5.2 Metoda za slanje podatka API servisu

Ukoliko su podaci točno upisani, a šifra i ponovljena šifra jednake, metoda Kôd 5.2 primljene podatke u JSON formatu šalje programskom sučelju – API.

Metoda Kôd 5.3 unutar programskog sučelja – API, prima podatke poslane od web aplikacije i prosljeđuje ih proceduri unutar baze podataka koja kreira novog klijenta.

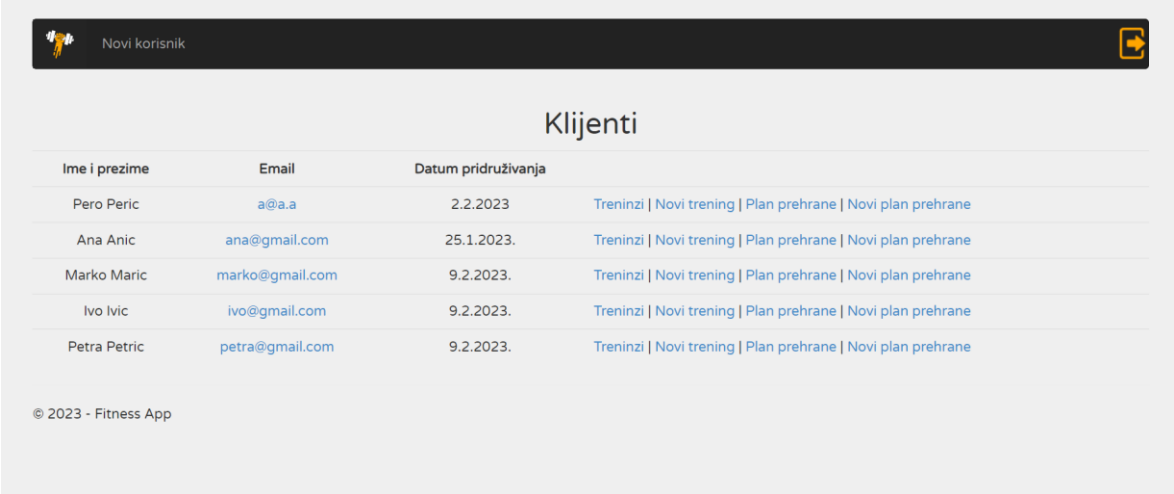
```
[HttpPost]
[Route("AddUser")]
public void AddUser(Users user)
{
 try
 {
  Random random = new Random();

  const string chars="ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
  string salt = new string(Enumerable.Repeat(chars, 32)
  .Select(s => s[random.Next(s.Length)]).ToArray());

  cmd = new SqlCommand("createUser", Con);
  cmd.CommandType = System.Data.CommandType.StoredProcedure;
  cmd.Parameters.Add("@name",
  System.Data.SqlDbType.VarChar).Value = user.Email;
  cmd.Parameters.Add("@userName",
  System.Data.SqlDbType.VarChar).Value = user.Username;
  cmd.Parameters.Add("@roll",
  System.Data.SqlDbType.VarChar).Value = user.Roll;
  cmd.Parameters.Add("@date",
  System.Data.SqlDbType.VarChar)
  .Value=DateTime.Now.ToString();
  cmd.Parameters.Add("@pwd",
  System.Data.SqlDbType.VarChar).Value = user.Pwd;
  cmd.Parameters.Add("@id",
  System.Data.SqlDbType.Int)
  .Direction=ParameterDirection.Output;
  Con.Open();
  cmd.Connection = Con;
  cmd.ExecuteNonQuery();
```

```
  Con.Close();
  HttpContext.Response
  .StatusCode=StatusCodes.Status201Created;
  }
  catch(Exception)
  {
   throw;
  }
}
```

Kôd 5.3 API metoda za registraciju klijenta.

Ukoliko su svi uvjeti zadovoljeni i ne postoji klijent s istom e-mail adresom u bazi, korisnik će biti uspješno registriran.

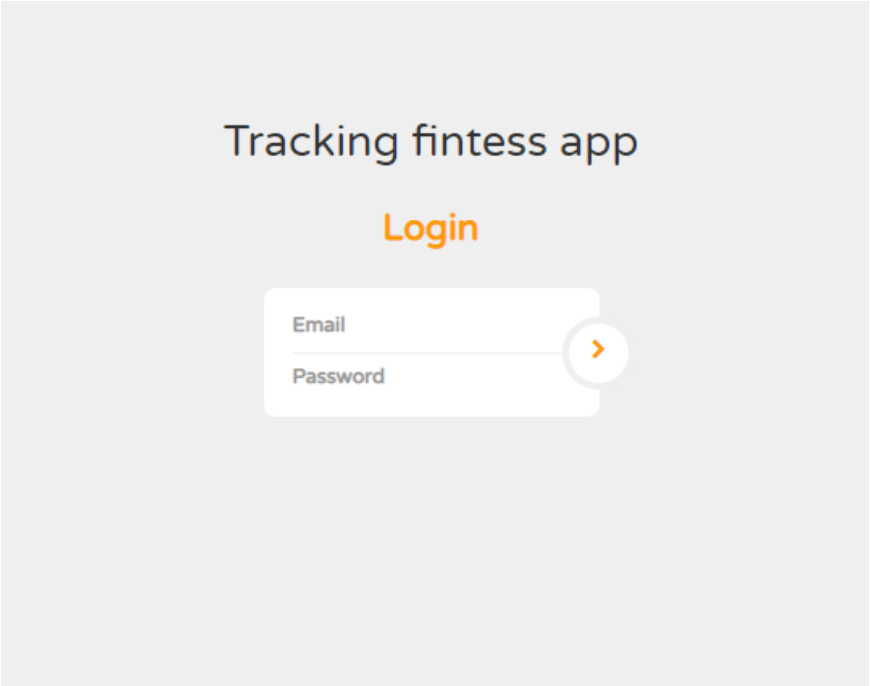
Nakon što je klijent uspješno dodan u bazu podataka, treneru se na početnoj kartici `TrenerIndex` prikazuje klijent u tablici sa svim ostalim klijentima. Odabirom klijenta, trener može kreirati novi trening i vježbe, kao i kreirati novi plan prehrane. Također, trener dobiva uvid u povijest svih treninga koji su odrađeni i sve planove prehrane. Na Slika 5.2 može se vidjeti kako izgleda tablica s klijentima.




Slika 5.2 Izgled tablice klijenta

### 5.1.2. Prijava u web aplikaciju

Aplikacija sadrži prijavu za trenera, fizioterapeuta i klijenta. Prilikom prijave korisnik unosi e-mail adresu i lozinku u polja (Slika 5.3). Nakon unošenja e-mail adrese i lozinke, aktivira se akcijska metoda koja primljene podatke prosljeđuje metodi `CheckLogin`. Metoda u `CheckLogin` šalje dobivene podatke programskom sučelju - API koji komunicira s bazom podataka. API izvršava provjeru podataka u bazi podataka. Po završetku provjere web aplikacija prima povratnu informaciju o tome jesu li uneseni podaci ispravni ili pogrešni. Ukoliko su podaci ispravni i nije bilo nikakve druge pogreške, korisnika se preusmjerava na početnu stranicu. Podaci o prijavljenom korisniku spremaju se u web sesiju (engl. *session*). Sve dok je korisnik prijavljen, dozvoljene su mu akcije unutar web aplikacije.




Slika 5.3 Forma za prijavu

Kôd 5.4 prikazuje implementiranu metodu u web aplikaciji koja komunicira s programskim servisom – API.

```
public static int CheckLogin(Users user)
{
     try
   {
    using (var client = new HttpClient())
    {
     var endpoint = new Uri(url + "User/uname=" +
     user.Email + "&pwd=" + user.Pwd);
     var result=client.GetAsync(endpoint).Result;
     var json=result.Content.ReadAsStringAsync().Result;
     return JsonConvert.DeserializeObject<int>(json);
    }
   }
   catch (Exception ex)
   {
    return -1;
   }
}
```

Kôd 5.4 Metoda `CheckLogin`

Kôd 5.5 prikazuje metodu unutar API servisa za dohvat podatka iz baze podataka pomoću generirane procedure te ovisno o postojanju zapisa u bazi podataka, prosljeđuje potrebne informacije u web aplikaciju. Ukoliko korisnik postoji, metoda vraća podatke, a ukoliko ne postoji, vraća vrijednost -1.

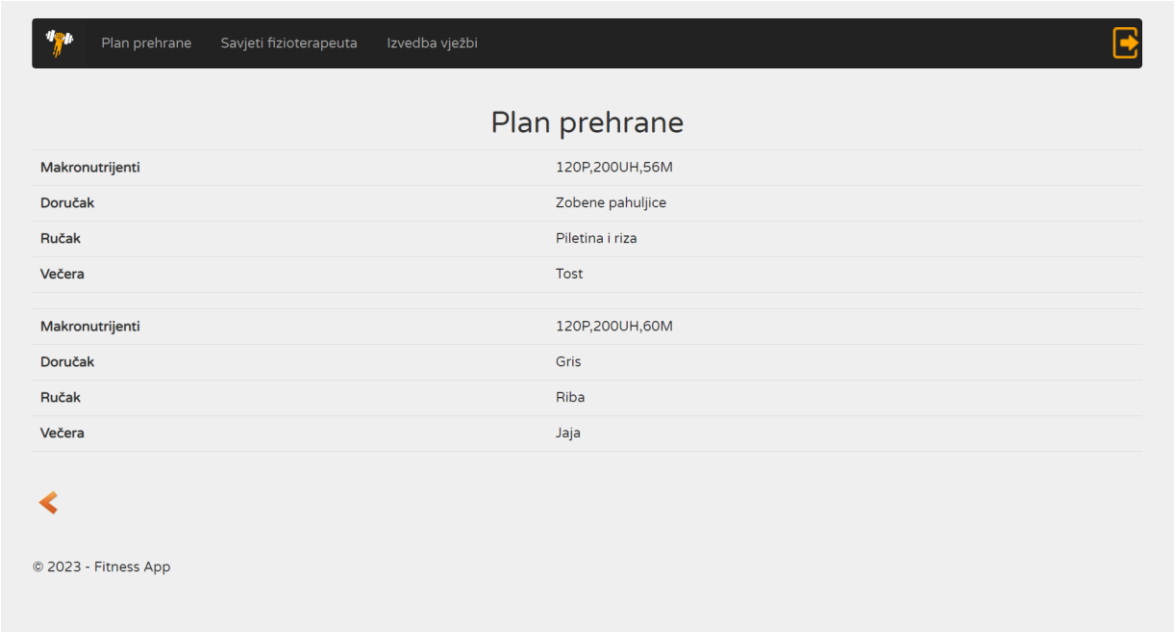
```
[HttpGet("uname={uname}&pwd={pwd}")]
public int GetLogin(string uname, string pwd)
{
 int a = -1;
 cmd = new SqlCommand("provjeriLogin", Con);
 cmd.CommandType = System.Data.CommandType.StoredProcedure;
 cmd.Parameters.Add("@name",
```

```
 System.Data.SqlDbType.VarChar).Value = uname.Trim();
 cmd.Parameters.Add("@pwd",
 System.Data.SqlDbType.VarChar).Value = pwd.Trim();
 Con.Open();
 cmd.Connection = Con;
 cmd.ExecuteNonQuery();
 dr = cmd.ExecuteReader();
 while (dr.Read())
 {
    a = (int)dr["IDUser"];
 }
 Con.Close();
 return a;
}
```

Kôd 5.5 Metoda za provjeru podataka korisnika

### 5.1.3. Praćenje prehrane

Uz trening, bitna stavka u fitnessu je prehrana i praćenje iste. Web aplikacija za praćenje klijenta omogućuje korisnicima koji imaju ulogu trenera da unutar aplikacije prate prehranu svojih klijenata. Unutar aplikacije trener putem opcije „Novi plan prehrane“ prikazano na Slika 5.2, otvara “`NewDietPlanView` i kreira personalizirani plan prehrane za svakog od klijenata. Plan prehrane sadrži makronutrijente koji se sastoje od dnevne potrebe proteina, ugljikohidrata i masti, izražene u gramima. Trener nakon izračuna upisuje podatke u polje za makronutrijente. Jedan primjer plana prehrane sadrži sva tri dnevna obroka - doručak, ručak i večeru. Kako bi trener upotpunio cjelodnevni plan prehrane upisuje obroke u polja. Po završetku plana prehrane, podaci plana prehrane i ID klijenta spremaju se pomoću programskog sučelja – API u bazu podataka. Trener unutar profila svakog klijenta, izgledom `GetUserDietPlanView` pristupa svim planovima prehrane tog klijenta.

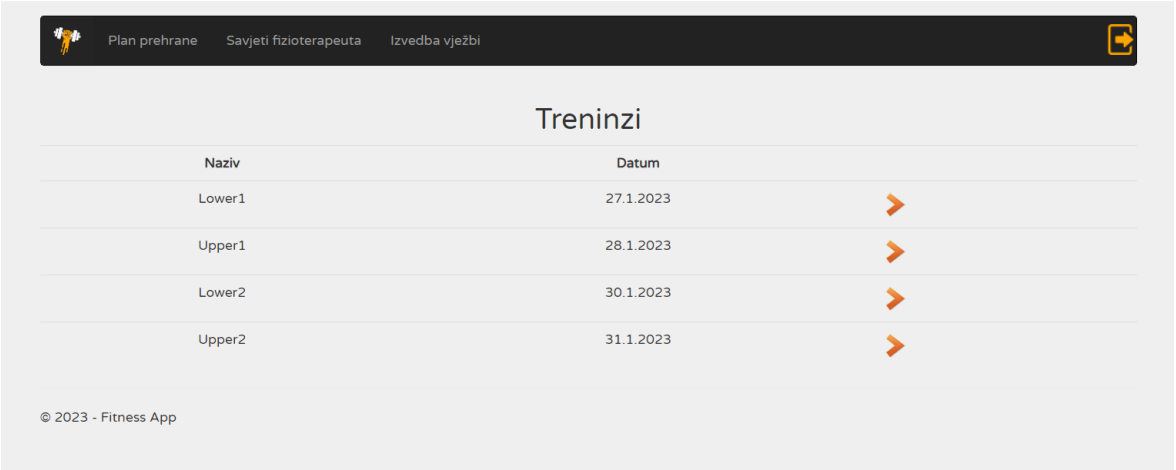



Slika 5.4 Prikaz izgleda plana prehrane

Klijent unutar svojeg sučelja pritiskom karticu `GetDietPlanView` otvara svoj plan prehrane, prikazan na Slika 5.4 te ostale podatka koji su kreirani od strane trenera.

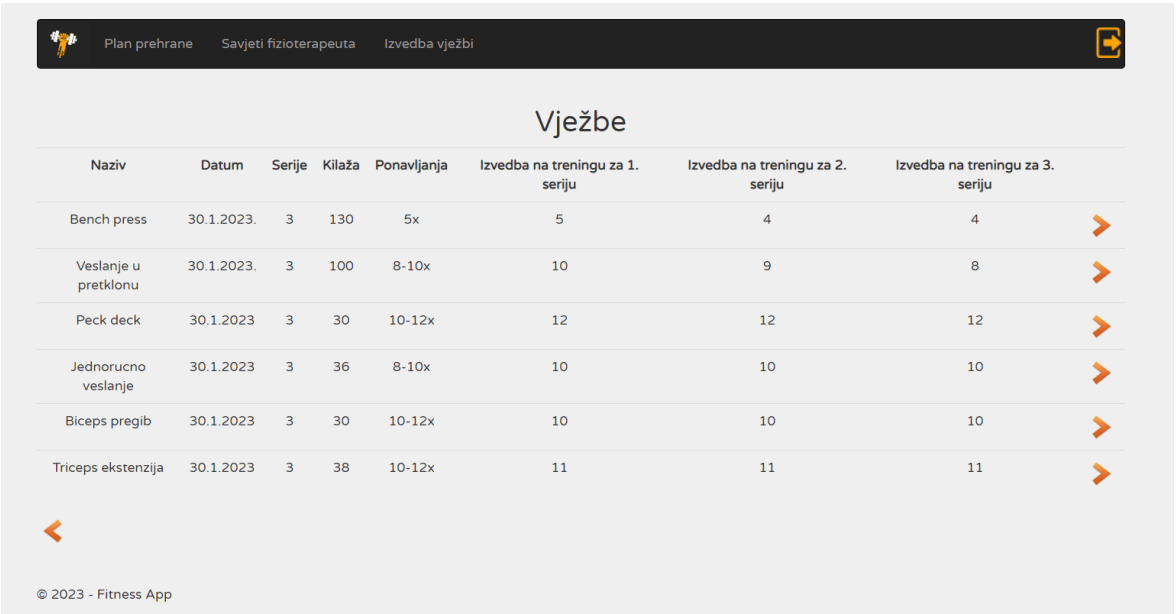
### 5.1.4. Praćenje treninga

Praćenje treninga je jedna od ključnih funkcionalnosti web aplikacije. Trenerima omogućuje na sofisticirani i jednostavan način praćenja treninga klijenata. Na sličan način kreiranja novog plana prehrane, trener kreira novi personaliziran trening. Klikom na „Novi trening“ prikazano na Slika 5.2, otvara se `NewWorkoutView` s poljem naziva treninga. Po završetku kreiranja novog treninga, trening s ID-em klijenta, nazivom treninga i današnjim datumom kreiranja treninga prosljeđuje se programskom sučelju – API i u konačnici sprema u bazu podataka. Svaki trening se sastoji od vježbi koje sadrže naziv, ponavljanja, radnu kilažu i ostale informacije prikazane na Slika 5.6. Izrađenom treningu trener dodaje vježbe putem `NewWorkoutExerciseView.` Nakon unosa svih podataka koji definiraju vježbu, vježba se sprema u bazu podataka na sličan način kao i trening. Podaci treninga i ID treninga spremaju se u bazu podataka.



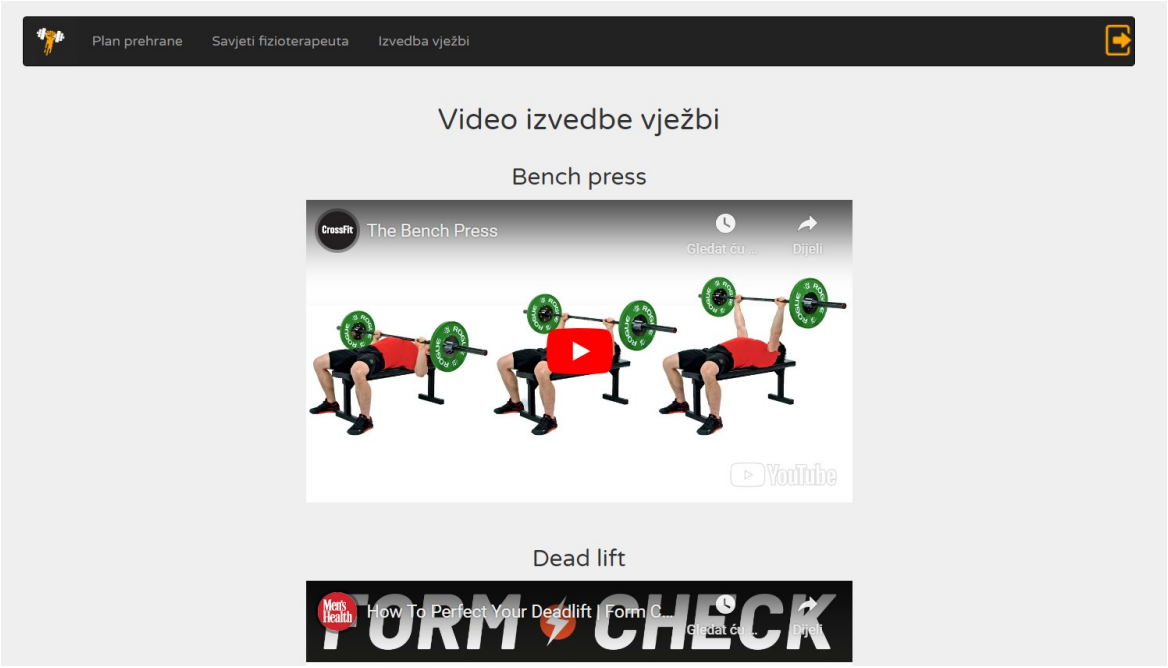

Slika 5.5 Prikaz tablice treninga

Slika 5.5 prikazuje početnu stranicu jednog klijenta. Na početnoj stranici vidljiv je naziv svakog treninga i datum kada je kreiran. Karticu s treninzima kao klijent, vidi i trener kad odabere određenog klijenta. Na taj način klijent ima uvid u treninge koje mora odraditi naredni tjedan i trener može u bilo kojem trenutku pristupiti treninzima i vidjeti koje treninge je odradio s klijentom.




Slika 5.6 Prikaz izgleda vježbi unutar treninga

Na Slika 5.6 vidljivo je kako izgleda jedan trening, odnosno vježbe unutar jednog treninga. Odabirom jednog od treninga `GetUserExercisesView` prikazuje vježbe i informacije koje je trener zadao klijentu.

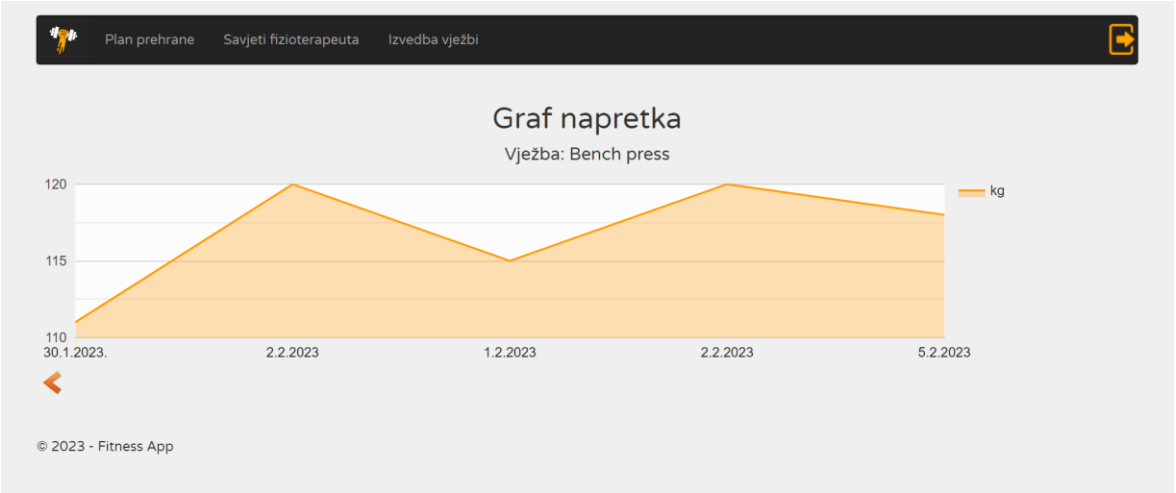



Slika 5.7 Prikaz video uputa

Funkcionalnost video uputa u web aplikaciji klijentu uvelike pomaže prilikom online suradnje s trenerom za savladavanje pravilne izvedbe vježbe. Slika 5.7 prikazuje video upute s aplikacije YouTube gdje su prikazane izvedbe za svaku vježbu.

### 5.1.5. Grafički prikaz napretka

Kod vježbanja vrlo je bitno pratiti napredak radi mišićnog rasta. Funkcionalnost praćenja napretka lakše je pratiti uz pomoć grafičkog prikaza. Podaci vezani uz treninge i vježbe klijenta čuvaju se u bazi podataka. Prilikom inicijalnog učitavanja stranice, podaci se učitavaju iz baze podatka te pohranjuju u trenutnu sesiju web stranice. Podaci potrebni za izradu grafičkog prikaza su logički organizirani i spremljeni u listu objekata. Na pogledu za vježbe `GetUserExercisesView`, klijent može vidjeti sve vježbe i njene informacije te pritiskom na određenu vježbu prikazati grafički prikaz za odabranu vježbu. Nakon izbora vježbe, podaci se prikazuju na `GrafView` pogledu.

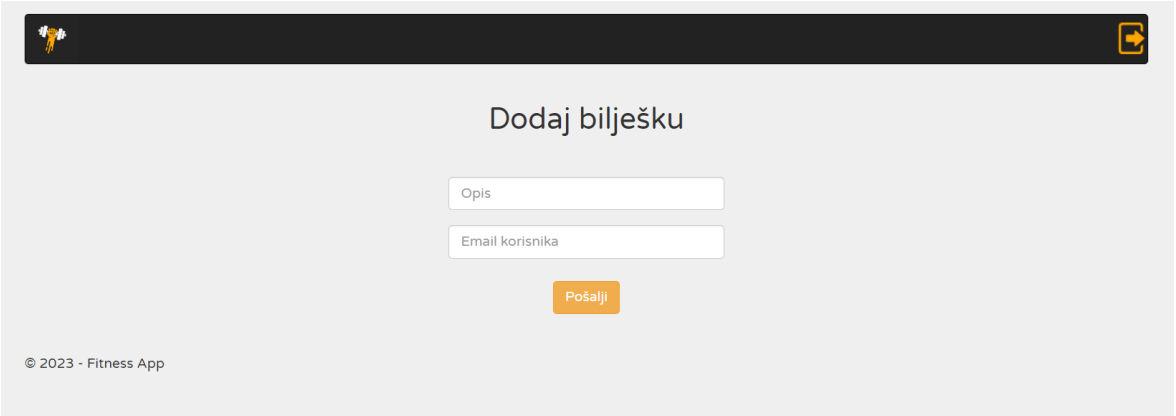



Slika 5.8 Primjer grafičkog prikaza

Slika 5.8 prikazuje `GrafView` za odabranu vježbu. Prikazuju se informacije datuma kada je vježba odrađena i s kojom je radnom kilažom odrađena. Grafički prikaz je dostupan za svaku vježbu koju trener kreira unutar treninga. Definiranje nove vježbe vrši se pomoću `NewWorkoutExerciseView.`

### 5.1.6. Podrška fizioterapeuta

Kod bilo koje vrste aktivnosti pa tako i u fitness dolazi do ozljeda. U fitnessu osim osobnog trenera javlja se potreba i za pomoći fizioterapeuta. Suradnja trenera i fizioterapeuta omogućena je kao dodatna mogućnost koju web aplikacija pruža. Klijent uz trenerske podrške dobiva i podršku fizioterapeuta kroz savjete u obliku bilješki. Klijentu je omogućen pregled svih savjeta fizioterapeuta. Ova funkcionalnost je značajna po tome što omogućuje prevenciju ozljeda ili oporavak istih, što ujedno izdvaja aplikaciju na tržištu od ostalih aplikacijskih rješenja na tržištu. Cilj ove implementacije je olakšati trenerima posao i klijentima pružiti potpunu podršku za osobni napredak.




Slika 5.9 Izgled forme za dodavanje bilješki od strane fizioterapeuta

Slika 5.9 prikazuje sučelje fizioterapeuta koje se otvara prilikom prijave fizioterapeuta. Fizioterapeut u formu za opis opisuje problem klijenta, dodjeljuje savjet i sve ostalo što je potrebno za pomoć klijentu, dok u drugu formu unosi e-mail adresu korisnika kojemu želi poslati bilješku. Nakon što su svi podaci uspješno uneseni, nakon slanja bilješke, programsko sučelje – API komunicira s bazom podataka gdje se bilješka povezuje s korisnikom i sprema.

Klijent unutar svog sučelja ima mogućnost pregledavanja svih savjeta koje je dobio od fizioterapeuta pomoću `GetUserFizioAdviceView`.

# Zaključak

Prema iskustvu autora, u današnje vrijeme je sve popularniji fitness što se može vidjeti po broju različitih fitness aplikacija ili proizvoda u dućanu. Prilikom pandemije počeo se sve više razvijati online format. Na taj način se počelo provoditi online treniranje i praćenje treninga. Također prema iskustvu autora većina trenera na ovim prostorima (osobito novi treneri) imaju komunikaciju sa svojim klijentima preko emaila odnosno Excel tablica gdje prate tjedni napredak.

Analizom i usporedbom postojećih rješenja i potrebama tržišta, napravljena je web aplikacija za praćenje treninga između trenera i klijenta uz podršku fizioterapeuta. Aplikacija sadrži funkcionalnosti više različitih konkurentskih aplikacija u jednoj kako bih se optimizirala suradnja između trenera, fizioterapeuta i klijenta te na taj način povezala dva zanimanja u jednu aplikaciju. Napravljeni izbor arhitekture i raspodjela aplikacije na slojeve omogućuje takvo što kao i lakše održavanje koda, jasniju raspodjelu zadataka aplikacije, proširenja na ostale platforme i implementaciju novih funkcionalnosti što je u budućnosti cilj.

Kroz ovaj rad sam tako naučio kako odabrati pravu arhitekturu, unaprijed isplanirati opseg posla i odabrati tehnologije. Također, naučio me kako isplanirati model baze podataka, klase unutar API-a i web aplikacije, pa tako i planiranje željenih funkcionalnosti određenog modela u aplikaciji. Ujedno sama organizacija i nove tehnologije su mi bile najveći izazov, no uz literaturu i dobru informiranost, problemi prilikom izrade rada bili su umanjeni.

Znanje i iskustvo stečeno kroz izradu ovoga rada bit će korisno za daljnje razvijanje web aplikacije i buduće nadogradnje iste.

*„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.*

*U Zagrebu, 23.02.2023.*

*Leon Pintarić*

# Popis kratica

| | | |
|---|---|---|
| API | *Application Programming Interface* | Programsko sučelje |
| CRUD | *Create, Read, Update, Delete* | Kreiranje, pisanje, ažuriranje, brisanje |
| ER | *Entity Relationship* | Grafički prikaz |
| HTTP | *Hyper Text Markup Language* | Programski jezik za web preglednike |
| JSON | *JavaScript Object Notation* | Tekstualni format za pohranu i prijenos podataka |
| MVC | *Model View Controller* | Arhitektura web aplikacija |
| PHP | *Hypertext Preprocessor* | Jezik za razvoj web aplikacija |
| REST | *Representational state transfer* | Softverski arhitektonski stil |
| SOAP | *Simple Object Access Protocol* | Protokol za razmjenu strukturiranih Informacija |
| SQL | Structured Query Language | Jezik u relacijskim bazama podataka |
| URI | *Uniform Resource Identifier* | Jedinstvena adresa resursa |
| URL | *Uniform Resource Locator* | Jednoznačna adresa dokumenta |
| XML | *Extensible Markup Language* | Jezik za označavanje podataka |
| XLT | *XML Log and Trace* | alat za generiranje i analizu logova u obliku XML-a |

# Popis slika

# Popis kôdova

# Literatura


[1] https://www.trainerize.com/ (pristupljeno 27.1.2023)

[2] https://truecoach.co/ (pristupljeno 27.1.2023)

[3] https://www.powerdiary.com/?fpr=jardel50 (pristupljeno 27.1.2023)

[4] https://www.getapp.co.uk/software/120943/power-diary (pristupljeno 27.1.2023)

[5] *FELIX ALVARO. SQL: Easy SQL Programming & Database Management For Beginners. 2016.* (pristupljeno 2.2.2023)

[6] https://www.techtarget.com/searchdatamanagement/definition/database (pristupljeno 2.2.2023)

[7] https://www.lucidchart.com/pages/er-diagrams (pristupljeno 2.2.2023)

[8] https://blog.axway.com/learning-center/apis/basics/what-is-an-api (pristupljeno 3.2.2023)

[9] https://www.redhat.com/zh/topics/api/what-are-application-programming-interfaces (pristupljeno 3.2.2023)

[10] https://restfulapi.net/ (pristupljeno 5.2.2023)

[11] https://dreamix.eu/blog/frontpage/to-rest-or-not-to-rest (pristupljeno 7.2.2023)

[12] https://stoplight.io/api-types/soap-api (pristupljeno 7.2.2023)

[13] https://www.wallarm.com/what/differences-soap-vs-rest (pristupljeno 7.2.2023)

[14] *JOYDIP KANJILAL. Asp.Net Web Api: Build RESTful web applications and services on the .NET framework. 2013.* (pristupljeno 8.2.2023)

[15] https://www.json.org/json-en.html (pristupljeno 8.2.2023)

[16] https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm, (pristupljeno 11.2.2023)

[17] https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5 (pristupljeno 11.2.2023)

[18] *ADAM FREEMAN. Pro Asp. Net Mvc 5. 2013.* (pristupljeno 12.2.2023)

[19] https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_razor.htm (pristupljeno 12.2.2023)