Razvoj web aplikacije primjenom Angular razvojnog okvira

Mavrić, Luka

Sveučilište u Rijeci, Fakultet informatike i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Luka Mavrić

Razvoj web aplikacije primjenom Angular razvojnog okvira

Završni rad

Mentor: Doc. dr. sc. Lucia Načinović Prskalo

Rijeka, rujan 2023.

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 20.4.2023.

Zadatak za završni rad

Pristupnik: Luka Mavrić

Naziv završnog rada: Razvoj web aplikacije primjenom Angular razvojnog okvira

Naziv završnog rada na engleskom jeziku: Development of a web application using the Angular framework

Sadržaj zadatka: Zadatak završnog rada je izraditi web aplikaciju uz primjenu Angular razvojnog okvira. U radu će biti opisane sve korištene tehnologije s posebnim naglaskom na implementaciju Angular funkcija i modula te će biti opisane funkcionalnosti izrađene aplikacije.

Mentor

Doc. dr. sc. Lucia Načinović Prskalo

Načinović Prskalo

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: datum

Luka Mavrić

(potpis pristupnika)

Adresa: Radmile Matejčić 2
51000 Rijeka, Hrvatska
Tel: +385(0)51 584 700
E-mail: [redacted]
OIB: [redacted]
IBAN: HR1524020061400006966

Sadržaj

Sažetak

Tema ovog završnog rada je izrada web aplikacije primjenom Angular razvojnog okvira. Razvojni okviri olakšavaju i ubrzavaju proces razvoja softverskih aplikacija pružajući gotove komponente, alate i standarde. U uvodu je kratko opisan Javascript programski jezik i kratko su pojašnjene funkcionalnosti aplikacije „Gym Planner“. Nakon uvoda, rad je fokusiran na tehnologije koje se koriste kod izrade „Gym Planner“ aplikacije. Kao najvažniji alat za izradu frontenda se koristio Angular, a uz njega je korišten i Bootstrap za dizajn aplikacije, a za backend dio aplikacije i spremanje podataka se koristila Firebase platforma. U drugom dijelu rada, prikazani su dijelovi koda aplikacije te su objašnjeni kako oni funkcioniraju. Uz to je objašnjena struktura cijele aplikacije te je prikazan izgled i organizacija Firebase baze.

Ključne riječi: Angular, Typescript, Bootstrap, Firebase, frontend, razvojni okvir, web aplikacija za vježbanje, web aplikacija

1. Uvod

Kada je nastao World Wide Web, prve web stranice su bile vrlo jednostavne te su bile statičke, nisu omogućavale interaktivnost ni dinamičnost. To se promijenilo pojavom JavaScript-a, koji je omogućio izradu dinamičnih elemenata na stranicama, čime su korisnici dobili mogućnost interakcije sa stranicom. I nakon Javascript-a se pojavilo još nekoliko tehnologija koje su omogućile još interaktivnije i dinamičnije web stranice.

Jedan od zadataka ovog završnog rada bio je nadogradnja aplikacije koju sam izrađivao u svoje slobodno vrijeme pod nazivom „Gym Planner“. S njom možemo izrađivati listu vježbi koje sadržavaju opis i video te način izvođenja vježbi te lista treninga gdje u svaki trening možemo dodati ranije izrađene vježbe. Aplikacija je rađena u Angularu - jednom od Javascript razvojnih okvira. Svrha aplikacije je olakšavanje i ubrzavanje kreiranja vježbi i treninga za korisnike. Za pohranu podataka korištena je nerelacijska baza podataka Firebase, a preko platforme Firebase implementirana je i autentifikacija i prijava korisnika. Aplikacija je postavljena na serveru te je pristup do nje omogućen preko bilo kojeg web preglednika.

Razvojni okviri (en. Frameworks) postaju u zadnjih par godina dosta popularni, pa ću kroz ovaj rad i opis aplikacije pokazati kako funkcionira Angular i od koji se elemenata sastoji. U drugom poglavlju su objašnjene sve tehnologije koje sam koristio za izradu aplikacije te su opisane njihove funkcionalnosti. U trećem su poglavlju opisane najvažnije komponente izrađene aplikacije i dijelove koda pomoću kojih su implementirani te način na koji funkcioniraju. Na kraju su dani najvažniji zaključci te planovi za nadogradnju funkcionalnosti aplikacije.

2. Korištene tehnologije

Za izradu frontend dijela aplikacije koristio sam Angular i njegove built-in module. Uz njih, korišteni su i dodatni paketi koji su bili potrebni za prikaz nekih podataka. Za definiranje stila cijele aplikacije koristio sam Bootstrap, a za backend dio aplikacije korištena je Firebase baza podataka. Firebase platforma omogućava i implementaciju autentifikacije pa je korištena za prijavu korisnika. U nastavku ću malo opširnije objasniti svaku od navedenih tehnologija.

2.1. Angular

Angular razvojni okvir je postao prvi put javno dostupan 2010. godine i njegov naziv je tada bio AngularJS. Olakšao je razvoj dinamičnih web aplikacija te je pomoću njega uveden „two way binding“ što omogućava automatsko ažuriranje podataka na sučelju koji korisnik vidi. No tada još nije bio optimiziran i nije bio skalabilan [1][2].

Novija verzija Angulara, Angular 2+, pojavila se 2016. godine. Uvedena je nova arhitektura u obliku komponentne arhitekture i uveden je Typescript kao glavni jezik za razvoj. Te promijene su omogućile bolju organizaciju i strukturiranje aplikacije [2].

Kada je izašla verzija 6, dodan je alat Angular CLI, koji je omogućavao brže generiranje komponenti, servisa, modula i drugih dijelova aplikacije [2]. Angular je redovito ažuriran od strane Google-a koji sluša zajednicu programera preko foruma i raznih drugih stranica.

Ranije spomenuta komponentna arhitektura Angulara se sastoji od tri glavna dijela, a to su *komponente*, *servisi* i *moduli*. Komponente čine glavne građevne jedinice aplikacije. Svaka komponenta se sastoji od 4 datoteke

(HTML, CSS, dvije Typescript datoteke gdje jedna predstavlja datoteku koja daje funkcionalnost komponenti, a druga je datoteka sa skriptom za testiranje). Servisi se koriste za razdvajanje logike aplikacije od komponenti, što bi značilo da se preko njih implementira pohrana, dohvaćanje i brisanje podataka. U aplikaciji „Gym Planner“ , pomoću servisa se izvršava većina poziva prema API[1]-jima. I zadnja bitna komponenta Angulara su moduli. Oni služe za organizaciju aplikacije. Slične komponente, servise i druge varijable grupiramo u module [3]. Kasnije, kada bude objašnjavan kod, slikama će biti prikazano kako sam koristio navedene tri glavne komponente za izradu aplikacije „Gym Planner“.

Uz komponente, servise i module, postoji još jedna dosta bitna datoteka, *naziv_modula-routing,* koja služi za usmjeravanje. Svaki modul ima svoju datoteku za usmjeravanje i postoji takva datoteka unutar glavne mape *app-routing* koja objedinjuje sve ostale datoteke za usmjeravanje na jednom mjestu.

Dodatne zanimljivosti vezane uz korištenje Angulara, bit će prikazane kod prikazivanja koda u sljedećem poglavlju.

2.2. Typescript

Typescript je programski jezik koji je nadogradnja na Javascript, a pojednostavljuje izradu aplikacija kada su one veće i složenije pošto su unutar njega uvedeni tipovi i klase koje Javascript nema. Sve to olakšava web developerima da pišu čišći kod te su sigurniji da će se uspješno izvesti [4].

[1] Skup određenih pravila i specifikacija koje programeri slijede i koje mogu koristiti za povezivanje s određenom aplikacijom [5]

2.3. Bootstrap

Bootstrap je napravljen od strane dva Twitter inženjera, Mark Otta i Jake Spicer, kao dio internog projekta [6]. Smatra se jednim od najpopularnijih frontend alata za izradu responzivnih web aplikacija [7]. Novije verzije imaju već gotove predloške za cijele web aplikacije u koje se samo trebaju postaviti podaci, a nudi i zasebne dijelove kao što su navigacijska traka, hamburger meni, gumbovi i slično. Uz već gotove komponente, nudi i neke css klase koje možemo dodati na bilo koji div ili element u html-u i dobiti sve atribute te klase.

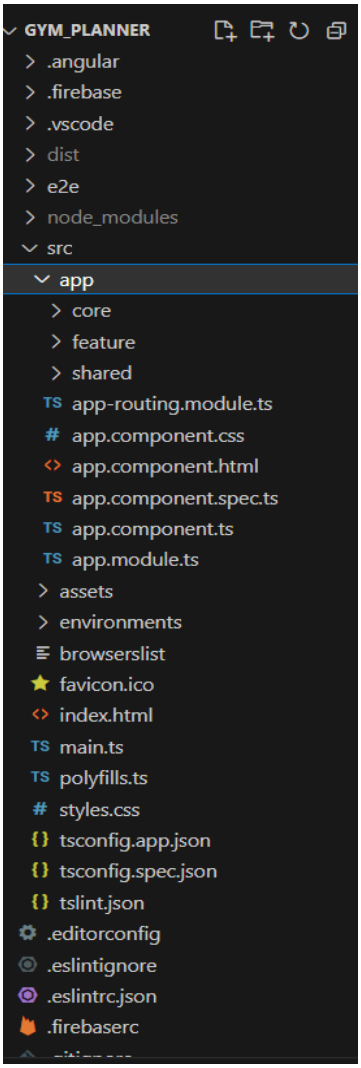
2.4. Firebase

Firebase ima dosta širok spektar alata i usluga koje nudi. Možemo ga koristiti kod izrade web i mobilnih aplikacija. Kod izrade aplikacije korišten je Firebase autentifikacija tako da ako korisnik želi pristup podacima mora se prijaviti ili registrirati. Za potrebe izrade aplikacije „Gym Planner“ korištena je RealTime baza podataka koja je dostupna unutar Firebase platforme. Ta baza podataka omogućava sinkronizaciju podataka između servera i klijenta u realnom vremenu [8][9]. Usluga Firebase platforme koja je također korištena je hosting. Na vrlo jednostavan način, korištenjem nekoliko komandi u konzoli, aplikacija je postavljena na server i dostupna je svima preko URL linka.

3. Gym Planner

3.1. Početni izgled aplikacije

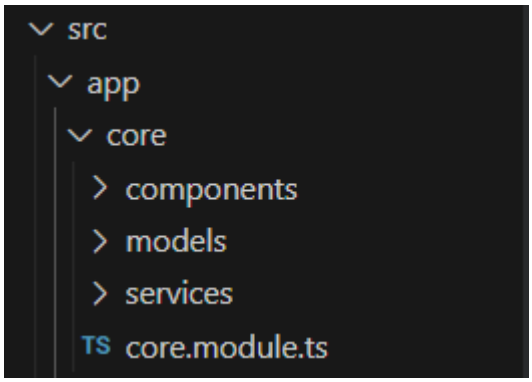
Aplikacija „Gym Planner“ postavljena je pomoću Angular CLI-a, a naredbom *ng new –ime projekta*—započinje izgradnja početne aplikacije koja ima jednu komponentu - *app* komponenta. Nakon uspješnog kreiranja, pokrenuta je aplikacija pomoću naredbe *ng serve* da provjerimo imamo li sve potrebne pakete (Slika 1).



Slika 1 Početni izgled mape aplikacije

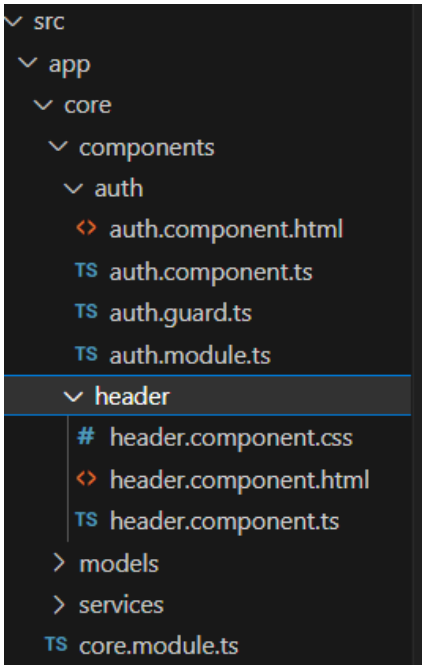
Kada su napravljene sve potrebne provjere, potrebno je napraviti skicu aplikacije, njenog izgleda i organizacije. Prvo su stvorene tri glavne mape u koje se komponente obično razvrstavaju kod izrade aplikacija u Angularu, a to su *core*, *shared* i *feature* mape i svakoj od tih mapa na kraju dodajemo *ime_mape.module.ts* datoteku. Svaka od tih mapa u sebi ima još nekoliko pod mapa, a to su *component*, *services* i *models* mape (Slika 2).

3.2.“Core“ modul

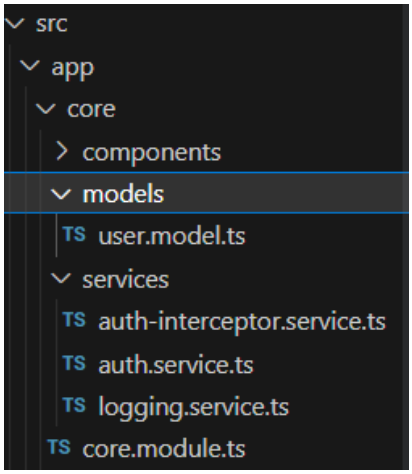


Slika 2 Glavne mape aplikacije

U *core* se stavljaju komponente koje se uvijek prikazuju, a u slučaju aplikacije „Gym Planner“, to su header komponenta i komponenta za autentifikaciju (Slika 3). Uz komponente koje vidimo na slici, *core* mapa također ima još jednu *model* datoteku te nekoliko *service* datoteka (Slika 4).

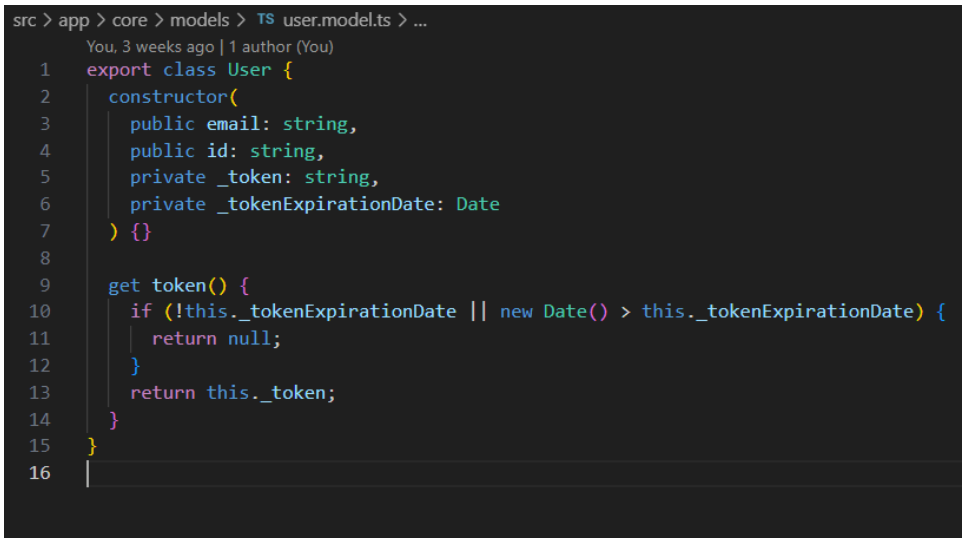


Slika 3 Sadržaj *component* mape



Slika 4 Sadržaj mapa *models* i *services*

Nakon prikaza organizacije *core* modula, prikazat će se neki bitniji dijelovi tog modula. Prvi je *user.model.ts* datoteka (Slika 5). U njoj smo izradili klasu koja predstavlja objekt kakav Firebase očekuje kada šaljemo podatke o korisniku koji se prijavljuje ili registrira u aplikaciju.



Slika 5 *User model* datoteka

Uz model, postoji još nekoliko Typescript datoteka. Jedna od najvažniji je *auth.service.ts* jer se u njoj nalazi cijela logika vezana za autentifikaciju korisnika. U njoj imamo dva poziva prema API-ju - jedan je za registraciju korisnika (Slika 6), a drugi za prijavu korisnika. Uz to,tu se nalazi i funkcija koja omogućava automatsku prijavu korisnika u slučaju osvježavanja stranice (Slika 7), tako dugo dok mu ne istekne sesija. Također imamo i funkciju za automatsku odjavu korisnika kada sesija istekne (sesija tokena u Firebase-u po standardu traje jedan sat) [10].

```
signup(email: string, password: string) {
  return this.http
    .post<AuthResponseData>(
      'https://www.googleapis.com/identitytoolkit/v3/relyingparty/signupNewUser?key='+environment.firebaseAPIKey,
      {
        email: email,
        password: password,
        returnSecureToken: true
      }
    )
    .pipe(
      catchError(this.handleError),
      tap(resData => {
        this.handleAuthentication(
          resData.email,
          resData.localId,
          resData.idToken,
          +resData.expiresIn
        );
      })
    );
}
```

Slika 6 Post poziv za registraciju na Firebase server

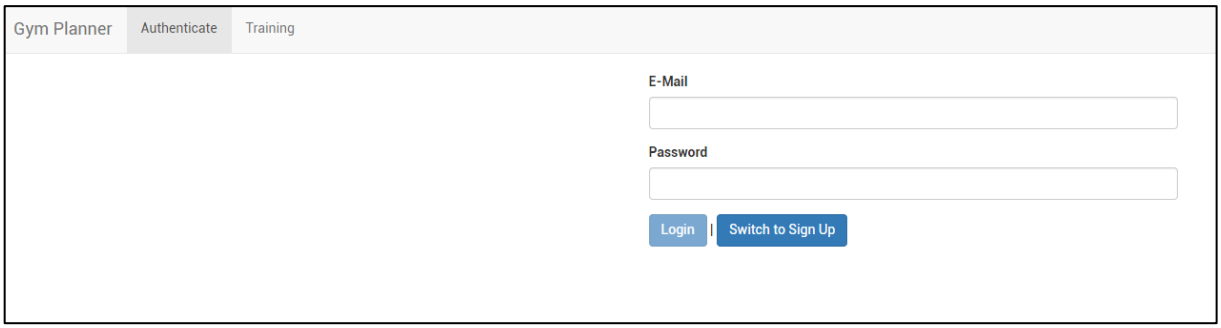
```
73    autoLogin() {
74      const userData: {
75        email: string;
76        id: string;
77        _token: string;
78        _tokenExpirationDate: string;
79      } = JSON.parse(localStorage.getItem('userData'));
80      if (!userData) {
81        return;
82      }
83
84      const loadedUser = new User(
85        userData.email,
86        userData.id,
87        userData._token,
88        new Date(userData._tokenExpirationDate)
89      );
90
91      if (loadedUser.token) {
92        this.user.next(loadedUser);
93        const expirationDuration =
94          new Date(userData._tokenExpirationDate).getTime() -
95          new Date().getTime();
96        this.autoLogout(expirationDuration);
97      }
98    }
99
```

Slika 7 Funkcija za automatsku prijavu

Preostaje nam još pojasniti *header* komponentu. U Typescript datoteci, *header.component.ts* imamo nekoliko funkcija (Slika 8), a unutar njih se nalaze pozivi na druge servise iz drugih datoteka koji će biti kasnije prikazati kod opisa *shared* mape. Unutar *ngOnInita* [2] se provjerava je li korisnik prijavljen. Ostale dvije funkcije koje se vide na slici *onSaveData()* i *onFetchData()*, služe za dohvat i spremanje podataka s Firebase baze. I te dvije komponente, *header* i *auth* komponenta, se prikazuju kroz cijelu aplikaciju (Slika 9).

```
20   ngOnInit() {
21     this.userSub = this.authService.user.subscribe(user => {
22       this.isAuthenticated = !!user;
23     });
24   }
25
26   onSaveData() {      You, 3 weeks ago • first commit
27     this.dataStorageService.storeExercises();
28     this.dataStorageService.storeTrainings();
29   }
30
31   onFetchData() {
32     this.dataStorageService.fetchExercises().subscribe();
33     this.dataStorageService.fetchTrainings().subscribe();
34   }
```

Slika 8 Bitne funkcije unutar header.component.ts

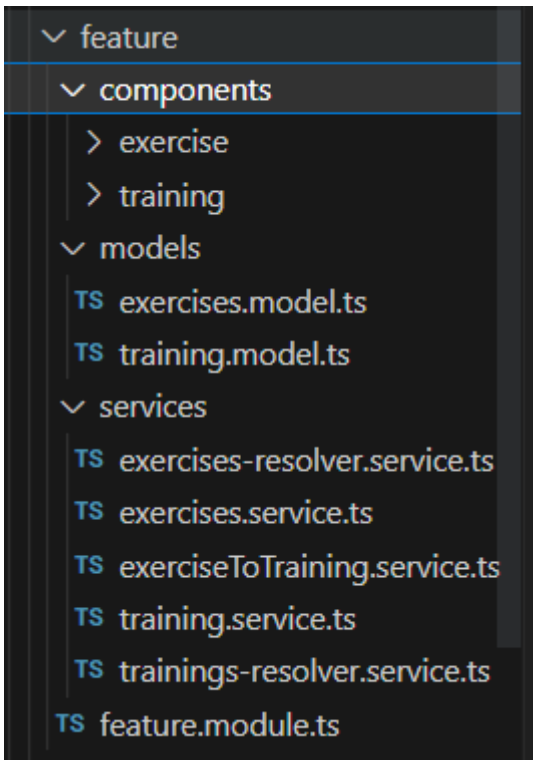


Slika 9 Prikaz dijaloškog okvira za prijavu i registraciju

[2] Jedan od Angular Life Cycle Hookova. Pokreće se kod prvog učitavanja komponente[11]

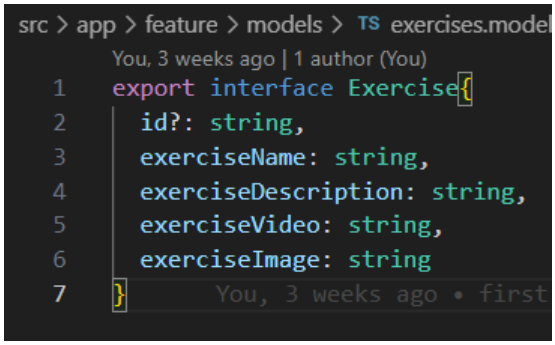
3.3. "Feature" modul

Feature modul je građen slično kao i *core* modul. Sastoji se od *component*, *services* i *model* mapa te *feature.modul.ts* datoteke (Slika 10).

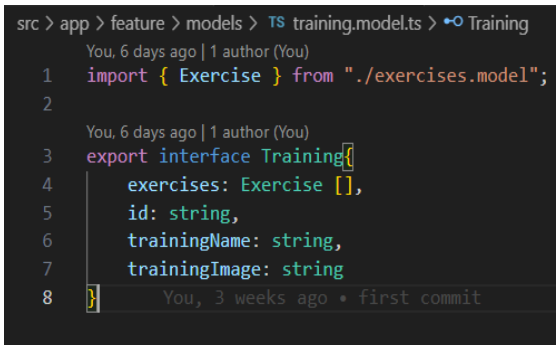


Slika 10 Sadržaj feature modula

Ovaj modul ima dva modela koji se koriste kroz sve komponente aplikacije „Gym Planner". To su model *Exercise* (Slika 11) te model *Training* (Slika 12).



Slika 11 Model vježbi



Slika 12 Model treninga

Nadalje, pojasnit ću i glavne servise ovog modula koji su usko vezano uz svaku komponentu. Svaki od servisa obavlja spremanje, dodavanje, brisanje, promjenu podataka objekta te postavljanje podataka unutar aplikacije (Slika 13). Servis za upravljanje treninzima i vježbama su slični, jedino po čemu se razlikuju su modeli podataka.

```
You, 3 weeks ago | 1 author (You)
@Injectable({ providedIn: 'root' })
export class ExerciseService {
  exerciseChanged = new Subject<Exercise[]>();

  private exercise: Exercise[] = [];

  constructor() {}

  setExercises(exercises: Exercise[]) {
    this.exercise = exercises;
    this.exerciseChanged.next(this.exercise.slice());
  }

  getExercises() {
    return this.exercise.slice();
  }

  getExercise(index: string) {
    return this.exercise[index];
  }

  addExercise(exercise: Exercise) {
    this.exercise.push(exercise);
    this.exerciseChanged.next(this.exercise.slice());
  }

  updateExercise(index: string, newExercise: Exercise) {
    this.exercise[index] = newExercise;
    this.exerciseChanged.next(this.exercise.slice());
  }

  deleteExercise(index: string) {
    this.exercise.splice(+index, 1);
    this.exerciseChanged.next(this.exercise.slice());
  }
}
```

Slika 13 Servis za upravljanje vježbama

Servis koji omogućava prijenos jedne vježbe na određeni trening zove se „ExerciseToTrainingService“. Funkcionira tako da koristimo gumb na vježbi koju želimo postaviti na neki trening i tada se *id* te vježbe spremi u varijablu. Nakon toga na kratici trening na kojem želimo imati tu vježbu postavimo željenu vježbu (Slika 14).

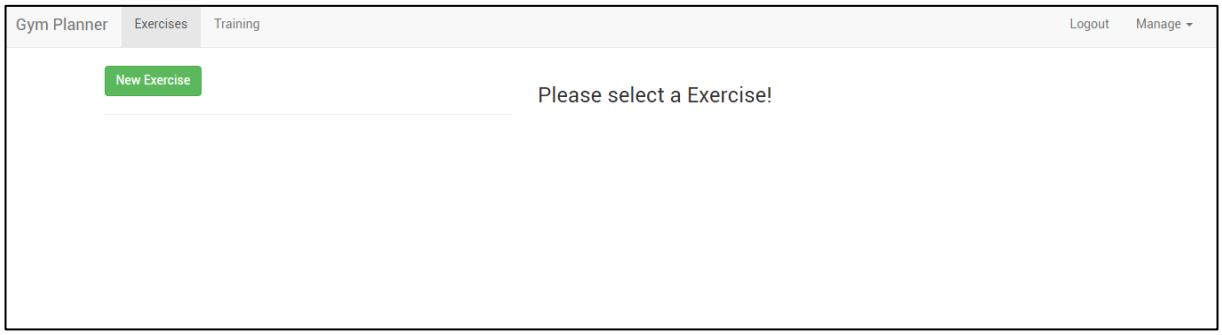
```
You, 3 weeks ago | 1 author (You)
@Injectable({ providedIn: 'root' })
export class ExerciseToTrainingService{

    private exercise: Exercise[] = [];

    exerciseChanged = new Subject<Exercise[]>();

    pullExercise(exercise: Exercise) {
        this.exercise.pop()
        this.exercise.push(exercise);
        this.exerciseChanged.next(this.exercise.slice());
      }

    pushToTraining(){
        return this.exercise[0]
    }

}
```

Slika 14 Servis za upravljanje treninzima

Nakon prikaza dijelova *feature* module, možemo krenuti na prikaz glavnih komponenti cijele aplikacije, a to su „Exercise“ komponenta i „Training“ komponenta.

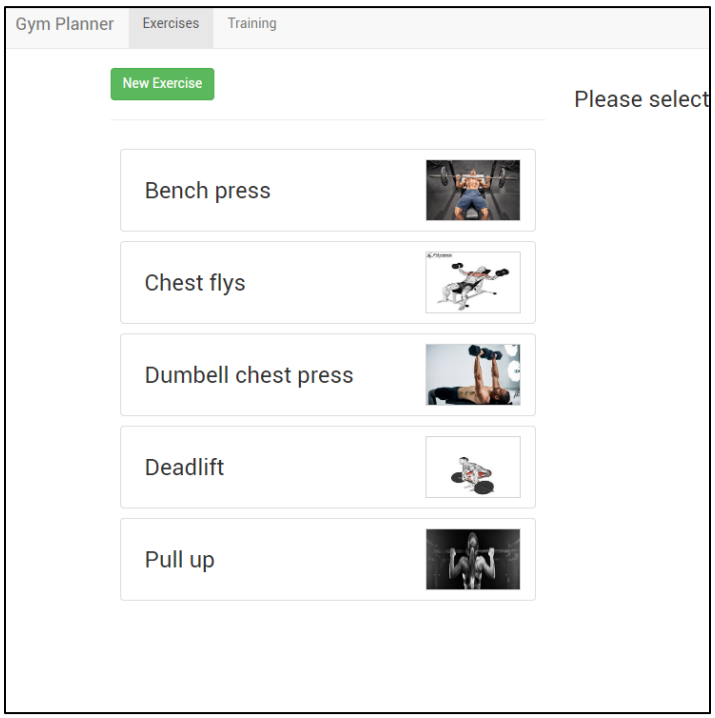
3.3.1.“Exercise“ komponenta

Kada na navigacijskoj traci kliknemo gumb na kojem piše „Exercises“, otvora nam se prazna stranica na kojoj vidimo tekst i gumb koji nam služi za dodavanje nove vježbe (Slika 15).



Slika 15 Lista vježbi kada je prazna

Za dohvat podataka idemo desno gore na padajući izbornik „Manage“ i pritisnemo „Fetch data“, koji pokreće funkciju „onFetchData()“ i time dobivamo podatke koje smo spremili u bazu u kolekciju „exerciseList“(Slika 16).



Slika 16 Lista vježbi nakon dohvata podataka

Ovaj prikaz je napravljen pomoću dvije HTML komponente - jedna služi za prikaz same liste (Slika 17) dok druga služi za popunjavanje svakog objekta u listi s podacima (Slika 18).

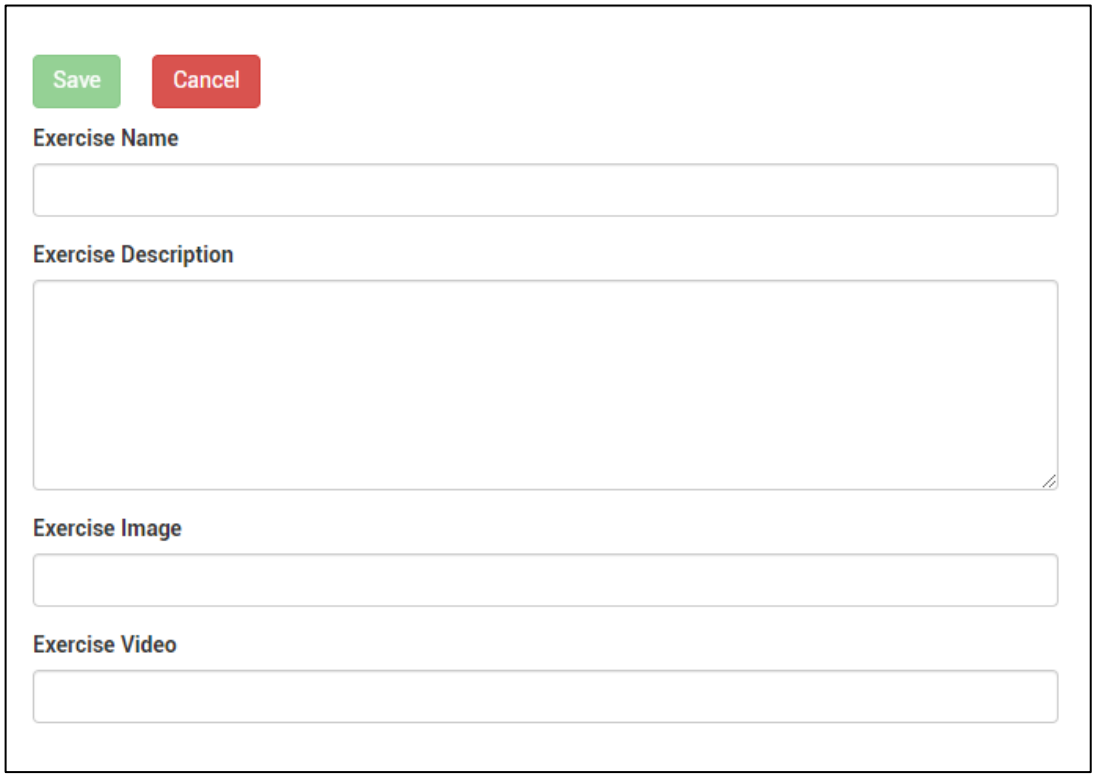
```
You, 3 weeks ago | 1 author (You)
<div class="row">       You, 3 weeks ago • first commit
  <div class="col-xs-12">
    <button class="btn btn-success" (click)="onNewExercise()">New Exercise</button>
  </div>
</div>
<hr>
<div class="row">
  <div class="col-xs-12">
    <app-exercise-item
      *ngFor="let exerciseEl of exercises; let i = index"
      [exercise]="exerciseEl"
      [index]="i"></app-exercise-item>
  </div>
</div>

```

Slika 17 HTML komponenta za prikaz liste

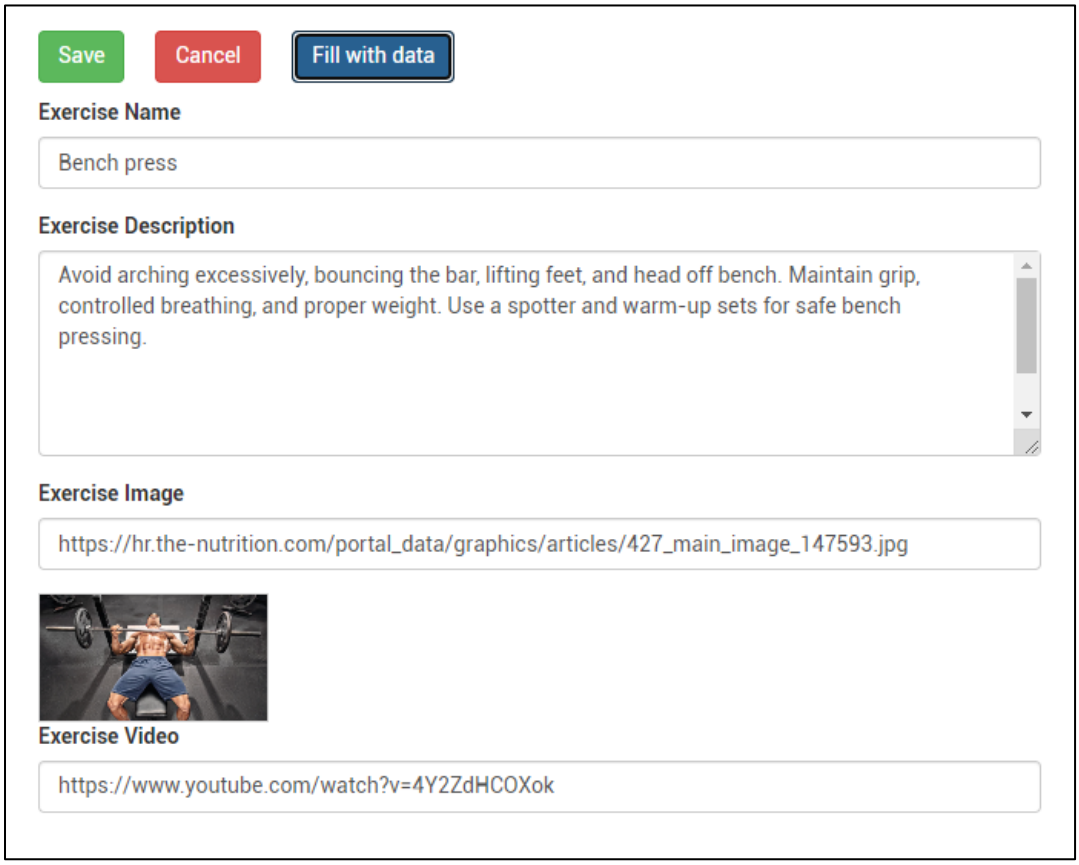
```
You, last week | 1 author (You)
<a       You, 3 weeks ago • first commit
  style="cursor: pointer;"
  [routerLink]="[index]"
  routerLinkActive="active"
  class="list-group-item clearfix spacing list-hover">
  <div>
    <div class="pull-left">
      <h3 class="list-group-item-heading custom-align-text-center">{{ exercise?.exerciseName }}</h3>
    </div>
    <span class="pull-right">
      <img [src]="exercise?.exerciseImage" class="img-responsive" style="height: 65px; width:100px;" />
    </span>
  </div>
</a>
```

Slika 18 HTML komponenta za ispunjavanje i prikaz svakog objekta liste

Kada kliknemo na gumb „New Exercise“, pokreće se funkcija koja otvara novu komponentu. Ta komponenta sadrži formu gdje unosimo podatke za novu vježbu ili mijenjamo podatke već postojeće vježbe. Ovisno o tome koji parametar šaljemo (može biti *new* za novu vježbu (Slika 19), ili *edit* kada mijenjamo postojeću vježbu (Slika 20)).

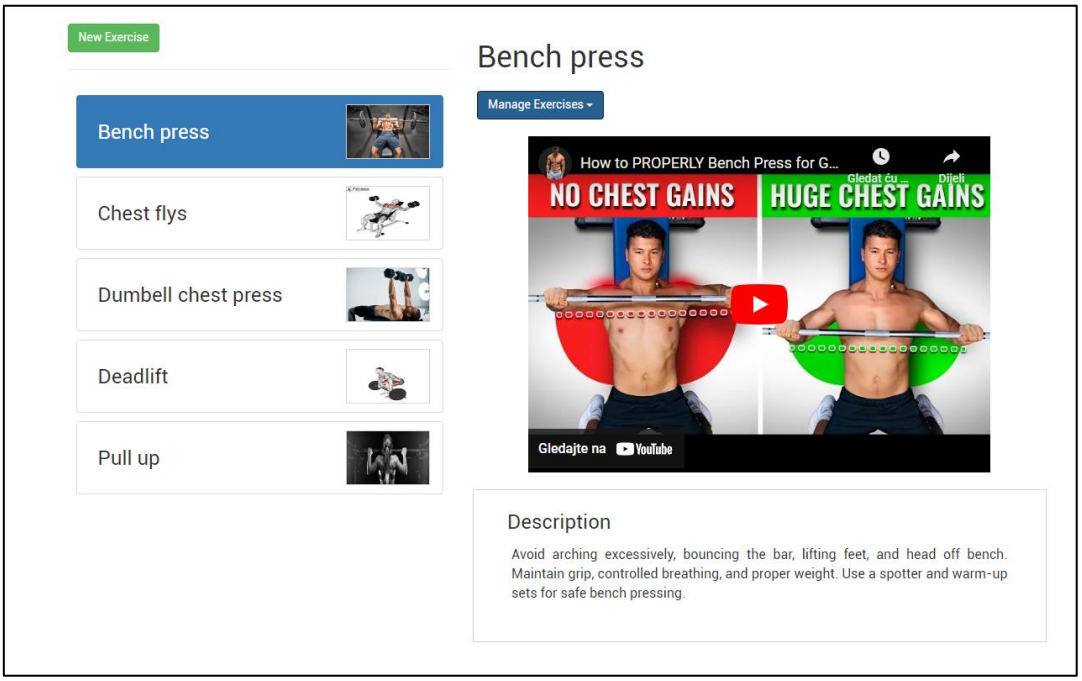


Slika 19 Komponenta za dodavanje nove vježbe



Slika 20 Komponenta za mijenjanje postojeće vježbe

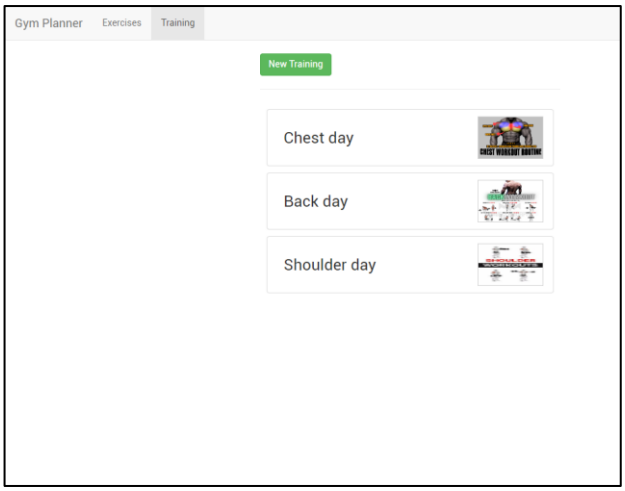
Kada kliknemo na jednu karticu na listi vježbi (Slika 16) otvaraju nam se detalji za tu vježbu (Slika 21). Na toj se komponenti prikazuje naziv vježbe, padajuća lista s opcijama za brisanje, izmjenu podataka ili spremanje promjena u aplikaciju za dodavanje na neki trening, te video za tu vježbu i opis vježbe.



Slika 21 Prikaz detalja za pojedinu vježbu

3.3.2. "Training" komponenta

Training komponenta je komponenta za upravljanje treninzima. Moguće je dodavati nove treninge i unutar njih odabrane vježbe za pojedini trening. Izgled i funkcionalnosti prvih par podkomponenti komponente „Training" slični su kao i kod „Exercise" komponente. Za dohvaćanje podataka, treba pritisnuti na padajuću listu u gornjem desnom kutu te odabrati „Fetch Data". (Slika 22).

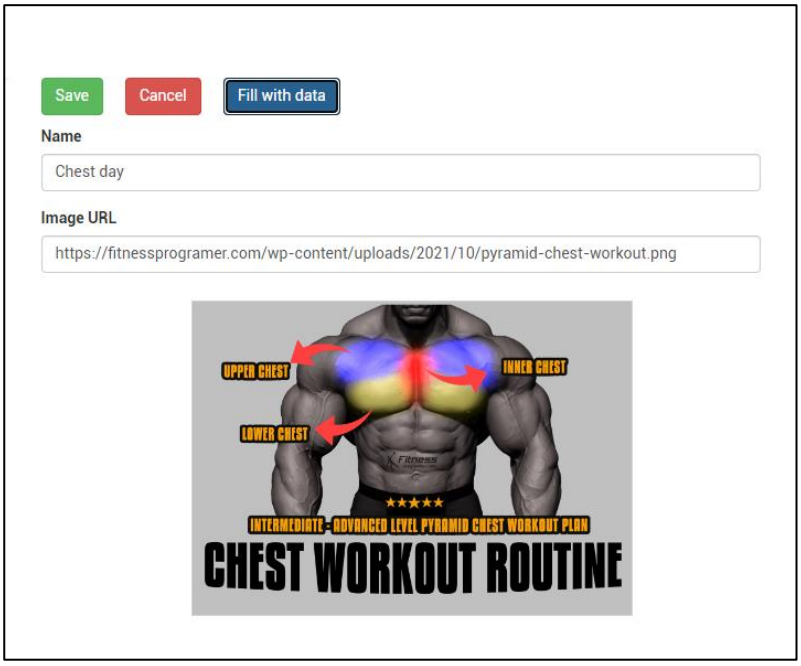


Slika 22 Lista treninga nakon dohvata podataka

Kod izrade novog treninga ispunjavaju se polja „Name“ u koji se upisuje naziv treninga i „Image URL“ pomoću kojeg se dodaje slika radi lakše prepoznatljivosti. (Slika 23)

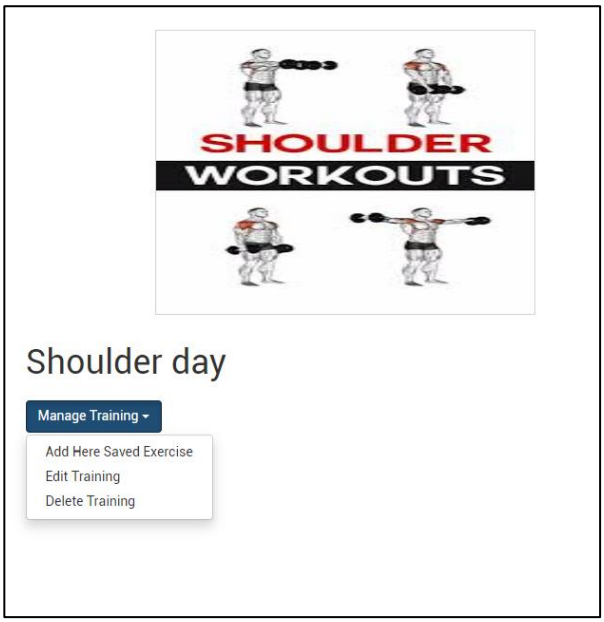
Slika 23 Forma za izradu novog treninga

Kod uređivanja treninga dobijemo istu formu kao i kod izrade novog treninga, samo što uz to dobijemo i gumb za ispunjavanje polja s već postojećim podacima koje želimo urediti (Slika 24).



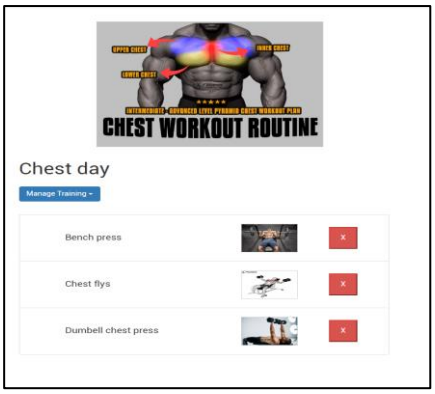
Slika 24 Forma za uređivanje treninga

Nakon izrade novog treninga, dobijemo desno pokraj liste treninga, sliku toga treninga i jednu padajuću listu s gumbovima za brisanje („Delete Training“), uređivanje („Edit Training“) te dodavanje odabrane vježbe („Add Here Saved Exercise“) (Slika 25).



Slika 25 Izgled novo izrađene vježbe

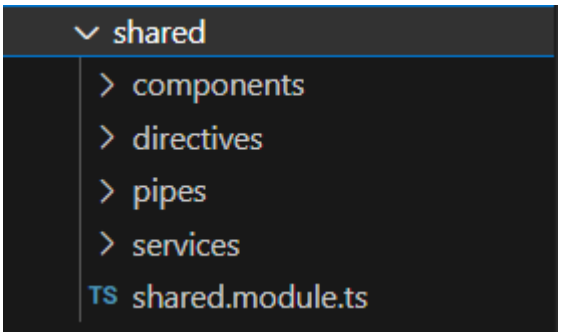
Kako bismo dodali željenu vježbu u trening, prvo moramo otići na vježbu koju želimo postaviti. Uđemo u detalje te vježbe klikom na karticu odaberete vježbe te na padajućoj listi odabiramo „Add to Training“ čime se vježba spremi u međuspremnik. Nakon toga odemo na „Training“ tab preko navigacijske trake te odaberemo karticu treninga na kojeg želimo dodati tu vježbu, otvorimo padajuću listu i odabiremo „Add Here Saved Exercise“. Dodavanjem više vježbi korisnik sam izrađuje trening kakav želi i koji se može prilagoditi njegovim potrebama. (Slika 26)



Slika 26 Izgled treninga nakon dodavanja vježbi

3.4. "Shared" modul

Slično kao „Feature" modul, „Shared" modul također može sadržavati *component*, *model* i *services* mape, ali one nisu obavezne (Slika 27). Međutim u ovom se modulu mogu pojaviti neke dodatne mape u kojima dodajemo posebne funkcionalnosti koje možemo koristiti u cijeloj aplikaciji. Dakle „Shared" modul predstavlja modul u kojeg stavljamo sve funkcionalnosti koje koristimo u cijeloj aplikaciji, ali to nisu osnovne funkcionalnosti koje aplikacija pruža, već primjerice za one omogućavanje prikaza podataka, *popup* dijaloga i slično.



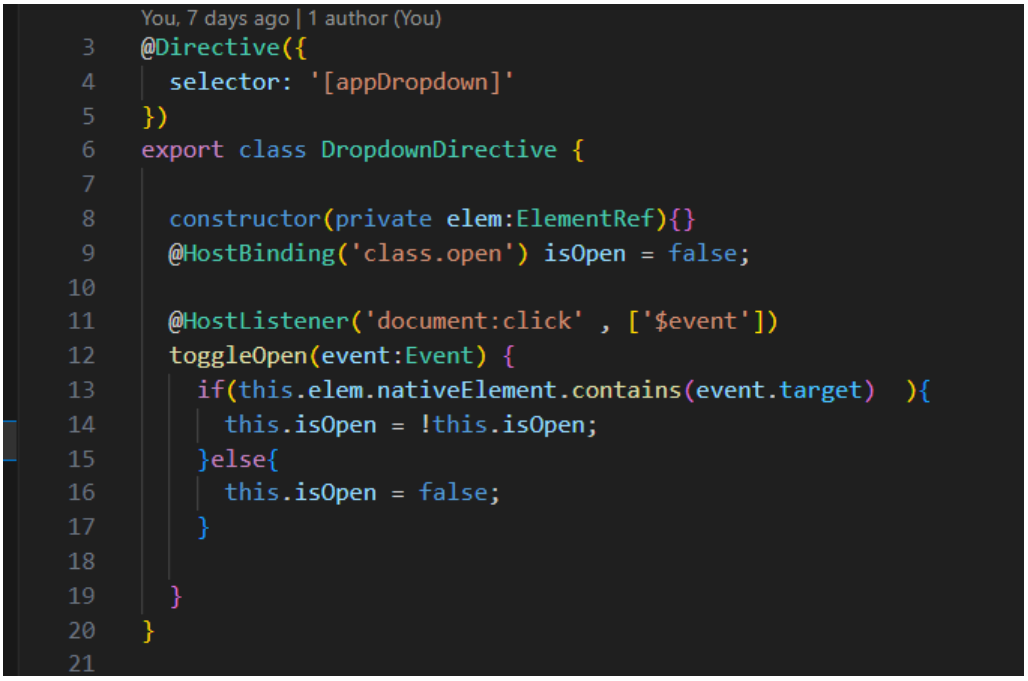
Slika 27 Izgled shared modula

U mapi *component*, nalaze sve dvije komponente, jedna je *loader* koji se pojavljuje kada treba duže vrijeme kod učitavanja, a druga je *popup* dijalog koji se pojavi kada se dogodi greška (Slika 28).

Slika 28 Greška kod prijave

Preostale su nam mapa s direktivama, servisima i pipe-ovima. Mapa direktive u sebi sadrži dvije direktive[3], jedna je za padajući meni, dok je drugi *placeholder* za nazive. (Slika 29).

Kod mape *Pipe* imamo samo jednu datoteku, s kojom je napravljen pipe[4] koji omogućava prikaz Youtube videa na aplikaciji.



Slika 29 Direktiva za padajući meni

[3] Postoje nekoliko built-in direktiva u Angularu, ali mogu se napraviti i vlastite, one su zapravo funkcije koje postavimo unutar HTML-a, a izvršavaju se kada se taj dio kompajlira [12]

[4] Pipe-ovi su funkcije koje dodajemo pored naših varijabla/podataka unutar HTML-a te pipe-ovi manipuliraju/transformiraju varijable/podatke unutar HTML-a kako je u njima zadano [13]

I zadnja mapa, *services*, sadrži pozive API-ju za dohvat i spremanje podatka s i u Firebase bazu. Preko poziva dobivamo objekt koji se onda mapira i dohvaća se podatak po podatak koji se spremaju u obliku lista, dok se za spremanje dohvaća lista u kojoj se nalaze svi novouneseni podaci i zatim šalju na server (Slika 30).

```
storeTrainings() {
  let trainings = this.trainingService.getTrainings();
  if(trainings.length !=0){
  this.http
    .put(
      'https://gym-planner-34d64-default-rtdb.europe-west1.firebasedatabase.app/trainingList.json',
      trainings
    ).subscribe()
    this.zone.run(() => {
      this.snackBar.open('Successful stored all trainings','',{
      duration: 2000,
      verticalPosition: 'top',
      panelClass:['success']
    })
  });
    trainings =[]
  }
}

fetchExercises() {       You, 3 weeks ago • first commit
  return this.http
    .get<Exercise[]>(
      'https://gym-planner-34d64-default-rtdb.europe-west1.firebasedatabase.app/exerciseList.json'
    )
    .pipe(
      map(exercises => {
        for(let e in exercises){
          exercises[e].id = e
        }
        return exercises.map(exercise => {
          return {
            ...exercise
          };
        });
      }),
      tap(exercises => {
        this.exerciseService.setExercises(exercises);
        this.zone.run(() => {
          this.snackBar.open('Successful fetched all exercises','',{
          duration: 2000,
          verticalPosition: 'top',
          panelClass:['success']
          })
```

Slika 30 Dohvat i spremanje podataka preko API-ja

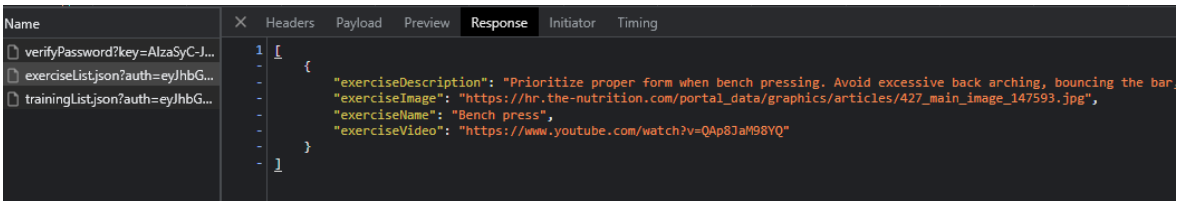
3.5. Implementacija Firebase funkcionalnosti u aplikaciju „Gym Planner“

Ranije je objašnjeno što je Firebase no nije prikazano na koji način implementirane funkcionalnosti Firebase platforme prilikom izrade „Gym Planner“ aplikacije.

Na platformi Firebase se nude dvije vrste baza podataka: Firestore i Realtime baza podataka. Svaka od tih baza ima svoje prednosti i mane. Za izradu aplikacije „Gym Planner“ korištena je Realtime bazu podataka pošto nemamo složenih upita,a olakšana je obrada podataka u Angularu.

Komunikacija između „Gym Planner“ i Firebase baze ide preko definiranih API-ja. Kada pošaljemo PUT poziv da spremimo podatke koje smo unijeli u aplikaciju, u bazi podataka stvori se novi objekt po uzorku koji je dobila u payload-u iz aplikacije (Slika 31). Isto tako dobivamo objekt kada povučemo podatke iz baze da prikažemo u aplikaciji (Slika 32).

Slika 31 Izgled objekta spremljenog u Firebase bazu



Slika 32 Izgled objekta kod GET poziva

Uz autentifikaciju koja je trenutno implementirana preko email/password registracije i prijave, postoji još jedna funkcionalnost koja se koristi za prepoznavanje podataka pojedinog korisnika. Kada se izvršava bilo kakav poziv prema Firebase bazi, postoji opcija da se postave pravila za ovlasti kako bi se ograničio pristup pojedinim podacima i radnjama. U bazi podataka „Gym Planner“ aplikacije postavljena su pravila tako da korisnik može vidjeti i mijenjati samo svoje podatke (Slika 33). Korisnike raspoznajemo po ID-ju koji im je Firebase dodijelio kod registracije u aplikaciju.

```
{
  "rules": {
    "$uid": {
      "exerciseList": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      },
      "trainingList": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    },
  }
}
```

Slika 33 Firebase pravila za zaštitu podataka korisnika

4. Zaključak

Glavni zadatak završnog rada bila je izrada web aplikacije za lakšu izradu plana za treniranje. Izrađena aplikacija „Gym Planner“ nudi korisniku mogućnosti izrade liste vježbi i treninga. U vježbama korisnik unosi ime, opis i video vježbe te sliku vježbe da se lakše raspoznaje u listi, a u treninzima korisnik unosi ime i sliku treninga te nakon toga može dodavati ranije dodane vježbe u trening koji želi. Na taj se način korisniku omogućuje personalizacija treninga i mogućnost prilagodbe vlastitim željama i potrebama kod izvođenja vježbi.

Za izradu web aplikacije korišteni su Angular za frontend dio, platforma Firebase za backend dio aplikacije i bazu podataka te Bootstrap za dizajn. Već neko vrijeme koristim Angular za izradu web aplikacija pa nisam imao nekih većih komplikacija kod izrade frontend dijela. Bootstrap alat je jednostavan za korištenje i dobro potkrijepljen dokumentacijom. Što se Firebase platforme tiče, već sam ju ranije koristio no sada sam morao malo proširiti svoje znanje i istražiti način implementacije Firebase alata u Angular aplikaciju. Za svaki dio koji sam koristio na Firebase platformi postoji opširna dokumentacija koja je puno pomogla.

Daljnji plan s „Gym Planner“ aplikacijom je napraviti moderniji dizajn, dodati nekoliko funkcionalnosti te admin korisnika.

5. Prilozi

Kode aplikacije : https://github.com/Mrkimedo69/Gym_Planner

Link aplikacije: https://gym-planner-34d64.web.app/

6.Literatura

[1] >>Angular<< , https://angular.io/docs, [Pokušaj pristupa Rujan 2023]

[2] Wikipedia, >> Angular (web framework)<<, https://en.wikipedia.org/wiki/Angular_(web_framework) , [Pokušaj pristupa Rujan 2023]

[3] InterviewBit, Angular Architecture – Detailed Explanation, https://www.interviewbit.com/blog/angular-architecture/, [Pokušaj pristupa Rujan 2023]

[4] C. McKenzie, >>JavaScript vs. TypeScript: What's the difference?<<, https://www.theserverside.com/tip/JavaScript-vs-TypeScript-Whats-the-difference, [Pokušaj pristupa Rujan 2023]

[5] Wikipedia ,>>API<<, https://hr.wikipedia.org/wiki/API , [Pokušaj pristupa Rujan 2023]

[6] Quora, >>DO Mark Otto and Jacob Thornton work on Bootstrap full-time at Twitter?<<, https://www.quora.com/Do-Mark-Otto-and-Jacob-Thornton-work-on-Bootstrap-full-time-at-Twitter, [Pokušaj pristupa Rujan 2023]

[7] Wikipedia, >>Bootstrap (front-end framework)<<, https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)#Bootstrap_3, [Pokušaj pristupa Rujan 2023]

[8] Firebase, >>Documentation<<, https://firebase.google.com/docs?gad=1&gclid=Cj0KCQjw0IGnBhDUARIsAMwFDLnC5unDnguwsjIO6kYYMgO_4wB8VQkF42_tRcHff2efftw5wB9r-fIaAlXaEALw_wcB&gclsrc=aw.ds, [Pokušaj pristupa Rujan 2023]

[9] Firebase, >>Firebase Realtime Database<<, https://firebase.google.com/docs/database, [Pokušaj pristupa Rujan 2023]

[10] David L, Stack Overflow, >>How long does a Firebase Auth session last when the user is offline ?<<, https://stackoverflow.com/questions/65397458/how-long-does-a-firebase-auth-session-last-when-the-user-is-offline, [Pokušaj pristupa Rujan 2023]

[11] >>Component LifeCycle<<, https://angular.io/guide/lifecycle-hooks, [Pokušaj pristupa Rujan 2023]

[12] Simplelearn, >>What Are Directives in Angular?<<, https://www.simplilearn.com/what-are-directives-in-angular-article, [Pokušaj pristupa Rujan 2023]

[13] >>Transforming Data Using Pipes<<, https://angular.io/guide/pipes, [Pokušaj pristupa Rujan 2023]

7. Popis slika