Razvoj web aplikacije za praćenje napredovanja u fitness centru

Vila, Leon

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

Leon Vila

RAZVOJ WEB APLIKACIJE ZA PRAĆENJE NAPREDOVANJA U FITNESS CENTRU

Završni rad

Pula, rujan 2020. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

Leon Vila

RAZVOJ WEB APLIKACIJE ZA PRAĆENJE NAnapredovanja U FITNESS CENTRU

Završni rad

JMBAG: 0316001376
Studijski smjer: Informatika
Kolegij: Programiranje
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Mentor: izv. prof. dr. sc. Tihomir Orehovački

Pula, rujan 2020. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Leon Vila, ovime izjavljujem da je ovaj završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student
Leon Vila

U Puli, rujan 2020. godine

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, dolje potpisani Leon Vila, ovime dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom "Razvoj web aplikacije za praćenje napredovanja u fitness centru" koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišta Jurja Dobrile u Puli te kopira javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Student
Leon Vila

U Puli, rujan 2020. godine

SAŽETAK

Cilj ovog završnog rada je dokumentirati izradu jedne jednostavne web aplikacije. Sama aplikacije trebala bi biti od pomoći korisnicima fitnessa u praćenju osobnih vježbi i treninga. Naime, korisnici bi mogli upisivati vježbe i dokumentirati treninge. Aplikacija će biti odvojena na frontend i backend dio. Frontend dio sadržavati će sve glavne funkcionalnosti, dok će backend služiti kao posrednik između frontenda i baze podataka.

The goal of this bachelor's thesis is to document a development process for a simple web application. This application is supposed to help people that are interested in fitness activities by giving them the ability to view and document their exercises. This application will be made of two parts: frontend and backend. The frontend will host the main functionalities, while the backend is going to act as a moderator between frontend and the database.

SADRŽAJ

1 Uvod

Internet je danas glavni medij komunikacije koji je uključen u gotovo sve aspekte naših života. Koristi se za gledanje filmova i serija, čitanje knjiga, članaka, dnevnih novosti, online kupovinu, dogovaranje druženja, traženje lokalnih restorana, za potrebe posla itd. Mnoge od tih aktivnosti ne bi bile moguće bez dinamičkog web programiranja. Dinamičko web programiranje ukratko znači da web stranica ima server i bazu u koju se mogu upisivati i preuzimati podatci. Web aplikacije su isto tako postale jako popularne zbog još jedne ključne stvari - ne zahtijevaju posebnu instalaciju na korisnikov uređaj. Za razliku od klasičnih programa, web aplikacije pokreće web preglednik. To uvelike olakšava korištenje i dopremu programa do njegovih korisnika. Zbog svih navedenih korisnosti i općenito važnosti dinamičkih web aplikacija za naše trenutno društvo, u nastavku rada bit će napravljena jedna takva aplikacija.

Ideja za ovu aplikaciju dolazi iz potrebe da se na jednostavan način spremaju vježbe i odrađeni treninzi za sportaše i korisnike fitnessa. Trenutno na tržištu postoje razne fitness web aplikacije kao što su Muscle & Strength, Muscle & Fitness, MyFitnessPal, te mnoge druge. Navedene aplikacije uistinu jesu vrlo korisne i informativne. Svaka od njih sadrži sekcije s mnogobrojnim člancima vezanim za fitness industriju, sekcije za vježbe, sekcije za planiranje prehrane, sekcije sa suplementima, razne alate i kalkulatore, te još mnogo toga drugog. Međutim, količina informacija i funkcionalnosti na tim stranicama može smetati nekim korisnicima. Primjerice, običnom korisniku često je samo potrebna mogućnost pregleda i unosa vježbi, te dokumentacija i praćenje odrađenih treninga. Ostatak funkcionalnosti je samo nepotrebna smetnja čija kompleksnost može biti obeshrabrujuća.

U sljedećem poglavlju bit će u grubo objašnjene sve aktivnosti koje je potrebno napraviti prilikom razvoja web aplikacije. Nakon toga, u trećem poglavlju biti će objašnjena implementacija pojedinih funkcionalnosti frontend dijela aplikacije. U četvrtom poglavlju bit će objašnjen njen backend, te korištena baza podataka. Za kraj će biti iznesen zaključak u kojem će se analizirati količina rada prilikom izrade web aplikacije i mogućoj podjeli u timove, te o mogućim nadogradnjama i proširenjima za napravljenu aplikaciju.

2 Plan razvoja softvera

Razvoj softvera može se podijeliti u manje skupine aktivnosti koje trebaju biti obavljene kako bi se dobio gotov softverski produkt. Te aktivnosti su: utvrđivanje zahtjeva, dizajn, implementacija, testiranje, isporuka, održavanje [1]. Navedene aktivnosti mnogi još nazivaju i životnim ciklusom aplikacije. To ime proizlazi iz činjenice da prilikom svakog poboljšanja aplikacije, developerski tim mora proći kroz sve korake izrade ponovno. Ponekad i kada je neki korak konstruktivno zanemaren ili izostavljen iz plana, on će biti obavljen bez da programer o njemu razmišlja. Primjerice, programer će prilikom pisanja koda aktivno isprobavati rezultate kompajlirajući kod i time u jednu ruku testirati napravljenu implementaciju. Naravno to "testiranje" neće biti ni blizu kvalitetno kao testiranje koje bi bilo obavljeno s većim uloženim trudom i vremenom u njega.

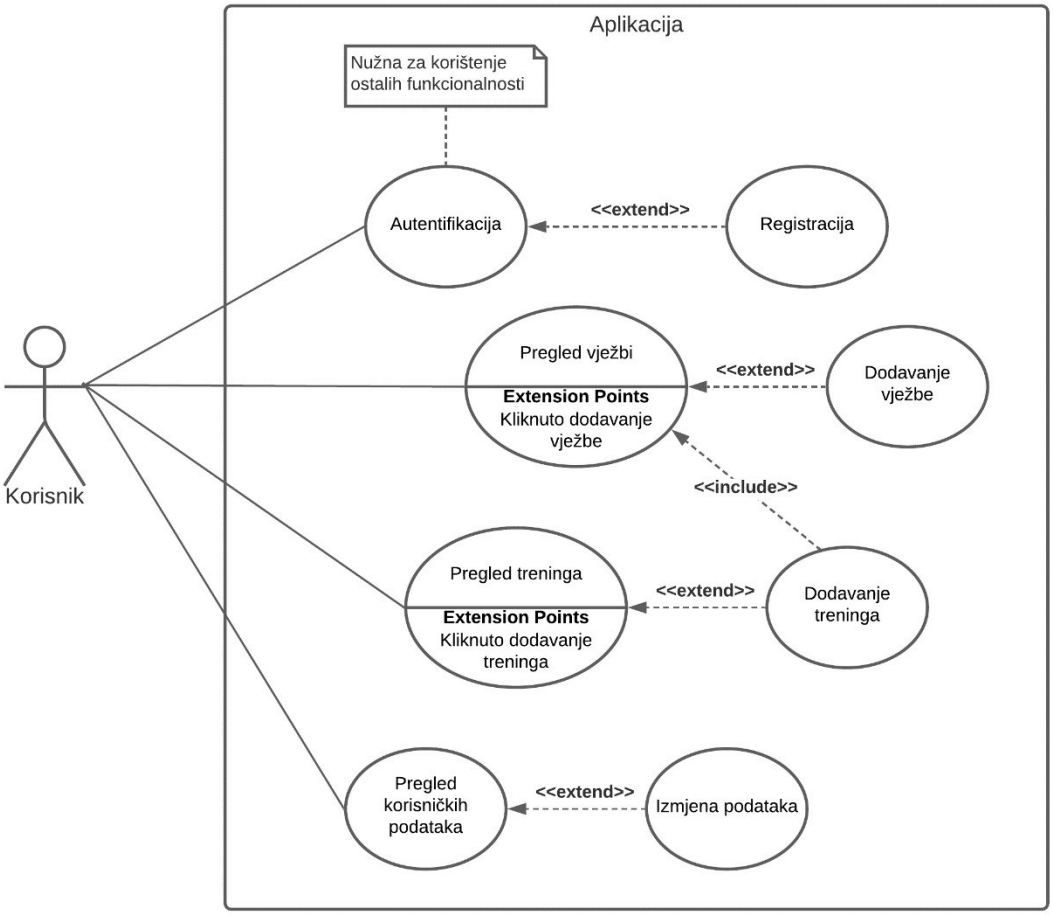
2.1 Prikupljanje zahtjeva

U fazi prikupljanja zahtjeva izrađuje se lista mogućnosti i ciljeva koje aplikacija treba obavljati. Ovaj korak može uključivati analizu tržišta, provođenje anketa, razgovor s naručiteljima, itd. Lista funkcionalnosti može biti vrlo opsežna i kompleksna, za ovaj projekt ona će u drugu ruku biti vrlo sažeta i jednostavna.

U srži fitness aplikacija koja će biti izrađena trebati će raditi sljedeće:

- Korisnik mora napraviti račun na servisu kako bi ga koristio
- Korisnik može mijenjati osobne podatke i zaporku
- Korisnik može pregledavati vježbe
- Korisnik može dodavati i brisati svoje vježbe
- Korisnik može dodavati i brisati svoje treninge
- Korisnik može pregledavati svoju statistiku treninga

U fazi utvrđivanja zahtjeva poželjno je koristiti i vizualne modele kao naprimjer use-case dijagram. Dijagrami pomažu pri lakšem razumijevanju problema ili struktura koje prikazuju. Isto tako, dijagrami su najkorisniji alat kada neki član tima želi podijeliti zamisao sa drugim članovima tima. Prema prije navedenim točkama koje ova aplikacija treba sadržavati napravljen je jedan use-case dijagram (Slika 1).

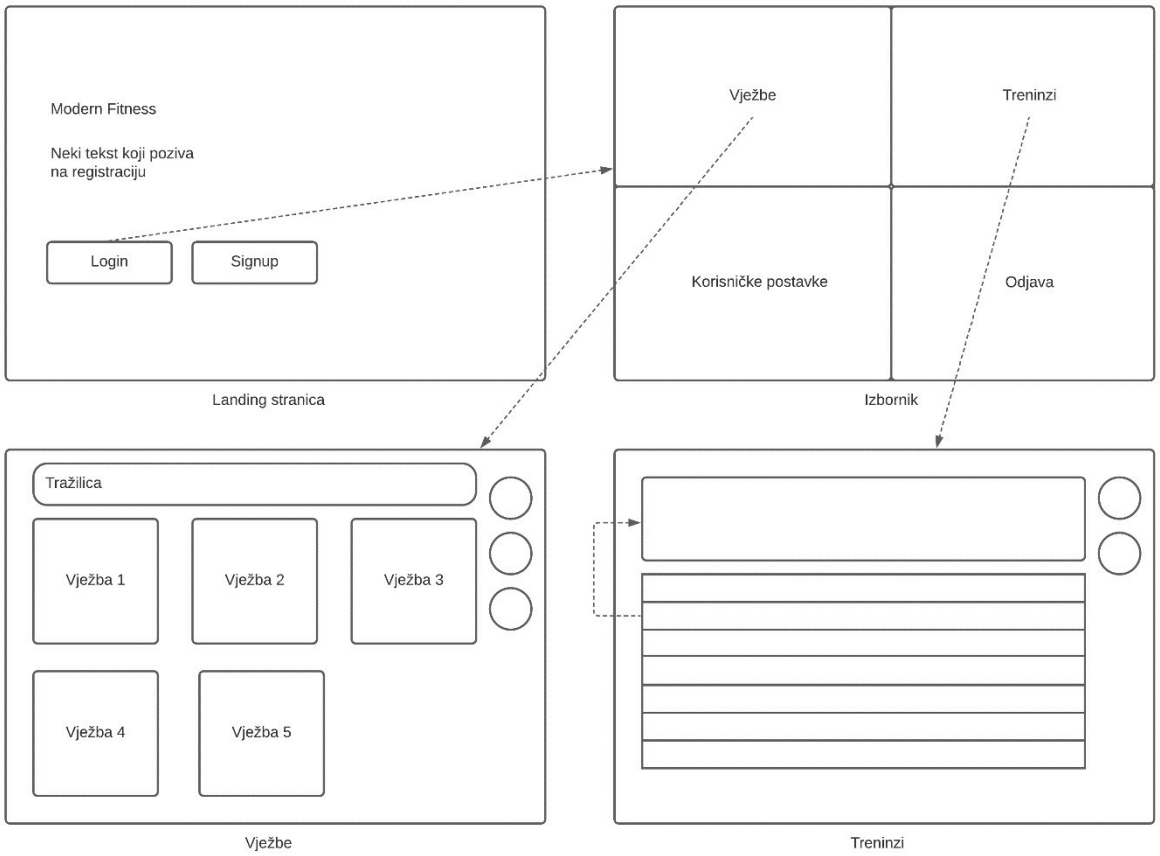


Slika 1 - Use-case dijagram

Često se u timskim okruženjima zanemaruje važnost preciznog definiranja problema i rješenja. No vrijeme i trud koji su "ušteđeni" na ovom koraku će vrlo vjerojatno doći na naplatu u kasnijim fazama razvoja. Jedna od dužnosti vođe tima je upravo ta da se potrudi da svaki član tima zna što treba napraviti i kako. Naravno, postoje šanse da će tokom razvoja softvera doći do kojekakvih promjena u kasnim fazama izrade, no to ne znači da treba početi rad bez dobro definiranog početka.

2.2 Dizajn

Nakon što je definirano što je potrebno napraviti vrijeme je za dizajn. U ovoj fazi se prvo rade grube skice izgleda programa. Slika 2 prikazuje skicu web aplikacije. Kao što se vidi iz priloženog ona nije previše detaljna, ali pokazuje dovoljno da iznese okvirnu ideju o rasporedu elemenata na pojedinim stranicama.



Slika 2 - Primjer grube skice programa

Nakon što je gruba skica gotova, može se još napraviti mockup i prototip. Mockup je detaljniji dizajn izgleda pojedinog dijela aplikacije, dok je prototip interaktivni skup mockupa. Prototip i mockup izgleda aplikacije moguće je napraviti pomoću programskih alata kao što je Adobe XD. Iako nema neku bitnu svrhu, prototip može pružiti mnogo vjerodostojniji dojam o cjelokupnom izgledu aplikacije. Za web aplikaciju u ovom radu nije bilo potrebe radili prototip.

Prije prelaska na implementaciju funkcionalnosti za aplikaciju dobro je napraviti sučelje za svaku stranicu aplikacije po primjeru napravljene skice ili mockupa. Pošto se radi o web aplikaciji, u ovom projektu korišteni su HTML i CSS jezici kako bi se napravila sučelja stranica. Slika 3 prikazuje HTML kod za početnu stranicu aplikacije. Neki čitatelji će primijetiti par neobičnih atributa unutar pojedinih tagova, ti atributi su ustvari direktive korištenog JavaScript frameworka o kojem će biti više riječi u sljedećoj sekciji. Mnoge CSS klase koje se mogu vidjeti u primjeru su sadržane u Tailwind CSS frameworku koji se isto tako koristi u ovom projektu [2].

```
<template>
  <div class="leading-normal tracking-normal text-gray-900">

    <LoginModal v-if="login" v-on:closeModalMessage="closeModals()" v-on:switchModalMessage="switchModals()" />
    <SignupModal v-if="signup" v-on:closeModalMessage="closeModals()" v-on:switchModalMessage="switchModals()" />

    <div class="h-screen pb-14 bg-right bg-cover bg-cool-dark bg-no-repeat" style="background-image:url('/img/hero-bg.jpg'); background-po
      <div class="container pt-24 md:pt-48 px-6 mx-auto flex flex-wrap flex-col md:flex-row items-center move-up-down">
      <div class="flex flex-col w-full xl:w-2/5 justify-center lg:items-start overflow-y-hidden">
          <h1 class="my-4 text-3xl md:text-5xl text-orange-700 font-bold leading-tight text-center md:text-left slide-in-bottom">Modern Fi
          <p class="leading-normal text-gray-300 text-base md:text-2xl text-center md:text-left slide-in-bottom">
            Do you have what it takes to be the biggest dude in the gym?
          </p>
          <p class="leading-normal text-gray-600 text-base md:text-1xl mb-8 text-center md:text-left slide-in-bottom">
            Sign up today and start getting in shape!
          </p>
          <div class="flex w-full justify-center md:justify-start pb-24 lg:pb-0 fade-in">
            <button v-on:click="login = true" class="bg-orange-500 hover:bg-orange-700 text-white font-bold py-2 px-4 rounded h-12 mr-4 sl
              Login
            </button>
            <button v-on:click="signup = true" class="bg-orange-500 hover:bg-orange-700 text-white font-bold py-2 px-4 rounded h-12 slide-
              Signup
            </button>
          </div>
        </div>
      </div>
    </div>

  </div>
</template>
```

Slika 3 - HTML kod za početnu stranicu

2.3 Implementacija

Jednom kada je gotov dizajn aplikacije vrijeme je za kodiranje svih tih funkcionalnosti koje su zamišljene još u prvom koraku izrade. Za web aplikaciju programske skripte se često pišu jezikom JavaScript. Kako bi se olakšao posao izrade korisno je koristiti programske okvire (eng. framework) koji u sebi sadrže već gotov kostur aplikacije, te biblioteke (eng. library routines) koje sadrže implementacije često korištenih rutina. U te svrhe za ovaj projekt korišteni su Vue.js framework [3], axios paket za HTTP upite, tailwindcss za dizajn, te nekoliko dodatnih paketa (Slika 4).

Slika 4 - Korišteni paketi

Za lakšu navigaciju unutar projekta, u "src" folderu napravljeni su sljedeći folderi: assets, components, router, services, views. U assets folderu stavljene su CSS datoteke. U components folderu nalaze se komponente koje se koriste na više mjesta unutar stranica. U router folderu stavljen je kod za putanje unutar domene. U services folderu nalazi se kod s upitima na backend. Za kraj u views folderu smješten je kod za pojedine stranice.

Spomenuto je da se u services folderu nalaze upiti na backend ali nije još ništa rečeno o backendu. Naime, u projektu će se koristiti MongoDB Atlas za bazu podataka, a na nju će se slati upiti preko backenda. Prvi razlog zašto nije dobro slati upite na bazu s frontenda je taj što je moguće filtrirati i manipulirati upitima bez brige o mogućim izmjenama koda s korisnikove strane. Drugi razlog je administrativne naravi pri izradi i održavanju aplikacije - lakše je odvojiti tim i kontrolirati tko ima pristup i tko je zadužen za koje dijelove koda. Backend aplikacija u ovom radu neće imati korisničko sučelje te će se sve vrtiti u konzoli. Slika 5 prikazuje pakete koji su korišteni za backend.

Slika 5 - Paketi za backend

Više o konkretnoj implementaciji pojedinih komponenti i backenda biti će riječ u nadolazećim poglavljima ovog rada.

2.4 Testiranje

Postoje mnogobrojne razine testiranja, od pojedine komponente do cijelog sustava. Isto tako postoje raznovrsne vrste testova kojima možemo testirati pojedine aspekte programa, npr. performance test, stress test, usability test i sl.

Pošto je projekt u ovom radu vrlo jednostavan i nije namijenjen za komercijalnu upotrebu, nije bilo potrebe podložili ga nikakvim rigoroznim testiranjima.

2.5 Isporuka

Kada je aplikacija gotova treba je isporučiti korisnicima. Za ovaj projekt biti će korišten besplatni servis na Heroku cloud platformi za posluživanje backenda, te Vercel za frontend. Način na koji će se projekt postaviti na te platforme je vrlo jednostavan. Za Heroku će biti potrebno instalirati Heroku command line interface, prijaviti se na račun preko interfacea, te gurnuti projekt kao novi projekt na Heroku. Nakon toga, projekt će se spremiti na njihov server i pokrenuti će "start" i "heroku-postbuild" komandu iz "package.json" dokumenta. Vercel će napraviti sličnu stvar, samo što se na njega može spremiti projekt izravno preko GitHub repozitorija.

3 Frontend

U ovom poglavlju će biti malo detaljnije dotaknut koda iza napravljenih funkcionalnosti frontenda aplikacije. Zbog praktičnih razloga neće biti objašnjen kompletan kod, već samo kod vezan uz ključne dijelove aplikacije.

3.1 Rute

U folderu router nalazi se dokument u kojem su sadržane sve potrebne informacijama o kraticama za stranice i ograničenjima za njih. Slika 6 prikazuje dio koda ove datoteke. Od linije 50 do 59 implementiran je kod za takozvanog čuvara navigacije [4], koji radi dvije stvari. Prvo provjerava prije svakog ulaza na stranicu da li je varijabla "store.authenticated" neistinita i da li postoje spremljeni "Credentials" u lokalnoj memoriji web preglednika (eng. cookies). Ako su obje stvari istinite to znači da je stranica osvježena i da se korisnik još nije odjavio. U tom slučaju će se postaviti varijabla "store.authenticated" nazad na true, te će se uzeti spremljeni "Credentials" podatci spremljeni na preglednik. Ovo je nužno napraviti zato što se prilikom svakog osvježavanja stranice sve varijable vraćaju na početne vrijednosti.

```
30    {
31      path: '/exercises',
32      name: 'Exercises',
33      meta: { requiresAuth: true },
34      component: () => import(/* webpackChunkName: "exercises" */ '../views/Exercises.vue')
35    },
36    {
37      path: '/workouts',
38      name: 'Workouts',
39      meta: { requiresAuth: true },
40      component: () => import(/* webpackChunkName: "workouts" */ '../views/Workouts.vue')
41    },
42  ]
43
44  const router = new VueRouter({
45    mode: 'history',
46    base: process.env.BASE_URL,
47    routes
48  })
49
50  router.beforeEach((to, from, next) => {
51    if (!store.authenticated && localStorage.getItem("Credentials") != null){
52      store.credentials = JSON.parse(localStorage.getItem("Credentials"))
53      store.authenticated = true
54    }
55    if (to.matched.some(record => record.meta.requiresAuth))
56      if (!store.authenticated) next({ name: 'Home' })
57      else next()
58    else next()
59  })
60
61  export default router
62
```

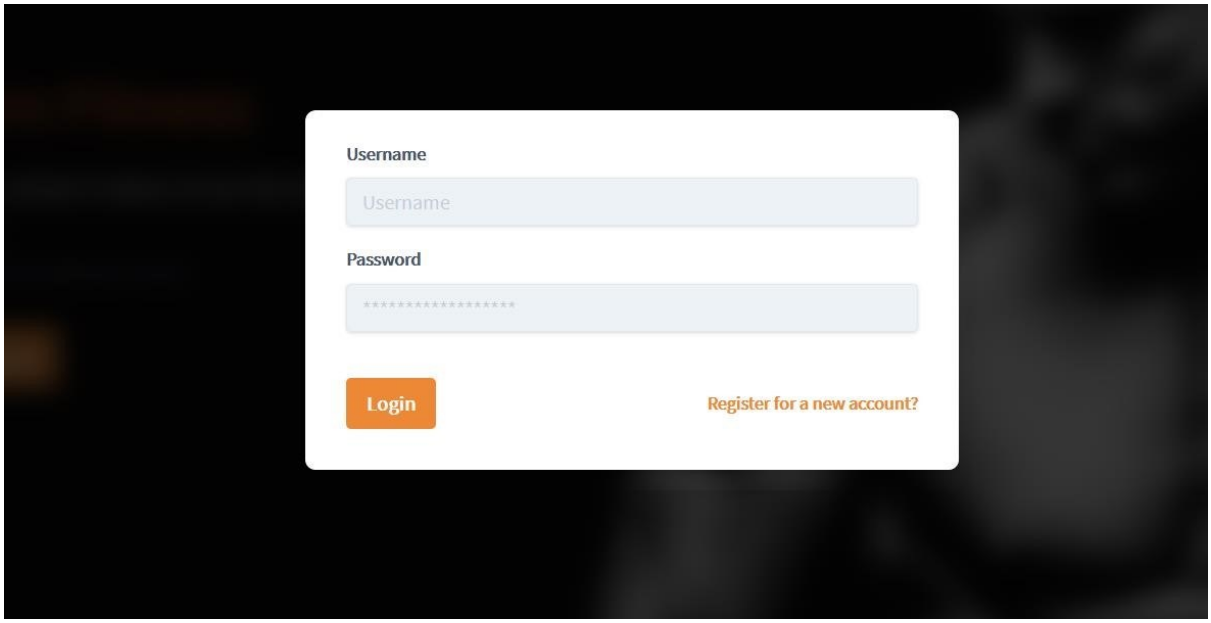
Slika 6 - Router

Druga važna uloga stražara je da preusmjeri stranicu na početnu ukoliko je varijabla "store.authenticated" i dalje neistinita nakon prve if selekcije. To će se desiti jedino u slučaju ukoliko korisnik nije bio prijavljen prije pokušaja ulaska na odabranu stranicu.

3.2 Početna stranica

Početna stranica ima vrlo malo koda. Razlog tomu je taj što je podijeljena na komponente za prijavu i registraciju. Slika 3 iz prijašnjeg poglavlja prikazuje mjesta ubacivanja tih komponenti. Komponente za prijavu i registraciju napravljene su na istom principu. Radi tog razloga biti će razmotrena samo komponenta za prijavu.

Kada se stisne na dugme za prijavu iskočiti će modal komponenta za prijavu (Slika 7). Na ovom modalu stavljena su samo 2 input elementa za upis. Na modalu za registraciju tih elemenata je puno više. Ti elementi vezani su za odgovarajuće varijable koje su instancirane u "store.js" dokumentu.



Slika 7 - Prijava

Slika 8, red 30, prikazuje uporabu v-model direktive kako bi se povezali input elementi sa store varijablom. Nekoliko redova ispod vidjiva je upotreba "v-on" event handlera za pozivanje željenih metoda klikom na pojedine elemente [5]. Slika 9 pokazuje definicije tih metoda. Ako se uzme login metoda za primjer, vidljivo je da će ona prvo staviti "loading" varijablu na true (ona nam služi kao prekidač za prikaz učitavanja) te pozvati novu login metodu u servisima koja će poslati željeni upit na backend (Slika 10).

```
25          </div>
26          <div class="mb-6">
27            <label class="block text-gray-700 text-sm font-bold mb-2" for="password">
28              Password
29            </label>
30            <input v-model="store.credentials.user.password" class="shadow appearance-none border rounded w-full py-2 px-3 text
31          </div>
32          <div class="flex items-center justify-between">
33            <button v-on:click="login()" class="bg-orange-500 hover:bg-orange-700 text-white font-bold py-2 px-4 rounded focus:
34              Login
35            </button>
36            <a v-on:click="switchModal()" class="inline-block align-baseline font-bold text-sm text-orange-500 hover:text-orang
37              Register for a new account?
38            </a>
```

Slika 8 - Vue event handleri

```
54  data() {
55    return {
56      store,
57      loading: false,
58      error: ''
59    }
60  },
61
62  methods: {
63    closeModal() {
64      this.$emit('closeModalMessage')
65    },
66    switchModal() {
67      this.$emit('switchModalMessage')
68    },
69    async login() {
70      try{
71        this.loading = true
72        await service.login()
73        this.$router.replace({ name: "Menu" })
74      }
75      catch (error) {
76        this.error = error.response.data
77      }
78      finally {
79        this.loading = false
80      }
```

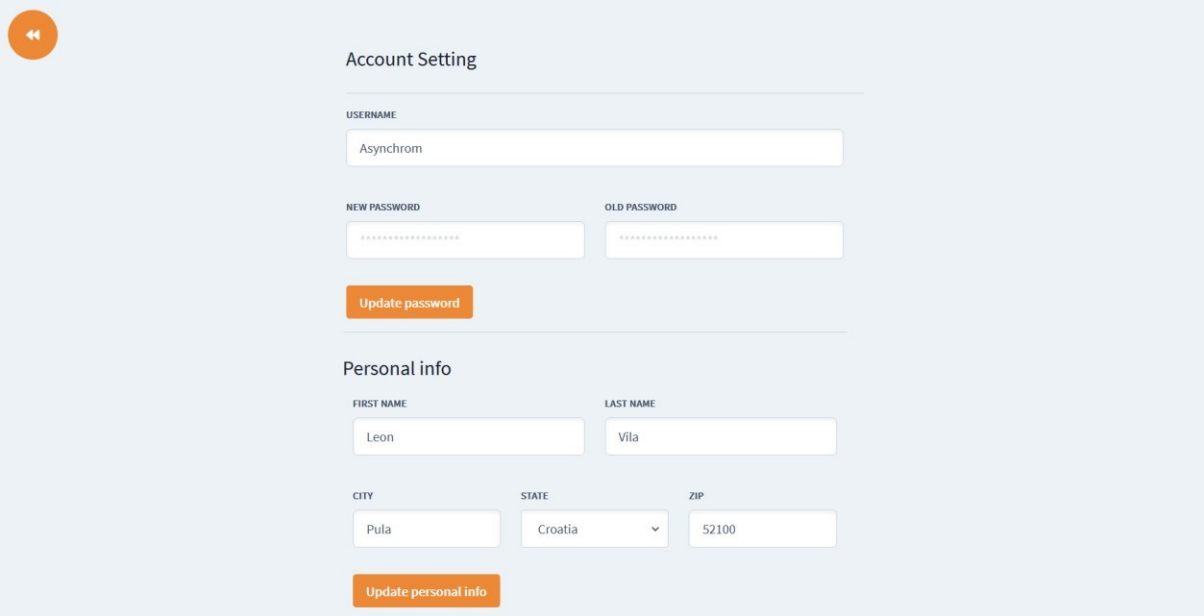
Slika 9 - Procedure na stranici za prijavu

```
17    },
18    async login() {
19        let response = await Service.post("/users/login", store.credentials.user)
20        store.credentials.user = response.data.user
21        store.credentials.token = response.data.token
22        localStorage.setItem("Credentials", JSON.stringify(store.credentials))
23        store.authenticated = true
24    },
```

Slika 10 - Login upit na backend

3.3 Korisničke postavke

Stranica sa korisničkim postavkama (Slika 11) služi za prikaz trenutnih postavki, te za njihovu izmjenu.



Slika 11 - Korisničke postavke

Elementi za inpute vezani su za pojedine store varijable na isti način kao i elementi za prijavu u prethodnom odlomku. Slika 10 iz prethodnog odlomka isto tako prikazuje da backend u odgovoru vraća token (više u njemu u backend poglavlju) i objekt koji sadržava korisničke podatke. Oni se odmah pri pristizanju spremaju u lokalnu memoriju web preglednika. Razlog zašto je to bitno objašnjen je kod čuvara ruta (3.1 Rute). Osobni podatci dobiveni prilikom prijave vidljivi su na ovoj stranici.

Slika 12 prikazuje upite koji se šalju klikom na dugmad za izmjenu. Prilikom izmjene šifre nije potrebno ažurirati podatke u memoriji web preglednika pošto se u kolačićima ne sprema korisnikova šifra. Izmjene na ostalim podatcima biti će spremljene kako bi se mogle točno prikazati za vrijeme trenutne prijavljene sesije.

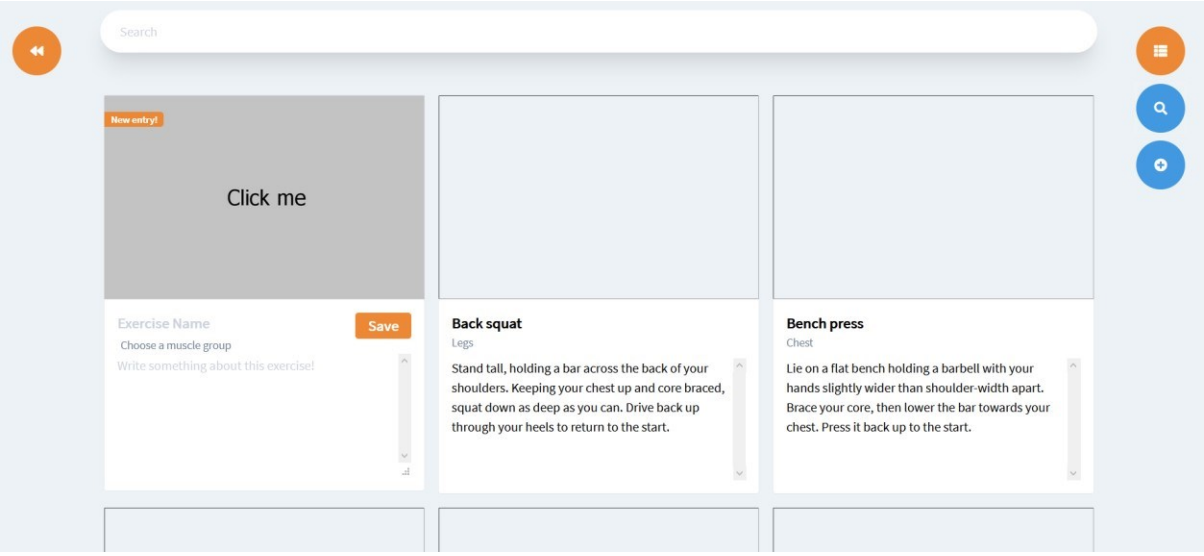
```
24      },
25      async updatePassword(newPass, oldPass) {
26          await Service.patch("/users/password", {id: store.credentials.user._id, newPassword: newPass, oldPassword: oldPass})
27      },
28      async updateInfo() {
29          await Service.patch("/users/info", store.credentials.user)
30          localStorage.setItem("Credentials", JSON.stringify(store.credentials))
31      },
```

Slika 12 - Upit za korisničke izmjene

3.4 Stranica s vježbama

Stranica s vježbama sadrži malo više koda nego prijašnje stranice. Na ovoj stranici prikazane su vježbe abecednim redom. Dugmad s desne strane sučelja redoslijedom od najgornjeg prema najdonjem imaju sljedeće funkcije: prikaz mojih vježbi, otvaranje tražilice, dodavanje nove vježbe.

Slika 13 prikazuje izgled stranice s otvorenom tražilicom i otvorenom karticom za dodavanje nove vježbe. Prazni kvadrati iznad opisa vježbi predviđeni su za njihove fotografije.



Slika 13 - Stranica s vježbama

Korišten je mounted lifecycle hook kojeg također pruža Vue.js framework kako bi se prilikom učitavanja stranice poslao zahtjev za vježbe na backend (Slika 14, linije 163 do 170). Preciznije govoreći, prilikom učitavanja ove stranice poziva se "getExercises" metoda koja šalje zahtjev za vježbama na backend i sprema odgovor (listu vježbi) u varijablu "exercises" u "store.js" datoteci (Slika 15, linije 32 do 37). Treba naglasiti da ta metoda poziva backend samo ukoliko je store varijabla s vježbama prazna (if selekcija), u protivnom će vratiti vježbe iz store datoteke. Na taj način smanjeni su upiti na backend i samim time na bazu podataka.

Slična ideja za smanjivanje upita je implementirana i prilikom spremanja i brisanja vježbi. Nakon što su upiti za spremanje tj. brisanje poslani, ne šalje se novi upit za dopremu vježbi iz baze, već se umeću u store varijablu (Slika 15, linije 39 do 53).

```
156
157     watch: {
158         search: _.debounce(function(value) {
159             this.filterExercises(value);
160         }, 500)
161     },
162
163     async mounted (){
164         if(store.exercises.length > 0) this.exercises = store.exercises
165         else{
166             this.loading = true
167             this.exercises = await service.getExercises()
168             this.loading = false
169         }
170     },
171
172     methods: {
173         filterExercises(value) {
174             if (value == '')
175                 this.exercises = store.exercises
176             else {
177                 this.exercises = store.exercises
178                 value = value.toUpperCase()
179                 this.exercises = this.exercises.filter(
180                     exercise => exercise.name.toUpperCase().includes(value) || exercise.muscle.toUpperCase().includes(value)
181                 )
182             }
183         },
184         editingClick() {
185             this.isEditing = !this.isEditing
186             this.muscleGroup = ''
```

Slika 14 - Tražilica i mounted hook

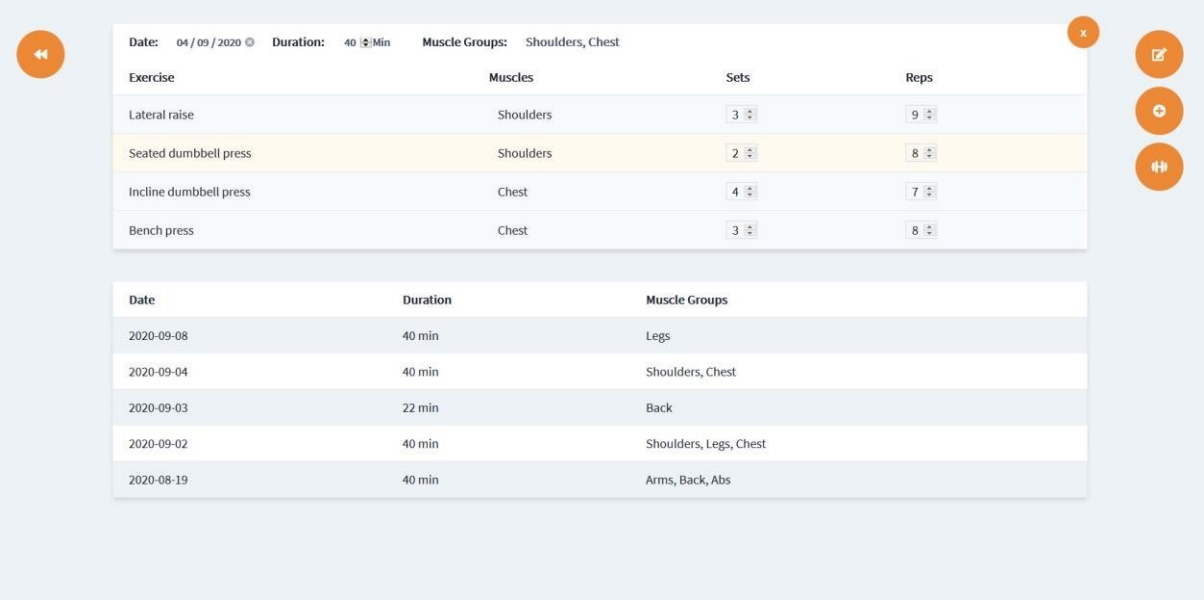
```
31     },
32     async getExercises() {
33         if(store.exercises.length == 0) {
34             let response = await Service.post("/exercises", {id: store.credentials.user._id})
35             store.exercises = response.data
36         }
37         return store.exercises
38     },
39     async saveExercise(exercise) {
40         exercise.owner = store.credentials.user._id
41         let response = await Service.put("/exercises/save", exercise)
42         exercise._id = response.data
43         for (let i = 0; i < store.exercises.length; i++) {
44             if (exercise.name < store.exercises[i].name) return store.exercises.splice(i, 0, exercise)
45         }
46         store.exercises.push(exercise)
47     },
48     async deleteExercise(exerciseId) {
49         await Service.patch("/exercises/delete", {id: exerciseId, owner: store.credentials.user._id})
50         for (let i = 0; i < store.exercises.length; i++) {
51             if (exerciseId == store.exercises[i]._id) return store.exercises.splice(i, 1)
52         }
53     },
```

Slika 15 - Upiti za vježbe

Tražilica je implementirana koristeći watch svojstvo (Slika 14, linija 157). To svojstvo omogućuje praćenje promjena na odabranim varijablama. U njemu je moguće upisati proceduru koja će se izvršiti prilikom svake promjene gledane varijable. Za ovaj projekt stavljeno je da se svakih pola sekunde od zadnje promjene varijable search pozove procedura "filterExercises" opisana na dnu iste slike. Ta procedura u srži filtrira listu vježbi pomoću "filter" metode koja je dio JavaScript standardne biblioteke.

3.5 Praćenje treninga

Stranica za praćenje treninga sadrži ključnu funkcionalnost ove web aplikacije. Na ovoj stranici moguće je dodavati, uređivati i brisati treninge (Slika 16). Tri dugmeta s desne strane redom služe za: uključivanje uređivanja treninga, dodavanje novog treninga, generiranje treninga. Za odabir treninga dovoljno je samo kliknuti njegov red u tablici.



Slika 16 - Uređivanje treninga

Prilikom učitavanja stranice treninzi se dopremaju na isti način kao što su se dopremale vježbe (Slika 17). Svi dijelovi stranice za praćenje treninga koriste iste tehnike koje smo do sad već imali, samo na malo složeniji način.

```
54  async getWorkouts() {
55      if(store.workouts.length == 0) {
56          let response = await Service.post("/workouts", {id: store.credentials.user._id})
57          store.workouts = response.data
58      }
59      return store.workouts
60  },
61  async saveWorkout(workout) {
62      workout.owner = store.credentials.user._id
63      let response = await Service.put("/workouts/save", workout)
64      workout._id = response.data
65      for (let i = 0; i < store.workouts.length; i++) {
66          if (workout.date > store.workouts[i].date) return store.workouts.splice(i, 0, workout)
67      }
68      store.workouts.push(workout)
69  },
70  async updateWorkout(workout) {
71      await Service.patch("/workouts/update", workout)
72      for (let i = 0; i < store.workouts.length; i++) {
73          if (store.workouts[i]._id == workout._id) return store.workouts[i] = workout
74      }
75  },
```

Slika 17 - Upiti za treninge

Novitet koji vrijedi spomenuti jest implementacija automatskog prikaza grupa mišića za pojedini trening. Naime, korisnik je prinuđen odabrati datum i upisati trajanje prilikom kreiranja treninga. Međutim, grupe mišića odrađene u tom treningu nije moguće odabrati. To je zato što se one automatski upisuju na temelju obavljenih vježbi u tom specifičnom treningu.

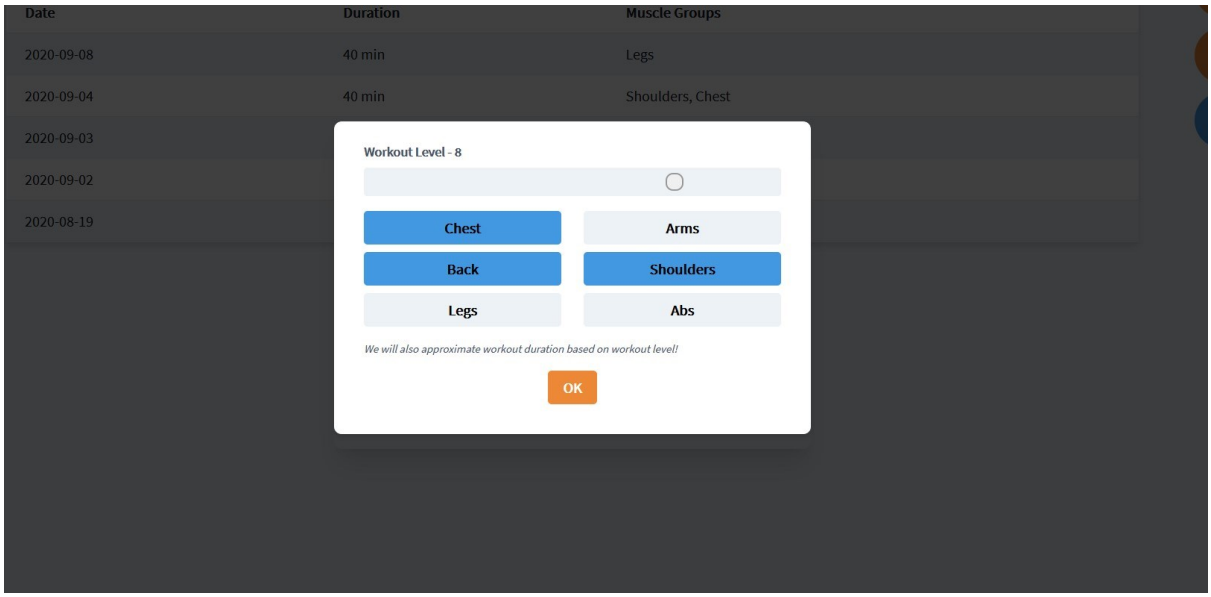
Način na koji je napravljeno da se odrađene grupe mišića automatski prikažu za svaki trening puno je jednostavniji nego što se na prvi pogled čini. Svaki trening je u stvari jedan objekt koji sadrži vrijednosti za datum, vrijeme, grupu mišića, i listu vježbi. Prije svakog spremanja treninga, vrijednosti u grupi mišića se zamjenjuju sa varijablom "currentMuscleGroups" (Slika 18). Ta varijabla je unutar "computed" svojstva koje ima obrnutu svrhu od prije objašnjenog watch svojstva. To znači da će se varijabla promijeniti kada dodamo novu vježbu u listu vježbi unutar trening objekta. Promjena će biti izvedena na način da će program proći kroz listu vježbi i dodati grupu mišića od svake vježbe u privremenu listu. Nakon toga će se privremena lista filtrirati na način da ostanu samo unikatne vrijednosti, linije koda od linije 121 do 124.

```
115     computed: {
116         currentMuscleGroups: function () {
117             this.computedUpdate
118             if(this.currentWorkout.exercises == undefined || Object.keys(this.currentWorkout).length === 0) return;
119             let arr = []
120             this.currentWorkout.exercises.forEach(el => arr.push(el.muscle))
121             let unique = arr.filter((value, index, self) => {
122                 return self.indexOf(value) === index
123               })
124             return unique.join()
125         }
126     },
127
128     async mounted (){
129         if(store.workouts.length > 0) this.workouts = store.workouts
130         else{
131             this.loading = true
132             this.workouts = await service.getWorkouts()
133             this.loading = false
134         }
135     },
136
```

Slika 18 - Izlučivanje grupa mišića iz liste

Slika 19 prikazuje modal koji se otvara klikom na generiranje treninga. Korisniku je dana traka za povlačenje na kojoj može odrediti težinu treninga. Najmanje težina treninga je 1, a najveća 10. Ispod nje nalazi se dugmad za izbor grupe mišića koju trening treba sadržavati. Klikom na "OK" trening će biti izgeneriran.

Način na koji algoritam određuje koliko vježbi sadržava trening, te koliko ponavljanja i setova svaka vježba sadrži, napravljen je na bazi broja ponavljanja petlje za odabir. Što je vježba teža, to će se više puta petlja ponoviti. Slika 20 prikazuje spomenutu petlju. Svakom iteracijom petlja će ili dodati novu vježbu u trening listu, ili će povećati broj setova jedne vježbe unutar petlje.



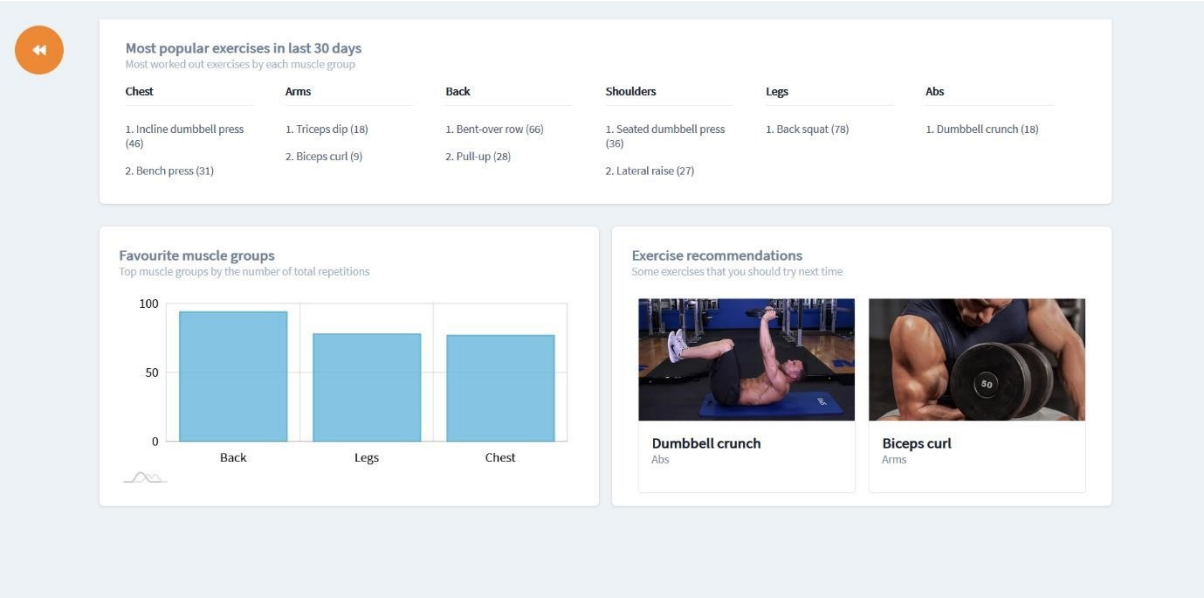
Slika 19 - Generiranje vježbe

```
139
140         let j = 0
141         while (points-- > 0) {
142           if (++j == chosenGroups.length) j = 0
143           let found = false
144           if (Math.floor(Math.random() * 2)){
145             for (let i = 0; i < exercises.length; i++) {
146                 if (exercises[i].muscle == chosenGroups[j]){
147                   exercises[i].sets = 1
148                   exercises[i].reps = Math.floor(Math.random() * 5) + Math.floor(Number(this.hardness)/2) + 4
149                   generatedExercises.push(exercises[i])
150                   exercises.splice(i, 1)
151                   found = true
152                   break
153                 }
154             }
155           }
156           if (!found) {
157             for(let i = generatedExercises.length-1; i > 0; i--){
158               const j = Math.floor(Math.random() * i)
159               const temp = generatedExercises[i]
160               generatedExercises[i] = generatedExercises[j]
161               generatedExercises[j] = temp
162             }
163             for (let i = 0; i < generatedExercises.length; i++) {
164               if (generatedExercises[i].muscle == chosenGroups[j]){
165                 generatedExercises[i].sets++
166                 break
167               }
168             }
169           }
170         }
```

Slika 20 - Dio koda za generaciju treninga

3.6 Statistika

Ova funkcionalnost dodana je nakon inicijalnog dizajna aplikacije, te zbog tog razloga nije uključena u dijagrame u drugom poglavlju. Na stranici za pregled statistike (Slika 21) možemo naći nekoliko zanimljivih informacija. U gornjoj kartici nalaze se korisnikove vježbe poredane po popularnosti na bazi njegovih treninga unutar zadnjih mjesec dana. U donjoj lijevoj kartici nalaze se najaktivnije korisnikove mišićne grupe u zadnjih mjesec dana. I za kraj na donjoj desnoj kartici nalaze se prijedlozi vježbi koje korisnik nije dugo radio.



Slika 21 - Statistika korisnika

```
124
125     for (let i = 0; i < workouts.length; i++) {
126         if (date > workouts[i].date) break
127         workouts[i].exercises.forEach ( el =>   {
128             for (let j = 0; j < this.workoutStats.length; j++) {
129                 if (el.muscle == this.workoutStats[j].name ) {
130                     this.workoutStats[j].totalReps += Number(el.sets) * Number(el.reps)
131                     let index = this.workoutStats[j].ranks.findIndex(x => x.name == el.name);
132                     if (index == '-1') {
133                         el.reps = Number(el.sets) * Number(el.reps)
134                         el.sets = Number(el.sets)
135                         this.workoutStats[j].ranks.push(el)
136                     }
137                     else {
138                         this.workoutStats[j].ranks[index].sets += Number(el.sets)
139                         this.workoutStats[j].ranks[index].reps += Number(el.sets) * Number(el.reps)
140                     }
141                     break
142                 }
143             }
144         })
145     }
146
147     for (let i = 0; i < this.workoutStats.length; i++) {
148         this.workoutStats[i].ranks.sort((a, b) => (a.reps > b.reps) ? -1 : 1)
149     }
150
```

Slika 22 - Prikupljanje ukupnih ponavljanja vježbi

Slika 22 prikazuje način na koji se prikupljaju podatci o korisnikovim vježbama. Prva (vanjska) petlja prolazi po svim treninzima dok ne dođe do treninga čiji je datum stariji od mjesec dana. Druga petlja ugniježđena je u prvu i prolazi kroz sve vježbe unutar trenutnog treninga. Treća petlja ugniježđena je u drugu i prolazi kroz sve grupe mišića za koje se skuplja statistika. Kada grupa mišića kojoj trenutna vježba pripada odgovara grupi mišića za koju se trenutno provjerava, probati će se naći indeks vježbe unutar liste. Ukoliko indeks nije nađen, vježba se ubacuje u tu listu. Ukoliko je indeks pronađen, onda se vježba ne ubacuje u listu već se povećava broj ponavljanja iste vježbe unutar liste. Neovisno o indeksu, broj ponavljanja te vježbe će se isto tako dodati varijabli koja prati ukupna ponavljanja za tu grupu mišića.

Slika 23 prikazuje na koji način se traže dvije vježbe koje korisnik nije dugo radio. Naime, petlja prolazi kroz sve vježbe sve dok se ne dobiju dvije vježbe ili dok ne dođe do kraja. U svakoj iteraciji traži se da li korisnikovi treninzi u zadnjih mjesec dana sadrže trenutnu vježbu (koristi se ista listu koja je napravljena za potrebe broja ponavljanja u prethodnom odlomku). Ukoliko ne sadrže, vježba je dodana kao jedna od vježbi koja nije dugo rađena. U suprotnom, ne dešava se ništa i iteracija ide dalje. Ako petlja nije našla dvije vježbe, vježbe koje nedostaje ubacuju se naknadno tako da se uzmu vježbe s malo ponavljanja od onih koje je korisnik već odradio taj mjesec.

```
156
157         let counter = 0
158         this.recom = []
159
160         for( let i = 0; i < exercises.length; i++ ) {
161             let found = false
162             for ( let j = statArray.length - 1; j >= 0; j--) {
163                 console.log(1, exercises[i].name)
164                 let index = statArray[j].ranks.findIndex(el => el.name == exercises[i].name)
165                 if(index != '-1') {
166                     found = true
167                 }
168             }
169             if(!found) {
170                 counter++
171                 this.recom.push(exercises[i])
172                 if(counter == 2) {
173                     break
174                 }
175             }
176         }
177
178         if(counter < 2) {
179             let arr = statArray[statArray.length - 1].ranks
180             this.recom.push(arr[arr.length - 1])
181         }
182
183         if(counter == 0) {
184             let arr = statArray[statArray.length - 2].ranks
185             this.recom.push(arr[arr.length - 1])
186         }
```

Slika 23 - Traženje zapostavljenih vježbi

4 Backend

Backend ima dva vrlo jednostavna zadatka: autorizaciju nadolazećih upita i komunikaciju s bazom podataka.

Prva linija autorizacije upita je JSON web token. Slika 24 prikazuje dio koda na backendu u kojem se potpisuje novi token prilikom upita za prijavu, te ga se šalje zajedno sa korisničkim informacijama. Slika 25 prikazuje dio koda na frontendu u kojem je definiran url backenda i dio u kojem se stavlja token u zaglavlje svakog odlaznog upita. Slika 26 prikazuje dio koda na backendu u kojem se provjerava vjerodostojnost primljenog tokena.

```
26          let token = jwt.sign(user, process.env.JWT_SECRET, {
27              algorithm: 'HS512',
28              expiresIn: '1 day',
29          })
30          res.json({token: token, user: user})
31      } catch {
32          res.sendStatus(400)
33      }
```

Slika 24 - Slanje tokena na upitu za prijavu

```
4   let Service = axios.create({
5       baseURL: 'https://modernfitness.herokuapp.com/',
6       timeout: 5000
7   })
8
9   Service.interceptors.request.use((request) => {
10      request.headers['authorization'] = 'Bearer ' + store.credentials.token;
11      return request;
12  })
13
```

Slika 25 - Token u headeru

```
MFBackend > src > JS auth.js > default
1   import jwt from "jsonwebtoken"
2
3   export default (req, res, next) => {
4       try {
5           let authorization = req.headers['authorization'].split(' ')
6           if (authorization[0] !== 'Bearer') res.sendStatus(401)
7           req.jwt = jwt.verify(authorization[1], process.env.JWT_SECRET)
8           next()
9       } catch {
10          res.sendStatus(403)
11      }
12  }
```

Slika 26 - JSON web token

4.1 Upiti na bazu

Upiti na bazu su ključni dio svakog backenda. Na koji način funkcioniraju pokazan je kroz upite za prijavu i registraciju (Slika 27). Za upite korišten je paket Express [6].

Upiti za registraciju su konstruirani na način da se s njima šalje šifra, korisničko ime i ostali podatci ispunjeni na formi za registraciju. U ovom projektu na backendu se provjerava da li je dužina imena i šifre dovoljna u slučaju da korisnik pokušava slati neodgovarajuće upite, tj. radi se validacija podataka. Nakon validacije, backend pokušavamo dopremiti jednog korisnika iz baze s imenom u poslanom zahtjevu. Ukoliko baza vrati korisnika to će značiti da korisnik s takvim imenom već postoji i registracija neće nastaviti dalje. Ako je sve u redu, šifra se enkriptira pomoću bcrypt biblioteke, te se dodaje korisnika u bazu.

Kada je riječ o prijavi, prvo će backend pokušati dopremiti korisnika s imenom iz zahtjeva. Ukoliko takav korisnik postoji, usporediti će se enkriptirana šifru iz baze s šifrom iz zahtjeva. Ako je sve u redu proizvodi se token (kako je objašnjeno u prošloj poglavlju) i šalju se nazad korisnički podatci s tokenom.

```
app.post('/users/login', async (req, res) => {
    try {
        let db = await connect()
        let user = await db.collection("users").findOne({ username: req.body.username })
        if (user == null || !(await bcrypt.compare(req.body.password, user.password)))
            return res.status(400).send("Credentials are wrong!")
        delete user.password
        let token = jwt.sign(user, process.env.JWT_SECRET, {
            algorithm: 'HS512',
            expiresIn: '1 day',
        })
        res.json({token: token, user: user})
    } catch {
        res.sendStatus(400)
    }
})

app.put('/users/signup', async (req, res) => {
    try {
        if (req.body.password.length < 6) return res.status(400).send("Password is too short!")
        if (req.body.username.length < 4) return res.status(400).send("Username is too short!")
        let db = await connect()
        let test = await db.collection("users").findOne({ username: new RegExp(req.body.username, "i") })
        if (test != null) return res.status(400).send("Username is taken!")
        req.body.password = await bcrypt.hash(req.body.password, 8)
        let result = await db.collection("users").insertOne(req.body)
        if (result.insertedCount == 1) {
            res.sendStatus(200)
        }
        else res.sendStatus(400)
    } catch {
        res.sendStatus(400)
    }
})
```

Slika 27 - Upiti na bazu za prijavu i registraciju

4.2 Baza podataka

U projektu korišten je MongoDB Atlas kao baza podataka. To je naime vrlo pristupačan database servis u oblaku koji pruža mnoge zgodne pogodnosti. Za razliku od relacijskih baza, MongoDB sprema podatke u obliku dokumenata [7]. Ti dokumenti se organizirani u kolekcije. Za ovaj projekt napravljene su tri kolekcije: jedna za korisnike, jedna za vježbe i jedna za treninge.

Za primjer dokumenta iz baze, napravljen je jedan query za dopremu testnog korisnika aplikacije i potom jedan query za dopremu njegovih vježbi. Slika 28 prikazuje polja spremljenog testnog korisnika u bazi. Prava šifra od korisnika nije vidljiva jer je enkriptirana na backendu kako je već bilo objašnjeno. Slika 29 prikazuje polja korisnikovih treninga u bazi.

Slika 28 - Korisnik u bazi

Slika 29 - Treninzi korisnika u bazi

5 Zaključak

Izrada aplikacija nije ni najmanje lak posao. Samo za izradu jednostavne web aplikacije kao što je bio slučaj u ovom radu, bila je potrebna određena količina znanja, vještina i vremena. Za izradu većih i kompliciranijih aplikacija količina posla rasla bi eksponencijalno. Isto tako, prilikom izrade ove aplikacije nije se vodilo računa o zakonima pojedinih država, trenutnim sigurnosnim standardnima, mikrotransakcijama, optimizaciji za tražilice, itd. Zbog svih tih problema ne bi se trebalo žuriti u izradu aplikacije prije nego se razmotre sve moguće poteškoće. Nadalje, kako je bilo viđeno kroz razvoj ove fitness aplikacije, prilikom izrade postojali su mnogi dijelovi koji su samostalni za sebe i koji se na kraju spajaju zajedno u složnu cjelinu. Neki od tih dijelova nemaju baš puno zajedničkih karakteristika. Zbog tog razloga puno je lakše podijeliti posao u manje timove koji će biti specijalizirani za specifične zadatke, npr. dizajn, baze, sigurnost, i sl. Na taj način povećavamo cjelokupnu kvalitetu proizvoda pošto developeri mogu usredotočiti svoje vrijeme na izoštravanje znanja i vještina na specifičnom području.

Što se tiče temeljnih funkcionalnosti izrađene aplikacije može se reći da zadovoljava sve točke koje su bile navedene u koraku prikupljanja zahtjeva. Međutim, kako je već bilo spomenuto u početnim poglavljima, životni ciklus aplikacije ne završava njenom izradom i isporukom. Korisnički zahtjevi ne samo da se mogu mijenjati, već će se oni najvjerojatnije i mijenjati tijekom života jedne aplikacije. Na primjer, već sada odmah neposredno nakon završetka izrade ove aplikacije, nove funkcionalnosti dolaze na listu mogućih proširenja. Jedna od takvih funkcionalnosti bila bi mogućnost pregleda količine vježbi i mišićnih grupa za sve treninge unutar nekog zadanog razdoblja. Druga korisna funkcionalnost bila bi mogućnost brzog i jednostavnog dijeljenja treninga na društvenim mrežama. Nove ideje kao što su te, mogu nastaviti pristizati i dugo nakon što je aplikacija inicijalno napravljena. Zato prilikom izrade isto tako valja voditi računa o modelu arhitekture koji pruža adaptabilnost.

REFERENCE

[1] R. Manger, Softversko inženjerstvo, Skripta, 2 ed., Zagreb: Sveučilište u Zagrebu, 2013.

[2] Tailwind Labs Inc., "Tailwind CSS Documentation," [Online]. Dostupno: https://tailwindcss.com/docs.

[3] O. Filipova, Learning Vue.js 2, Brimingham: Packt Publishing Ltd., 2016.

[4] Vue.js, "Navigation Guards," [Online]. Dostupno: https://v3.router.vuejs.org/guide/advanced/navigation-guards.html.

[5] Vue.js, "Event Handling," [Online]. Dostupno: https://v2.vuejs.org/v2/guide/events.html.

[6] Express contributors, "API reference," [Online]. Dostupno: https://expressjs.com/en/4x/api.html.

[7] MongoDB, Inc., "MongoDB Documentation," [Online]. Dostupno: https://docs.atlas.mongodb.com.

POPIS SLIKA

PRILOZI ZA OBRANU RADA

Repozitorij za pregled koda frontenda

https://github.com/vilaleon/modern-fitness

Repozitorij za pregled koda backenda

https://github.com/vilaleon/modern-fitness-backend