Aplikacija za planiranje vježbanja

Koružnjak, Leon

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

APLIKACIJA ZA PLANIRANJE VJEŽBANJA

Završni rad

Leon Koružnjak

Osijek, 2023.

SADRŽAJ

1. UVOD

Aplikacije za vježbanje služe kao alati koji korisnicima pomažu u planiranju, praćenju i provođenju vježbi kako bi postigli svoje fitness ciljeve. Fizička aktivnost ima brojne prednosti i važno je biti fizički aktivan, posebno u današnjem svijetu gdje se mnogi ljudi manje kreću zbog sjedilačkih poslova i upotrebe tehnologije. Neki od benefita fizičke aktivnosti su: zdravlje srca i krvnih žila, poboljšanje raspoloženja i mentalnog zdravlja, prevencija kroničnih bolesti, poboljšanje kvalitete sna [1]. Iz tih razloga važno je biti fizički aktivan pri čemu nam pomažu aplikacije za vježbanje. Glavni problem u izradi ovakve aplikacije je odabrati najpogodniji način za spremanje podataka te osigurati pristupačno i estetski zadovoljavajuće korisničko sučelje koje pruža korisnicima jednostavan pristup podacima. U drugom poglavlju navedene su slične aplikacije koje rješavaju ove probleme, u trećem poglavlju opisane su tehnologije i alati koji su korišteni u svrhu izrade aplikacije. Četvrto poglavlje opisuje način izrade svakog zasebnog zaslona u aplikaciji te rješenja navedenih problema spremanja podataka i vizualne pristupačnosti. Konačno, peto poglavlje predstavlja zaključak rada u kojemu se sumiraju rješenja i procesi korišteni u izradi aplikacije.

1.1. Zadatak završnog rada

Izraditi mobilnu aplikaciju koja će omogućiti odabir raznih vježbi po kategorijama mišićnih skupina, te na taj načini izraditi plan treninga. Tijekom vježbanja omogućiti da aplikacija vodi korisnika kroz planirani popis vježbi. Aplikacija treba omogućiti i unos određenih ciljeva koje korisnik želi postići. Kroz kalendarski prikaz omogućiti prikaz svih ostvarenih i neostvarenih ciljeva kao i realiziranih i nerealiziranih treninga na pregledan način.

2. PRIJEDLOG RJEŠENJA

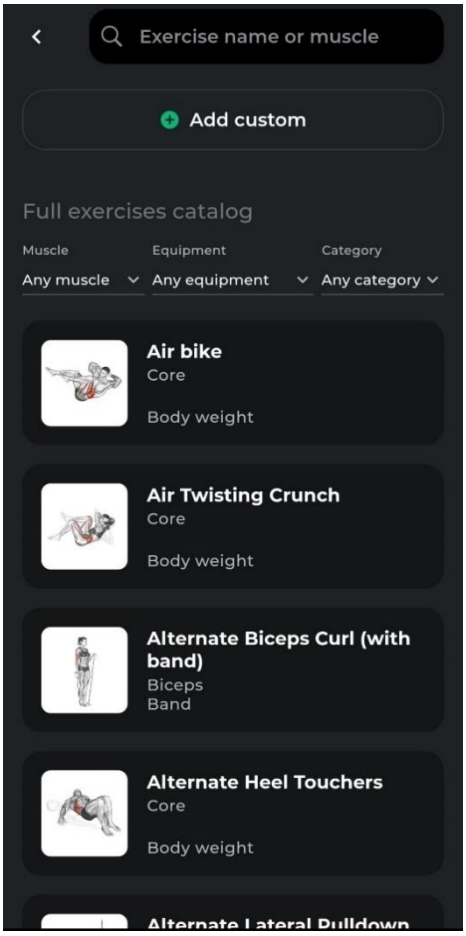
Aplikacija se sastoji od 5 glavnih zaslona gdje je početni zaslon „Home Page“ u kojemu se nalaze izbornici za ostala 4 glavna zaslona. Ako korisnik odabere izbornik „Start a workout“ on će ga odvesti na novi zaslon gdje će odabrati jedan od svojih generiranih treninga, zatim će se otvoriti novi zaslon koji će imati prikaz svih vježbi u tom treningu. Ovaj zaslon korisnik koristi kada fizički radi odabrani trening, te mu je omogućeno praćenje proteklog vremena i mogućnost označavanja obavljenih vježbi. Kada je korisnik gotov prikazuje se zaslon na kojemu piše koliko je vremena ukupno proteklo i koliko je vježbi korisnik napravio. Nadalje, imamo „Goals & Calendar“ zaslon na kojemu se nalazi kalendar koji prikazuje datume kada je korisnik odradio trening. Na ovom zaslonu također se nalazi popis ciljeva koje korisnik može dodavati, brisati te označiti ako je cilj ostvaren. Treći zaslon naziva se „My workouts“ u kojemu korisnik stvara i imenuje novi program. Pritiskom na određeni program korisnik može odabrati vježbe kategorizirane po mišićnim skupinama koje želi staviti u svoj program. Korisniku je također omogućeno brisanje i preimenovanje programa te brisanje neželjenih vježbi u programu. Zadnji zaslon „My PR's“ sadrži popis postignutih rekorda u osnovnim vježbama te mogućnost dodavanja novih i brisanja starih rekorda.

2.1. Opis postojećih rješenja

Na tržištu postoji obilje aplikacija za vježbanje koje nude različite mogućnosti prilagođene svakom pojedincu. Bez obzira jeste li početnik ili iskusni vježbač, bez obzira preferirate li treninge snage, kardio, jogu ili nešto drugo, postoji aplikacija koja odgovara vašim potrebama i željama. Aplikacije na tržištu koje su najsličnije aplikaciji prezentiranoj u ovom radu prikazane su u nastavku.

2.1.1. My Workout Plan – Daily Workout Planner

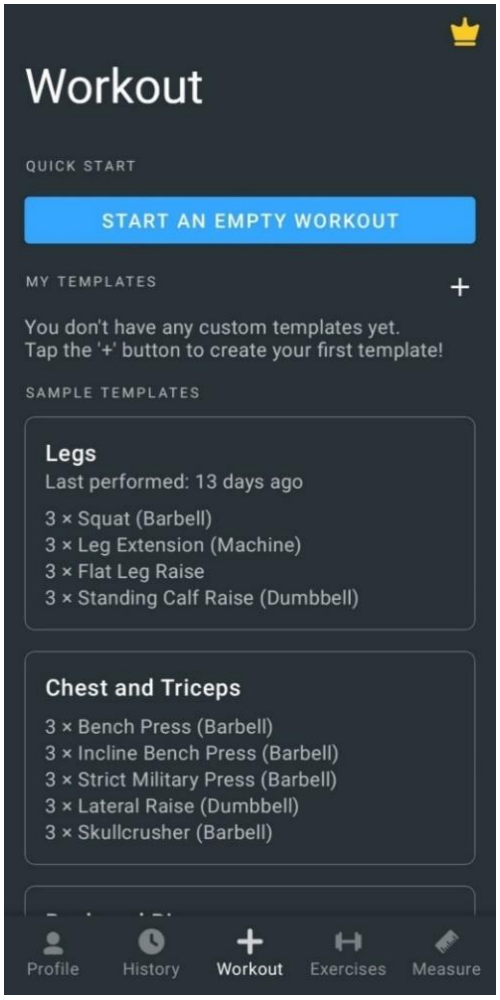
Aplikacija „My Workout Plan – Daily Workout Planner“ je jedna od popularnijih aplikacija za vježbanje koja vam pomaže stvoriti i organizirati svoj dnevni plan treninga (Slika 2.1.1.). Ova aplikacija pruža strukturirani pristup vježbanju i omogućuje vam da personalizirate svoje treninge prema svojim ciljevima i raspoloživom vremenu.



Sl. 2.1.1. „My Workout Plan“ aplikacija

2.1.2. Strong

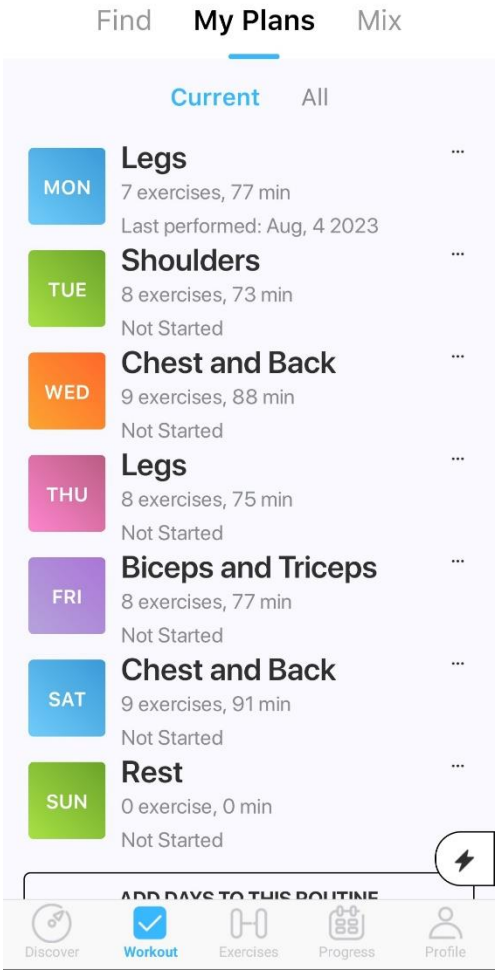
Pomoću aplikacije Strong korisnici mogu pratiti svoje treninge, bilježiti broj ponavljanja, težinu koju koriste i trajanje treninga. Aplikacija također pruža grafički prikaz napretka kako bi korisnici mogli vizualno pratiti svoje postignute ciljeve (Slika 2.1.2.).



Sl. 2.1.2. „Strong“ aplikacija

2.1.3. Jefit

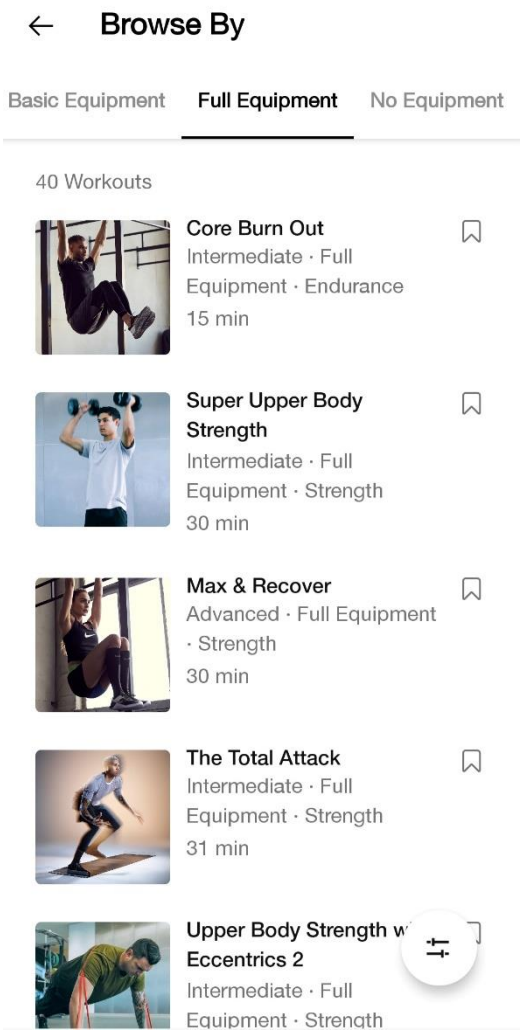
Ova aplikacija, kao i Strong, omogućuje vam praćenje vaših treninga (Slika 2.1.3.). Ona već ima unaprijed registrirane vježbe i planove, ali imate mogućnost dodati i druge vježbe prema vašim željama. Također, možete stvarati svoje trening rutine, pratiti napredak tijekom vremena te bilježiti težinu i tjelesne mjere. S ovom aplikacijom imate pristup zajednici korisnika Jefit-a i njihovim stvorenim planovima treninga, što nije moguće s aplikacijom Strong.



Sl. 2.1.3. „Jefit“ aplikacija

2.1.4. Nike Training Club: Fitness

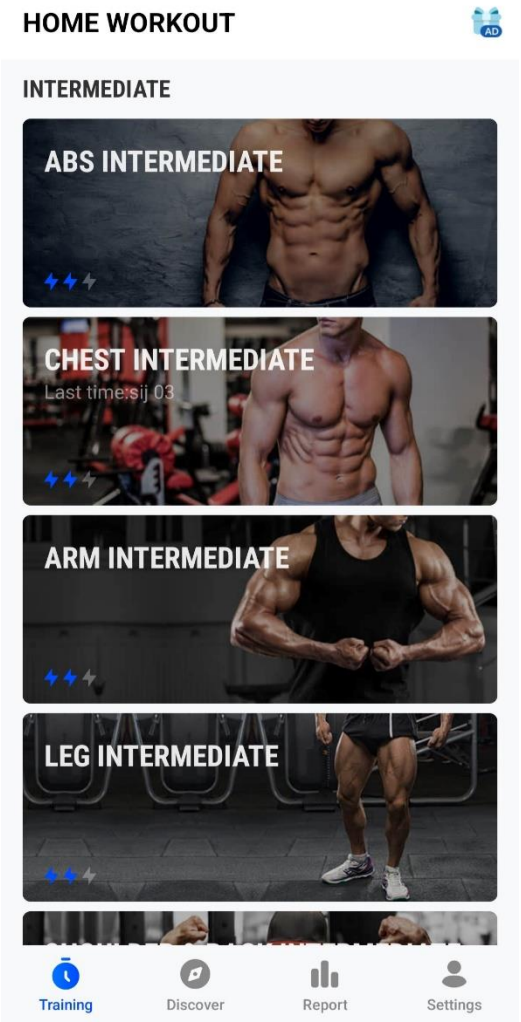
Nike aplikacija nudi više od 185 treninga podijeljenih u kategorije poput snage, izdržljivosti, pokretljivosti i joge (Slika 2.1.4.). Pomoću aplikacije možete stvarati personalizirane planove prema vašim ciljevima, odabranoj opremi i broju treninga tjedno. Također, aplikacija će vam davati prijedloge za treninge na temelju vaše povijesti treninga, te također nudi i videozapise koji vam pomažu u pravilnom izvođenju vježbi.



Sl. 2.1.4. „Nike Training Club“ aplikacija

2.1.5. Home Workout – No Equipment

"Home Workout - No Equipment" spada među najbolje sveobuhvatne aplikacije za vježbanje kod kuće. Ova aplikacija ima visoku ocjenu u Play Storeu, što je sasvim opravdano (Slika 2.1.5.). Nudi izuzetno funkcionalno korisničko sučelje i odličan set treninga koje možete koristiti ako vježbate kod kuće. Uključuje rutine zagrijavanja i istezanja, automatsko bilježenje napretka u treningu te praćenje trendova vaše tjelesne težine. Možete prilagoditi podsjetnike za treninge i dobiti detaljne video i animacijske upute.



Sl. 2.1.5 „Home Workout - No Equipment" aplikacija

3. TEHNOLOGIJE KORIŠTENE U IZRADI RJEŠENJA

Aplikacija za planiranje vježbanja napravljena je koristeći Android Studio. Pisana je u programskom jeziku Dart u razvojnom okviru Flutter. Za pohranu podataka koristit se Hive i shared_preferences.

3.1. Android Studio

Android Studio je službeno integrirano razvojno okruženje (IDE) za razvoj Android aplikacija. Android Studio pruža više značajki koje poboljšavaju produktivnost pri izgradnji Android aplikacija. Android Studio posjeduje fleksibilni sustav izgradnje temeljen na Gradle-u. Pruža brzi emulator s bogatim skupom značajki za testiranje aplikacija [7]. Omogućuje razvoj za sve Android uređaje u jedinstvenom okruženju. Moguće je primijeniti promjene u izvornom kodu pokrenute aplikacije bez ponovnog pokretanja iste. Nudi opsežne alate i okvire za testiranje aplikacija [8]. Podržava C++ i NDK (Native Development Kit).

3.2. Dart

Dart je objektno orijentirani, otvoreni, višenamjenski programski jezik s sintaksom sličnom C-u kojeg je razvio Google 2011. godine. Glavni cilj Dart-a je stvaranje korisničkih sučelja za web i mobilne aplikacije. Inspiriran je drugim programskim jezicima poput C#-a, Java-e, i JavaScript-a. Dart podržava većinu uobičajenih koncepata programskih jezika kao što su klase, sučelja i funkcije [9].

3.3. Flutter

Flutter je otvoreni okvir za razvoj aplikacija koji je razvio Google. Ovaj okvir je sve popularniji među programerima zbog svojih brojnih prednosti i jednostavnosti korištenja. Flutter omogućuje programerima da izgrade korisnička sučelja za različite platforme, uključujući mobilne uređaje i web, koristeći jedinstveni kod. To znači da se aplikacija može razviti brže i učinkovitije, smanjujući potrebu za pisanjem i održavanjem odvojenih kodova za svaku platformu [10]. Zahvaljujući korištenju programskog jezika Dart i kompilaciji u strojni kod, Flutter postiže visoku performansu koja je usporediva s izvornim aplikacijama. To omogućuje brzo i glatko korisničko iskustvo (Web 4). Flutter također dolazi s nizom alata koji olakšavaju razvoj aplikacija. Na primjer, "hot reload" omogućuje programerima da brzo vide rezultate izmjena u kodu, čime se ubrzava proces razvoja [11].

3.4. Hive

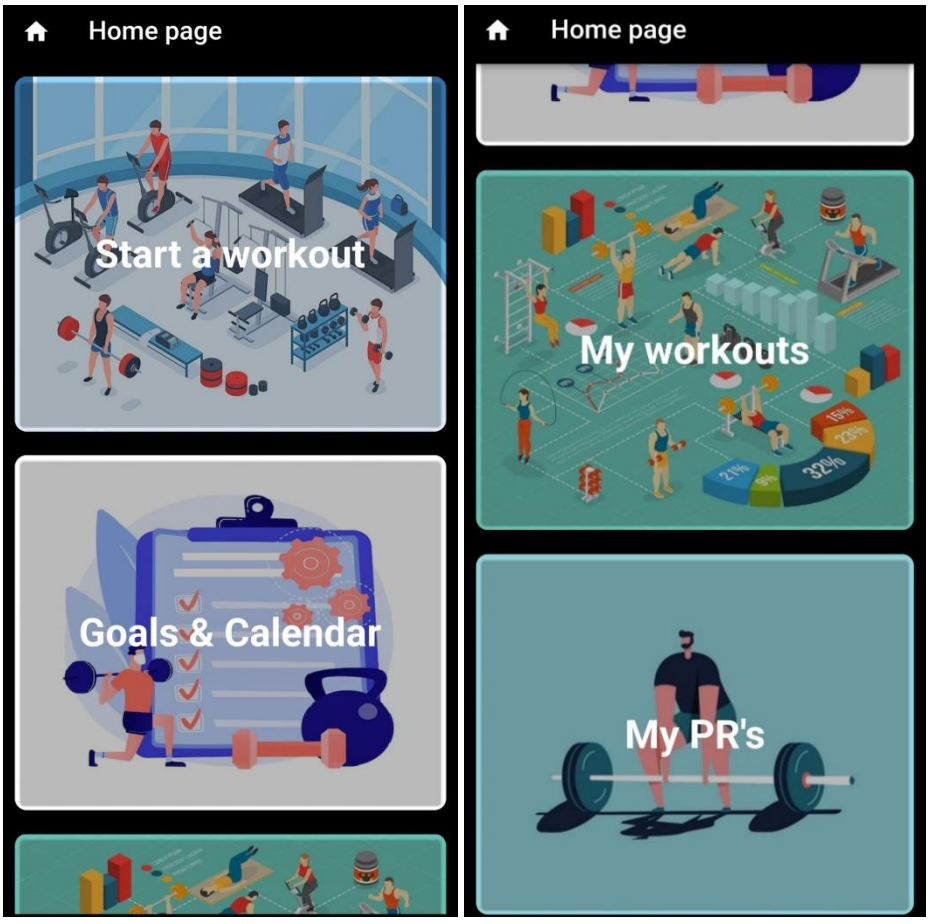
Hive je brza i efikasna baza podataka ključ-vrijednost napisana u čistom Dartu. Inspirirana Bitcaskom, Hive podržava ne samo osnovne tipove podataka, liste i mape, već i bilo koji Dart objekt po vašem izboru. Prije spremanja objekata, potrebno je generirati adapter za odgovarajući tip. Hive je razvijen s Flutterom na umu te predstavlja idealan izbor ako vam je potrebna lagana pohrana podataka za vašu aplikaciju. Nakon dodavanja potrebnih ovisnosti i inicijalizacije Hive-a, možete ga koristiti u svom projektu [12].

4. RAZVOJ APLIKACIJE

4.1. Početni zaslon

Početni zaslon naziva se „Home Page“ (Slika 4.1.1.) na kojemu se nalaze izbornici za 4 glavne funkcionalnosti aplikacije, to su:

1. „Start a workout“ – vodi korisnika kroz odabrani program
2. „Goals & Calendar“ – vodi korisnika na zaslon na kojem se nalazi kalendar i ciljevi
3. „My workouts“ – vodi korisnika na zaslon gdje se stvaraju novi programi i dodaju nove vježbe u postojeće programe
4. „My PR's“ – vodi korisnika na zaslon sa prikazom osobnih rekorda



Sl. 4.1.1. Prikaz početnog zaslona

Izgled početnog zaslona ostvaren je pomoću widgeta „SingleChildScrollView“ (Slika 4.1.2.) koji omogućava korisniku klizanje (scrollanje) po zaslonu. On omotava stupac („Column“) u kojemu se nalaze 4 widgeta naziva „Card“. „Card“ widget nam omogućava da na njega stisnemo kao gumb. On je umotan u „Container“ widget koji nam omogućava da ga uredimo pomoću „BoxDecoration“ svojstva. Također, u svakom kontejner widgetu su smještene slike. Za stvaranje razmaka između izbornika koristi se „SizedBox“ widget kojeg konfiguriramo svojstvima „height“ i „width“.

```
25      body: SingleChildScrollView(
26        child: Column(
27          children: [
28            Padding(
29              padding: const EdgeInsets.all(10.0),
30              child: Container(
31                decoration: BoxDecoration(
32                  color: Colors.grey,
33                  borderRadius: BorderRadius.circular(10),
34                  image: DecorationImage(
35                    image: AssetImage("lib/assets/start_a_workout.jpeg"),
36                    fit: BoxFit.fill,
37                  ),  // DecorationImage
38                ),  // BoxDecoration
39                height: 300,
40                child: Card(
41                  color: Colors.transparent,
42                  clipBehavior: Clip.hardEdge,
43                  child: InkWell(
44                    splashColor: Colors.white60,
45                    onTap: () {
46                      Navigator.of(context).push(
47                        MaterialPageRoute(builder: (context) {
48                          return const startWorkoutPage();
49                        }),  // MaterialPageRoute
50                      );
51                    },
52                    child: const SizedBox(
53                      width: double.infinity,
54                      height: 100,
55                      child: Center(
56                        child: Text(
57                          "Start a workout",
58                          style: TextStyle(
59                            fontWeight: FontWeight.bold,
60                            fontSize: 30,
61                            color: Colors.white,
62                          ),  // TextStyle
63                        ),  // Text
```

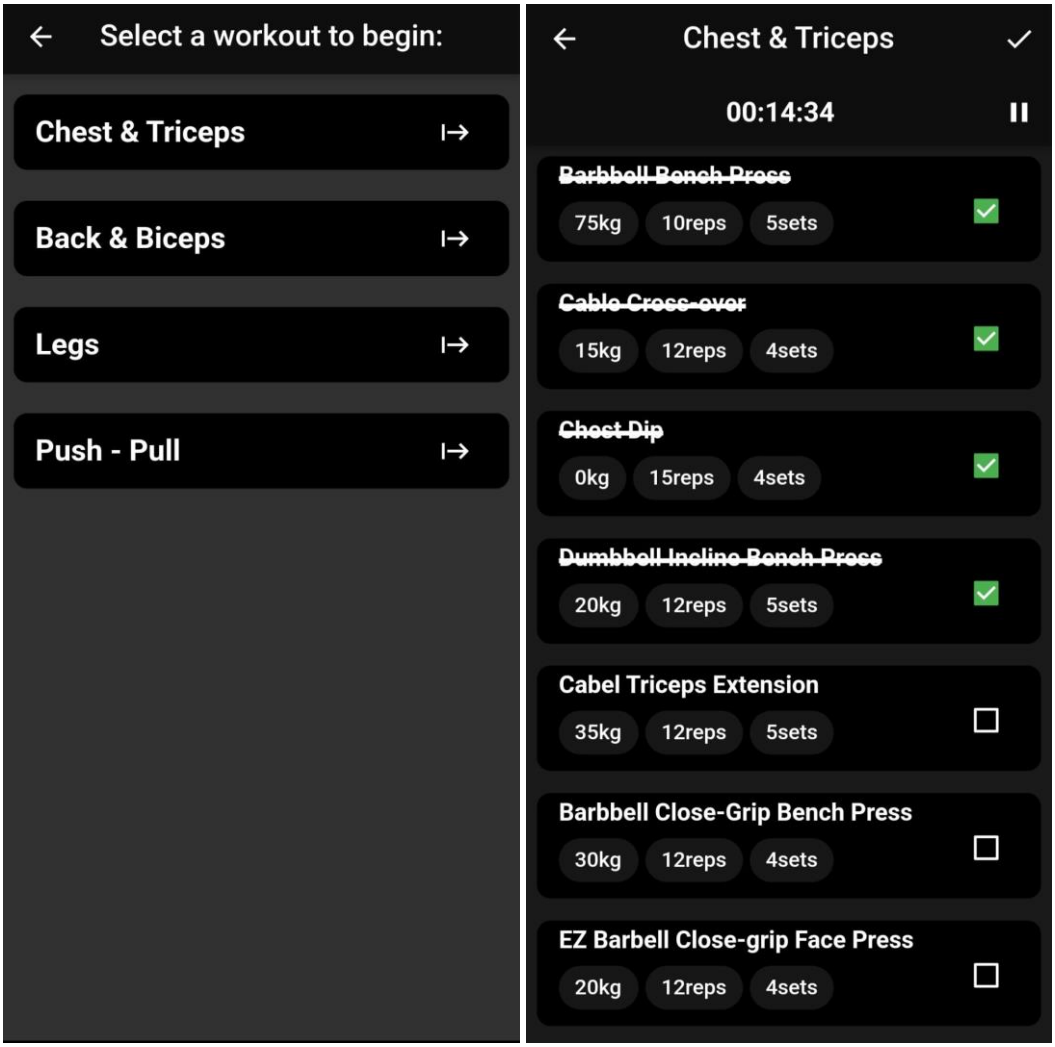
Sl. 4.1.2. Prikaz implementacije izgleda početne stranice

4.2. „Start a workout“ zaslon

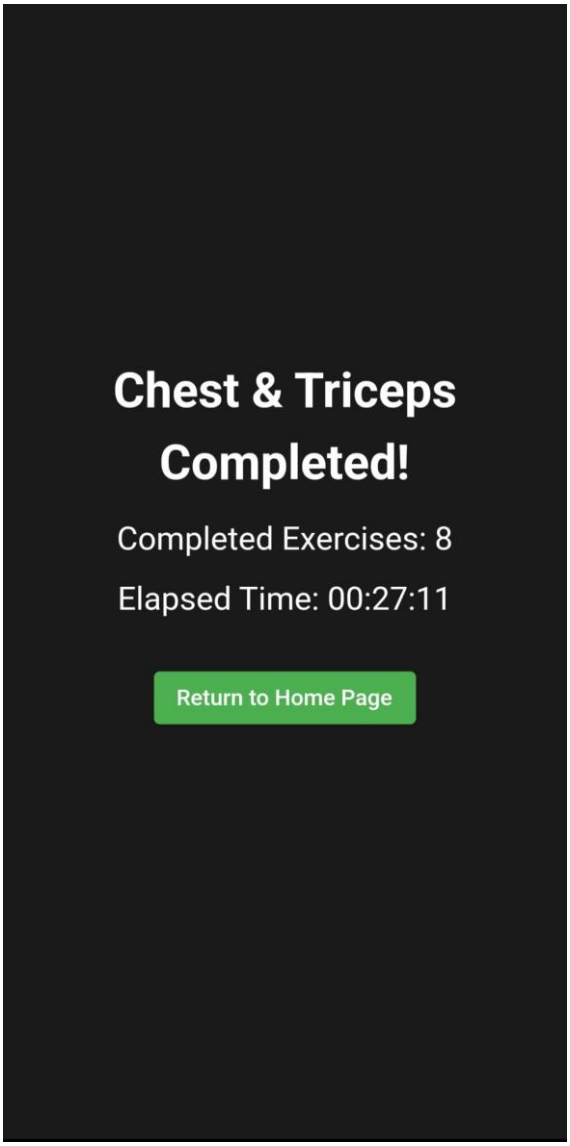
Za zaslon „Start a workout“ koristi se „Container“ widget kojemu je dijete widget „ListView.builder“. „ListView.builder“ omogućuje dinamičko stvaranje listi odnosno „ListTile“ widgeta koji predstavlja jedan korisnički napravljeni program. U njemu se nalazi „Text“ widget u koji upisujemo ime programa. Pomoću „trailing“ svojstva na rub desne strane svakog kontejnera smješta se „IconButton“ gumb koji pritiskom vodi korisnika na novi zaslon. Na novom zaslonu u gornjem području možemo vidjeti ime odabranog programa i desno od njega gumb kojim završavamo trening. Ispod imena nalazi se štoperica koja mjeri proteklo vrijeme te se može pokrenuti i zaustaviti pritiskom na gumb desno od nje. Kao i u prošlom zaslonu implementacija je ostvarena pomoću „ListView.builder“ widgeta. Dodatno sa desne strane svake zasebne vježbe nalazi se kućica čiji nam pritisak daje vizualnu potvrdu da smo završili vježbu na način da se kroz ime vježbe provuče crta te kućica poprima zelenu boju. Kada korisnik završi program može stisnuti „IconButton“ desno od imena programa koji ga vodi na „WorkoutCompletionPage“ zaslon na kojemu se nalazi tekst koji sadrži ime programa i informaciju da je program završen. Ispod se nalaze tekstovi koji nam daju informacije o broju završenih vježbi u programu i ukupno trajanje treninga.

```
@override
Widget build(BuildContext context) {
  return Consumer<WorkoutData>(
    builder: (context, value, child) => Scaffold(
      backgroundColor: Colors.white10,
      appBar: AppBar(
        backgroundColor: Colors.black26,
        title: Text("Select a workout to begin:"),
      ),  // AppBar
      body: Container(
        decoration: BoxDecoration(
          color: Colors.white10,
        ),  // BoxDecoration
        child: ListView.builder(
            itemCount: value.getWorkoutList().length,
            itemBuilder: (context, index) => Padding(
              padding: const EdgeInsets.only(left: 8.0,right: 8.0,bottom: 8.0, top: 15),
              child: Container(
                decoration: BoxDecoration(
                  color: Colors.black,
                  borderRadius: BorderRadius.circular(10),
                ),  // BoxDecoration
                child: ListTile(
                  title: Text(
                    value.getWorkoutList()[index].name,
                    style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold,color: Colors.white),
                  ),  // Text
                  trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      IconButton(
                        icon: Icon(Icons.start,color: Colors.white),
                        onPressed: () => goToWorkoutPage(
                            value.getWorkoutList()[index].name),
                      ),  // IconButton
                    ],
                  ),  // Row
                ),  // ListTile
              ),  // Container
```

Sl. 4.2.1. Prikaz implementacije „Start a workout“ zaslona



Sl. 4.2.2. Prikaz zaslona „Start a workout“



Sl. 4.2.3. Prikaz „WorkoutCompletionPage“ zaslona

4.3. „Goals & Calendar“ zaslon

Zaslon „Goals & Calendar“ implementiran je kao „Column“ widget koji se sastoji se od tri dijela. Prvi dio je „Container“ widget kojeg smo uredili sa rubom pomoću „BoxDecoration“ svojstva, također smo podesili visinu koju želimo zauzeti na zaslonu. U tom kontejneru nalazi se naš kalendar pod nazivom „MyHeatMap“ (Slika 4.3.1.).

```
167        body: Column(
168          children: [
169            Container(
170              decoration: BoxDecoration(
171                border: Border.all(color: Colors.white),
172                color: Colors.black,
173              ),  // BoxDecoration
174              height: MediaQuery.of(context).size.height *
175                  0.5, // Adjust the height as needed
176              child: MyHeatMap(
177                datasets: value.heatMapDataSet,
178                startDateYYYYMMDD: value.getStartDate(),
179              ),  // MyHeatMap
180            ),  // Container
```

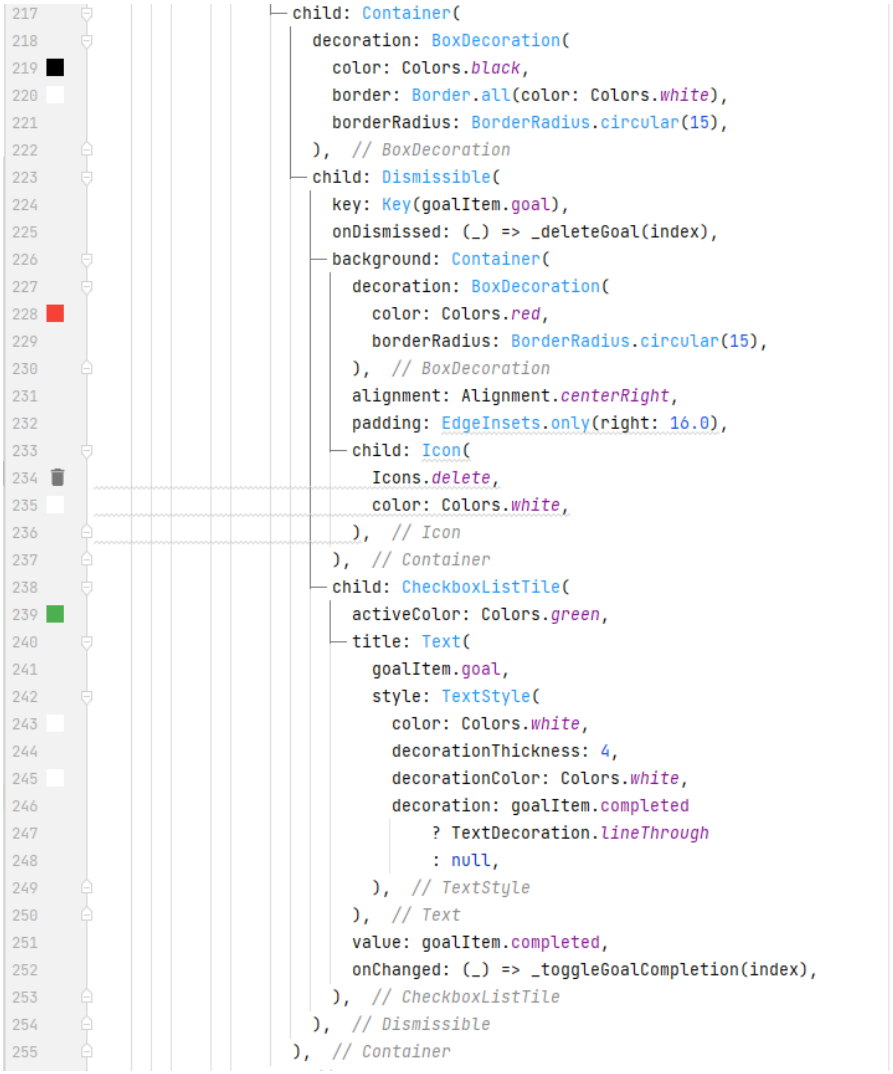
Sl. 4.3.1. Implementacija Kalendara na zaslonu „Goals and calendar“

Zatim korištenjem „SizedBox“ widgeta stvaramo razmak između kontejnera koji sadrži kalendar i između widgeta „Row“. Pomoću „Row“ widgeta stvara se red u koji smještamo „Text“ widget u kojem piše „My goals:“ desno od njega nalazi se „FloatingActionButton.extended“ widget koji nam predstavlja gumb sa kojim dodajemo nove ciljeve u naš popis ciljeva (Slika 4.3.2.) Nastavak „.extended“ nam omogućava da dodatno uredimo standardni „FloatinActionButton“ gumb.

```
181            SizedBox(height: 20),
182            Row(
183              children: [
184                SizedBox(width: 10),
185                Text(
186                  "My goals:",
187                  style: TextStyle(
188                      color: Colors.white,
189                      fontSize: 20,
190                      fontWeight: FontWeight.bold),  // TextStyle
191                ),  // Text
192                Expanded(child: Container()),
193                // Empty container to expand and fill the remaining space
194                FloatingActionButton.extended(
195                  onPressed: () => _showAddGoalDialog(context),
196                  label: const Text(
197                    'Add a new goal',
198                    style: TextStyle(fontWeight: FontWeight.bold),
199                  ),  // Text
200                  icon: const Icon(
201                    Icons.add,
202                    color: Colors.white,
203                  ),  // Icon
204                  backgroundColor: Colors.green,
205                ),  // FloatingActionButton.extended
206              ],
207            ),  // Row
```

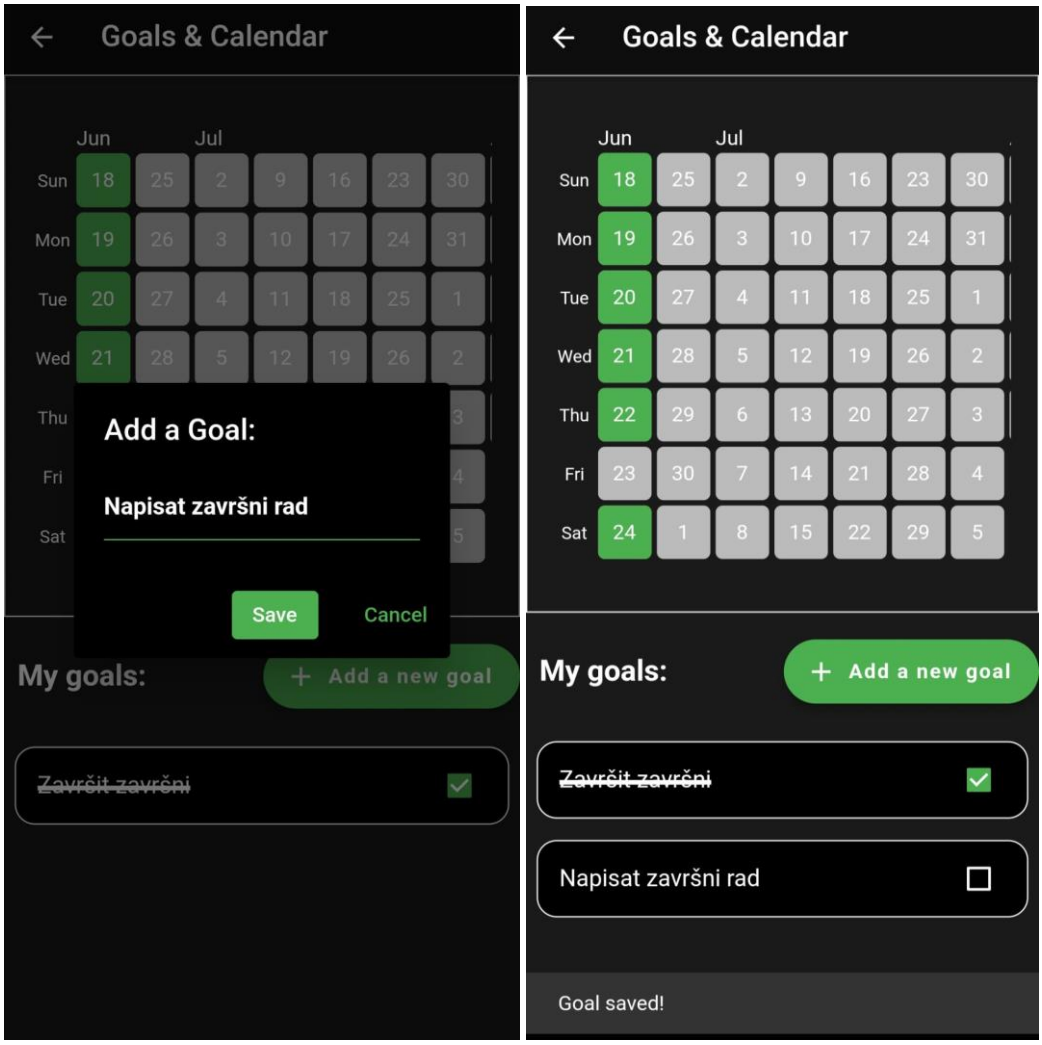
Sl. 4.3.2. Implementacija opisnog teksta i gumba za dodavanje novih ciljeva

Nadalje, moramo stvoriti dinamički popis korisnički generiranih ciljeva, a to ponovno postižemo pomoću „ListView.builder“ widgeta. U njemu se nalazi „Dismissible“ widget koji omogućava korisniku da klizanjem po cilju s lijeve ili s desne strane obriše odabrani cilj. On je ponovno umotan u „Container“ widget kako bi mogli urediti pozadinsku boju i okvir pomoću „BoxDecoration“ svojstva. Također, na kraju kontejnera s desne strane nalazimo kućicu za označavanje završenih ciljeva (Slika 4.3.3.). To nam omogućuje „CheckBoxListTile“ widget kojemu uređujemo aktivnu boju kućice, te postavljamo uvjet da ako je cilj označen kao odrađen kroz tekst se provlači linija koja daje vizualnu povratnu informaciju da je cilj ispunjen (Slika 4.3.4.).



```
child: Container(
  decoration: BoxDecoration(
    color: Colors.black,
    border: Border.all(color: Colors.white),
    borderRadius: BorderRadius.circular(15),
  ), // BoxDecoration
  child: Dismissible(
    key: Key(goalItem.goal),
    onDismissed: (_) => _deleteGoal(index),
    background: Container(
      decoration: BoxDecoration(
        color: Colors.red,
        borderRadius: BorderRadius.circular(15),
      ), // BoxDecoration
      alignment: Alignment.centerRight,
      padding: EdgeInsets.only(right: 16.0),
      child: Icon(
        Icons.delete,
        color: Colors.white,
      ), // Icon
    ), // Container
    child: CheckboxListTile(
      activeColor: Colors.green,
      title: Text(
        goalItem.goal,
        style: TextStyle(
          color: Colors.white,
          decorationThickness: 4,
          decorationColor: Colors.white,
          decoration: goalItem.completed
              ? TextDecoration.lineThrough
              : null,
        ), // TextStyle
      ), // Text
      value: goalItem.completed,
      onChanged: (_) => _toggleGoalCompletion(index),
    ), // CheckboxListTile
  ), // Dismissible
), // Container
```

Sl. 4.3.3. Implementacija liste ciljeva na zaslonu „Goals & Calendar“



Sl. 4.3.4 Izgled zaslona „Goals & Calendar“ i dodavanje novog cilja

Za trajno spremanje ciljeva koristi se „Shared Preferences“. Pomoću njega možemo spremiti male količine podataka na osnovi ključ-vrijednost parova u lokalnu memoriju uređaja. Funkcija _initSharedPreferences() (Slika 4.3.5.) je asinkrona i ne vraća ništa (void). Ako postoje spremljeni ciljevi, oni se dekodiraju iz JSON formata u odgovarajuće Dart objekte i pohranjuju u _goals listu. Ako ne postoje spremljeni ciljevi, _goals se postavlja kao prazna lista. Kreira se kontroler _goalController za unos novih ciljeva. Na kraju se poziva setState() kako bi se obavijestio Flutter framework da se stanje promijenilo i da se odgovarajući widgeti trebaju ponovno izgradit.

```dart
Future<void> _initSharedPreferences() async {
  _preferences = await SharedPreferences.getInstance();
  List<String>? savedGoalsJson = _preferences.getStringList('goals');

  if (savedGoalsJson != null) {
    List<dynamic> savedGoals =
        savedGoalsJson.map((json) => jsonDecode(json)).toList();

    _goals = savedGoals
        .map((goalMap) =>
            GoalItem(goalMap['goal'], goalMap['completed'] ?? false))
        .toList();
  } else {
    _goals = [];
  }

  _goalController = TextEditingController();

  setState(() {});

}
```

Sl. 4.3.5. Implementacija Shared Preferences

„GoalItem“ koji predstavlja cilj je klasa koja se sastoji od Stringa „goal“ koji predstavlja ime cilja i boolean vrijednost „completed“ koja služi za praćenje da li je cilj ispunjen ili ne.

```dart
class GoalItem {
  final String goal;
  bool completed;

  GoalItem(this.goal, this.completed);
}
```

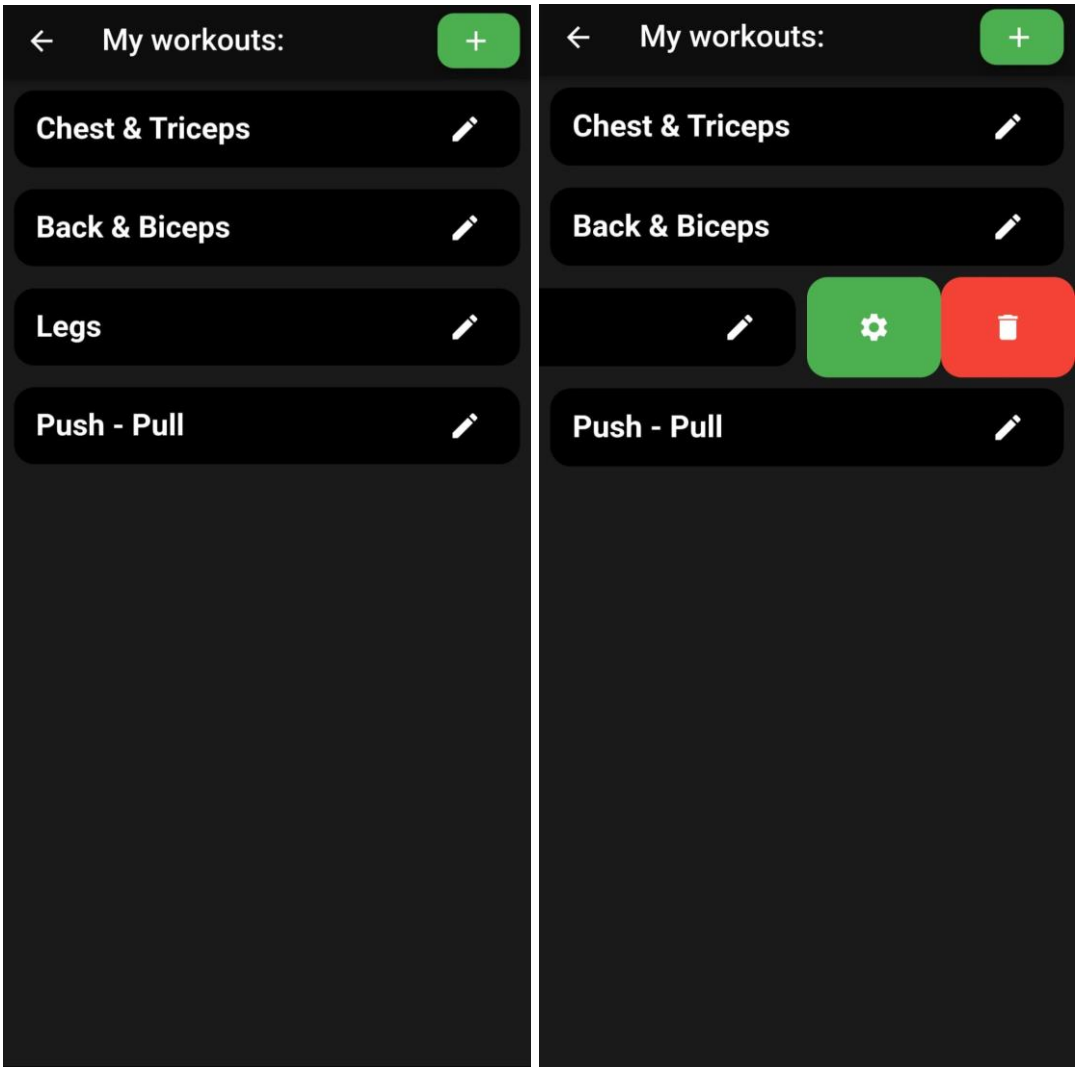
Sl. 4.3.6. Implementacija klase „GoalItem“

4.4. „My workouts“ zaslon

Zaslon „My workouts“ vrlo je sličan zaslonu „Start a workout“. „ListView.builder“ koristi se za prikazivanje liste programa. Svaki program je prikazan unutar „Slidable“ widgeta koji omogućuje korisniku otkrivanje dodatnih akcija (preimenovanje i brisanje programa) povlačenjem programa lijevo. Za preimenovanje programa koristi se metoda „renameWorkout“ koja otvara dijaloški okvir koji omogućuje korisniku preimenovanje vježbe. Uneseno novo ime se sprema pomoću „renameWorkout“ metode iz „WorkoutData“ objekta. Metoda „deleteWorkout“ briše odabranu vježbu pomoću deleteWorkout metode iz WorkoutData objekta .Unutar „ListView.builder-a“, svaki program je prikazan unutar „Container“ widgeta s odgovarajućim stilovima i sadrži naslov vježbe i gumb za uređivanje programa „IconButton“ koji vodi korisnika na novi zaslon gdje dodaje specifične vježbe u odabrani program.

```
207         body: ListView.builder(
208             itemCount: value.getWorkoutList().length,
209             itemBuilder: (context, index) => Slidable(
210                   endActionPane: ActionPane(
211                     motion: StretchMotion(),
212                     children: [
213                       SlidableAction(
214                         onPressed: (context) => renameWorkout(
215                           value.getWorkoutList()[index].name,
216                         ),
217                         icon: Icons.settings,
218                         backgroundColor: Colors.green,
219                         borderRadius: BorderRadius.circular(15),
220                       ),
221                       SlidableAction(
222                         onPressed: (context) =>
223                             deleteWorkout(value.getWorkoutList()[index].name),
224                         icon: Icons.delete,
225                         backgroundColor: Colors.red,
226                         borderRadius: BorderRadius.circular(15),
227                       ),
228                     ],
229                   ),
230                   child: Padding(
231                     padding: const EdgeInsets.all(8.0),
232                     child: Container(
233                       decoration: BoxDecoration(
234                         color: Colors.black,
235                         borderRadius: BorderRadius.circular(15),
236                       ),
237                       child: ListTile(
238                         title: Text(
239                           value.getWorkoutList()[index].name,
240                           style: TextStyle(
241                               fontSize: 20,
242                               fontWeight: FontWeight.bold,
243                               color: Colors.white),
```

Sl. 4.4.1. Implementacija prikaza programa na zaslonu „My workouts“



Sl. 4.4.2. Izgled zaslona „My workouts“ i prikaz funkcionalnosti „Slidable“ widgeta

Također, prikazuje se „AppBar“ s naslovom "My workouts" i s desne strane nalazi se gumb za dodavanje novog programa. Metoda „createNewWorkout“ otvara dijaloški okvir („AlertDialog“) koji omogućuje korisniku stvaranje novog programa. Uneseno ime programa sprema se pomoću „addWorkout“ metode iz „WorkoutData“ objekta.

```
void createNewWorkout() {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      backgroundColor: Colors.black,
      title: Text(
        "Create a new workout!",
        style: TextStyle(
          color: Colors.white,
          fontWeight: FontWeight.bold,
        ),
      ),
      content: TextField(
        controller: newWorkoutNameController,
        decoration: InputDecoration(
          hintText: "New workout name:",
          hintStyle: TextStyle(color: Colors.white),
          enabledBorder: UnderlineInputBorder(
            borderSide: BorderSide(color: Colors.green),
          ),
          focusedBorder: UnderlineInputBorder(
            borderSide: BorderSide(color: Colors.green),
          ),
        ),
        style: TextStyle(
          color: Colors.white,
          fontWeight: FontWeight.bold,
        ),
      ),
      actions: [
        ElevatedButton(
          onPressed: save,
          child: Text(
            "Save",
            style: TextStyle(color: Colors.white),
          ),
          style: ElevatedButton.styleFrom(
            primary: Colors.green,
          ),
```

Sl. 4.4.3. Dio implementacije metode „createNewWorkout“

Od važnijih metoda spomenuo bih „build“ metodu koja je odgovorna za izgradnju sučelja koje se prikazuje na zaslonu. Ova metoda se automatski poziva kada je potrebno ažurirati sučelje widgeta. U „build“ metodi, widget se omotava s „Consumer“ widgetom, što omogućuje pristup podacima iz WorkoutData objekta. Ovdje se koristi value.getWorkoutList() kako bi se dohvatila lista vježbi. Nadalje, imamo metodu „initState“ koja služi za inicijalizaciju podataka ili postavljanje početnog stanja widgeta. Ova metoda se poziva samo jednom, prije nego što se metoda „build“ izvrši prvi put. „InitState“ metoda koristi se za inicijalizaciju popisa vježbi unutar „WorkoutData“ objekta. Pozivanje „Provider.of<WorkoutData>(context, listen: false)“ omogućuje pristup WorkoutData objektu.

Korištenje „listen: false“ znači da widget neće pratiti promjene u podacima koje pruža „WorkoutData“, čime se izbjegava nepotrebno ponovno iscrtavanje widgeta ako dođe do promjena u podacima.

```
20    @override
21    void initState() {
22      super.initState();
23      Provider.of<WorkoutData>(context, listen: false).initializeWorkoutList();
24    }
```

Sl. 4.4.4. Implementacija „initState“ metode

Kada korisnik odabere željeni program otvara se novi zaslon „WorkoutPage“ gdje korisnik može dodavati i brisati vježbe. Lista vježbi ponovno se prikazuje pomoću „ListView.builder“ widgeta. Zasebna vježba prikazuje se pomoću „ExerciseTile“ widgeta. „ExerciseTile“ je widget koji prikazuje pojedinu vježbu unutar liste. Vanjski okvir „Container“ koristi se za pružanje vizualnog stila vježbi. Definirana je crna pozadina s zaobljenim rubovima. Unutar „Container“ widgeta nalazi se „Slidable“ widget koji omogućuje dodavanje mogućnosti klizanja za brisanje vježbe. Postavljena je samo jedna akcija, koja je brisanje vježbe. Kada se pritisne gumb za brisanje, poziva se metoda deleteExercise. „Child“ unutar Slidable widgeta je ListTile, koji prikazuje detalje vježbe. Title prikazuje naziv vježbe („exerciseName“). Ako je vježba označena kao završena („isCompleted“), stil teksta će sadržavati kroz crtu. „Subtitle“ prikazuje dodatne informacije o vježbi, kao što su težina („weight“), broj ponavljanja („reps“) i broj serija („sets“). Ove informacije su prikazane kao čipovi („Chip“). „Trailing“ prikazuje potvrdni okvir („checkbox“) samo ako je „showCheckBox“ postavljen na true.

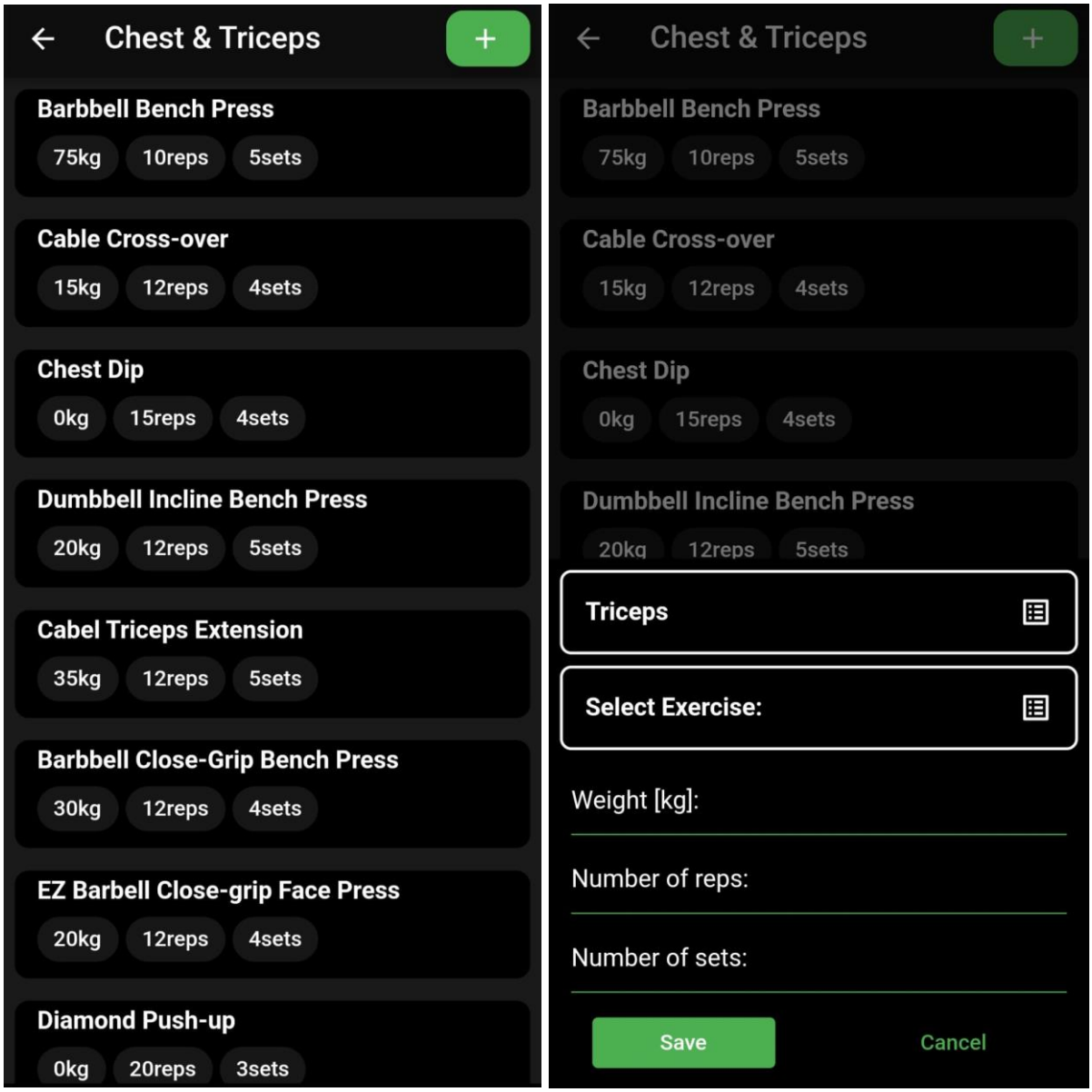
```
child: Container(
  decoration: BoxDecoration(
    color: Colors.black,
    borderRadius: BorderRadius.circular(10),
  ),  // BoxDecoration
  child: Slidable(
    endActionPane: ActionPane(
      motion: StretchMotion(),
      children: [
        SlidableAction(
          onPressed: deleteExercise,
          icon: Icons.delete,
          backgroundColor: Colors.red,
          borderRadius: BorderRadius.circular(15),
        )  // SlidableAction
      ],
    ),  // ActionPane
    child: ListTile(
      title: Text(exerciseName,
          style: TextStyle(
              decorationThickness: 4,
              color: Colors.white,
              fontWeight: FontWeight.bold,
              decoration: isCompleted
                  ? TextDecoration.lineThrough
                  : TextDecoration.none)),  // TextStyle, Text
      subtitle: Row(
        children: [
          Chip(
            label: Text(
              "${weight}kg",
              style: TextStyle(color: Colors.white),
            ),  // Text
          ),  // Chip
```

Sl. 4.4.5. Dio implementacije „ExerciseTile“ widgeta

Korisnik dodaje novu vježbu pritiskom gumba u gornjem desnom kutu. Pritiskom na gumb pokreće se funkcija createNewExercise. Ova funkcija otvara modalni donji list(„showModalBottomSheet“) za stvaranje nove vježbe. Modalni donji list sadrži nekoliko komponenti. „Container“ s „ListTile-om“ koji prikazuje odabranu kategoriju. Kada korisnik pritisne „ListTile“, poziva se funkcija openCategoryDrawer, koja otvara drugi modalni donji list za odabir kategorije vježbi. Ako je odabrana kategorija (varijabla „selectedCategory“ nije null), prikazuju se još tri ListTile-a. Svaki „ListTile“ sadrži „TextField“ za unos težine, broja ponavljanja i broja serija za vježbu. Unesene vrijednosti pohranjuju se u odgovarajuće kontrolere teksta („exerciseWeightController“, „exerciseRepsController“, „exerciseSetsController“). Na kraju, prikazuju se dva „ElevatedButton-a“ ("Save" i "Cancel") smještene u „Row“. Gumb "Save" poziva funkciju „save“, a gumb "Cancel" poziva funkciju „cancel“.

```dart
void createNewExercise() {
  showModalBottomSheet(
    backgroundColor: Colors.black,
    context: context,
    builder: (context) {
      return StatefulBuilder(
        builder: (BuildContext context, StateSetter setState) {
          return Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Container(
                  decoration: BoxDecoration(
                    color: Colors.black,
                    border: Border.all(
                      color: Colors.white,
                      width: 2.0,
                    ),  // Border.all
                    borderRadius: BorderRadius.circular(8.0),
                  ),  // BoxDecoration
                  child: ListTile(
                    trailing: Icon(Icons.list_alt, color: Colors.white),
                    title: Text(
                      selectedCategory ?? 'Select Category',
                      style: TextStyle(
                        color: Colors.white,
                        fontWeight: FontWeight.bold,
                      ),  // TextStyle
                    ),  // Text
                    onTap: () {
                      openCategoryDrawer(setState);
                    },
                  ),  // ListTile
                ),  // Container
              ),  // Padding
              if (selectedCategory != null) ...[
                Padding(
                  padding: const EdgeInsets.only(
                    left: 8.0,
```

Sl. 4.4.6. Dio implementacije funkcije „createNewExercise“



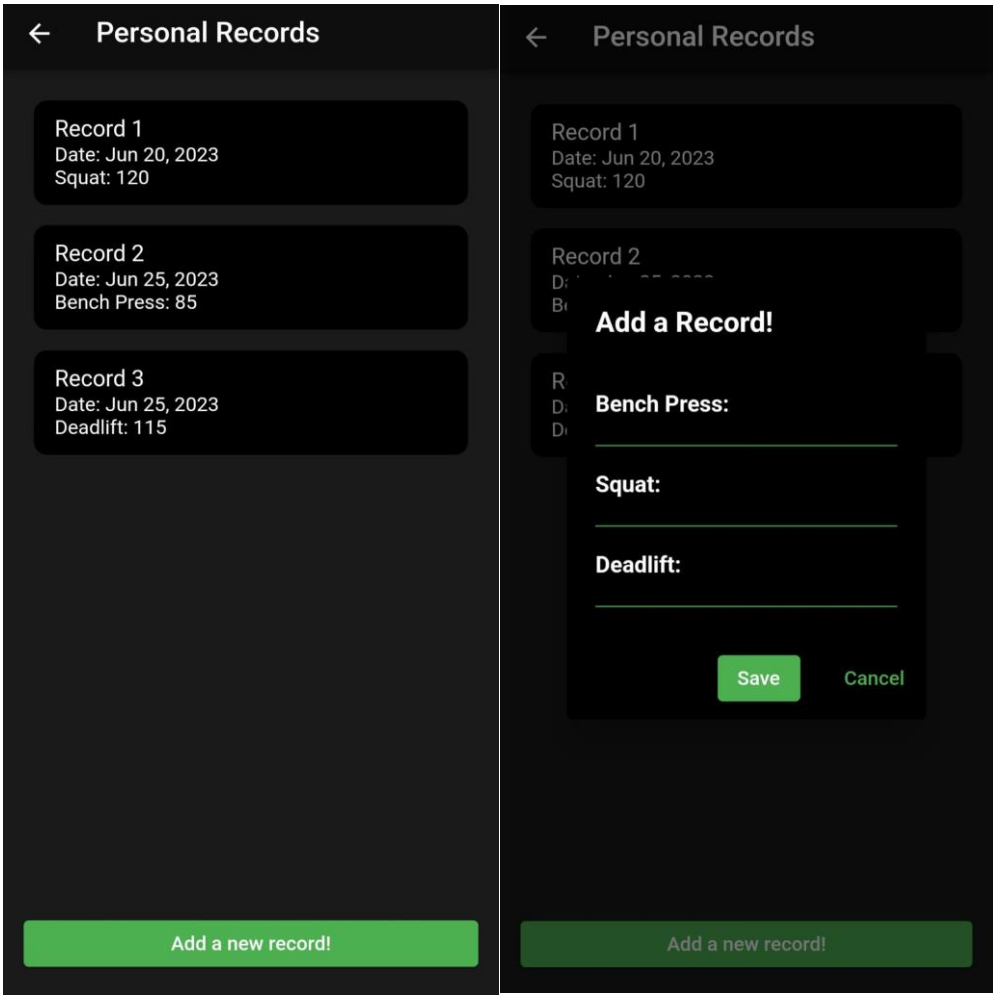
Sl. 4.4.7. Prikaz zaslona „WorkoutPage“ i dodavanje novih vježbi

4.5. „My PR's“ zaslon

Zaslon „My PR's“ omogućuje prikaz, unos i brisanje osobnih rekorda za bench press, čučanj i mrtvo dizanje. Metoda „buildRecordList“ prikazuje listu unosa rekorda pomoću „ListView.builder“. Svaki unos rekorda prikazuje se unutar „Container“ widgeta s mogućnošću brisanja klizanjem koristeći „Slidable“ widget. Nadalje, „showAddRecordDialog“ metoda otvara dijalog za unos novog rekorda. Unosi se provjeravaju putem forme _formKey, a zatim se dodaju u _recordsList. Metode „loadRecords“, „submitForm“ i „deleteRecord“ koriste se za manipulaciju podacima o rekordima. Metoda „dispose“ oslobađa resurse prilikom zatvaranja stranice.

```
105        Expanded(
106          child: ListView.builder(
107            itemCount: _recordsList.length,
108            itemBuilder: (context, index) {
109              final record = _recordsList[index];
110              final date = DateFormat('MMM dd, yyyy').format(record['date']);
111              final bench = record['bench'];
112              final squat = record['squat'];
113              final deadlift = record['deadlift'];
114
115              return Padding(
116                padding: const EdgeInsets.all(8.0),
117                child: Container(
118                  decoration: BoxDecoration(
119                    color: Colors.black,
120                    borderRadius: BorderRadius.circular(10),
121                  ),  // BoxDecoration
122                  child: Slidable(
123                    endActionPane: ActionPane(
124                      motion: StretchMotion(),
125                      children: [
126                        SlidableAction(
127                          onPressed: (context) {
128                            _deleteRecord(index);
129                          },
130                          icon: Icons.delete,
131                          backgroundColor: Colors.red,
132                          borderRadius: BorderRadius.circular(15),
133                        )  // SlidableAction
134                      ],
135                    ),  // ActionPane
136                    child: Card(
137                      color: Colors.black,
138                      margin: EdgeInsets.symmetric(vertical: 8),
139                      child: ListTile(
140                        title: Text(
141                          'Record ${index + 1}',
142                          style: TextStyle(color: Colors.white),
143                        ),  // Text
```

Sl. 4.5.1. Dio implementacije „buildRecordsList“



Sl. 4.5.2. Izgled zaslona „My PR's“ i dodavanje novog rekorda

4.6. Klase „HiveDatabase“ i „WorkoutData“

Jedna od najvažnijih klasa u aplikaciji je klasa „HiveDatabase“. Ova klasa omogućuje spremanje i čitanje podataka iz Hive baze podataka. Klasa se sastoji od nekoliko metoda od kojih su najvažnije navedene i objašnjene. Metoda „previousDataExists“ provjerava postoji li prethodno spremljeni podaci u bazi. Ako nema podataka, bilježi se trenutni datum kao početni datum. Metoda „saveToDatabase“ sprema predane objekte treninga u bazu. Prethodno pretvara objekte treninga i vježbi u odgovarajući format (listu stringova) kako bi se mogli spremiti u Hive bazu. Također provjerava jesu li neke vježbe obavljene i bilježi odgovarajući status. Metoda „readFromDatabase“ čita podatke iz baze i vraća listu objekata treninga. Iz baze se dobivaju imena treninga i detalji vježbi, a zatim se ti podaci pretvaraju natrag u objekte treninga i vježbi. Metoda „exerciseWasCompleted“ provjerava je li bilo koja vježba u predanim objektima treninga obavljena. Prolazi kroz sve treninge i vježbe te vraća „true“ ako je bilo koja vježba obavljena. Metoda „convertObjectToWorkoutList“ pretvara objekte treninga u listu stringova radi spremanja u bazu. Svaki objekt treninga se pretvara u ime treninga. Metoda „convertObjectToExerciseList“ pretvara objekte vježbi u listu listi stringova radi spremanja u bazu. Svaka vježba se pretvara u listu stringova koja sadrži ime, težinu, ponavljanja, serije i status obavljanja. Metoda „getStartDate“ vraća početni datum koji je zabilježen prilikom prvog spremanja podataka u bazu. Metoda „getCompletedStatus“ vraća status obavljanja treninga za određeni datum. Status može biti 0 (nije obavljeno) ili 1 (obavljeno), a ako za taj datum nema zabilježenog statusa, vraća se 0.

```
class HiveDatabase {
  // reference our Hive box
  final _myBox = Hive.box("workout_database_test2");

  // check if there is data stored if not record the save date
  bool previousDataExists() {
    if (_myBox.isEmpty) {
      print("Previous data does not exists");
      _myBox.put("START_DATE", todaysDateFormatted());
      return false;
    } else {
      print("Previous data does exists");
      return true;
    }
  }

  //write data
  void saveToDatabase(List<Workout> workouts) {
    final workoutList = convertObjectToWorkoutList(workouts);
    final exerciseList = convertObjectToExerciseList(workouts);

    //check if any exercises have been done
    if(exerciseWasCompleted(workouts)){
      _myBox.put("COMPLETION_STATUS_${todaysDateFormatted()}",1);
    }else{
      _myBox.put("COMPLETION_STATUS_${todaysDateFormatted()}",0);
    }

    // save into hive
    _myBox.put("WORKOUTS", workoutList);
    _myBox.put("EXERCISES", exerciseList);
  }

  //read data and return a list of workouts
  List<Workout> readFromDatabase() {
    List<Workout> mySavedWorkouts = [];

```

Sl. 4.6.1. Dio implementacije klase „HiveDatabase“

Klasa „WorkoutData“ predstavlja model podataka i upravljačku logiku za rad s treninzima i vježbama. Najvažnije metode i njihove funkcionalnosti objašnjene su u nastavku. Metoda „getWorkoutList“ vraća listu objekata treninga. Metoda „numberOfExercisesInWorkout“ vraća broj vježbi u određenom treningu. Metoda „numberOfCompletedExercises“ vraća broj obavljenih vježbi u određenom treningu. Metoda „addWorkout“ dodaje novi trening s zadanim imenom u listu treninga. Metoda „deleteWorkout“ briše trening s određenim imenom iz liste treninga. Metoda „renameWorkout“ mijenja ime treninga s određenim imenom. Metoda „addExercise“ dodaje novu vježbu u određeni trening. Metoda „deleteExercise“ briše vježbu s određenim imenom iz određenog

treninga. Metoda „checkOffExercise“ označava vježbu kao obavljenu u određenom treningu. Metoda „ResetExercies“ resetira status obavljanja svih vježbi u određenom treningu. Metoda „initializeWorkoutList“ inicijalizira listu treninga na temelju podataka iz baze ili postavlja početne vrijednosti ako nema prethodno spremljenih podataka. Metoda „getRelevantWorkout“ vraća odgovarajući objekt treninga na temelju imena treninga. Metoda „getRelevantExercise“ vraća odgovarajući objekt vježbe na temelju imena treninga i imena vježbe. Metoda „getStartDate“ vraća početni datum spremljen u bazi podataka. Metoda „loadHeatMap“ učitava podatke o obavljanju vježbi za svaki dan od početnog datuma do trenutnog datuma i stvara mapu s podacima za prikaz toplinskog prikaza („heatmap“) obavljanja vježbi. Ova klasa koristi „HiveDatabase“ klasu za komunikaciju s Hive bazom podataka. Također koristi „ChangeNotifier“ kako bi obavijestila o promjenama u podacima i osigurala ažuriranje korisničkog sučelja.

```
20
21    // get list of workouts
22    List<Workout> getWorkoutList() {
23      return workoutList;
24    }
25
26    //get lenght of workout
27    int numberOfExercisesInWorkout(String workoutName) {
28      Workout relevantWorkout = getRelevantWorkout(workoutName);
29      return relevantWorkout.exercises.length;
30    }
31
32    //number of completed exercies
33    int numberOfCompletedExercises(String workoutName) {
34      Workout relevantWorkout = getRelevantWorkout(workoutName);
35      int completedCount = 0;
36
37      for (Exercise exercise in relevantWorkout.exercises) {
38        if (exercise.isComepleted) {
39          completedCount++;
40        }
41      }
42
43      return completedCount;
44    }
45
46    //add a workout
47    void addWorkout(String name) {
48      workoutList.add(Workout(name: name, exercises: []));
49      notifyListeners();
50
51      db.saveToDatabase(workoutList);
52    }
53
54    void deleteWorkout(String name) {
55      workoutList.removeWhere((element) => element.name == name);
56      notifyListeners();
57      db.saveToDatabase(workoutList);
58    }
```

Sl. 4.6.2. Dio implementacije klase „WorkoutData“

```
66
67    //add exercise to workout
68    void addExercise(
69      String workoutName,
70      String name,
71      String weight,
72      String reps,
73      String sets,
74    ) {
75      // find relevant workout
76      Workout relevantWorkout = getRelevantWorkout(workoutName);
77      relevantWorkout.exercises.add(Exercise(
78        name: name,
79        weight: weight,
80        reps: reps,
81        sets: sets,
82        isComepleted: false,
83        wasCompleted: false,
84      ));
85      notifyListeners();
86      db.saveToDatabase(workoutList);
87    }
88
89    void deleteExercise(String workoutName, String exerciseName) {
90      print('Deleting exercise: $exerciseName from workout: $workoutName');
91      Workout relevantWorkout = getRelevantWorkout(workoutName);
92      relevantWorkout.exercises
93          .removeWhere((element) => element.name == exerciseName);
94      notifyListeners();
95      db.saveToDatabase(workoutList);
96    }
```

Sl. 4.6.3. Dio implementacije klase „WorkoutData“

5. ZAKLJUČAK

Aplikacija za planiranje vježbanja korisnicima omogućava kreiranje, uređivanje i brisanje treninga. Nadalje, omogućava korisnicima da preko kalendarskog prikaza prate dane u mjesecu kada su odradili trening, također korisnici mogu stvoriti popis ciljeva kao i zapisivat osobne rekorde u osnovnim vježbama. Aplikacija je napravljena u Flutter razvojnom okviru koji omogućava izgradnju lijepih i reaktivnih korisničkih sučelja što je i postignuto. Također omogućava pisanje jednog koda koji se može izvršavati na različitim platformama što je velika prednost Fluttera u odnosu na druge razvojne okvire. U svrhu spremanja podataka korišteni su „Hive“ i „shared_preferences“ gdje se „Hive“ koristi za spremanje strukturiranih skupova podataka odnosno treninga i informacija potrebnih za generiranje kalendara, na drugu ruku „shared_preferences“ se koristi za jednostavnije tipove podataka poput u ovom slučaju stringova. Zadovoljan sam sa konačnim proizvodom, sučelje je pristupačno, intuitivno i ugodno oku, također, spremanje podataka radi brzo i stabilno što mi je predstavljalo najveći problem u procesu izrade. Aplikaciju ću koristiti u svojim treninzima kako bi mi pomogla postići bolju organizaciju, a samim time i bolje rezultate. Aplikacija ima još puno mjesta za napredak, u smislu dodavanja novih funkcionalnosti i proširivanja implementiranih funkcionalnosti. U svrhu proširivanja bilo bi pogodno da se na kalendarskom prikazu može odabrati određeni datum kada je trening odrađen te da se mogu vidjeti informacije o treningu koji je odrađen taj dan. Nadalje, bilo bi poželjno imati vizualne prikaze pojedinih vježbi te kratki opis svake vježbe u cilju lakšeg razumijevanja i traženja željene vježbe. Kada pričamo o dodavanju novih funkcionalnosti, u ovakvim aplikacijama poželjno je imati funkcionalnosti poput kalkulatora za izračunavanje tjelesne masnoće (bodyfat), grafove i statistiku koji pomažu korisniku u praćenju svog napretka.

LITERATURA

[1] G. N. Ruegsegger1, F. W. Booth, Health Benefits of Exercise, Cold Spring Harbor Perspectives in Medicine 13, 8, a029694, August 2023.

[2] SosisApps, Google Play, My Workout Plan - Daily Workout Planner, Dostupno: https://play.google.com/store/apps/details?id=com.myworkoutplan.myworkoutplan. [Pristupljeno: 16.kol.2023.]

[3] Strong Fitness PTE. LTD., Google Play, Strong Workout Tracker Gym Log , Dostupno: https://play.google.com/store/apps/details?id=io.strongapp.strong. [Pristupljeno: 16.kol.2023.]

[4] Jefit Inc., Google Play, JEFIT Gym Workout Plan Tracker, Dostupno: https://play.google.com/store/search?q=jefit&c=apps. [Pristupljeno: 16.kol.2023.]

[5] Nike Inc., Google Play, Nike Training Club: Fitness, Dostupno: https://play.google.com/store/apps/details?id=com.nike.ntc. [Pristupljeno: 16.kol.2023.]

[6] Leap Fitness Group, Google Play, Home Workout – No Equipment, Dostupno: https://play.google.com/store/search?q=home+workout+no+equipment&c=apps. [Pristupljeno: 16.kol.2023.]

[7] Javatpoint, Android Studio, Dostupno: https://www.javatpoint.com/android-studio. [Pristupljeno: 16.kol.2023.]

[8] H. R. Esmaeel, Apply android studio (SDK) tools, International Journal of Advanced Research in Computer Science and Software Engineering 5, 5 May 2015.

[9] Javatpoint, What is Dart Programming, Dostupno: https://www.javatpoint.com/flutter-dart-programming. [Pristupljeno: 16.kol.2023.]

[10] Flutter, FAQ, Dostupno: https://docs.flutter.dev/resources/faq. [Pristupljeno: 16.kol.2023.]

[11] Amazon, What is Flutter, Dostupno: https://aws.amazon.com/what-is/flutter/. [Pristupljeno: 16.kol.2023.]

[12] pub.dev, hive 2.2.3, Dostupno: https://pub.dev/packages/hive [Pristupljeno: 16.kol.2023.]

SAŽETAK

Cilj ovog rada bio je napraviti aplikaciju za vježbanje koja korisnicima omogućava odabir raznih vježbi po kategorijama mišićnih skupina, u svrhu izrade plana treninga. Tijekom vježbanja aplikacija vodi korisnika kroz planirani popis vježbi. Također pruža unos određenih ciljeva koje korisnik želi postići. Kroz kalendarski prikaz omogućuje prikaz svih ostvarenih treninga te praćenje realiziranih i nerealiziranih ciljeva na pregledan način. Korisnik je i u mogućnosti zapisivanja osobnih rekorda u osnovnim vježbama. Aplikacija je izrađena u Android Studio-u, a pisana je Dart programskim jezikom u razvojnom okviru Flutter.

Ključne riječi: Aplikacija za planiranje vježbanja, Mišićne skupine, Plan treninga, Ciljevi vježbanja

ABSTRACT

The aim of this project was to develop a fitness application that allows users to choose various exercises based on muscle group categories, in order to create a workout plan. During the workout, the application guides the user through a planned list of exercises. It also allows the user to set specific goals they want to achieve. Through a calendar view, the application displays all completed workouts and provides a clear overview of achieved and unachieved goals. The user can also record personal records in basic exercises. The application was developed in Android Studio and written in the Dart programming language using the Flutter framework.

Keywords: Fitness planning application, Muscle groups, Training plan, Exercise goals