VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis bakalauro darbas

Sporto treniruočių programos parinkimo mobilioji programa

Fitness Training Program Selection Mobile Application

Atliko: 4 kurso, 2 grupės studentas

Tautvydas Milčiūnas (parašas)

Darbo vadovas:

dr. Joana Katina

Vilnius
2018

Turinys

Anotacija

Darbo tikslas yra sukurti automatinę sporto programos parinkimo sistemą pagal pateiktus naudotojo duomenis ir pritaikytą mobiliesiems įrenginiams. Šiame darbe yra aprašoma panašių sistemų analizė, mobiliosios programos projektavimas ir programavimas. Programėlėje yra keturios naudotojo būsenos: neprisijungęs, anoniminis, prisijungęs ir administratorius. Naudotojas gali užsiregistruoti, pateikti savo asmeninius duomenis ir gauti sporto programą pagal pateiktus duomenis. Taip pat, naudotojas gali prisijungti prie jau sukurtos paskyros ir matyti sukurtą sporto programą. Prisijungę naudotojai turi profilio langą, kuriame gali keisti savo asmeninius duomenis, ištrinti savo paskyrą ir perkurti sporto programą. Administratoriai turi visas paprasto naudotojo funkcijas, tačiau profilio lange yra specialios administratoriaus funkcijos: programos keitimas pasirinktam naudotojui, naudotojo pavertimas administratoriumi ir sporto pratimo kūrimas. Visi apsibrėžti reikalavimai buvo įgyvendinti ir užtikrintas sklandus jų veikimas su serveriu ir naudotojo sąsaja.

Raktiniai žodžiai: mobilioji programa, sportas, sveikata, automatizacija

Summary

Fitness Training Program Selection Mobile Application

The main goal of this thesis is to develop a fitness program selection mobile application. Report covers an analysis of similar systems, mobile application design, analysis and programming. The system has four user states: not logged in, anonymous, logged in and administrator. User can registrate, provide personal data, and see the generated fitness program when registration or login is completed. A user can also login to a registered account as it is saved in a database. Logged in user can see its profile with functions to recreate fitness program, delete account and change personal details. Administrators have all the same functions as a user but in addition, can make any user an administrator, recreate fitness program for any user and create new exercise. The goals that were set were completed and the system was made to be stable and work fluently with database as well as when user interacts with the system.

Keywords: mobile application, sport, health, automation

Įvadas

Šiuo darbu yra siekiama sukurti mobiliąją programėlę, kuri pagal pateiktus naudotojo duomenis automatiškai sugeneruotų sporto programą ir dietos planą. Visame pasaulyje šiuo metu yra naudojama milijardai išmaniųjų telefonų ir dauguma jų visada yra šalia naudotojo. Dėl šios priežasties nuspręsta kurti būtent mobiliąją programėlę. Žmonės, aktyviai leisdami laisvalaikį, turi turėti galimybę greitai pasiekti informaciją apie jų sporto programą, o programėlė tokią galimybę suteikia. Mobili programėlė yra pranašesnė nei internetinė sistema, nes yra lengviau pasiekiama, paprastesnė naudoti ir labiau pritaikyta prie mobilaus įrenginio veikimo ir interaktyvaus dizaino.

Darbo tikslas – sukurti automatizuotą sporto programos parinkimo mobiliąją programėlę. Ši programėlė, pagal naudotojo pateiktus duomenis, nustato tinkamą sporto programą.

Pagrindiniai darbo uždaviniai:

- Išanalizuoti panašias sistemas.
- Sukurti sistemos modelius bei diagramas.
- Sukurti registraciją, prisijungimą ir skirtingas naudotojo aplinkas.
- Sukurti sporto programos generavimo algoritmą.
- Panaudoti formules, skirtas naudotojo fiziniams duomenims paversti į kūno masės indeksą, rekomenduojamą svorį ir reikiamą kalorijų suvartojimą per dieną.

Sistemos kūrimui buvo pasirinktas *JavaScript* programavimo kalbos *React Native,* karkasas. Išanalizavus rinkoje esančius karkasus pasirinkta naudoti *Expo* platforma, kuri turi integruotus servisus, dažniausiai naudojamus programuojant *React Native* karkasu. Duomenims saugoti buvo pasirinkta *Google Firebase* platforma. Tai yra realiuoju laiku atnaujinama duomenų bazė, kuri veikia kaip servisas (ang. *Backend-as-a-Service*).

1. Susijusių darbų analizė

Prieš pradedant kurti mobiliąją programėlę buvo nuspręsta atlikti panašių sistemų tyrimą tam, kad remiantis gautais palyginimo rezultatais būtų galima išsiaiškinti ar norima kurti programėlė neatsilieka nuo rinkos lyderių.

Pirmoji analizuota programėlė „Gym Mentor“ yra skirta tik sporto programoms ir nesuteikia individualizuotų sporto pratimų ar dietos planų pasirinkimo. Šioje programėlėje nereikia registruoti paskyros ir nėra jokio suasmeninto nuoseklaus naudojimosi gido. Programėlėje naudojama daug specifinių terminų, kurie gali būti painūs pradedančiam sportuoti naudotojui. Taip pat yra labai daug reklamų, kurios trukdo įvesti duomenis ir gadina naudojimosi patirtį.

Antroji analizuota programėlė „Fitness & Bodybuilding” leidžia naudotojui susikurti savo treniruočių planą pasirenkant norimą dienų skaičių sportui. Pratimus į programą naudotojas turi pridėti iš pateiktų pratimų sąrašo, todėl programa nėra generuojama automatiniu būdu. Taip pat programėlėje galima matyti pavienius pratimus bei pratimų grupes, skirtas skirtingoms raumenų grupėms treniruoti. Šioje programėlėje taip pat trūksta naudotojo asmeniškumo ir individualių nustatymų, nes negalima sužinoti, kurie pratimai tinka būtent kiekvienam atskiram naudotojui.

Šioje programėlėje taip pat nėra asmeniškai nustatomų naudotojo fizinių duomenų įvedimo. Yra pateikiamas sąrašas iš anksto sukurtų sporto programų, kurias galima pasirinkti ir matyti progresą, išsidėsčiusį mėnesio eigoje. Yra gaunami pranešimai apie progresą ir priminimai apie ateinančias treniruotes.

Šioje programėlėje yra registracija, po kurios seka asmeninių duomenų užpildymas: tikslas, aktyvumo kiekis, lytis, gimimo data ir vieta, ūgis ir svoris, tikslas kurį norima pasiekti. Toliau buvo parodytas mano dienos kalorijų tikslas ir liepta pasirinkti koks paskutinis maistas buvo vartojamas, tam, kad apskaičiuoti, kiek buvo suvartota kalorijų tą dieną. Taip pat yra leidžiama planuoti maistą iš anksto pasirenkant iš produktų sąrašo. Programėlė labai patogi dietos planavimui, tačiau pati automatiškai negeneruoja reikiamų vartoti produktų sąrašų.

Išanalizavus minėtąsias programėles, pastebima, jog vyrauja registracijos nebuvimo tendencija. Taip pat daugumoje yra suasmenintų parametrų įvedimo trūkumas ir automatiškai generuojamų sporto programų bei dietos planų nebuvimas (žr. 1 lentelė).

1 lentelė. Panašių programėlių analizė

Programėlė	Registracija	Programėlės išbandymas	Sporto plano sudarymas	Dietos plano sudarymas
„Gym Mentor“	Nėra	Nėra	Nėra	Nėra
„Fitness & Bodybuilding“	Nėra	Nėra	Dalinai	Nėra
„7 minute workout“	Nėra	Nėra	Nėra	Nėra
„MyFitnessPal“	Nėra	Nėra	Nėra	Yra

Apibendrinus visas apžvelgtas programėles buvo pastebėta tendencija, kad vyrauja sistemos naudojimas be registracijos. Tai padaro sistemą paprastesnę, tačiau nėra galimybės dirbti su asmeniniais naudotojo duomenimis ir kurti specifikuotų produktų pritaikytų būtent konkrečiam naudotojui. Taip pat nėra pateikiamas programėlės išbandymas, bet tai yra susiiję su registracijos nebuvimu, kadangi jos apeiti nereikia, todėl ir programėlės išbandymui prasmės nėra. Automatinis sporto planų sudarymas šiose sistemose taip pat neegzistuoja, kadangi tai būtų sudėtinga realizuoti neturint asmeninių naudotojo duomenų. Būtų įmanoma sukurti suasmenintą sporto programą, jeigu žmogus įvestų savo duomenis be registracijos, tačiau tokiu atveju sporto programa nebūtų atgaunama atsisiuntus sistemą į kitą įrenginį, todėl tokia problemos sprendimo išeitis irgi nebūtų tinkama. Viena iš peržiūrėtų sistemų buvo skirta dietos plano sudarymui. Ši sistema yra panašiausia į mano kuriamą sistemą, nes įvedus asmeninius duomenis yra pateikiamas dietos planas.

2. Panašios internetinės sistemos analizė

Panaši internetinė sistema buvo sukurta vieno iš VU MIF studentų bakalaurinio darbo metu [Pov17]. Sistema sukurta *Python* programavimo kalba su *Django* karkasu. Duomenų valdymui buvo pasirinkta *PostgreSQL* duomenų bazių valdymo sistema. Išorinės naudotojo sąsajos kūrimui buvo pasirinktos šios technologijos: *HTML, CSS, JavaScript, jQuery, AJAX* ir *Bootstrap* karkasas.

2.1. Duomenų bazė

Panašioje internetinėje sistemoje naudota *PostgreSQL* duomenų bazių valdymo sistema. Priedas nr. 9 pateikta panašios sistemos duomenų bazės diagrama. Lentelėse yra pasikartojančios ir nereikalingos informacijos, kurią galima apjungti ir taip optimizuoti algoritmų veikimą. Pavyzdžiui, lentelės „diet_products“ ir „diet_productgroups“ gali būti apjungtos į vieną „diet_productslist“ lentelę. Lentelių, skirtų sporto pratimams saugoti, taip pat yra per daug. Efektyvesnis būdas saugoti sporto pratimus būtų lentelėje „pratimai“, saugoti po vieną pilną pratimą. Tuomet iš „pratimai“ lentelės formuoti atskiras pratimų grupes lentelėje „programos“. Tokiu pačiu būdu galima optimizuoti ir dietos generavimui skirtas lenteles.

2.2. Naudotojo sąsaja

Panašioje sistemoje yra sukurta tik pati paprasčiausia – bazinė, naudotojo sąsaja. Priedas nr. 10 pateikta panašios sistemos vartotojo sąsaja. Mano kuriamoje sistemoje naudotojo sąsajai yra skiriamas daug didesnis dėmesys - sistema prisitaiko prie mobilaus įrenginio dydžio, nėra iškraipomas tekstas, dizaino elementai prisitaiko savo dydžiu ir forma. Panašioje sistemoje yra panaudotas *Bootstrap* dizaino kūrimo šablonas, ir visa informacija pateikta viename lange. Sistemos dizainas turėtų būti interaktyvus, su kuo mažiau informacijos viename lange, taip išvengiant naudotojo nesusipratimo ir nežinojimo, kaip elgtis tam tikruose scenarijuose. Dėl šios priežasties, šiame darbe kuriamoje sistemoje, buvo išvengta panašios sistemos naudotojo sąsajos kūrimo klaidų.

3. Sistemos architektūra

Sistemos architektūra susideda iš kelių dalių: sistemos reikalavimų, sistemos teorinio modeliavimo, panaudos atvejų diagramos, technologijų pasirinkimo, duomenų bazės modeliavimo. Visos šios dalys sudaro pilną sistemos architektūrą ir nustato reikalavimus sistemos programavimui.

3.1. Sistemos reikalavimai

Sistemai yra keliami funkciniai ir nefunkciniai reikalavimai, kurie padės įgyvendinti darbo tikslą.

Funkciniai sistemos reikalavimai:

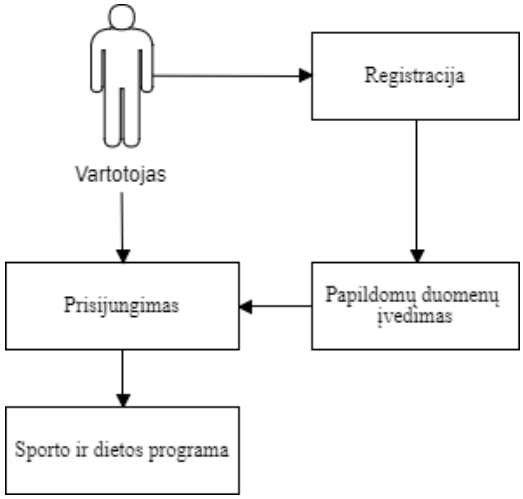
1. Prisijungimo ir registracijos su papildomu duomenų įvedimu formos.
2. Naudotojo duomenų saugojimas duomenų bazėje.
3. Sporto programos automatinis generavimas pasitelkiant papildomus asmeninius naudotojo duomenis.
4. Sugeneruotos sporto programos saugojimas duomenų bazėje.
5. Sugeneruotos sporto programos atvaizdavimas, keitimas pagal datą ir detalesnis sporto pratimo atvaizdavimas.
6. Naudotojo profilio langas su ištrynimo ir redagavimo funkcijomis.
7. Administracinis modulis profilio lange, pasiekiamas administratoriaus tipo naudotojų, kuriame būtų galimybė pakeisti sporto programą bei padaryti pasirinktą naudotoją administratoriumi.

Nefunkciniai sistemos reikalavimai:

1. Kuo mažesnis atvaizduojamos informacijos kiekis. Tai supaprastina vaikščiojimą tarp skirtingų langų ir padidina intuityvumą naudojantis programėle.
2. Programėlės dizainas buvo kurtas pagal naujausias dizaino tendencijas, minimas [BL14] šaltinyje. Šioje knygoje kalbama apie tinkamą spalvų pasirinkimą, elementų išdėstymą, kiek informacijos turi būti pateikiama viename lange ir kaip ji turi būti formuojama.

3.2. Teorinis sistemos modelis

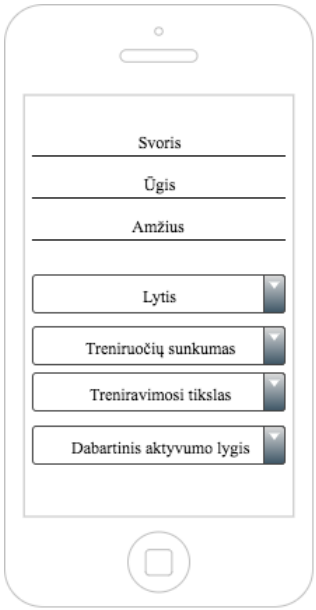
Programėlė yra suskirstyta į dvi pagrindines dalis: naujo naudotojo registraciją ir individualių duomenų įvedimą (žr. 1 pav.). Pirmoji dalis yra būtina norint automatiškai generuoti sporto bei dietos programas kiekvienam naudotojui pagal jo įvestus asmeninius duomenis. Antrojoje dalyje yra vaizduojami sugeneruoti programų rezultatai.



1 pav. Naudotojo registracija ir prisijungimas

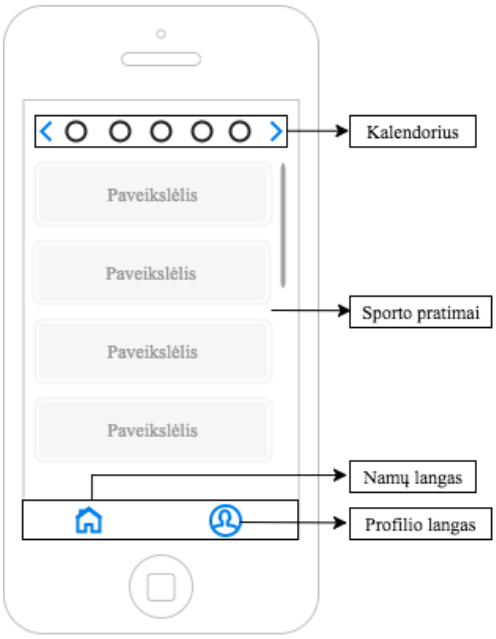
3.3. Sistemos naudotojo sąsaja

Sąsajos planavimas yra svarbi sistemos architektūros dalis, kurioje yra nusprendžiama, koks funkcionalumas bus kuriamas ir kaip komponentai turi atvaizduoti norimus elementus bei kokia informacija turi būti gaunama ir siunčiama kitiems komponentams.



2 pav. Naudotojo profilio langas

2 pav. yra pavaizduotas naudotojo asmeninių duomenų įvedimo langas, kurį užpildžius gaunami pagrindiniai naudotojo asmeniniai duomenys, reikalingi sporto programos generavimui, kūno masės indekso, kalorijų suvartojimui per dieną ir rekomenduojamo svorio apskaičiavimams. Svoris, ūgis ir amžius yra ranka įrašomos reikšmės su apribojimais ir patikrinimu po formos užpildymo. Lytis, treniruočių sunkumas, treniravimosi tikslas ir dabartinis aktyvumo lygis yra sąrašo tipo elementai, turintys kelis pasirinkimus, kurių negalima įrašyti rankiniu būdu, todėl jų tikrinti nereikia.



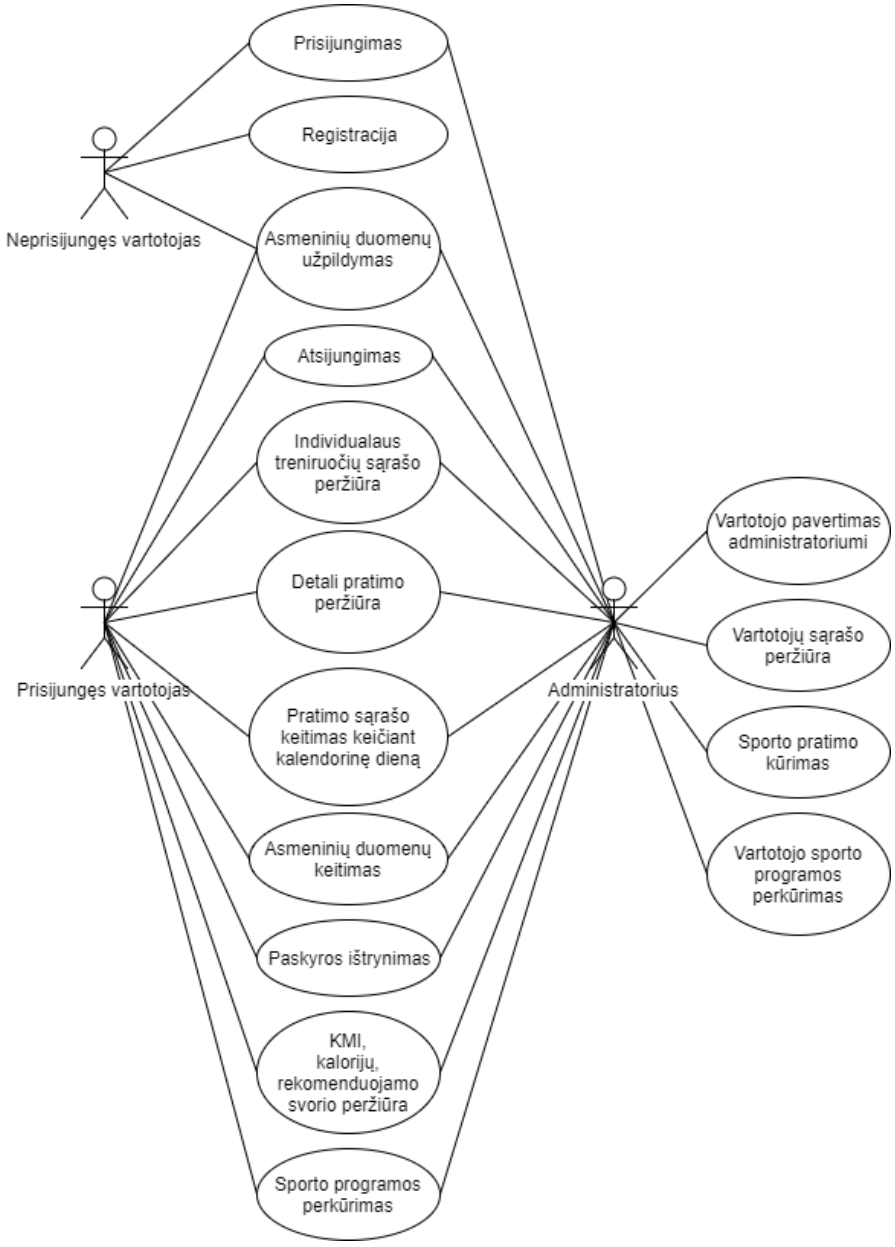
3 pav. Sugeneruotos sporto programos langas

3 pav. yra pavaizduotas naudotojui individualiai sugeneruojamos sporto programos langas. Šis langas turi tris svarbiausius elementus: datos keitimo juostą, pratimų atvaizdavimo slenkantį sąrašą ir navigacijos juostą, skirtą pereiti į profilio redagavimą. Paspaudus kiekvieną paveikslėlį, naudotojas yra nukreipiamas į sporto pratimo detalųjį langą, kuriame yra atvaizduojamas sporto pratimo pakartojimų skaičius, treniruojamas raumuo, pratimo pradėjimo ir baigimo formos paveikslėliai. Paspaudus profilio lango paveikslėlį, naudotojas yra nuvedamas į valdymo meniu, kuriame yra atvaizduojamas kūno masės indeksas, kiek reikia suvartoti kalorijų per dieną bei rekomenduojamas svoris pagal naudotojo ūgį. Šalia šių duomenų yra suteikiama galimybė iš naujo sugeneruoti sporto programą, keisti asmeninius duomenis, ištrinti savo paskyrą bei atsijungti nuo sistemos. Administratorius prisijungęs turi lygiai tuos pačius langus, tačiau naudotojo profilio lange papildomai yra atvaizduojama administratoriaus sekcija, kurioje yra valdymo meniu. Naudotojų valdymo lange galima bet kuriam naudotojui suteikti administratoriaus teises ar iš naujo sugeneruoti pasirinktam naudotojui jo sporto programą. Taip pat administratorius gali sukurti

sporto pratimą atsidaręs tam skirtą formą. Sporto pratimo kūrimo metu yra tikrinami visi įvedami laukai. Tikrinama ar pavadinimas nėra per trumpas ar per ilgas, tikrinama ar paveiksliukai yra įkeliami *https* formatu. Šis paveikslėlių formatas yra svarbus, nes *React-Native* biblioteka neleidžia atvaizduoti *http* paveikslėlių, kadangi tai yra nesaugus protokolas ir tuo pasinaudojus galima įsilaužti į programėlę.

3.4. Sistemos panaudos funkcijos

Naudotojas ir administratorius turi tą pačią sisteminę aplinką. Administratorius paveldi visas funkcijas iš naudotojo ir taip pat turi specialias tik jam skirtas funkcijas (žr. 4 pav.).



4 pav. Panaudos atvejų diagrama

Diagramoje pavaizduoti naudotojo ir administratoriaus atliekami funkcionalumai. Visi funkcionalumai papildo vienas kitą ir yra sukurti kaip eiga veiksmų, skirtų sukurti sporto programą automatiškai, ją atvaizduoti ir keisti. Taip pat yra papildomi funkcionalumai, skirti naudotojui keisti savo duomenis, ištrinti savo profilį ar inicijuoti sporto programos generavimą.

3.5. Pasirinktų technologijų analizė

Šiame poskyryje bus aiškinamos pasirinktos technologijos su pagrindimu kodėl jos buvo pasirinktos bei pagrindiniais jų veikimo principais.

3.5.1. *Google Firebase*

Tai yra realiuoju laiku atnaujinama duomenų bazė [Goo17]. Tai reiškia, kad bet kokie duomenų pakeitimai yra akimirksniu sinchronizuojami su duomenų baze. Sąryšis su šia duomenų bazė vyksta ne per *HTTP* protokolą, kaip daugelyje senesnių duomenų bazių, o per žiniatinklio jungtį (ang. *WebSocket*). Žiniatiklio jungtis leidžia klientui duomenis siųsti į serverį tokiu greičiu, kokį pajėgia tiekti kliento interneto tiekėjas.

Privalumai:

- Duomenys yra saugomi *JSON* formatu, todėl yra lengvai formuojami ir išformuojami.
- Nereikia mokėti *SQL* sakinių, nes serveris veikia kaip servisas ir pateikia funkcijas (ang. *No-SQL*), kuriomis naudojantis galima išsaugoti, gauti ir modifikuoti duomenis.
- Galimybė nustatyti individualiai kiekvienai lentelei ir jos reikšmėms taikomas tikrinimo taisykles norint nuskaityti ar nustatyti duomenis konkrečioje lentelėje ar reikšmėje.

Trūkumai:

- Duomenų indeksavimas turi būti kuriamas rankiniu būdu.
- Tikrinimo taisyklės nepalaiko sudėtingų objektų, reikia nustatyti tikrinimo taisykles kiekvienam vaikiniam objektui.

Naudotojų registracija yra paremta *Firebase Auth* moduliu, kurį suteikia *Google Firebase*. Šioje sistemoje yra integruota elektroninio pašto ir slaptažodžio registravimosi ir prisijungimo metodika. Kai naudotojas registracijos metu užpildo elektroninio pašto ir slaptažodžio laukus, pateikti duomenys yra siunčiami į *Firebase Auth* naudotojų lentelę, kuri yra generuojama pačio *Firebase* serviso. Lentelėje yra išsaugomas naudotojo objektas, kuris turi šiuos duomenis: unikalų identifikacinį numerį (ang. *Unique identifier*), elektroninį paštą, slaptažodį ir kitus programai

neaktualius duomenis. Naudotojo identifikacinis numeris yra generuojamas pagal keturias savybes:

1) Numeris yra paremtas registracijos data ir laiku, tam, kad būtų įmanomas filtravimas pagal egzistuojančius identifikacinius numerius. Data ir laikas yra konvertuojami į simbolius.
2) Numeris turi 72 bitus atsitiktinių simbolių, tam, kad nebūtų numerių kolizijos.
3) Numeris yra rūšiuojamas leksikografiškai pagal raidžių darinius, arba kitaip sakinius.
4) Numeriai yra monotoniškai didinami. Tai yra pasiekiama saugant panaudotus atsitiktinius bitus ir juos didinant po vieną.

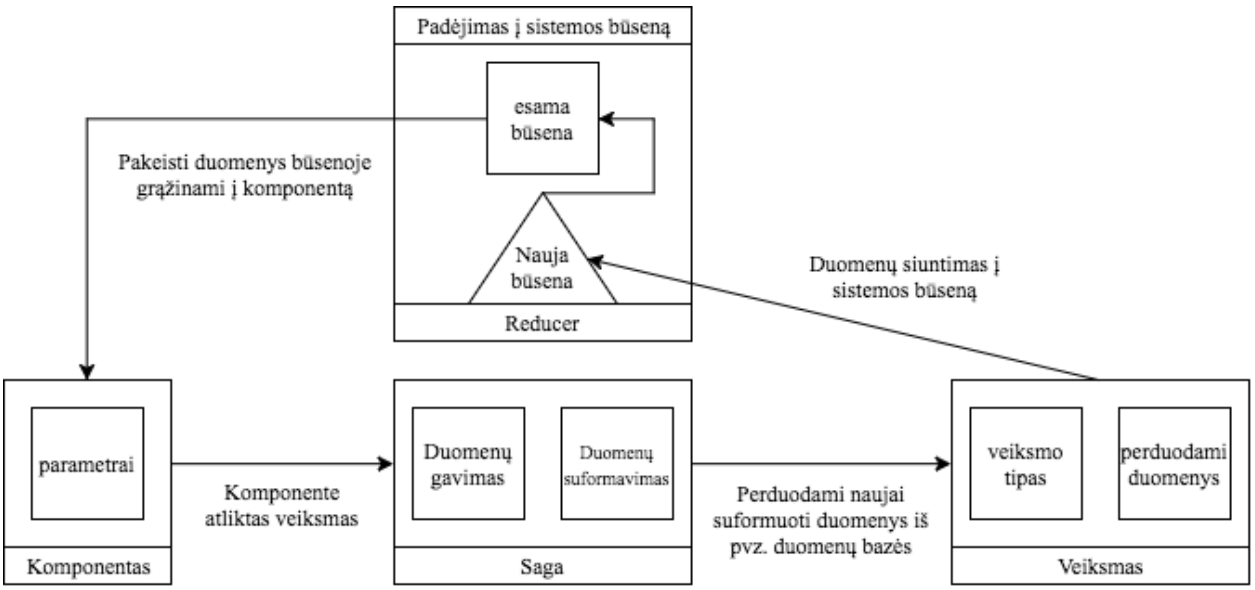
Naudotojo slaptažodis prieš išsaugojimą yra apsaugomas *Bcrypt* [PMS12] maišos būdu. *Bcrypt* yra adaptyvi maišos funkcija, paremta *Blowfish* [Sch93] simetrinio bloko šifro algoritmu, kuris leidžia funkcijai nuspręsti, kokio sudėtingumo bus maišos funkcija. Pagal sudėtingumo lygį yra nusprendžiama, kokia lėta bus maišos funkcija, o tai keičia funkcijos generuojamus maišos rezultatus pagal skirtingą laiko tarpą.

3.5.2. *React Native*

React yra *JavaScript* biblioteka, skirta kurti naudotojo sąsają [Rea17]. Ji yra kuriama ir palaikoma kompanijos *Facebook* bei individualių programuotojų ir korporacijų. *React Native* leidžia kurti mobilias sistemas, kuriose duomenys gali keistis realiuoju laiku be poreikio perkrauti užkrautą langą, o perkraunant tik dalį lango, kurioje pasikeitė duomenys. Mobili sistema susideda iš *JavaScript* variklio ir *React* blokų, kurie yra sukurti *Objective-C* ir *Java* programavimo kalbomis. *React Native* taip pat turi pagalbinių bibliotekų, skirtų navigacijai tarp sistemos langų, sistemos būsenos valdymui, komunikavimui su duomenų baze ir kt.

3.5.3. *Redux – Saga*

Redux-saga [Red17] yra JavaScript biblioteka, kuri yra skirta rašyti asinchronines funkcijas, kurios atrodo kaip sinchroninės funkcijos. Ši biblioteka yra skirta duomenų gavimui, formavimui ir saugojimui programos būsenoje, tokiu būdu duomenis turint bet kuriame programos lange. Ši biblioteka veikia principu, pavaizduotu 5 pav.



5 pav. *Flux* mobilios programos architektūros sprendimas

Flux architektūra turi tris pagrindines dalis, veiksmo siuntėją, programos būseną ir komponentus. Kai komponente atliekamas tam tikras veiksmas, pavyzdžiui, navigacija į kitą komponentą, būna siunčiamas veiksmas, kuris gali savyje nešti ir duomenų rinkinį. Pateiktų veiksmų yra klausomasi funkcijose *saga* arba *reducer*. Saga funkcija yra skirta duomenų gavimui, formavimui, funkcijoms ir duomenų siuntimui į reducer funkciją. *Reducer* funkcija yra skirta tik darbui su programos būsena. Ji keičia programos būseną naujais duomenimis, atėjusiais iš *saga* funkcijos arba tiesiai iš komponento. Pakeista programos būsena yra pasiekiama bet kuriame programos komponente, todėl, kai duomenys yra atnaujinami, komponentai gali būti atnaujinami atitinkamai, be reikalo neperkraunant viso komponento, o tik tam tikrą dalį, kuri gavo naujų duomenų. Taip yra pagreitinamas programos veikimas ir sumažėja kiekis duomenų, kuriuos naudotojui reikia siųstis.

4. Pagrindinio funkcionalumo įgyvendinimas

Sistemos funkcionalumas susideda iš daug mažų dalių, skirtų naudotojo individualios programos sudarymui. Šiame skyriuje bus aprašomas duomenų bazės sukūrimas, jos sąveika su sistema ir funkcijos, skirtos automatinei sporto programai sukurti bei naudotojui valdyti savo paskyrą.

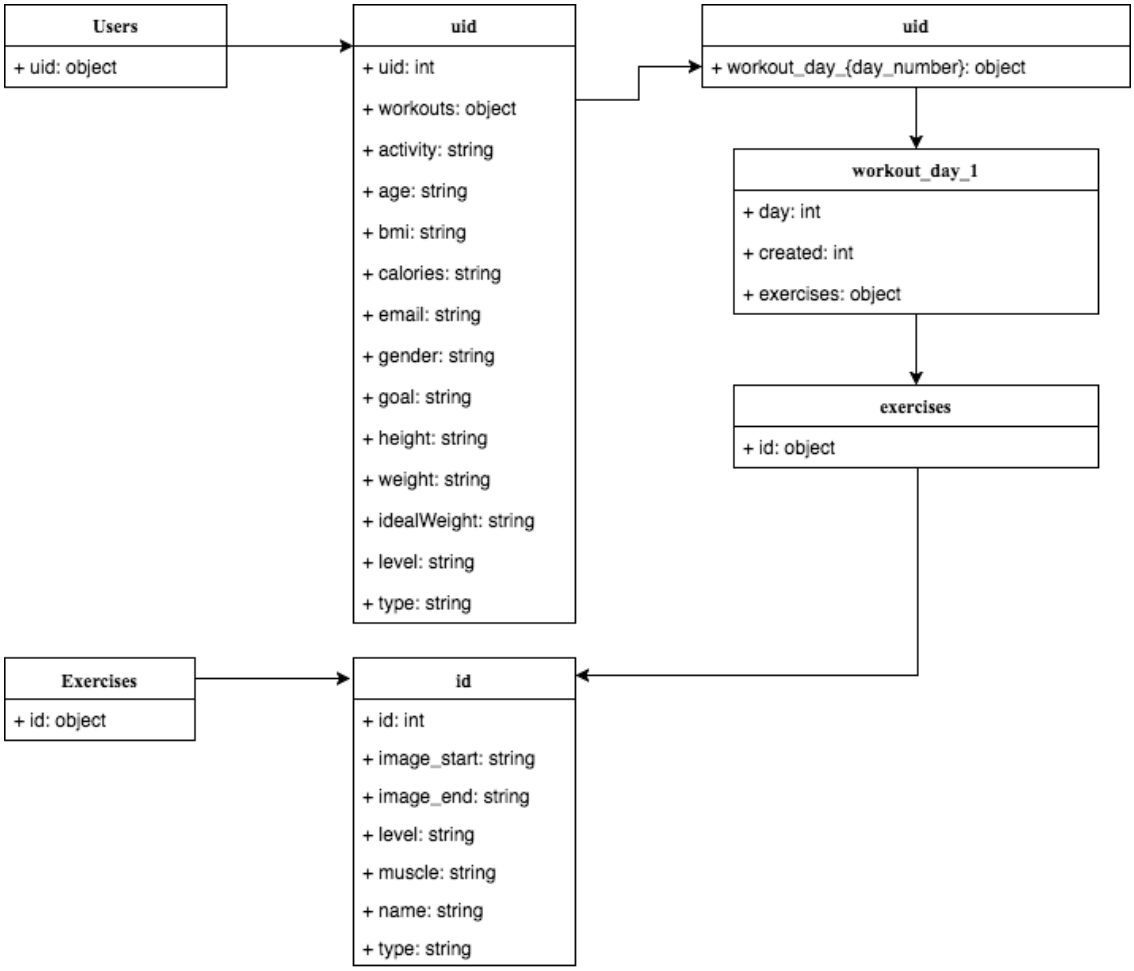
4.1. Duomenų bazė

Sistemos duomenų saugojimui buvo naudojama *Google Firebase* realiuoju laiku atsinaujinanti duomenų bazė. Ši duomenų bazė buvo lengvai integruota į sistemą, nes turi *JavaScript* biblioteką, kuri yra skirta atlikti funkcijas darbui su duomenų baze, tokiu būdu leidžianti nenaudoti tiesioginių *SQL* užklausų.

Duomenų bazėje yra saugomos dvi lentelės:

- *Exercises* lentelėje yra saugomi sporto pratimai, kurie yra skirti sporto programos sudarymui. Lentelėje yra saugomi pratimų tipai, kurie leidžia atskirti pratimus generavimo metu ir atskirti jų eilės tvarką, apšilimo pratimams būnant aukščiau nei jėgos pratimams. Taip pat yra saugoma, kurį specifinį raumenį naudotojas treniruos ir kokio sunkumo yra pratimas. Lentelėje taip pat yra išsaugomos nuorodos į paveikslėlius, kurios nurodo, kaip taisyklingai pradėti ir baigti sporto pratimą.
- *Users* lentelėje yra saugomi papildomi duomenys. Naudotojas turi tris skirtingas lentelės dalis:
    - Asmeniniai duomenys, kurie nusako fizinę būklę.
    - Naudotojui pagal jo asmeninius duomenis apskaičiuoti parametrai: kūno masės indeksas, kalorijų suvartojimas per dieną, rekomenduojamas svoris.
    - Sporto programa, kuri susideda iš tiek dienų, kiek buvo nuspręsta sporto programos generavimo metu. Kiekviena sporto programos diena turi sąrašą sporto pratimų, kurie yra pateikiami iš lentelės *exercises*.
- *Authentication* lentelėje yra saugomi naudotojų prisijungimo duomenys. Sistemos administratorius šioje lentelėje gali matyti tik naudotojo elektroninį paštą, kokiu būdu buvo užsiregistruota, kada paskutinį kartą buvo prisijungta bei unikalųjį identifikacinį numerį. Registracijos būdai yra keli, galima registruotis ne tik naudojant elektroninį paštą ir slaptažodį, bet naudojantis *Facebook, Google ar Github* sistemomis ar prisijungiant *Anonymous* tipu. Šie prisijungimo būdai programėlėje nėra naudojami. Naudotojo *uid* yra taip pat išsaugomas

lentelėje *users* registracijos metu, tam, kad prisijungimo metu būtų galima išgauti naudotojo asmeninius duomenis, sporto programą ir profilyje atvaizduojamus duomenis.



6 pav. Firebase duomenų bazės objektų schema

6 pav. pavaizduota duomenų bazės schema. Ši duomenų bazė nėra reliacinė, todėl neturi ryšių tarpusavyje. Vienintelis dalinis sąryšis yra toks, kad kai yra generuojama individuali sporto programa, iš duomenų bazės yra paimamas pratimų sąrašas ir pratimai po vieną dedami į *workouts* objektą. W*orkouts* objekte yra saugomas kiekvienos sudarytos sporto programos objektas, kuriame yra tai dienai reikiamų atlikti pratimų sąrašas.

4.2. Sistemos struktūra

React Native karkasu kuriamos sistemos turi specifinę failų struktūrą, tam, kad kompiliavimo metu būtų atpažintas pradžios taškas ir kompiliavimo proceso eiga:

- *App.js* – sistemos įeities taškas, nuo kurio prasideda sistemos darbas. Čia taip pat panaudojamas */navigation* aplanke aprašytas komponentas, kuris užkrauna pradinį langą ir sprendžia, kokį kelią reiktų pateikti, kai yra iškviečiamas vienas iš veiksmų, aprašytų aplanke */actions*.
- *package.json* – naudojamų bibliotekų versijavimas.
- */node_modules* – visos papildomos bibliotekos naudojamos sistemoje.
- */src* – JavaScript sisteminiai failai.
    - */screens* – sistemos naudotojų matomi langai.
    - */components* – languose naudojami komponentai, tokie, kaip specialus mygtukas, naudojamas keliose vietose ar antraštė, esanti kiekviename lange.
    - */navigation* – sistemos naršymo konfigūracija.
    - */state* – flux architektūros dalys.
        - */actions* – veiksmai, kurie gali būti kviečiami komponentuose ar saga funkcijose.
        - */config* – firebase serviso konfigūraciniai failai.
        - */reducers* – sistemos būsenos keitimo funkcijos, kurios suveikia automatiškai, nes priklauso nuo veiksmų, aprašytų aplanke: */actions*
        - */sagas* – funkcijos, skirtos darbui su duomenų baze ir duomenų perdavimu.

4.3. Prisijungimas ir registracija

Registracijos funkcionalumui buvo sukurtas komponentas, kuriame atvaizduojami trys laukai: *email, name, password*. Įvedus šiuos laukus yra kviečiama funkcija, kuri iš perduoto veiksmo nuskaito *email, name* ir *password* laukus ir su jais kviečia *Google Firebase* funkciją *createUserWithEmailAndPassword*, skirtą sukurti paskyrą su elektroniniu paštu ir adresu. Jeigu yra gaunamas atsakymas iš minėtos funkcijos, tuomet yra išsaugomas naudotojo unikalus identifikacinis numeris ir kviečiamas kitas metodas, skirtas išsaugoti naują naudotoją realiojo laiko duomenų bazės lentelėje - *users*. Į minėta metodą perduodamas tik identifikacinis numeris, kadangi kitus duomenis saugo *authentication* lentelė. Kai naudotojas yra sėkmingai išsaugotas visose lentelėse, tuomet yra kviečiamas lango pakeitimo veiksmas ir atidaromas papildomų duomenų įvedimo langas. Registracijos veiksmo metu yra rodomas laukimo ženklas, ir, jeigu

registracija yra nesėkminga, parodoma žinutė su klaidos kodu. Jei registracija visgi yra sėkminga, tai naudotojas yra nukreipiamas į papildomų duomenų registracijos langą. Šiame lange užpildžius savo duomenis ir paspaudus mygtuką tęsti, inicijuojami trys skaičiavimo metodai, kurie randa kūno masės indeksą, rekomenduojamą svorį ir kiekį kalorijų, kurį naudotojas turėtų suvartoti per dieną [Del06].

- Kūno masės indeksas (toliau – KMI) yra skaičiuojamas pagal šią formulę:

$$KMI = \frac{mass(\text{kg})}{height^2(m)}$$

KMI apskaičiuojamas papildomos duomenų registracijos metu įvedus ūgį ir svorį. Svoris lieka nepakitęs, o ūgis yra konvertuojamas iš centimetrų į metrus. Gautasis kūno masės indeksas yra prilyginamas KMI išvadų lentelei [HF18] (žr. 7 pav.). Pagal gautą KMI yra nustatoma ar žmogaus masė yra normali, ar yra antsvoris bei nutukimas, ar svorio trūkumas.

Kūno masės indeksas	Išvada
16 ir mažiau	Ryškus kūno masės trūkumas
16—18	Nepakankama kūno masė
18—25	Norma
25—30	Kūno masės perteklius
30—35	Pirmo lygio nutukimas
35—40	Antro lygio nutukimas
40 ir daugiau	Trečio lygio nutukimas

7 pav. KMI išvadų lentelė

- Rekomenduojamas svoris yra skaičiuojamas pagal šią formulę formulę:

$$Vyrams: I(k_g) = (Height(\text{cm}) - 100) - ((Height(cm) - 100) * 10\%)$$

$$Moterims: I(k_g) = (Height(\text{cm}) - 100) - ((Height(cm) - 100) * 15\%)$$

Ši formulė skiriasi vyrams ir moterims, nes moterų ūgis paprastai būna žemesnis. Ši formulė skaičiuoja idealų svorį su galima 5 kg paklaida ir tai yra rekomenduojamas svoris, kurį įmanoma pasiekti tokio ūgio žmogui, koks yra pateikiamas formulėje.

- Kalorijų kiekis per dieną skaičiuojamas pradžioje randant bazinę medžiagų apykaitos (toliau BMR) normą pagal šią formulę:

$$BMR = (10 \times w) + (6.25 \times h) - (5 \times a) \pm s$$

Čia *w* yra svoris kilogramais, *h* – ūgis centimetrais, *a* – amžius metais, o *s* – kintamasis parametras (vyrams + 5, o moterims – 161). Radus BMI, yra pritaikomas Harris-Bennedict principas, kuris pagal žmogaus aktyvumą išveda skirtingą koeficientą, skirtą rasti vartotiną kalorijų kiekį per dieną. Šis principas teigia, kad nesportuojant, koeficientas turi būti 1.2, sportuojant vieną – tris kartus per savaitę, koeficientas turi būti 1.375, antliekant tris – penkias treniruotes per savaitę, koeficientas turi būti 1.55, o atliekant virš šešių treniruočių per savaitę koeficientas turi būti 1.725. Sudauginus BMI su koeficientu yra gaunamas kasdienis reikalingas kalorijų suvartojimo kiekis.

Išvedus visas tris formules, žmogaus duomenys ir formulių rezultatai yra pridedami prie bendro objekto ir siunčiami į *users* lentelę naudojantis *fire.database().update* metodu ir nustatant objektą į užsiregistravusio žmogaus unikalųjį numerį.

Prisijungimo funkcionalumui sukurta forma, kurioje naudotojas turi įvesti savo elektroninio pašto adresą ir slaptažodį. Šie du parametrai yra siunčiami į saga funkciją *loginWithEmail*, kur naudojantis firebase metodu *fire.auth().signInWithEmailAndPassword({email}, {password}),* naudotojas yra prijungiamas prie sistemos, arba yra gaunama duomenų neatitikimo klaida. Jei yra gaunama klaida, saugantis nuo įsilaužėlių, nėra pasakoma, kas įvesta neteisingai, bet grąžinama žinutė, kurioje teigiama, kad kažkurie įvesti duomenys nėra teisingi. Jeigu naudotojas buvo sėkmingai prijungtas prie sistemos, yra kviečiamas metodas, kuris bando surasti, ar toks naudotojas yra įvedęs visus asmeninius duomenis ir jeigu visi duomenys nėra randami, yra pereinama į papildomų duomenų registracijos langą. Jeigu visi duomenys yra rasti, kviečiamas veiksmas, kuris veda naudotoją į namų langą, kuriame yra atvaizduojama sporto programa esamai dienai.

Anoniminis prisijungimas yra pateikiamas kaip galimybė išbandyti sistemą. Šis prisijungimo būdas yra ypatingas tuo, kad yra nereikalaujama sukurti paskyrą, tačiau vis tiek reikia įvesti asmeninius duomenis, pagal kuriuos turi būti generuojama sporto programa. Toks vartotojas skiriasi nuo paprasto vartotojo tik tiek, kad neturi galimybės ištrinti savo paskyros, kadangi ji yra sunaikinama automatiškai po paskyros atsijungimo nuo aktyvios sesijos.

Priedas nr. 3 pavaizduotas prisijungimo langas. Priedas nr. 4 pavaizduotas registracijos langas. Priedas nr. 5 pavaizduotas papildomų duomenų įvedimo langas.

4.4. Pagrindinio funkcionalumo įgyvendinimas

Sporto programos generavimas yra šios sistemos tikslas, tad buvo sukurta nesudėtinga forma, kurią užpildęs naudotojas tą pačią akimirką gali eiti į sporto salę ir atlikti nurodytus pratimus. Sistema yra paprasta naudotis ir lengva pasiekti, nes daugelis naudotojų šiais laikais visą laiką su savimi turi mobilųjį įrenginį. Sporto programos generavimas yra pagrįstas teorinėmis žiniomis iš [Sch12].

4.4.1. Individualios sporto programos automatinis kūrimas

Kai papildomi duomenys yra įrašyti į duomenų bazę, pradedamas individualios sporto programos kūrimo procesas. Pirmiausia yra kviečiamas metodas *getWorkoutMuscleGroups,* kuris pagal naudotojo įvestus duomenis nusprendžia, kiek dienų turi trukti sporto programa, ir kokias raumenų grupes reikia treniruoti tomis dienomis (žr. 1 išeities kodas.).

1 išeities kodas. getWorkoutMuscleGroups programinio kodo ištrauka

```
1.     export function getWorkoutMuscleGroups(details) {
2.         const {
3.             weight, height, goal, level, gender, age
4.         } = details;
5.         if (goal === 'gain') {
6.             if (level === 'beginner') {
7.                 if (age >= 16 && age <= 50) {
8.                     return ['chest/shoulders', 'cardio/legs', 'back/arms'];

9.                 }
10.                else {
11.                    return ['legs/back/chest', 'cardio/arms/shoulders'];
12.                }
13.            }
14.            else {
15.                if (age >= 16 && age <= 50) {
16.                    return ['chest/arms', 'cardio/legs', 'back/arms', 'cardio/shoulders
'];
17.                }
```

Toliau yra kviečiamas metodas, *getWorkoutDays* kuris yra skirtas išdėstyti raumenų grupių treniravimą per tam tikrą dienų tarpą, kuris yra geriausiai pritaikytas tinkamai pailsėti nuo treniruočių. Toliau yra kviečiama duomenų bazė kviečiant firebase metodą *fire.database.read('users')*. Šis metodas grąžiną visą sąrašą sporto pratimų, iš kurių bus renkama sporto programa.

Taigi sudėjus gautus duomenis: *treniruotini_raumenys, treniruočių dienos, sporto_pratimai, naudotojo_duomenys,* yra kviečiamas metodas *createWorkoutByDay* (žr. Priedas nr. 1). Šis metodas suka ciklą per reikiamų treneruoti raumenų grupę, iš kurios išima po vieną raumenį ir

ieško, ar raumuo yra iš duomenų bazės gautame sporto pratimų sąraše. Jeigu pratimas yra randamas, jis yra pridedamas į laikiną sąrašą. Rezultate yra gaunamas kiekvienai raumenų grupei sugeneruotas pratimų sąrašas. Toks sąrašas yra paduodamas į kitą funkciją *calculateWorkoutSize*, kuri yra skirta apskaičiuoti, kiek pratimų iš atrinkto sąrašo, reikia atlikti per dieną (žr. Priedas nr. 2). Šioje funkcijoje aprašytos kelios svarbios konstantos: *exercisesPerWorkout* – kiek pratimų turėtų būti vienos sesijos metus, *gain/loss* – vidutinis pratimo pakartojimų skaičius. Tuomet yra vykdomas ciklas, kuris yra sukamas tiek kartų, kiek pratimų turi būti per treniruotę ir yra gaunamas vienas iš pratimų, kuris pilnai atitinka naudotojo fizinę būklę iš susiaurinto visų pratimų sąrašo. Jei pratimas yra *cardio* tipo jis yra dedamas į treniruotės priekį, nes tai yra skaitoma kaip apšilimo pratimas, jam taip pat yra priskiriamas pratimo trukmės parametras, nes pakartojimų skaičiumi šio pratimo matuoti negalima. Toliau, gavus iki dešimties ar mažiau pratimų sutrumpintą sporto sesijos sąrašą, jis yra grąžinamas į *createWorkoutByDay* metodą. Kuriame yra vėlgi sukamas ciklas per sąrašą, nes jame yra pagal dienas suskirstyti pratimai. Kiekvienai sesijai yra priskiriama diena iš dienos sąrašo gauto anksčiau ir sudaromas naujas sąrašas, kuriame yra sporto pratimai, programos identifikacinis numeris, sudarymo data ir savaitės diena, kurioje bus sportuojama ši programa. Tuomet toks sąrašas yra išsaugojamas duomenų bazėje išsiunčiant jį tam naudotojui, kuris šiuo metu yra prisijungęs prie sistemos. Išsiuntimas vyksta gaunant aktyvų naudotoją metodu *fire.auth().currentUser*, šis metodas grąžina unikalų numerį į kurį ir yra siunčiama sporto programa.

4.4.2. Sporto programos atvaizdavimas

Prisijungusį naudotoją pasitinka jo automatiškai sugeneruota sporto programa, kalendorius ir galimybė pereiti į profilio langą (žr. Priedas nr. 7). Sporto programa yra gaunama kviečiant *fire.database().read(„users/“ + uid)* metodą, kuriam davus prisijungusio žmogaus unikalų numerį yra grąžinamas žmogaus duomenų sąrašas. Gautasis sąrašas yra padedamas į sistemos būsėną ir taip pat atnaujinamas pratimų sąrašas namų komponente. Namų komponentas iš visų dienų sporto sąrašų rodo būtent tą, kuris atitinka esamą dieną arba tą, kuri būna pasirinkta kalendoriuje. Sporto pratimų sąrašas yra atvaizduojamas React Native komponentu *scrollView*, šis komponentas leidžia sąrašo elementus sutalpinti slenkančiame sąraše, tad elementų perteklius yra paslepiamas ir leidžiama ekraną stumti žemyn atskleidžiant nematomus sąrašo elementus. Paspaudus ant vieno iš elementų yra parodomas detalusis sporto pratimo langas. Šiame lange yra rodomas pratimo pakartojimų skaičius, treniruojamas raumuo ir pratimo pradžios ir pabaigos paveikslėliai. Kokį svorį turėtų naudoti žmogus, atlikdamas sporto pratimus, nėra nurodoma, todėl galima paspausti ant nuorodos *weight*, kuris nurodo, kad turi būti naudojamas svoris, kuris išvargintų vartotoją

pakankamai, per nurodytą serijų ir pakartojimų skaičių. Paveikslėliai ir pratimų informacija yra imama iš [Bod17] šaltinio.

4.4.3. Profilis

Profilio langas yra svarbi sistemos dalis, nes čia yra keli būtini funkcionalumai:

- Sporto programos perkūrimas yra metodas, kurį kviečiant yra perduodami žmogaus asmeniai duomenys ir yra iš naujo inicijuojamas *createWorkoutByDay* metodas. Šis metodas grąžina skirtingus pratimus, nes iš sutraukto pratimų sąrašo yra atsitiktiniu būdu atrenkami kiti pratimai nei buvę praeitame sporto pratimų sąraše. Kai kurie pratimai gali nesikeisti, dėl gana nedidelio sporto pratimų kiekio duomenų bazėje.
- Duomenų keitimas perduoda visus naudotojo duomenis į tą patį langą, kuris yra naudojamas papildomų duomenų registracijos metu, tačiau jau įvesti laukai yra užkraunami iš anksto, tad tereikia įvesti naujas reikšmes nevedant žinomų reikšmių iš naujo.
- Paskyros ištrynimas kviečia metodą, kuris ištrina naudotoją tiek *users* lentelėje tiek *authentication* lentelėje ir perveda naudotoją į pradinį sistemos langą, tuo pat metu jį atjungdamas iš aktyvios sesijos.

Kartu su paminėtais funkcionalumais yra pateikiami KMI, rekomenduojamas svoris bei rekomenduotinas suvartoti kalorijų kiekis per dieną.

4.4.4. Administratorių funkcijos

Administratoriaus funkcijos yra randamos profilio lange žemiau įprasto naudotojo funkcijų (žr. Priedas nr. 7). Šios funkcijos yra gan smarkiai limituotos, nes *Google Firebase* nėra integravusi daugėlio metodų, kaip pasirinkto naudotojo ištrynimas, ar duomenų keitimas. Norint atlikti šiuos funkcionalumus, būtina prisijungti prie paskyros, o tai padarius būtų nutraukiama administratoriaus sesija. Todėl yra pateikti trys pradiniai pasirinkimai:

- Sporto pratimo kūrimas yra paprasta forma, kurioje reikia suvesti sporto pratimo duomenis (žr. Priedas nr. 8). Suvedus reikiamus duomenis yra kviečiamas metodas, kuris gauna paskutinio sporto pratimo indeksą ir jį padidinus vienu skaičiumi siunčia naują sporto pratimo objektą su padidintu indeksu į *exercises* duomenų bazės lentelę.
- Naudotojų tvarkymo lange yra pateikiamas užregistruotų naudotojų sąrašas su galimybe ant kiekvieno iš jų paspausti ir atidaryti papildomų pasirinkimų sąrašą. Papildomų pasirinkimų sąraše yra galimybė padaryti pasirinktą naudotoją administratoriumi arba atnaujinti sporto

programą, ją perkuriant tuo pačiu metodu, kaip naudotojas tai padarytų pats, per savo profilio funkciją.

- Kiekvieno naudotojo redagavimas, bet kokių duomenų keitimas ar trynimas gali būti atlikti prisijungus prie *Google Firebase* duomenų bazės valdymo panelės, todėl šios funkcijos nebuvo įtrauktos į sistemos administratoriaus langą.

Išvados ir rekomendacijos

Šio darbo metu buvo aprašyta ir sukurta automatizuota individualios sporto programos kūrimo sistema. Pagrindinis tikslas buvo sukurti sistemą žmogui, kuris nesupranta, kaip reikia pradėti sportuoti. Pasitelkiant asmeniniais, naudotojo pateiktais duomenimis, jam sugeneruojama sporto programa, parodoma, kaip atlikti kiekvieną pratimą. Sistema buvo sukurta mobiliems įrenginiams su iOS ir Android operacinėmis sistemomis. Darbo metu buvo išanalizuotos panašios sistemos bei panašus mokslinis darbas, jų funkciniai ir dizaino sprendimai. Remiantis padaryta analize buvo sukurta kuriamos sistemos architektūra. Kuriant sistemos architektūrą buvo aprašyta panaudos atvejų diagrama, pagrindiniai reikalavimai bei funkcijos. Taip pat buvo aprašytos pasirinktos technologijos, paaiškintas jų veikimas bei panaudojimas sistemoje.

Siekiant tobulinti esamą sistemą reiktų sukurti šiuos papildomus funkcionalumus:

1. Galimybę naudotojui atstatyti pamirštą slaptažodį ar pakeisti esamą.
2. Sporto pratimų įvykdymo sekimą ir pažymėjimą kalendoriuje.
3. Sporto programos keitimą po tam tikro laikotarpio.
4. Dietos plano sudarymą.
5. Pridėti galimybę naudotojui nurodyti turimas traumas ir pagal tai atmesti tam tikrus pratimus.

Literatūros sąrašas

[Pov17] Povilaitis, Žygimantas. „Sporto treniruočių programos parinkimo sistema: bakalauro darbas“, Vilnius: Vilniaus universitetas, 2017. Prieiga per „eLABa“ – nacionalinė Lietuvos akademinė elektroninė biblioteka.
URL: http://talpykla.elaba.lt/elaba-fedora/objects/elaba:23218132/datastreams/MAIN/content

[PMS12] Provos, Niels; Mazières, David; Sutton, Jason 2012 (1999). „A Future-Adaptable Password Scheme“. Proceedings of 1999 USENIX Annual Technical Conference: 81–92.

[Sch93] Schneier, Bruce. „Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)“, 1993. Fast Software Encryption, Cambridge Security Workshop Proceedings. Pages: 191–204. URL: https://www.schneier.com/academic/archives/1994/09/description_of_a_new.html

[Del06] Dellova, et al. „ABC’s of Nutrition and Diet Therapy“. 2006 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1091498/

[Sch12] Schwarzenegger, Arnold. „The New Encyclopedia Of Modern Bodybuilding“, 2012

[Bod17] Sporto pratimai ir pratimų nuotraukos. URL: https://www.bodybuilding.com/ Žiūrėta: 2018-01-07

[Rea17] React Native dokumentacija. URL: https://facebook.github.io/react-native/ Žiūrėta: 2018-01-07

[Red17] Redux-saga dokumentacija. URL: https://redux-saga.js.org/ Žiūrėta: 2018-01-07

[Goo17] Google Firebase dokumentacija. URL: https://firebase.google.com/docs/ Žiūrėta: 2018-01-07

[HF18] Harris, Arthur;Francis G. Benedict. „A Biometric Study of Human Basal Metabolism“. Proc Natl Acad Sci USA. 1918.
URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1091498/

[BL14] Borowska, Paula; Laurinavičius, Tomas. „Mobile Design Book“ , 2014.
URL: https://www.amazon.com/Mobile-Design-Book-Paula-Borowska-ebook/dp/B00MNWKM1A

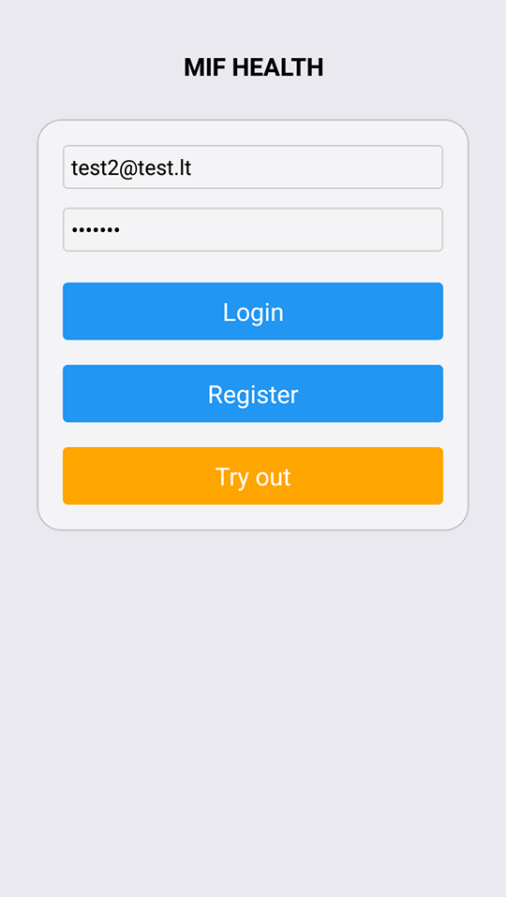
Priedas nr. 1 createWorkoutByDay algoritmas

```
1.      export function createWorkoutByDay(details, muscles, days, exercises) {
2.          const muscleGroups = [{
3.              chest: 'chest/abs'
4.          }, {
5.              back: 'traps/middle_back/lats/lower_back'
6.          }, {
7.              arms: 'biceps/forearms/triceps'
8.          }, {
9.              shoulders: 'shoulders'
10.         }, {
11.             legs: 'quads/hamstrings/glutes/calves'
12.         }, {
13.             cardio: 'cardio'
14.         }];
15.         for (let i = 0; i < days.length; i++) {
16.             const muscleGroupToTrain = muscles[i].split('/');
17.             const workoutByDay = [];
18.             muscleGroupToTrain.forEach(trainGroup => {
19.                 muscleGroups.map(muscleGroup => {
20.                     for (const group in muscleGroup) {
21.                         if (trainGroup === group) {
22.                             for (const name in muscleGroup) {
23.                                 if (name === group) {
24.                                     const musclesToFind = muscleGroup[name].split('/');
25.                                     musclesToFind.forEach(muscle => {
26.                                         const exercise = findExercise(muscle, details, exer
cises);
27.                                         if (exercise.length > 0) {
28.                                             exercise['muscle_type'] = exercise[0].muscle.to
LowerCase();
29.                                             workoutByDay.push(exercise);
30.                                         }
31.                                     });
32.                                 }
33.                             }
34.                         }
35.                     }
36.                 });
37.             });
38.             const workout = calculateWorkoutSize(workoutByDay, details);
39.             if (workout.length > 0) {
40.                 const day = getDayNumber(days[i]);
41.                 const workout_id = `workout_day_${day}`;
42.                 const todaysDate = moment().format('YYYY-M-D');
43.                 firebase.database().ref('users/' + details.uid + '/workouts/').child(workou
t_id).set({
44.                     day: day,
45.                     created: todaysDate,
46.                     exercises: workout
47.                 });
48.             }
49.         }
50.         return true;
51.     }
```

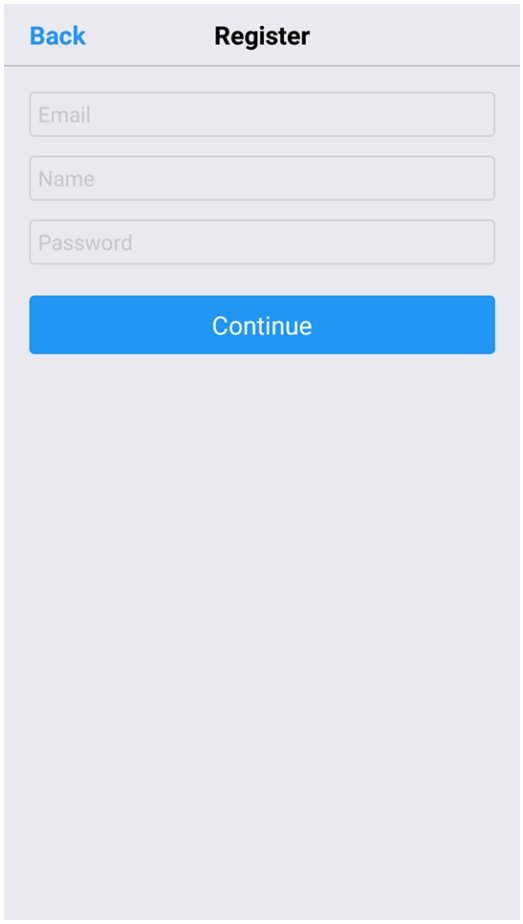
Priedas nr. 2 calculateWorkoutSize algoritmas

```
1.      export function calculateWorkoutSize(workout, details) {
2.          const exercisesPerWorkout = Math.floor(10 / workout.length);
3.          const result = [];
4.          const gain = {
5.              beginner: {
6.                  sets: 2,
7.                  reps: 10
8.              },
9.              intermediate: {
10.                 sets: 2,
11.                 reps: 20
12.             }
13.         };
14.         const loss = {
15.             beginner: {
16.                 sets: 2,
17.                 reps: 8
18.             },
19.             intermediate: {
20.                 sets: 3,
21.                 reps: 8
22.             }
23.         };
24.         workout.map(w => {
25.             if (w[0].type === 'Cardio') {
26.                 const exercise = w.getExercise();
27.                 if (exercise) {
28.                     if (details.goal === 'gain') {
29.                         exercise['duration'] = '10min.';
30.                     } else {
31.                         exercise['duration'] = '20min.';
32.                     }
33.                     removeExercise(w, exercise);
34.                     result.push(exercise);
35.                 }
36.             } else {
37.                 for (let i = 0; i < exercisesPerWorkout; i++) {
38.                     const exercise = w.getExercise();
39.                     if (exercise) {
40.                         if (details.level.toLowerCase() === 'beginner') {
41.                             if (details.goal === 'gain') {
42.                                 exercise['sets_x_reps'] = gain.beginner.sets + 'x' + gain.b
eginner.reps;
43.                             } else {
44.                                 exercise['sets_x_reps'] = loss.beginner.sets + 'x' + loss.b
eginner.reps;
45.                             }
46.                         } else {
47.                             if (details.goal === 'gain') {
48.                                 exercise['sets_x_reps'] = gain.intermediate.sets + 'x' + ga
in.intermediate.reps;
49.                             } else {
50.                                 exercise['sets_x_reps'] = loss.intermediate.sets + 'x' + lo
ss.intermediate.reps;
51.                             }
52.                         }
53.                         removeExercise(w, exercise);
54.                         result.push(exercise);
55.                     }
56.                 }
57.             }
58.         });
59.         return result;
```

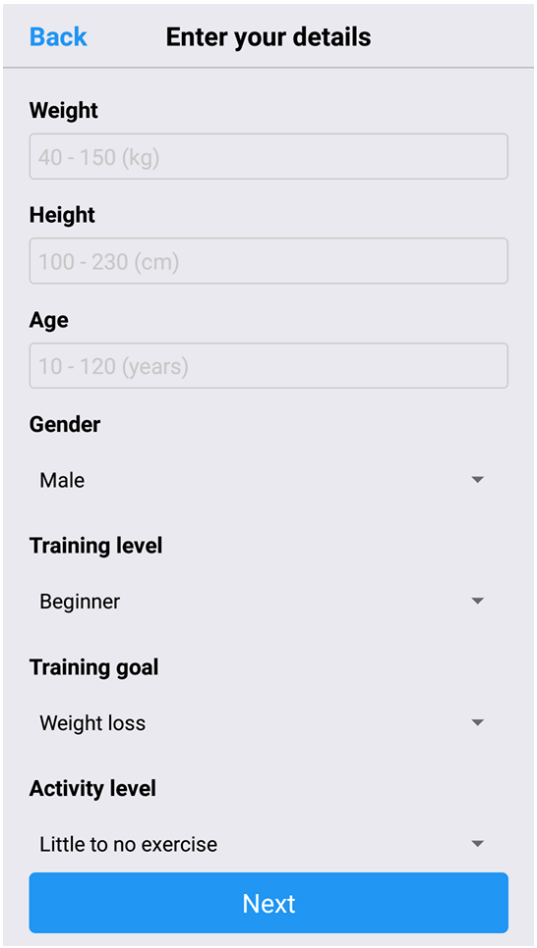
Priedas nr. 3 pradinis sistemos langas



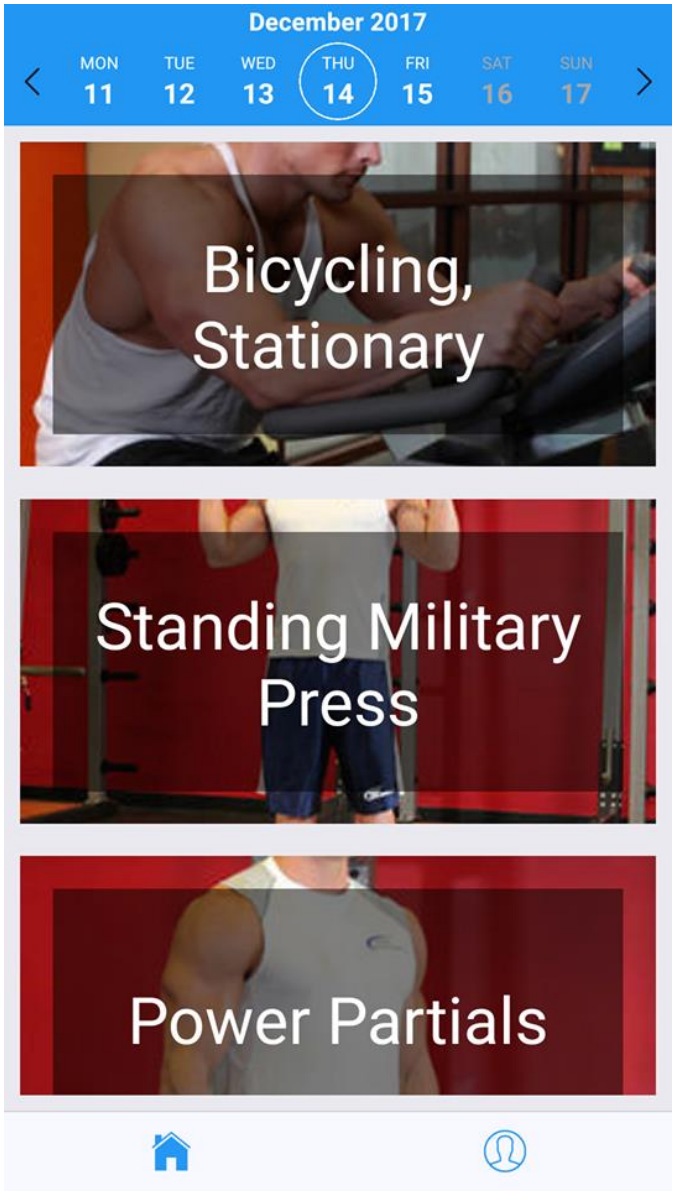
Priedas nr. 4 registracijos langas



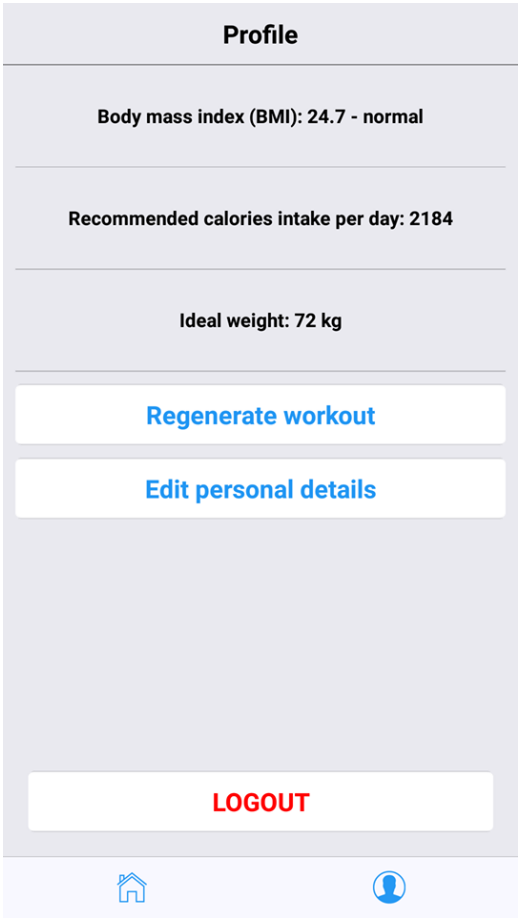
Priedas nr. 5 papildomų duomenų įvedimo langas

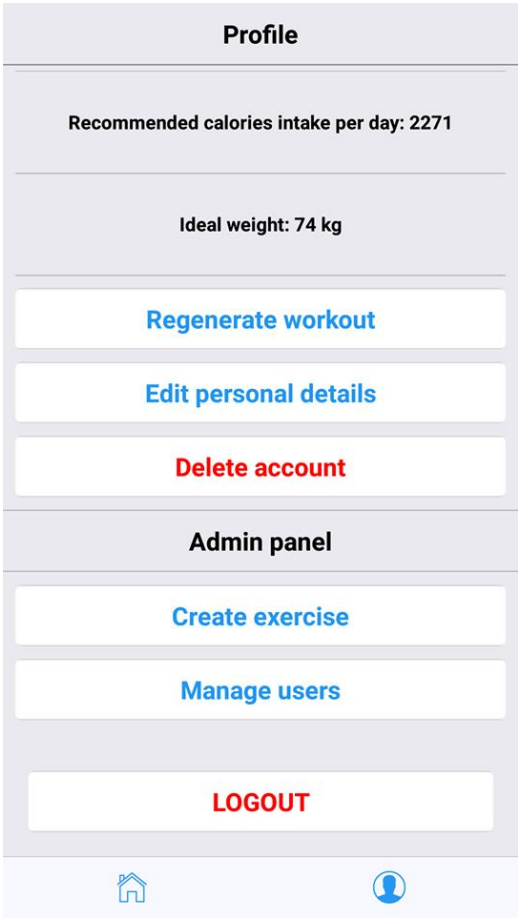


Priedas nr. 6 prisijungusio naudotojo langas

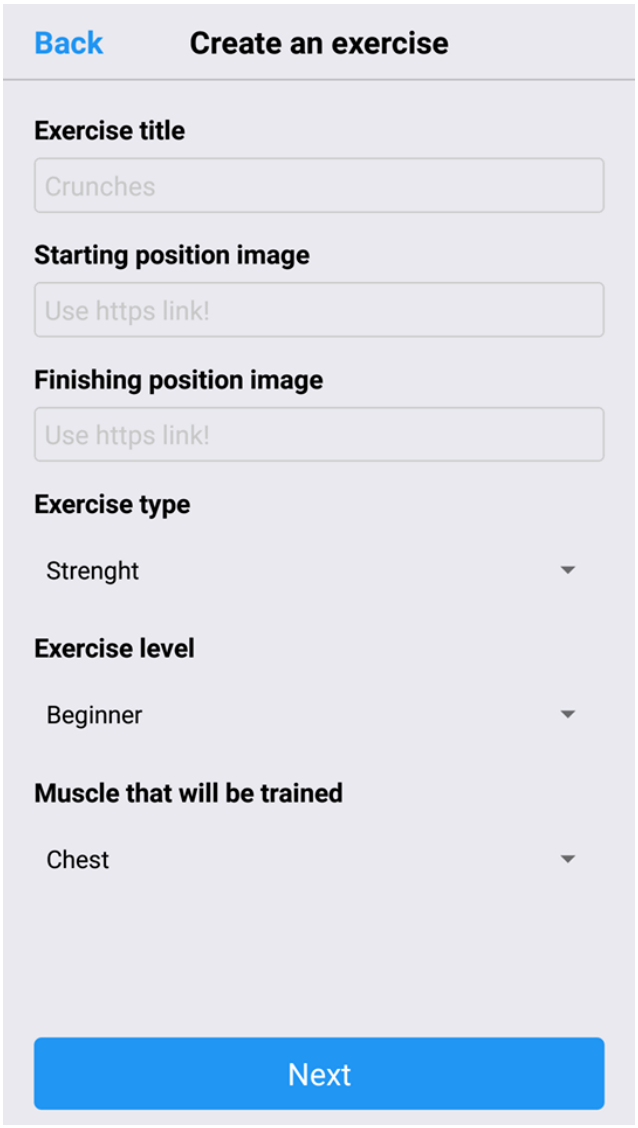


Priedas nr. 7 naudotojo profilio langas

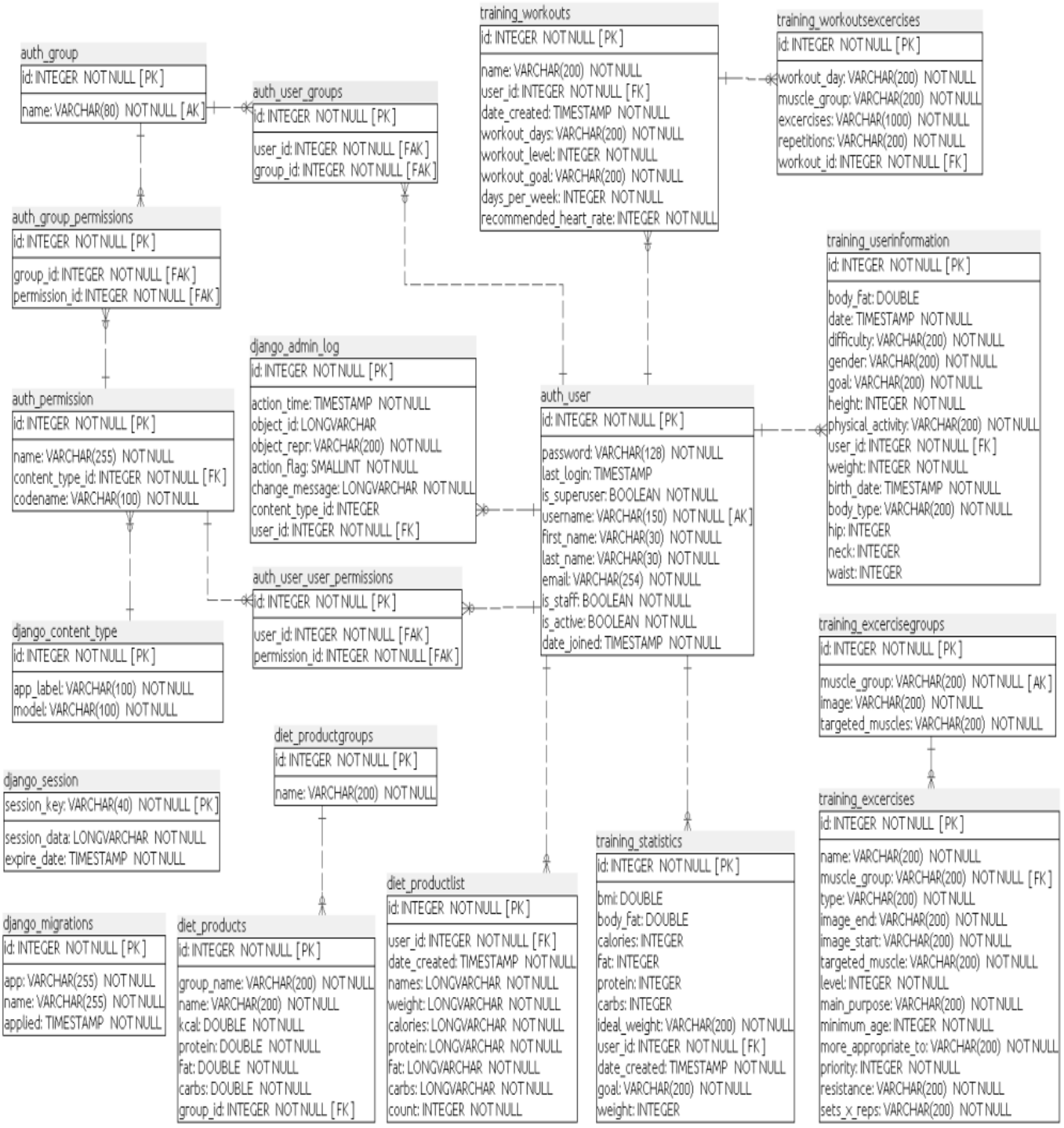




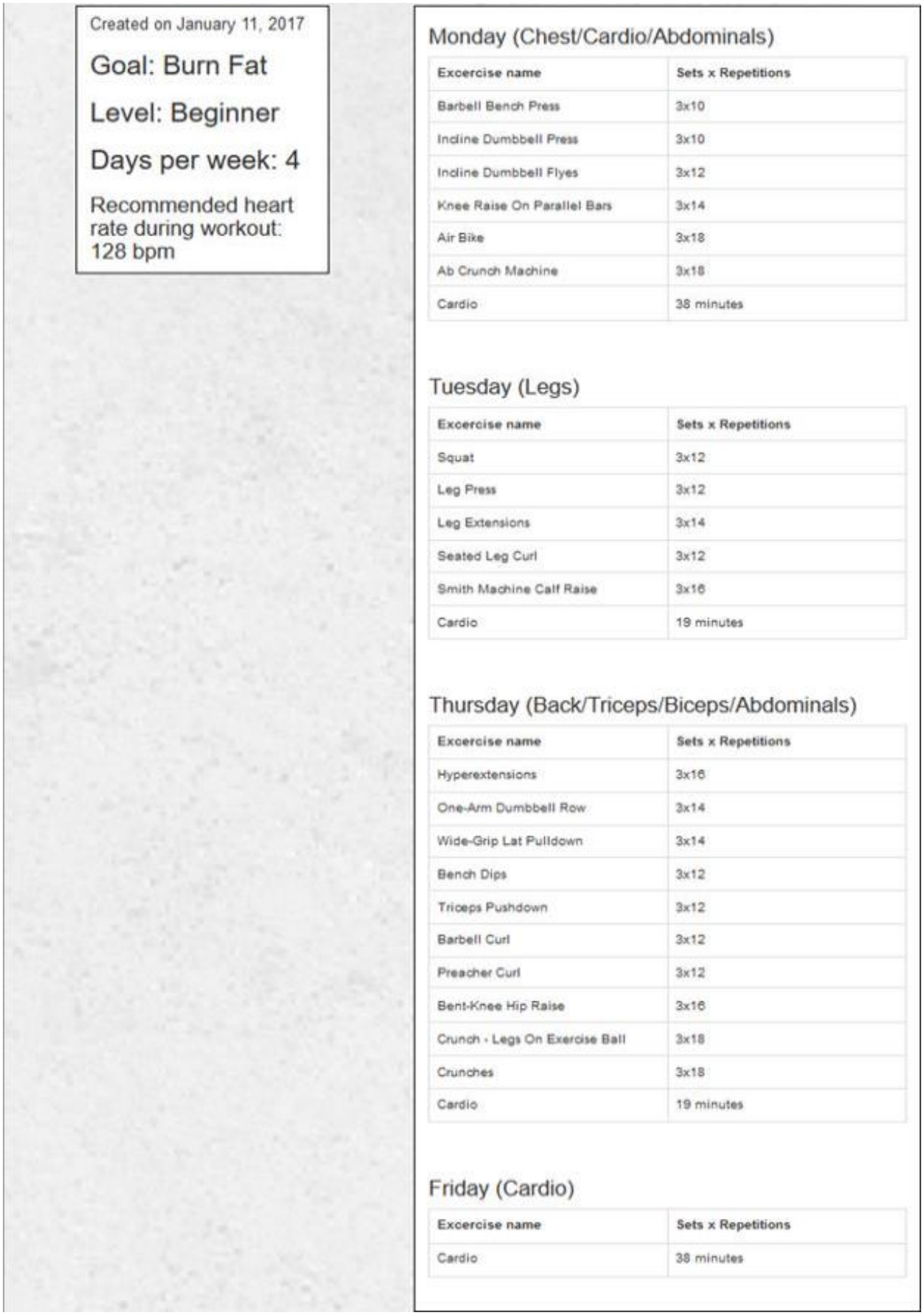
Priedas nr. 8 sporto pratimo kūrimo langas



Priedas nr. 9 panašios sistemos duomenų bazė



Priedas nr. 10 panašios sistemos naudotojo sąsaja



Created on January 11, 2017

Goal: Burn Fat

Level: Beginner

Days per week: 4

Recommended heart rate during workout: 128 bpm

Monday (Chest/Cardio/Abdominals)

Excercise name	Sets x Repetitions
Barbell Bench Press	3x10
Incline Dumbbell Press	3x10
Incline Dumbbell Flyes	3x12
Knee Raise On Parallel Bars	3x14
Air Bike	3x18
Ab Crunch Machine	3x18
Cardio	38 minutes

Tuesday (Legs)

Excercise name	Sets x Repetitions
Squat	3x12
Leg Press	3x12
Leg Extensions	3x14
Seated Leg Curl	3x12
Smith Machine Calf Raise	3x16
Cardio	19 minutes

Thursday (Back/Triceps/Biceps/Abdominals)

Excercise name	Sets x Repetitions
Hyperextensions	3x16
One-Arm Dumbbell Row	3x14
Wide-Grip Lat Pulldown	3x14
Bench Dips	3x12
Triceps Pushdown	3x12
Barbell Curl	3x12
Preacher Curl	3x12
Bent-Knee Hip Raise	3x16
Crunch - Legs On Exercise Ball	3x18
Crunches	3x18
Cardio	19 minutes

Friday (Cardio)

Excercise name	Sets x Repetitions
Cardio	38 minutes